ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ
Государственное образовательное учреждение
высшего профессионального образования
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ

А. Г. Варжапетян

ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ НА GPSS/H

Учебное пособие

Санкт-Петербург
2007

УДК 519.682
ББК 22.18
В18

Рецензенты:

кафедра морских информационных технологий Российского государственного гидрометеорологического университета;
доктор технических наук, профессор кафедры вычислительных систем и информатики Государственного университета водных коммуникаций
В. В. Фомин

Утверждено редакционно-издательским советом университета в качестве учебного пособия

Варжапетян А. Г.

В18 Имитационное моделирование на GPSS/H: учебное пособие / А. Г. Варжапетян; ГУАП. — СПб., 2007. — 384 с.: ил.
ISBN 5-8088-0228-8

В учебном пособии рассматриваются вопросы построения, логики действия и начал программирования на современной версии языка имитационного моделирования GPSS/H, который повышает эффективность первой версии Д. Гордона.

В пособии дается сравнение GPSS/H с другими языками имитационного моделирования, исследуются вопросы статистического оценивания результатов моделирования и сокращения числа реализаций. В отдельных главах подробно изучаются особенности работы мощного отладчика языка имитационного моделирования; рассматриваются вопросы повышения эффективности GPSS/H за счет анимации результатов и интеграции с языком более высокого уровня SLX. Описываемые принципы иллюстрируются большим числом примеров и упражнений, что позволяет освоить методы моделирования.

Предназначено для преподавателей основ имитационного моделирования; исследователей, желающих использовать методы языка имитационного моделирования в практической деятельности; студентов и аспирантов в качестве учебно-методического материала.

УДК 519.682
ББК 22.18

ISBN 5-8088-0228-8

СОДЕРЖАНИЕ

УСЛОВНЫЕ СОКРАЩЕНИЯ

АМП — амперпеременная
БСВ — базовая случайная величина
ВД — временная дискрета
ВДв — время движения
ВЭ — вычислительный эксперимент
ГПС — гибкая производственная система
ГСЧ — генератор случайных чисел
ДО — дисциплина обслуживания
ДСВ — дискретная случайная величина
ЕВА — единица времени анимации
ЖЦ — жизненный цикл
ИН — идентификационный номер
ИПС — индикатор перехода к сканированию
ИС — индикатор состояния
ИТ — информационные технологии
КМ — компьютерное моделирование
ЛМВ — локальное модельное время
ММ — математическое моделирование
МФ — модельный файл
НОД — наибольший общий делитель
ОБ — оператор блока
ОВМ — обобщенный «будстреп»-метод
ОВО — оператор выходного отчета
ОО — оператор описания
ООП — объектно-ориентированное программирование
ОУ — оператор управления
ПО — программное обеспечение
ПП — программный пакет (продукт)
ППП — пакет прикладных программ
РР — равномерно распределенная случайная величина
СБС — список будущих событий
СЛА — стандартный логический атрибут
СМО — система массового обслуживания
СОБ — следующий оператор блока
СП — список пользователя
СС — счетчик свершений
ССА — стандартный символьный атрибут
СТС — список текущих событий
СУ — система управления
СУБД — система управления базами данных
СЧА — стандартный числовой атрибут
СЭ — случайный элемент
ТОБ — текущий оператор блока
УПМ — управляющая программа моделирования
ФИС — флаг изменения состояния
ЦКП — центральное композитное планирование
ЯИМ — язык имитационного моделирования

ПРЕДИСЛОВИЕ

Нарастающее усложнение систем управления всех видов (технических, организационно-технических и организационных) приводит к пониманию невозможности адекватного описания процессов функционирования таких систем только аналитическими методами. Недаром мировая практика научных исследований свидетельствует о том, что методы компьютерного моделирования (КМ) занимают около 70 % в общем объеме исследовательского инструментария.

Стремительное внедрение новых информационных технологий (ИТ) в корне меняет методы познания окружающего нас мира. На знамени XXI века начертаны слова: информационные технологии, методы управления и качество.

Информационные технологии включают в себя компьютерную поддержку жизненного цикла изделий (CALS-технологии), интеграцию всех этапов разработки и изготовления продукции (ICAM-технологии), автоматизацию разработки программного обеспечения (ПО) (CASE-технологии), создание вычислительных сетей различного уровня и многое-многое другое.

Современные методы управления должны учитывать идеи реинжиниринга корпораций, ситуационного менеджмента, целевого проектирования, ориентацию на суженное производство (lean production) и, соответственно, опираться на возможности современных ИТ.

Методы менеджмента качества сложных систем не мыслятся сегодня без использования инструментов инжиниринга качества, таких как структурирование или развертывание функции качества, робастное проектирование, система Махаланобиса—Тагути, метод шести сигм и т. д. Естественно, что ни один из этих методов не может не опираться на современное ПО.

Даже такое простое перечисление ключевых направлений современного научно-технического прогресса позволяет заключить, что исследование и разработка новых проектов невозможна без использования методов компьютерного моделирования. Мир КМ многогранен и охватывает спектр от моделирования простых систем и производственных ситуаций до исследования взаимодействия социальных, экономических и природных систем в их сложном переплетении и взаимодействии. Рассмотрению проблем КМ посвящен ряд публикаций [5, 6, 8, 9].

На протяжении ряда лет автор, работая в промышленности, занимался вопросами моделирования сложных систем управления [2, 3]. В данном учебном пособии автор ставит более скромную задачу — рассмотреть особенности языка имитационного моделирования (ЯИМ) GPSS/H. Зачастую упоминание об этом ЯИМ вызывает легкое отторжение, так как у всех возникает воспоминание о первой версии GPSS 1963 г. Однако рассматриваемая здесь версия GPSS/H 1999 г. не только коренным образом отличается от ранних версий, но и имеет ряд уникальных черт, делающих ЯИМ приближенным к универсальным языкам программирования и открывающим ряд новых возможностей, весьма важных при исследовании систем

массового обслуживания (СМО). Под разряд СМО подпадают многие задачи из перечисленных направлений. Опыт преподавания ЯИМ в Санкт-Петербургском государственном университете аэрокосмического приборостроения в рамках курсов «Исследование систем управления», «Автоматизированные системы научных исследований» и «Менеджмент качества» показывает, что студенты и аспиранты не только осваивают идеи ЯИМ GPSS/H, но и успешно применяют их в своей практической инженерной и управленческой деятельности.

Прекрасно понимая, что проблема КМ весьма сложна и ее разные аспекты требуют отдельных монографий, автор ограничился рассмотрением специфики и возможностей одного из ЯИМ, а именно GPSS/H. За рубежом его описанию посвящено большое число публикаций, например [12–15], а на русском языке написана только глава в монографии автора [4]. Тем не менее, ЯИМ заслуживает большего внимания и несомненно достоин распространения не только для целей обучения студентов, но и для практического использования инженерами, менеджерами и научными работниками.

Литературные источники легко могут быть пополнены с помощью поисковых систем Интернета по ключевому слову Simulation — GPSS/H.

Автор выражает искреннюю признательность профессору Мичиганского университета США Томасу Шрайберу, творческие контакты с которым способствовали написанию этого учебного пособия, а также президенту Wolverine Software Corp. Джеймсу Хенриксену, предоставившему в распоряжение автора ряд уникальных материалов.

Автор с сожалением констатирует, что в известных ему учебных заведениях до сих пор пользуются первыми версиями ЯИМ GPSS, и надеется, что пособие будет способствовать продвижению новых идей.

Учебное пособие рассчитано на широкий круг читателей: так, исследователи смогут применять возможности ЯИМ для решения своих задач, преподаватели — использовать материалы пособия для лекций, а студенты и аспиранты — для изучения общих идей имитационного моделирования.

Глоссарий

Необходимость введения словаря, предваряющего основной текст, объясняется тем, что:

— в существующей научно-технической литературе используются различные термины для определения одних и тех же или сходных понятий, связанных с КМ;

— область моделирования, представляемая в пособии, достаточна узка и имеет специфические определения, не используемые порой при моделировании глобальных систем.

Определения, вошедшие в словарь, заимствованы из источников [4, 13] и ГОСТов.

Амперпеременная — переменная, введенная в GPSS/H и резко увеличивающая его вычислительную мощность. Название объясняется тем, что переменная обозначается знаком амперсанд — &.

Антитеза — значение случайного числа дополнительного потока, являющееся разностью между единицей и значением случайного числа основного потока.

Атрибут — стандартные числовые, символьные и логические характеристики, а также характеристики встроенных функций.

Буфер (накопитель или память) — одна из единиц аппаратной категории, способная накапливать приходящие заявки.

Вычислительный эксперимент (прогон) — процесс получения характеристик моделируемой системы на компьютере в течение одной *реплики*. Число реплик в прогоне зависит от заданной статистической точности или от времени, отведенного на прогон.

Имитационное моделирование — часть компьютерного моделирования, позволяющая: а) получать на ЭВМ статистические характеристики случайного процесса функционирования системы, б) определять лучший способ управления в детерминированных структурах путем проведения вычислительного эксперимента.

Заявка (в GPSS — транзакт) — динамический элемент, движущийся по модели от оператора к оператору блоков строго один в единицу времени.

Канал — путь прохождения заявок через систему массового обслуживания (различают одноканальные и многоканальные СМО).

Компьютерное моделирование — расширение возможностей математического моделирования за счет использования последних достижений ИТ, внедряемых во все сферы человеческой деятельности.

Модель — отображение особенностей реальной системы путем создания концептуального или машинного описания.

Модуль — часть программы ИМ, включающая неисполняемый модуль описания, модуль исполнения и модуль управления.

Оператор — элемент ЯИМ, предназначенный для построения машинной модели (различают операторы описания (compiler directives), операторы блоков (blocks) и операторы управления (control statements)).

Операнд — числовое или символьное значение одной из характеристик оператора любого вида.

Поток — последовательность событий при моделировании (различают потоки случайных чисел, генерируемых встроенными генераторами, входные потоки заявок на обслуживание, выходные потоки обслуженных транзактов, потоки необслуженных заявок).

Прибор (устройство) — одна из единиц аппаратной категории, осуществляющая обслуживание транзактов.

Приоритет — показатель значимости приходящего транзакта, являющийся одним из его операндов.

Прогон — проведение вычислительного эксперимента с моделью, состоящего из n реплик.

Процесс — совокупность событий, действий и связанных с ними временных отрезков, характеризующих функционирование системы или ее модели.

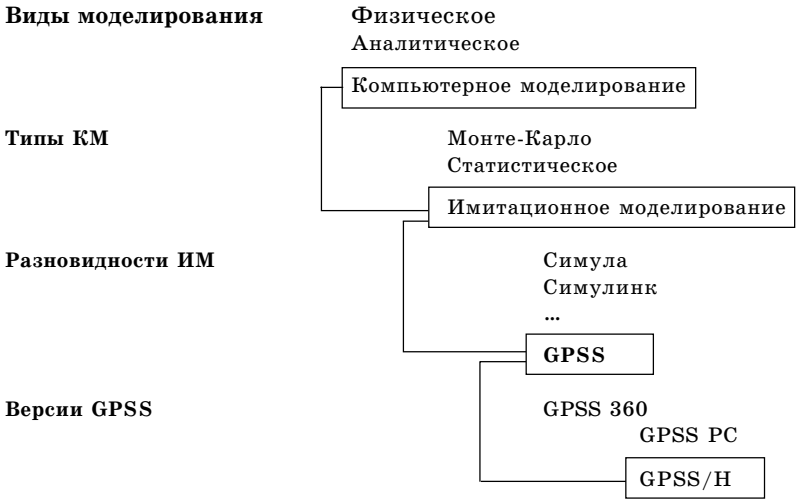
Список (последовательность) (в первоисточнике Chain — цепь) — перечень транзактов, находящихся в одном из шести возможных списков: будущих событий, текущих событий, прерываний, синхронизации, пользователя и, очень редко, применяемого системного.

Реплика — одна реализация вычислительного эксперимента или прогона.

Примечание. Приведенный список терминов дает общее представление о направленности пособия, внесенные в глоссарий термины будут ниже определены более подробно. Здесь будем придерживаться англоязычной нотации и употреблять сочетания «плавающая точка», хотя нам привычней говорить «запятая».

Структура учебного пособия

Структура предлагаемого учебного пособия может быть представлена следующей блок-схемой, на которой показана логика выбора целевой направленности пособия, которая ограничена исследованием особенностей и возможностей GPSS/H.



Следуя указанной логике, пособие состоит из трех частей (десяти глав) и приложений.

Первая часть «Общие вопросы компьютерного моделирования» содержит три главы. Первая посвящена общим представлениям о классификации моделей, уточнению места компьютерного моделирования, содержанию его отдельных разделов, уточнению схемы и порядка КМ. Во второй главе рассматриваются элементы теории массового обслуживания, являю-

щиеся основой для транзактного способа создания квазипараллелизма при ИМ. В третьей главе рассматриваются вопросы определения модельного времени, создания квазипараллелизма и особенности генерирования базовых случайных величин (БСВ) с помощью генераторов случайных чисел.

Вторая часть «Язык имитационного моделирования GPSS/H» содержит пять глав. В четвертой и пятой главах уточняется место GPSS/H среди других ЯИМ, проводится сравнительный анализ всех версий ЯИМ GPSS, рассматриваются принципы построения и функционирования ЯИМ GPSS/H. В первую очередь обращается внимание на внутреннюю организацию ЯИМ, что позволит более осознанно работать с пакетом. Опыт преподавания выявил ряд моментов, наиболее трудных для восприятия студентами. К их числу относятся правила прекращения испытаний, организация непоследовательного прохождения транзактов, правила терминирования и некоторые другие. Шестая глава посвящена рассмотрению моделей одноканального и многоканального обслуживания в пакетном режиме. Глава содержит ряд параграфов, посвященных различным аспектам ИМ (сбор дополнительной информации, автоматизации процесса ИМ, использование приоритета и т. п.). Седьмая глава посвящена работе с отладчиком (дебаггером) ЯИМ — рассмотрены особенности дебаггера и приведены примеры. В восьмой главе рассмотрены некоторые статистические задачи, посвященные использованию антитез, позволяющих сократить число реплик при одновременном уменьшении дисперсии, а также задача выбора наилучшего среди различных вариантов.

Третья часть содержит краткое описание новых программных пакетов (ПП), расширяющих возможности ЯИМ GPSS/H.

В приложениях приведены многие дополнительные характеристики ЯИМ, которые важны для специалистов, непосредственно использующих пакет ЯИМ, но могут затруднять восприятие материала при изучении основных положений.

Адрес приобретения ПП

Разработчиком и распространителем ПП GPSS/H:

— профессиональной 32-разрядной версии, работающей под Windows 98, 2000, NT, XP;

— студенческой версии, работающей в эмуляции DOS или под любой оболочкой типа NC,VC, Far;

— пакета анимации движения транзактов Proof Animation;

— интегрированного пакета SLX (C + GPSS/H + специализированный язык) — является

Wolverine Software Corporation
7617 Little River Tumpike, Suite 900
Annandale, VA 22003-2603
mail@wolverinesoftware.com

Все реквизиты и прайс-листы ПП можно найти на сайте корпорации.

Часть 1
ОБЩИЕ ВОПРОСЫ КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ

Глава 1
ПРЕДСТАВЛЕНИЕ О КОМПЬЮТЕРНОМ МОДЕЛИРОВАНИИ

§ 1.1. КЛАССИФИКАЦИЯ МОДЕЛЕЙ

Возможность представления моделью (лат. modulus — мера, образец) природных явлений, процессов или объектов окружающего нас мира присуща человеку-исследователю еще с ранних этапов развития общества. «Модель (по определению БСЭ) — образ (в том числе условный или мысленный) какого-либо объекта или системы объектов, используемый при определенных условиях в качестве их "заместителя" или "представителя".» Достаточно вспомнить хрустальный небесный свод геоцентрической модели, модель атома и т. д. При представлении модели средствами математики и логики возникает абстрактный образ реального объекта, при исследовании образца реального объекта в качестве модели имеет место конкретное исследование. Таким образом, моделирование, в том числе и имитационное, находится в промежутке между этими двумя крайними точками (рис. 1.1).

Модель должна ответить на множество вопросов исследователя: что будет, если ...?; каковы размеры ...?; насколько корректны упрощения ...? и множество других.

В результате модель из вспомогательного средства, заменяющего исследуемый объект (модель автомашины, портновский манекен), стала превращаться в способ получения информации о вновь создаваемой исследуемой или управляемой системе.

Подчеркнем, что под *информацией* будем понимать не столько продукт человеческого разума, получаемый в процессе познания, сколько объективную философскую категорию, связывающую темп

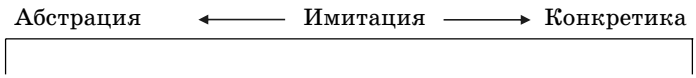


Рис. 1.1. *Место имитационного моделирования в модельном пространстве*

процессов, происходящих в системе, с уровнем организации самой системы.

Информация через философскую категорию *отражение* связана с категорией *материя* [4]. Под отражением понимается свойство материи воспринимать и сохранять в своей структуре следы воздействия другой системы. По А. Урсулу, информация — отраженное разнообразие реальности мира. Отсюда, чем полнее и разнообразнее модель динамической, нелинейной и неравновесной системы, тем более адекватно она отображает реальную систему. Информация объективно присутствует всегда, даже в случае классической механики; так, при соударении двух тел второе получало информацию от первого, как ему двигаться. Правда, при этом вся информация сводилась к одной динамической силе и не давала ничего нового, поэтому и не бралась в расчет. Учет информации даже в простых случаях классической динамики мог бы дать единую концепцию, объединяющую все разделы физики. Однако этот вопрос при всей его важности не имеет отношения к настоящему учебному пособию.

Вернемся к нашему представлению о моделях, полагая, что любой алгоритм — это модель деятельности, а в силу системности Вселенной любая целесообразная деятельность невозможна без моделирования.

Классификация системного мира моделей весьма широка, поэтому рассмотрим суженную классификацию, отвечающую задачам пособия, и дадим к ней краткое пояснение (рис. 1.2).

Ф — физическое (прямое) моделирование, Ф1 предусматривает использование в качестве модели саму систему (опытный образец), а Ф2 — другую систему со схожей физической природой (макет авто-

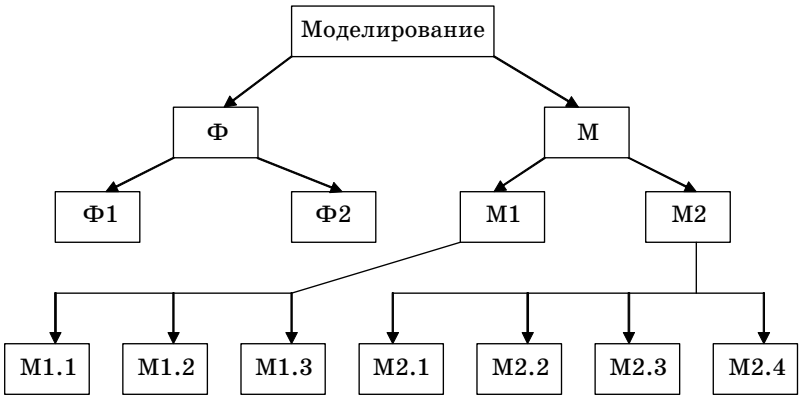


Рис. 1.2. *Классификация методов моделирования*

мобиля, сооружения, плотины). Такой вид моделирования способствовал созданию теории подобия.

М — математическое моделирование (ММ), распадающееся на две большие группы М1 и М2.

М1 — аналитическое моделирование, которое можно разделить на М1.1 — явное аналитическое описание искомых характеристик системы на одном из языков математики (см. гл. 2); М1.2 — приближенные численные методы, когда все объекты аппроксимируются числами или их комплектами в принятой числовой сетке, а результаты получаются в виде таблиц или графиков; М1.3 — качественные методы, когда изучаются свойства решений задач данного класса без нахождения самих решений. Зачастую эти методы реализуются с помощью экспертного оценивания. Такого вида методы широко используются в теории качества, квалиметрии, экономике, социологии и т. д.

М2 — компьютерное моделирование (см. § 1.2), когда математическая модель интерпретируется в программу для ЭВМ. Характерно, что с появления статьи Дж. Неймана и С. Улама в 1948 г. — первой работы по применению метода Монте-Карло, многие специалисты продолжают называть КМ методами Монте-Карло или статистических испытаний. Это в принципе не верно, так как КМ разделилось на четыре направления [5, 9] (см. рис. 1.2).

М2.1 — методы Монте-Карло или методы вычислительной математики, использующие методы М1.2 с учетом возможностей современных компьютеров. Этими методами можно вычислять любые, не берущиеся аналитическим путем, многократные интегралы, решать системы уравнений. Интересующимся методами вычислительной математики следует обратиться к многочисленной литературе.

М2.2 — методы ИМ (simulation), для которых характерно воспроизведение на ЭВМ процесса функционирования системы с сохранением его логической структуры и последовательности его протекания во времени, что позволяет путем многократного повторения набрать необходимые статистические данные и судить о состоянии объекта в различные моменты времени, оценивать выходные характеристики, выбирать оптимальное поведение или проводить сравнение альтернативных вариантов. Основной акцент пособия делается на рассмотрение именно ИМ.

М2.3 — методы статистической обработки данных моделирования на основе методов планирования эксперимента. Имеется целый ряд монографий, посвященных этим вопросам, в том числе [1, 7].

В настоящее время существует большое число пакетов прикладных программ (ППП), которые условно можно разбить на три группы:

— пакеты углубленного статистического анализа, написанные специалистами по статистике для таких же специалистов, с собственным языком, позволяющим программировать новые статистические процедуры (SAS, Statgraphics);

— пакеты базовой статистики, ориентированные на пользователей, не являющихся специалистами по статистическому анализу, и содержащие классические методы анализа с дружественным пользовательским интерфейсом в виде многочисленных пояснений, примеров и подсказок, среди них необходимо отметить прекрасный раздел по статистике в ППП MATLAB и ППП «Статистика»;

— проблемно-ориентированные пакеты, использующие терминологию и критерии в данной предметной области (экология, медицина и т. п.).

М2.4 — комплексы ИМ, объединяющие все названные виды КМ, пользовательский интерфейс, автоматизированные системы поддержки принимаемых решений и т. д. Это перспективное развивающееся направление предназначено для исследования сложных систем (подробнее см. работу [9]).

§ 1.2. КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ

Из классификации (см. рис. 1.2) в силу направленности пособия будем рассматривать только один из видов КМ, а именно имитационное моделирование СМО, и частично вопросы статистической обработки результатов ИМ.

Тем не менее, приведем краткое описание математического моделирования, которое позволяет благодаря абстрактным математическим формулам точно, однозначно и количественно оценить исследуемый объект. Усложнение исследуемых систем привело к резкому усложнению их математического описания, что, в свою очередь, приводило к необходимости делать всевозможные упрощающие допущения. При этом возникла опасность ухода от реального представления о системе. Выходом из этого положения являлся либо прогресс самих математических методов, либо изыскание иных методов описания. Появление мощных современных компьютеров и возникновение информационных технологий привело ко второму рождению ММ. Оно стало вторгаться практически во все сферы человеческой деятельности. В ряде областей ММ стало вытеснять физическое моделирование, так произошло, в частности, в авиационной промышленности, где начался демонтаж аэродинамических труб. Дальнейший прогресс ММ идет за счет:

— разработки новых численных методов решения задач моделирования, возможных только при условии использования ИТ;

— стремительного увеличения объемов памяти и производительности ЭВМ, что позволяет на порядки увеличить размерности решаемых задач и перейти к качественно новым задачам моделирования. Так, прогресс ММ позволил: исследовать эффекты синергизма (комбинированное действие, когда выходной эффект системы превышает действие, оказываемое компонентами по отдельности); оценивать бифуркационные состояния (вероятностное разветвление процесса функционирования системы); прогнозировать развитие диссапативных структур (переход в качественно новое состояние, характеризующееся более высоким уровнем самоорганизации); создавать более совершенные модели Вселенной (Большой взрыв, горячая Вселенная, теория струн), теории искусственного интеллекта, исследований в разных областях социологии, экономики и т. д.

В то же время возникла мощная оппозиция применению ММ в плохо структурируемых, не формализуемых областях, т. е. в таких областях, где человек (оператор, ЛОР, ЛПР) является основным элементом. По определению академика РАН РФ А. А. Самарского, процесс ММ базируется на триаде «модель—алгоритм—программа». До появления ЭВМ основную роль играла модель в виде математических уравнений, алгоритм представлял собой схему ручных расчетов для приближенного решения уравнений, а программа отсутствовала вообще. В начале использования ЭВМ первого поколения программе отводилась второстепенная роль — представление алгоритма в машинных кодах. Развитие ИТ привело к тому, что ЭВМ стали использовать для моделирования процессов функционирования системы, причем в этом случае имелись алгоритм и программа, а математическая модель в ее классическом виде практически отсутствовала или молчаливо предполагалось, что математической моделью является одно из аналитических представлений. Это направление получило название имитационного моделирования и представлено в работах Н. П. Бусленко, Н. Н. Моисеева, Р. Шеннона и многих других. Таким образом, в ММ началось опережающее развитие третьей компоненты триады — программы или программного обеспечения процесса моделирования.

Необходимо четко понимать разницу между программированием и моделированием. Программирование в настоящее время располагает большим арсеналом языков программирования, средств автоматизации управления вычислительными ресурсами, создания программ, автоматизации работ с большими массивами данных, так называемых систем управления базами данных (СУБД).

Однако весь этот арсенал направлен на создание программ, но модель не должна превращаться только в программу, описывающую абстрактные алгоритмы или логические отношения. Компьютерная модель должна оставаться прежде всего моделью реального объекта не зависимо от того, чем описывается его поведение: набором формул или правил, графиком или прогнозными оценками экспертов. Поэтому модель должна допускать исследование всех интересных возможностей: анализ чувствительности, изменение выходных характеристик, определение областей устойчивости и степень робастности, оптимизацию параметров, оценку вариантов построения и т. д. В связи со сказанным все чаще в литературе [4–6, 11] появляется термин *компьютерное моделирование*. КМ объединяет достижения математического моделирования, системного программирования и информационных технологий.

Автор не претендует на формулирование понятия КМ, помня все перипетии известной басни С. Михалкова «Слон живописец», однако приводит некоторые характерные черты КМ, инвариантные к области применения и направлениям исследования. Итак, КМ обладает:

• способностью понимать, интерпретировать и использовать формализованную и не формализованную информацию (математические формулы, логические правила, вербальные описания и т. п.);

• различными формами представления данных и знаний, заполняя пространство между ММ с его аналитическими формами описания и искусственным интеллектом с его формами и правилами представления знаний;

• способностью участвовать в процессе не только автоматизации научных исследований за счет использования самой ЭВМ для модификации различных режимов применения КМ, но и интеграции всех этапов жизненного цикла системы путем использования быстро развивающихся методов ИТ (широко распространенные во всем мире CALS-технологии, CASE-технологии, технологии ICAM и IDEF);

• возможностью расширения круга пользователей, от узкого круга специалистов-математиков и профессиональных программистов до большого класса исследователей, не обладающих профессиональными знаниями в областях математики и программирования, но хорошо знающих предметную область и умеющих обращаться с ППП.

Помня, что КМ разделяется на четыре подгруппы, следует иметь в виду, что обычно при использовании ЯИМ приходится обращаться и к методам Монте-Карло, и к возможностям ППП, созданных для обработки статистических данных. Поскольку задачей пособия является рассмотрение одного из ЯИМ—GPSS/H, специальные вопросы, относящиеся к М2.1, М2.3, М2.4, здесь не рассматриваются. Неко-

торые вопросы освещаются в работах [4, 5, 11], имеются многочисленные ссылки на литературные источники; кроме того, нельзя забывать о неисчерпаемых ресурсах Интернета.

§ 1.3. МЕСТО ИМИТАЦИОННЫХ МОДЕЛЕЙ В ОБЩЕЙ СТРУКТУРЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

По Р. Шеннону, имитация — это «процесс конструирования реальной системы и постановки эксперимента на ней». При этом любые характеристики определяются за счет проведения прогона или нескольких прогонов модели, каждый из которых включает заданное число реплик (реализаций вычислительного эксперимента). ИМ можно использовать в двух направлениях:

1) рассматривать случайные процессы функционирования системы и определять статистические характеристики, что интересно в первую очередь разработчикам и исследователям системы;

2) при известном или детерминированном процессе функционирования системы определять разные варианты построения, элементов конструкции или стратегии управления, что интересно в первую очередь конструкторам, архитекторам или менеджерам.

Оба направления имеют право претендовать на соответствие классическому определению Шеннона. Чтобы уяснить место имитационных моделей в общей структуре ПО, рассмотрим уровни построения ПО.

Уровень 1. Машинные коды, автокоды, машинно-ориентированные языки, операционные системы.

Уровень 2. Алгоритмические языки высокого уровня (C^{++}, Pascal и др.), системы программирования СУБД.

Уровень 3. Специализированные алгоритмические языки моделирования, в том числе и имитационного (SIMULA, SIMSCRIPT, GPSS и др.).

Уровень 4. Интегрированные системы ИМ (например, SLX, СИМ), автоматизированные системы искусственного интеллекта (экспертные, поддержки принятия решений).

Объекты 1-го уровня не требуют никаких комментариев.

Языки 2-го уровня при их универсальности дороги и сложны.

Языки 3-го уровня, теряя в универсальности, приобретают направленность на конкретную область и становятся простыми. Отметим, что GPSS/H, сохранив все преимущества языков 3-го уровня, вобрал в себя многие положительные черты языков 2-го уровня (подробнее см. гл. 4).

Учитывая, что число ЯИМ на сегодняшний день превышает 600, выбор ЯИМ зависит от многих факторов: предметной области, квалификации пользователя, наличия соответствующей ИТ и т. д.

Языки ИМ обладают рядом весьма привлекательных достоинств:

— удобством описания структуры исследуемой системы;

— учетом динамики функционирования и начальных условий;

— возможностью повторения испытаний;

— возможностью отладки и контроля;

— сбором и анализом необходимой статистики и принятием решений;

— простотой восприятия;

— простотой обучения;

— возможностью использования разработчиками системы, не являющимися специалистами в области математики и программирования.

Четвертый уровень включает в себя проблемно-ориентированные интерактивные системы, которые можно разделить на автоматизированные экспертные, оптимизационные системы, а также имитационно-моделирующие комплексы.

На фоне роста информационных потоков, создания баз знаний, сложных поисковых систем ИМ превращается из средства системного анализа в средство исследования, испытаний и отладки сложных систем. Поэтому перспективы применения ЯИМ не только не утрачивают актуальности, а приобретают все большее значение в процессе создания сложных систем.

§ 1.4. ДОСТОИНСТВА И НЕДОСТАТКИ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ

Имитационное моделирование позволяет решать ряд сложных задач и имеет *преимущества*:

— при создании ИМ законы функционирования системы могут быть неизвестны, поэтому постановка задачи исследования является не полной и ИМ служит средством изучения особенностей процесса. При этом можно руководствоваться связями между компонентами и алгоритмами их поведения;

— при проведении ИМ выявляется характер связей между внутренними параметрами системы и выходными характеристиками;

— при проведении ИМ можно менять темп моделирования: ускорять при моделировании явлений макромира (например, процессов на Солнце) или замедлять при моделировании явлений микромира (например, процесс существования элементарных частиц);

— при проведении сравнения и выбора альтернатив;

— при изучении узких мест в системе;

— при подготовке специалистов, осваивающих новую технику.

Из перечисленного следует, что ИМ применяется для решения широкого спектра задач практически любой сложности в условиях неопределенности, когда аналитическое моделирование оказывается практически не применимым.

Достоинства ИМ

1. Возможность объединять традиционные математические и экспериментальные компьютерные методы.

2. Высокая эффективность применения при исследовании АСНИ, САПР, экспертных систем, сложных систем управления. По данным RAND Corp., консалтинговые фирмы из всей гаммы возможных средств анализа: линейного, нелинейного, динамического программирования, методов исследования операций, вычислительных методов — более чем в 60 % случаев прибегают к ИМ, так как ИМ позволяет получать ответы в терминах, понятных и привычных для пользователя.

3. Возможность исследовать объекты, физическое моделирование которых экономически нецелесообразно или невозможно.

4. Испытания объекта связаны с опасностью для здоровья человека.

5. Исследование еще не существующих объектов.

6. Исследование труднодоступных или ненаблюдаемых объектов.

7. Исследование плохо формализуемых экологических, социальных или экономических систем.

8. Исследование объектов практически любой сложности при большой детализации и снятии ограничений на вид функций распределения случайных величин.

Недостатки ИМ

1. Самым существенным недостатком является невозможность получить точечную оценку исследуемых характеристик, так как в результате ИМ можно оценить только математическое ожидание и дисперсию.

2. Потеря общности результатов, так как при ИМ оценивается конкретная система.

3. Трудности оптимизации, так как ИМ отвечает на вопрос, «что будет в случае, если..?», но не определяет, будут ли эти условия наилучшими.

4. Трудности с оценкой адекватности ИМ.

5. Создание ИМ сложной системы длительно по времени и требует значительных денежных средств.

Несмотря на эти недостатки, все большее число исследователей прибегает к использованию ИМ в силу достоинств, указанных выше.

Для составления сложной ИМ необходимы опыт и приобретаемые на практике навыки. Это следует учитывать, чтобы при первых неудачах не наступило разочарование в возможностях ИМ.

§ 1.5. ОСНОВНЫЕ ЭТАПЫ И ЗАДАЧИ, РЕАЛИЗУЕМЫЕ ПРИ ИМИТАЦИОННОМ МОДЕЛИРОВАНИИ

Процесс ИМ системы можно разделить на три последовательно выполняемых этапа (рис. 1.3):

I — построение математической (концептуальной) модели S' (см. гл. 2).

II — разработка моделирующего алгоритма S'' (см. гл. 4);

III — исследование системы S с помощью модели S'' (см. гл. 6, 8).

Процесс ИМ не является строго поступательным, между этапами существуют обратные связи, позволяющие вводить новую информацию, вносить уточнения и корректировки.

Построение математической модели на I этапе включает в себя пять взаимосвязанных подэтапов [4, 11]:

1) уяснение и постановка задачи, определение целей исследования;

2) декомпозиция системы на компоненты, допускающие удобное математическое или алгоритмическое описание;

3) определение параметров, переменных, пространства состояний системы, установление пределов изменения каждой характеристики;

4) выбор выходных показателей, т. е. вектора $\mathbf{Q}$;

5) аналитическое описание с помощью выбранной концептуальной модели S' системы S и проверка ее адекватности.

Построение математической модели системы S начинается с определения параметров системы и переменных, определяющих процесс функционирования системы.

На II этапе при переходе от концептуальной модели S' к моделирующему алгоритму и имитационной модели S'' можно выделить пять основных подэтапов:

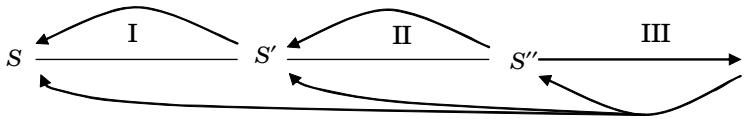


Рис. 1.3. *Этапы представления модели*

1) выбор способа имитации (см. гл. 3), а также вычислительных и программных средств реализации ИМ;

2) построение логической схемы моделирующего алгоритма;

3) алгоритмизация математических моделей, описывающих поведение элементов системы и связей между ними в рамках выбранного способа имитации;

4) программирование моделирующего алгоритма, т. е. разработка самой имитационной модели;

5) отладка, тестирование и проверка адекватности ИМ.

На III этапе можно выделить три основных подэтапа:

1) планирование вычислительных экспериментов;

2) проведение прогона или прогонов;

3) обработка, анализ и интерпретация результатов.

На I этапе проводится выяснение особенностей, которые представляют интерес для исследователя, уточняются виды и формы преобразования материальных, энергетических и информационных потоков, уточняются взаимодействия с другими системами и окружающей средой. Таким образом осуществляется проблемная ориентация. При этом сближаются точки зрения и снимаются имеющиеся разногласия. Само концептуальное описание правильнее начинать со специальной задачи, а затем универсализировать ее на класс задач. Информация должна получаться из всех возможных источников: Интернета, публикаций в специальных журналах, выбора аналитических моделей, суждений экспертов. Серия стандартных вопросов этого этапа такова:

- что представляет собой система (характеристики, состав, взаимодействия, отношение с окружающей средой);
- как характеристики эволюционируют во времени;
- каков характер корреляционных связей;
- обладает ли система свойствами робастности и т. д.

Все это позволяет описать концептуальную модель, для чего определяются входные параметры, возмущающие воздействия, выходные реакции системы. После чего обязательно проверяется ее адекватность, т. е. насколько модель отвечает реальной системе и какова степень доверия результатам ИМ (см. подробнее § 1.6).

На II этапе производится составление ИМ. В случае описания сложных систем ИМ может создаваться усилиями специалистов разных отделов и даже разных организаций, при этом уровень может доходить до подмоделей, их отладки и сборки общей ИМ. На этом этапе также проводится оценка адекватности, но в отличие от I этапа, где определяется, *правильна ли модель*, на II этапе проверяют, *насколько правильно работает эта модель*, и проводят пилотный прогон модели.

Наконец, III этап является основным, ради которого и строилась модель, на этом этапе набирается статистика, позволяющая принимать решения по любой из основных задач ИМ.

Следующие группы задач ИМ можно отнести к основным.

1. Оценка значений показателей выходного вектора **Q**, а также значений параметров компонентов.

2. Нахождение функциональной зависимости между вектором **Q** и значениями параметров системы.

3. Сравнение систем с разными вариантами структур и разными значениями параметров для одной функциональной структуры.

4. Оптимизация системы S на множестве параметров на основе двойственной задачи оптимизации (достижение максимума выходного параметра при заданном значении ресурсов либо минимизация ресурсов при достижении заданного значения выходного параметра).

Любая из названных задач может быть решена на III этапе с помощью методов планирования имитационных экспериментов. В результате ИМ системы S при заданном времени и векторе параметров системы получается фазовая траектория системы S

$$\{x(t) \in X,\ t \in T\} \tag{1.1}$$

и значения выходных показателей для каждого интервала модельного времени и для всего интервала моделирования в целом

$$q(t) = q(\theta',\ T). \tag{1.2}$$

Результаты одного вычислительного эксперимента (ВЭ), имитирующего фазовую траекторию (1.1) и выходные показатели (1.2), называются *репликой* или реализацией. Итогом одной реплики является одно значение искомой характеристики. Для получения представительной статистики необходимо провести ряд реплик в рамках одного прогона или ряд прогонов с меняющимися условиями, после чего проводится подэтап обработки данных ВЭ. Анализ полученной статистики может выявить и контринтуитивное поведение системы, что может быть вызвано плохим проведением I этапа или невыявленной неадекватностью модели исследуемой системе. Целью обработки статистики является не только оценка характеристик, но и уточнение асимптотических свойств, определение колебательности системы и областей устойчивости. На основе полученных результатов принимаются необходимые решения. Названные группы задач практически перекрывают всю область возможных применений ИМ.

§ 1.6. ОЦЕНКА АДЕКВАТНОСТИ ИМИТАЦИОННЫХ МОДЕЛЕЙ

При ИМ неизбежно возникает проблема обоснования возможности перенесения на исследуемую систему управления (СУ) выводов, полученных при функционировании модели. Под адекватностью ИМ понимаем степень отражения параметрами модели характеристик исследуемой системы с точностью, требуемой для конкретного исследования. Оценка адекватности впрямую мало реальна, и чаще всего руководствуются косвенными соображениями типа:

— согласуются ли результаты со здравым смыслом, причем факт отсутствия противоречий или их наличие еще не доказывают неадекватность модели;

— согласуются ли результаты ИМ с предполагаемыми статистическими оценками.

Тем не менее, в последнее время появился ряд работ, делающих обнадеживающие попытки оценки адекватности, распадающейся на два связанных процесса:

верификации — т. е. проверки идентичности концептуальной модели исследуемой системы;

пригодности модели — возможности перенесения результатов моделирования на исследуемую систему.

Верификация — общепринятая процедура и чаще всего неизбежная. Верификация предусматривает предупредительные и отладочные процедуры.

К *предупредительным* относятся:

- проверка пригодности входных данных, контроль набора и т. п.;
- построение программы в виде трех разделов:
  структура модели,
  исходные данные,
  запуск программы со строгой последовательностью операторов;
- проверка датчиков БСВ;
- проверка точности вычислений (формат данных, округление, усечение).

Отладка начинается с анализа ошибок предыдущих этапов, возможности их повторения, изучения логики программы. Иногда полезно после написания машинной программы снова попытаться построить концептуальную модель. В процедуру отладки также входят корректировка синтаксиса и семантики, анализ чувствительности модели. Отладка ведется по разделам программы.

Оценка пригодности в ИМ — процедура достаточно спорная, считается даже, что она может дискредитировать полезную модель. Кроме того, оценка пригодности, являясь многокритериальным процес-

сом, достаточно сложна, и единой системы критериев для такой оценки не существует.

Анализ ряда статей позволил представить классификацию критериев оценки пригодности (табл. 1.1).

Техническая пригодность должна выяснить обоснованность теоретических посылок, положенных в основу модели. Вначале оцениваются все сделанные допущения, затем оценивается пригодность данных. Выявленные расхождения относятся к «узким местам» модели и должны быть уменьшены.

Операционная пригодность менее категорична, чем техническая, и допускает большие рассогласования. Особое внимание обращается на робастность, включающую анализ чувствительности на ошибки в процессе ИМ при задании экстремальных значений входным пара-

Таблица 1.1. *Критерии оценки пригодности*

Вид пригодности ИМ	Оценка пригодности		Условия анализа пригодности
Техническая	Теоретическая		Учет математических содержательных и причинных допущений
	Пригодность данных	необработанных	Оценка точности, беспристрастности и репрезентативности данных
		структуризованных	Учет точности операции сравнимости позиций
	Структурная		Правильность отражения внутренних взаимосвязей ИС
	Прогнозная		Способность предсказывать будущее
Операционная	Репликативная		Точность воспроизведения характеристик ИС
	Робастность		Учет чувствительности модели
	Реализационная		Вероятность практического внедрения
Динамическая	–		Актуализация, практика успешного использования модели

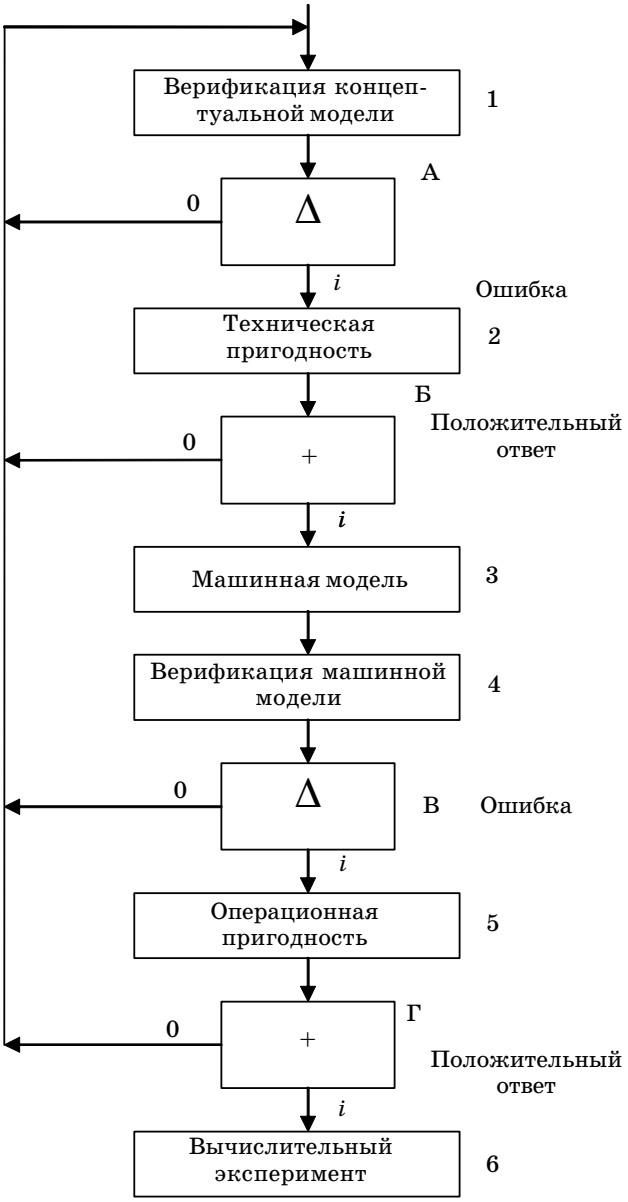


Рис. 1.4. *Фрагмент блок-схемы алгоритма*

метрам. Репликативная пригодность должна оценивать уровень точности воспроизведения характеристик индикатора состояния (ИС). Формально необходимо иметь две выборки и оценивать их статистическими методами (регрессионный и факторный анализ, хи-квадрат и F-критерии, тесты Тьюринга).

Динамическая пригодность. Расхождения во временном диапазоне, влияющие на операционную пригодность, оцениваются динамической пригодностью ИМ, а также возможностью актуализации и расширения данных.

В результате блок-схема алгоритма ИМ включает ряд действий, показывающий последовательность проведенных работ.

Методы оценки верификации и пригодности делятся на две группы:

— формальные (статистические);

— неформальные с привлечением пользователей и лиц, принимающих решение.

Из фрагмента блок-схемы алгоритма (рис. 1.4) видно, что оценка верификации и пригодности взаимосвязаны. Очевидно, что оценка адекватности является пошаговым процессом и обязательной частью процесса ИМ. Если необходимо внести коррективы в имеющуюся систему, то оценка адекватности приобретает реальные очертания. Так, результаты моделирования сравниваются с накопленными данными, и если совпадение находится в границах заданной статистической точности, то адекватность достаточно высока.

После этого в модель вносятся предполагаемые коррективы, и о проверке адекватности можно не беспокоиться. В случае вновь разрабатываемой системы приходится прибегать к сложным процедурам оценки адекватности, описанной выше.

Глава 2
КОНЦЕПТУАЛЬНАЯ МОДЕЛЬ ДЛЯ GPSS/H

§ 2.1. КЛАССИФИКАЦИЯ МАТЕМАТИЧЕСКИХ МОДЕЛЕЙ

Существует большое число аналитических моделей описания процессов функционирования систем, решения задач оптимизации или нахождения рационального решения, принятия решений и т. п. Пусть не обидятся на нас специалисты-математики, которые считают аналитические описания истиной в последней инстанции, но, с позиций системного подхода, в естественных и искусственных сферах природы можно классифицировать любые понятия. Поэтому и аналитические методы по-крупному можно свести в классификационную таблицу аналитических моделей, исходя из двух основополагающих понятий — протяженности во времени и характера возникновения событий (отношения с внешней средой).

Рассмотрим эти понятия подробнее.

Протяженность во времени

Можно рассматривать только два вида протекания какого-либо процесса во времени. Время может рассматриваться либо как непрерывная переменная $t \in [0, T]$, либо как дискретная $t = i\Delta t$, $i = 0,1, \ldots, M$, $M = [T/\Delta]$, где Δ — шаг дискретизации. Соответственно припишем индексы Н и Д этим двум видам процессов, описываемых аналитическими моделями. Индекс Н соответствует аналоговым сигналам (постоянный, монотонный, синусоидальный и т. д.). Индекс Д — дискретным сигналам (импульсный — в виде отдельного импульса или их последовательности; цифровой, подобно 1 и 0 в ЭВМ, и т. п.).

Характер возникновения событий

При отсутствии случайных возмущений со стороны среды и полной определенности поведения компонентов системы имеем постоянную или детерминированную модель, для которой выберем индекс П (во избежание путаницы с ранее введенным индексом Д). В большинстве случаев необходимо учитывать вероятностные характеристики всех воздействий, для этого класса моделей введем индекс В.

С учетом сказанного, классификация позволит рассматривать четыре типа моделей: НП-, ДП-, НВ-, ДВ-модели. Очевидно, что все многообразие аналитических моделей можно разнести по этим клас-

Таблица 2.1. Классификация математических моделей

Характеристика	Тип ММ			
	НП	ДП	ДВ	НВ
Вид зависимости	Дифференциальные и интегральные уравнения	Теория разностных уравнений, конечные автоматы	Разностные стохастические уравнения, вероятностный автомат	Стохастические дифференциальные уравнения, **теория массового обслуживания**

сам. Обзор аналитических моделей невозможно провести в рамках одного пособия, определенная попытка была сделана в работах [4, 11]. Ограничимся сводной табл. 2.1, которая содержит только некоторые аналитические модели и определяет область их применения.

Данная таблица не претендует на полноту, а является лишь иллюстрацией предлагаемой классификации, и в ней для дальнейшего изложения представляет интерес только часть НВ-моделей — теория массового обслуживания. Именно теория массового обслуживания является концептуальной моделью для GPSS, поэтому в следующих параграфах кратко рассмотрены основные идеи теории МО.

§ 2.2. ОСНОВНЫЕ ОБОЗНАЧЕНИЯ ТЕОРИИ МАССОВОГО ОБСЛУЖИВАНИЯ

Системы массового обслуживания встречаются на каждом шагу — невозможно назвать область человеческой деятельности, где не возникают проблема обслуживания и создание очереди при занятости органа обслуживания (документооборот, телефонная связь, бизнес, торговля и т. д. В гл. 4 приведен список сфер, где можно применять ИМ на GPPS/H, а значит, где применимы методы СМО).

В теории СМО разработаны идентичные модели для:

— проектирования и эксплуатации систем, состоящих из большого числа аналогичных или похожих, но отличающихся производительностью компонентов (количество персонала, число линий связи, количество таможенных пунктов и т. п.);

— отыскания оптимального количества оборудования (число лифтов в офисном здании, число грузовых терминалов и т. п.);

— определения производительности той или иной системы (производительность ЭВМ, пропускная способность канала связи и т. п.).

Очевидно, что можно назвать дополнительные возможности СМО и заключить, что во многих исследуемых системах сочетаются все указанные выше модели. Более того, теория СМО быстро развивается и появляются все новые модели. В связи с этим ниже будут изложены только основополагающие идеи СМО, не претендующие на какую-либо новизну, являющиеся ориентиром для читателей, не знакомых с теорией СМО, и приводимые для лучшего понимания концептуальных основ ЯИМ GPSS/H.

Систему МО можно описать рядом параметров.

• *Входной поток заявок или требований* $v(t)$ (в GPSS/H — транзакты), задающий вероятностный закон поступления заявок на обслуживание. Заявки могут поступать либо по одиночке, либо группами (пакетами). В GPSS/H входной поток задается оператором блока (ОБ) GENERATE (см. гл. 4, 5).

• *Поток обслуживания* $u(t)$, задающий вероятностный закон процесса обслуживания заявок. В GPSS/H поток обслуживания задается ОБ ADVANCE (см. гл. 5).

• *Прибор обслуживания* P_i, $i = 1, 2, ..., N$, состоящий из накопителя H_i емкостью $0 \le m \le \infty$, при $m = 0$ происходит потеря обслуживания, а при $m = \infty$ все заявки ожидают обслуживания, промежуточные значения определяют емкость накопителя. В состав прибора также входит канал обслуживания $К = n_j$, $j = 1, 2, ..., L$; при $n = 1$ обслуживание называется *одноканальным,* а при $n > 1$ — *многоканальным.*

Если приборы обслуживания соединяются параллельно, то такое обслуживание называется *однофазным,* а если приборы соединяются последовательно, — *многофазным* (ряд последовательных операций).

• *Очередь* — задержка в обслуживании поступающих заявок, характеризующаяся *дисциплиной очереди*, т. е. порядком обслуживания заявок. Можно назвать разные виды дисциплины обслуживания:

FIFO — первый пришел — первый вышел (обслужился), в англоязычной литературе эта известная аббревиатура все чаще заменяется на FCFS (first come first serve — первый пришел — первый обслужился);

LCFS — последний пришел — первый обслужился, эта дисциплина предназначена для заявок с более высоким приоритетом, но используется крайне редко, а чаще используется дисциплина следующего вида;

SPT (shortest processing time) — кратчайшее время обслуживания, которое применяется для заявок с приоритетом, в GPSS/H эта дисциплина реализуется ОБ PRIORITY;

случайная дисциплина, например система опроса слушателей на практических занятиях.

В пособии будут рассматриваться только дисциплины FCFS и SPT. Подчеркнем, что в GPSS/H факт создания очереди реализуется связанной парой ОБ QUEUE / DEPART.

• *Выходной поток* $y(t)$ — функция распределения, представляющая собой сумму двух вероятностных законов: $y_1(t)$ — поток обслуженных заявок и $y_2(t)$ — поток потерянных (не обслуженных) заявок, который образуется за счет отказа в обслуживании из-за малого объема накопителя по принципу $m + 1 < K$, где K — число заявок на входе прибора. В отдельных случаях заявка может остаться в приборе из-за окончания времени моделирования, поэтому в неравенстве появляется единица. Все сказанное объединим в рис. 2.1.

Таким образом, однофазные (простые) СМО могут быть либо одноканальными, либо многоканальными, многофазные СМО, представляющие последовательность различных операций, выполняемых различными приборами обслуживания, могут представлять собой комбинацию одно- и многоканальных СМО.

В 1953 г. Г. Кендалл предложил стандартные обозначения для введенных выше определений, которые и используются исследователями без изменений. Для однофазных СМО символика Кендалла выглядит следующим образом:

$$A / B / n / m,$$

где A и B — входной поток и поток обслуживания соответственно; n — число каналов, $n \le 1$; m — емкость накопителя.

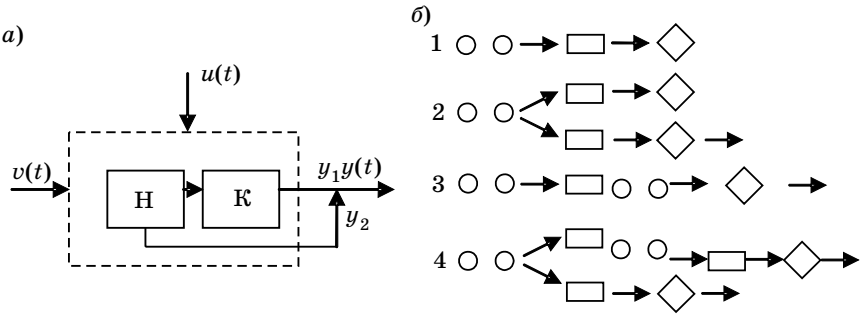


Рис. 2.1. *Блок-схема элемента СМО: а — прибор обслуживания; б — виды СМО:*
1 — одноканальная однофазная; 2 — многоканальная однофазная; 3 — одноканальная многофазная; 4 — многоканальная многофазная: Н — накопитель; К — канал обслуживания; ○ — поток входных заявок; ▭ — прибор обслуживания; ◇ — поток обслуженных заявок

Потоки случайных событий могут иметь различный вид:

M — экспоненциальное распределение длительностей интервалов поступления заявок или длительностей обслуживания (индекс M — от определяющего слова *марковский процесс*, т. е. такой, когда поведение процесса после момента времени t зависит лишь от состояния процесса в момент времени t и не зависит от поведения до момента времени t);

D — детерминированное распределение длительностей интервалов поступления заявок или длительностей обслуживания;

E_k — поток Эрланга k-го порядка для длительностей интервалов между приходами заявок или длительностей обслуживания;

GI — рекуррентный поток (длительности интервалов статистически независимы и имеют одинаковое распределение);

G — общий вид распределения.

Тогда в символах Кендалла вместо A и B подставляется символ одного из упомянутых потоков, например:

M/M/1 — экспоненциальные потоки с одним каналом обслуживания и неограниченной емкостью;

D/GI/5/10 — детерминированный входной поток, рекуррентный поток обслуживания, многоканальное СМО с пятью одинаковыми каналами, емкость накопителя 10 и т. д.

Описание любого варианта СМО на языке математики несложно, но практически малоценно, так как не отражает динамики процесса функционирования системы. Поэтому всегда существует необходимость, получив предварительные числовые ориентиры на основе аналитической модели, далее строить имитационную модель, для чего незаменим ЯИМ GPSS/H.

§ 2.3. НЕКОТОРЫЕ АНАЛИТИЧЕСКИЕ МОДЕЛИ СИСТЕМЫ МАССОВОГО ОБСЛУЖИВАНИЯ

Наиболее сложный случай оценки параметров потоков заявок и обслуживания — когда потоки являются *простейшими*. Простейшим называется такой поток, при котором время прихода заявок удовлетворяет условиям стационарности, отсутствия последействия и ординарности.

Стационарность случайного процесса означает, что на любом промежутке времени Δt вероятность прихода n заявок зависит только от числа n и величины промежутка Δt, но не изменяется от сдвига Δt по оси времени. При этом выполняется эргодическое свойство — статистическое равенство n заявок, полученных при ИМ одной системы,

или испытания n систем до прихода первой заявки. Формулирование этого свойства звучит достаточно просто: «Совокупное по времени наблюдений равняется совокупному по ансамблю наблюдений».

Отсутствие последействия означает, что вероятность прихода n заявок в течение промежутка времени Δt не зависит от того, сколько пришло заявок до этого момента времени. Выполнение этого условия гарантирует случайность и независимость событий.

Ординарность потока заявок означает невозможность появления более одной заявки в один и тот же момент времени. Как это реализуется в GPSS/H, будет рассмотрено в ч. 2.

Не будем рассматривать причины, приводящие к искажению сформулированных свойств простейшего потока, отметим только, что интервалы времени прихода заявок подчиняются в этом случае функции распределения Пуассона, а время обслуживания описывается экспоненциальной функцией распределения. Любые другие функции распределения приведут к лучшим параметрам потоков, поэтому считается, что если параметры СМО удовлетворяют условиям простейшего потока, то они наверняка обеспечат удовлетворительную работу СМО при всех других потоках. В связи с этим в рассматриваемых далее моделях используются функции распределения Пуассона и экспоненциальные.

2.3.1. Распределение вероятностей длительности интервалов между заявками

Пусть $f(t)$ — плотность распределения длительностей t интервалов между любой парой смежных заявок. Определим параметр потока λ как среднюю частоту появлений заявок, а $1/\lambda$ — как среднее значение длительности интервала, тогда

$$\int_0^\infty t f(t) \mathrm{d}t = 1/\lambda. \qquad (2.1)$$

Например, если за дискрету времени примем 1 ч, а $\lambda = 4$, то среднее количество поступлений равняется 15 мин ($1 / \lambda = 0{,}25$) и наоборот, если каждые 10 мин в систему поступает одна заявка, то частота поступлений λ равняется 0,1 заявки в минуту.

Для стационарного потока плотность определяется как

$$f(t) = \lambda e^{-\lambda t}, \ t \geq 0, \qquad (2.2)$$

такое распределение называется экспоненциальным.

Вычисляя вероятность попадания n заявок в произвольно выбранный интервал T, приходим к распределению Пуассона:

$$P_n(t) = ((\lambda t)^n / n!)\, e^{-\lambda t},\ n = 0, 1, 2, \ldots \qquad (2.3)$$

Полученные распределения отвечают всем свойствам простейшего потока, а именно:

— длительность интервалов взаимонезависима и одинаково распределена;

— ненулевая вероятность поступления заявок при $\Delta t > 0$;

— при интервале Δt, стремящемся к нулю, количество поступающих заявок не превышает единицы.

Впредь будем полагать, что отсчет времени начинается с момента $T \geq 0$.

Нетрудно показать, что экспоненциальная функция распределения заявок и пуассоновский процесс обладают одинаковыми статистиками, и их можно считать синонимами, поэтому и принято обозначение марковский процесс или М-процесс.

2.3.2. Распределение вероятностей длительностей обслуживания

Будем считать, что каждый канал в одно и то же время может обслуживать только одну заявку. Следующие друг за другом интервалы обслуживания независимы и имеют идентичное распределение.

Пусть плотность распределения равна $g(t)$, тогда среднее время обслуживания

$$T_0 = \int_0^\infty t g(t) \mathrm{d}t = 1/\mu, \qquad (2.4)$$

где μ — параметр (скорость) потока обслуживания.

Так, например, если за дискрету времени принять 1 ч, а $\mu = 5$, то в течение часа прибор обслужит 5 требований и среднее время обслуживания равно 12 мин и наоборот; если на обслуживание заявки уходит 30 мин, то скорость обслуживания $\mu = 2$. При расчете среднего времени обслуживания учитывается только время занятости прибора обслуживания.

Для получения верхней границы пропускной способности канала обычно полагают, что распределение длительностей обслуживания является экспоненциальным:

$$g(t) = \mu\, e^{-\mu t} \text{ при } t \geq 0, \qquad (2.5)$$

при этом вероятность завершения обслуживания в интервале $(t + \Delta t)$ не зависит от того, сколько времени уже потрачено на обслуживание этой заявки (пример системы, не обладающей памятью). Таким образом, если в момент t заявка уже обслуживалась, то в силу (2.5) в момент $(t + \Delta t)$ вероятность того, что в этом интервале обслуживание не заканчивается:

$$P(t + \Delta t) \approx e^{-\mu \Delta t}. \qquad (2.6)$$

Следовательно, при очень малых Δt вероятность того, что обслуживание в рассматриваемом интервале не заканчивается:

$$P(t + \Delta t) \approx 1 - \mu \Delta t, \qquad (2.7)$$

а что заканчивается:

$$P(t + \Delta t) \approx \mu \Delta t. \qquad (2.8)$$

2.3.3. Одноканальное обслуживание с пуассоновским входным потоком и экспоненциальным распределением длительностей обслуживания

Рассмотрим пример, в котором имеется возможность аналитического определения показателей эффективности функционирования СМО (M/M/1).

Пусть процесс обслуживания начинается при отсутствии заявок в накопителе, тогда состояние СМО описывается следующей системой уравнений:

$$P_n(t + \Delta t) = P_n(t)(1 - (\lambda + \mu)\Delta t + P_{n-1}(t)\lambda \Delta t + P_{n+1}(t)\mu \Delta t, \ n \geq 1;$$
$$P_0(t + \Delta t) = P_0(t)(1 - \lambda \Delta t) + P_1(t)\mu \Delta t, \qquad (2.9)$$

где $P_n(t)$ — вероятность нахождения системы в состоянии $x_n(t) \in X$ в момент t, т. е. когда в ней находятся n заявок.

Вероятность нахождения в системе n заявок в момент времени $(t + \Delta t)$ равна сумме трех вероятностей:

1) вероятности нахождения в системе n заявок в момент t, умноженной на вероятность того, что за время Δt в систему не поступает ни одной заявки и ни одна заявка не будет обслужена;

2) вероятности нахождения в системе $(n - 1)$ заявки в момент t, умноженной на вероятность поступления одной заявки за время Δt, и ни одна заявка не будет обслужена;

3) вероятности нахождения в системе $(n+1)$ заявки в момент $t+o$, где o — пренебрежимо малый интервал времени, умноженной на вероятность ухода одной заявки, при условии непоступления ни одной заявки.

Заметим, что

$$(1-\lambda\Delta t)(1-\mu\Delta t) = 1-\lambda\Delta t-\mu\Delta t+o(\Delta t);$$

$$\lambda\Delta t(1-\mu\Delta t) = \lambda\Delta t+o(\Delta t),\ \mu\Delta t(1-\lambda\Delta t) = \mu\Delta t+o(\Delta t). \quad (2.10)$$

Образуя разностное уравнение и переходя к пределу, получаем дифференциальные уравнения:

$$\begin{cases} \dfrac{dP_n(t)}{dt} = -(\lambda+\mu)P_n(t)+\lambda P_{n-1}(t)+\mu P_{n+1}(t),\ t\ge 1; \\ \dfrac{dP_0(t)}{dt} = -\lambda P_0(t)+\mu P_1(t). \end{cases} \quad (2.11)$$

Найдем выражение среднего числа заявок, находящихся в накопителе, и среднего *времени ожидания заявок* в накопителе для стационарного состояния при загрузке $\rho = \lambda/\mu < 1$, коэффициент ρ иногда называют траффик-интенсивностью. Поскольку он также представляет долю полного времени, в течение которого прибор не простаивает, его иногда называют *коэффициентом использования или загруженности.*

Приравняв производные по времени t к нулю, получим уравнения:

$$\begin{cases} (\lambda+\mu)p_n = \lambda p_{n-1}+\mu p_{n+1},\ n\ge 1; \\ \lambda p_0 = \mu p_1; \end{cases} \quad \begin{cases} (1+\rho)p_n = p_{n-1}+\rho p_{n+1},\ n\ge 1; \\ p_1 = \rho p_0. \end{cases} \quad (2.12)$$

Положим $n=1$, тогда $(1+\rho)p_1 = p_2 + p\rho_0$, $p_2 = \rho^2 p_0$, повторяя эти операции, имеем $p_n = \rho^n p_0$, причем:

$$\sum_{n=0}^{\infty} p_n = 1 \Rightarrow \sum_{n=0}^{\infty} \rho^n p_0 = 1 \Rightarrow p_0 \sum_{n=0}^{\infty} \rho^n = 1 \Rightarrow p_0 = 1-\rho.$$

Следовательно, получим, что $p_n = p^n(1-\rho)$ — геометрическое распределение. Среднее число заявок в системе равно

$$E\{x_n\} = l_n^S = \sum_{n=1}^{\infty} np_n = (1-\rho)\sum_{n=1}^{\infty} np^n = \rho/(1-\rho), \quad (2.13)$$

$$D\{x_n\} = \sum_{n=1}^{\infty}(n - l_n^S)^2 p_n = \sum_{n=1}^{\infty} n^2 p_n - (\rho/(1-\rho))^2 = \rho/(1-\rho) + 2\rho^2/(1-\rho)^2.$$

Среднее число заявок, находящихся в накопителе:

$$l_n^{\text{н}} = \sum_{n=1}^{\infty}(n-1)p_n = \rho^2/(1-\rho). \qquad (2.14)$$

Среднее время ожидания заявок в накопителе:

$$E\{t_n\} = \frac{l_n^{\text{i}}}{\lambda} = \rho^2/(\lambda(1-\rho)). \qquad (2.15)$$

Среднее время пребывания в системе

$$E\{w\} = 1/\mu(1-\rho) = 1/\mu - \lambda. \qquad (2.16)$$

Сведем основные операционные характеристики рассматриваемой системы с дисциплиной FCFS (FIFO) в табл. 2.2. Следует обратить внимание, что при возрастании коэффициента использования такие параметры как число заявок в системе, длина очереди, время пребывания в системе начинают быстро возрастать. При заданной скорости обслуживания μ, когда коэффициент занятости невелик, основная доля среднего времени пребывания заявки в системе связана только с процедурой обслуживания, при возрастании интенсивности входного потока большая часть времени пребывания заявки в системе обусловлена ожиданием обслуживания.

Таблица 2.2. *Операционные характеристики СМО*

ρ	1 − ρ	ρ/(1 − ρ)	ρ²/(1 − ρ)	μ = 10			μ = 20		
				λ	E_w	E_{tn}	λ	E_w	E_{tn}
0.1	0.911	10.11	0.011	111	110.11	0.01	12111	0.06	0.01
0.3	0.711	10.43	10.1311	113	110.14	0.04	16111	0.07	0.02
0.5	0.511	11.01	10.5111	115	10.2	0.11	10111	0.11	0.05
0.7	0.311	12.33	1.631	117	110.33	0.23	14111	0.17	0.12
0.8	0.211	4.0	3.211	118	10.5	0.41	16111	0.25	0.21
0.9	0.111	9.0	8.111	119	11	0.91	18111	0.51	0.45
10.95	0.051	19.01	18.0511	9.5	21	1.91	19111	111	0.95
10.99	0.011	99.01	98.011	9.9	1011	9.91	19.81	511	4.95
0.999	0.001	999.0	998.01	119.99	100	99.911	19.98	50111	49.951

Рассмотрим конкретный пример использования табл. 2.2. Пусть дискрета времени равна 1 ч, а $\rho = 0{,}8$, при этом прибор простаивает в среднем 0,2 ч (12 мин), а среднее количество требований в системе равно 4. При $\mu = 10$ (скорость обслуживания равняется 10 ед./ч) средняя продолжительность пребывания заявки в системе равняется 0,5 (30 мин), а пребывание в очереди из них занимает 24 мин.

Возможны ситуации, когда длина очереди ограничена. Если в СМО не может быть более L заявок, то длина очереди ограничена величиной $L-1$ и любая заявка сверх этого значения теряется и статистическое равновесие в этом случае достигается при любом значении ρ [1].

2.3.4. Многоканальное обслуживание с пуассоновским входным потоком и экспоненциальным распределением длительностей обслуживания

Очевидно, что в большинстве случаев СМО являются многоканальными, символика Кендалла в этом случае имеет вид M/M/L, где входной поток имеет вид $f(t) = \lambda e^{-\lambda t}$, поток обслуживаний $g(t) = \mu e^{-\mu t}$, а число каналов равно L. Учтем, что режим функционирования любого канала не влияет на режимы функционирования других каналов, длительности обслуживания являются случайными величинами. Уравнения в конечных разностях аналогичны уравнениям (2.11). Решение системы уравнений приводится без доказательств:

$$P_n = \rho^n / n!\, P_0 \text{ при } L > n \geq 0; \tag{2.17}$$

$$P_n = \rho^n / L!\, L^{n-L} P_0 \text{ при } n \geq L, \tag{2.18}$$

где $\rho = \mu / \lambda$;

$$P_0 = \frac{1}{\sum\limits_{j=0}^{L-1} \frac{\rho^j}{j!} + \frac{\rho^L}{L!(1-\rho/L)}}. \tag{2.19}$$

Установившийся режим функционирования СМО, определяемой выражениями (2.17) – (2.19), достигается при условии $\lambda < \mu L$. Когда P_n определено, то большинство операционных характеристик вычисляется с помощью элементарных алгебраических операций [1].

Глава 3
ПРИНЦИПЫ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ

§ 3.1. ОБЩИЕ ПРЕДСТАВЛЕНИЯ

В настоящей главе будут рассмотрены некоторые основополагающие принципы построения любой имитационной модели, не обязательно реализуемой с помощью ЯИМ GPSS/H. В табл. 3.1 в качестве иллюстративного примера приведена классификация некоторых известных ЯИМ [4].

С появлением персональных ЭВМ интенсивно развиваются специализированные системы моделирования для этих типов компьютеров. Часто они представляют собой развитие известных систем моделирования для универсальных ЭВМ, например: GPSS, SIMSCRIPT II.5, SIMULAP, MiniDYNAMO, MicroDYNAMO/DOS, SLAM II/PC. В книге Е. Киндлера «Языки программирования» приведен каталог наиболее известных зарубежных систем ИМ, включающий 200 наименований. Некоторые из этих систем вошли в табл. 3.2. В таблице указаны, в основном, системы, предназначенные для использования на IBM-совместимых универсальных (MF) и персональных (PC) ЭВМ. Дополнительно указываются требуемый транслятор (с базового языка программирования), операционная система, а также некоторые

Таблица 3.1. *Классификация ЯИМ по способу имитации*

Тип модели	Способ имитации	Примеры ЯИМ
Дискретный	Событийный Просмотр активностей Процессный Транзактный	SIMSCRIPT SMPL, FORSIM SIMPULA GPSS
Агрегативный	Агрегатный	АИС, САПАС
Непрерывный	Аппроксимация дифференциальных уравнений	DYNAMO, MIMIC
Непрерывно-дискретный	Комбинированный (возможны различные комбинации вышеперечисленных способов имитации)	SLAM, НЕДИС
Сетевой	Имитации сетевых моделей	SIMMET, NETWORK

Таблица 3.2. Классификация ЯИМ

Название	Тип ЭВМ, транслятор	Примечания
Дискретные модели		
GPSS/PC	PC	Интерактивная графика и анимация
GPSS/H	PC	Версия системы GPSS для ПЭВМ
SIMSCRIPT II.5	PC, MF	Параметрическое моделирование
SIMULA	PC, MF	
SIMULAP	PC, MS DOS, Windows	
MIC-SIM	PC, PS/2	Графика и анимация
OPTIC	PC	
PC-SOL 4.0	PC, Паскаль 4.0	
Непрерывные модели		
DYNAMO III /F+/ 370	MF, Фортран	Нелинейные модели
MicroDYNAMO	PC, 16-разрядные, PC	
MiniDYNAMO		
ProDYNAMO+		
CSSL-IV	PC	Дополнительные возможности оптимизации моделируемых систем
ISIM	PC, MF	
OPT ISIM	Фортран-77	
SYSL/M	PC, Фортран	
ACSL	PC, Фортран	
Сетевые модели		
SIMNET	PC, Фортран-77	–
Network II.5	PC, MF, VAX	
BEST-NETWORK	MF, Фортран	
Непрерывно-дискретные модели		
SLAM II	MF, Фортран	Комбинированные модели дискретного, непрерывного и сетевого типа
SLAM II/PC	PC	
MicroPASSIM	PC, Паскаль	Расширенные возможности статистического анализа; ориентация на производственные системы; интерактивная графика и анимация
PASION	PC, Паскаль	
SIMAN + Cinema	PC, Фортран-77	
Динамические системы (ДС)		
ENPORT-7	MF, Фортран	Нелинейные ДС
TSIM	VAX	
LSMP	PC	
DSL/VS	MF, Фортран, Фортран-77, ПЛ /1	Линейные ДС

Продолжение табл. 3.2

Название	Тип ЭВМ, транслятор	Примечания
Системы реального времени		
NET Real Time	PC	Ориентация на моделирование автоматизированных производств
DATE-INTER-ACTIVE	VAX	Непрерывные модели; интерактивная графика
Системы, ориентированные на производственные процессы и контроль качества		
GEMS-II	PC, Фортран-77	Сетевые модели
MAST	PC, Фортран-77	Анализ производственных процессов
SIMPLE	PC	
SPAR	PC	Оценка возможности роста производства
XCELL+	PC	Быстрое построение прототипов производственных систем средствами интерактивной графики
SIMIS III	PC, MF	Моделирование производственных и материальных потоков
Process Quality Simulator	PC	Моделирование систем управления качеством продукции
Системы моделирования экономических процессов		
LIBRA	PC	Построение моделей равновесия экономических систем
MULTISIM	DEC-20, Simula	Многоуровневые модели рынка
Моделирование вычислительных и коммутационных систем (КС)		
CIRCUITS	Macintosh	Моделирование мощных вычислительных систем; интерактивная графика
Моделирование робототехнических систем		
ROSCAD	Silicon, Sun	Графическое моделирование в реальном времени
ROSY	VAX	

Окончание табл. 3.2

Название	Тип ЭВМ, транслятор	Примечания
Системы статистического моделирования и обработки данных		
СТАТМОД	PC	Статистическое моделирование случайных величин, векторов, процессов
СТАН	PC	Статистический анализ результатов экспериментов

особенности или возможности систем (в графе «Примечания»). По своему назначению системы разбиты на разделы. Первые четыре раздела включают ЯИМ общего назначения, ориентированные на определенный тип модели. В последующих разделах системы объединены по типу предметной области либо условиям применения.

Таблицы 3.1, 3.2 дают общее представление о системном мире ЯИМ, однако существуют принципы, одинаковые для всех типов моделей:

— понятие о модельном времени;

— структура процесса и способы представления параллельно происходящих в реальной системе событий в виде квазипараллельных событий при реализации на компьютере;

— генерирование базовых случайных величин (чисел, векторов, процессов).

Все упомянутые принципы будут рассмотрены в последующих параграфах настоящей главы.

§ 3.2. МОДЕЛЬНОЕ ВРЕМЯ

Как уже упоминалось (§ 1.5), содержание любой задачи ИМ получается в результате реализации прогона или ряда прогонов модельного файла, получения и обработки собранной статистики и принятия решений на основе статистического анализа. Длительность испытаний зависит либо от заданной статистической точности, либо от заданного числа реплик в одном прогоне, либо от времени рассмотрения функционирования реальной системы (см. гл. 8). В связи со сказанным необходимо четко понимать, какие времена рассматриваются при ИМ, и представлять их различие.

• $T_р$ — реальное время функционирования исследуемой системы S, которое может быть очень большим, например, 10^n лет при иссле-

довании космогонических процессов, либо, наоборот, очень малым — 10^{-n} с при исследовании процессов, происходящих в микромире.

• $T_{\text{и}}$ — машинное время имитации, отражающее затраты ресурса времени ЭВМ на организацию ИМ. В случае использования суперкомпьютеров, производительность которых превышает сотни гигафлоп, минута машинного времени может стоить несколько тысяч долларов. Это ограничение должно учитываться создателями ИМ и далее не рассматривается.

• $T_{\text{м}}$ — модельное время (МВ), используемое в ИМ. Оно может быть сжато при исследовании процессов макромира или растянуто при оперировании со сверхбыстрыми процессами в реальной системе. Кроме того, именно МВ позволяет избежать сложностей моделирования поведения реальной системы. Так, в реальной системе события могут происходить одновременно в разных компонентах системы; в обычных, не мультипроцессорных, ЭВМ параллельные события воплотить нельзя. Модельное время позволяет синхронизовать все события и реализовать квазипараллелизм (см. § 3.2). При создании ИМ задание временной дискреты модельного времени является обязательным условием до начала процесса ИМ. *Естественно, что разные значения времени процессов обязательно должны быть выражены в едином масштабе временной дискреты модельного времени.* Так, например, если временная дискрета задана в минутах, другие временные периоды: часы, сутки, годы — должны быть также представлены в минутах.

Введем обозначение временного интервала моделирования системы S (интервала МВ)

$$\tau = [t_0, T_{\text{i}}],$$

где t_0 — время начала моделирования (обычно полагают $t_0 = 0$); $T_{\text{м}}$ — время окончания моделирования; $t \in \tau$ — текущее значение МВ.

Построение модели системы S начинается с определения параметров системы и переменных, определяющих процесс функционирования системы.

1. ***Параметры системы*** $\theta_1, \theta_2, \ldots, \theta_m$ — это характеристики системы, остающиеся постоянными на всем интервале моделирования τ. Если значения $\{\theta_i\}$ определены на некотором множестве Θ

$$\theta = (\theta_1 \ldots \theta_m) \in \Theta \subset R^m,$$

то говорят, что имеется параметрическое семейство систем.

2. ***Множество переменных*** разбивают на два подмножества — независимых и зависимых переменных.

К *независимым* переменным отнесем следующие характеристики.

• Входные воздействия на систему (сигналы) $u_1, u_2, \ldots, u_{n1}$. Входные воздействия в момент $t \in T$ характеризуются вектором

$$\mathbf{u} = \mathbf{u}(t) = (u_1(t), \ldots, u_{n1}(t)) \in U \subset R^{n1}.$$

Среди $\{\mathbf{u}_i\}$ могут быть управляющие воздействия, например $u_1, u_2, \ldots, u_{n1'}$ $(n_{1'} \le n_1)$, а остальные $n_1 - n_{1'}$ воздействий — неуправляющие.

• Воздействия внешней среды: среди них могут быть контролируемые (наблюдаемые) и неконтролируемые (ненаблюдаемые), детерминированные и случайные воздействия. В момент $t \in T$ они характеризуются вектором

$$\mathbf{v} = \mathbf{v}(t) = (v_1(t), \ldots, v_{n2}(t)) \in V \subset R^{n2}.$$

• Переменные, характеризующие состояние системы: $x_1, x_2, \ldots, x_{n3}$. В отличие от $\{\theta_i\}$ состояния $\{x_i\}$ характеризуют свойства системы, изменяющиеся во времени. Состояние системы в момент времени $t \in T$ описывается вектором

$$\mathbf{x} = \mathbf{x}(t) = (x_1(t), \ldots, x_{n3}(t)) \in X \subset R^{n3},$$

где X — пространство состояний или фазовое пространство системы (множество возможных значений вектора $\mathbf{x}$). Если $t_1 < t_2 < \ldots$ — моменты изменения состояния системы, то последовательность $x(t_1)$, $x(t_2), \ldots$ называется фазовой траекторией системы.

К *зависимым* переменным отнесем следующие характеристики.

• Выходные характеристики (сигналы) системы $y_1, y_2, \ldots, y_{n1}$, определяемые в момент времени $t \in T$ вектором

$$\mathbf{y} = \mathbf{y}(t) = (y_1(t), \ldots, y_{n4}(t)) \in Y \subset R^{n4}.$$

• Выходные показатели системы $q_1, q_2, \ldots, q_k$ характеризуют ее цели (т. е. характеризуют достижение системой заданных или оптимальных величин) и образуют вектор

$$\mathbf{q} = \mathbf{q}(t) = (q_1(t), \ldots, q_k(t)) \in Q \subset R^k,\ t \in \tau.$$

Выходные показатели представляются в виде некоторых функционалов

$$q = q(t) = q_1(t) = (q_1(t), \ldots, q_k(t)) \in Q \subset R^k,\ t \in \tau.$$

При наличии в системе случайных факторов (например, случайных воздействий внешней среды) значения $\{q_i\}$ являются также слу-

чайными, при этом в качестве Q используют средние значения $\{Q_i\}$, определяемые соотношениями

$$Q_i = E\{q_i\},\ i = \overline{1,k},$$

где $E\{.\}$ — символ математического ожидания.

Связи между зависимыми и независимыми переменными показаны на рис. 3.1.

Процесс функционирования системы во времени описывается *операторными соотношениями* (заданными аналитически или алгоритмически) для состояний и выходных характеристик системы:

$$x(t) = F_1(u^{(t)}, v^{(t)}, \theta, t);$$

$$y(t) = F_2(u^{(t)}, v^{(t)}, x^{(t)}, \theta, t);$$

$$q(t) = F_3(u^{(t)}, v^{(t)}, x^{(t)}, y^{(t)}\theta, t),\ t \in \tau, \qquad (3.1)$$

где $u^{(t)}$ обозначают реализацию процесса $u(t)$ на отрезке $[0, t]$, аналогично обозначены $x^{(t)}, y^{(t)}$; $F_1(\cdot)$, $F_2(\cdot)$, $F_3(\cdot)$ обозначают соответствующие операторы, описывающие динамику зависимых и независимых переменных и показателей эффективности.

Зависимости (3.1) называются *законами функционирования системы* S; зависимость $y = y(t)$, $t \in \tau$ называется *выходной траекторией* системы, а зависимость $x = x(t)$, $t \in \tau$ — *фазовой траекторией*.

Под фазовой траекторией будем понимать отображение движения системы из одного состояния фазового пространства в другое в каждый момент МВ. В общем случае фазовое пространство при N компонентах системы S характеризуется $3N$ координатами и $3N$ импульсами, т. е. имеет размерность $6N$, что не изобразить графически. Простейшим случаем фазового пространства является *фазовая плоскость*.

Например, для маятника с трением имеем уравнение

$$d^2x/dt^2 + \eta/m\, dx/dt + k/mx = 0, \qquad (3.2)$$

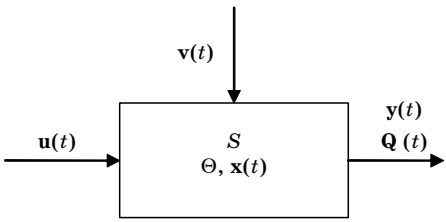


Рис. 3.1. *Связь между переменными*

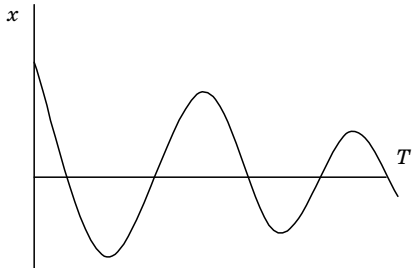


Рис. 3.2. *Зависимость координаты*

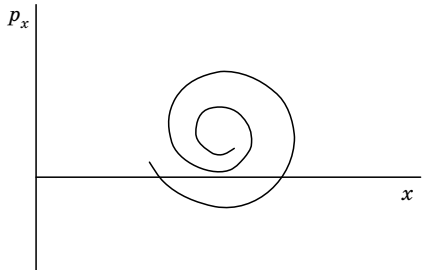


Рис. 3.3. *Фазовая траектория*

где x — координата; η — сила вязкого трения; m — масса маятника; k — жесткость.

Решением этого уравнения являются затухающие колебания. Зависимость координаты от времени приведена на рис. 3.2, а на рис. 3.3 — фазовая траектория в виде спирали, имеющая *аттрактором* (устойчивым состоянием) фокус в нуле.

Сделаем важное замечание, которое используется при моделировании: процесс моделирования всегда начинается в момент времени 0.0 или после точки 0.0. Если операндом С ОБ GENERATE задано смещение, то начало координат перемещается из точки 0.0 в точку, заданную операндом С.

Прежде чем дать описание способов представления модельного времени, рассмотрим взаимосвязь таких компонентов ИМ, как *событие* $A_j^{(i)}$, *действие* $d_j^{(i)}$, *локальное модельное время* $\tau_j^{(i)}$ (ЛМВ) и *процесс*. Под событием будем понимать изменение фазового состояния системы при совершении каких-то действий в течение локального момента времени.

В реальной системе процесс функционирования происходит в реальном времени, в модели — в течение МВ $T_м$. На рис. 3.4 показана взаимосвязь компонентов ИМ [4].

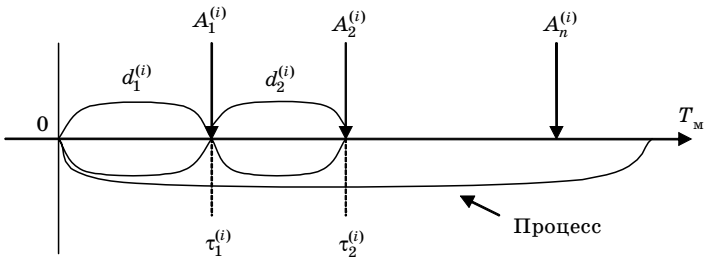


Рис. 3.4. *Взаимосвязь компонентов ИМ*

Очевидно, что возможны три способа изменения вектора состояний $\mathbf{x}(t)$ системы S:

1) в моменты наступления событий $\{A_j^{(i)}\}$;

2) в результате выполнения действий $\{d_j^{(i)}\}$, на что требуются затраты МВ $\{\tau_j^{(i)}\}$. Пара $\{d_j^{(i)}; \tau_j^{(i)}\}$ называется (i, j)-активностью (Р. Шеннон определил это сочетание как «работа»);

3) в результате выполнения хронологической последовательности событий и действий, называемых процессом.

Для осуществления изменения состояния системы одним из указанных способов необходимо каким-то образом управлять изменением МВ.

Существуют два способа формирования конечного множества моментов времени $T_м$, известных как принципы организации изменения МВ Δt и Δx.

Принцип Δt заключается в изменении МВ с фиксированным шагом Δt.

Принцип Δx заключается в изменении МВ при скачкообразном изменении вектора состояния $\mathbf{x}$ системы S на некоторую величину Δx ($\Delta x \neq 0$).

Для моментов времени t^* из множества $T_м$, сформированного по принципу Δx, справедливо

$$x(t^* + 0) = x(t^*) + \Delta x,\ t^* \in T. \qquad (3.3)$$

Для моментов времени из множества $[0, T] \setminus T_м$ вектор состояний изменяется непрерывно (либо остается неизменным).

Заметим, что скачкообразные изменения состояния системы S происходят при наступлении таких «особых» событий, как поступление управляющих сигналов и внешних воздействий, выдача выходных сигналов и т. п.

Приведем более строгое описание принципов Δt и Δx и поясним их особенности. Пусть система S состоит из N элементов: $A^{(1)}, \ldots, A^{(N)}$, поведение которых предполагается моделировать:

$$S = \{A^{(1)}, \ldots, A^{(N)})\}.$$

Для каждого элемента $A^{(i)} \in S$ $(i = 1, ..., N)$ определим ЛМВ $t^{(i)} \in [0, T_м]$. Поведение элемента $A^{(i)} \in S$ в течение интервала моделирования определяется некоторой последовательностью действий

$$g_1^{(i)}, g_2^{(i)}, \ldots, g_M^{(i)}, g_j^{(i)} \in G,\ j = 1, \ldots, M_i,$$

где G — множество всевозможных действий для элементов S. На множестве G будем выделять подмножество действий D: $D \subset G$, для вы-

полнения которых в ИМ требуется некоторое ненулевое модельное время.

Будем обозначать такие действия $d_1^{(i)}, \ldots, d_{m_i}^{(i)}$ ($d_j^{(i)} \in D \subset G$, $j = \overline{1, m_i}$, $m_i \le M_i$, $i = \overline{1, N}$), а интервалы МВ, затрачиваемые на выполнение этих действий, соответственно, $\tau_1^{(i)}, \ldots, \tau_{m_i}^{(i)}$. Последовательность $\{\tau_j^{(i)}\}$ ($j = \overline{1, m_i}$) является последовательностью случайных величин с заданными законами распределения $L\{\tau_j^{(i)}\}$, $i = \overline{1, N}$.

Действия $\{d_j^{(i)}\} \subset D$ приводят к наступлению в системе S *особых событий* $\{A_j^{(i)}\}$. События $\{C_j^{(i)}\}$, к которым приводят действия $\{g_j^{(i)}\} : \{g_j^{(i)}\} \in G / D$, не требующие затрат МВ, считаются *неособыми*.

Момент ЛМВ наступления события $A_j^{(i)}$ для $A_i \in S$ определяется по формуле

$$t_j^{(i)} = t + \tau_j^{(i)},\ j = 1, 2, \ldots,\ i = 1, \ldots, N, \qquad (3.4)$$

где t — текущее значение МВ; $\tau_j^{(i)}$ имитируется на ЭВМ в соответствии с законом распределения $L\{\tau_j^{(i)}\}$.

Состояние системы S в момент времени $t \in [0, T_м]$ определяется вектором состояния $\mathbf{x}(t) \in X \subset R^n$. Состояния системы в моменты наступления особых событий будем называть *особыми состояниями*, а состояние $\mathbf{x}(0)$ — *начальным состоянием* системы.

Для иллюстрации принципов Δt и Δx используем временную диаграмму (рис. 3.5).

Описание временной диаграммы.

Пусть число моделируемых элементов в S равно 2, т. е. $N = 2$, и $S = \{A^{(1)}, A^{(2)}\}$.

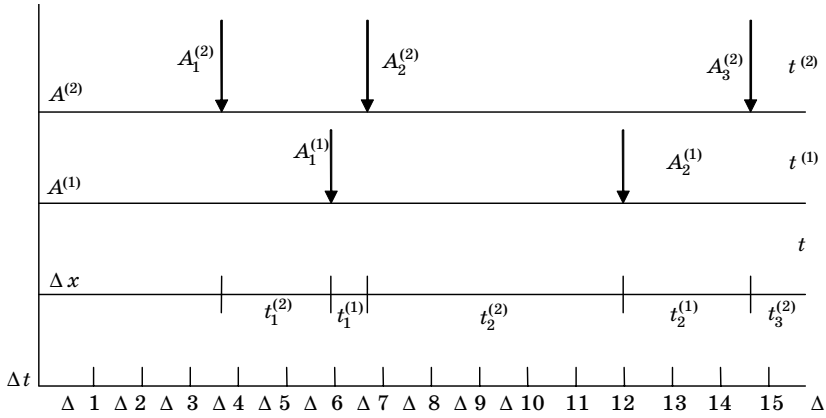


Рис. 3.5. *Временная диаграмма*

Временная диаграмма включает:

— временную ось ЛМВ $t^{(1)}$ для элемента $A^{(1)}$;

— временную ось ЛМВ $t^{(2)}$ для элемента $A^{(2)}$;

— временную ось модельного времени по принципу Δx;

— временную ось модельного времени по принципу Δt.

Временные оси будем помечать символами $A^{(1)}, A^{(2)}, \Delta t, \Delta x$.

Пусть в течение рассматриваемого интервала моделирования $[0, T_{\text{м}}]$ для элемента $A^{(1)}$ произошло 2 события: $A_1^{(1)}$, $A_2^{(1)}$ в моменты $t_1^{(1)}, t_2^{(1)}$, для элемента $A^{(2)}$ — 3 события: $A_1^{(2)}, A_2^{(2)}, A_3^{(2)}$ в моменты $t_1^{(2)}, t_2^{(2)}, t_3^{(2)}$.

Предположим, что хронологическая последовательность событий такова:

$$0 \le t_1^{(2)} < t_1^{(1)} < t_2^{(2)} < t_2^{(1)} < t_3^{(2)} \le T.$$

Принцип Δt.

В соответствии с принципом Δt изменение модельного времени t происходит через промежутки времени, равные Δt, т. е. t в течение времени моделирования T принимает конечное множество значений:

$$T_a = \{0, \Delta t, 2\Delta t, \ldots, \upsilon\Delta t = T\}.$$

При этом событиям, которые попадают в интервал постоянства МВ $\delta_r = ((r-1)\Delta t, r\Delta t)$, $r = \overline{1, \upsilon}$, в ИМ присваивается один и тот же момент наступления: $t = r\Delta t$. Выбор величины Δt существенно влияет как на быстродействие ИМ, так и на точность аппроксимации системы S с помощью ИМ. Пусть Δt выбран таким, как указано на диаграмме (см. рис. 3.5), т. е. моменты наступления событий в S принадлежат следующим интервалам:

$$t_1^{(1)}, t_2^{(2)} \in \delta_6, t_2^{(1)} \in \delta_{10}, t_3^{(2)} \in \delta_{15}.$$

Это означает, что соответствующим событиям в ИМ будут присвоены следующие моменты наступления:

$$A_1^{(2)} \sim 3\Delta t,\ A_1^{(1)},\ A_2^{(2)} \sim 6\Delta t,\ A_2^{(1)} \sim 10\Delta t,\ A_3^{(2)} \sim 15\Delta t.$$

При этом фазовая траектория системы S с вектором состояний $x(t) \in X$ будет иметь вид

$$x(0), x(\Delta t) = x(2\Delta t) = x(0), x(3\Delta t) = x(t_1^{(2)}), x(4\Delta t) = x(5\Delta t) = x(3\Delta t),$$
$$x(6\Delta t) = x(t_2^{(2)}), x(7\Delta t) = x(8\Delta t) = x(9\Delta t) = (6\Delta t), x(10\Delta t) = x(t_2^{(1)}),$$
$$x(11\Delta t) = \ldots = x(14\Delta t) = x(10\Delta t), x(15\Delta t) = x(t_3^{(2)}).$$

На основании полученной фазовой траектории можно сделать следующие выводы относительно выбора Δt:

1) если Δt мало, то выполняется много лишних вычислений состояний системы в моменты, когда вектор $\mathbf{x}(t)$ не изменяется (за счет этого возрастает время выполнения ИМ);

2) даже при сравнительно малом значении Δt моменты наступления событий в системе (а следовательно, и моменты изменения состояния системы) не совпадают с моментами наступления событий в ИМ, поэтому фазовая траектория, построенная с помощью ИМ, на множестве $T \subset [0, T_{\text{м}}]$ не совпадает с фазовой траекторией системы S.

Принцип Δx.

В соответствии с принципом Δx изменение модельного времени происходит в моменты наступления событий или, что то же самое, в моменты особых состояний, т. е. для нашего примера

$$T_{\text{i}} = \left\{0, t_1^{(2)}, t_1^{(1)}, t_2^{(2)}, t_2^{(1)}, t_3^{(2)} \le T_{\text{i}}\right\},$$

а фазовая траектория, построенная с помощью ИМ, будет совпадать на множестве $T_{\text{м}} \subset [0, T_{\text{м}}]$ с фазовой траекторией системы S:

$$x(0), x(t_1^{(2)}), x(t_1^{(1)}), x(t_2^{(2)}), x(t_2^{(1)}), x(t_3^{(2)}).$$

Приведем более строгие формулировки правил изменения МВ по принципам Δt и Δx.

Пусть $t^* < T_{\text{м}}$ — некоторый момент особого состояния системы S;

r_i — число событий, произошедших с элементом $A^{(i)} \in S$ до момента t^* включительно ($i = 1, \ldots, N$)′; $t_{r_i}^{(i)}\left(t_{r_i}^{(i)} \le t^*\right)$ — момент наступления последнего для элемента $A^{(i)}$ события до момента t^* включительно; $t_{r+1_i}^{(i)} > t^*$ — момент наступления ближайшего после r_i будущего события; $r = \sum_{i=1}^{N} r_i$ — общее число событий в момент t^*; t^{**} и τ^{**}— моменты ближайших будущих событий в ИМ, вычисленные по принципам Δt и Δx соответственно.

Модельное время t в ИМ можно рассматривать как функцию от числа событий, происходящих в ИМ. Очевидно: $t(r) = t^* < T_{\text{м}}$, $r = 0, 1, 2, \ldots$,

$$t(r+1) = t^{**} = \min\left\{t_{r_1+1}^{(1)}, \ldots, t_{r_N+1}^{(N)}, T_{\text{м}}\right\} = t(r) + \min\left\{\tau_{r_N+1}^{(N)}, T_{\text{м}} - t(r)\right\}; \quad (3.6)$$

$$t(r+1) = \tau^{**} = r\Delta t, \text{ если } \quad t^{**} \in \delta_r \ (r = \overline{1,\upsilon}, \upsilon\Delta t = T_{\text{м}}), \quad (3.7)$$

где $\{t_j^{(i)}\}$, $\{\tau_j^{(i)}\}$ определяются соотношением (3.6). Заметим, что моменты $t^{**} = T_м$ и $\tau^{**} = \upsilon\Delta t$ (если $T \in \delta_\upsilon$) являются моментами завершения моделирования. Правила (3.6) и (3.7) называются правилами изменения *модельного времени* по принципам Δt и Δx соответственно.

На практике отдается предпочтение принципу Δx. Принцип Δt используется лишь в случаях, когда:

1) события $\{A_j^{(i)}\}$ таковы, что $t_j^{(i)} - t_{j-1}^{(i)} \cong$ const на всем интервале моделирования $T_м$, и, следовательно, можно подобрать интервал Δt изменения МВ, обеспечивающий минимальную погрешность аппроксимации (например, для разностных уравнений);

2) событий очень много и они появляются группами. В этом случае за счет групповой обработки событий $\{A_j^{(i)}\}$, попавших внутрь очередного шага изменения Δt, удается уменьшить затраты машинного времени.

В большинстве практически важных случаев события $\{A_j^{(i)}\}$ наступают через случайные интервалы времени $\{t_j^{(i)}\}$. Поэтому способ задания шага до следующего события экономичнее (в смысле затрат машинного времени) и точнее (в смысле точности аппроксимации) фазовой траектории способа фиксированного изменения МВ. В связи с реализацией такого способа представления МВ при моделировании на GPSS/H в англоязычной литературе он называется моделированием дискретных событий (Discrete Event Simulation). Такое название никоим образом не противоречит выбранному типу концептуальной НВ-модели, а лишь отражает способ представления модельного времени.

§ 3.3. СПОСОБЫ СОЗДАНИЯ КВАЗИПАРАЛЛЕЛИЗМА ПРИ ИМИТАЦИОННОМ МОДЕЛИРОВАНИИ

По Р. Шеннону, каждая ИМ представляет собой комбинацию компонентов, параметров, переменных, функциональных зависимостей, ограничений, целевых функций.

Под *компонентами* будем понимать составную часть исследуемой системы, величина которой зависит от степени детализации рассмотрения системы (элемент, блок, устройство, подсистема).

Параметры выбираются исследователем, и обычно в течение прогона, а иногда и в процессе всего ИМ они сохраняются неизменными.

Переменные бывают внешние и внутренние.

Функциональные зависимости связывают переменные и параметры внутри компоненты или между компонентами.

Ограничения представляют собой пределы значений переменных или условия их изменений.

Целевая функция представляет отображение целей или задач системы и выражается в терминах, интересных исследователю (мощность, быстродействие, экономические показатели и т. д.).

Для рассмотрения особенностей ИМ воспользуемся примером И. В. Максимея [8], который представляется наиболее наглядным для иллюстрации связи между процессами в реальной системе и в модели (рис. 3.6). Пример подчеркивает то важное обстоятельство МВ, что при моделировании надо осуществлять создание квазипараллелизма одним из способов, рассмотренных ниже.

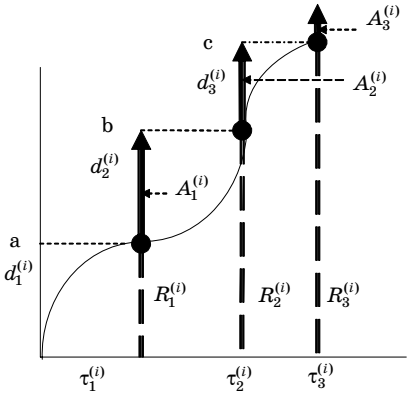


Рис. 3.6. *Процессы в модели и системе: $A_j^{(i)}$ — события в системе и модели; $R_j^{(i)}$ — реальные действия в системе; $d_j^{(i)}$ — имитирующие действия в модели; $\tau_j^{(i)}$ — отрезки времени между событиями*

Пример. Объектом имитации является движение космической ракеты. В режиме запуска можно выделить последовательность реальных действий от $R_1^{(i)}$ на стартовой площадке до $R_3^{(i)}$ — сброса второй ступени. Движение множества ракет представляет собой систему, а каждая i-я ракета является компонентой. В результате выполнения действий возникают события $A_j^{(i)}$.

Отличие $R_j^{(i)}$ от $d_j^{(i)}$ определяет уровень детализации модели и порождает ошибки имитации реальной системы. Очевидно, что каждое действие $d_j^{(i)}$ описывается алгоритмом $\mathrm{AE}_j^{(i)}$. Тогда при переходе от события к событию реализуется действие при неизменном значении времени, а потом изменяется время на $\tau_j^{(i)}$. В принципе, возможно и обратное поведение, т. е. вначале изменяется время, а затем реализуется действие посредством алгоритма. Для реальной системы фазовая траектория запишется в виде $0 - A_1^{(i)} - A_2^{(i)} - A_3^{(i)}$; в имитационной модели — в виде $0 - a - A_1^{(i)} - b - A_2^{(i)} - c - A_3^{(i)}$ или $\tau_1^{(i)} - A_1^{(i)} - \tau_2^{(i)} - A_2^{(i)} - \tau_3^{(i)} - A_3^{(i)}$.

В любом случае мы имеем активность $\mathrm{AK}_j^{(i)}(d_j^{(i)}, \tau_j^{(i)})$ как некую молекулу, содержащую описание *алгоритма*, приводящего к действию, и *оператор* изменения временной координаты $\mathrm{I}_{tj}^{(i)}$ (временной модификатор).

На рис. 3.7 дается представление о различиях между исследуемой системой и ее моделью. Каждому компоненту системы соответствует аналогичный компонент модели, причем каждый компонент может

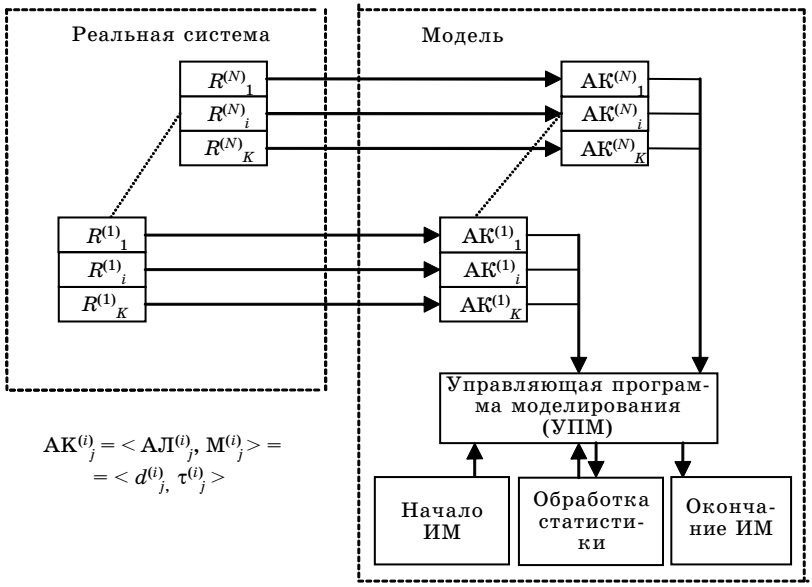


Рис. 3.7. *Отличия между системой и ее моделью*

включать ряд элементов со своими действиями, которым в модели соответствую алгоритм и модификатор времени.

Общее управление ведется с помощью *УПМ*, связывающей запуск алгоритмов, временных модификаторов, проверку условий имитации, обработку статистики, окончание испытаний, выдачу результатов. Таким образом, описание ИМ разрастается в объеме по сравнению с описанием системы. ИМ состоит из двух частей: первая часть является переменной, объектно-ориентированной и создается исследователем с помощью определенных процедур; вторая часть практически инвариантна к виду исследуемой системы и представляет собой реализацию процедур синхронизации модели, запуска, сбора статистики и завершения имитации.

В зависимости от состава алгоритмов, связей между компонентами, целей и задач моделирования можно выбрать различные способы представления квазипараллелизма, а соответственно, способа имитации (приставка «квази» отражает последовательный характер обслуживания событий в ИМ, одновременно возникающих в разных компонентах реальной системы).

В табл. 3.3 приведены типы описания ИМ и способы организации квазипараллелизма в ИМ.

Одну и ту же систему можно представить одним из способов табл. 3.3, но ИМ на их основе отличаются размерами и количеством

Таблица 3.3. Способы организации квазипараллелизма

Тип описания ИМ	Способ организации квазипараллелизма
Активностями	Просмотр активностей в УПМ
Событиями	Составление расписания событий
Транзактами	Управление обслуживанием транзактов
Агрегатами	Управление агрегатами
Процессами	Синхронизация процессов

ресурсов, затрачиваемых на их создание, испытания и использование. Рассмотрим особенности и принципы организации квазипараллелизма в ИМ каждым из указанных способов.

Просмотр активностей

Система, описываемая этим способом, характеризуется следующим:

- все действия для элемента $A_j^{(i)}$ системы S различны и приводят к наступлению разных событий;
- каждое действие $d_j^{(i)}$ характеризуется набором условий его выполнения, представляемых алгоритмически;
- времена выполнения действий являются случайными величинами с известными законами распределения.

Имитационная модель описывается в виде двух частей: множества активностей $\left\{\mathrm{AE}_j^{(i)}\right\}$ и набора процедур выполнимости условий инициализации активностей. (Инициализация — передача управления от УПМ на выполнение алгоритма данной активности.) Затем происходит модификация временной координаты $\mathrm{M}_{jt}^{(i)}$. Таким образом, ИМ представляет собой чередование выполнения алгоритмов активностей, операторов модификации временной координаты t_i и алгоритма УПМ. При этом способе приходится проверять много условий попадания активностей в список инициализированных, поэтому затраты машинного времени весьма велики. Этот способ применяется, когда важно оценить влияние действия на поведение системы.

Составление расписания событий

Используется для систем, характеризующихся следующим:

- множество событий разбивается на небольшое число типов;
- определяются условия перехода от одного события к другому для всех типов событий;
- для каждого типа событий определена последовательность действий, приводящая к изменению состояния системы;
- интервалы времени между последовательными наступлениями событий — случайные величины с известными законами распределения.

Имитационная модель также состоит из двух частей: множества активностей и набора процедур проверки появления событий и инициализации соответствующих активностей. Объединение нескольких активностей в группу существенно сокращает размеры начальных циклов и уменьшает расходы на организацию ИМ. Большим недостатком этого способа является то, что из-за объединения активностей различных подсистем в процедуре событий описание ИМ может потерять сходство с реальной системой. Так, в одной процедуре могут обслуживаться активности, не связанные друг с другом, но приводящие к одним событиям, что затрудняет анализ результатов ИМ.

Транзактный способ

При этом способе действия подсистем одинаковы, активности лишь корректируют значения временных координат. Кроме того, существует зависимость действий друг от друга, которую можно представить в виде СМО. Инициаторами появления событий являются заявки (транзакты) на обслуживание. В ИМ должна быть схема рождения транзактов, их перемещения, уничтожения обслуженных. Для описания ИМ создается фиксированный набор операторов, в GPSS/H совместное число всех операторов (блоков управления и описания немногим превышает 100) и УПМ сканирует списки транзактов, инициализирует блоки, сдвигает модельное время. Событием в ИМ является момент инициализации транзакта, в результате транзакт выступает в роли активности. (Поскольку далее будет рассматриваться только этот способ создания квазипараллелизма, то подробное описание порядка действия при ИМ приводится в гл. 4, 5).

Агрегатный способ

При описании агрегата (под агрегатом понимается объединение ряда устройств для унификации концептуального описания [4]) применим любой из рассмотренных способов, так что выделение агрегатного способа достаточно условно. Функцией УПМ является проверка условий перехода агрегата в одно из особых состояний и моделирование выходных сигналов агрегата. Агрегатный способ удобен при интерпретации системы, но требует больших затрат машинного времени.

Процессный способ

Используется при моделировании систем с различными компонентами, события в которых возникают в различное время, и у каждого компонента своя последовательность действий. При большой степени детализации описания структуры реальной системы и ее модели хорошо совпадают. При этом существует связь не только между компонентами, но и между алгоритмами. Этот метод обычно используют при проектировании новых систем большой размерности, он сочета-

ет в себе черты событийного способа и просмотра активностей. Однако метод требует создания специализированных языков.

В пособии не проводится сравнительный анализ методов создания квазипараллелизма, так как при дальнейшем изложении рассматривается только транзактный метод, для более развернутого изучения этих методов необходимо обратиться к специальной литературе, например [4].

§ 3.4. МЕТОДЫ ИМИТАЦИИ СЛУЧАЙНЫХ ЧИСЕЛ

3.4.1. Исторический экскурс

Любой процесс ИМ прежде всего зависит от качества случайных чисел, векторов или функций, вводимых в модель системы. В нашем случае они должны корректно представить входной поток заявок и поток их обслуживания (см. § 2.3). Любое случайное число или вектор, попадающий на вход модели системы, будем называть случайным элементом Θ^* (СЭ). Во всех случаях СЭ Θ^* должен удовлетворять двум основным принципам:

1) сходство между оригиналом Θ^* и его моделью Θ состоит в совпадении вероятностных законов распределения или числовых характеристик;

2) всякий СЭ Θ конструируется как некая борелевская функция на основе БСВ, генерируемых тем или иным путем.

Первым способом получения БСВ можно назвать попытку У. Госсета (псевдоним Стьюдент) в 1908 г., когда он использовал семизначные телефонные номера, отбрасывая первые три цифры, которые не являлись случайными по определению.

Предположим, получалась комбинация из 16 чисел 2127210128891172, если требовалось создать равномерно распределенную (РР) 8-разрядную БСВ, то эта комбинация давала только две РР БСВ: $\Theta_1 = 0.21272101$ и $\Theta_2 = 0.28891172$.

Первая *таблица случайных чисел* была разработана в 1927 г. Л. Типпетом и содержала 41600 БСВ, достаточных для создания 5200 8-разрядных БСВ. Кульминацией таблиц БСВ явился выпуск в 1955 г. корпорацией RAND одного миллиона БСВ, достаточных для получения 125000 8-разрядных БСВ. Очевидно, что для проведения даже простого имитационного эксперимента такого количества БСВ явно недостаточно. Кроме того, хранение и воспроизведение табличных БСВ весьма сложно и длительно, поэтому табличные датчики используются только для назначения номеров телефонов, автомашин и практически не используются при ИМ на ЭВМ.

Следующим возможным вариантом датчиков БСВ является *физический датчик* — специальное радиоэлектронное устройство, являющееся приставкой к ЭВМ, выходной сигнал которого имитирует БСВ. Он состоит из источника флуктуационного шума (например, «флуктуационно шумящей» радиолампы), значение которого в произвольный момент времени является случайной величиной $\eta \geq 0$ с плотностью $p_\eta(y)$, и нелинейного преобразователя

$$\alpha = \{\eta\}_\Delta / \Delta, \tag{3.8}$$

где $\{\eta\}_\Delta = \eta - \Delta[\eta/\Delta]$ — дробная часть числа η относительно заданного $\Delta > 0$ ($[\eta]$ — целая часть числа η).

Исследуем вероятностные свойства α. По правилам функционального преобразования случайных величин, из (3.8) следует, что плотность распределения α

$$p_\alpha(x) = \begin{cases} \sum\limits_{j=0}^{\infty} p_\eta((x+j)\Delta)\Delta, \ 0 \leq x < 1; \\ 0, \ x \notin [0,1). \end{cases} \tag{3.9}$$

Будем предполагать, что $p_\eta(y)$ непрерывно дифференцируема и

$$p_\eta(0) < \infty, p_\eta(\infty) < \infty. \tag{3.10}$$

Применим к j-му слагаемому из (3.10) в окрестности точки $y = j\Delta$ линейную формулу Тейлора с остаточным членом Лагранжа:

$$p_\alpha(x) = \begin{cases} \sum\limits_{j=0}^{\infty} \Delta p_\eta(j\Delta) + x\Delta \sum\limits_{j=0}^{\infty} \Delta p'_\eta((\theta x + j)\Delta)\Delta, \quad 0 \leq x < 1; \\ 0, \quad x \notin [0,1), \end{cases} \tag{3.11}$$

где $0 < \theta < 1$. При $\Delta \to 0$ по свойствам плотности распределения

$$\sum_{j=0}^{\infty} \Delta p_\eta(j\Delta) \to \int_0^{\infty} p_\eta(y)\mathrm{d}y,$$

$$\sum_{0}^{\infty} \Delta p'_\eta((\theta x + j)\Delta) \to \int_0^{\infty} p'_\eta(y)\mathrm{d}y = p_\eta(\infty) - p_\eta(0),$$

поэтому из (3.9), (3.10) следует, что

$$p_\alpha(x) \to \begin{cases} 1, \ x \in [0,1); \\ 0, \ x \notin [0,1). \end{cases} \tag{3.12}$$

Таким образом, выбирая Δ достаточно малой величиной, удается получить БСВ α.

Недостатки физического датчика БСВ:

1) невозможность повторения некоторой ранее полученной реализации a (поскольку $P\{\alpha = a\} = 0$);

2) схемная нестабильность, приводящая к необходимости контролировать работу датчика при очередном его использовании.

По этим причинам на современных компьютерах физические датчики БСВ используются весьма редко.

Указанными недостатками не обладает *программный датчик* БСВ — это программа, служащая для имитации на ЭВМ реализации $a_1, a_2, \ldots$ БСВ. Он может быть получен из физического датчика БСВ введением обратной связи. Будем рассматривать функционирование датчика во времени и обозначим: η_t — случайную величину, подвергаемую преобразованию (3.8) в момент времени t; α_t — выходную величину датчика в момент t (случайное число). Источник флуктуационного шума в физическом датчике заменяется обратной связью

$$\eta_t = \psi\left(\alpha_{t-1}, \alpha_{t-2}, \ldots, \alpha_{t-p}\right), \qquad (3.13)$$

использующей p ранее полученных выходных значений датчика. В (3.13) $t = 1, 2, \ldots$, а $\alpha_0, \alpha_{-1}, \ldots, \alpha_{1-p}$ фиксируются заранее: $\alpha_i = a_i (i = 1 - p, 0)$ и называются исходными (стартовыми) случайными числами.

Согласно (3.8), (3.13):

$$\alpha_t = \frac{\{\eta_t\}\Delta}{\Delta} = \frac{\left\{\psi\left(\alpha_{t-1}, \alpha_{t-2}, \ldots, \alpha_{t-p}\right)\right\}\Delta}{\Delta} = \Phi\left(\alpha_{t-1}, \alpha_{t-2}, \ldots, \alpha_{t-p}\right). \qquad (3.14)$$

Рекуррентная формула (3.14) определяет последовательность *псевдослучайных чисел* $a_{1-p}, a_{2-p}, \ldots, a_0, a_1, \ldots, a_t, \ldots$. Термин «псевдослучайные» используется по следующим причинам:

1) по происхождению эти числа не случайные; они получаются по известному детерминированному закону (3.14);

2) при специальном выборе функции $\Phi(\cdot)$ по вероятностным характеристикам эти числа похожи на реализации независимых БСВ.

Отметим, что понятие случайности последовательности можно связать со сложностью моделирующего алгоритма и, в частности, со сложностью функции $\Phi(\cdot)$ в (3.14).

Программные датчики, или, как их чаще принято называть, генераторы случайных чисел, встроены непосредственно в ЯИМ и должны отвечать следующим основным условиям:

— скорость генерации БСВ;

— требования к памяти ЭВМ;

— количество и качество генерируемых независимых РР чисел.

Прогресс генераторов БСВ весьма стремителен, и в настоящее время существует много конкурирующих методов создания генераторов, равно как и самих типов программных генераторов. Не менее стремителен и прогресс ЭВМ, поэтому первые два условия давно перестали служить ограничениями, и на первый план выдвинулось требование количества и качества БСВ. Простые расчеты показывают, что для моделирования даже несложной системы может потребоваться несколько миллионов БСВ.

3.4.2. Принципы моделирования БСВ

Простейшим для моделирования на ЭВМ случайным экспериментом является эксперимент, заключающийся в бросании точки наудачу в промежуток [0, 1). Результатом этого эксперимента является координата точки. Математической моделью такого эксперимента является вероятностное пространство (Ω, F, P), где $\Omega = [0, 1)$ — пространство элементарных событий (элементарное событие $\omega \in \Omega$ заключается в том, что координата брошенной точки равна ω); F — δ — алгебра, порожденная интервалами из δ; P — вероятностная мера, определенная для событий (подмножеств) $A \in F$ и совпадающая с мерой Лебега, так что для события $A = \{\omega : \omega \in [0, x)\}$

$$P(A) = P\{\omega \in [0,\ x)\} = x,\ x \in [0, 1). \tag{3.15}$$

Базовой случайной величиной на (Ω, F, P) будем называть непрерывную случайную величину

$$\alpha = \alpha(\omega) = \omega, \tag{3.16}$$

равномерно распределенную на полуинтервале [0, 1).

Функция, распределенная БСВ, имеет вид

$$F_\alpha(x) = P\{\alpha < x\} = \begin{cases} 0, x \le 0; \\ x, 0 < x < 1; \\ 1, x \ge 1, \end{cases} \tag{3.17}$$

а плотность распределения определяется формулой

$$P_\alpha(x) = \begin{cases} 1,\ 0 \le x < 1; \\ 0,\ x \notin [0,\ 1). \end{cases}$$

Будем обозначать закон распределения $\alpha : R\,[0, 1]$.

Математическое ожидание БСВ (первый начальный момент)

$$\mu = E\{\alpha\} = 1/2;$$

дисперсия (второй центральный момент)

$$\sigma^2 = D\{\alpha\} = E\left\{(\alpha - \mu)^2\right\} = 1/12.$$

Наряду с простейшим экспериментом будем рассматривать составной случайный эксперимент, получающийся в результате r-кратного ($r \geq 1$) повторения независимо друг от друга простейших экспериментов. Результатом составного случайного эксперимента является последовательность из r независимых БСВ $\alpha_1, \ldots, \alpha_r$ таких, что

$$\alpha_i = \alpha_i(\omega) = \omega_i, \quad i = \overline{1, r}; \quad \omega = (\omega_1, \ldots, \omega_r) \in \Omega^r,$$

где ω_i — координата точки, брошенной наудачу в $[0, 1)$ в i-м простейшем эксперименте.

Совместная плотность распределения вероятностей $\alpha_1, \ldots, \alpha_r$

$$p_{\alpha_1,\ldots,\alpha_r}(x_1, \ldots, x_r) = \begin{cases} 1, x_i \in [0,1), \ i = \overline{1, r}; \\ 0 \text{ в противном случае.} \end{cases}$$

Согласно второму принципу моделирования случайных элементов, любой СЭ Θ представляется для некоторого натурального r в виде функции $f(.)$ от r независимых БСВ:

$$\Theta = f(\alpha_1, \ldots, \alpha_r).$$

Таким образом, задача моделирования произвольного СЭ Θ^* разбивается на две подзадачи:

1) моделирование на ЭВМ независимых БСВ $\alpha_1, \ldots, \alpha_r$;

2) нахождение функции $f(.)$ такой, чтобы СЭ Θ обладал требуемыми вероятностным законом распределения и числовыми характеристиками.

Поэтому моделирующий алгоритм состоит из двух блоков (рис. 3.8):

Для имитации одного и того же СЭ Θ^* может быть предложено несколько вариантов функциональных преобразований. Обычно пред-

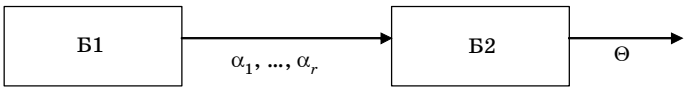


Рис. 3.8. *Моделирующий алгоритм БСВ: Б1 — блок моделирования БСВ (общий для всех Θ); Б2 — блок функционального преобразования f(.) БСВ (различный для различных законов распределения вероятностей)*

почтение отдается варианту $f(.)$, требующему меньших вычислительных затрат; для этого применяется понятие коэффициента использования БСВ.

Коэффициентом использования БСВ K назовем величину, обратную числу r базовых случайных величин, используемых для моделирования одной реализации случайного элемента Θ^*:

$$K = 1/r, \quad 0 < K \le 1.$$

Величина K является мерой вычислительных затрат на моделирование Θ^*. Чем меньше K, тем больше затраты. Целесообразно выбирать такую функцию $f(.)$, для которой K принимает наибольшее значение.

Очевидно, чтобы моделировать на ЭВМ случайные элементы с заданным вероятностным законом распределения, необходимо уметь моделировать БСВ. БСВ α является абсолютно непрерывной случайной величиной. Однако на ЭВМ приходится иметь дело с *дискретными случайными величинами* (ДСВ). Поэтому моделирование БСВ основано на аппроксимации непрерывной случайной величины α ДСВ α'. Опишем способ построения ДСВ α'.

Рассмотрим случай, когда представление целых неотрицательных чисел на ЭВМ осуществляется с помощью k двоичных разрядов (битов). Тогда $C = \{0, 1, 2^k - 1\}$ — множество 2^k неотрицательных целых чисел, представимых в ЭВМ. Определим на (Ω, Φ, P) ДСВ $\beta = \beta(\omega)$ следующим образом:

$$\beta = \beta(\omega) = \begin{cases} 0, 0 \le \omega \le 2^{-k}; \\ \vdots \\ i = i2^{-k} \le \omega < (i+1)2^{-k}, i \in C; \\ \vdots \\ 2^k - 1, 1 - 2^{-k} \le \omega < 2^k 2^{-k} = 1. \end{cases} \qquad (3.18)$$

Построение β проиллюстрируем с помощью рис. 3.9. Разобьем промежуток $[0, 1)$ на 2^k отрезков одинаковой длины 2^{-k}; для ω, попадающих в промежуток $[i2^{-k}, (i+1)2^{-k})$, полагаем $\beta = \beta(\omega) = i$, $i \in C$.

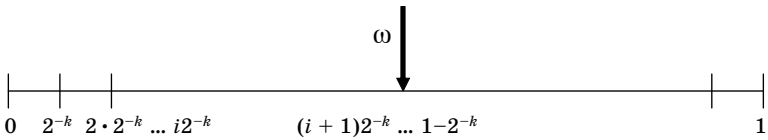


Рис. 3.9. *Построение* β

По построению распределение ДСВ β является равномерным на множестве C, т. е. все значения β равновероятны, действительно:

$$P\{\beta(\omega)=i\}=P\left\{\omega:\omega\in\left[i2^{-k},\,(i+1)2^{-k}\right)\right\}=2^{-k}. \tag{3.19}$$

Теперь перейдем от СВ β к искомой ДСВ α':

$$\alpha'=\alpha'(\omega)=\beta(\omega)/2^k=\beta(\omega)2^{-k}. \tag{3.20}$$

Согласно (3.19):

$$\alpha'=\begin{cases}0,\ 0\le\omega\le 2^{-k},\ (\beta=0);\\ \vdots\\ i2^{-k},\ i2^{-k}\le\omega<(i+1)2^{-k},\ (\beta=i),\ i\in C;\\ \vdots\\ 1-2^k,\ 1-2^{-k}\le\omega<1,\ (\beta=2^k-1),\end{cases}$$

т. е. от целочисленной ДСВ β мы перешли к ДСВ α' со значениями в [0, 1).

Очевидно, все возможные значения α' определяются множеством $C'=\{0,\,2^{-k},\,\ldots,\,1-2^{-k}\}$ и являются равновероятными:

$$P\{\alpha'(\omega)=i2^{-k}\}=P\{\beta(\omega)=i\}=2^{-k},\ i\in C,$$

т. е. закон распределения α' является равномерным на C'.

Точность аппроксимации α с помощью α' устанавливается с помощью леммы.

Лемма 3.1. Для СВ $\alpha=\alpha(\omega)$ и $\alpha'=\alpha'(\omega)$, определенных на (Ω, Φ, P) и имеющих вид (3.16), (3.17) соответственно, равномерное уклонение удовлетворяет выражению

$$\sup_{\omega\in\Omega}|\alpha'(\omega)-\alpha(\omega)|<2^{-k}. \tag{3.21}$$

Доказательство. Разобьем $\Omega=[0,1]$ на 2^k промежутков согласно рис. 3.9. Пусть $\omega\in\left[i2^{-k},(i+1)2^{-k}\right)$, $i\in C$. Тогда, согласно (3.16), справедливо представление

$$\alpha=\alpha(\omega)=\omega=i2^{-k}+\delta,\ 0\le\delta<2^{-k},\ i\in C,$$

а согласно (3.20):

$$\alpha'=\alpha'(\omega)=i2^{-k}.$$

Поэтому $|\alpha-\alpha'|=\delta<2^{-k}$, $\omega\in\Omega$. Отсюда заключаем справедливость (3.21).

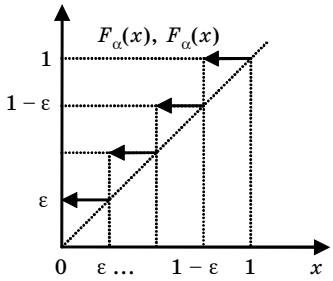


Рис. 3.10. *Функция распределения* $F_\alpha(x)$

Таблица 3.4. *Соотношения между дисперсиями величин* α *и* α'

k	2	3	5	10	15
$\left(\frac{D\{\alpha'\}}{D\{\alpha\}}\right)^{1/2}$	1.290	1.140	1.030	1.001	1.00

Из (3.21) следует, что если $k \to \infty$, то последовательность $\alpha'_k \to \alpha$ равномерна по $\omega \in \Omega$. Таким образом, случайная величина $\alpha' = \alpha'(\omega)$ является аппроксимацией для БСВ[1] $\alpha = \alpha(\omega)$; α' называется в связи с этим *квазиравномерной случайной величиной*. Ее функция распределения $F_{\alpha'}(x)$ показана на рис. 3.10 и аппроксимирует $F_\alpha(x)$ с точностью $\varepsilon = 2^{-k}$.

Между математическими ожиданиями величин α, α' справедливо соотношение

$$E\{\alpha'\} = 0{,}5(1 - 2^{-k}) \to E\{\alpha\} = 0{,}5,$$

представленное в табл. 3.4.

При достаточно больших значениях k (например, для ПЭВМ IBM PC AT 486, Pentium $k = 31$, $\varepsilon = 0.5 \cdot 10^{-9}$) величины α и α' отождествляют.

3.4.3. Методы построения программных генераторов

Существует ряд методов и алгоритмов, используемых при построении генераторов БСВ [4, 11]. Наиболее широкое распространение получили генераторы, построенные на использовании различных

[1] Напомним, что аппроксимация непрерывных случайных величин дискретными широко используется в теории вероятностей при построении интеграла Лебега.

модификаций метода вычетов (мультипликативный конгруэнтный (совпадающий) метод), который в разных модификациях используется в разных редакциях GPSS/H. В конце параграфа будут приведены параметры генераторов GPSS/H. Рассмотрим этот метод подробнее.

Любое следующее число образуется на основании выражения

$$x_{i+1} = ax_i + c \mod m \quad i = 1, 2, \ldots, \tag{3.22}$$

где a, m — положительные целые числа; c — целое число.

Исторически этот генератор строился из принципа

$$x_n = x^n \mod m. \tag{3.23}$$

Чтобы не возникало вычислительных проблем, (3.23) можно заменить

$$x_n = xx_{n-1} \mod m. \tag{3.24}$$

При $c = 0$ как раз и получается мультипликативный конгруэнтный генератор и генерируемые числа $U_i = x_i / m$. Методы проверки случайности генерируемых чисел будут рассмотрены в п. 3.5.3. Выражение (3.24) основывается на значении $m = 2^k$, где k — целое число, обычно равное длине машинного слова. Любые генерируемые БСВ характеризуются *периодом повторения* или *циклом генератора* $d = 2^{k-2}$ при нечетном x_0 (примем это без доказательства).

В качестве примера рассмотрим генератор $x_n = 3x_{n-1} \bmod 32\ (2^5)$.

При $c = 0$ [см. (3.22)] и $a = \pm 3 \pmod 8$ период этого генератора $d = 8^{5-2} = 8$. Приняв $x_0 = 1$, найдем числовую последовательность, создаваемую генератором:

$x_0 = 1$	*00001*
$x_1 = 3$	*00011*
$x_2 = 9$	*01001*
$x_3 = 27$	*11011*
$x_4 = 81 \bmod 32\ (81 - 2*32) = 17$	*10001*
$x_5 = 51 \bmod 32 = 19$	*10011*
$x_6 = 57 \bmod 32 = 25$	*11001*
$x_7 = 75 \bmod 32 = 11$	*11001*
$x_8 = 33 \bmod 32 = 1$	*00001*

Последовательность повторилась, т. е. период равен 8, как и было подсчитано, в случае четного x_0 *период сокращается вдвое!*

Современные ЭВМ (Pentium 1 и выше) сняли все ограничения по быстродействию и объемам памяти, необходимым для генерации качественных БСВ. Потому при выборе генератора БСВ надо руководствоваться следующими правилами:

1) период повторения должен быть более миллиона;

2) БСВ, производимые генератором, должны отвечать всем тестам на случайность;

3) стараться пользоваться стандартными генераторами типа RANDU (URN01 — URN42), встроенными непосредственно в ЯИМ.

Рассмотрим более подробно схему повторения БСВ. На интервале $(0, m)$ должны быть такие числа r и s $(0 \le r < s < m)$, для которых $x_r = x_s$, а следовательно, и $x_{r+l} = x_{s+l}$ для всех положительных l. Если s наименьшее положительное число, для которого $x_r = x_s$, тогда выражение (3.22) состоит из начального сегмента $x_0, x_1, ..., x_{r-1}$, предшествующего второму сегменту $x_r, x_{r+1}, ..., x_r = x_s$, который затем постоянно повторяется. Тогда целое число $d = s - r$ называется *периодом повторения, последовательностью* или *циклом генератора*. Последовательность имеет максимальный период, если $d = m$. Проиллюстрируем зависимость периода d от a, c, m, x_0 на двух простых примерах.

Пример 1. Пусть $x_0 = 1$, $a = 5$, $c = 0$, $m = 10$, тогда определяем последовательность $x_1 = 9 \bmod 10 = 9$, $x_2 = 81 \bmod 10 = 1$, $x_3 = 9 \bmod 10 = 9$, ... Последовательность имеет вид 1, 9, 1, 9, ... В данном случае $r = 0$, $s = 2$ и период $d = s - r = 2$.

Пример 2. Пусть $x_0 = 1$, $a = 5$, $c = 0$, $m = 10$, тогда определяем последовательность $x_1 = 5 \bmod 10 = 5$, $x_2 = 25 \bmod 10 = 5$, ... Последовательность имеет вид 1, 5, 5, ... В данном случае $r = 1$, $s = 2$, $d = 1$. У данного генератора начальный сегмент 1 предваряет последовательное число 5.

В табл. 3.5 приведены данные генератора для различных значений x_0, a для уравнения (3.23) — при $c = 0$ и $m = 10$.

Таблица 3.5. *Данные генератора*

x_0	a								
	1	2	3	4	5	6	7	8	9
1	1	1,2,4,8,6,	1,3,9,7,	1,4,6,	1,5,	1,6,	1,7,9,3,	1,8,4,2,6,	1,9,
2	2	2,4,8,6,	2,6,8,4,	2,8,	2,0,	2,	2,4,8,6,	2,6,8,4,	2,8,
3	3	3,6,2,4,8,	3,9,7,1,	3,2,8,	3,5,	3,8,	3,1,7,9,	3,4,2,6,8,	3,7,
4	4	4,8,6,2,	4,2,6,8,	4,6,	4,0,	1,4,	4,8,6,2,	4,2,6,8,	4,6,
5	5	5,0,	5,	5,0,	5,	5,0,	5,	5,0,	5,
6	6	6,2,4,8,	6,8,4,2,	6,4,	6,0,	6,	6,2,4,8,	6,8,4,2,	6,4,
7	7	7,4,8,6,2,	7,1,3,9,	7,8,2,	7,5,	7,2,	7,9,3,1,	7,6,8,4,2,	7,3,
8	8	8,6,2,4,	8,4,2,6,	8,2,	8,0,	1,8,	8,6,2,4,	8,4,2,6,	8,2,
9	9	9,8,6,2,4,	9,7,1,3,	9,6,4,	9,5,	9,4,	9,3,1,7,	9,2,6,8,4,	9,1,

По данным табл. 3.5, наибольшее значение периода d равно 4, а плохой выбор a и x_0 значительно уменьшает этот период, в таблице подчеркнуты повторяющиеся сегменты.

Для выбора генератора, обладающего наибольшим периодом, рассмотрим несколько лемм.

Лемма 3.2. Если a и m — простые числа, всегда есть положительное целое d, для которого $x_0 = x_d$. Для простых чисел наибольшим общим делителем (НОД) является единица.

Лемма 3.3. Если НОД $(a, m) = 1$, то тогда период последовательности (3.22) является наименьшим целым положительным числом d, для которого

$$\sigma_d((a-1)\,x_0 + c) = 0 \pmod{m}, \qquad (3.25)$$

где $\sigma_d = 1 + a + a^2 + ... + a^{d-1}$.

Необходимо отметить, что эти леммы действительны только для случая, когда НОД равен 1, поэтому для гарантии надо выбирать только такой генератор, у которого НОД равен 1.

Стремление увеличивать период является необходимым, но не достаточным условием. При увеличении периода может произойти ухудшение качества БСВ, поэтому их необходимо проверять на случайность, что и рассмотрено в § 3.5.

В заключение параграфа рассмотрим тип генератора БСВ, применяемый в GPSS/H. В первой версии языка использован генератор URN 27, работающий на сдвиговом принципе с обратной связью и использующий выражение $x^{31} + x^3 + 1$. Это выражение комбинируется с последовательностью $x_i = 69069\, x_{i-1} \bmod 2^{32}$. Сдвиг вправо на 15 разрядов и влево на 17 обеспечивает получение 32-разрядного числа y_i. Эти x_i и y_i складываются по правилам алгебры логики и создают последовательность z_i, так, например:

$x_i = 01001...01010$
$y_i = 00101...10110$
$z_i = 01100...11100$

Этот генератор не проходил по некоторым тестам случайности, поэтому в более поздних версиях использован усовершенствованный генератор, имеющий в качестве минимального 32-разрядного числа (положительный ноль и 31 значащий разряд) минимальное значение 1 и максимальное 2 млрд 147 млн 483 тыс. 646 ($2^{31} - 2$), а выражение принимает вид $x_{i+1} = 742938285\, x_i \bmod 2147483647$, при этом $x_0 = 266301881$. Таким образом, период повторения этого генератора весьма велик (более 2 млрд), что вполне достаточно для моделирования сложных систем. Независимость потоков БСВ достигается тем,

что потоки заявок и обслуживания берутся с разных генераторов. В ранних версиях использовались 8 встроенных генераторов, стартующих с одинаковых начальных позиций, что делало потоки идентичными, а задачу сбора статистики достаточно сложной. В последних версиях статистическая независимость достигается практически неограниченным числом потоков и отличием каждой последующей выборки на 100 000. Такое отличие потоков практически исключает возможность их корреляции. Более тонкие моменты исследования регулярности этих процессов в пособии не рассматриваются. Кроме того, регулировка сдвига начальных значений легко осуществляется с помощью оператора управления (ОУ) RMULT и ОБ BRMULT, которые позволяют сдвигать начальные значения на один миллион и более! Следует подчеркнуть некоторые методические особенности использования генераторов БСВ. Разброс времени A±B в ОБ GENERATE и ADVANCE всегда берется с генератора RN1, с этого же генератора берется случайное значение ОБ TRANSFER в статистическом виде.

§ 3.5. ОЦЕНКА КАЧЕСТВА БАЗОВЫХ СЛУЧАЙНЫХ ВЕЛИЧИН, ПОЛУЧАЕМЫХ ОТ ПРОГРАММНЫХ ГЕНЕРАТОРОВ

3.5.1. Общие представления

Рассмотрим кратко вопросы оценки качества БСВ, т. е. соответствие последовательности БСВ равномерности распределения в интервале (0,1). Такую оценку позволяют осуществлять статистические тесты. Поскольку причин отклонения БСВ от случайности много, то и тестов проверки также много [15]. Идеальным генератором считается тот, который проходит все статистические тесты. Каждый из таких тестов ориентирован на определение частной причины отклонения последовательности от независимости и случайности. Будем считать, что существует k возможных причин отклонений, и при выполнении гипотезы случайности вероятность появления i-й причины равняется π_i, $i = 1, 2, ..., k$. Предположим, что рассматривается достаточно большая последовательность, в которой по разным причинам возникло n несоответствий по всем k причинам. Количество несоответствий по каждой группе обозначим как f_i, тогда $f_1 + ... + f_k = n$; так как f_i имеет вероятность π_i, то число несоответствий равно $n\pi_1 = e_i$. Измерение различия между f_i и e_i определяется из выражения

$$D = (f_1 - e_1)^2 / e_1 + ... + (f_k - e_k)^2 / e_k = \sum_{i=1}^{k} (f_i - e_i)^2 / e_i. \qquad (3.26)$$

Таблица 3.6. Значения c для различного числа степеней свободы γ

γ	1	2	3	4	5	10	20	50	100
c	3.841	5.992	7.814	9.487	11.071	18.307	31.411	67.504	124.342

При истинной случайности разность между f_i и e_i должна стремиться к нулю, а D должно быть достаточно малым, следовательно, гипотеза о случайности отвергается при большом значении D. При уровне значимости 0.05 вероятность отвергнуть истинную гипотезу о случайности равна

$$P_{H0}(D > c) = 0{,}05. \qquad (3.27)$$

Оценить точное распределение D достаточно сложно, однако при $n > 5$ D приблизительно распределено по хи-квадрат-распределению с числом степеней свободы $\gamma = k - 1$. Для большей уверенности общее число несоответствий обычно принимают равным 10. В табл. 3.6 приведены значения c для различных значений числа степеней свободы с тремя значащими цифрами после точки, при которых справедливо равенство (3.27).

Тесты позволяют исключить процедуру статистического оценивания, при которой из-за ошибок 1-го и 2-го рода оценка вероятности принятия правильной гипотезы равномерности распределения оказывается достаточно сложной. Ниже кратко рассмотрим два типа тестов: тесты на равномерность БСВ и тесты проверки качества самих генераторов.

3.5.2. Тесты оценки качества БСВ

1. Тест равномерности распределения (хи-квадрат-тест).

Второго названия этого теста следует избегать, чтобы не возникло путаницы из-за совпадения названий при рассмотрении распределения D. При этом тесте интервал (0, 1) делится на сто равных и не пересекающихся интервалов длиной 0.01, вероятности появления несоответствий в каждой подвыборке из n равны $e_1 = e_2 = \ldots = e_{100} = n/100$.

Затем проверяется равенство вероятностей в каждом интервале.

2. Тест замещения (сбора купонов).

Имеем последовательность случайных чисел $U_1, U_2, \ldots$, произведем замену чисел на целочисленные значения $1, 2, \ldots, M$ по правилу:

Заменим U_i на 2, если $1/M \le U_i < 2/M$

Заменим U_i на 1, если $0.0 \le U_i < 1/M$

.

.

.

Заменим U_i на M, если $(M-1)/M \le U_i \le 1.0$

Эти новые целые числа от 1 до M должны быть случайными, если U_i действительно случайные числа. Рассмотрим последовательно всю полученную совокупность и обозначим через Q новые числа, полученные после M, например $M+1$, $M+2$, ...,. При соблюдении случайности вероятности $P(Q=M)$, $P(Q=M+1)$, $P(Q=M+2)$, ... известны. Если вычислить такие значения Q повторно, то тогда тест 2 может быть применен для проверки случайности начальных чисел U_i.

3. Тест расхождения.

Положим, что α и β лежат между 0.0 и 1.0 при условии, что $\alpha < \beta$. Имеющуюся последовательность чисел U_1, U_2, ..., будем рассматривать с позиции их нахождения: либо в интервале $[\alpha, \beta]$ включительно границы, либо вне его. Будем считать, что каждое число, лежащее в интервале, равно 1, а вне его равно 0. Эта операция преобразует последовательность в набор 0 и 1. Положим, что $p = \beta - \alpha$, тогда, если последовательность чисел случайна, то вероятность того, что j-й 0 появится после 1, прежде чем появится следующая 1, равняется

$$p_j = p(1-p)^j, \quad j = 0, 1, 2, ... \qquad (3.28)$$

После определения, сколько раз появится каждая категория, можно применить тест 1, который оценивает случайность в соответствии с (3.28). Этот тест введен Кендаллом и Бабингтон-Смитом и предусматривает следующие правила для выбора α и β:

вариант 1 $\alpha = 0.0$, $\beta = 0.5$ (ниже среднего);

вариант 2 $\alpha = 0.5$, $\beta = 1.0$ (выше среднего);

вариант 3 $\alpha = 0.333$, $\beta = 0.667$ (нахождение в середине).

4. Тест перестановки.

Считается, что полученная последовательность случайных чисел U_1, U_2, ..., содержит наборы T чисел. Каждый такой набор содержит $T!$ возможных перестановок, которые можно рассматривать как T чисел, расположенных в порядке убывания от наибольшего к наименьшему. При случайности вероятность появления каждого набора $T!$ равна $1/T!$. При $T!$-категориях можно определить, сколько кортежей образовано из соответствующих наборов T и сколько из них попало в соответствующую $T!$-категорию. Затем применяется тест 1, для этого теста рекомендованы значения $T = 3, 4, 5$.

5. Покер-тест.

В полученной последовательности случайных чисел $U_1, U_2, \ldots$, произведем преобразование в целые числа 1, 2, ..., 10 по правилу:

Заменим U_i на 1, если $0.0 \leq U_i < 0.1$

Заменим U_i на 2, если $0.1 \leq U_i < 0.2$

.

.

.

Заменим U_i на 10, если $0.9 \leq U_i \leq 1$

Новые числа должны быть случайными, если U_i — случайные числа, в этом случае используется тест 2 с $M = 10$.

Вариант 1. Рассмотрим последовательные наборы из пяти чисел:

— пять любых одинаковых чисел из 1–10 обозначим *AAAAA*;

— одно измененное число типа *AAAAB*;

— сочетание типа *AAABB*;

— сочетание типа *AAABC*;

— сочетание типа *AABBC*;

— сочетание типа *AABCD*;

— сочетание типа *ABCDE*.

Только такие сочетания могут встречаться при проверке чисел. Вероятности появления сочетаний из пяти цифр при истинной случайности:

$P\,(AAAAA) = 0.0001$
$P\,(AAAAB) = 0.0045$
$P\,(AAABB) = 0.0090$
$P\,(AAABB) = 0.0720$
$P\,(AAABC) = 0.1080$
$P\,(AABCD) = 0.5040$
$P\,(ABCDE) = 0.3024$

Обычно в связи с малостью значений вероятности первое и второе сочетания объединяются вместе, в результате получается шесть сочетаний из пяти цифр (отсюда карточное название теста).

Вариант 2. Для варианта 1 можно подсчитать, сколько различных цифр будет в пяти выбранных. Значения легко подсчитываются по данным варианта 1:

$P\,(1) = 0.0001$
$P\,(2) = 0.0135$
$P\,(3) = 0.1800$
$P\,(4) = 0.5040$
$P\,(5) = 0.3024$

В этом варианте первое и второе значения чаще всего также объединяются.

Вариант 3. Часто ограничиваются выбором не пяти цифр в сочетании, а четырех, в этом случае:

$P\,(AAAA) = 0.001$
$P\,(AAAB) = 0.036$
$P\,(AABB) = 0.027$
$P\,(AABC) = 0.432$
$P\,(ABCD) = 0.504$

И здесь первое и второе значения чаще всего объединяются.

6. Тест последовательных пар.

Имеем последовательность чисел $U_1, U_2, \ldots,$. Пусть имеется целое число M ($M \geq 2$). Произведем замену каждого числа U_i на выражение $1 + \text{ЦЧ}\,(U_i M)$, где ЦЧ (.) — целая часть произведения случайного числа на M; в случае равенства произведения выбранному числу M ЦЧ (.) $= M - 1$.

Новые числа будут случайными целыми от 1 до M, если U_i случайны. Теперь образуем матрицу из пар полученных чисел вида

$(1, 1)\,(1, 2)\,\ldots\,(1, M-1)\,(1, M)$
$(2, 1)\,(2, 2)\,\ldots\,(2, M-1)\,(2, M)$
.
.
.
$(M, 1)\,(M, 2)\ldots\,(M, M-1)\,(M, M)$

и определим, в какой из M^2-категорий лежит каждое число. При истинной случайности каждая пара должна иметь одинаковую вероятность, равную $1/M^2$, и после этого можно использовать тест 1. Чаще всего принимают $M = 3, 10, 20$ для первого, второго и третьего варианта соответственно.

Каждый из этих шести тестов оценивается статистиками, которые при случайности последовательности U_i примерно подчиняются хи-квадрат-распределению с числом степеней свободы, заданным в каждом тесте. Обычно выборка проверяемых чисел ограничивается количеством от 1000 до 10000. Если все тесты приводят к значению $D\,(3.26) \geq 0{,}95$, то качество случайных чисел достаточно высокое. Если хотя бы один из тестов не проходит, то такой генератор не должен использоваться. Однако при этом следует помнить, что при увеличении объема выборки все тесты могут дать положительный ответ.

Проверка по критерию хи-квадрат, распределений хи-квадрат, получаемых при использовании каждого теста, носит название теста CHCH (chi-square on chi-square). Его введение позволяет нивелиро-

вать непрохождение на случайность по какому-либо из тестов. В настоящее время существует программа TESTRAND, проверяющая 10 млн случайных чисел по всем названным критериям, однако ее применение прежде всего должно интересовать разработчиков генераторов и в меньшей степени — пользователей. *Вместе с тем при ИМ сложных, ответственных систем небесполезно проводить тщательную проверку генерируемых БСВ!*

3.5.3. Теоретическая оценка качества генераторов

В некоторых случаях до проверки последовательности БСВ на случайность проводится теоретическая проверка конгруэнтных генераторов, реализующих последовательность (3.22).

1. Последовательный корреляционный тест.

При случайности последовательности $U_1, U_2, \ldots$, корреляция строго равна 0, т. е. $R(U_i, U_{i+1}) = 0$.

Подставив $U_i = x_i / m$ в (3.22), можно показать, что

$$R(U_i, U_{i+1}) = 1/a\,(1 - 6c/m + 6\,(c/m)^2 + \varepsilon) \qquad (3.29)$$

при

$$|\varepsilon| \le (a + 6)/m. \qquad (3.30)$$

Если a и m простые числа, возможно (3.30) заменить на равенство. Дадевичем [14] показано, что не рекомендуется применять генератор, у которого правая часть (3.29) больше 0.01, а правая часть (3.30) меньше 0.005. Например, для генератора URN13 при $c = 0$, $a = 663\,608\,941$ и $m = 2^{32}$, проводя оценку по (3.29), (3.30), получаем $1/a = 0.0000000015$, $(a + 6)/m = 0.1545$, следовательно, генератор отвечает сформулированным правилам и проходит корреляционный тест.

2. Спектральный тест (межплоскостных расстояний).

Этот тест основан на утверждении, что n выборок $(U_1, \ldots, U_n)$, $(U_2, \ldots, U_{n+1})$, $(U_3, \ldots, U_{n+2}), \ldots$, полученных от генератора по формуле (3.22), лежат в конечном и малом числе параллельных, одинаково расположенных гиперплоскостей [15]. То, что они лежат в конечном числе плоскостей, очевидно, а то, что количество плоскостей мало, является предметом обсуждения. Каждая плоскость должна содержать по меньшей мере три точки, тогда наименьшее число плоскостей будет равно $(n!m)^{1/n}$. В табл. 3.7 приведены значения этого произведения для разного числа выборок n при $m = 2^{32}$.

Естественно, что существуют различные наборы плоскостей, содержащие все эти точки. Рассмотрим такой набор, когда расстояние

Таблица 3.7. Значения произведения

n	2	3	4	5	6	7	8	9
$m = 2^{32}$	92681	2953	566	220	120	80	60	48

между плоскостями будет наибольшим, тогда критерием спектрального теста будет являться *расстояние между плоскостями* d_n при n измерениях. Существует алгоритм TSTM4, входящий подпрограммой в TESTRAND, легко подсчитывающий это расстояние при различных значениях коэффициентов (3.22). Очевидно, что при фиксированном n, чем меньше d_n, тем лучше генератор. Критерий, применяемый в литературе, выглядит следующим образом: $d_n \le 2^{-30/n}$, в табл. 3.8 это значение приводится в зависимости от n.

Например, для генератора RANDU, широко применяемого в России, при данных $c = 0$, $a = 2^{16} + 3$, $m = 2^{32}$ имеем: $d_2 = 0.00002$, $d_3 = 0.092$, $d_4 = d_5 = d_6 = d_7 = d_8 = d_9 = 0.093$, т. е. этот генератор соответствует спектральному тесту только для двумерного случая. Кстати, генератор, используемый в GPSS/H, проходит спектральный тест до $n = 9$.

Название «спектральный тест» пришло из теории распространения волн, где наибольшее значение $v_n = 1/d_n$ служило критерием наилучших условий распространения (а в нашем случае — случайности).

Все векторы чисел $(U_i, ..., U_{i+n-1})$ лежат в n-мерном единичном кубе. Наименьшее расстояние между двумя плоскостями этого куба равно расстоянию от начального значения (0,0, ..., 0,0) до (0,0, ..., 0,1) и вычисляется как

$$d_L(n) = ((0-0)^2 + (0-0)^2 + ... + (0-1)^2)^{0.5} = 1,$$

а наибольшее расстояние до (1, 1, ..., 1,1) — из выражения

$$d_U(n) = ((0-1)^2 + ... + (0-1)^2)^{0.5} = \sqrt{n}\ .$$

Поскольку расстояние длиной l может быть разделено r плоскостями на $r-1$ равных промежутка длиною $l/(r-1)$, то отсюда следует, что генератор номера плоскости $(n!m)^{1/n}$ будет иметь расстояние между плоскостями, лежащее между

$$1/((n!m)^{1/n} - 1) \text{ и } \sqrt{n}\ /((n!m)^{1/n} - 1). \qquad (3.31)$$

Таблица 3.8. Зависимость d_n от n

n	2	3	4	5	6	7	8	9
d_n	0.00003	0.001	0.005	0.016	0.03	0.05	0.07	0.1

Таблица 3.9. *Значения наилучшего расстояния*

n	2	3	4	5	6	7	8	9
$m = 2^{16}$	0.00281	0.0141	0.0291	0.044	0.056	0.065	0.071	0.076
$m = 2^{32}$	0.00001	0.0003	0.0018	0.005	0.008	0.013	0.017	0.021

Первое значение является наилучшим возможным расстоянием при n-мерной выборке для генератора модуля m. В табл. 3.9 представлены значения наилучшего расстояния в зависимости от n и m, и можно считать, что они являются стандартом для оценки добротности генератора, выраженной в расстоянии между плоскостями $d_2, ..., d_9$ при заданном m.

Если поделить d исследуемого генератора на лучшее расстояние, приведенное в табл. 3.9 для выбранного m, то полученное значение r_i в пределе равно 1 и генератор тем лучше, чем ближе результат деления к 1. Например, проверим на основе этого правила генератор RANDU, пользуясь значениями d_L, полученными выше:

$$r_2 = 0.00002/0.00001 = 2,\ r_3 = 307,\ r_4 = 52,\ r_5 = 19,$$
$$r_6 = 12,\ r_7 = 7,\ r_8 = 5,\ r_9 = 4.$$

Из расчета видно, что наиболее плохое поведение генератора — при размерностях 3 и 4. Из (3.31) найдем число плоскостей для 3-размерного случая для этого генератора. В этом случае расстояние $d_3 = 0.092$, следовательно, нижний предел равен $1/0.092 + 1 = 11.8$, а верхний — $\sqrt{3}\ /\ 0.092 + 1 = 19.8$. На самом деле, для этого генератора возможное число плоскостей (см. табл. 3.7) равно 2953, поэтому с определенными оговорками этот генератор можно принять.

Пример. Проиллюстрируем сказанное на примере генератора с коэффициентами

$$x_{i+1} = 7x_i \bmod 11,\ x_0 = 1.$$

Из (3.29) оценим корреляцию $R(U_{i+1}, U_L) = 1/7 + \varepsilon$, а из (3.30) $\varepsilon \le 13/11 = 1.18$

Таким образом, такой генератор может пройти тест благодаря большому значению ошибки. При $n = 2$ получим последовательность 1, 7, 5, 2, 3, 10, 4, 6, 9, 8, 1, которая имеет период $m - 1 = 10$. В результате получим 10 пар случайных чисел (1, 7), (7, 5), (5, 2), (2, 3), (3, 10), (10, 4), (4, 6), (6, 9), (9, 8), (8, 1). Спектральный тест предусматривает получение 10 плоскостей с интервалом $1/11 = 0.909$, каждая из которых содержит одну из пар (рис. 3.11, линии *A*).

Но для конгруэнтного генератора при данных параметрах плоскостей должно быть не больше $(2!\,11)^{1/2} = 4$ (рис. 3.11, линии *B*) — это, естественно, предельный — худший — случай. Линии *B* (рис. 3.11) представляют

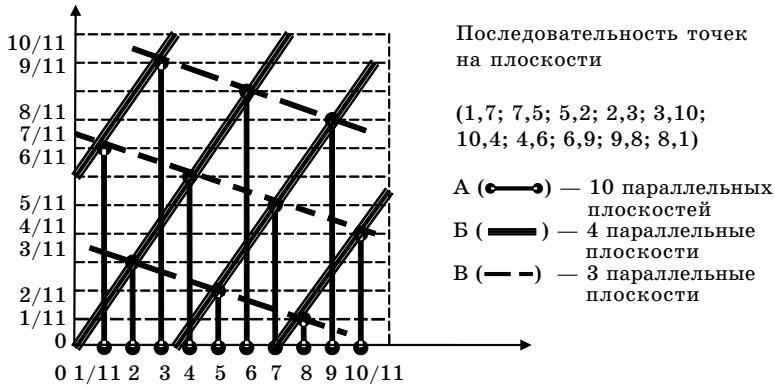


***Рис. 3.11**. Расположение пар чисел*

три плоскости, содержащие все точки. Расстояние между параллельными плоскостями $d_2 = 0.3162$. Лучшее расстояние для этого генератора при $m = 11$, $n = 2$ равно $1 / (4 - 1) = 0.33$; таким образом, $r = 0.362/0.333 = 0.95$. Следовательно, рассматриваемый генератор для заданных параметров весьма хорош. Однако, если принять $m = 2^{32}$, что естественно для современных компьютеров, у которых в этом случае лучшее расстояние равно 0.00001, то значение $r = 0.3162/0.00001 = 31620$. Это отношение очень большое, и такой генератор не может использоваться категорически!

Приведенный пример подчеркивает, что рекомендации по выбору генератора должны всегда соблюдаться, среди них главными являются:

— период генератора должен быть не менее 1 млрд;

— генератор должен пройти все проверки.

Использование плохого генератора обесценит все усилия исследователя даже при идеальной модели.

Часть 2
ЯЗЫК ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ GPSG/H

Вторая часть пособия является основной; она посвящена исключительно описанию ЯИМ GPSS/H и предназначена для двух целей: как вспомогательный материал к курсу лекций по ИМ и как основа для самостоятельного изучения принципов ИМ на GPSS/H. Рассматриваемый ЯИМ GPSS/H построен таким образом, что значительный объем ИМ ведется самой программой без участия исследователя, более того, исследователь даже при желании не может вторгаться в некоторые процессы (например, трансляция написанного пользователем модельного файла, перевод транзактов из одного списка в другой, переключение режимов, контроль протекания ИМ и т. д.). В связи с этим очередность чтения материала диктуется самим читателем. Так, общие основы построения ЯИМ GPSS/H, его основные объекты, правила записи операторов необходимо читать в первую очередь, а вопросы генерации транзактов, их передвижения из списка в список можно читать после ознакомления с общими принципами. Для подчеркивания этого обстоятельства параграфы, весьма важные по сути, но не влияющие на освоение логики ЯИМ, будут отмечаться знаком *. Появление этого знака в заголовке совершенно не означает, что этот параграф может быть опущен, а лишь указывает на последовательность изучения. Из методических соображений, чтобы не нарушать логику изложения, эти параграфы не обязательно собраны вместе. Материал раздела не может быть адекватно воспринят без:

- практического моделирования на компьютере;
- решения упражнений вначале без помощи компьютера, и лишь затем с проверкой ответов на компьютере;
- освоения и использования отладчика (дебаггера), причем в тексте оба названия используются равноправно;

Итак, логика освоения материала должна быть выстроена следующим образом:

- освоение общих принципов ЯИМ GPSS/H;
- выполнение элементарных упражнений, помогающих освоить смысл операндов ОБ;
- написание элементарных модельных файлов (МФ) и моделирование их на компьютере;
- освоение принципов работы отладчика;
- создание простых МФ, имитирующих реальные ситуации;

• освоение более сложных ОБ и ОУ;

• создание реальных моделей с учетом принципов внутренней логики ЯИМ GPSS/H (параграфы, отмеченные *).

Подобная логика основана на опыте преподавания ИМ студентам 4–5 курсов, не знакомых ранее с ИМ, и, естественно, не ориентирована на опытных пользователей, которые должны следовать собственной логике.

Кроме основного материала, описывающего базовые положения ЯИМ GPSS/H, включены многочисленные примеры и упражнения, которые должны способствовать реализации вышеупомянутой логики. Большинство упражнений имеют ответы, помещенные в конце книги, однако, ответами не следует злоупотреблять, а использовать их только для проверки полученных самостоятельно решений. Отсюда ясно, что понимание и закрепление материала второй части — путь к Вашему успеху при использовании ЯИМ GPSS/H.

Опыт преподавания показывает, что вроде бы очевидные и логичные вещи, как, например, правила окончания испытаний или вес терминирования, вызывают у студентов большую трудность при освоении материала. Поэтому автор взял на себя смелость изложить эти вопросы более подробно, что может заставить улыбнуться достаточно «продвинутых» пользователей, приношу заранее свои извинения — это написано не для них.

В четвертой главе проводится сравнение GPSS/H с предыдущими версиями; рассматриваются основные отличия и нововведения языка; дается общая классификация объектов GPSS/H; приводится материал по особенностям программной реализации процесса функционирования, формальные сведения о работе с пакетом GPSS/H, формат записи и структура МФ.

В пятой главе рассматриваются основные операторы GPSS/H (список всех применяемых операторов см. в прил. 1), изучаются их операнды и основные атрибуты, описываются принципы движения транзактов, рассматриваются правила прекращения ИМ по числу стартов и времени испытаний.

Шестая глава посвящена моделям одноканального и многоканального обслуживания, вопросам автоматизации прогонов при ИМ, сбора необходимой статистики, создания итоговых отчетов.

В седьмой главе изучаются вопросы, связанные с использованием отладчика (дебаггера), рассматривается его построение, использование основных команд.

Восьмая глава включает в себя вопросы, связанные со статистическими проблемами GPSS/H, а именно: сокращение числа реализаций в одном прогоне, использование потока встречных (дополняю-

щих или антитез) БСВ, уменьшение дисперсии за счет двухэтапного просчета, а также выбор рациональных решений (субоптимальных) среди имеющихся альтернатив.

Читателю надо ответить на вопрос, что является для него первичным: изучение принципов имитационного моделирования или непосредственное изучение GPSS/H. В данной части первичным признается изучение ЯИМ GPSS/H, поэтому материал совершенно не обязательно изучать в изложенной последовательности. Так, к содержанию главы 7 неминуемо придется обращаться при первом упоминании об отладчике и использовать его при рассмотрении примеров или решении упражнений 5-й и 6-й глав.

Автор не считает логику изложения идеальной, открыт для дискуссий и с благодарностью примет любые конструктивные предложения.

Глава 4
ОСОБЕННОСТИ ЯЗЫКА ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ GPSS/H

§ 4.1. ИСТОРИЯ РАЗВИТИЯ GPSS/H И ОБЩАЯ ИДЕОЛОГИЯ

Язык ИМ GPSS/H был запущен в коммерческое использование в 1977 г. Благодаря быстродействию, надежности, наличию многоцелевого, мощного симулятора он получил широкое распространение в мире. Эффективность и гибкость GPSS/H позволяют одинаково просто, но с высоким качеством создавать модели для систем различной сложности.

Первая версия GPSS была создана в октябре 1962 г. IBM Corp. по заказу военно-морского флота США. Главная идея, которую реализовал создатель ЯИМ Джеффри Гордон (Geoffrey Gordon), заключалась в том, что ЯИМ адресован пользователям, не владеющим навыками программирования. Эта версия основывалась на движении динамических (активных) элементов, названных транзактами или Хакт. Это название применяется еще в ряде ЯИМ, хотя динамические элементы могут носить и другое название (item, token, puck и т. д), сохраняя основной функциональный смысл движения по логической схеме модели. При этом *активные* объекты используют *пассивные ресурсы*. Интересно отметить, что спустя 40 лет, в начале XXI в. идеология языка для пользователей не только не устарела, но стала определяющей при использовании ЯИМ. Первоначально мо-

дели GPSS изображались в виде блок-схем, в которых каждый оператор блока (см. гл. 5) имел свои характерные отображения. Но спустя годы размеры МФ настолько возросли, что представление их в виде блок-схем сделало их трудно воспринимаемыми и необозримыми. Кроме того, вывод их на печать крайне затруднителен. Поэтому в последние годы возобладала тенденция отказа от использования блок-схем, что, кстати, сделало GPSS/H приближенным к универсальным языкам программирования. Учитывая сказанное, в пособии не используются блок-схемы, исходя также из:

• методического упрощения, так как исключается запоминание графического отображения операторов, что ускоряет освоение ЯИМ;

• простоты визуализации как МФ, так и получаемого автоматически окончательного отчета (листинга), который к тому же может быть сокращен за счет наличия операторов редактирования ЯИМ GPSS/H.

Из наиболее известных вариантов GPSS следует назвать GPSS/360, GPSS V, GPSS PC (для персональных компьютеров) и его современное развитие GPSS World (Minuteman SW) и, наконец, **GPSS/H.** Последний разработан авторским коллективом Wolverine Software Corp. под руководством президента корпорации Джима Хенриксена (James O. Henriksen). Он объединил в себе достоинства GPSS, а кроме того, включил целый ряд преимуществ, сделавших его языком программирования в широком смысле. Большинство ЯИМ, которых на сегодняшний день насчитывается несколько сотен, предназначены в основном для решения специализированных задач, а GPSS/H способен решать широкий круг задач, относящихся к ИМ систем различного назначения. Он, естественно, требует определенных навыков, приобретаемых достаточно быстро, но основная идея Д. Гордона сохранена и усовершенствована. Рассмотрим некоторые отличительные черты ЯИМ GPSS/H.

Текстовое представление информации

Язык ИМ GPSS/H основан на использовании текстовых файлов, создаваемых в любом варианте текстового редактора, а поскольку в пособии все рассмотрение ведется на базе студенческой версии GPSS/H, основывающейся на MS DOS, то в тексте описывается редактор таких программных оболочек как NC, VC, Far. Использование текстовых файлов требует некоторых пояснений. Зачастую мощный ЯИМ ограничивает свои возможности из-за применяемого интерфейса, который состоит из иконок, всплывающих меню и таблиц данных. Тем не менее, многие исследователи превозносят достоинства визуа-

лизации данных, когда модель строится быстро из набора имеющихся иконок, соответствующих компонентам системы (например, в ЯИМ Симулинк широко используемого ППП MATLAB). И все происходит достаточно быстро и сравнительно точно, пока исследуемая система имеет малую размерность, однако по мере нарастания сложности системы необходимо включать процедурную ориентацию, которую достаточно сложно отразить графически. В таких системах требуется включать и процедурную, и лексикографическую информацию и создавать многоуровневое ее представление. Все это ведет к резкому увеличению сложности рассмотрения, редактирования и документирования информации. Большое количество связей между меню и таблицами заставляет исследователя проходить через иерархию уровней рассмотрения. Блуждание по лабиринту иконок, меню, таблиц ведет к усложненному восприятию и увеличению вероятности появления ошибок. Возникающее стремление к упрощению процедуры ведет к нарастающей потере точности, делая модель не адекватной реальной системе. Визуальное представление ограничивает исследователя имеющимся набором иконок, меню и таблиц. Очевидно, что такой подход хорош только в двух случаях: когда система простая или исследователь не опытен. Все преимущество визуализации превращается в серьезное ограничение при расширенном представлении достаточно сложной реальной системы. В отличие от такого подхода, текстовая ориентация проста для чтения и просмотра. При этом не надо запоминать, какая деталь имеет отношение к той или иной иконке, или метаться с уровня иконного представления на уровень процедурной логики. Текстовое описание позволяет детализировать представление об усложняющейся схеме, делая все это в едином стиле.

Простота

Эта характеристика зависит как от общих принципов, так и от специфических особенностей. В общее понятие простоты входят простота изучения, использования, модификации при переходе к усложненному модельному файлу; построения моделей любой сложности; верификации.

Гибкость

Эта характеристика присуща именно GPSS/H и включает в себя несколько составляющих:

— концептуальную гибкость, т. е. приспособленность к ИМ разнообразных систем любой сложности, если их можно представить в

виде объектов СМО (люди в общественном транспорте, операции в банке, данные в вычислительной сети и т. д.);

— гибкость использования, т. е. возможность применения сложных математических формул, выражений и постоянных в любом месте МФ. Для большей читаемости элементам и объектам можно присваивать собственные имена;

— гибкость представления данных, т. е. возможность автоматического сбора статистики об очередях, параметрах обслуживания, прохождении транзактов в любой наперед заданный момент модельного времени;

— гибкость к машинной реализации, т. е. GPSS/H может использоваться как на персональных компьютерах, так и на мощных рабочих станциях и при длине машинного слова, равного 32 разрядам, работать под ОС MS DOS, Windows XP, 2000, 98, 95, NT, 3.x,OS/2;

— гибкость ввода данных, заключающаяся в различных способах ввода (с клавиатуры, со стандартных текстовых файлов) и вывода (непосредственно на экран дисплея или в виде текстовых файлов) данных.

Статистическая независимость БСВ

Для GPSS/H был специально разработан программный генератор (см. § 3.4), обеспечивающий действительную независимость и случайность БСВ и неограниченное число потоков чисел.

Методы отладки

В современной версии GPSS/H, в отличие от первой версии 1977 г., отладчик (дебаггер) стал пакетно-ориентированным и независимым. Отладчик позволяет проверять корректность модели, а с помощью простых команд контролировать исполнение МФ и проверять состояние процесса моделирования. Отличительными особенностями отладчика являются (см. гл. 7):

— оконное представление всей необходимой информации;

— возможность прерывания длительных программ с целью проверки правильности процесса моделирования;

— возможность использования отладчика непосредственно для реализации процесса моделирования, поскольку этот режим практически не влияет на время исполнения МФ.

Более подробно основные преимущества и нововведения GPSS/H будут рассмотрены в следующем параграфе.

Процесс моделирования, адекватно отражающий особенности реальной системы («правильное моделирование»), требует хорошего

знания как программного обеспечения, так и исследуемой системы. Чаще всего такое единение достигается крайне редко, и здесь на помощь приходит симулятор специального назначения, являющийся одной из особенностей GPSS/H. Этот симулятор облегчает построение простых МФ и позволяет провести экспертизу модели внешними экспертами. Тем самым пользователь освобождается от необходимости изучения основ моделирования и особенностей ПО. Симулятор создан опытными программистами специально для GPSS/H. Основой его является управляемая данными модель реальной системы или набор таких моделей. Симулятор обеспечивает пользователю возможность изменять параметры модели, определять порядок эксперимента и выдавать результирующие данные. Симулятор обычно обеспечивает последовательный ввод данных с начала МФ до конца. Такой метод ввода позволяет изменять параметры прогона без изменения модели. При этом существуют два способа: первый, наиболее часто применяемый — редактирование текстового файла вручную, при втором модель сама приглашает пользователя вводить данные с клавиатуры в процессе ИМ. Кроме того, предусмотрена возможность подключения внешних подпрограмм или таблиц данных. В современной версии используется улучшенная версия симулятора (рис. 4.1), компоненты которой выполняют следующие функции.

• Оболочка управляется меню или непосредственно с клавиатуры. Окна ввода данных и диалоговые меню руководят пользователем в процессе определения параметров, реализации МФ и сбора выходной статистики, оболочка обеспечивает также проверку того, что время начала испытаний не отрицательно.

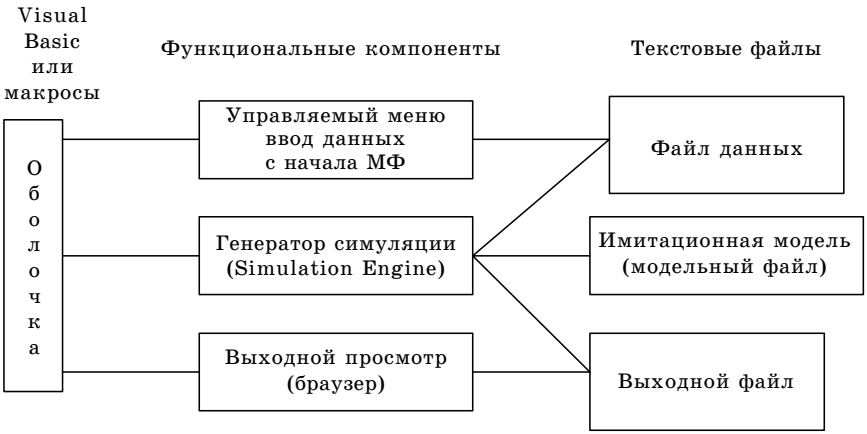


Рис. 4.1. *Компоненты симулятора специального назначения*

• Управляемый ввод данных позволяет вводить параметры модели и записывать файл данных.

• Генератор симуляции производит запуск модели с введенными параметрами и считывает информацию файла данных, кроме того, он подготавливает выходной отчет (листинг). Основными характеристиками генератора являются гибкость и быстродействие (скорость исполнения МФ). Эти характеристики симулятора весьма высоки, что является большим преимуществом GPSS/H. Так, к примеру, скорость компиляции превышает скорости компиляции Fortran в пятьдесят и более раз.

• Браузер осуществляет просмотр формата и вида данных, а также анализирует результаты. Информация отображается на дисплее в виде, удобном для понимания и анализа. Данные, представляемые в отчете (без вмешательства пользователя), достаточны для довольно полного представления о характеристиках системы. В случае, если пользователь обладает необходимыми навыками, то он может расширить круг получаемых статистических характеристик. Статистическая обработка данных может производиться оболочкой, внешними подпрограммами или специальным ППП по статистике. Одной из форм выходного отчета является анимация (см. гл. 9).

• Наиболее важной частью симулятора является исследуемая модель. Поскольку МФ чаще всего меняется в процессе испытаний, то этот компонент обеспечивает наибольшую гибкость, учитывая также возможность изменения входных данных и периодическую проверку достоверности МФ. Использование возможностей программируемой гибкости позволяет рассматривать и динамику поведения модели, а также выбирать рациональные варианты из ряда имеющихся альтернатив (см. гл. 8).

• Представление о файлах входных и выходных данных не требует дополнительных комментариев.

Следующим моментом, относящимся к общей идеологии GPSS/H, является рассмотрение его объектов. При использовании языка GPSS/H могут возникнуть две ситуации:

1) модель упрощена и требуется только начальное представление о реальной системе, а в основу положена концептуальная модель со многими допущениями — в этом случае использование языка не является оправданным;

2) модель излишне детализирована — это приводит к увеличению времени создания модели и возрастанию стоимости.

Читателю всегда надо помнить об афоризме Р. Беллмана (одного из создателей теории динамического программирования): «Исследователь всегда должен искать узкую тропу между Сциллой переупрощения и

Харибдой переусложнения». Поэтому основной задачей является выбор начального варианта модели, рационального в плане расхода ресурсов, построение уточненной модели после реализации первого варианта и т. д. Таким образом, только достигнув нужного уровня абстракции, можно переходить к составлению блок-схемы модели.

Вначале рассмотрим основные категории и типы объектов GPSS / H.

В § 3.3 отмечалось, что транзактный способ реализации квазипараллелизма является развитием процессного способа имитации применительно к моделированию систем, представляемых в виде СМО. Для указанного класса систем выделяется конечное множество абстрактных элементов, служащих для описания реальных компонентов системы (например, источников заявок на обслуживание накопителей, каналов обслуживания и т. п.), и конечное множество стандартных операций, описывающих связи между элементами. Выделенным множествам компонентов и операций ставится в соответствие множество объектов языка. Модель системы на GPSS/H строится путем объединения объектов в некоторую фиксированную логическую структуру — текстовый МФ. Объекты GPSS/H подразделяются на 7 категорий и 15 типов (табл. 4.1).

Каждому объекту соответствуют *арифметические* или *логические атрибуты,* описывающие состояние объекта в текущий момент модельного времени. Большинство из атрибутов не доступны для программиста. Атрибуты, к которым в ИМ можно обращаться, называются *стандартными числовыми атрибутами* (СЧА). Основными объектами GPSS/H являются транзакты и ОБ (блоки).

• *Транзакты (сообщения)* описывают единицы исследуемых потоков (заявки на обслуживание), например: задания пользователей в вычислительной системе; детали, подлежащие обработке на производстве; автомобили в очереди у бензоколонки; корабли, разгружающиеся в порту, и т. п.

• *Операционная категория* включает блоки, которые задают логику функционирования ИМ системы и определяют пути движения транзактов между объектами аппаратной категории. Практически все изменения состояний ИМ (события) системы S происходят в результате входа транзактов в блоки и выполнения блоками своих функций. Основные функции блоков следующие:

— создание (генерация) и уничтожение транзактов;

— изменение числовых атрибутов объектов;

— задержка транзакта на определенный интервал времени;

— изменение маршрута движения транзакта и др.

Проиллюстрируем эти функции на простом примере.

Таблица 4.1. *Категории и типы объектов**

№ категории	Категория объекта	№ типа	Тип объекта	Мнемоническое обозначение	Функции ОБ
1	Динамическая	1	Транзакт	Хакт	Создание транзактов (GENERATE, SPLIT) Уничтожение транзактов (TERMINATE, ASSEMBLE)
2	Операционная	2	ОБ	–	Объяснены в гл. 5
3	Аппаратная	3	Устройства	F (facilities)	Занятие — освобождение (SEIZE — RELEASE) Захват — возврат (PREEMPT — RETURN) Доступно — не доступно (FAVAIL – FUNAVAIL) Выбор обусловленного направления (GATE)
		4	Памяти (накопители)	S (storages)	Войти – покинуть (ENTER — LEAVE) Свободна — занята (SAVAIL — SUNAVAIL) Ожидание изменения статуса (GATE) Изменение емкости памяти (BSTORAGE)
		5	Логические ключи	L (logic switch)	Включение, выключение, инверсия (LOGIC) Ожидание изменения положения (GATE)
4	Вычислительная	6	Арифметическая переменная	V (variable)	Целочисленное значение (VARIABLE) Плавающая точка (FVARIABLE)
		7	Булева переменная	BV	Задается стандартными логическими атрибутами
		8	Функция	FN	Задается пользователем или встроенной функцией

Окончание табл. 4.1

№ категории	Категория объекта	№ типа	Тип объекта	Мнемоническое обозначение	Функции ОБ
5	Статистическая	9	Очереди	Q	Создание очереди — покидание (QUEUE — DEPART)
		10	Таблицы	T	Создать таблицу (TABULATE)
6	Запоминающая	11	Ячейки	X	Создание скалярной переменной
		12	Матрицы	M	Создание двух размерных матриц
		13	Амперпеременная	&	Создание переменных пяти типов
7	Группирующая	14	Списки пользователей	C	Включить — исключить (LINK — UNLINK)
		15	Группы	G	Поместить — удалить (JOIN — REMOVE) Проверка принадлежности (EXAMINE) Определение вида транзакта (SCAN) Изменение атрибутов (ALTER)

111* Конкретное описание ОБ будет дано в гл. 5, в таблице же дано общее начальное представление.

Пример. Блок, «создающий» транзакты в модели, обеспечивает поступление заявок в СМО через определенные интервалы времени. Занятие или освобождение заявкой канала обслуживания (или места в накопителе) приводит к изменению состояния канала (накопителя). В модели это осуществляется с помощью изменения СЧА объекта GPSS/H, описывающего состояние канала обслуживания (накопителя). В случае занятости одного из каналов СМО заявка может быть направлена на другой канал. Для этого в модели используется блок изменения маршрута движения транзакта. Блок, осуществляющий задержку транзакта, имитирует процесс обслуживания заявки в течение определенного времени. Выход обслуженной (или потерянной по каким-либо причинам) заявки из СМО в модели имитируется с помощью блока уничтожения транзактов.

• *Объекты аппаратной категории* служат для описания единиц оборудования или *ресурсов*, имеющих ограниченную емкость. Воздействуя на эти объекты, транзакты могут изменять их состояния и влиять на движение других транзактов. Ресурсы включают в себя три типа объектов

Устройства описывают оборудование, которое в любой момент времени может быть занято только одним транзактом (одноканальные СМО): обрабатывающий центр, терминал, центральный процессор, АЦПУ, кассир и т. д., — а также оборудование, на котором обслуживание одной заявки может быть прервано поступлением другой заявки (например, с более высоким приоритетом).

Памяти (многоканальные устройства) описывают оборудование, которое может использоваться несколькими транзактами одновременно (многоканальные СМО): оперативную память ЭВМ, бункер-накопитель в гибкой производственной системе (ГПС), стоянки автомобилей и т. д.

Логические ключи используются для блокировки или изменения движения транзактов в зависимости от ранее наступивших в ИМ событий.

• *Объекты вычислительной категории* описывают связи между элементами системы, задаваемые с помощью аналитических или логических соотношений. Они могут служить для задания вероятностных законов распределения случайных величин в ИМ; для численного или логического описания условий, определяющих движение транзактов.

• *Статистические объекты* обеспечивают вычисление и представление в стандартном виде для показателей эффективности функционирования системы: средних значений, стандартных отношений, эмпирических функций распределения и т. п.

• *Запоминающие объекты* служат для задания условий моделирования, хранения, накопления и обработки информации, получение которой не предусмотрено стандартными средствами GPSS/H.

• *Объекты группирующей категории* содержат информацию о транзактах, находящихся в модели. Продвигаясь по модели, транзакты, имитирующие заявки на обслуживание, могут приводить к наступлению таких событий, как: поступление заявки в СМО; занятие (освобождение) места в накопителе; занятие (освобождение) канала обслуживания; прерывание обслуживания заявки с более низким приоритетом; совпадение значений определенных числовых атрибутов двух и более транзактов, называемых синхронизируемыми, и т. п. При этом соответствующие транзакты помещаются в один из пяти списков (цепей, в оригинале — chain):

Таблица 4.2. Соответствие английских и русских названий

Обозначение оригинальной версии языка	Обозначение, введенное в пособии	Примечания
BLOCKS —блоки	Операторы блоков (блоки)	См. прил. 1
Control Statements — инструкция управления	Операторы управления	См. прил. 2
Compiler Directive — директива компиляции	Операторы описания	См. прил. 2

— текущих событий (СТС) (время наступления меньше либо равно текущему модельному времени),

— будущих событий (СБС) (время наступления больше текущего модельного времени);

— прерываний (транзакты, обслуживание которых прервано);

— синхронизируемых транзактов (находящихся в состоянии сравнения);

— пользователя (транзакты, удаленные программистом из СТС).

Чтобы при дальнейшем изложении придерживаться единой терминологии, введем единое понятия оператора (табл. 4.2), которое отличается от дословного перевода терминов оригинальной версии языка.

Кроме указанных в табл. 4.2 операторов, для эффективного решения задач исследования системы необходимо иметь средства управления процессами модификации и отладки программ, сбора и обработки статистики и т. п.

В языке такими средствами являются команды, которые, в отличие от операторов, являются своеобразной надстройкой к языку и могут отличаться от версии к версии. Основное назначение команд — организация интерактивного взаимодействия с моделью на различных этапах моделирования.

§ 4.2*. СРАВНЕНИЕ GPSS/H С ДРУГИМИ ВЕРСИЯМИ

Основные отличительные черты GPSS/H рассмотрены в § 4.1, здесь кратко рассмотрим отличия ЯИМ от других версий GPSS (подробнее см. литературу [13]). Эти отличия можно разделить на три группы:

— общие отличия, расширяющие возможности ранних версий, но позволяющие моделировать программы, написанные на этих языках, на GPSS/H;

— частные отличия, расширяющие возможности ранних версий, но требующие определенной доработки;

— коренные отличия, не поддерживающие некоторые особенности других версий и заменяющие ряд устаревших возможностей.

Общие отличия

• *Логика языка*

1. Наличие мощной внутренней системы отладки (дебаггера), обеспечивающей остановку процесса ИМ в точках прерывания (break point) с выводом данных на дисплей, контроль продвижения транзактов с помощью различных методов выделения — трепирования (trap), запоминание анализируемой ситуации.

2. Введение нового вида переменных — амперпеременных (АМП — переменные, введенные только в GPSS /H от названия & — амперсенд, нотируются символом &), позволяющих повысить гибкость и мощность вычислений и представлять программы, написанные в других языках, за счет наличия внешних АМП.

3. Возможность легкого считывания и записи данных без выхода из программы, а также переименования объектов.

4. Расширение возможностей нотации: увеличение длины символов сверх восьми, применение любого числа скобок различного вида, снятие ограничений на число строк и столбцов матричной переменной, расширенное использование макросов, повышенное удобство перекрестных ссылок и словаря объектов.

5. Возможность использования встречных потоков БСВ — антитез.

• *Операторы управления и описания*

1. Применение операторов IF/ELSEIF, GOTO, HERE, DO, ENDDO придает большую гибкость при управлении процессом ИМ и создании отчета.

2. Возможность переназначения общей памяти за счет использования оператора REALLOCATE COMMON.

3. Возможность вывода на печать любых строк отчета за счет использования операторов LIST, UNLIST.

4. Улучшение читаемости отчета за счет использования оператора OVERCOL

• *Расширение операторов блоков*

1. Ряд ОУ могут быть записаны непосредственно в модуле исполнения МФ в виде ОБ, например BCLEAR, BRESET, BSTORAGE и т. д. (см. прил. 1).

2. Операнды ОБ могут быть представлены в виде аналитических выражений или записи закона распределения.

• *Расширение особенностей транзактов*

1. Расширен диапазон представления приоритета до $2^{31} - 10$.

2. Идентификационный номер (ИН — IN) транзакта используется только один раз, каждый новый транзакт получает ИН, следующий за уже использованным.

3. Транзакт может находиться в любом числе очередей (в GPSS V это число ограничено пятью).

4. Допускается задание времени транзакта числом с плавающей точкой (в русском написании — с плавающей запятой).

• *Расширение стандартных числовых и стандартных логических атрибутов (СЛА)*

1. Не существует ограничения на число потоков БСВ (по умолчанию число генераторов БСВ равно восьми).

2. СЛА могут использоваться вне оператора B VARIABLE и принимают два целочисленных значения 1 или 0, соответствующих истинному или ложному высказыванию.

3. СЧА допускают использование чисел с плавающей точкой.

Частные отличия

1. Использование нового улучшенного алгоритма для генератора БСВ (см. § 3.3).

2. Возможность использования отсчета времени в виде чисел двойной точности с плавающей точкой (имеющих как целую, так и дробную части), в то время как в других версиях используются только целые числа, причем отсчет времени может начинаться с 0 (в других версиях — с 1). Такое представление чисел используется везде, где встречаются числовые значения (разные операторы, СЧА, статистические данные).

3. Операнды ОБ GENERATE задаются только при поступлении в блок, а не дважды, как в GPSS V.

4. Компилятор GPSS/H допускает только одно наименование символа, причем одинаковое значение не может быть у нескольких символов (за исключением параметров транзакта), что исключает возможность разночтений.

5. Все контекстно-зависимые ссылки воспринимаются как часть системы полуавтоматического определения объектов. В GPSS V Ассемблер записывает только какую-то часть ссылок, поэтому в разных версиях эти последовательности не совпадают.

Коренные отличия (не поддерживаемые особенности других версий)

1. Размещение объектов в дополнительной памяти с помощью ОБ AUXILIARY. При включении этого ОБ возникает предупреждение, и его указания игнорируются.

2. Отсутствует опция REPLY и операнд B у ОУ SIMULATE.

3. Отсутствует PL/I HELP в отличие от GPSS V.

Примечание. У многих исследователей, чьи программы написаны на других версиях, может возникнуть желание переписать свои программы на GPSS/H в связи с его многочисленными преимуществами, описанными выше. Сделать это достаточно просто.

Устаревшие операторы

1. Оператор HELP в GPSS/H считается устаревшим, так как все действия понятны и без него, кроме того, этот оператор описан на другом языке, поэтому при подключении внешних подпрограмм всегда можно воспользоваться их HELP.

2. Исключен ОБ JOBTAPES, который в ранних версиях GPSS/H обеспечивал введение внешних потоков БСВ. Расширение возможностей входа—выхода сделало этот блок ненужным. Поэтому исключены и ОУ JOB, JOBTAPE.

3. Исключены некоторые операторы, имеющие отношение только к группе специфических пользователей, в частности Norden System.

4. Исключен оператор описания (ОО) ABS, свидетельствующий о том, что последующее сообщение кодировано в абсолютном формате, аналогично исключен ОО ENDABS, указывающий на окончание сообщения в абсолютном формате. В GPSS/H прямая связь с симулятором невозможна, поэтому все указания проходят через компилятор, в том числе и данные в абсолютном формате.

5. Исключены редко используемые указания типа ICT, ORG, REWIND.

6. Исключен оператор OUTPUT EDITOR, так как GPSS/H последней версии обладает целым набором операторов, формирующих выходной отчет. К их числу можно отнести REPORT, SPACE, GRAPH, COMMENT и ряд других.

§ 4.3. ОСНОВНЫЕ ПРАВИЛА РАБОТЫ С ПАКЕТОМ GPSS/H (СТУДЕНЧЕСКАЯ ВЕРСИЯ)

Студенческая версия GPSS/H дает полное представление о возможностях ЯИМ, способствует его осознанному изучению и позволяет приобрести необходимые навыки. Отличия от профессиональной версии заключаются:

• *в способе управления пакетом* — студенческая версия работает под MS DOS непосредственно или в любой оболочке типа NC, VC,

Far. В качестве рекомендации можно посоветовать не выходить в эмуляцию DOS, а работать в любой из указанных оболочек;

• *в размере МФ* — не более 125 ОБ, 250 операторов всех видов; при увеличении числа операторов появляется сообщение: ERROR: STUDENT VERSION IS LIMITED TO 125 BLOCKS;

• *в ограничении общей памяти* — не более 32 720 байт. Это ограничение особенно заметно при плохом выборе параметров входных потоков и потоков обслуживания, так как обычно используется только 10 000 байт памяти. При этом следует сообщение о переполнении памяти, которого не следует бояться (Error 411 -Out of COMMON. — Add/change REALLOCATE Stmt?), с просьбой изменить параметры, или переназначить объем памяти до 32 720 байт командой MAXCOM, или изменить значения в пределах больше 10 000 и до 30 000 командой REALLOCATE, имеющей синтаксис

< REALLOCATE COM, byte >

• *в стоимости пакета* — студенческая версия дешевле более чем на полтора порядка.

В остальном идеологии студенческого и профессионального пакетов не отличаются друг от друга.

При инсталяции дистрибутива необходимо проделать следующее:

• войти в сессию какой-либо командной оболочки;

• поставить дискету с дистрибутивом в дисковод;

• выбрать директорию для размещения GPSS/H;

• в меню запуска программы набрать: A:\ INSTALL;

• внести изменения в AUTOEXEC.BAT;

• перезапустить компьютер.

Запуск программы осуществляется путем набора в командной строке

< GPSSH filename.gps > Enter

filename означает имя МФ, выбранного из списка файлов, имеющихся в программе, или созданного исследователем с помощью редактора оболочки. После успешного моделирования появляется новый файл (листинг отчета) с тем же именем, но имеющий расширение .lis, который можно просмотреть, используя клавиши F3 или F4. При наличии в МФ ошибок компиляции появляется усеченный листинг, содержащий указания об ошибках, но не имеющий результатов моделирования. Исправления в МФ вносятся с помощью редактора оболочки в начальный МФ с расширением .gps, и процедура моделирования повторяется заново.

Предупреждение!

1. Несмотря на наличие в командной строке названия GPSS/H, после приглашения надо обязательно набирать имя программы GPSSH и после

пробела, имя МФ (либо использовать горячие клавиши Ctrl–j), находясь курсором на МФ.

2. Вновь создаваемому МФ обязательно присваивать расширение .gps.

3. Никогда не пытайтесь запускать программу файлом gpssh.exe!

После создания МФ и его записи в редакторе оболочки (формат записи см. § 4.4) процесс моделирования распадается на две фазы:

1) начало компиляции с появлением на экране сообщения “Pass 1 (with source listing)”, во время этой фазы считывается МФ, проверяются синтаксические ошибки, происходит нумерация строк МФ;

2) процесс компиляции с появлением сообщения “Pass 2 ...”, во время этой фазы МФ преобразуется в форму, удобную для исполнения, и производится распределение памяти. При отсутствии ошибок начинается моделирование, предваряемое сообщением “Simulation begins”.

Эти сообщения читаются в сеансе DOS, при моделировании в командной оболочке эти сообщения не появляются, процесс моделирования оканчивается практически сразу (время моделирования МФ из 100 операторов исчисляется миллисекундами), и в списке файлов появляется файл отчета .lis (листинг), содержащий данные либо о процессе моделирования, либо об ошибках. Никакой специальной команды для выхода из процесса моделирования при работе в командных оболочках подавать не надо.

Предупреждение! В процессе работы с программой никогда не вносите никаких изменений в имеющиеся файлы задач основного пакета. Если возникла необходимость изменения данных в задачах пакета, создайте свой файл, скопируйте в него желаемую задачу и только после этого проводите эксперименты с вновь созданным файлом, носящим присвоенное Вами имя.

Модельный файл создается в редакторе оболочки по клавише F4, файлу обязательно присваивать собственное имя с расширением *.gps, форматы записи МФ рассмотрены в § 4.4.

§ 4.4. СТРУКТУРА ОБЪЕКТОВ МОДЕЛИ

Транзакты. В модели эти динамические элементы обозначаются Хакт или просто Х с его ИН. ИН задается в порядке появления транзактов в модели; когда в МФ существует несколько источников транзактов, то назначение ИН не меняется (в порядке появления Хакт). Хакт в модели имеют разный смысл, они могут представлять собой людей или предметы, движущиеся по модели от одного ОБ к другому, причем типы людей или предметов могут быть разными в одной модели. Например, рассмотрим пункт таможенного досмотра на международной трассе. Через пункт следуют туристы или шоферы-профессионалы, автобусы, трейлеры, частные автомобили и т. д., однако ОБ модели обрабатывают все эти разные типы Хакт одинаково. Каждый Хакт имеет свой уникальный ИН, в начале моделирования в

модели нет Хакт, с началом моделирования Хакт время от времени вводятся в модель (рождаются), а также по прошествии какого-то времени, пройдя через последовательность ОБ МФ, Хакт выводятся из модели (терминируются, умирают). Каждый Хакт находится в исследуемый момент времени в *текущем* блоке (Current Block) и делает попытку войти в *следующий* ожидаемый блок (Next Block Attempted). При прохождении Хакт через последовательность ОБ МФ от момента зарождения до момента терминирования могут возникнуть следующие ситуации:

1) задержка Хакт при входе в ОБ (например, ADVANCE);

2) препятствие входу Хакт (блокирование) в следующий ОБ (например, SEIZE);

3) уничтожение (терминирование) Хакт при попадании в ОБ TERMINATE;

4) возвращение Хакт к началу МФ при появлении ОБ TRANSFER вместо ОБ TERMINATE.

Две последние ситуации не могут возникать одновременно; так, терминирование происходит при выходе Хакт из системы (обслуженный покупатель, посланное сообщение, прошедший пункт контроля автомобиль и т. д.). В случае, когда Хакт не покидает систему, а возвращается к началу, используется ОБ TRANSFER, изменяющий направление движения (например, клерки в нотариальной конторе представляют собой Хакт, но, подготовив один документ и подписав его у менеджера, возвращаются на рабочее место для подготовки нового документа). Как только Хакт прекращает движение (две первые ситуации), сразу начинается движение следующего транзакта, становящегося активным. Отсюда следует правило, которое неуклонно соблюдается при моделировании: **В каждый момент времени в модели движется только один Хакт.** Порядок движения транзактов в модели подробно рассмотрен в гл. 5.

Хакт может иметь один или больше атрибутов (атрибут — характеристика, относящаяся к отдельному транзакту: цвет, тип, число покупок и т. д.). Одни из атрибутов вводятся программой, например ИН, другие назначаются исследователем, например тип движущегося по дороге транспортного средства. В случае, когда надо рассмотреть людей и предметы, представляемые транзактами, то, скорее всего, в МФ надо их представить отдельными цепочками ОБ.

Ресурсы. В отличие от транзактов ресурсы (устройства, памяти) — неподвижные, постоянные объекты, которые за время одного прогона ИМ остаются неизменными. Ресурсы используются транзактами для решения задачи, определенной исследователем. Хакт соревнуются между собой за использование ресурсов, ожидают их освобож-

дения предыдущим Хакт, для реализации задачи моделирования могут потребовать несколько разных ресурсов. Ресурсы также могут быть людьми (число кассиров в банке) или предметами (станок, автомат на АЗС), возможно сочетание ресурсов в одной модели.

Каждая из групп операторов, применяемых в МФ, имеет свой набор характеристик, определяющих их индивидуальные особенности.

• Операторы блоков характеризуются размещением, действием (названием совершаемой операции) и операндами.

Размещение — каждый ОБ занимает предназначенную ему позицию и нумеруется самой программой, начиная от 1 и далее в порядке их появления в МФ. Исследователь не имеет возможности присваивать номера ОБ, но при необходимости может давать имя (присваивать ярлык) ОБ одинакового названия для их четкого различения. Особенно это важно при изменении последовательного движения транзактов или при возвращении к уже пройденному участку МФ.

Код действия — каждый ОБ обозначается ключевым словом, уточняющим характер операции, совершаемой ОБ в процессе ИМ. Ключевые слова типа GENERATE, TERMINATE, ADVANCE, SEIZE (см. прил. 1) обязательно пишутся прописными буквами и могут быть сокращены до четырех символов, например, GENE, TERM и т. д. Ошибки в написании ОБ приводят к появлению в листинге сообщений об ошибках компиляции, и процесс моделирования не начинается.

Операнды — каждый ОБ имеет от одного до нескольких операндов, дающих информацию, на которой основано действие ОБ. Операнды обозначаются A, B, C... . Отсутствие операнда на назначенном ему месте определяется значением по умолчанию, чаще 0, а иногда ∞ (будет оговорено при рассмотрении конкретных ОБ в гл. 5). Нередко бывают ситуации, когда часть операндов задается в явном виде, а часть используется по умолчанию.

• Операторы управления и описания. Структура этих операторов схожа со структурой ОБ, только вместо номера строки размещения первой характеристикой является ярлык (label) — сравните с возможностью приписывания ярлыка для ОБ. Вторая и третья характеристики совпадают по смыслу (код действия и операнды). Рассмотрим отличия в назначении этих характеристик.

Ярлык для ОУ в некоторых случаях требуется, а в некоторых является ненужным, а для некоторых ОУ и ОО он просто не предусмотрен (подробнее см. прил. 2 и текст дальнейших глав).

• *Код действия* — каждый ОУ обозначается ключевым словом, определяющим результат исполнения этого ОУ; ОО также представ-

ляет ключевое слово, несущее информацию при трансляции модели, но не влияющую на процесс моделирования.

Операнды — ОУ имеют нуль или несколько операндов. Они также обозначаются A, B, C... и для определенных типов операторов задаются в явном виде, а в большинстве применяются по умолчанию.

Кроме названных операторов в ЯИМ используются указания по созданию отчета (EJECT, INCLUDE и др.). В МФ часто включаются и комментарии, предоставляющие возможность стороннему пользователю понять смысл МФ; хотя комментарии не обязательны, их применение весьма полезно. Следует отметить, что комментарии можно писать на русском языке.

Каждому члену категории или типа объектов приписывается ряд атрибутов, которые разделяются на три разновидности – стандартные числовые атрибуты (SNA), стандартные логические аттрибуты (SLA) и стандартные символьные атрибуты (CCA—SCA). Большая часть этих атрибутов автоматически назначается симулятором, и в процессе моделирования на них производятся ссылки, обеспечивающие логику исполнения модели. Однако существует определенное количество СЧА, не относящихся к конкретному объекту, часть из них описывает статус системы, например AC1 — абсолютное время или TG1 — счетчик свершений или терминирований, часть из них характеризует встроенные функции, например RVEXPO(.), RVNORM(.), а часть соединена с транзактами (как было указано выше). CCA появились в GPSS/H в связи с появлением трех новых переменных — символьных АМП: LEN, SSG, SYM.

Рассмотрены практически все элементы GPSS/H (не рассмотрены понятия групп и списков (chain) — это сделано сознательно, так как в занятиях со студентами практически не удавалось дойти до этих представлений. Для продвинутых пользователей операторы, относящиеся к этим объектам, даны в приложениях).

Приведя начальные данные по структуре объектов, перейдем к структуре МФ, которую можно представить в виде трех укрупненных модулей (рис. 4.2) (жирным шрифтом выделены операторы, без которых процесс ИМ не может быть осуществлен).

Рассмотрим более подробно отличительные черты этой структуры.

1. Всегда, в любом случае на первом месте модуля задания стоит ОУ SIMULATE, который дает команду на компиляцию МФ; отсутствие этого ОУ приводит к машинной ошибке, и процесс ИМ не начинается.

2. ОУ START дает команду на исполнение МФ, при этом счетчик свершений (СС) устанавливается в начальное значение, инициали-

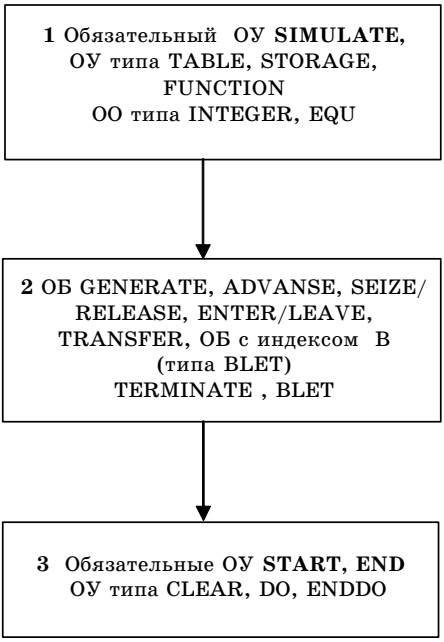


Рис. 4.2. *Укрупненная структура МФ: 1 — модуль задания; 2 — модуль исполнения; 3 — модуль управления*

зируется ОБ GENERATE, т. е начинают поступать транзакты, которые движутся последовательно блок за блоком.

3. ОУ END, стоящий обязательно последним в МФ, прекращает процесс ИМ, выполняя две операции: сигнализирует о том, что МФ физически кончился, и, прекращая исполнение МФ, дает команду о возвращении в командную оболочку.

Итак, любой МФ состоит из этих трех обязательных модулей.

Модуль задания (описания) может включать кроме обязательного SIMULATE, стоящего на первом месте, необходимое для процесса ИМ число ОУ и ОО. Порядок их расположения в принципе не важен, так как симулятор располагает их в необходимой последовательности. Следует учесть, что операторы модуля описания *не исполняются, а лишь задают параметры и структуру модели*.

Модуль исполнения включает необходимое число исполняемых ОБ. В модуле исполнения на первом месте стоит ОБ GENERATE; если в МФ предусмотрено исследование нескольких типов транзактов, то в каждой из параллельных ветвей МФ предусмотрен свой ОБ GENERATE; при использовании временного таймера также исполь-

зуется ОБ GENERATE. Число ОБ в модуле исполнения зависит от сложности и логики модели. Этот модуль является главным, и в результате его исполнения собирается вся необходимая информация.

Модуль управления начинается с ОУ START и заканчивается ОУ END, между ними может быть необходимое число ОУ, а в некоторых случаях и ОБ GENERATE. Операторы этого модуля не исполняются, а задают команды на выполнение действий с модулем исполнения.

Примечание. Порядок построения и действия этих модулей будет в дальнейшем проиллюстрирован примерами. Для лучшего восприятия МФ при текстовом написании каждый модуль может иметь собственное имя (допускается написание имени на русском языке).

§ 4.5. ФОРМАТ ЗАПИСИ МОДЕЛЬНОГО ФАЙЛА

Общие правила записи для символов, имен, чисел

1. Символы и имена могут включать в себя от одного до восьми буквенно-цифровых знаков (ярлыки — от одной до шести литер), на первом месте обязательно должна быть буква. Первый символ не должен повторять первой буквы СЧА, СЛА или ССА, предшествующей цифре.

Правильно	Не правильно	
CPU	X125A	X — является СЧА
B14	1BETA	Первая литера — число
LONGNAME	TOOLONGNAME	Больше 8 литер
A1B2	SYM#BOL	Литера не буквенно-цифровая

2. Целые числа имеют впереди не обязательные символы + или – и могут содержать от 1 до 10 десятичных знаков. Максимальное значение $2^{31} - 1 = 2\,147\,483\,647$, минимальное значение $-2^{31} = -2147483648$.

3. Числа с плавающей точкой имеют следующий формат:

< не обязательные + или –, от 0 до 10 знаков — целая часть и от 0 до 10 знаков —дробная часть >.

При этом целая или дробная части могут по отдельности равняться 0, но никогда вместе. Хотя целая и дробная части могут содержать по 10 знаков, на самом деле принцип двойной точности позволяет использовать только 16 знаков.

Правильно	Не правильно
1.5	1.5.1 — более одной точки
.1	.1 +7 — смешаны знаки

4. Символьные константы отделяются кавычками “/”; если необходимо включить текст, то он выделяется одинарными кавычками. Использование символов без кавычек является ошибкой.

Правильно	Не правильно
‘ UPPER CASE ‘	‘No end — отсутствует кавычка в конце,
‘ LOWER CASE ‘	...’’ — не сбалансировано

5. Названия действий операторов всегда пишутся большими латинскими буквами.

Правильно	Не правильно
ADVANCE	advance
TERMINATE	ТЕРМИНЕЙТ

6. Фиксированная длина записи — обычно редакторы оболочек имеют 80 колонок, тогда используются первые 72 колонки, так как оставшиеся 8 содержат ИН транзакта, сам пакет GPSS/H позволяет работать с длиной строки редактора, равной 132 колонкам. Для автоматического форматирования строки можно использовать клавишу TAB, функции которой поддерживаются всеми IBM совместимыми версиями. Функция TAB работает в 9, 17, 25, 33, 41, 49, 57, 81, 89, 97, 105, 113, 121, 129 колонках.

7. Запись функций — задается в модуле задания OO FUNCTION формата

< n FUNCTION A,B>

n — номер или имя функции, A — операнд, указывающий на аргумент функции, обычно СЧА, B — тип функции (Dm, Cm) — дискретная или непрерывная с числом точек табуляции m. Функция может быть записана в фиксированном формате, что практикуется при передаче данных в программы, записанные не на GPSS, но чаще используется свободный формат — в нем нет ограничений на разрядность числа. Правила записи сводятся к следующему:

— запись начинается с 1-й колонки и следует не далее 71-й;

— координаты точки разделяются запятой, а пары чисел — слэшем ‘/‘;

— X записываются в порядке возрастания.

Пример. 1 FUNCTION RN1.D5
0,0/.5,26.0/.83,40.8/.89,6.08/.995,5.0

8. Запись СЧА и СЛА — они могут быть встроены в конструкцию языка или вводиться в процессе моделирования в любое время и в любом месте указанием имени СЧА-СЛА.

• Ссылка на постоянную величину позволяет ввести целочисленное значение в имя желаемого СЧА, например: XI означает полнословную сохраняемую величину с номером, равным 1.

• Язык позволяет делать записи, используя нотацию, принятую в ЯИМ, например: X(PH(X1+3)+2) или MH(XI + 7,1,2), последний пример обозначает полусловную матрицу, номер которой равен содержимому полнословной ячейки 1 плюс 7, с одной строкой и двумя столбцами.

• Запись символических имен. При одновременной записи символического и мнемонического обозначений последнее отделяется символом $, например: XF$TERM — полнословная ячейка с именем TERM.

• Запись АМП. Они следуют сразу за именем СЧА, например: F (&I) — устройство, номер которого задан амперпеременной с именем I. Все АМП служат для увеличения вычислительной мощности ЯИМ и связи с внешними источниками информации и заявляются до их первого появления в модели, т. е. в модуле задания. Рассмотрим разницу между записями:

SEIZE FRED — имя остается постоянным;

SEIZE &SAM — имя может меняться, о чем говорит символ АМП.

Имя АМП обязательно вначале имеет символ &, а затем буквенно-цифровую комбинацию от одного до восьми знаков, первый из которых обязательно буква, причем эта буква может совпадать с наименованиями СЧА и СЛА. Для придания АМП числового значения используется ОУ LET, например:

```
INTEGER &SAM
LET &SAM=3.
```

Существуют 5 типов АМП:

целочисленные	INTEGER
действительные	REAL
символьные	CHAR*n
переменные символьные	VCHAR*n
внешние	EXTERNAL

Все типы АМП могут быть скалярами или одноразмерными матрицами, например:

INTEGER &I,&J,&K(100) — первые два члена определяют скалярные АМП, а последний — матрицу размером 100; все значения до начала ИМ равны 0;

REAL &X,&Y,&Z(100) — первые два члена определяют действительные скалярные числа, а третий — матрицу размером 100;

все значения до начала ИМ равны 0;

CHAR*5 &STRING,&X(10) — первый член определяет скалярный символьный АМП с фиксированной длиной 5 знаков, а второй — матрицу размером 100, каждый член которой имеет 5 знаков.

Задание АМП осуществляется с помощью ОУ LET или DO, а также ОБ BLET.

9. Контекстно-зависимые ссылки позволяют осуществлять специальные действия компилятора при адресации посредством операндов блоков, СЧА или операторов описания и управления.

• Назначение величин символов. Каждый символ, используемый в модели, должен иметь какую-то величину. Компилятор не распознает значения символов, конфликтующих с постоянными ссылками. Следовательно, надо избегать одинаковых символов для различных объектов.

• Автоматическое перераспределение объектов. При моделировании число объектов какого-либо типа, заданных командой SIZE, может быть изменено командой REALLOCATE

• Представление значений транзакта. Язык поддерживает 4 формата представления значения транзакта: B — байт $\pm(2^7 - 1)$, H — полуслово $\pm(2^{15} - 1)$, F — слово $\pm(2^{31} - 1)$, L — плавающая точка $\pm$ $\pm (2^{24} - 1)$, — причем транзакты могут иметь более одного параметра в разных форматах. Уточнение достигается за счет помещения суффикса формата в операнд блока (см. прил. 5). При буквенном обозначении параметра ставится разделитель $, например: MARK TIMEIN$PH. При цифровом обозначении разделитель не используется: MARK 1PH.

• Макрокоманды (макросы). Если в модели какие-либо инструкции повторяются часто, полезно использовать макросы. При этом вначале необходимо определить макрос по правилам определения операторов, а затем вызвать командой MACRO. Начало определения задается командой STARTMACRO со своими операндами (в пособии не рассматриваются), а окончание определения задается командой ENDMACRO.

Формат записи операторов

В GPSS/H допускается два формата записи операторов — фиксированный и свободный.

Фиксированный формат представлен на рис. 4.3 (для 80-колонкового редактора).

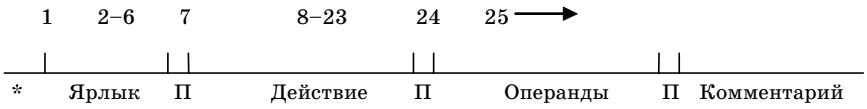


Рис. 4.3. *Схема фиксированного формата (индекс П означает обязательный пробел, с 12-й по 68-ю колонку может применяться действие OPERCOL)*

1. В первой колонке может быть поставлен знак *, тогда эта строка не читается симулятором и воспринимается как строка комментариев. В этой строке можно писать любые сообщения, в том числе на русском языке, например: «Модуль задания для модели банка», или продолжать длинный текст комментариев предыдущей строки.

2. Ярлык состоит из 6 буквенно-цифровых знаков, некоторые ОУ могут иметь только цифровой ярлык, ОБ никогда не имеют цифровой ярлык. Появление одинаковых ярлыков для разных объектов недопустимо. Также нельзя допускать, чтобы ярлык совпадал с именем оператора или кодом операции. После написания ярлыка следует пробел до 8-й колонки. У ОУ ярлык может быть, например ОУ TABLE, а может и отсутствовать; так, он может быть лишним в ОУ STORAGE.

3. Код действия является ключевым словом для ОБ, ОУ и ОО и начинается с 8-й колонки. Он может быть сокращен до 4-х первых букв, однако для лучшей читаемости МФ сокращениями надо пользоваться осторожно и редко. Для новых ОБ, начинающихся на букву B, такие сокращения просто недопустимы, например: BGETLIST, BGETSTRING. Вспомогательные коды в силу их краткости не могут быть сокращены. После кода действия обязательны пробелы до значения OPERCOL.

Здесь целесообразно дать определение оператору OPERCOL (operand start column), который назначает колонку, с какой начинается написание операндов в обоих возможных форматах записи. Значение этого оператора по умолчанию — 25-я колонка, однако это можно переназначить, написав

< OPERCOL n >

где n может принимать значения от 10 до 60. Значение этого оператора больше 25 оправдано, когда используется петля DO, сдвигаемая вправо. Этот оператор равно применим в обоих типах форматов.

4. Операнды. Их запись начинается с колонки, определенной OPERCOL. GPSS/H позволяет кодировать операнды в виде аналитических выражений, что придает ЯИМ большую гибкость. В случае, когда выражения имеют большую длину, запись операндов возможно производить на следующей строке, предваряя запись символом " _ ", что воспринимается симулятором как продолжение предыдущей строки. Когда рассматривается последовательность объектов, то в этом

операнде последовательность объектов пишется через "–", например: FUNAVAIL 1–5. В написании операндов можно использовать скобки. Сами операнды A,B,C,... пишутся через запятую без пробелов. Отсутствие какого либо операнда отмечается двумя запятыми без пробела, например:

SAVEVALUE (PH3 – 5) – (PH3 + 5),25,XH

GENERATE RVEXPO(1,2),,10,,5

После написания последнего операнда обязателен как минимум один пробел, а при наличии макроса (оператор MACRO) — как минимум два пробела.

5. Комментарий. При отсутствии у кода действия операндов запись комментариев возможна в любой колонке, начиная с OPERCOL + + 1 до 72-й колонки 80-колонкового редактора или до конца строки при 132-х колонках. При необходимости написать более длинные комментарии переход на новую строку предваряется *. Комментарий имеет смысл писать всегда для лучшей читаемости как МФ, так и выходного отчета.

Свободный формат

1. Ярлык может начинаться в 1-й или 2-й колонке и иметь от одного до восьми знаков.

2. Код операции отделяется от ярлыка одним или бóльшим числом пробелов. При отсутствии ярлыка код операции может начинаться с 3-й колонки, но не попадать в поле OPERCOL.

3. Операнды начинаются через один или большее число пробелов и пишутся до или непосредственно в поле OPERCOL.

4. Комментарий пишется через один или два пробела (при наличии макросов); если код операции не имеет операндов, комментарии начинаются с колонки OPERCOL + 1 и продолжаются до конца строки.

Далее в тексте будет использоваться только фиксированный формат, так как его отображение в МФ имеет стандартный вид.

Глава 5
ПРИНЦИПЫ ФУНКЦИОНИРОВАНИЯ ЯЗЫКА ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ GPSS/H

§ 5.1. ХАРАКТЕРИСТИКИ ОСНОВНЫХ ОПЕРАТОРОВ

Прежде чем переходить к построению МФ, необходимо разобраться с рядом важных обстоятельств, первыми среди которых являются принципы действия операторов. Всего в последней версии GPSS/H используется 115 операторов: ОБ — 57, ОУ — 34, ОО — 17, ОВО (операторы выходного отчета) — 7. Количество операторов дано по последней версии GPSS/H, в это количество не включены 29 устаревших операторов, применяющихся в других вариантах ЯИМ (см. § 4.2).

Естественно, что в рамках одного параграфа трудно рассмотреть особенности всех операторов, поэтому подробно рассмотрены только те операторы, которые будут использоваться в дальнейшем в тексте пособия при описании примеров, условно назовем их основными или, точнее, наиболее часто употребляемыми. Среди основных операторов в обязательном порядке рассматриваются и те, без которых невозможно создать любую модель. Характеристики остальных операторов сведены в приложения. В отличие от ряда источников [12, 16] для каждого рассматриваемого оператора его операнды приводятся не по мере возникшей необходимости, а сразу все. При этом следует иметь в виду, что некоторые операнды при дальнейшем изложении могут не использоваться.

Вторым важным обстоятельством являются правила окончания испытаний, которые вызывают затруднение у лиц, впервые изучающих основы GPSS/H.

Третьим важным обстоятельством являются принципы движения транзактов. Хотя этим движением управляет сама программа, но понимание этого процесса значительно расширяет кругозор пользователя. То же самое можно сказать и о работе со списками. Несмотря на то, что §§ 5.3 и 5.4 отмечены *, их следует обязательно разобрать после уяснения основ GPSS/H.

Все эти вопросы последовательно рассматриваются в гл. 5.

5.1.1. Операторы блоков[1]

Операторы блоков будут описываться не по алфавиту, а по степени важности оператора.

А. ОБ, создающие, задерживающие и выводящие транзакты из модели

А1. ОБ GENERATE (Произвести)

Этот ОБ призван последовательно производить и вводить Хакт в его следующий блок один за другим, как это предписывается его операндами. Необходимо ввести понятие времени моделирования и времени между последовательными появлениями Хакт. Время моделирования меняется непрерывно от 0.0 до заданного предела (см. § 5.2), время последовательного появления представляет собой промежуток времени входа очередного Хакт от момента входа предыдущего (рис. 5.1).

Время последовательного появления случайно, за исключением единственной ситуации, когда оно детерминировано (A = const, B = 0). Случайный характер времени последовательного появления задается исследователем по его желанию, на основе законов распределения, встроенных в ЯИМ.

Оператор блоков GENERATE относится к числу особых операторов, которые имеют только выход, попытка войти в GENERATE приводит к появлению сообщения об ошибке и прекращению процесса моделирования (см. § 8.5, упражнения). Моделирование нескольких потоков событий, замыкающихся на одно устройство обслуживания, в GPSS/H решается весьма просто — созданием нескольких параллельных цепочек со своими ОБ GENERATE. Достаточно обратиться к классическому примеру Т. Шрайбера о желающих подстричься и/или побриться у одного и того же мастера. Формат ОБ GENERATE имеет вид

< GENERATE A,B,C,D,E,F,G,H,I >

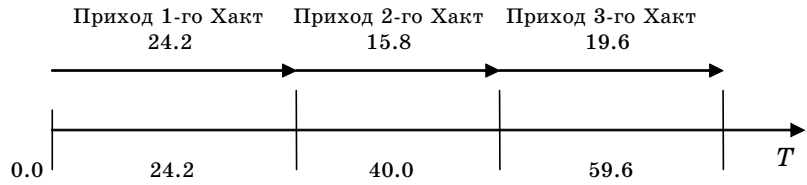


Рис. 5.1. *Иллюстрация последовательного появления Хакт во времени*

[1] Полный список ОБ см. прил. 1.

Все операнды A–I могут ссылаться на любые СЧА или СЛА, не ориентированные на транзакты, так как транзактно-ориентированные атрибуты могут появиться только после появления транзакта из ОБ GENERATE.

A — значение по умолчанию 0, определяет среднее время последовательного появления Хакт. При желании можно задать эту величину по какому-либо закону распределения. Тогда в поле операнда A вписываются параметры этого распределения, например: RVNORM (n, μ, σ), которое читается следующим образом: БСВ, распределенная по нормальному закону, взятая с генератора n, с параметрами математического ожидания μ и стандартного отклонения σ.

B — значение по умолчанию 0, характеризует отклонение относительно среднего значения, заданного операндом A. Если операнд B не задан функцией (что возможно, но используется редко), то каждое время последовательного появления вычисляется из выражения $A \pm B$. Значение B может быть равным A или меньше, но никогда не больше A, так как в этом случае время становится отрицательным и появляется сообщение об ошибке симуляции. Значение B, равное нулю, обязательно задается в двух случаях:

— когда значение времени последовательного появления детерминировано;

— когда операнд A задан из выражения типа RVEXRO, RVNORM и т. д.

Факт задания значения любого операнда по умолчанию фиксируется второй запятой (без пробела) на месте операнда. Если операнд B задается функцией, конструируемой пользователем (см. § 8.1), то время последовательного появления вычисляется путем умножения значения операнда A на значение функции операнда B, при этом A является целым числом, B может отображаться дробным числом, результат округляется до целого числа.

C — значение по умолчанию 0, при использовании операнда C задается время появления первого Хакт, отличное от нуля, таким образом C определяет сдвиг точки отсчета времени моделирования. Если, например, C = 6.75, то оно откладывается не от нуля, а от точки 6.75, с которой начинается отсчет времени последовательного появления Хакт.

D — значение по умолчанию ∞, когда появляется операнд D, то он задает число Хакт, производимых данным GENERATE. Такую ситуацию не трудно представить в любом применении ИМ, например: система состоит из 3-х приборов, после изготовления 3-го поступление приборов прекращается и цепочка в МФ перестает функционировать. Изменить значение операнда D в процессе моделирования нельзя,

поэтому ограничить число Хакт можно и другим способом, поместив после ОБ GENERATE ОБ GATE в положении запрета.

E — значение по умолчанию 0, определяет уровень приоритета Хакт, уровень приоритета может меняться в пределах от – 2 147 483 632 до + 2 147 483 632.

F–I — значение по умолчанию 12PH (полусловный параметр). Напомним, что в ЯИМ используются 4 типа представления числа: F — полнословный (Fullword), H — полусловный (Halfword), L — с плавающей точкой (Floating point), B — байтовый (Byte). Указанные операнды определяют число параметров любого типа для произведенного транзакта, они кодируются суффиксами, поэтому порядок их написания может быть любым. Хакт может иметь от ноля до 255 параметров каждого типа. В дальнейшем в пособии эти операнды использоваться не будут.

Приведем несколько примеров написания ОБ GENERATE.

```
1.  GENERATE   4.1
2.  GENERATE   4.1,,3
3.  GENERATE   4.1,2.1,,10,10
4.  GENERATE   RVEXPO(2,8.5),,,,5
5.  GENERATE   ,,,,4
```

В первом примере транзакты поступают детерминированно последовательно каждые 4.1 единицы времени без каких-либо ограничений, т. е. в 4.1, 8.2, 12.3 и т. д.

Во втором примере транзакты поступают точно так же, но существует временной сдвиг прихода первого транзакта, т. е. время равно 7.1, 11.2, 15.3, и т. д.

В третьем примере транзакты поступают по равномерному распределению и время определяется из выражения A ± B, т. е. в любой момент от 2-х до 6.2 число таких транзактов равно 10, а их приоритет равен 10.

В четвертом примере операнд A задан выражением, говорящим о том, что время распределено по экспоненциальному закону, БСВ берется со 2-го генератора с параметром потока $\lambda = 8.5$, операнды B, C, D отсутствуют, приоритет равен 5.

Пятый пример говорит о том, что в системе существуют 4 транзакта (в гл. 6 будет рассмотрен пример работы нотариальной конторы, когда Хакт представляют собой клерков, не покидающих систему).

А2. ОБ TERMINATE (Уничтожить)

Этот ОБ удаляет (разрушает, терминирует) Хакт из модели. Он принадлежит также к числу особых ОБ и никогда не имеет выхода, наличие выхода вызывает машинную ошибку. Когда Хакт подходит к ОБ

TERMINATE как к своему следующему блоку, то он никогда не встречает препятствий, проходит в ОБ TERMINATE без задержки и разрушается, а в некоторых случаях не влияет на изменение параметров модели (в частности, показаний счетчика свершений, см. § 5.2), а только фиксирует факт прохода через TERMINATE. В МФ может быть любое число ОБ TERMINATE. Формат ОБ TERMINATE имеет вид

< TERMINATE A >

A — по умолчанию 0, операнд всегда является целым положительным числом, указывающим на изменение показания счетчика свершений, именно на значение операнда A (подробнее см. § 5.2).

Приведем несколько примеров использования ОБ TERMINATE.

1. TERMINATE 1
2. TERMINATE 5
3. TERMINATE [0]

В первом примере декремент равен 1.

Во втором примере декремент равен 5, что достаточно просто представить практически, например: крупная сделка занимает при оформлении одно и то же время, но доход приносит в 5 раз больше.

В третьем примере A по умолчанию равно 0, т. е. показания счетчика не меняются, а сам Хакт выходит из системы. Для лучшей читаемости МФ рекомендуется значение 0 ставить на место операнда A. Скобки [ ] говорят о необязательности этого действия.

А3. ОБ ADVANCE (Продвинуть)

Этот ОБ предназначен для моделирования времени задержки в приборе обслуживания и может использоваться либо самостоятельно, либо с сопутствующими парными блоками при одноканальном обслуживании SEIZE /RELEASE, PREEMPT/ RETURN и при многоканальном обслуживании ENTER/LEAVE. Одиночный ОБ ADVANCE легко представить в виде времени задержки при подходе к операционному окну банка, транспортировке детали по транспортеру и т. д. Если используются последовательно несколько ОБ ADVANCE, то нельзя суммировать их время и заменять общим оператором, так как времена задержки в большинстве случаев являются случайными. Формат ОБ ADVANCE имеет вид

< ADVANCE A,B >

A — по умолчанию 0, этот операнд аналогичен операнду A ОБ GENERATE и может использовать встроенные функции.

B — по умолчанию 0, аналогичен операнду B ОБ GENERATE.

Примеры использования ОБ ADVANCE не приводятся, так как они аналогичны примерам с операндами A и B для ОБ GENERATE.

Оператор блоков ADVANCE никогда не препятствует входу Хакт, поэтому в нем могут находиться несколько транзактов, один из которых обслуживается, а другие ожидают разрешения на вход в следующий блок после окончания времени их обслуживания. При поступлении в ОБ ADVANCE с нулевыми значениями операндов A и B (время задержки на обслуживание равно 0) Хакт немедленно проходит через ADVANCE и пытается войти в следующий для него блок. Такой ОБ называется фиктивным (dummy) ADVANCE. В сложных моделях ОБ ADVANCE может служить линией раздела модели (см. гл. 6).

Б. ОБ, реализующие обслуживание объектов

Б1. ОБ SEIZE/RELEASE (Захватить / Освободить)

Эти два парных ОБ предназначены для одноканального обслуживания устройств (Facility), вначале захватываемых Хакт, а после обслуживания освобождаемых им. Если устройство захвачено (SEIZE), или обслуживание прервано (PREEMPT) другим транзактом, или устройство не готово к действию посредством ОБ FUNAVAIL, то ОБ SEIZE препятствует входу Хакт, и Хакт начинает движение только после снятия блокировки. ОБ SEIZE принадлежит к небольшому числу ОБ, препятствующих входу Хакт, это: SEIZE, ENTER, PREEMPT, GATE, ATNWAIT. Исполнение ОБ RELEASE снимает блокировку. Таким образом ОБ SEIZE осуществляет контроль над имеющимся ресурсом. В модели может быть много различных устройств, каждое из которых связано со своим ОБ SEIZE, имеющим уникальный символ идентификации, поэтому формат ОБ SEIZE и ОБ RELEASE имеет вид

< SEIZE A >

< RELEASE A >

A — не имеет значения по умолчанию, поэтому отсутствие записи в поле операнда воспринимается как ошибка, операнд представляет собой имя или номер используемого устройства. Подчеркнем, что у обоих ОБ операнд A должен быть абсолютно идентичным (какое устройство занял, то и освободил), кроме того, операнд A не должен носить имя кода операции, иначе программа будет вынуждена задействовать большие объемы памяти для распознавания их отличий. Несмотря на то, что можно нумеровать устройства (операнд A является целым положительным числом), методически правильнее давать устройству имя, например: если устройство — клерк в банке, то следует присвоить имя CLARKA, какому-то следующему — CLARKB и т. д. Устройство при функционировании модели переходит от состояния готовности (не занятости — on-duty) к состоянию не готов-

ности (занятости — off-duty) и обратно (порядок снятия запрета и перехода транзакта из списка в список будет рассмотрен в § 5.4.)

Рассмотрим несколько примеров.

```
1.  SEIZE      SERVER
2.  SEIZE      1
3.  SEIZE      SERVER
    ADVANCE    1.1,0.6
    SEIZE      CLARK
    ADVANCE    3.5,2.3
    RELEASE    CLARK
    RELEASE    SERVER
```

В первом примере захватывается устройство SERVER.

Во втором примере — устройство за номером 1.

Третий пример представляет фрагмент программы, состоящий последовательно из двух устройств, обратите внимание на порядок освобождения устройств.

Б2. ОБ PREEMPT /RETURN (Прерывать / Возвращать)

Эти парные ОБ служат для прерывания занятости устройств Хакт с большим приоритетом, нежели приоритет обслуживаемого транзакта. Указанные ОБ используются достаточно редко в моделях большой сложности и в пособии не рассматриваются.

Б3. ОБ ENTER/LEAVE (Входить / Покидать)

Эти два парных ОБ предназначены для многоканального обслуживания памятей, т. е группы идентичных обслуживающих устройств. При этом Хакт поступает в точку модели, где находится группа серверов, и запрашивает сервер для обслуживания. Если все серверы заняты, то транзакт ожидает, затем при появлении возможности поступает на обслуживание, обслуживается, а затем покидает сервер. Все эти операции поддерживает связка ОБ ENTER — ADVANCE — LEAVE. ОБ ENTER /LEAVE и ОБ SEIZE /RELEASE аналогичны по принципу своего действия, только первые обслуживают памяти и для памятей отсутствует функция захвата (PREEMPT), которая существует в устройствах. В остальном идеология функционирования совпадает; так, ОБ ENTER может препятствовать входу Хакт (блокировать), а ОБ LEAVE после исполнения снимает блокировку. Имена обоих ОБ должны быть идентичными, т. е. идентифицировать одну и ту же память, имя памяти не должно повторять название кода операции и т. д. Формат этих ОБ имеет вид

```
< ENTER A,B >
```

A — не имеет значения по умолчанию, поэтому отсутствие записи в поле операнда воспринимается как ошибка, операнд представляет собой имя или номер памяти, в которую Хакт пытается войти. Операнд A не может представлять собой выражение.

B — по умолчанию имеет значение 1, определяет собой число единиц памятей, затребованных Хакт. Если такого числа памятей нет в наличии, то вход Хакт блокируется до изменения статуса памяти. Причем управление памятью со стороны блока исключается до тех пор, пока не будет свободно требуемое число памятей.

Приведем несколько примеров записи ОБ.

1. ENTER BUFFER
2. ENTER TOOLS,2

В первом примере Хакт требуется одна единица памяти по имени BUFFER.

Во втором примере Хакт потребно две единицы памяти TOOLS.

Примеры для второго парного блока не приводятся, так как они аналогичны.

Между парными блоками может стоять не только один ОБ ADVANCE, но и несколько таких ОБ, а также ОБ, относящиеся к устройствам или к изменению направления движения Хакт. Такая структура будет зависеть от логики и сложности модели и не является неверной (в гл. 6 эти соображения будут проиллюстрированы примерами).

Б4. Логический переключатель

Эта комбинация ОБ LOGIC (Проверить логику) — GATE (Запирать) и ОУ INITIAL может находиться в одном из двух состояний «включено — выключено» и служит для сигнализации о состоянии моделируемой системы по отношению к сложившимся условиям. По терминологии GPSS/H, логический переключатель может быть в установочном состоянии (set) — код состояния LS, либо свободен (clear) — код состояния LC. ОУ INITIAL задает начальное состояние или изменение состояния логического переключателя, а также сохраняемых величин (Savevalue) и матричных величин (Matrix Savevalue), его формат имеет вид

< INITIAL A(n)/ A(n)/...>

A — не имеет значения по умолчанию и обозначает один из кодов состояния LS, LC, содержимое скобок указывает на имя инициализируемого устройства, слэш "/" символизирует возможный набор устройств, например:

INITIAL LS(KEY)/LC(KNOB)

В этом примере начальное состояние логического переключателя KEY — в положении “установить”, а начальное состояние логического переключателя KNOB — в положении “свободен“.

ОБ LOGIC изменяет состояние логического переключателя. Кроме кода действия, ОБ имеет дополнительные коды: S — установлен, C — свободен или R — переустановлен (reset) и I — инвертирован, т. е. изменил состояние на противоположное, ОБ имеет формат

< LOGIC X A >

X — дополнительный код S, C(R), I.

A — не имеет значения по умолчанию и обозначает имя или номер логического переключателя, устанавливаемого в одно из состояний, задаваемых дополнительным кодом, например:

1. LOGIC S SWCH
2. LOGIC I KL1

В первом примере проходит команда на установку переключателя SWCH.

Во втором примере инвертируется переключатель KL1.

Существуют два СЧА, относящиеся к логическому переключателю: LS и LC, — которые обозначают следующее:

LC (имя или номер) — истинно, когда логический переключатель свободен, если это так, то СЧА равен 1, в противном случае — 0;

LS (имя или номер) — истинно, если логический переключатель установлен; если это так, то СЧА равен 1, в противном случае — 0.

Используемый в этой цепочке ОБ GATE препятствует входу следующего транзакта, если определяемые условия ложны, что делается с помощью дополнительных кодов ОБ, отличающихся от условий применения ОБ GATE. Вначале рассмотрим его использование в общем виде, формат ОБ GATE в этом случае имеет вид

< GATE XXX A[,B] >

XXX — дополнительный код.

Для логического переключателя — это LS и LC.

Для устройств:

I — прервано, NI — не прервано, U — захвачено или прервано, NU — не захвачено и не прервано, FV — доступно, FNV — не доступно, FS — готово к захвату, FNS — не готово к захвату.

Для памятей:

SE — пуста, SNE — не пуста, SF — полна, SNF — не полна, SV — доступна, SNV — не доступна.

Для транзактов, имеющих одинаковые характеристики (групп, ансамблей):

M — является членом группы, NM — не является членом группы.

A — не имеет значения по умолчанию ни для одного объекта, представляет собой имя или номер логических переключателей, устройств, памятей или ОБ, проверяющих наличие транзактов, отвечающих заданным условиям (к числу таких ОБ относятся ASSEMBLE, GATHER, MATCH).

B — при умолчании 0, в режиме перехода относится ко всем формам ОБ и указывает на имя или номер дополнительного, не последовательного блока, к которому направится Хакт при ложности дополнительного кода, т. е указывает альтернативный адрес.

Приведем несколько примеров.

```
1.  GATE  LS     1
2.  GATE  FNV    7
3.  GATE  SF     16
4.  GATE  FNI    21,ALTR
```

В первом примере транзакт блокируется до установки ключа 1.

Во втором примере транзакт блокируется, пока устройство 7 занято.

В третьем примере транзакт блокируется до заполнения памяти 16.

В четвертом примере транзакт после прерывания устройства 21 переходит к ALTR.

B. ОБ, размножающие Хакт и объединяющие их в группы или ансамбли

B1. ОБ SPLIT (Расщепить)

Этот ОБ позволяет моделировать ситуации, когда несколько элементов должны встретиться в одном месте для их объединения (детали для объединения в устройство, ряд параграфов для включения в общий документ и т. д.). Для создания копий (клонов) транзактов и служит ОБ SPLIT, формат записи которого имеет вид

< SPLIT A,B,[C,D-G] >

A — не имеет значения по умолчанию и задает число копий с начального транзакта, причем, и начальный транзакт и его копии принадлежат к одному семейству. Как только транзакт определен, он становится последним в классе приоритета СТС, каждый ИН транзакта при появлении копии увеличивается на 1.

B — не имеет значения по умолчанию и определяет цель (следущий блок) для копии транзакта. Операнд назначается индивидуально для каждой копии и при желании назначить какие-то дополнительные индивидуальные параметры должен объединяться с необязательным операндом C.

C — может быть по умолчанию 0, но тогда он просто не используется. Дополнительные параметры могут назначаться только в сторону увеличения по сравнению с основным транзактом на 1, приращение для каждой следующей копии равняется 1.

D–G — редко применяемые операнды, не задаваемые по умолчанию, используются для изменения копий, отличающихся от первоначального транзакта как по типу, так и по числу параметров. Транзакты обладают всеми свойствами основного до тех пор, пока операнды D–G не изменят эти условия.

B2. ОБ ASSEMBLE (Собирать, объединять)

Этот ОБ объединяет созданные копии транзактов после прохождения ими каких-то необходимых ОБ, как только ОБ ASSEMBLE исполняется, сразу начинается операция объединения и транзакт перемещается из СТС в список ансамблей. Как только проходит установленное время моделирования, другие транзакты из ансамбля поступают в ОБ ASSEMBLE и там разрушаются. Как только установленное число копий будет уничтожено, первоначальный Хакт, находящийся в списке ансамблей, вновь переходит в СТС и продолжает свое движение по МФ. Формат этого ОБ имеет вид

< ASSEMBLE A >

A — не имеет значения по умолчанию и определяет число копий транзакта в ансамбле, которые должны быть объединены.

B3. ОБ GATHER (Собирать)

Этот ОБ действует аналогично ОБ ASSEMBLE, но ОБ GATHER не разрушает копий, копии одна за другой при исполнении этого блока поступают в список объединений. После определения числа копий они в одно и то же время выходят из списка объединений, переходят в СТС и следуют своим путем по МФ. Формат записи аналогичен предыдущему.

B4. ОБ MATCH (Соответствовать, подходить под пару)

Этот ОБ синхронизирует движение членов ансамбля, создавая параллельные траектории движения, по идее, надо ставить в каждую параллельную цепочку по одному блоку MATCH. Когда Хакт достигает ОБ MATCH, он останавливается и ждет, когда другой член ансамбля достигнет сопряженного ОБ MATCH. При этом Хакт из СТС поступает в список синхронизации. Когда собираются все члены ансамбля, Хакт возвращается в СТС и начинает дальнейшее движение по МФ. Формат этого ОБ имеет вид

< [label] MATCH A >

Метка у этого ОБ чаще всего присутствует, но не является обязательной.

A — не имеет значения по умолчанию и определяет имя и число ОБ, называемых сопряженными, которые проверяются на наличие в них транзактов при условии синхронизации, т. е. быть идентичным членом ансамбля и ожидать выполнения условий, предписываемых вышеописанными ОБ. Эти ОБ чаще всего используются сопряженными парами и практически никогда их не бывает больше. Такая схема синхронизации наиболее удобно реализуется в МФ.

В качестве примера, поясняющего механизм действия этих ОБ, рассмотрим программу, моделирующую процесс изготовления изделия из двух деталей.

Пример. Детали поступают с интервалом (300 ± 50) единиц времени. Обработкой деталей занимаются двое рабочих (один рабочий обрабатывает одну деталь). Процесс обработки каждой детали включает по две операции. Для первой детали время выполнения операций равно (70 ± 20), (20 ± 10), для второй — (60 ± 30), (30 ± 20). После выполнения первой операции рабочие производят сверку, время выполнения которой пренебрежительно мало (полагается равным нулю). По окончании обработки деталей третий рабочий производит сборку готового изделия из двух деталей со временем (50 ± 20).

```
*       МОДУЛЬ 1 Описания
*       процесс изготовления деталей
                SIMULATE
*       МОДУЛЬ 2 Исполнения
*       поступление деталей на обработку
                GENERATE        300,50
                SPLIT           l,MANB
*       обработка первым рабочим первой детали
        MANA    SEIZE           1
                ADVANCE         70, 20    операция 1
        MET1    MATCH           MET2      сверка
                ADVANCE         20,10     операция 2
                RELEASE         1
                TRANSFER        ,MANC
*       обработка вторым рабочим второй детали
        MANB    SEIZE           2
                ADVANCE         60,30     операция 1
        MET2    MATCH           MET1      сверка
                ADVANCE         30,20     операция 2
                RELEASE         2
*       сборка изделия третьим рабочим
        MANC    ASSEMBLE        2
                SEIZE           3
```

```
                ADVANCE     50,20   сборка
                RELEASE     3
*       выход готового изделия
                TERMINATE   1
*       МОДУЛЬ 3 Управления
                START       100
                END
```

Комментарий. Приведенную программу можно воспринимать как иллюстрацию применения описанных выше ОБ, а можно записать в собственный файл (находясь в дистрибутиве программы), промоделировать, изучить листинг (см. гл. 6) и воспользоваться помощью отладчика (дебаггера, см. гл. 7). В программе на русском языке выделены три модуля программы, отдельные фрагменты и дан комментарий, который, в принципе, можно расширить.

Г. ОБ, изменяющие маршрут движения транзактов

Основными операторами группы являются GATE, TRANSFER, TEST, LOOP. Транзакты, входящие в соответствующие блоки, далее продвигаются не к следующему ОБ, а к блокам, адрес которых определяется либо указывается в ОБ. Операторы GATE и TEST имеют расширенное поле операции.

ОБ GATE был рассмотрен в п. Б4. Напомним, что он разрешает движение транзактам (в основном либо альтернативном направлениях) при определенном состоянии оборудования: устройств, памятей, ключей. Существуют два режима работы блока GATE: режим отказа (условного входа) и режим перехода (безусловного входа).

Г1. ОБ TRANSFER (Перенести, переместить)

Этот ОБ позволяет осуществлять непоследовательное прохождение Хакт разными путями. Формат записи в общем виде выглядит следующим образом:

< TRANSFER A,B,C >

Существуют четыре основных варианта применения ОБ.

1. *Безусловный переход:*

< TRANSFER ,B >

A — по умолчанию 0, заменяется обязательной запятой.

B — не имеет значения по умолчанию, характеризует имя (адрес) блока, к которому направляется транзакт.

2. *Условный переход с одним альтернативным адресом (режим "BOTH"):*

< TRANSFER BOTH,B,C >

A — не имеет значения по умолчанию, операнд заменяется словом BOTH, указывающим тип режима.

B — по умолчанию обозначает, что Хакт следует в первый последовательный блок, при именовании операнда B характеризует имя (адрес) блока, к которому направляется транзакт (основной адрес).

C — не имеет значения по умолчанию, характеризует собой альтернативный адрес (при невозможности войти в блок с адресом B).

3. Условный переход с многими альтернативами (режим "ALL"):
<TRANSFERALL,B,C,D>

A — не имеет значения по умолчанию, операнд заменяется словом ALL, указывающим тип режима.

B — по умолчанию обозначает, что Хакт следует в первый последовательный блок, при именовании представляет собой первый адрес.

C — не имеет значения по умолчанию, определяет последний адрес.

D — не имеет значения по умолчанию, представляет собой константу M, используемую для вычисления возможных адресов движения транзактов: адрес в поле B, затем — B+M, B+2M, ..., адрес в поле C.

4. Статистический переход (переход с заданной вероятностью):
<TRANSFER A,B,C>

A — не имеет значения по умолчанию, характеризует вероятность перехода транзакта по адресу C или часть времени, используемую ОБ C.

B — по умолчанию является следующим последовательным ОБ, при именовании представляет собой альтернативный адрес.

В GPSS/H удобно использовать так называемую относительную адресацию, т. е. обращение к какому-либо блоку, не имеющему метки, осуществляется с помощью его относительного адреса. Для примера рассмотрим три варианта записи блока безусловного перехода.

```
1. TRANSFER     ,CPU2
2. TRANSFER     ,*+3
3. TRANSFER     ,CPU1+2
```

В первом примере ОБ (прямая адресация) направляет транзакт к блоку с именем CPU2. Два остальных — примеры относительной адресации. Второй ОБ посылает транзакт к третьему по счету блоку после рассматриваемого ОБ. Блок, к которому направляется транзакт последним ОБ, является вторым по счету от блока CPU1.

Приведем примеры для варианта BOTH.

```
1. TRANSFER     BOTH,KASS1,KASS2
2. TRANSFER     BOTH,,TWO
```

В примере 1 транзакт первоначально направляется к блоку с отметкой KASS1. При невозможности войти в этот блок транзакт направляется к блоку с меткой KASS2.

В примере 2 транзакт делает попытку войти в следующий блок, если ему это не удается, он направляется к блоку с именем TWO.

Пример для варианта ALL (фрагмент программы):

```
            TRANSFER    ALL, FIRST, LAST, 3
FIRST       GATE LS     AAA
            ADVANCE     5
            GATE LR     BBB
            TRANSFER    ,LAST+2
            ADVANCE     2
            TRANSFER    ,LAST+2
LAST        GATE LS     CCC
            ADVANCE     3,1
```

Транзакт последовательно пытается войти в следующие блоки: FIRST, FIRST+3, FIRST+6 (LAST).

Упражнение. Проанализировать работу участка программы, приведенного в примере для варианта ALL, использовать оператор TRANSFER, поскольку в случае, когда транзакт не может продвинуться ни по одному из адресов, моделирующая программа направляет транзакты по каждому из адресов при каждом изменении модельного времени, что может привести к большим затратам времени моделирования.

Вариант вероятностного перехода

```
TRANSFER  .370,THIS,THAT
```

В примере с вероятностью 0.37 транзакт перейдет к блоку THAT (37% транзактов), а с вероятностью 0.63 — к блоку THIS (63% транзактов).

Г2. ОБ TEST (Проверить)

Этот ОБ определяет направление движения транзакта в зависимости от выполнения условия, заданного алгебраическим соотношением. Оператор имеет расширенное поле операции, включающее общепринятые обозначения логических операций: L (меньше), LE (меньше или равно), E (равно), NE (не равно), G (больше), GE (больше или равно). Формат записи имеет вид

```
< TEST XX        A,B,C >
```

XX — дополнительный код логической операции (L, LE, E, NE, G, GE.).

A — не имеет значения по умолчанию, представляет собой выражение, левая часть которого сравнивается с требуемой операцией: если сравнение истинно, то Хакт продвигается в следующий последовательный ОБ.

B — не имеет значения по умолчанию, представляет собой выражение, правая часть которого сравнивается с операндом A.

C — не имеет значения по умолчанию и представляет собой имя или номер ОБ, к которому направляется Хакт, если результат сравнения ложный (альтернативный адрес). Если операнд C отсутствует, а результат сравнения ложный, Хакт запрещен, вход в ОБ TEST и сравнение проводятся каждый раз, пока Хакт находится в СТС. Такое обстоятельство приводит к избыточному использованию памяти, в этом случае (операнд C не определен) следует пользоваться ОБ GATE.

Рассмотрим примеры использования ОБ.

```
1.  TEST LE     Q1,10
2.  TEST NE     S1,PF2
3.  TEST GE     PF1,PF2,CPU
```

В первом примере транзакт задерживается, если длина очереди 1 больше 10.

Во втором примере транзакт задерживается, если текущее содержимое памяти 1 равно значению 2-го параметра транзакта форматом «слово».

В третьем примере транзакт переходит к следующему блоку, если PF1 ≥ PF2, либо, в противном случае, направляется к блоку CPU.

Г3. ОБ LOOP (Создать петлю)

Этот ОБ служит для организации циклов в процессе ИМ, формат его имеет вид

< LOOP A,B>

A — не имеет значения по умолчанию и характеризует номер параметра транзакта, используемого в качестве счетчика цикла с указанием формата: «слово» (PF), «полуслово» (PH), «байт» (PB).

B — не имеет значения по умолчанию и определяет имя или номер блока, являющегося начальным в повторяющейся группе блоков.

При каждом вхождении транзакта в блок LOOP значение счетчика уменьшается на 1 и сравнивается с 0. Если оно больше 0, его адрес определяется операндом B, который задает петлю; если становится равным нулю, транзакт переходит к следующему блоку; если становится меньше 0, появляется ошибка исполнения.

Рассмотрим пример использования ОБ (фрагмент программы).

```
        ASSIGN      5,4,,PH
WAIT    ADVANCE     15,3
        LOOP        PH5,WAIT
```

Для транзакта, вошедшего в блок ASSIGN, значение 5-го параметра форматом «полуслово» положено равным 4, внутри цикла этот параметр не меняется, следовательно, этот транзакт еще 3 раза возвратится к началу цикла — блоку WAIT.

Д. ОБ, производящие выбор объектов, удовлетворяющих заданному условию

Д1. ОБ COUNT (Сосчитать)

Для определения количества объектов (из заданного множества объектов), удовлетворяющих заданному условию, используется ОБ COUNT, он относится к блокам, имеющим дополнительные коды, и его формат имеет вид

< COUNT XXX A,B,C,D,E >

XXX — дополнительные коды, определенные выше: для устройств —U, NU, I, NI, FV, FNV; для памятей — SE, SNE, SF, SNF, SV, SNV, для переключателей — LR, LC, для операций сравнения — G, GE, L, LE, E, NE.

A — не имеет значения по умолчанию, определяет имя или номер параметра с указанием формата (PF, PH, PB), используемого для организации счетчика числа объектов.

B, C — не имеют значения по умолчанию, ими задается нижняя и верхняя границы соответственно диапазона изменения номеров объектов, для которых проверяется заданное условие.

D — не имеет значения по умолчанию, не используется для устройств, памятей и переключателей, для операции сравнения представляет собой СЧА, значение которого сравнивается со значением СЧА объекта, указанного в поле E. В тех случаях, когда сравнение истинно, вычисленное значение присваивается параметру, определенному операндом A.

E — не имеет значения по умолчанию, не используется для устройств, памятей, переключателей, определяет атрибуты анализируемых объектов, может использоваться любой одноразмерный СЧА или СЛА, кроме матричных.

Рассмотрим несколько примеров.

1. Устройства, памяти, переключатели — в этом случае поля операндов D, E должны быть пустыми.

COUNT SF 5PH,10,20

В результате исполнения данного ОБ количество заполненных памятей из множества памятей с номерами от 10 до 20 будет занесено в пятый параметр формата «байт» транзакта.

2. Операция сравнения, при этом поля D и E всегда должны быть заданы:

COUNT LE 1PB,1,5,X10,FC

На множестве устройств с номерами 1–5 подсчитывается количество устройств, у которых счетчик числа входов (FC) меньше либо

равен текущему значению ячейки 10 формата «слово». Полученное число записывается в параметр 1 формата «байт», вошедшего в блок транзакта.

Д2. ОБ SELECT (Выбрать)

Этот ОБ в логическом варианте позволяет сканировать набор членов какого-то класса объектов для определения задачи: удовлетворяет хотя бы один из проверяемого класса заданным логическим условиям. Так, Хакт может проверять набор устройств, памятей или переключателей на предмет выяснения их состояния (готовности, заполненности, включения). Идентификатор объекта, удовлетворяющий условиям, заносится в список параметров транзакта, исполняющего ОБ. В варианте сравнения этот ОБ позволяет определять численные характеристики объектов, например, процент использования устройств не менее 50, содержимое какой-то очереди равно 0 и т. д. Если условия выполняются, то Хакт движется из ОБ SELECT последовательно, если нет, то движение становится непоследовательным. Наконец, ОБ SELECT позволяет определить, какой из членов ансамбля имеет наименьшее и наибольшее значение. Формат этого ОБ имеет вид

< SELECT XXX A,B,C,D,E,F >

XXX — дополнительные коды полностью повторяют логические операции и условия сравнения ОБ COUNT. Для третьего варианта использования ОБ SELECT, а именно варианта MAX/MIN, дополнительные коды вида MAX, MIN сравниваются с содержимым операнда E.

A,B,C,D,E — назначения этих полей операндов аналогичны блоку COUNT и поэтому не повторяются.

F — не имеет значения по умолчанию и используется для записи имени или номера ОБ, используемого как альтернативный адрес, если заданному условию не удовлетворяет ни один из объектов указанного множества. Если F равен 0, то Хакт всегда направляется в следующий ОБ, если операнд F определен, но нет ни одного объекта, соответствующего критерию селектирования, то значение параметра, определяемого операндом A, остается неизменным; если же операнд равен 0 и нет ни одного объекта, удовлетворяющего условиям селектирования, то параметр, определенный операндом A, обнуляется.

Е. ОБ, устанавливающие параметры Хакт

Е1. ОБ ASSIGN (Задать)

Этот ОБ изменяет атрибуты транзактов. К числу таких атрибутов относятся значения, записываемые в полях операндов E–I ОБ

GENERATE. Операнд E относится к приоритету и будет рассмотрен ниже, а операнды F,G,H,I, относящиеся к параметрам транзакта, задаются по умолчанию в виде листов параметров и с помощью ОБ ASSIGN могут быть видоизменены. Такую же функцию может исполнять ОБ BGETLIST, являющийся блочной формой ОУ GETLIST. Формат ОБ ASSIGN имеет вид

< ASSIGN A,B,C,D >

A — не имеет значения по умолчанию, определяет имя или номер параметра, которому назначается значение. Если последний символ в операнде A + или −, то параметр уменьшается или увеличивается на значение, определяемое операндом B.

B — не имеет значения по умолчанию, представляет собой замещаемое значение, которое вычитается или прибавляется к значению операнда A. Если модифицируется параметр, представляемый числом с плавающей точкой, то операнд B тоже должен иметь такой вид.

C — не имеет значения по умолчанию, определяет имя или номер оцениваемой функции, значение операнда B умножается на значение функции после определения знака функции.

D — не имеет значения по умолчанию, определяет формат параметра PF,PH,PB,PL (по умолчанию PH). Если функция отсутствует (операнд C), то формат параметра сдвигается влево на место операнда C.

Рассмотрим несколько примеров.

```
2.  ASSIGN      3+,5,,PB
3.  ASSIGN      1-7,3
4.  ASSIGN      5,2.5,,PL
```

В первом примере производится прибавление к значению параметра 3 пяти единиц.

Во втором примере параметрам с 1 по 7 приписывается значение 3 форматом «полуслово». В третьем примере задается значение параметру форматом «плавающая точка».

E2. ОБ PRIORITY (Приоритет)

В процессе моделирования иногда приходится менять как дисциплину обслуживания, так и приоритет транзакта. Для этой цели используется ОБ PRIORITY, формат которого имеет вид

< PRIORITY A >

A — не имеет значения по умолчанию, определяет новое значение приоритета для Хакт. Транзакты удаляются при этом из СТС и возвращаются обратно с новым приоритетом, становясь последними в список данного приоритета. Уровень приоритета в последней версии GPSS/H может изменяться от − 2 147 483 632 до + 2 147 483 632.

Приведем пример использования ОБ.

```
PRIORITY        5
```

В примере у транзакта переназначается приоритет.

Ж. ОБ, собирающие статистику

Ж1. Парные ОБ QUEUE /DEPART (Встать в очередь/Покинуть очередь)

Часто в процессе моделирования возникают вопросы, ответы на которые нельзя найти в стандартной статистике выходного отчета, например, такие.

Сколько транзактов проходит через какую-то точку МФ?

Чему равно среднее число транзактов, прошедших между точками А и В МФ?

Чему равно максимальное число транзактов, прошедших между точками А и В?

Какое в среднем время Хакт находился между точками А и В?

Ответы на эти и аналогичные вопросы исследователя можно получить, используя специальные ОБ сбора статистики QUEUE/DEPART. Причем первый ОБ ставится в начале сбора статистики (точка А), второй — по окончании сбора статистики (точка В) — это позволяет получить дополнительную информацию, которая не содержится в окончательном отчете. Название ОБ QUEUE (очередь или встать в очередь) представляется не совсем удачным, так как создается впечатление об организации физически существующей очереди в каком-то текущем ОБ, члены которой ожидают входа в следующий ОБ. На самом деле физически очереди не существует, а в GPSS/H этот термин используется для объекта, собирающего информацию о всех транзактах, проходящих между точками А и В. Количество очередей в МФ зависит только от желания пользователя, но при этом необходимо выполнить два условия.

1. Каждая очередь должна иметь свое собственное имя, отличное не только от другой очереди, но и от других объектов ЯИМ (что, впрочем, не всегда обязательно).

2. Очереди вложены друг в друга, поэтому вначале описывается ОБ QUEUE AA, относящийся к более длинной очереди, затем описывается более короткая очередь QUEUE BB — DEPART BB и лишь затем описывается ОБ DEPART AA.

Очереди могут нумероваться либо иметь имя не менее трех буквенных символов.

В качестве коренного отличия очередей можно отметить то, что они не являются ресурсами, а лишь собирают информацию. Исполь-

зование очереди в МФ не является обязательным, а при правильном использовании ее наличие или отсутствие не влияет на результат моделирования (проверьте это обстоятельство на простом МФ). Формат парных ОБ имеет вид

<QUEUE A,[B]>
<DEPART A,[B]>

A — не имеет значения по умолчанию, характеризует имя или номер очереди, членом которой становится Хакт.

[B] — по умолчанию равен 1, определяет число единиц, добавляемых (или отнимаемых для ОБ DEPART) в текущее содержание очереди. Чаще всего операнд B в процессе моделирования используется лишь для получения взвешенной статистики.

Рассмотрим пример использования ОБ (фрагмент программы).

```
QUEUE      LINE
SEIZE      SERVER
DEPART     LINE
ADVANCE    2
```

В примере рассмотрено стандартное применение парных ОБ, собирающих статистику по использованию устройства SERVER, как только устройство освобождается исполнением ОБ SEIZE, Хакт сразу выходит из очереди.

Ж2. ОБ TABULATE (Создать таблицу)

Этот ОБ используется для сбора дополнительной информации, которая не выводится в стандартном отчете. Структура таблицы задается ОУ TABLE (см. п. 5.1.2). МФ может содержать несколько таблиц, которые описываются в отчете, и каждая содержит размер выборки, стандартное выборочное отклонение, относительные и накопленные частоты. Каждая таблица имеет свои СЧА, формат ОБ имеет вид

<TABULATE A,[B]>

A — не имеет значения по умолчанию, определяет имя или номер таблицы, в которую вносятся желаемые наблюдения, описание таблицы дается в модуле описания с помощью ОУ TABLE.

B — по умолчанию равен 1, является не обязательным и служит в качестве весового коэффициента при создании взвешенных таблиц.

Рассмотрим примеры использования ОБ.

1. TABULATE BLACK
2. TABULATE 2
3. TABULATE BLACK,2

В первом примере заполняется таблица с именем BLACK, во втором — таблица за номером 2, в третьем примере все данные вносятся с коэффициентом 2.

З. ОБ, выполняющие функции управления

В последних версиях GPSS/H применяется большое число ОБ, позволяющих проводить управление или переназначать параметры непосредственно в МФ, что повышает удобство работы с моделью. Таких блоков насчитывается 12 (см. прил. 1), и их отличительным признаком является прибавление буквы B перед кодом операции ОУ и непосредственное размещение в модуле исполнения.

Примечание. На этом ограничим рассмотрение ОБ, некоторые будут приведены в конкретных примерах следующих глав, а некоторые помещены только в прил. 1.

5.1.2. Операторы управления[1]

Введение в рассматриваемой версии языка новых операторов управления и описания, а также расширение возможности уже имеющихся позволили гораздо удобней взаимодействовать с моделью, что невозможно было сделать раньше. В таблицах прил. 2 приведены в алфавитном порядке списки операторов управления с указанием их основных функций. Более подробно опишем наиболее важные и часто употребляемые ОУ.

И. Основные ОУ

И1. ОУ SIMULATE (Воспроизвести)

Этот ОУ — первый обязательный оператор в программе, помещаемый первым в модуле описания, указывающий на необходимость начала симуляции МФ. Отсутствие его приводит к машинной ошибке и прекращению трансляции. Формат ОУ имеет вид

< SIMULATE [A] >

[A] — по умолчанию лимит времени не задан, при наличии операнда он задает число минут машинного времени, по истечении которых моделирование будет завершено и на печать будет выведена накопленная к этому времени статистическая информация. В студенческой версии операнд не применяется.

И2. ОУ START (Начать)

Этот ОУ помещается первым в модуле управления, указывая, что входные данные прошли ввод без ошибок, и дает команду на начало

[1] Полный список ОУ см. прил. 2.

исполнения МФ. Если при моделировании происходит переназначение данных, то каждый новый прогон предваряется очередным ОУ START, т. е их может быть на единицу больше числа прогонов с новыми данными. При исполнении ОУ START происходит три события:

• счетчик свершений устанавливается в заданное начальное значение;

• инициализируются все имеющиеся в МФ ОБ GENERATE;

• транзакты начинают движение от ОБ к ОБ.

Формат этого ОУ имеет вид

< START A,[B],[C],[D] >

A — не имеет значения по умолчанию, указывает начальное значение СС (TG1), которое уменьшается на значение операнда А ОБ TERMINATE при каждом выходе Хакт из модели.

B — не учитывается по умолчанию, редко применяемый операнд, в случае его применения пишется символ NP, запрещающий все выходные данные, кроме стандартного отчета.

C — не учитывается по умолчанию, редко применяемый операнд для задания контрольного отсчета (snap count), изменяется одновременно с операндом А; когда значение становится равным или меньше нуля, следует команда остановки.

D — не учитывается по умолчанию, редко применяемый операнд, при значении операнда 1 в отчет включаются все данные о списках модели.

Примечание. В дальнейшем будем пользоваться только операндом А.

Приведем примеры задания ОУ START.

1. START 100
2. START 1

В первом примере задано число 100, которое определяет порядок окончания процесса моделирования (см. § 5.3) при задании числа стартов.

Во втором примере число 1 определяет порядок окончания процесса моделирования при задании продолжительности процесса ИМ.

ИЗ. ОУ END (Окончить)

Этот ОУ стоит всегда самым последним в модуле управления и реализует два действия: 1) указывает на фактическое окончание записей МФ; 2) останавливает исполнение МФ и передает управление командной оболочке или MS DOS. При этом все открытые файлы закрываются, а программы возвращаются к исходному состоянию. Его формат имеет вид

< END >

Следует подчеркнуть, что у данного ОУ нет никаких операндов.

К. Операторы управления данными (в алфавитном порядке)

К1. ОУ CLEAR (Очистить)

Этот ОУ позволяет провести от двух и более последовательных прогонов модели. Исполнение ОУ происходит после окончания одной реплики и перед началом другой, таким образом осуществляется связь между репликами. ОУ CLEAR всегда располагается в модуле управления между первым ОУ START и последним ОУ END, появление ОУ CLEAR требует обязательного парного ему ОУ START, т. е. при наличии в модуле управления нескольких ОУ CLEAR обязательно должно быть столько же ОУ START, не считая первого, стоящего в начале модуля. Исполнение ОУ CLEAR приводит к следующим последствиям:

- относительное и абсолютное время обнуляются;
- все транзакты, находящиеся в модели к моменту окончания процесса ИМ, т. е. вошедшие в модель, но не дошедшие до ОБ TERMINATE, уничтожаются;
- все устройства и памяти модели возвращаются в состояние готовности (незанятости);
- все статистические данные обнуляются (число использованных ОБ, число входов в устройства, памяти и очереди, максимальное содержание памятей и очередей и т. д.).

Повторение нового прогона необходимо для сбора статистических данных о модели, однако при этом нельзя забывать того обстоятельства, что при простом повторении прогона генераторы случайных чисел (ГСЧ) стартуют с одной и той же позиции и выходная статистика от прогона к прогону не будет изменяться. Для того чтобы прогоны сделать статистически независимыми, необходимо, чтобы ГСЧ не возвращались в исходное положение. ОУ CLEAR позволяет, дополнительно к вышесказанному, сделать это за счет реализации следующих положений:

- ГСЧ остаются в том положении, в котором они находились на момент окончания предыдущего прогона, т. е. начальное положение ГСЧ следующего прогона отличается от предыдущего, что позволяет сделать результаты прогонов статистически независимыми;
- счетчик ИН транзактов также не обнуляется, и номера транзактов нового прогона являются продолжением после последнего номера транзакта предыдущего прогона (например, если предыдущий прогон кончился ИН 104, то новый начинается с ИН 105).

Формат ОУ CLEAR имеет вид

< CLEAR >

Необходимо отметить, что ОУ никогда не имеет ни ярлыков, ни операндов. Примеры применения ОУ CLEAR будут приведены непосредственно при рассмотрении примеров в гл. 6.

К2. ОУ FUNCTION (Функция)

Этот ОУ используется для определения характера функции, связывающей независимую переменную с рядом зависимых величин. Эта связь может иметь непрерывный или дискретный вид (подробнее см. гл. 8). Кроме того, функция может быть представлена в символьном виде, когда значение функции вычисляется из выражения, приводимого в списке. Формат ОУ FUNCTION имеет вид

< label FUNCTION A,B,[C] основной формат

x1,y1/x2,y2/,.../xn,yn > точки функции

label — ярлык употребляется в обязательном порядке и служит для идентификации функции (правила записи см. п. 4.4.2).

A — не имеет значения по умолчанию, определяет независимую переменную функции (аргумент), например, если используется равномерное распределение и БСВ берется с 3-го генератора, то запись будет иметь вид RN3 (random number).

Операнд может кодироваться дополнительным выражением. Единственным ограничением является то, что аргумент не может прямо или косвенно ссылаться на функцию, для которой он является независимой переменной. Значения функции могут задаваться числом с плавающей точкой.

B — не имеет значения по умолчанию, состоит из двух символов (без пробела), первый из которых представляет собой обозначение типа функции, вида: C — непрерывная числовая, D — дискретная числовая, L — список числовых значений, E — дискретная символьная, M — список дискретных символов, S — селектор объектов. Второй символ представляет собой целое число связанных пар значений функции, которые представляют собой точки функции и приводятся во второй строке описания формата. Если используется символ S, то он уточняется операндом C, используемым только в этом случае.

C — по умолчанию равен 0, применяется для перечисления типов объектов; если используются объекты разных типов, то приводятся их символы, разделяемые запятой без пробелов. (Операнд C в тексте пособия использоваться не будет, поэтому символы объектов приводятся лишь в прил. 2.)

Рассмотрим пример записи функции.

SERVTIME FUNCTION RN7,D5

.2,4/.55,7.5/.7,10.5/.8,13.5/1,16.5

В примере описывается ОУ FUNCTION, имеющий ярлык SERVTIME, PP время обслуживания берется с ГСЧ7, распределение времени дискретно, функция имеет 5 пар точек: x — представляет собой накопленную частоту, а y — время обслуживания. Необходимо отметить, что начало и конец записи точек функции не выделяется никакими символами, пары точек отделяются слэшем, значения внутри пары разделяются запятой, 0 не пишется, а запись начинается с точки.

К3. ОУ INITIAL (Начальное значение)

Этот ОУ используется для установления начальных значений сохраняемым величинам (Savevalue), матричным величинам и логическим переключателям. Формат ОУ INITIAL имеет вид

< INITIAL символ инициализации[/ символ инициализации]...>

В формате под символом инициализации понимается обозначение величины, представляемое как символ величины, отделенный от имени величины знаком $, например: X$SAM,25 — полнословная сохраняемая величина SAM, имеющая значение 25. Если в качестве начального значения используется выражение, то оно представляется в скобках. При нескольких символах инициализации они отделяются друг от друга слэшем, который не ставится в начале и конце наименований. Например:

1. INITIAL XH1-XH10,13/XB$SAM-XB$JOE,(5+XH$INIT)
2. INITIAL LS10/LR1-LR5

Из первого примера следует, что в первом символе полусловные сохраняемые величины с 1-й по 10-ю имеют значение 13, а во втором сохраняемые величины SAM –JOE имеют значение, указанное в скобках, т. е. к полусловному значению сохраняемой величины INIT добавляется 5.

Во втором примере, относящемся к логическим переключателям, устанавливается, что логический переключатель 10 находится во включенном состоянии, а переключатели с 1-го по 5-й выключены.

К4. ОУ LET (Назначить)

Этот ОУ используется в последовательности ОУ или внутри отчета для назначения величины АМП. Формат ОУ LET имеет вид

< [label] LET A >

label — используется крайне редко и применяется только в тех случаях, когда в МФ используется ОУ GOTO.

A — не имеет смысла по умолчанию, используется для задания значений АМП. Например:

LET &I=1

В примере ОУ LET задает значение целочисленной АМП равным 1.

К5. ОУ PUTPIC (Выдать отображение)

При проведении нескольких прогонов сложных систем в процессе ИМ формируется выходной отчет большого объема. При этом весьма затруднительно найти информацию по отдельному прогону. Поэтому в GPSS/H используется ОУ PUTPIC (PUT out a PICture), выдающий на печать информацию по назначенному номеру отдельного прогона (действие этого ОУ подобно WRITE в Fortran, PRINT в Basic и т. д.). Отметим, что ОУ PUTPIC не используется один, а всегда сопровождается одним или несколькими указаниями по отображению (Picture Statement), формирующими представление о выходном отчете. Формат этого ОУ имеет вид

< [label] PUTPIC [A],[B],[C]

^... текст и /или поля редактирования для отображения строки 1...

^ ...текст и/или поля редактирования для отображения строки 2 и т. д. >

label — не обязательный ярлык.

A — не обязательный операнд, по умолчанию равен 1, включает в себя запись LINES = *x*, определяющую число строк *x*, включаемых в отчет.

B — не обязательный операнд, по умолчанию отображается непосредственно на дисплее, указывает имя объекта в виде записи FILE=имя _ файла.

C — не обязательный операнд, не имеющий значения по умолчанию, включает в себя заключенный в круглые скобки перечень переменных, значения которых помещаются в поля редактирования выходного отчета вида [,(output_list)]. Если таких переменных несколько, то они разделяются запятыми.

После ОУ PUTPIC следует несколько указаний по отображению, каждое из которых соответствует одной строке выходного отчета, производимого при исполнении ОУ PUTPIC. В первой колонке каждого указания по отображению ставится символ (обозначенный в формате ^), управляющий организацией разбивки выходных строк:

отсутствие индекса соответствует одинарной разбивке;

0 соответствует двойной разбивке;

(–) соответствует тройной разбивке.

В остальных колонках печатается текст и/ или сохраняются поля редактирования. Текст состоит из последовательности символов, которые без изменений копируются в отчет при исполнении ОУ PUTPIC. Поля редактирования представлены одной или большим

количеством звездочек *, включенных в указание по отображению в начале, внутри или после текста, в которых могут редактироваться значения переменных из выходного списка.

Пример использования ОУ PUTPIC с разными вариантами использования одного указания по отображению:

```
        PUTPIC LINES=1,FILE=SYSPRINT,(&I)
0The above output is for replication number *.
                                a)
The above output is for replication number 5.
                                b)
```

В примере основная запись содержит название кода операции, не предваряемой отсутствующим в данном примере ярлыком. Запись числа строк в операнде A, равная 1, могла бы быть опущена, поскольку по умолчанию число строк равняется единице, но именованная запись не является ошибочной, операнд B задает имя файла SYSPRINT, операнд C задает переменную &I, значение которой копируется в выходном отчете при исполнении ОУ PUTPIC.

Указание к отображению приведено во второй строке над индексом a), в указании первым символом является 0, что обозначает двойную разбивку (см. примеры гл. 6), текст строки указания переносится без изменений в отчет, в колонку со знаком * копируется значение переменной &I, знаком завершения записи является конечная точка. После исполнения указания a) происходит исполнение указания b) при значении &I, равном 5. Размещение указаний к отображению может производиться между ОУ START и CLEAR, тогда отмеченная реплика размещается в конце отчета, в случае расположения указания между ОУ DO и START — в начале отчета.

Рекомендация. Эти достаточно тонкие манипуляции необходимо проделать непосредственно на МФ, чтобы получить полное представление о гибкости использования ОУ PUTPIC, не забывая при этом, что в тексте указания по отчету необходимо использовать либо above, либо following.

К6. ОУ RESET (Установить)

Этот ОУ похож на ОУ CLEAR — также используется для целей сбора статистических данных, но имеет ряд отличительных черт: так, при исполнении ОУ RESET не происходит разрушения транзактов, абсолютное время не обнуляется (обнуляется только относительное время). Использование ОУ RESET эффективно при определении момента времени, когда процесс ИМ становится установившимся. В пособии этот ОУ не применяется.

К7. ОУ RMULT (Переустановить)

Этот ОУ используется для управления текущей позицией ГСЧ. Вводя данные с помощью ОУ RMULT, можно переназначать положе-

ние любого ГСЧ, что позволяет достигать статистической независимости ИМ и избегать корреляции между отдельными потоками БСВ. Формат ОУ RMULT имеет вид

< [label] RMULT A,B,C,D,.... >

label — отсутствует по определению, наличие ярлыка приводит к ошибке.

A — по умолчанию текущее значение ГСЧ1, появление операнда A устанавливает новую текущую позицию для ГСЧ1 (RN1).

B — по умолчанию текущее значение ГСЧ2, появление операнда B устанавливает новую текущую позицию для ГСЧ2 и т. д.

Приведем несколько примеров использования ОУ RMULT.

1. RMULT 50000
2. RMULT ,125000
3. RMULT 25000,,100000

В первом примере устанавливается значение ГСЧ1, равное 50000, остальные остаются неизменными.

Во втором примере начальные значения всех ГСЧ остаются неизменными, а новое текущее значение ГСЧ2 равняется 125000.

В третьем примере все ГСЧ сохраняют неизменное положение, кроме первого и третьего.

Положение, занимаемое ОУ RMULT в МФ, зависит от задач, стоящих перед исследователем. Если необходимо с целью исключения корреляции разнести потоки заявок от потоков обслуживания, то ОУ RMULT помещается в первом модуле описания, если нужно провести переназначение между прогонами, то ОУ RMULT помещается в модуле управления после ОУ START.

К8. ОУ STORAGE (Запомнить)

Этот ОУ используется для определения емкости одной или нескольких памятей. Исполнение ОУ STORAGE (впрочем, как и всех рассмотренных ОУ) происходит только после успешного завершения процесса компиляции, при этом вводится в действие оговоренное число ресурсов ОУ STORAGE. ОУ STORAGE в силу своей специфики должен располагаться в модуле описания, до того как началось движение транзактов, так как после компиляции программа последовательно исполняет все ОУ, стоящие до первого ОУ START. Формат ОУ STORAGE может быть представлен двумя способами.

Первый формат, который и будет использоваться в пособии, имеет вид

< label STORAGE A >

label — не имеет значения по умолчанию, отсутствие ярлыка приводит к ошибке компиляции. Ярлык является идентификатором

памяти, емкость которой определяется. При этом формате можно определить только одну память.

A — не имеет значения по умолчанию, отсутствие приводит к ошибке компиляции. Операнд определяет емкость памяти, т. е. число моделируемых обслуживающих устройств. Необходимо иметь в виду, что в этом формате можно использовать одну единицу емкости в заданный момент времени.

Приведем примеры записи.

```
1. SAM      STORAGE     10
2. 3        STORAGE     5
```

В первом примере памяти SAM определяется емкость 10.

Во втором примере памяти за номером 3 определяется емкость 5.

Используя многоканальное обслуживание, можно не определять ОУ STORAGE, в этом случае емкость по умолчанию практически равна бесконечности (а именно, 2 147 483 647).

При записи во втором формате

< STORAGE Sname1,A/Sname2, B/Sname3,C ... >

можно определять несколько памятей сразу. Определитель памяти представляет пары обозначений, разделенных запятой, между собой пары отделяются слэшем. Вначале пишется символ памяти, а затем ее имя — первый член пары, после запятой пишется операнд A,B,..., характеризующий емкость памяти — второй член пары. Символ S может быть отделен от имени памяти символом $ или имя может быть включено в круглые скобки, памяти одинаковой емкости могут записываться через тире.

Приведем комбинированный пример записи ОУ STORAGE:

```
STORAGE S1,10/S$CORE,5/S(SAM)-S(JOE),15
```

Комментарий. Все виды записей воспринимаются программой одинаково и могут комбинироваться.

В примере памяти S1 приписывается емкость 10, памяти S с именем CORE приписывается емкость 5, а ряд памятей с именами S(SAM)–S(JOE) имеют одинаковую емкость 15.

В случае, если память имеет емкость, равную единице, то она полностью аналогична устройству, осуществляющему одноканальное обслуживание. Отсюда следует, что в GPSS/H одноканальное обслуживание можно моделировать двумя способами. Поэтому имеет смысл кратко рассмотреть разницу между этими двумя возможностями. Во-первых, обслуживание в устройствах может быть прервано (PREEMPT) транзактом с более высоким приоритетом, что не допускается при обслуживании с памятями. Во-вторых, у устройств поддерживается понятие готовности — не готовности (рабочее состояние — состояние ожидания), для памятей их состояния имеют дру-

гой смысл, что будет рассмотрено ниже. В- третьих, при исполнении ОБ SEIZE программа записывает ИН Хакт и при исполнении ОБ RELEASE проверяет, тот ли транзакт вышел из устройства обслуживания. При исполнении ОБ ENTER ИН Хакт не записывается.

Предупреждение! В GPSS/H четко разделены функции устройств и памятей и необходимо соблюдать внутренние правила программы.

Выше были высказаны замечания в адрес не очень четкого определения STORAGE. Необходимо помнить, что этот термин прежде всего относится к устройству, моделирующему группу идентичных серверов, а не определяющему физическое расположение или объем. *Следует обратить внимание на обязательность идентичности: как только серверы не идентичны, их следует моделировать устройствами, а не памятями!* Для определения числа идентичных серверов используется термин *емкость* (который, впрочем, также не удачен, как и память, так как сразу возникает ассоциация с конкретным местом, куда что-то можно положить и, соответственно, забрать). Для определения состояния памятей в GPSS/H используется несколько терминов:

- память считается заполненной (Full), если каждая единица ресурса (сервер) занята в момент обращения к ней;
- память считается пустой (Empty), если каждая единица ресурса свободна;
- текущее содержание (Current Contents) памяти определяется числом единиц ресурса, занятых на текущий момент времени;
- остающаяся емкость (Remaining Capacity) памяти — число единиц ресурса, свободных на текущий момент. Так, например, при заполненности памяти остающийся объем равен 0.

Следует иметь в виду, что все перечисленные термины имеют отношение только к GPSS/H.

К9. ОУ TABLE (Составить таблицу)

Этот ОУ позволяет получить дополнительную, по сравнению со стандартным отчетом, статистическую информацию. Например, исследователя интересует текущее содержание памяти в различных репликах. Полученное значение математического ожидания может использоваться для выбора числа серверов в проектируемой системе. ОУ TABLE позволяет получить математическое ожидание (среднее значение), стандартное отклонение и относительные частоты попадания в заданный интервал. Формат ОУ TABLE имеет вид

< label TABLE A,B,C,D,[E] >

label — не имеет значения по умолчанию, отсутствие ярлыка приводит к ошибке компиляции, определяет имя или номер таблицы.

A — не имеет значения по умолчанию, определяет вид операций с таблицей. Существует 4 вида возможных таблиц, задаваемых операндом A.

1. Операнд A определяется выражением, на основе которого определяется статистика, помещаемая в таблицу. Эта статистика для совместимости с другими версиями должна представлять собой только простой СЧА и не должна быть представлена в матричных форматах типа MX, MH, MB, ML.

2. Операнд A определяется так же, как в случае 1, но впереди стоит знак "–". В этом случае таблица называется дифференциальной, т. е. берется разница между текущим значением и предыдущим и результат помещается в таблицу, текущее значение сохраняется для последующих вычислений.

3. Операнд A имеет впереди символы IA (interarrival), что обозначает определение промежутков между последовательными входами. При этом определяется разность между текущим значением абсолютного времени и значением абсолютного времени в предыдущий момент, результат помещается в таблицу, текущее значение времени сохраняется для последующих вычислений.

4. Операнд A имеет впереди символы RT (arrival rate), что обозначает определение доли приходов (относительной частоты) в какой-то интервал. При этом варианте операндом E оговаривается интервал времени, являющийся шагом таблицы. Этот вид используется для определения такой статистики как «приход в течении часа». Счетчик приходов ОБ TABULATE при каждом исполнении ОУ TABLE увеличивается на единицу. После того, как определена таблица, управление осуществляется с помощью «контролера транзактов», располагающегося в СБС, где к текущему значению абсолютного времени прибавляется значение времени, стоящее в операнде E. При этом каждый раз производятся следующие действия: накопленное время записывается в таблицу, счетчик приходов обнуляется, контролируемый транзакт включается в СБС со временем, равным текущему значению абсолютного времени плюс значение операнда E.

B — не имеет значения по умолчанию, определяет собой верхнее значение первого (нижнего) интервала частот, в GPSS/H допускается использование числа с плавающей точкой. Первый интервал может простираться до $-\infty$.

C — не имеет значения по умолчанию, представляет собой ширину интервала и постоянен для всех интервалов, кроме первого и последнего. Естественно, что ширина интервала должна быть больше нуля и может представляться числом с плавающей точкой (для совместимости с другими версиями следует использовать целочисленные значения).

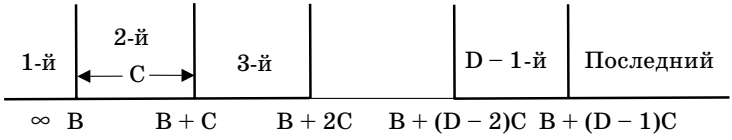


Рис. 5.2. *Схема построения таблицы*

D — не имеет значения по умолчанию, определяет число частотных интервалов таблицы. Если первый символ операнда D буквенный — а именно W, то такая таблица является взвешенной. При этом ОБ TABULATE, используя свой операнд B (см. Ж2) создает взвешенные значения средних величин, стандартных отклонений, количества наблюдений (взвешенные таблицы в пособии не рассматриваются).

На рис. 5.2 приведена схема построения таблицы.

При построении таблицы используются следующие правила:

- если записываемое в таблицу значение меньше или равно верхней границе первого интервала, оно записывается в этот первый интервал;
- если записываемое в таблицу значение больше верхней границы предпоследнего интервала, оно записывается в последний интервал;
- все другие значения включаются в свои частотные интервалы по принципу: меньше или равно значению верхнего предела промежуточного интервала;
- когда происходит запись наблюдаемого значения, показание счетчика частотного интервала увеличивается на единицу.

Рассмотрим примеры записи ОУ TABLE.

```
1. SAM     TABLE M1,100,100,10
2. SAM     TABLE M1,100,100,W10
3. 10      TABLE IA,50,50,5
4. ARRIV   TABLE RT50,50,10,1000
```

Первый и второй примеры для таблицы SAM отличаются только тем, что вторая таблица взвешенная. В них верхняя граница первого интервала равна 100, ширина промежуточных интервалов равна 100, число интервалов 10.

В третьем примере для таблицы с номером 10 определяются промежутки между приходами транзактов, верхняя граница первого интервала равна 50, ширина промежуточных интервалов равна 50, число интервалов равно 5.

В четвертом примере для таблицы с именем ARRIV определяется доля приходов в долях 1000 (операнд E), верхняя граница первого интервала равна 50, ширина промежуточных интервалов равна 50, число интервалов равно 5.

В GPSS/H используется также специальная форма ОУ QTABLE для записи данных очередей, формат записи и логика заполнения намного проще ОУ TABLE, так как операнд A представляет собой только имя или номер очереди, а все остальные нотации совпадают, операнд E отсутствует.

Приведем пример записи ОУ QTABLE.

TSAMQ QTABLE SAMQ,100,100,10

В примере таблица с именем TSAMQ составляется для очереди с именем SAMQ, параметры таблицы: верхняя граница первого интервала равна 100, ширина промежуточных интервалов равна 100, число интервалов 10.

Л. ОУ управления логикой

Л1. ОУ DO/ENDDO (Выполнить/ Завершить выполнение)

Эти два парных оператора служат для организации петли управления процессом ИМ. Когда необходимо провести несколько независимых прогонов МФ, можно воспользоваться двумя способами получения этих прогонов, не выходя из процесса ИМ. Первый осуществляется с помощью ОУ CLEAR (см. К.1) и парного с ним ОУ START, при этом число этих парных ОУ равняется числу желательных прогонов. Естественно, что при большом числе прогонов запись МФ перегружается этими указаниями. Второй способ — организация компактной записи с помощью парных ОУ DO/ENDDO. Формат этих ОУ имеет вид

< [label] DO A,B,[C] >

< [label] ENDDO >

label — не обязательный ярлык, который чаще всего отсутствует, характеризует собой имя организуемой петли.

A — не имеет значения по умолчанию, символизирует индекс начала петли, представляющий собой целочисленную скалярную АМП и записываемый в виде &I=1.

B — не имеет значения по умолчанию, характеризует собой предельный номер конечного члена петли.

C — не обязательный операнд, по умолчанию равен единице, задает приращение индекса.

Когда индекс достигает предельного значения, исполняется ОУ ENDDO и петля управления прекращает свое существование. Если значения заданы числом с плавающей точкой, то число усекается до целого значения, отрицательные приращения программой не поддерживаются. Количество петель управления может достигать 19, причем они вкладываются друг в друга, создавая своеобразную иерар-

хию, второстепенные петли в МФ изображаются с отступом вправо от основной петли, следующая вложенная петля также отступает вправо. Между ОУ DO/ENDDO размещаются другие ОУ, например CLEAR/START. Отметим важные особенности петли ОУ DO/ENDDO:

• DO/ENDDO создают структуру петли управления, а не петли транзактов, поэтому в петле присутствуют только ОУ, а не ОБ;

• операнды в петле должны записываться по правилам записи операндов, предусмотренных программой, т. е. без пробелов, появление пробела справа воспринимается программой как начало комментариев.

Рассмотрим пример записи петли управления.

```
    DO          &I=1,10,1
        START 1
        CLEAR
    ENDDO
END
```

В примере петля управления задана операндами ОУ DO, операнд С можно было бы не писать, так как его значение по умолчанию и так равно 1, петля начинается с первого шага и заканчивается на 10-м шаге. При каждом значении индекса происходит исполнение ОУ START/CLEAR. При отсутствии петли управления эту пару ОУ надо было бы написать 10 раз подряд. Отметим также, что при введении петли управления на первое место в модуле управления встает ОУ DO.

ОУ DO/ENDDO являются важными компонентами мощного и гибкого языка операторов управления (control statement language — CSL), являющегося частью GPSS/H.

Л2. Другие ОУ управления логикой

Другими компонентами CLS являются ОУ, обеспечивающие простоту вычислений: LET, ОУ ввода и вывода информации – GETLIST и PUTPIC, логические операторы — IF, ELSEIF, ELSE, ENDIF; ОУ, способный обращаться и вызывать внешние подпрограммы — CALL; ОУ безусловного обращения — GOTO. Все эти операторы представляют несомненный интерес, но их рассмотрение выходит за рамки пособия.

5.1.3. Операторы описания[1]

Рассмотрим только основные ОО, которые будут широко использоваться в материале последующих глав пособия (приводимые ОО даются в алфавитном порядке).

[1] Полный список ОО см. в прил. 2.

М. ОО, используемые в пособии

М1. ОО INTEGER (Целочисленный)

Этот ОО используется для описания целочисленных АМП, представляющих собою скалярную одноразмерную переменную. Все целочисленные АМП должны быть определены в ОО INTEGER до их первого появления в МФ. Формат записи ОО INTEGER имеет вид

< INTEGER A,B,C,... >

label — не имеет значения по умолчанию, появление ярлыка воспринимается как ошибка компиляции.

A — не имеет значения по умолчанию, отсутствие операнда воспринимается как ошибка компиляции, определяет имя целочисленной АМП.

B,C — имеют тот же смысл, что и операнд A, и при необходимости определяют дополнительные целочисленные АМП.

Таким образом, ОУ INTEGER может определять несколько АМП одновременно, при этом АМП разделяются запятыми. Если определяется одноразмерный массив, то число членов массива заключается в круглые скобки. Рассмотрим примеры использования ОУ INTEGER.

```
1. INTEGER      &I,&J,&K
2. INTEGER      &IVEC(25)
```

В первом примере задаются 3 различные АМП, во втором примере определяется массив, включающий 25 членов.

Существует еще четыре ОО, описывающих другие типы АМП (см. п. 4.4.2 и прил. 2).

М2. ОО OPERCOL
(Переназначить начало записи операндов)

Этот ОО используется для переназначения стартовой позиции записи операндов, которая по умолчанию начинается с 25-й колонки (подробнее см. п. 4.4.2) и может меняться от 10-й до 60-й колонки. Формат записи ОО OPERCOL (ORERand start COLumn) имеет вид

< OPERCOL A >

A — по умолчанию равен 25, при наличии операнда меняет позицию начала записи операндов в соответствии с числом, стоящим в поле операнда (10 – 60).

Приведем пример записи ОО OPERCOL.

```
OPERCOL 35
```

В примере стартовая позиция начала записи операндов изменена на 35.

Применение ОО OPERCOL особенно полезно, когда создаются одинарные или вложенные петли управления, ОО OPERCOL располагается в модуле описания.

М3. ОО REALLOCATE (Перераспределять)

Этот ОО используется для указания об увеличении общей памяти, которая в студенческой версии имеет объем 32 720 байт, а обычно используется 10 000 байт. Располагается ОО REALLOCATE в любом месте МФ, но разумней располагать его в модуле описания, чтобы не дожидаться появления сообщения об ошибке № 411 — «Out of COMMON» — переполнение общей памяти. Формат записи ОО REALLOCATE имеет вид

< REALLOCATE A,B >

A — не имеет значения по умолчанию, представляет собой имя переназначаемого объекта (в нашем случае — памяти).

B — по умолчанию равен 10 000 байт, представляет собой число, обозначающее запрашиваемый объем памяти (до 32 720 байт).

В программе существует другая возможность увеличения объема общей памяти: непосредственно в командной строке можно записать опцию MAXCOM после имени файла, что автоматически увеличивает объем общей памяти до максимума. Действие этой опции перекрывает эффект действия ОО REALLOCATE, но увеличивает время ИМ.

Рассмотрим пример применения ОО REALLOCATE.

REALLOCATE COM,20000

В примере общая память переназначается до значения 20 000 байт.

ОО REALLOCATE может также использоваться для переназначения числа членов какого-либо класса объектов (устройств, памятей), однако эта возможность в пособии не используется.

М4. ОО UNLIST / LIST
(Исключить из списка / Включить в список)

Эти два ОО, по сути дела, являются парными и служат для исключения или включения в итоговый отчет сообщений о кодах исходных файлов — ABS, сведений о строках записи макросов — MACX, списка ОУ, выводимых на печать — CSECHO (Control Statement ECHO). Рассмотрим более подробно наиболее интересный для нас вариант повторения (echo) образов ОУ в итоговом отчете, которые после исполнения ОУ по умолчанию выдаются на печать сразу после завершения фазы движения транзактов (см. § 5.4). С одной стороны, повторение (эхо) ОУ помогает исследователю при чтении отчета, а, с другой стороны, эти повторения сильно перегружают отчет. Ис-

пользование ОО UNLIST позволяет подавить повтор исполняемых ОУ. Для того чтобы эффект ОО UNLIST был наилучшим, его помещают в модуле описания сразу после ОУ SIMULATE. Формат записи ОУ UNLIST имеет вид

< UNLIST A >

A — не имеет значения по умолчанию, представляет собой символы, представленные выше.

Сочетание этих двух ОО позволяет включать и исключать из итогового отчета различные виды информации (в пособии рассматривается только вариант подавления повторения в отчете исполняемых ОУ).

Рассмотрим пример записи ОО UNLIST.

UNLIST CSECHO

В примере дается указание о запрете вывода на печать названий исполняемых ОУ.

§ 5.2. ОСНОВНЫЕ АТРИБУТЫ GPSS/H

Практически все объекты GPSS/H обладают специфическими характеристиками, которые содержат информацию об объектах, позволяют осуществлять вычисления и производить логические действия.

Большинство атрибутов устанавливается программой автоматически, но некоторые могут задаваться исследователем (например, разные типы автомобилей, проходящих через пост контроля). Большинство атрибутов требуют не только указания класса объектов, но и его имени или номера. Когда атрибут относится строго к одному классу объектов, то имя класса может не ставиться, а идентифицируется только член этого класса. При попытке записи в параметр очень большого числа появляется предупреждение о необходимости принять меры, чтобы не наступила критическая ошибка. Всего в GPSS/H используется 158 атрибутов, включая встроенные распределения БСВ (типа RVEXPO, RVNORM и т. д. — всего 26), встроенные математические функции (типа COS, EXP, LOG — всего 10). Оба списка приведены в прил. 3.

Естественно, что нет смысла рассматривать 122 остальных атрибута подробно. Ниже будут рассмотрены только основные СЧА, относящиеся к классам объектов. СЛА позволяют определять логические свойства функционирования устройств памятей и логических переключателей путем задания условий, приведенных в § 5.1, для

конкретных объектов, например запись FNU (SERVER) истинна в том случае, если устройство не занято, запись SNE (CLARK) истинна в том случае, если память не пуста, и т. д.

5.2.1. Стандартные числовые атрибуты

А. Системные атрибуты

AC1 — текущее абсолютное время
AN1 — число транзактов в ансамбле
C1 — текущее относительное время
NAC1 — следующее значение абсолютного времени
TG1 — текущее значение
XID1 — ИН Хакт

Б. Атрибуты транзактов

Транзакты имеют СЧА встроенные и назначаемый пользователем. К числу *встроенных* относятся:

AN1 — число транзактов в ансамбле

M1 — транзитное время Хакт, которое определяется как разность между текущим временем AC1 и отмеченным временем транзакта (Transaction Mark Time). Время M может быть представлено во всех возможных форматах (H,F,B,L), соответственно будет выглядеть и нотация времени

PR — приоритет Хакт

XID1 — ИН Хакт

Атрибут, *назначаемый* пользователем:

P — величина параметра (во всех форматах), назначаемая с помощью ОБ BLET

В. Атрибуты блоков

N — полное число транзактов, вошедших в ОБ (с указанием имени или номера ОБ)

W — текущее содержание (число транзактов в блоке)

Г. Атрибуты устройств

F — истинно (1), если устройство занято

FC — число транзактов в устройстве

FT — среднее время пребывания транзакта в устройстве

FR — использование устройства в долях тысячи (например, 453 = = 0.453)

Примечание. Не приведены СЧА, определяющие логические операции, вида FI, FNI, FNS, FNU, FNV, FPU, FPV, FRU, FRV, FS, FU, FV.

Д. Атрибуты памятей

R — оставшаяся емкость
S — число используемых единиц памяти
SA — среднее содержание
SC — полное число Хакт, вошедших в память
SM — максимальное содержание
SR — использование в долях тысячи
ST — среднее время нахождения Хакт в памяти

Примечание. Не приведены СЧА, определяющие логические состояния, вида SNE, SNF, SNV, SPU, SPV, SRU, SRV, SUA, SUN, SUT, SV

Е. Атрибуты очередей

Q — текущее содержание
QA — среднее содержание
QC — полное число Хакт, находившихся в очереди
QM — максимальное содержание
QT — среднее время пребывания в очереди для всех Хакт
QX — среднее время пребывания в очереди, исключая нулевые входы
QZ — число нулевых входов

Ж. Атрибуты таблиц

TB — среднее выборочное значение для таблиц и очередей
TBW — взвешенное значение среднего
TC — число наблюдений
TCW — взвешенное число наблюдений
TD — выборочное стандартное отклонение
TDW — взвешенное стандартное отклонение

З. Атрибуты списка пользователя

CA — среднее содержание
CC — число транзактов в списке
CH — текущее содержание
CM — максимальное содержание
CT — среднее время пребывания транзакта в списке

И. Атрибуты функций и переменных

FN — значение функции
BV — значение булевой переменной
V — значение переменной

К. Атрибуты БСВ

RN — БСВ, взятые из заданного потока случайных чисел ГСЧ
RNX — положение следующей выборки из потока случайных чисел

5.2.2. Стандартные логические атрибуты

Логические переключатели

LC, LR — синонимы для обозначения состояния выключено
LS — обозначение состояния включено

5.2.3. Стандартные символьные атрибуты

CHAR — целочисленный символ в ASCII-кодах
CHARSTOF — символьный поток с плавающей точкой
CHARSTOI — символьный целочисленный поток
CURDATE — текущая дата
CURTIME — текущее время
ENTNUM — сравнение символьного потока с названием класса объекта
LEN — длина символьного потока
LISDATE — текущая дата, приводимая в начале отчета
LISTIME — текущее время, показываемое в начале отчета
SSG — обозначение подпотока символьного потока
SYM — имя, ассоциируемое с именем класса объектов

§ 5.3. ПРАВИЛА ОКОНЧАНИЯ ПРОЦЕССА ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ

Как уже отмечалось, одним из главных моментов ИМ является выработка команды на прекращение процесса ИМ при выполнении заданного условия. Этот вопрос достаточно очевиден, но, тем не менее, вызывает определенные трудности у студентов, поэтому он выделен в отдельный параграф.

Процесс ИМ в GPSS/H прекращается при выполнении одного из двух условий:

1) окончание ИМ по числу стартов, когда показания СС (TG1) обнуляются или приобретают отрицательное значение;

2) окончание ИМ по времени испытаний, когда накопленное время движения транзактов превышает заданное время испытаний.

Рассмотрим оба условия.

5.3.1. Правило окончания по числу стартов

В данном случае основанием для окончания ИМ является число, задаваемое операндом А ОУ START. Сразу после успешной компиляции и начала моделирования это число вносится в счетчик свершений. Проход первого транзакта через ОБ TERMINATE (уничтожение транзакта) приводит к вычитанию значения операнда А ОБ TERMINATE из показаний СС, каждый следующий проход транзакта через свой ОБ TERMINATE (напомним, что их может быть много, но СС в МФ один!) уменьшает показания СС на число, определяемое операндом А. Значение операнда А по умолчанию равно 0, т. е. проход Хакт через такой ОБ TERMINATE не изменяет показаний СС. В других случаях значение операнда А может быть любым целым положительным числом. На основании сказанного можно предложить выражение, позволяющее понять логику работы СС:

$$\text{A(START)} - \sum_{i=1}^{x} \text{A(TERMINATE)} \le 0,$$

которое читается следующим образом: для выполнения условия прекращения ИМ по числу стартов необходимо из числа, первоначально заданного в СС операндом А ОБ START, вычитать накапливаемую сумму терминирований, задаваемых операндами разных ОБ TERMINATE (если их несколько в МФ), до выполнения условия, что разность станет равной нулю или примет отрицательное значение.

Рассмотрим несколько простых примеров, решаемых устно, но при желании их можно промоделировать в режиме контроля с помощью отладчика (описание см. гл. 7).

1.
```
SIMULATE
GENERATE    50
TERMINATE   1
START       3
END
```

2.
```
SIMULATE
GENERATE    50
TERMINATE   2
START       3
END
```

3.
```
SIMULATE
GENERATE    50
TERMINATE   5
START       3
END
```

4.
```
SIMULATE
GENERATE    50
TERMINATE   1
GENERATE    100
TERMINATE    0
START 3
END
```

5.
```
SIMULATE
GENERATE    50
TERMINATE   1
GENERATE    75
TERMINATE   1
START       3
END
```

В каждом из этих примеров необходимо ответить на вопрос, в какой момент времени окончится процесс ИМ? Рассмотрим эти примеры последовательно. Для начала представим себе временную ось, начинающуюся всегда в момент времени 0.0. Временная дискрета выбирается исследователем, предположим, что цифра 50 в ОБ GENERATE соответствует 50 минутам (секундам, часам и т. д.). На рис. 5.3 представлена эта временная ось. Используем рисунок для решения первого примера.

Поскольку задано детерминированное значение операнда А ОБ GENERATE, то транзакты будут поступать каждые 50 временных дискрет, а именно: 50, 100, 150 и т. д. Первый транзакт придет в момент 50, сразу пройдет через ОБ TERMINATE, операнд А которого равен единице. В СС записано число 3, из которого вычитается

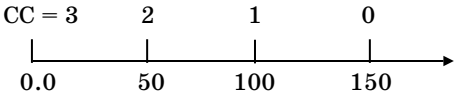


Рис. 5.3. *Решение примера 1*

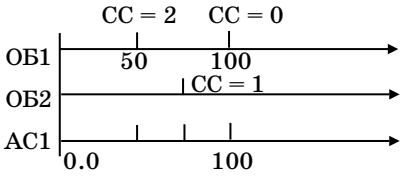


Рис. 5.4. *Решение задачи 5*

1: 3 – 1 = 2. Приход следующего транзакта через 50 единиц, но в 100 единиц абсолютного времени, уменьшит показания еще на единицу и т. д. Окончательное решение, что ИМ закончится в 150 временных дискрет, очевидно из рис. 5.3.

Во втором примере ИМ окончатся в 100 временных дискрет, так как вычитается число 2.

В третьем примере ИМ окончится в 50 временных дискрет, так как из числа 3 (число стартов) вычитается число 5 (значение операнда А ОБ TERMINATE) и сразу получается отрицательный результат.

В четвертом примере появление второго ОБ GENERATE ни на что не влияет, так как ОБ TERMINATE, сопряженный с ним, имеет операнд А, равный 0 (это значение можно было бы и не писать, так как оно задано по умолчанию, но наличие его более наглядно, ошибки при этом не возникает), и время окончания ИМ равно 150 временным дискретам.

Значительно интереснее пятый пример (рис. 5.4). Поскольку в МФ появилось два самостоятельных ОБ GENERATE, необходимо рисовать временные оси для движения транзактов из каждого источника, а временные засечки отдельных ОБ GENERATE сносить на ось абсолютного времени.

В момент времени 50 появляется транзакт с ОБ1 и отмечается также на оси абсолютного времени АС1, показания СС уменьшаются на единицу, приход первого транзакта с ОБ2 во временную дискрету 75 уменьшает показания СС еще на единицу, наконец, приход второго транзакта с ОБ1 и его терминирование останавливает процесс ИМ в момент времени 100.

Задание. Используя сеанс отладчика, создайте свой файл с несколькими ОБ и, используя пошаговое рассмотрение, разберитесь окончательно с логикой прекращения процесса ИМ.

5.3.2. Правило окончания по времени испытаний

Довольно часто в процессе ИМ требуется определить не средние значения атрибутов объектов, а значения, зависящие от времени: производительность, динамику процесса, суммарные затраты за какой-то период времени и т. д. В этом случае в качестве контрольной точки берется не число стартов, а период времени (рабочая смена, сутки, месяц и т. п.). Основным условием при этом является приведение всех рассматриваемых при моделировании интервалов времени к одному масштабу. Например, если предельное время задано в сутках, а темп прихода транзактов в минутах, то правильнее все интервалы времени исчислять в минутах. Для реализации правила окончания ИМ по времени программа должна проделать следующие действия:

1) принять предельный интервал абсолютного времени АС1пр. за условную единицу отсчета;

2) суммировать времена движения транзактов $\sum_{i=1}^{y} Tact$;

3) проверить условие АС1пр.

$$\sum_{i=1}^{y} Tact \le 0. \qquad (5.1)$$

Проиллюстрируем условие (5.1) на рис. 5.5. Первый уничтоженный транзакт отмечает свое время на оси АС1, второй приплюсовывает время своего движения к точке *T1*, третий — к точке *T2* и т. д., до тех пор, пока последний *y*-й интервал текущего времени транзакта не сравняется или не превысит значение АС1пр. На рисунке это значение *Ty* больше предельного значения;

4) выполнение условия (5.1) приводит к появлению машинной единицы (см. п. 1), которая сравнивается со значением операнда A =1 ОБ START, результат обнуляется, что приводит к выдаче сигнала о прекращении процесса ИМ.

Указанные действия программы реализуются следующим образом.

1. Обязательно вводится дополнительный программный модуль, называемый *таймером*, который учитывает накапливаемое время движения транзактов:

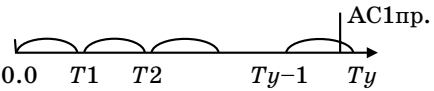


Рис. 5.5. *Иллюстрация правила окончания ИМ по времени*

```
GENERATE        АС1пр.
TERMINATE       1
```

Значение операнда А ОБ GENERATE равняется заданному предельному интервалу АС1пр., других операндов нет.

2. В обязательном порядке обнуляются значения операндов А всех ОБ TERMINATE модуля исполнения МФ.

3. Значение операнда А ОУ START устанавливается равным 1.

Рассмотрим этот простой алгоритм на примере 5, повторим этот пример, а затем модифицируем его в виде примера 5b:

```
        SIMULATE
        GENERATE    50
        TERMINATE   1
        GENERATE    75
        TERMINATE   1
        START       3
        END
```

```
5b.     SIMULATE
        GENERATE    50
        TERMINATE   0
        GENERATE    75
        TERMINATE    0
        GENERATE    300
        TERMINATE   1
        START       1
        END
```

Рассмотрим некоторые особенности примера 5b.

1. Транзакты в МФ не терминируются, а лишь отмечают факт своего прохода отметкой на оси АС1 (запомним это важное обстоятельство на будущее).

2. Таймер срабатывает при выполнении условия (5.1).

3. Показания операндов А ОБ TERMINATE и ОУ START вычитаются, обнуляются, что служит командой для прекращения процесса ИМ.

Для обратного перехода к управлению с помощью числа стартов необходимо:

1) исключить из МФ таймер;

2) записать в поле операнда А ОУ START желаемое число стартов;

3) записать в поля операндов А ОБ TERMINATE желаемые значения.

В примере 5b мы впервые сталкиваемся с весьма важным обстоятельством, когда 2 транзакта из разных источников приходят в одно

и то же время. Такие моменты в нашем примере наступают в 150 и 300 временных дискрет, в эти моменты транзакты становятся *связанными по времени*. Возникает вопрос, как они проходят по МФ? Нельзя забывать, что транзакты двигаются один за другим по одному в единицу времени. Для решения этой проблемы в GPSS/H предусмотрены четкие правила.

1. Транзакты одинакового приоритета записываются в СТС в порядке возрастания их ИН. Следовательно, проблема решается просто: вначале исполняется транзакт с меньшим ИН, а затем с большим, но имеющим одно и то же время. При этом засечка на оси АС1 будет одинаковой.

Предложение. В идеале следует записать данные примера в МФ и провести пошаговую проверку в тестовом режиме с помощью отладчика (дебагера), для чего обратиться к начальным сведениям гл. 7.

2. При разных приоритетах, если транзакты связаны по времени, первым движется транзакт, имеющий больший приоритет.

§ 5.4. ПРИНЦИПЫ ДВИЖЕНИЯ ТРАНЗАКТОВ

5.4.1. Общие основы движения транзактов

Напомним главные принципы, относящиеся к транзактам:

— транзакты являются динамическими элементами, движущимися от ОБ к ОБ;

— транзакты появляются в МФ с помощью одного или нескольких ОБ GENERATE;

— в каждый момент времени движется только один транзакт;

— каждый транзакт находится в *текущем* ОБ и пытается войти в *следующий* ОБ;

— жизненный цикл Хакт состоит из зарождения, движения, ожидания, обслуживания и разрушения, причем этапы движения, ожидания и обслуживания могут многократно повторяться;

— ряд ОБ никогда не препятствуют входу транзактов, ряд ОБ при определенных условиях препятствуют входу транзактов, ряд ОБ изменяют последовательность движения транзактов, ряд ОБ создают копии транзактов и объединяют их в ансамбли (см. § 5.1).

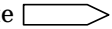
— транзакты могут переходить из одного списка в другой и обратно;

— транзакты за время жизненного цикла могут изменять приоритет и параметры;

— транзакты имеют строго индивидуальный ИН;

— каждый транзакт имеет свое время движения.

Для изображения на временной оси этапов жизненного цикла транзакта введем следующие обозначения:

— зарождение 
— движение
— ожидание
— обслуживание 
— разрушение 

Время воспринимается как событие, происходящее в системе. В GPSS/H воспроизводится МВ, которое используется для фиксации абсолютного времени АС1 процесса моделирования, которое складывается из последовательности времен движения транзакта и времен его обслуживания. Различия между абсолютным и относительным МВ будут определены в гл. 6. МВ записывается в виде целых положительных десятичных чисел, а также в виде положительных десятичных чисел с плавающей точкой. Отсчет времени начинается в момент 0.0, а в случае наличия операнда С у ОБ GENERATE отсчет времени движения первого транзакта сдвигается на это значение, а отсчет абсолютного МВ все равно начинается в точке 0.0. В пособии, при использовании записи времени числом с плавающей точкой, ограничимся одним десятичным знаком, хотя профессиональная версия программы поддерживает четыре десятичных знака. Сам исследователь выбирает дискрету МВ, задаваемую в модуле описания и используемую в течение всего процесса ИМ. Временные промежутки, характеризующие все виды объектов, должны быть представлены в одинаковом масштабе, т. е. в ИМ не могут одновременно использоваться секунды и сутки. МВ в процессе моделирования может только возрастать, и не существует ситуаций, допускающих обратный отсчет времени. Для иллюстрации сказанного рассмотрим фрагмент графика работы отдела технического контроля (табл. 5.1).

Таблица 5.1. *Фрагмент графика работы ОТК*

Событие	РВ	МВ
1. Начало работы	9.00	0.0
2. Поступление первой детали на контроль, начало контроля	9.24.30	24.5
3. Поступление второй детали, ожидание контроля	9.40.24	40.4
4. Поступление третьей детали, ожидание контроля	9.48.18	48.3
5. Окончание контроля первой детали, начало контроля второй детали	9.52.36	52.6

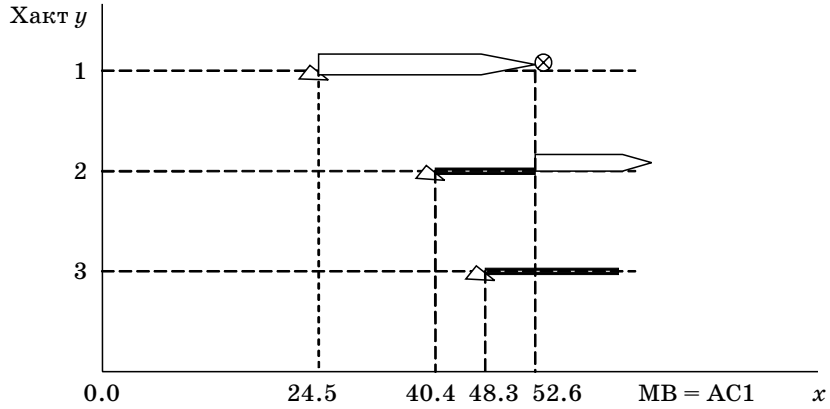


Рис. 5.6. *Временная эпюра фрагмента графика*

Примечание. РВ — реальное время — выражено в часах, минутах, секундах, а МВ — в минутах, числом с плавающей точкой с точностью одного десятичного знака.

На рис. 5.6 представлена временная эпюра фрагмента графика. По оси y отложены номера приходящих на контроль деталей (транзактов), по оси x — время их движения по МФ. X1 входит в модель 24.5 и сразу становится на контроль (обслуживание), меняя при этом состояние (см. п. 5.4.2). Как только это произошло, начинает движение X2, который поступает в 40.4 и, поскольку ему запрещен вход на пост контроля, становится в режим ожидания, сразу после этого начинает движение X3, который поступает в МФ в 48.3 и, поскольку пост контроля занят, также становится в очередь ожидания обслуживания вслед за X2 (по дисциплине обслуживания FCFS). В МВ 52.6 оканчивается обслуживание X1, который, так как не оговорено других условий, сразу терминируется. В этот же момент 52.6 начинается обслуживание X2, а X3 продолжает находиться в режиме ожидания и т. д. Следует различать настоящее и будущее время движения транзакта. Так, для X1 будущее время t_1 определится как момент начала моделирования $0.0 + t_1$, определяемое производящим его ОБ GENERATE, соответственно, для X2 его будущее время $t_2 = t_1 + t_2$ и т. д. Каждое будущее время отмечается на оси МВ = АС1, создавая непрерывную цепь событий.

При ИМ на GPSS/H можно выделить два режима:

— *пакетный* (см. гл. 6), автоматически реализуемый программой и создающий файл выходного отчета с расширением .lis;

— *тестовый* или отладочный (см. гл. 7), реализуемый исследователем с помощью отладчика (дебаггера).

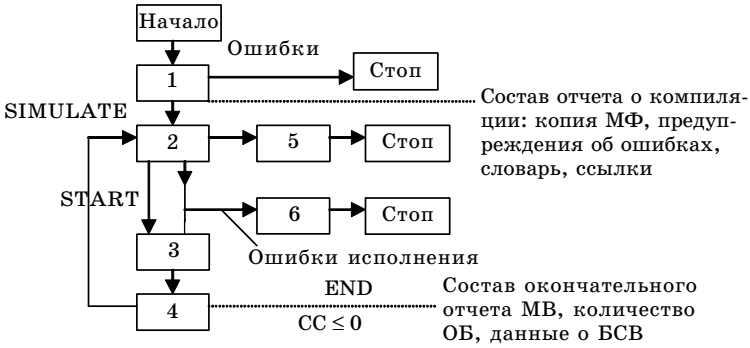


Рис. 5.7. *Этапы ИМ при пакетном режиме*

Рассмотрим основные этапы пакетного режима (рис. 5.7).

Блок 1 — фаза компиляции модели. Вначале МФ, записанный в редакторе, подвергается компиляции для превращения его во внутренний машинный документ, удобный для исполнения. Одним из подэтапов является составление отчета о компиляции, содержащего сведения, указанные на рисунке. Рассмотрим разницу между предупреждениями (Warning) и сообщениями об ошибке (Error message). Первые являются консультативными и предлагают обратить внимание на некоторые не очень четкие места модели (например, у ОБ записан ярлык, но обращений к нему нет). Моделирование при этом не останавливается. Сообщение об ошибке содержит информацию о кардинальных ошибках (неверно написаны коды операций, отсутствует необходимый операнд, забыто имя в одном из парных ОБ и т. д.). При этом появляется сообщение об ошибке и управление возвращается к командной оболочке по линии «Ошибки компиляции». При успешном прохождении процесса компиляции управление передается блоку *2* по линии SIMULATE.

Блок 2 — фаза исполнения ОУ. ОУ начинают исполняться сверху вниз в том порядке, в котором они записаны в рис. 5.6, а именно с исполнения ОУ START, инициализирующего СС и один или все существующие в МФ ОБ GENERATE. Затем по линии START осуществляется переход к фазе движения транзактов. При этом существование предупреждений допустимо, а ошибки исполнения приводят к остановке процесса ИМ (линия к блоку *6*).

Блок 3 — фаза движения транзактов (исполнение ОБ). Движение транзактов и исполнение ОБ происходит только во время этой фазы, при этом транзакты время от времени вводятся в модель, движутся по своей траектории от ОБ к ОБ и могут выводиться из модели (терминироваться). Существуют моделируемые ситуации, когда транзак-

ты (например, клерки в нотариальной конторе) не покидают модель, а только отмечают время выполнения своей операции. Во время этой фазы появление предупреждений допустимо. Фаза продолжится до выполнения условия CC ≤ 0, после этого создается окончательный отчет (линия CC к блоку *4*).

Блок 4 — фаза создания окончательного отчета. В отчете фиксируется все, что произошло при исполнении ОБ (см. рис. 5.6). Затем эти данные объединяются с данными об ОУ (линия к блоку *2*) и по линии END проходит команда об окончании процесса ИМ, выдается команда на подготовку окончательного отчета и после подготовки отчета — к возврату программы в командную оболочку.

Блок 5 — создание окончательного отчета. В отчете подводятся итоги исполнения МФ, выдаются данные о МВ, числе использованных ОБ, использовании памяти и т. д. (содержание отчета приведено в прил. 5). Возможности переполнения памяти COMMON и ее переназначения рассматривались в § 5.1.

Примечание. В коммерческих версиях GPSS/H, работающих в Windows 95,98, ME, XP, NT, никаких ограничений ни на объем памяти, ни на число ОБ, используемых в МФ, не существует.

Блок 6 — отчет об ошибках. Ошибки могут возникать во время выполнения фаз 2 и 3; как только эти ошибки обнаруживаются, программа прекращает исполнение фазы, выходит из процесса ИМ, создавая отчет об ошибках.

Можно привести много примеров, когда над транзактом не совершается активных действий, т. е. когда он свободно перемещается или находится в режиме ожидания (например, деталь перемещается по конвейеру, пациент ожидает в очереди у врача и т. п.), и когда транзакт находится в активном режиме (деталь обрабатывается на станке, банковский клерк обслуживает клиента и т. п.). Для создания активного режима используется ОБ ADVANCE (см. § 5.1). Довольно часто по логике МФ приходится менять направление движения транзактов, например с помощью ОБ TRANSFER. Все эти действия приводят к изменению состояния транзактов, о чем более подробно говорится в п. 5.4.2.

5.4.2 *. Принципы управления транзактами[1]

Язык ИМ GPSS/H обладает достаточно разветвленной логикой, которую не может изменить пользователь, но понимание внутрен-

[1] Примеры взяты из работы [17].

них процессов важно для осознания структуры ЯИМ и успешного использования тестового режима. Необходимо понимать логику размещения и статуса транзактов на разных этапах его жизненного цикла (ЖЦ). Внешняя организация со стороны пользователя позволяет оценить только вершину айсберга, а именно текущий ОБ и следующий ожидаемый ОБ. Внутренняя организация со стороны ЯИМ GPSS/ H основывается на концепции списков (в оригинале Chain — цепь), представляющих собой перечень (список) транзактов. В каждый момент времени каждый транзакт находится только в одном из списков, но в то же время размещается в ОБ. Обычно транзакты движутся из списка в список, аналогично их перемещению из ОБ в ОБ. Список имеет начало и конец, и транзакт занимает свое специфическое место в списке (в начале, в конце или где-то в промежутке). Его размещение тесно связано с порядком, по которому транзакт снова предпримет попытку начать свое движение.

Идея построения списка с начала до конца следующая (рис. 5.8). Каждый транзакт представлен прямоугольником со своим ИН, транзакты расположены в порядке, известном как «первый пришел — первый обслужился» (FCFS). В ЯИМ GPSS/H используется пять категорий списков:

1) список текущих событий — СТС;
2) список будущих событий — СБС;
3) списки пользователя — СП;
4) списки прерываний;
5) списки пар, ансамблей и групп.

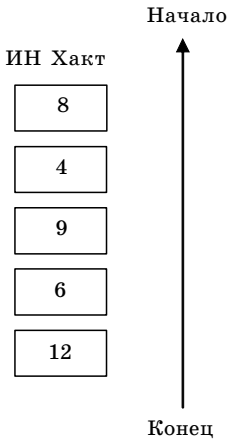


Рис. 5.8. Фрагмент СТС

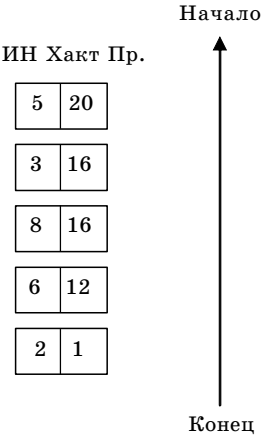


Рис. 5.9. Фрагмент СТС с учетом приоритета

Два последних списка нами не рассматриваются, поэтому им не присвоена аббревиатура. В ЯИМ GPSS/H существует только один СТС и только один СБС, другие списки могут вообще отсутствовать или, наоборот, их может быть несколько. Рассмотрим более подробно СТС и СБС.

Список текущих событий. В СТС входят все транзакты, которые могут начать движение в текущий момент МВ, но блокированы ОБ, запрещающими вход транзакту (например, SEIZE, ENTER) или условиями МФ. На рис. 5.9 приведен фрагмент СТС, транзакты в котором расположены в порядке убывания приоритета (на рисунке — Пр.), а при одинаковом приоритете — в порядке возрастания ИН. Из числа блокированных транзактов начать движение может первый в СТС с приоритетом 20. Транзакты на рисунке представлены разделенными пополам прямоугольниками, в которых слева стоит ИН, а справа — приоритет. Транзакт при одинаковом приоритете, пришедший последним, попадет на последнее место в своем приоритете. Учитывая, что транзакт одновременно находится и в ОБ, прямоугольник, представляющий собой транзакт, надо разделить на пять ячеек (рис. 5.10) для первых двух транзактов (см. рис. 5.9).

На рисунке приведен образ транзакта, включающий пять ячеек: первая — ИН; вторая — номер текущего ОБ (ТОБ); третья — номер следующего ОБ (СОБ); четвертая — время движения (ВДв) транзакта, т. е. время, когда Хакт сделает попытку перейти из текущего ОБ в следующий ОБ, но по определению все блокированные транзакты в СТС имеют одинаковое время настоящего момента, поэтому время не внесено; пятая — приоритет транзакта.

Список будущих событий. СБС включает в себя транзакты, не могущие в данный момент предпринимать попытку дальнейшего движения, пока не наступит будущее состояние. Можно отметить две причины такого поведения: 1) транзакты войдут в модель из ОБ GENERATE, когда наступит предназначенное для них время; 2) транзакты находятся в ОБ ADVANCE, как в своем текущем ОБ, и не пытаются менять состояние, пока не будет окончено обслуживание. Транзакты в СБС располагаются в порядке возрастания времени дви-

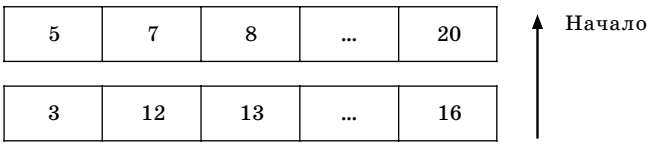


Рис. 5.10. *Фрагмент СТС с учетом нахождения Хакт в ОБ*

Рис. 5.11. *Фрагмент СБС*

жения *вне зависимости от приоритета*, а при одинаковом времени — в порядке возрастания ИН (рис. 5.11).

Необходимо пояснить смысл записи «Рожден» в ячейках ТОБ транзактов с ИН9 и с ИН4. Эта запись сделана для еще не произведенных ОБ GENERATE транзактов, но этот ОБ не может быть текущим блоком, так как он не допускает входа в него извне. ОБ является генератором (родителем) нового Хакт, вследствие этого использована запись «Рожден», т. е. впервые произведен. Каждый транзакт представлен прямоугольником с пятью ячейками, Хакт с наименьшим временем движения стоит в начале списка. Таким образом, ВДв диктует порядок расположения в СБС, транзакты с ИН9 и ИН4 вошли в модель из ОБ GENERATE, а остальные три — после нахождения на обслуживании в ОБ ADVANCE, транзакты с ИН2 и ИН11 имеют одинаковое значение ВДв, т. е. являются связанными по времени, первым начнет движение транзакт с ИН2, несмотря на то, что его приоритет ниже. Следует запомнить эту разницу между СТС (главенство приоритета) и СБС (главенство ИН). В принципе нежелательно иметь связанные по времени транзакты, и возможности генерации миллионов БСВ позволяют этого избегать, за исключением детерминированных значений времени (см. пример 5b § 5.3).

В этом разделе имеет смысл конкретизировать содержание фазы движения транзактов (блок *3* рис. 5.7), поскольку именно на этой фазе транзакты переходят из СБС в СТС и обратно. Эта фаза содержит два подэтапа:

— *сканирования,* в котором проверяется состояние модели в определенный текущий момент МВ;

— *изменения времени* за счет определения ожидаемого будущего времени.

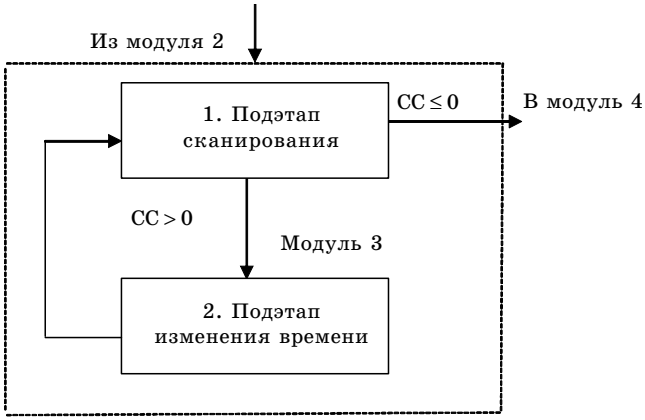


Рис. 5.12. *Подэтапы фазы движения транзактов*

Логическая связь между этими двумя подэтапами показана на рис. 5.12.

Подэтапы сканирования и изменения времени чередуются последовательно в течение всей фазы движения транзактов. Процесс ИМ начинается с подэтапа сканирования с целью определить состояние модели в момент времени 0.0, затем на подэтапе изменения времени определяется первый момент будущего времени, в которое один или несколько транзактов смогут начать движение. Затем на этапе сканирования проверяется состояние системы в новое время, на которое изменено абсолютное время, и т. д. до конца фазы движения транзактов, т. е. выполнения условия CC ≤ 0. Отметим, что этот процесс может быть закончен раньше, если тестовый режим будет прерван пользователем.

Прежде чем более подробно рассмотреть логику подэтапов сканирования и изменения времени, выясним, что происходит в модели после инициализации ОБ GENERATE. Алгоритм поведения системы для этого случая показан на рис. 5.13.

Блок 1 — инициализация каждого ОБ GENERATE. Инициализация проводится сверху вниз, в порядке их появления в МФ. Будущее время определяется относительно МВ=0.0.

Блок 2 — создание транзактов. Для каждого ОБ GENERATE создается один и только один транзакт со своим ИН, начиная с ИН1 в порядке их создания. Так, если в МФ имеется три ОБ GENERATE, то создаются X1, X2, X3 соответственно.

Блок 3 — проверка логического условия: наличие операнда C, не равного 0.

Рис. 5.13. *Алгоритм поведения после начала процесса ИМ*

Блок 4 — операнд С не равен нулю. Производится сдвиг начала времени движения первого транзакта (аналогично для всех первых транзактов ОБ GENERATE) по принципу 0.0 + время, определяемое операндом С.

Блок 5 — операнд С равен нулю. Устанавливается время, определяемое операндами А и В (порядок установления см. п. 5.1.1).

Блок 6 — проверка логического условия: равняется ли нулю время движения конкретного транзакта.

Блок 7 — ВДв равняется нулю. Размещение транзакта в СТС и мгновенное начало движения по модели.

Блок 8 — ВДв не равняется нулю. Размещение транзакта в СБС, что означает, что транзакт не может двигаться до момента достижения своего будущего времени.

Рассмотрим теперь подробнее подэтапы фазы движения транзактов.

Подэтап сканирования. На этом подэтапе программа дает возможность каждому из транзактов, находящихся в СТС, начать движение по МФ настолько, насколько это позволяет логика построения МФ. Порядок операций на подэтапе сканирования показан на рис. 5.14.

Перед началом подэтапа сканирования программа проверяет наличие запрета на проведение этого подэтапа (команда trap отладчи-

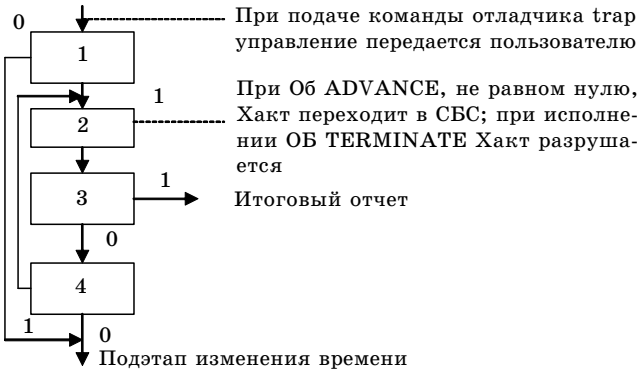


Рис. 5.14. *Логика действия подэтапа сканирования*

ка — см. гл. 7). Если такой запрет существует, то управление из пакетного режима переходит в режим контроля или отладки, осуществляемый пользователем. Если запрета нет, то осуществляется первый шаг, реализуемый блоком *1*.

Блок 1 — проверка логического условия: имеется ли Хакт в начале СТС. Если нет, то СТС пуст и осуществляется переход к подэтапу изменения времени.

Блок 2 — продвижение Хакт. При наличии Хакт в СТС он начинает двигаться до наступления одного из условий, приведенных на рис. 5.14, т. е. начинается либо обслуживание и соответствующий переход в СБС, либо уход Хакт из модели при терминировании.

Блок 3 — проверка логического условия: выполнено ли условие окончания процесса ИМ, т. е. СС ≤ 0. При выполнении следует команда на подготовку отчета и возвращения в командную оболочку. Если нет, то переход к блоку *4*.

Блок 4 — проверка логического условия: существуют ли еще Хакт в СТС. Если да, то переход к блоку *2* для продвижения следующего Хакт; если нет, то переход к подэтапу изменения времени.

Таким образом, на подэтапе сканирования необходимо выяснять, по какой причине Хакт не может двигаться дальше: если он попадает в ОБ TERMINATE, то он вообще исключается из СТС и его ИН уже не используется ни одним транзактом. Если же Хакт попадает в ОБ ADVANCE, то он задерживается в нем на некоторый интервал МВ и не делает попытки дальнейшего движения до наступления ближайшего момента его будущего времени, а это означает, что Хакт переводится из СТС в СБС. Следовательно, ОБ ADVANCE переводит Хакт из одного списка в другой, при значении операнда А ОБ ADVANCE равным нулю (отсутствие по умолчанию, 0 или 0.0) Хакт не перево-

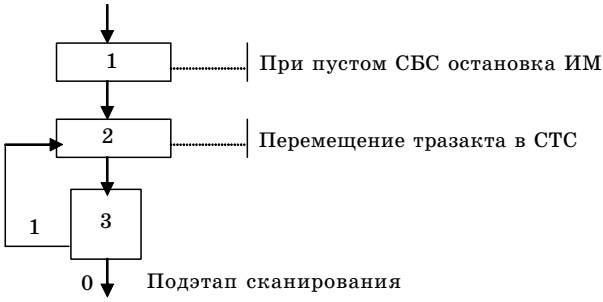


Рис. 5.15. *Логика действия подэтапа изменения времени*

дится в СБС, остается в СТС и немедленно делает попытку начать движение в свой следующий ОБ.

Подэтап изменения времени. При начале этого подэтапа происходит изменение МВ до значения времени движения Хакт, стоящего во главе СБС (времена в СБС располагаются в порядке возрастания). Следовательно, на этом подэтапе Хакт с наименьшим будущим ВДв перемещается из СБС обратно в СТС и начинает движение, согласуясь с приоритетом транзактов, находящихся в СТС.

Если в СБС минимальное время соответствует нескольким транзактам с одинаковым временем, они переводятся в СТС друг за другом в порядке их ИН, а в СТС начинают движение в порядке старшинства приоритета. Логика этого подэтапа представлена на рис. 5.15.

Блок 1 — размещение минимального значения ВДв транзакта в начало списка СБС. Если при этом выясняется, что СБС пуст, то появляется сообщение об ошибке № 410 «No next event in System» — не существует следующего события, и процесс ИМ останавливается.

Блок 2 — переводит транзакты из СБС в СТС и помещает транзакт последним в группу списка СТС, имеющую одинаковый приоритет с переводимым транзактом.

Блок 3 — проверка логического условия: помещено ли ВДв Хакт в начало СБС для оценки времени. Если нет, то управление передается подэтапу сканирования; если да, то управление передается блоку *2*.

Таким образом, на этом подэтапе происходит реальное движение одного или нескольких (при связи по времени) транзактов из списка СБС в СТС, но не из ОБ в ОБ. На подэтапе сканирования происходит реальное перемещение Хакт от ОБ к ОБ и потенциально из списка в список.

5.4.3*. Общие основы изменения флага состояния модели и индикатора состояния Хакт

В языке ИМ GPSS/H существуют парные ОБ (например, SEIZE/RELEASE, ENTER/LEAVE и др.), блокирующие и, соответственно, разблокирующие дальнейшее продвижение транзактов. Индикатором блокирования и снятия блокировки движения Хакт в программе служит *флаг изменения состояния* (ФИС) (Status Change Flag), отображаемый на экране отладчика в виде s/c. ФИС может находиться в состоянии включено / выключено (on /off), которое является сигналом к перезапуску подэтапа сканирования. Например, исполнение ОБ SEIZE приводит к переключению ФИС, так как состояние устройства обслуживания меняется с состояния готовности на состояние занятости. Состояние блокировки длится до момента исполнения другим транзактом ОБ RELEASE. Транзакт, блокированный перед ОБ ADVANCE и ждущий обслуживания, носит название *однозначно (уникально) блокированного*, потому что только исполнение ОБ RELEASE снимает с него блокировку. Если Хакт блокирован, то не существует условий, позволяющих ему двигаться за время фазы сканирования, пока не произошло обновление времени. Такое свойство программы значительно экономит время моделирования, так как не разрешает предпринимать попытки продвижения блокированных транзактов. Для реализации преимущества экономии времени каждому Хакт закреплен индикатор перехода к сканированию (ИПС) (Scan-Skip Indicator — SSI), также имеющий два состояния: включено /выключено. Если движущийся Хакт подвергается блокированию, программа переключает ИПС и Хакт остается в СТС, таким образом становится известно, что Хакт уникально блокирован. Кратко рассмотрим, как вышесказанное реализуется в программе.

Транзакты, находящиеся в СТС во время подэтапа сканирования, вначале проверяются на вид состояния ИПС. Если индикатор включен, транзакт блокирован и не может двигаться дальше, программа немедленно ищет следующий Хакт в СТС и пытается двигать его. Если индикатор выключен, транзакт не блокирован и программа пытается продвинуть его из текущего ОБ в следующий предполагаемый ОБ. Транзакты с включенным ИПС считаются неактивными, и программа при осуществлении подэтапа сканирования пропускает их. Такие транзакты в GPSS/H называются «заснувшими» (asleep). В отличие от них, транзакты с выключенным ИПС являются активными, программа пытается продвигать их в следующие ОБ, они носят название «пробудившихся» (awake). Переход от активного к пассивному состоянию происходит при снятии условий блокирования.

Если несколько транзактов блокированы перед ОБ, запрещающим вход, например SEIZE или ENTER, исполнение ОБ RELEASE или LEAVE пробуждает эти транзакты и переключает ИПС, а также ФИС. После разблокирования перезапускается подэтап сканирования, давая возможность пробудившимся разблокированным транзактам двигаться к их следующим ОБ. В связи с этим логика фазы сканирования (см. рис. 5.14) должна быть расширена за счет включения ФИС и ИПС (рис. 5.16).

Блок 1 — проверка логического условия: есть ли Хакт в начале СТС. Если есть, то управление передается блоку *2*; если нет, то переход к подэтапу изменения времени.

Блок 2 — проверка логического условия: активен ли (пробудился ли) Хакт? Если активен, то управление передается блоку *3*; если нет, то к блоку *7*.

Блок 3 — продвижение Хакт по его траектории с уточненными условиями:

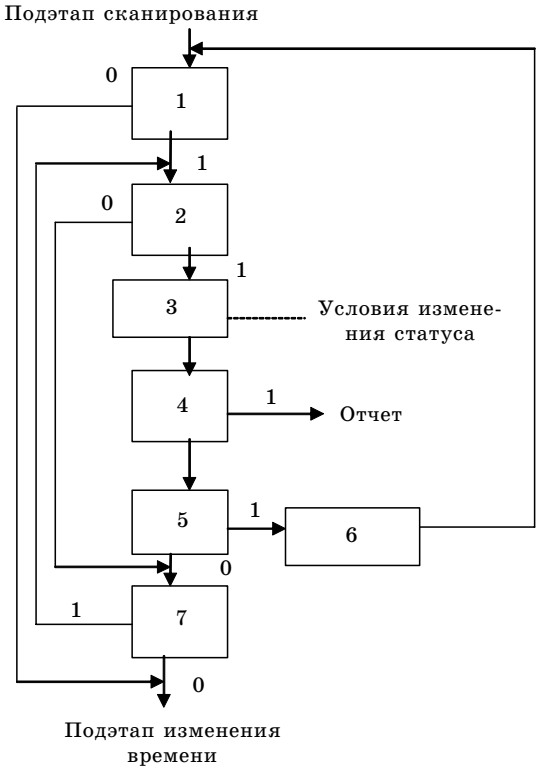


Рис. 5.16*. Роль ИПС и ФИС на подэтапе сканирования*

— изменение статуса, т. е. включение ФИС и пробуждение всех уснувших транзактов;

— при ненулевом времени обслуживания — помещение этого Хакт в СБС;

— при терминировании — удаление транзакта из модели;

— при запрете входа в следующий ОБ — блокирование, оставление транзакта в СТС и перевод в неактивное (заснувшее) состояние.

Блок 4 — проверка логического условия: равно ли нулю состояние счетчика свершений. Если равно или меньше нуля, то готовится выходной отчет; если нет, то управление передается блоку *5*.

Блок 5 — проверка логического условия: включен ли ФИС. Если да, то управление передается блоку *6*; если нет, то блоку *7*.

Блок 6 — переключение ФИС в состояние выключено.

Блок 7 — проверка логического условия, есть ли Хакт в СТС. Если есть, то управление передается блоку *2*; если нет, то переход к подэтапу изменения времени.

Таким образом, введение в программу ФИС модели и индикатора каждого транзакта о его активном или не активном состоянии – ИПС делает логику работы весьма гибкой и экономит машинное время, так как неактивные (спящие) транзакты не просматриваются во время подэтапа сканирования. На первом шаге при сканировании выясняется состояние ИПС транзакта, и пробужденные получают возможность движения (см. блок *3*). Важно отметить, что исполнение ОБ, снимающего блокировку, снимает блокировку и пробуждает все спящие транзакты, но только один из них начнет движение. Когда же первый по приоритету или ИН Хакт начнет движение, происходит исполнение ОБ SEIZE или ENTER, что сразу явится командой к засыпанию остальных транзактов, которые были активизированы тактом раньше. Во время работы программы последовательность засыпаний и пробуждений длится до тех пор, пока в очереди на обслуживание существуют транзакты. Отметим также, что условия прерывания, реализуемые в отладчике командой trap, передающей управление пользователю, действительны только для активных транзактов.

§ 5.5. РАЗБОР ОШИБОК И УПРАЖНЕНИЯ

В § 5.1 подробно рассмотрены многие виды операторов, используемых в GPSS/H. Ниже будут приведены примеры возможных ошибок, возникающих при написании этих операторов в МФ, а также рассмотрены простые примеры использования некоторых ОБ. Ошиб-

ки рассматриваются непосредственно в тексте параграфа, а ответы на упражнения приведены в прил. 5. Первые упражнения рассчитаны только на развитие навыков применения простых моделей и могут не требовать непосредственного использования программы, такие примеры снабжены тильдой ~, для более сложных упражнений целесообразно вначале разобраться с логикой работы отладчика (см. гл. 7), написать МФ и исследовать его в тестовом режиме с помощью программы. В гл. 6 и 7 также будут приведены упражнения, но их решение потребует некоторых навыков в написании программ на GPSS/H.

5.5.1. Разбор ошибок

GPSS/H обладает способностью эффективно обнаруживать ошибки. Для того чтобы с большей очевидностью понять это, приведем несколько примеров, содержащих те или иные ошибки. Каждому примеру дается пояснение, которое следует читать после неуспешной попытки самому ответить на вопрос, звучащий в примере, при этом приветствуется возможность самим проверить эти примеры с помощью отладчика. Полезно иметь в виду, что если программа начнет работать неправильно, всегда можно прервать ее исполнение нажатием клавиш CTRL-BREAK или, если вы находитесь в тестовом режиме, набором в командной строке отладчика клавиши быстрого выхода QQ (Quit Quckly) с последующим нажатием Enter.

1.
```
SIMULATE
GENERATE    20,30
TERMINATE   1
START       5
END
```

Вопрос: Какие проблемы у этой модели, появится ли сообщение об ошибке?

Пояснение: Такая запись не верна в принципе, так как если модификатор (операнд B) больше операнда A, то значение MB окажется отрицательным, а оно всегда положительно, время не может двигаться в обратном направлении и в начальный момент имеет значение 0.0. При этом появляется запись о возникновении ошибки компиляции “B Operand (modifier) exceeds A operand (mean)”.

2.
```
SIMULATE
GENERATE    100
TERMINATE   0
START       1
END
```

Вопрос: Что произойдет с процессом ИМ?

Пояснение: Это пример процесса ИМ, вышедшего из-под контроля, программа не имеет возможности обнулить показания СС, так как операнд А ОБ TERMINATE равен 0. В этом случае не появляется никаких сообщений об ошибках, ИМ продолжается до тех пор, пока не будет прервано пользователем.

Предупреждение! В МФ обязательно должен быть хотя бы один значащий операнд А у тех ОБ TERMINATE, которые имеются в МФ.

```
3.  SIMULATE
    GENERATE    35,15
    TERMINATE   1
    GENERATE    20,10
    START       3
    END
```

Вопрос: Какая ошибка в этом МФ? Появится ли сообщение об ошибке?

Пояснение: В данном примере не определен следующий ОБ, в который может двигаться транзакт со второго ОБ GENERATE. Такое построение МФ вызывает появление ошибки № 408 – не определенный ОБ (indeterminate Block) и остановку процесса ИМ.

```
4.  SIMULATE
    GENERATE    35,15
    GENERATE    20,10
    TERMINATE   1
    START       3
    END
```

Вопрос: Какая ошибка в этом МФ? Появится ли сообщение об ошибке?

Пояснение: В данном примере делается попытка войти в ОБ GENERATE, что категорически запрещено логикой работы программы (см. § 5.1). Появляется сообщение об ошибке исполнения № 409 “Attempt to enter a GENERATE Block”.

```
5.  SIMULATE
    GENERATE    50,25,–10
    TERMINATE   1
    START       3
    END
```

Вопрос: Что не правильно в этом МФ? Появится ли сообщение об ошибке?

Пояснение: Наименьшее модельное время может иметь значение 0.0. Время прихода первого транзакта (время сдвига) — операнд С не

может иметь отрицательного значения. Появляется сообщение об ошибке компиляции “GENERATE: offset (operand C) is negative”.

```
6.  SIMULATE
    GENERATE    100,,,4
    TERMINATE   1
    START       5
    END
```

Вопрос: Какие проблемы в этом МФ? Появится ли сообщение об ошибке?

Пояснение: Операнд D ОБ GENERATE определяет число Хакт, пришедших с указанного генератора, после прохода 4-го, последнего транзакта счетчик свершений еще не обнуляется и предпринимается попытка продолжать процесс ИМ, но транзактов больше нет. Появляется сообщение об ошибке № 410 “No Next Event in System”, и если вы находитесь в тестовом режиме, управление передается пользователю для корректировки МФ, а при пакетном режиме происходит возврат в командную оболочку.

```
7.  SIMULATE
    GENERATE    20,10
    ADVANCE     10,15
    TERMINATE   1
    START       3
    END
```

Вопрос: Что не верно в этом МФ? Появится ли сообщение об ошибке?

Пояснение: Операнд B ОБ ADVANCE, так же как и операнд B ОБ GENERATE, не может быть больше операнда A, иначе значение MB станет отрицательным, что недопустимо. Появится такое же сообщение, как в примере 1.

Очевидно, что для рассмотрения этих примеров читателю пришлось обратиться к помощи отладчика (см. гл. 7). Используя созданные вами собственные файлы, проведите отработку навыков, увеличивая и уменьшая значения операндов, исключая некоторые ОУ, например ОУ START, и наблюдайте за результатами, используя тестовый режим.

5.5.2. Упражнения

Предупреждение! Прежде чем начинать решение упражнений, внимательно прочтите введение в прил. 5 «Ответы к упражнениям»!

```
1. ~    SIMULATE
        GENERATE    100
        TERMINATE   1
```

```
            START       3
            END
```

Вопрос: В какое время кончится процесс ИМ?

```
2.~         SIMULATE
            GENERATE    50
            TERMINATE   1
            GENERATE    35
            TERMINATE   1
            START       3
            END
```

Вопрос: В какое время кончится процесс ИМ? Какой ИН будет у последнего Хакт?

```
3.          SIMULATE
            GENERATE    60,30
            TERMINATE   1
            GENERATE    50,25
            TERMINATE   2
            START       5
            END
```

Вопрос: В какое время кончится процесс ИМ? Чему будет равняться значение СС? Какие ИН Хакт примут участие в ИМ?

```
4.          SIMULATE
            GENERATE    50
            ADVANCE     250,250
            TERMINATE   1
            START       5
            END
```

Вопрос: В каком порядке Хакт (в соответствии с их ИН) будут терминироваться? Может ли Хакт с большим ИН обогнать Хакт с меньшим ИН? В какое время кончится процесс ИМ?

```
5.          SIMULATE
            GENERATE    100,50
            ADVANCE     125,110
            ADVANCE     75,40
            TERMINATE   1
            START       5
            END
```

Вопрос: Можно ли сложить операнды двух последовательных ОБ ADVANCE? В какое время кончится процесс ИМ и какой при этом порядок следования Хакт?

```
6.~         SIMULATE
            GENERATE    50
```

```
            ADVANCE     50
            TERMINATE   1
            GENERATE    75
            ADVANCE     0
            TERMINATE   1
            START       3
            END
```

Вопрос: Какой из ОБ TERMINATE даст сигнал к окончанию процесса ИМ? В какое время кончится процесс ИМ?

7. ~
```
SIMULATE
            GENERATE    25
            TERMINATE   1
            GENERATE    25
            TERMINATE   1
            START       6
            END
```

Вопрос: В каком порядке ИН будут двигаться транзакты?

8.
```
            SIMULATE
            GENERATE    50,,,,3
            TERMINATE   1
            GENERATE    25,,,,7
            TERMINATE   1
            START       6
            END
```

Вопрос: В какое время кончится процесс ИМ? В каком порядке будут двигаться транзакты?

9.
```
            SIMULATE
            GENERATE    10,,,4
  REPEAT    ADVANCE     40,10
            TRANSFER    ,REPEAT
*
            GENERATE    100             Временной таймер
            TERMINATE   1
*
            START       1
            END
```

Вопрос: Каково состояние модели в МВ, равное 100? Сколько транзактов одновременно находится в ожидании перед ОБ ADVANCE в момент окончания процесса ИМ? Меняется ли порядок движения Хакт после ОБ ADVANCE? Напомним, что в данной ситуации Хакт не терминируются, а остаются в модели, а временной таймер задает правило остановки процесса ИМ.

Глава 6
МОДЕЛИ ОДНОКАНАЛЬНОГО И МНОГОКАНАЛЬНОГО ОБСЛУЖИВАНИЯ

§ 6.1. ОБЩИЕ ПРЕДСТАВЛЕНИЯ

В настоящей главе на примере конкретных систем будут рассмотрены общие принципы построения и реализации моделей с использованием возможностей GPSS/H. Основные операторы и их операнды, форматы записи и другие особенности ЯИМ GPSS/H были рассмотрены в гл. 4 и 5, в настоящей главе они используются без пояснений. При необходимости уточнить какие-то детали следует обратиться к указанным главам, а также к прил. 1, в котором приведены все операторы ЯИМ, как упоминаемые, так и не вошедшие в пособие, но используемые в GPSS/H.

Устройство обслуживания (сервер) в случае одноканального обслуживания представляется в GPSS/H устройством (Facility) и реализуется парными ОБ SEIZE /RELEASE, а в случае многоканального обслуживания с использованием идентичных серверов — памятью (Storage) и реализуется парными ОБ ENTER /LEAVE. Под каналом будем понимать весь путь (траекторию) прохождения Хакт по МФ от зарождения до выхода из системы, причем траектория может включать в себя комбинацию различных устройств и памятей. Основные отличия между указанными парными ОБ рассмотрены в п. 5.1.1. Физическая природа серверов может быть самой разнообразной: это либо люди (врач, кладовщик, клерк, кассир и т. д.), либо предметы (станок, конвейер, погрузочная эстакада, принтер, операционная и т. д.). Поскольку серверы являются частью моделируемой системы, то они воспринимаются как ресурсы этой системы, а так как число серверов обычно ограничено, то возникает задача рационального использования ресурсов. Отдельно взятый сервер может выполнять в какое-то время только одну операцию. Необходимо четко определять состояния, в которых может находиться сервер, таких состояний может быть три:

1) *готовности*, когда сервер не занят, но готов приступить к выполнению своих функций;

2) *занятости*, когда сервер обслуживает поступивший Хакт, при этом все остальные находятся в режиме ожидания (см. § 5.4);

3) *недоступности*, когда сервер либо не задействован в связи с графиком работы (перерыв, ночное время, отпуск), либо находится в режиме восстановления или профилактического обслуживания.

В приведенных ниже примерах состояние недоступности будет рассматриваться лишь в том случае, если это оговаривается условиями задачи.

Имеет смысл более подробно остановиться на рассмотрении состояния занятости. В ЯИМ GPSS/H предусмотрена возможность прерывания процесса обслуживания Хакт с низким приоритетом другим Хакт с более высоким приоритетом. Такой режим реализуется парным ОБ PREEMPT/RETURN. В пособии этот режим не рассматривается. Когда сервер находится в состоянии занятости, перед ним в режиме ожидания может оказаться несколько Хакт. В этом случае необходимо договариваться о виде дисциплины обслуживания (ДО), типы которой кратко рассмотрены в § 2.2, а ниже более подробно рассмотрим виды ДО, применяемые в пособии.

1. ДО FCFS (first come, first serve) — см. § 2.2. Именно на основе этой ДО построен СТС, задающий хронологический порядок обслуживания.

2. ДО FCFS внутри своего приоритета. При этой ДО транзакты располагаются в порядке убывания приоритетов, а внутри своего класса приоритета обслуживаются по ДО FCFS.

3. ДО SPT (shortest processing time) — кратчайшее время обслуживания (обработки). При этой ДО транзакты выстраиваются в порядке возрастания времени обслуживания, причем, чем меньше время обслуживания, тем больший приоритет присваивается транзакту. В случае равенства времен обслуживания вступает в силу ДО FCFS.

Когда время ожидания обслуживания оказывается большим, чем предполагалось, транзакт может выбрать другие линии поведения:

— отказ от обслуживания (потеря заявки на обслуживание);

— выбор другой траектории, что реализуется ОБ TEST или TRANSFER.

Таким образом, за время ЖЦ Хакт проходит через ряд состояний одноканального и многоканального обслуживания (рис. 6.1, *а*, *б*), которые могут неоднократно повторяться в процессе ИМ. Еще раз отметим, что многоканальное обслуживание может быть реализовано только для группы идентичных серверов (обслуживание с общей очередью). В случае, если серверы не идентичны (станки разного типа, операторы разной квалификации, автомашины разной грузоподъемности и т. д.), то моделирование проводится по каждому серверу в отдельности, причем сервер воспринимается как устройство (обслуживание с разными очередями).

Для единообразного представления примеров, приводимых в дальнейшем, примем следующий алгоритм.

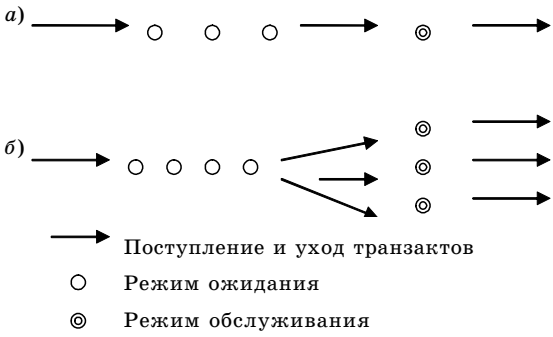


Рис. 6.1. *Эпюра одноканального (а) и многоканального (б) обслуживания*

1. Постановка задачи. Описывает с требующейся степенью детализации задачу, которую необходимо промоделировать. По мере приобретения навыков читателю после уяснения задачи следует попытаться самостоятельно построить МФ.

2. Допущения, принятые в модели. Уточняют, каким образом особенности моделируемой системы реализуются средствами GPSS/H.

3. Таблица определений. Устанавливает соответствие между элементами системы и объектами ЯИМ GPSS/H. Кроме того, в обязательном порядке задается значение временной дискреты (ВД), которая является масштабным значением для всех используемых в модели интервалов времени (например, если ВД принята равной одной минуте, то все другие временные интервалы переводятся в минуты).

4. Модельный файл. Представляется МФ, созданный исследователем. После успешного прохождения процесса симуляции создается эхо МФ. Для простоты обращения к обозначениям эха МФ, а также разделов итогового отчета все пояснения и перевод приведены в прил. 4.

5. Итоговый отчет. Выводы итогового отчета являются основой для обсуждения, подробно названия пунктов отчета приведены в прил. 4.

6. Выводы и обсуждение. В этом разделе обсуждаются логика построения модели, касающаяся использования ОБ, и исполнение модели, осуществляемое ОУ.

§ 6.2. ОДНОКАНАЛЬНОЕ ОБСЛУЖИВАНИЕ

Выше было установлено, что МВ при одноканальном обслуживании складывается из времени подхода к серверу (всем понятный пример расчетной кассы в супермаркете), ожидания обслуживания, об-

служивания и ухода. Естественно, что при немедленном занятии сервера пришедшим Хакт время ожидания равно нулю. Время подхода реализуется ОБ ADVANCE, время обслуживания — комбинацией ОБ SEIZE—ADVANCE—RELEASE. Блокирующие свойства ОБ SEIZE рассмотрены подробно в § 5.4. Блокированные транзакты остаются в ОБ, предшествующих ОБ SEIZE, и при снятии блокирования (при исполнении ОБ RELEASE) будут стремиться друг за другом войти в устройство, так как оно оказывается в состоянии готовности. Обслуженный же транзакт будет предпринимать попытку немедленного продвижения в ОБ, следующие за ОБ RELEASE. Для моделирования состояния недоступности используются ОБ FAVAIL — устройство доступно, т. е. включено или находится в рабочем состоянии и ОБ FUNAVAILE — устройство не доступно, т. е. выключено или находится в режиме профилактики. Если в МФ эти ОБ отсутствуют, то устройство считается по умолчанию на 100 % доступным. В пособии эти ОБ не используются. При рассмотрении одноканального обслуживания необходимо обсудить некоторые тонкие моменты появления временного рассогласования при последовательном расположении ОБ GENERATE и ОБ, препятствующих входу Хакт, например ОБ SEIZE. В связи с тем, что ОБ, препятствующих входу Хакт, немного, об этой проблеме не задумываются. Тем не менее рассмотрим случай, когда следующим после ОБ GENERATE расположен ОБ SEIZE. Рассмотрим в качестве примера МФ

```
SIMULATE
GENERATE        10
SEIZE           SERVER
ADVANCE         15
RELEASE         SERVER
TERMINATE       1
START           25
END
```

В примере Хакт поступают в 10,20, 30, ... ВД, но без задержки на обслуживание поступит только X1; X2 мог бы прийти в 20, но начнет движение только в 25, когда исполнится ОБ RELEASE для X1; X3 мог бы прийти в 35, но обслуживаться начнет только в 40, при исполнении ОБ RELEASE для X2, и т. д., т. е. появляется временной сдвиг между попаданием Хакт в модель и началом его продвижения в следующий ОБ, т. е. фактически Хакт движутся не через 10 дискрет, как им предписано, а через 15 из-за временного сдвига. Для исключения этого обстоятельства между первым и вторым ОБ можно включить фиктивный ОБ ADVANCE со значением операнда A, равным нулю, поскольку этот ОБ никогда не препятствует входу Хакт и не влияет на логику работы модели.

Рассмотрим некоторые особенности использования ОБ в модели.

Использование нескольких ОБ, препятствующих входу Хакт, включенных последовательно. Например, в цеху существует несколько разных станков, но всего один электрокар, доставляющий детали к станкам, тогда надо дождаться совпадения двух событий — освобождения кара и освобождения станка. В этом случае ОБ SEIZE со своими именами следуют друг за другом и построение МФ зависит от логики, сформулированной в постановке задачи. При исполнении МФ ОБ можно разделить на две группы: с нулевым и с ненулевым временем входа.

К первому типу ОБ можно отнести ОБ ADVANCE 0, RELEASE, которые отличают следующие свойства:

1) они никогда не препятствуют входу транзактов;

2) при входе в них Хакт стремится немедленно их покинуть;

3) последовательность таких ОБ определяется одной выходной траекторией.

Время исполнения последовательности таких ОБ всегда одинаково, поэтому хронологический порядок прохождения последовательности не имеет никакого значения, а сами ОБ располагаются в произвольном порядке.

Примерами с ненулевым временем являются ОБ SEIZE, ADVANCE A, TERMINATE, TRANSFER, GENERATE. Так, ОБ SEIZE не удовлетворяет свойству 1, ОБ ADVANCE A и ОБ TERMINATE не удовлетворяют свойству 2, ОБ TRANSFER не удовлетворяет свойству 3, а ОБ GENERATE не допускает входа в него.

Использование фиктивных устройств. Иногда у исследователя возникает желание получить статистику по использованию конкретного устройства при исполнении конкретной операции, такую информацию нельзя получить непосредственно из итогового отчета. Поэтому приходится вводить новое фиктивное устройство, собирающее часть информации, интересующей исследователя. Тонкость заключается в том, что ОБ SEIZE фиктивного устройства не препятствует входу транзакта, что достигается отсутствием в последовательности истинного и фиктивного устройства в рассматриваемый момент времени какого-либо другого Хакт, кроме изучаемого. Отсюда следует, что фиктивное устройство всегда находится в состоянии готовности, а его ОБ SEIZE всегда является следующим ОБ. Фиктивное устройство никогда не служит целям увеличения ресурсов, а лишь выступает как редко используемый инструмент сбора информации.

Использование разных точек отсчета движения транзактов. Иногда интересно рассматривать фактическое состояние модели в какой-то момент после начала работы, так как разные транзакты

могут находиться в разном состоянии. Чтобы эти рассуждения не оказались голословными, рассмотрим вначале пример 6.1, а затем приведем модификацию примера для разных начальных условий.

Естественно, что невозможно рассмотреть все нюансы построения моделей, но и приведенные соображения иллюстрируют возможности ЯИМ GPSS/H.

Рассмотрим несколько примеров одноканального обслуживания с использованием различных ОБ.

6.2.1. Схема одноканального обслуживания без ухода Хакт из модели

Рассмотрим вначале не совсем типичную схему одноканального обслуживания (пример 6.1) без ухода Хакт из модели. В этой схеме роль регулятора, возвращающего Хакт к началу МФ, играет ОБ TRANSFER, используемый в безусловном виде. Для рассмотрения различных начальных условий представлен модифицированный фрагмент этого примера (пример 6.1а).

Пример 6.1. Модель работы нотариальной конторы

1. Постановка задачи.

Работу нотариальной конторы можно разделить на два этапа: 1) подготовка документов клерками, что занимает 30±5 мин; 2) клерк несет подготовленный им документ для регистрации, постановки необходимых печатей и подписи у менеджера, что занимает 8±2 мин, — после чего клиент уходит, а клерк начинает работу с очередным клиентом.

Посещение менеджера моделируется дисциплиной обслуживания FCFS. Клерк действует независимо и не влияет на деятельность другого клерка.

На рис. 6.2 показаны схемы движения начальных запросов клиентов и выход подготовленных документов, а также передвижение клерков от своего рабочего стола в кабинет менеджера и обратно к своему рабочему месту. В данном примере транзактами являются клерки, а устройством — менеджер. ВД примем равной одной минуте и предположим, что в начале моделирования каждый менеджер начинает подготовку нового документа. Промоделируем работу конторы за пятидневную рабочую неделю с продолжительностью рабочего дня 8 ч, т. е. за 40 ч работы. В рассмотрение не принимаются обеденные перерывы, перерывы другого рода, сдвиг начальных условий и т. п. В результате моделирования следует ответить на вопрос, сколько документов будет подготовлено и какова эффективность работы клерков и менеджера? Будем полагать, что клиенты приходят в заданном выше темпе и их отсутствие не учитывается в модели.

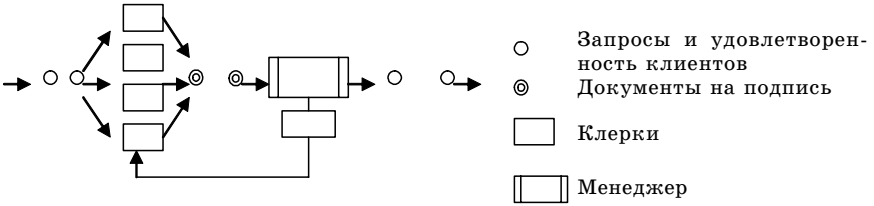


Рис. 6.2. *Схема движения клерков и документов*

2. Допущения, принятые в модели.

1. В конторе имеется только один менеджер, реализуемый в программе устройством, распоряжающийся необходимыми печатями.

2. В конторе работает фиксированное число клерков, представляющих собой транзакты, не выходящие из системы и не проходящие через ОБ TERMINATE.

При этих допущениях поток документов является результатом деятельности клерков. Естественно, что клерки-транзакты находятся в СТС при ожидании приема у менеджера и переходят в СБС, когда они обслуживаются менеджером. (В примере 6.1а рассмотрена ситуация неодновременности начала работы каждого клерка, в случае многоканального обслуживания клерков можно представить памятями.) Количество клерков задается операндом D ОБ GENERATE и транзакты возвращаются к начальной позиции подготовки нового документа с помощью ОБ TRANSFER. Для того чтобы промоделировать процесс функционирования за 40 ч, обнуляется операнд A основного ОБ TERMINATE, затем вводится дополнительная пара ОБ — временной таймер (см. § 5.3), Хакт которого является элементом управления по длительности моделирования, а не непосредственным участником процесса ИМ.

3. Таблица определений (табл. 6.1).

Временная дискрета: 1 мин

Таблица 6.1

Объект GPSS/H	Объект в системе
Транзакты Фрагмент 1 Фрагмент 2	 Клерки Хакт управления
Устройство MANAGER	Менеджер

4. Модельный файл.

Модельный файл разбит на модули (см. § 4.4) и внутри модуля исполнения — на фрагменты, знак * позволяет симулятору не читать строки, являющиеся комментарием.

```
*                Модуль описания
                 SIMULATE                Пример 6.1. Нотариальная контора
*
```

```
*           Модуль исполнения       Временная дискрета: 1 мин
*
*           Фрагмент 1
*
            GENERATE    0,,,4       число клерков
REPEAT      ADVANCE     30,5        время подготовки документа
            SEIZE       MANAGER     переход к менеджеру
            ADVANCE     8,2         проведение регистрации
            RELEASE     MANAGER     уход от менеджера
            TRANSFER    ,REPEAT     начало новой операции
*
*           Фрагмент 2              временной таймер
*
            GENERATE    2400        моделирование в течение 40 ч
            TERMINATE   1           сигнал на СС,
*                                   прекращение движения транзактов
*           Модуль управления
*
            START       1           установка СС
            END                     окончание моделирования
```

Примечание. Модельное эхо этого МФ (результат успешной компиляции) приведено в прил. 4, предваряя описание граф итогового отчета.

5. Итоговый отчет.

Отчет приведен в п. 2 прил. 4.

6. Выводы и обсуждение.

1. Логика модели основана на использовании одноканального обслуживания с применением одного и того же набора транзактов, проходящих через сервер неоднократно. Такая схема не является очень распространенной, чаще транзакты зарождаются непрерывно и используют какой-то именованный сервер однократно.

2. Для четкой привязки возвращающихся к началу модели Хакт у ОБ ADVANCE предусмотрен ярлык.

3. Постоянство числа Хакт, обращающихся в модели, обеспечивается введением операнда D. При необходимости нахождения рационального числа Хакт следует провести ИМ с разными значениями операнда D, ниже рассмотрим возможности автоматизации подобной операции.

4. В МФ применены два фрагмента, второй из которых — временной таймер — управляет процессом окончания процесса ИМ по времени.

5. Структура МФ содержит некоторые излишества в виде названий отдельных фрагментов, так как можно обойтись и без них, но их наличие упрощает чтение МФ, в том числе и другими пользователями, не имеющими прямого отношения к МФ. Поэтому такие не обязательные элементы зависят от квалификации и приверженности создателей МФ.

6. В итоговом отчете значения абсолютного и относительного времен совпадают по причине неизменности начальных данных, абсолютное вре-

мя возрастает по сравнению с относительным, когда меняются входные данные и используется ОУ CLEAR.

Пример 6.1а. Модифицированная модель нотариальной конторы

Рассмотрим фрагментарно учет неодинаковости старта каждого клерка в отдельности для примера 6.1.

1. Постановка задачи.

При сохранении данных примера 6.1 введем измененные стартовые условия для каждого из 4-х клерков:

— клерк 1 начинает работу над документом;

— клерк 2 освободится у менеджера через 3 мин;

— клерк 3 закончит подготовку документа через 10 мин;

— клерк 4 ожидает приема у менеджера.

4. Модельный файл.

Чтобы учесть измененные условия, необходимо ввести 4 отдельных ОБ GENERATE, имеющих свои ярлыки CLERK1, CLERK2 и т. д. Ниже приводится МФ, построенный для рассматриваемой ситуации.

```
            SIMULATE
*
CLERK1      GENERATE    0,,,1       начало работы в МВ 0.0
REPEAT      ADVANCE     30,5        подготовка документа
GETMAN      SEIZE       MANAGER     обращение к менеджеру
            ADVANCE     8,2         проведение регистрации
FREEMAN     RELEASE     MANAGER     уход от менеджера
            TRANSFER    ,REPEAT     возврат к новой операции
*
CLERK2      GENERATE    0,,,1       начало работы в МВ 0.0
            SEIZE       MANAGER     переход без задержки
                                    к менеджеру
            ADVANCE     3           оставшееся время регист-
                                    рации
            TRANSFER    ,FREEMAN    возврат к новой операции
*
CLERK3      GENERATE    0,,,1       начало работы в МВ 0.0
            ADVANCE     10          оставшееся время
                                    на подготовку документа
            TRANSFER    ,GETMAN     возврат к новой операции
*
CLERK4      GENERATE    0,,,1       начало работы в МВ 0.0
            TRANSFER    ,GETMAN     возврат к новой операции
*
            GENERATE    2400        моделирование в течение 40 ч
            TERMINATE   1           сигнал на СС,
*                                   прекращение движения транз-
                                    актов
```

```
START      1          установка СС
END                   окончание процесса ИМ
```

Конец фрагмента

6.2.2. Сбор дополнительной информации (использование очередей)

В результате процесса ИМ накапливается информация об использовании устройств, но часто возникает необходимость получить не средние данные по окончании процесса ИМ, а собрать информацию об использовании какой-то части ОБ МФ. Если рассматривать интересующие исследователя точки А и В МФ, то встают вопросы типа:

— сколько Хакт пройдет между точками А и В;

— каково среднее число транзактов;

— каково среднее время прохождения Хакт;

— какое количество Хакт имеют нулевое время прохождения?

На эти вопросы позволяет ответить использование необязательных парных ОБ QUEUE/DEPART (см. п. 5.1.1 и прил. 1), а описание разделов итогового отчета дано в п. 2 прил. 6. Для сбора информации о всей траектории движения транзактов ОБ QUEUE ставится сразу после ОБ GENERATE, а выходной ОБ DEPART — непосредственно перед ОБ TERMINATE. В принципе, в МФ может быть представлено несколько очередей, вложенных друг в друга, но при этом должно соблюдаться обязательное условие выхода из внутренней (вложенной) очереди раньше, чем будет осуществлен выход из главной очереди. Использование этих парных ОБ (в случае их правильного применения) никак не влияет на логику и результаты моделирования.

Задание. Читателю предлагается самому промоделировать в тестовом режиме приводимый ниже МФ для моделирования процесса функционирования придорожного ресторана быстрого питания с двумя очередями, организованными для сбора информации. Моделирование предлагается провести в двух вариантах: а) без ОБ, создающих очереди, и б) с ОБ, создающими очередь, — а затем сравнить полученные результаты. Логика функционирования ресторана сводится к следующему.

Автомашины (транзакты) прибывают к ресторану с интервалом 10±±10 мин, для заказа на пункте двусторонней связи (сервер) тратится 3±1 мин, затем за 15±3 с автомашина переезжает к окну выдачи заказа (сервер) и за 3±2 мин получает упакованный заказ, после чего автомашина уезжает. Собрать статистику по загрузке ресторана в целом и окна выдачи при обслуживании 50 автомашин. Проделайте то же самое за сутки работы ресторана (см. § 5.3).

```
            SIMULATE
*
            GENERATE    10,10
```

```
QUEUE SNAKTIME
SEIZE          INTERCOM
ADVANCE        3,1
RELEASE        INTERCOM
ADVANCE        0.25,0.05
QUEUE WINDOW
SEIZE          WINDOW
DEPART         WINDOW
ADVANCE        3,2
RELEASE        WINDOW
DEPART         SNAKTIME
TERMINATE      1
*
START          50
END
```

Необходимые комментарии. Обратите внимание на то, что имена у второй очереди и устройства совпадают (WINDOW), при этом в программе не возникает разночтений потому, что они определены однозначно, но если бы речь шла об имени какого-либо ОБ, то оно не могло бы совпадать ни с каким именем другого объекта. Заметим также, что расположение ОБ очереди точно соответствует тому месту, с которого необходимо снять информацию. Не вызывает никакого сомнения и тот факт, что в процессе ИМ каждый Хакт одновременно является членом двух разных очередей. Не следует забывать, что физического нахождения в очереди на самом деле нет, просто снимается одна и та же информация для использования ее в разных контекстах, поэтому Хакт одновременно может быть членом любого количества разных очередей. И последнее, что необходимо заметить: ОБ очереди WINDOW имеют между собой всего один ОБ SEIZE WINDOW, т. е. как только очередь организуется, при исполнении SEIZE Хакт немедленно выходит из очереди, так как информация об использовании окна выдачи уже получена.

Пример 6.2. Модель обслуживания в инструментальной кладовой

1. Постановка задачи.

В цеху существует инструментальная кладовая, в которой работает один кладовщик, сами инструменты уникальны и дороги, для того чтобы они были у каждого механика по ремонту оборудования. Механики разделяются на специалистов по электронике и механике. Время их прихода в кладовую и обслуживания представлено в табл. 6.2.

Таблица 6.2

Тип механиков	Время прихода, с	Время обслуживания, с
1	420±360	300±90
2	360±240	100±30

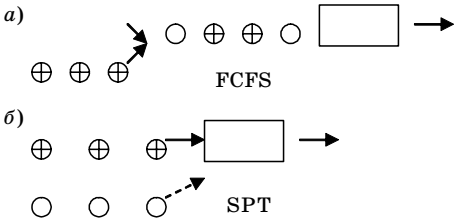


Рис. 6.3. *Эпюра разных дисциплин обслуживания: а — общая очередь; б — очередь с приоритетом: ⊕ — приоритет*

Кладовщик работает по ДО FCFS, можно предположить, что ДО SPT может оказаться более предпочтительной по среднему количеству механиков, ожидающих обслуживания. Это позволит сократить расходы за счет увеличения эффективности работы механиков и сокращения времени простоя оборудования. Эпюра двух разных дисциплин обслуживания представлена на рис. 6.3.

Для сравнения двух дисциплин обслуживания необходимо ввести значение приоритета, задаваемое операндом E ОБ GENERATE. При этом МФ будут идентичными, за тем исключением, что в случае ДО FCFS операнд E отсутствует, а в случае ДО SPT операнд E появляется. Процесс ИМ проводится за 8 ч и оканчивается точно в это время, даже если есть запросы на обслуживание.

2. Допущения, сделанные в модели.

Будем полагать, что в начальный момент кладовщик свободен и ни один механик не обращался с запросом. Для создания двух потоков заявок используются два разных ОБ GENERATE. При ДО FCFS приоритет механиков обоих типов одинаков, при ДО SPT приоритет механиков второго типа (чаще приходят, быстрее обслуживаются) выше и равен 10, а у механиков первого типа равен 5. В МФ используются очереди для определения среднего числа механиков, ожидающих обслуживания.

3. Таблица определений (табл. 6.3).

Временная дискрета: 1 с

Таблица 6.3

Объект GPSS/H	Объект в системе
Транзакты Фрагмент 1 Фрагмент 2 Фрагмент 3	 Механики 1-го типа Механики 2-го типа Хакт управления
Устройства CLERK	 Кладовщик
Очереди TOOLWAIT	Информация по обоим типам механиков

4. Модельный файл.

```
*     Модуль описания
      SIMULATE                    Пример 6.2.
*                                 Инструментальная кладовая ДО SPT
*                                 Временная дискрета: 1 с
*     Модуль исполнения
*                  Фрагмент 1     1-й тип механиков          *
      GENERATE     420,360,,,5    приход механиков 1-го типа
*                  (Приоритет 5)
      QUEUE        TOOLWAIT       постановка в очередь
      SEIZE        CLERK          запрос на обслуживание
      DEPART       TOOLWAIT       выход из очереди
      ADVANCE      300,90         обслуживание
      RELEASE      CLERK          освобождение кладовщика
      TERMINATE    0              уход из кладовой
*                  Фрагмент 2     2-й тип механиков
      GENERATE     360,240,,,10   приход механиков 2-го типа
                   (Приоритет 10)
      QUEUE        TOOLWAIT       постановка в очередь
    SEIZE          CLERK          запрос на обслуживание
    DEPART         TOOLWAIT       выход из очереди
    ADVANCE        100,30         обслуживание
    RELEASE        CLERK          освобождение кладовщика
    TERMINATE      0              уход из кладовой
*                  Фрагмент 3     Хакт управления *
    GENERATE       28800          время моделирования 8 ч
    TERMINATE      1              уменьшение СС на 1,
*                                 окончание движения транзактов
*   Модуль управления
    START          1              установка СС=1
    END                           окончание процесса ИМ
```

5. Итоговый отчет.

Отчет имеет стандартный вид (см. прил. 4). Конкретные цифры, относящиеся к рассматриваемому примеру для различных дисциплин обслуживания, сведены в табл. 6.4.

Таблица 6.4

ДО	SPT	FCFS
Загрузка устройства	0.903	0.894
Число входов	142	141
Максимум в очереди	6	8
Среднее очереди	1.51	1.705
Нулевые входы	26	24
Время ожидания	296.093	375.867

6. Выводы и обсуждение.

1. Для сравнения двух ДО в примере использованы различные значения приоритета, для чего потребовалось два отдельных прогона. (Способы совмещения нескольких прогонов в одном цикле ИМ и автоматизации этих прогонов см. в § 6.4.)

2. Применение ДО SPT уменьшает время ожидания обслуживания на 12 %. (Более подробное сравнение двух ДО будет сделано в гл. 8.)

§ 6.3. МНОГОКАНАЛЬНОЕ ОБСЛУЖИВАНИЕ

Довольно часто в моделируемой системе может одновременно работать несколько идентичных серверов (несколько однотипных причалов в порту, несколько однотипных станков и т. п.). Несколько сложнее решается вопрос при рассмотрение в качестве серверов людей. Так, условие идентичности операторов должно специально оговариваться условиями задачи, поскольку операторы могут различаться по квалификации, темпераменту, психологическому состоянию и т. д.

Если в системе действуют несколько операторов и их отличие подчеркнуто условиями задачи, то они должны моделироваться отдельными устройствами (см. § 6.2), а не памятями, как в случае идентичных серверов. В GPSS/H группа идентичных серверов моделируется памятями (Storage).

Это название представляется не совсем удачным, так как оно ассоциируется с физическим пространством для размещения чего-либо (гараж, сигаретная пачка и т. п.), но будем нормы ЯИМ принимать как данность. Итак, группа серверов с неразличимыми отличиями моделируется объектом GPSS/H, называемым память.

Операции с памятью реализуются парными ОБ ENTER / LEAVE (см. п. 5.1.1 и прил.1).

ОБ ENTER обладает способностью запрещать вход Хакт. Если все единицы памяти заняты, Хакт остается в предыдущем ОБ и ждет освобождения необходимых единиц памяти. Следует отметить полную аналогию в работе парных ОБ SEIZE/RELEASE и ENTER/LEAVE. Имя у ОБ ENTER/LEAVE идентифицирует название памяти, которое задается в модуле описания МФ (см. п. 5.1.2 и прил. 1). В отличие от устройств, Хакт может сразу занять несколько единиц памяти. В операнде А ОУ STORAGE задается емкость памяти. Например, памяти представляют собой причалы разной длины, а транзакты — грузовые суда также разной длины. Если свободен короткий причал, а судно длинное, то оно не сможет использовать этот причал, а будет ожидать освобождения более длинного причала. Специфика использования памятей такова, что можно создавать

единую очередь при ожидании обслуживания группой идентичных серверов (транзакт движется из очереди к первому освободившемуся серверу) либо создавать отдельные очереди к каждому серверу из группы идентичных серверов. Эпюры возможного обслуживания для названных случаев приведены на рис. 6.4, *а*, *б*.

Обслуживание с одной очередью предусматривает по умолчанию использование ДО FCFS при условии одинакового приоритета. При обслуживании с разными очередями транзакт может не только выбрать сервер, но и переходить к другому серверу, если время обслуживания первым сервером окажется неоправданно большим. В GPSS/H моделирование с разными очередями реализуется устройствами. Так, в варианте рис. 6.4, *б* каждая очередь обслуживается своим устройством, имеющим отличное от других имя, что позволяет четко реализовать привязку Хакт к устройству.

Такое, ориентированное на устройства, моделирование проводится даже для идентичных серверов при наличии разных очередей! Напомним, что ОБ ENTER имеет два операнда, первый из них — A — идентифицирует имя памяти, а второй — B — оговаривает число единиц емкости, занимаемых у этой памяти. При этом текущее содержание занимаемой памяти увеличивается на число единиц, оговоренных операндом B, а остающаяся емкость уменьшается на то же число единиц. При исполнении ОБ LEAVE происходят обратные процессы, т. е. остающаяся емкость увеличивается на число освобожденных единиц, а текущее содержание уменьшается.

Необходимо отметить следующую особенность GPSS/H. Если для обслуживания какого-либо транзакта необходимо, например, 5 единиц емкости, а имеется только 2, то они не резервируются до тех пор, пока не освободятся еще 3. В ЯИМ действует правило: *все или ничего*! Это правило служит своеобразным приоритетом при обслуживании приходящих транзактов. Например: для Хакт, пришедшего поз-

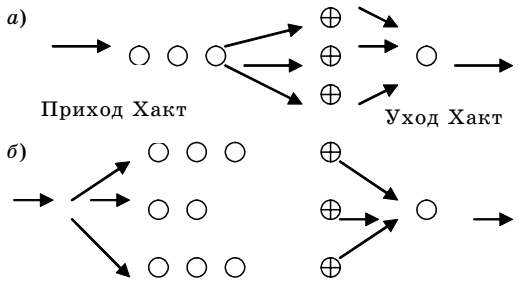


Рис. 6.4. *Эпюра многоканального обслуживания: а — с одной очередью; б — с разными очередями*

же, требуется 2 единицы емкости памяти; при наличии 2-х единиц такой Хакт обслуживается сразу, а первый будет по-прежнему блокирован. В отдельных случаях такой порядок может повлиять на логику работы реальной системы, поэтому существует способ накопления единиц емкости путем последовательного размещения ОБ ENTER со значением операнда B, взятого по умолчанию (емкость по умолчанию равна 1).

Для того чтобы определить состояние памяти, в ЯИМ предусмотрены ОБ SAVAIL (память доступна) и SUNAVAIL (память не доступна); если эти ОБ отсутствуют в МФ, то память считается доступной всегда (в пособии использование этих ОБ не рассматривается).

Использование памятей позволяет определить ограничения по количеству ресурсов, необходимых для нормального функционирования исследуемой системы. Так, минимальное число серверов позволяет определить уровень удовлетворения запросов пользователя системы, а максимальное — определить ограничения на используемые ресурсы. В GPSS/H память является практически неограниченным ресурсом и ее емкость может достигать 2 млн. После окончания процесса ИМ получается среднее значение потребной емкости и максимальное содержание; следующее целое число после среднего значения может представлять собой минимальное число потребных серверов. Максимальное число единиц емкости характеризует ограничение по числу серверов. Таким образом, полученный интервал значений емкости может служить основанием для выбора рационального числа памятей.

6.3.1. Многоканальное обслуживание групп идентичных серверов

Пример 6.3. Модель обслуживание судов в порту

1. Постановка задачи.

В порт для погрузо-разгрузочных работ заходят грузовые суда типов A и B. В порту есть три причала только для судов типа A и два причала только для судов типа B. Для ввода и вывода судов в гавань порта используются три буксира, которые после швартовки конкретного судна в процессе погрузки-разгрузки больше не используются. Судам типа A для провода в порт требуется 3 буксира и для вывода из порта 2 буксира, судам типа B — 2 и 1 буксир соответственно. При подходе судна к внешнему рейду порта происходят следующие события:

— судно запрашивает необходимый ему причал;

— после подтверждения наличия причала следует запрос о потребном числе буксиров;

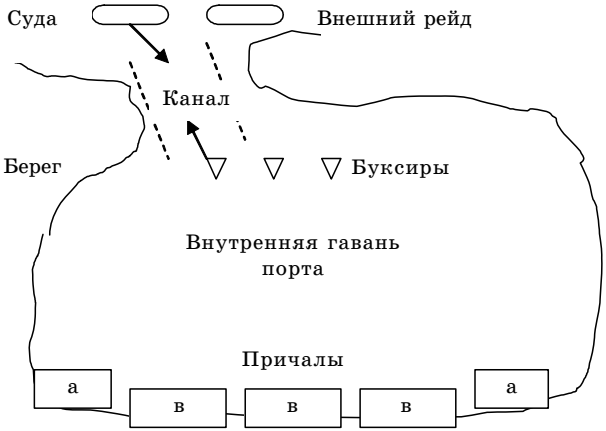


Рис. 6.5. *Примерная схема расположения объектов*

— после подхода буксиров начинается этап входа в порт и швартовки к причалу;

— после швартовки следует освобождение буксиров;

— начинаются погрузо-разгрузочные работы;

— следует запрос на буксиры, потребные для выхода из гавани;

— происходит процесс отшвартовки;

— после вывода на внешний рейд буксиры освобождаются.

Суда получают буксиры по принципу: все или ничего.

Схема расположения показана на рис. 6.5

Данные о временах действий судов обоего типа сведены в табл. 6.5.

Таблица 6.5

Тип судна	Время прихода, ч	Время разгрузки / погрузки, ч	Время швартовки/ отшвартовки, мин
А	3.2±1.5	8.4±1.5	30/15
Б	1.5±0.75	2.1±0.6	30/15

Постройте МФ для указанной задачи, принимая за временную дискрету 1 ч. Соберите дополнительную информацию о пребывания судна в порту и о вводе судна во внутреннюю гавань порта по морскому каналу от внешнего рейда (учитывая наличие двух типов судов, необходимо создать 4 очереди). Процесс моделирования протекает до момента оставления порта сотым судном.

2. Допущения, сделанные в модели.

Причалы разных типов представляют собой группы идентичных серверов, а следовательно, моделируются как памяти. В модуле описания дол-

жно появиться описание этих памятей. Очередь, описывающая весь процесс моделирования, начинается сразу после первого ОБ (ОБ GENERATE) и кончается перед последним ОБ (ОБ TERMINATE). Первый ОБ QUEUE не препятствует входу Хакт, поэтому отпадает необходимость ставить фиктивный ОБ ADVANCE (см. § 6.2).

3. Таблица определений (табл. 6.6).

Временная дискрета: 1 ч

Таблица 6.6

Объект GPSS/H	Объект в системе
Транзакты	
Фрагмент 1	Судно типа A
Фрагмент 2	Судно типа B
Очереди	
ABERTH	Общая очередь судов типа A
AINPORT	Очередь на ввод в гавань судов типа A
BBERTH	Общая очередь судов типа B
BINPORT	Очередь на ввод в гавань судов типа B
Памяти	
ABERTH	Причалы для судов типа A
BBERTH	Причалы для судов типа B
TUGBOATS	Буксиры для обоих типов судов

4. Модельный файл.

```
*               Модуль описания
                SIMULATE                    Пример 6.3. Обслуживание
*                                            грузовых судов в порту
*                                           Временная дискрета: 1 ч
ABERTHS         STORAGE     3               число причалов для судов типа A
BBERTHS         STORAGE     2               число причалов для судов типа B
TUGBOATS        STORAGE     3               число буксиров
*               Модуль исполнения
*               Фрагмент 1                  суда типа A *
                GENERATE    3.2,1.5         приход судов типа A, одно за
                                             другим
                QUEUE       AINPORT         создание общей очереди судов
                                             типа A
                QUEUE       ABERTH          создание очереди на ввод
                                             в гавань
                ENTER       ABERTHS         запрос на причал для судов
                                             типа A
                DEPART      ABERTH          уход из очереди на ввод в гавань
                ENTER       TUGBOATS,3      запрос на 3 буксира
                ADVANCE     0.5             время швартовки
```

```
                LEAVE     TUGBOATS,3   освобождение 3-х буксиров
                ADVANCE   8.4,1.5      докерские работы
                ENTER     TUGBOATS,2   запрос на 2 буксира
                ADVANCE   0.25         время отшвартовки
                LEAVE     TUGBOATS,2   освобождение 2-х буксиров
                LEAVE     ABERTHS      освобождение причала
                DEPART    AINPORT      уход из общей очереди
                TERMINATE 1            уменьшение показаний СС
                                       на 1
*                         Фрагмент 2   суда типа B *
                GENERATE  1.5,0.75     приход судов типа B,
                                       одно за другим
                QUEUE     BINPORT      создание общей очереди
                QUEUE     BBERTH       создание очереди на ввод
                                       в гавань
                ENTER     BBERTHS      запрос на причал
                DEPART    BBERTH       выход из очереди на ввод
                                       в гавань
                ENTER     TUGBOATS,2   запрос на 2 буксира
                ADVANCE   0.5          время швартовки
                LEAVE     TUGBOATS,2   освобождение буксиров
                ADVANCE   2.1,0.6      докерские работы
                ENTER     TUGBOATS     запрос одного буксира
                ADVANCE   0.25         время отшвартовки
                LEAVE     TUGBOATS     освобождение буксира
                LEAVE     BBERTHS      освобождение причала
                DEPART    BINPORT      уход из общей очереди
                TERMINATE 1            уменьшение показаний СС
                                       на 1
*               Модуль управления
                START     100          установка СС на число 100,
                                       начало ИМ
                END                    окончание процесса ИМ
```

5. Итоговый отчет.

Отчет имеет стандартную форму (см. прил. 4). Основные результаты ИМ сведены в табл. 6.7.

Таблица 6.7

Объект	Данные		
Время ИМ	Абсолютное и относительное времена одинаковы и равны 112.038		
Типы судов Число входов	A = 31	B = 69	
Памяти	ABERTHS	BBERTHS	TUGBOATS
Коэффициент	0.927	0.894	0.457

Окончание табл. 6.7

Объект	Данные			
Число входов	34	70	373	
Среднее время	9.166	2.862	0.412	
Среднее число	2.782	1.788	1.37	
Текущее число	3	1	2	
Очереди	AINPORT	ABERTH	BINPORT	BBERTH
Максимум	6	3	4	2
Среднее	3.449	0.667	2.092	0.304
Число входов	36	36	71	71
Нульвходы	0	12	0	29
Среднее время	10.732	2.075	3.302	0.48

6. Выводы и обсуждение.

1. Суда типа A не конкурируют с судами типа B за использование причалов, поскольку это оговорено в постановке задачи, но использование буксиров является предметом конкурентного запроса ввиду их идентичности для обоих типов судов. В связи с этим существует корреляция между двумя фрагментами модуля исполнения МФ.

2. Описание памятей дается до начала исполнения МФ, что диктуется логикой симуляции сверху вниз.

3. Среднее число судов типа A на 70% превышает число судов типа B.

4. Среднее время нахождения судов типа A в гавани порта равно 10.732 часа. Если бы эти суда не пребывали в состоянии ожидания потребного числа буксиров, то ожидаемое время могло бы равняться 9.15 (0.5 — ввод в гавань, 0.25 вывод из гавани, 8.4 — среднее время погрузо-разгрузочных работ). Следовательно, если бы число буксиров не было бы ограничено тремя, то можно было бы экономить 1.5 на пребывания судна в порту. Отсюда следует, что результаты моделирования могут служить основой для экономического расчета эффективности работы порта (см. гл. 8).

6.3.2. Перекрытие памятей и введение Хакт только при необходимости

Рассмотрим некоторые особенности моделирования многоканальных систем, включающих группы идентичных серверов.

Возможность перекрытия памятей. Памяти, так же как устройства, могут представляться с перекрытием, т. е. памяти как бы вкладываются друг в друга. В этом случае ОБ ENTER первой по очереди памяти предваряет ОБ ENTER вложенной памяти, а ОБ LEAVE вложенной памяти, наоборот, ставится в МФ раньше ОБ LEAVE основной.

Введение транзактов по внешнему условию. Иногда возникает необходимость введения в модель нового Хакт только в тот момент

времени, когда он необходим, но такую необходимость трудно предусмотреть при построении модели, особенно если условия ввода меняются по вероятностному закону.

Изучим эти особенности на примере цеха, в котором заготовки поступают в цех только тогда, когда в них возникает необходимость. Будем считать, что запас заготовок не ограничен и новое изделие из поступающей заготовки начинает производиться только тогда, когда готовое изделие покидает цех. Все это происходит в момент времени, который нельзя оценить заранее. Такую ситуацию можно моделировать с помощью «голого» ОБ GENERATE (не имеющего операндов, naked — голый, лишенный перьев) и Хакт, стоящего перед ОБ и способного постоянно осуществлять запрет на вход в него (в нашем случае ОБ ENTER). Такой генератор должен производить Хакт в модельное время 0.0, но этого не происходит из-за наличия ОБ, препятствующего входу. Ввода транзактов не происходит до наступления момента, когда Хакт (заготовка) потребуется. В цеху имеется 10 рабочих мест и работают 5 рабочих типа A и 5 рабочих типа B. Работа распадается на три этапа: 1) со временем 35±15 работает рабочий типа A; 2) со временем 50±10 второй этап работы совершает рабочий типа B; 3) со временем 15±5 работу завершает рабочий типа A. В цеху непрерывно находится 10 изделий разной степени готовности, как только одно готовое изделие покидает цех, на смену поступает заготовка. Представим МФ системы.

```
*  Модуль описания
                          SIMULATE
   SYSTEM      STORAGE    10          10 изделий в процессе
                                      изготовления
   AWORKERS    STORAGE    5           5 рабочих типа A
   BWORKERS    STORAGE    5           5 рабочих типа B
*  Модуль исполнения
               GENERATE   0           заготовки
               ENTER      SYSTEM      ввод заготовки
                                      при необходимости
               ENTER      AWORKERS    занятие рабочего типа A
               ADVANCE    35,15       1-й этап изготовления
               LEAVE      AWORKERS    освобождение рабочего
                                      типа A
               ENTER      BWORKERS    занятие рабочего типа B
               ADVANCE    50,10       2-й этап изготовления
               LEAVE      BWORKERS    освобождение рабочего
                                      типа B
               ENTER      AWORKERS    занятие рабочего типа A
               ADVANCE    15,5        3-й этап изготовления
               LEAVE      AWORKERS    освобождение рабочего
                                      типа A
               LEAVE      SYSTEM      сигнал на ввод заготовки
               TERMINATE  1           уменьшение значения СС
```

```
* Модуль управления
        START       100         создание 100 изделий
        END                     окончание процесса ИМ
```

Отметим, что в данном случае работает принцип: когда один уходит, другой приходит на смену. Этот принцип реализуется путем использования «голого» ОБ GENERATE. При терминировании происходит два события (см. п. 5.4.3): во-первых, переключается флаг изменения статуса модели и, во-вторых, просыпается ранее блокированный в генераторе транзакт и вводится в модель.

Задание. Промоделируйте этот пример в тестовом режиме и отметьте, что происходит в статусном окне при терминировании Хакт.

6.3.3. Получение ряда реплик в одном пакетном режиме

Часто возникает проблема получения ряда реплик в одном пакетном режиме, ниже рассмотрим наиболее простой способ получения независимых реплик в течение одного прогона модели с использованием ОБ CLEAR (см. п. 5.1.2 и прил. 1).

В примере 6.1 использовался ОБ TRANSFER в его наиболее простой форме безусловного перехода, гораздо интереснее случай статистической формы этого ОБ, когда вероятность перехода на другую траекторию может быть весьма малой (см. п. 5.1.1). Этот ОБ никогда не препятствует входу Хакт; если последующий ОБ препятствует входу Хакт, то он остается в ОБ TRANSFER, как в своем текущем блоке. Рассмотрим эту ситуацию в примере 6.4.

Пример 6.4. Модель контроля качества

1. Постановка задачи.

Собранные телевизоры проходят через группу контрольных постов выходного контроля. Если телевизор не проходит выходной контроль, то он отправляется на участок подрегулировки и после успешного завершения этой операции возвращается в очередь телевизоров, ждущих выходного контроля. Прошедшие выходной контроль телевизоры поступают на склад готовой продукции. Эта схема представлена на рис. 6.6. Каждые 5.5±2 мин телевизоры поступают на выходной контроль, на котором находится два контрольных поста, время проверки 10±3 мин. С вероятностью 12 % телевизоры не проходят выходной контроль и отправляются на подрегулировку. Время подрегулировки 30±10 мин. Как перед постами выходного контроля, так и перед регулировщиком могут скапливаться телевизоры. Необходимо определить число потребных стеллажей для хранения поступающих телевизоров. Процесс моделирования прекращается после проверки 100 телевизоров. В качестве дополнительного условия рассмотрим возможность получения результатов нескольких последовательных реплик в

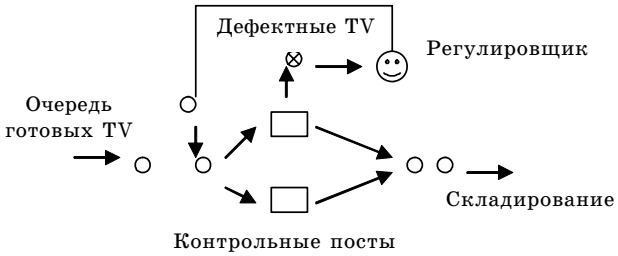


Рис. 6.6. *Эпюра движения транзактов*

одном цикле ИМ, что достигается размещением в модуле управления ОУ CLEAR.

2. Допущения, сделанные в модели.

После прохождения выходного контроля телевизоры разделяются на два потока: 88 % годных отправляются на склад готовой продукции, 12 % дефектных — на подрегулировку. Для исследования вопроса о числе стеллажей создаются две очереди: одна — перед постами выходного контроля, вторая — на линии регулировки.

Для возвращения отрегулированных телевизоров в общий поток проверяемых используется ОБ TRANSFER безусловного вида. Для получения динамики производственного процесса совместим 5 последовательных прогонов (реплик) в одном цикле ИМ путем последовательного применения пар ОУ START—CLEAR.

3. Таблица определений (табл. 6.8).

Временная дискрета: 1 мин.

Таблица 6.8

Объект GPSS/H	Объект в системе
Транзакты Фрагмент 1 Фрагмент 2	 Готовые телевизоры Телевизоры после подрегулировки
Устройства ADJUSTER	 Регулировщик
Очереди ADJUSTO LASTTEST	 Очередь на подрегулировку Общая очередь на контроль
Памяти TESTERS	 Два контролера

4. Модельный файл.

```
* Модуль описания                    Пример 6.4. Модель контроля качества
SIMULATE                             Временная дискрета: 1 мин
*
```

```
TESTERS   STORAGE     2              контролеры выходного
                                     контроля
* Модуль исполнения
*         Фрагмент 1                 общий выходной контроль
          GENERATE    5.5,2          поступление готовых
                                     телевизоров, один за другим
RETEST    QUEUE       LASTTEST       организация очереди
                                     контроля
          ENTER       TESTERS        запрос контролера
          DEPART      LASTTEST       выход из очереди контроля
          ADVANCE     10,3           время контроля
          LEAVE       TESTERS        освобождение контролера
          TRANSFER    .120,,ADJUSTIT 12% на подрегулировку
          TERMINATE   1              оставшиеся 88 % на упаковку
*         Фрагмент 2                 осуществление
                                     подрегулировки
ADJUSTIT  QUEUE       ADJUSTQ        организация очереди
                                     подрегулировки
          SEIZE       ADJUSTER       запрос регулировщика
          DEPART      ADJUSTQ        выход из очереди
                                     подрегулировки
          ADVANCE     30,10          время подрегулировки
          RELEASE     ADJUSTER       освобождение
                                     регулировщика
          TRANSFER    ,RETEST        возврат на общий контроль
* Модуль управления
          START       100            CC=100, проведение
                                     1-й реплики
          CLEAR                      удаление информации для
                                     2-й реплики
          START       100            CC=100, проведение
                                     2-й реплики
          CLEAR                      удаление информации для
                                     3-й реплики
          START       100            CC=100, проведение
                                     3-й реплики
          CLEAR                      удаление информации для
                                     4-й реплики
          START       100            CC=100, проведение
                                     4-й реплики
          CLEAR                      удаление информации для
                                     5-й реплики
          START       100            CC=100, проведение
                                     5-й реплики
          END                        окончание процесса ИМ
```

5. Итоговый отчет.

Отчет имеет стандартный вид (см. прил. 4), но в связи с тем, что в рамках одного процесса ИМ проводилось 5 реплик, для каждой из реплик дается свой вариант отчета в рамках одного листинга. Основные результаты сведены в табл. 6.9. Для точного сбора информации проведите ИМ в пакетном режиме и оцените полученные результаты по файлу с вашим именем, имеющим расширение .lis.

Таблица 6.9

Объект	Основные данные реплики				
	1-й	2-й	3-й	4-й	5-й
Время	586.13	607.43	588.46	549.62	590.64
Устройство					
Использование	0.706	0.659	0.738	0.206	0.627
Число входов	14	13	15	5	13
Среднее время	29.56	30.79	28.92	28.22	28.23
Памяти					
Использование	0.978	0.965	0.978	0.952	0.965
Число входов	115	117	116	106	114
Среднее число	9.95	10.02	9.96	9.87	10.01
Очереди					
LASTTEST					
Максимум	6	6	6	3	4
Среднее	2.24	1.335	2.676	0.644	1.388
Число входов	120	123	120	107	117
Нуль-входы	10	14	8	24	13
ADJUSTQ					
Максимум	3	2	3	1	2
Среднее	0.603	1.28	0.587	0.033	0.421
Число входов	14	16	15	5	13
Нуль-входы	5	5	5	4	4

6. Выводы и обсуждение.

1. Для проведения нескольких реплик в пакетном режиме необходимо внести изменения только в модуль управления. Комбинация ОУ CLEAR и START позволяет проводить столько независимых реплик, сколько раз использована эта комбинация. При этом обнуляются все данные предыдущей реплики, за исключением положения ГСЧ и ИН транзактов. Если обратиться к табл. 6.9, то имеющийся разброс данных как раз и говорит о независимости реплик.

2. Если при использовании ОБ TRANSFER в статистической форме пересылка по пути с большей вероятностью осуществляется к следующему последовательному ОБ (ОБ TERMINATE), то операнд B у ОУ может отсутствовать, а ОБ, к которому идет пересылка, при этом не имеет ярлыка, что видно из МФ.

3. Показанная разбивка МФ на фрагменты не является обязательной, зависит только от предпочтений пользователя и может иметь любой другой вид при условии сохранения правил записи МФ. Более того, сам порядок расположения фрагментов также не является обязательным. Так, фрагмент 2 может предварять фрагмент 1 (проверьте это на практике, изменив порядок следования фрагментов).

4. Отвечая на вопрос примера относительно мест на стеллажах, очевидно, что при максимальном числе членов очереди для общей очереди необходимо 6 мест, а для подрегулировки достаточно 3-х. Однако если оценивать среднее содержание, то необходимо гораздо меньше мест. Поэтому этот вопрос должен решаться с учетом условий производства и является компетенцией менеджера, а результаты ИМ служат основанием для принятия решения.

6.3.4. Изменение приоритета транзакта в процессе ИМ

В заключение этого параграфа интересно рассмотреть пример изменения приоритета Хакт непосредственно в процессе ИМ, когда приоритет изменяется скачком с помощью ОБ PRIORITY, что позволяет отслеживать изменяющиеся условия функционирования. Порядок, в котором транзакты будут предпринимать попытку к движению в следующий ОБ, зависит от их приоритета. При зарождении транзакта приоритет задается операндом Е ОБ GENERATE, однако существуют ситуации, когда первоначально заданный приоритет должен измениться, причем изменение приоритета может происходить несколько раз в течение процесса ИМ. Эту операцию позволяет осуществить ОБ PRIORITY (см. п. 5.1.1 и прил. 1). Этот ОБ никогда не препятствует входу Хакт. Когда происходит исполнение ОБ PRIORITY, то Хакт приобретает то значение приоритета, которое содержится в операнде А ОБ PRIORITY. При этом транзакт остается в СТС и флаг изменения состояния переключается на состояние «включено». Переключение ФИС происходит в силу того, что изменение приоритета приводит к перераспределению времен движения транзактов в СТС, и необходимо проведение нового подэтапа сканирования (см. § 5.4) для определения претендента на продвижение по МФ. Предположим, что в какой-то системе имеется два последовательных сервера и используется ДО SPT, т. е. движение транзактов зависит от кратчайшего времени обслуживания. При этом транзакты первого вида быстрее обслуживаются на втором сервере, естественно, что они должны сменить значение приоритета перед вторым сервером на большее. Причем расположение ОБ PRIORITY четко зави-

сит от решаемой задачи, т. е. этот ОБ должен располагаться после ОБ первого сервера, но до начала исполнения ОБ второго сервера. Так, если ОБ PRIORITY ставится перед ОБ ADVANCE второго сервера, то это позволяет Хакт перейти в СБС с новым значением приоритета, а затем вернуться в СТС на новой позиции. Если же поставить ОБ PRIORITY после ОБ ADVANCE, произойдет ошибка, так как Хакт уйдет в СБС со старым значением приоритета. В связи с этим необходимо иметь в виду, что ОБ ADVANCE является в модели своеобразной линией раздела, поэтому при изменении приоритета ОБ PRIORITY всегда должен находиться выше линии раздела!

Пример 6.5. Модель использования оборудования

1. Постановка задачи.

После предварительной механической обработки деталь со склада поступает на финишную доводку. Финишные работы состоят из двух последовательных процессов доводки: процесса 1 и процесса 2, которые выполняются рабочим на его доводочном станке. Таких станков в цеху несколько и они закреплены за конкретным рабочим. Поскольку детали тяжелые, то для транспортировки со склада, изменения положения на доводочном станке и отправки на склад готовой продукции требуется тельфер. Хронологический порядок действий таков:

1) произвести процесс 1;
2) переменить положение детали на доводочном станке;
3) произвести процесс 2;
4) произвести подготовительные операции:
— снять готовую деталь со станка;
— отправить готовую деталь на склад готовой продукции;
— забрать заготовку со склада;
— установить заготовку на станке;
— начать с п. 1.

Очевидно, что тельфер нужен на этапах 2 и 4. Перед менеджером встает вопрос, каково оптимальное соотношение доводочных станков и тельферов и когда надо приобретать дополнительные тельферы? В приводимом примере рассмотрено соотношение 3 тельфера/15 доводочных станков.

(В качестве самостоятельного задания рассмотрите: число станков от 3-х до 6 при одном тельфере и 10 станков при двух тельферах.)

Построить модель работы доводочного участка за рабочую неделю 40 ч.

Тельфер используется по ДО SPT, время использования тельфера на этапе 2 15±5 мин, на этапе 4 — 30±5 мин, отсюда следует, что операции на этапе 2 (в связи с меньшим временем) обладают большим приоритетом, чем операции на этапе 4. Время, затрачиваемое на первую операцию, составляет 60±10 мин, на вторую — 90±20 мин.

2. Допущения, сделанные в модели.

В модели приняты ограничения на число станков и число тельферов, поскольку станки закреплены за конкретными рабочими, то ограничение

распространяется и на число рабочих. Поэтому за транзакты принимаются рабочие, определяемые операндом D ОБ GENERATE, которые и проходят через все этапы. Тогда число тельферов будут представлять собой памяти, а ДО SPT реализуется присвоением разного приоритета: для этапа 2 — приоритет 10, а для этапа 4 — приоритет 5. Будем предполагать, что в начале процесса моделирования доводочные станки реализуют процесс 1, а это означает, что в МВ 0.0 сразу начинается обработка детали (ОБ ADVANCE) и все Хакт помещены в СБС, где Хакт соревнуются за занятие тельфера для реализации этапа 2. Как было отмечено, ОБ ADVANCE является разделительной линией, перед которой меняется приоритет Хакт. Рассмотрим, как это реализуется в МФ примера 6.5 (см. п. 4 примера), ОБ ADVANCE отмечены цифрами в скобках. При зарождении приоритет каждого Хакт задается операндом E ОБ GENERATE. В ОБ ADVANCE (1) — первая разделительная линия, при выполнении процесса 1 Хакт используется с начальным значением приоритета, равным 10, и переходит из СБС в СТС с целью занять тельфер для перестановки детали на станке (ОБ ADVANCE (2)). Затем приоритет снижается с 10 до 5 в преддверии выполнения этапа 4 (перед ОБ ADVANCE (3) — вторая разделительная линия). Процесс 2 следует сразу после второй разделительной линии, Хакт снова совершают перемещение из СБС в СТС для запроса тельфера. И, наконец, после освобождения тельфера на этапе 4 приоритет должен быть снова повышен до 10 для осуществления в будущем этапа 2. Следовательно, ОБ ADVANCE (4) будет являться еще одной разделительной линией. После этой операции Хакт приобретает начальное значение и рабочий цикл начинается снова. В данном примере рассматривается также возможность получения данных об использовании операторов, что достигается с помощью введения так называемой *фиктивной памяти*. Это условное введение позволяет получить данные только по использованию интересующего объекта, достигается это следующим образом:

— вводится определение памяти в модуль управления;

— вводятся парные ОБ ENTER/LEAVE для получения информации об использовании операторов, которые не препятствуют входу Хакт, так как они не связаны с ограничениями на ресурсы, в частности, тельферы.

В МФ все операторы, связанные с фиктивными памятями, набраны наклонным шрифтом. Поэтому при первом прогоне модели они могут не использоваться. В этом случае в итоговом отчете будет получена информация только по загрузке тельферов.

Задание. После отработки МФ и получения данных по использованию тельферов отредактируйте МФ, введя в него фиктивные памяти для получения данных о загрузке операторов. Проверьте разные места введения ОБ ENTER/LEAVE, так как неправильное подключение приведет к появлению ошибок исполнения.

3. Таблица определений (табл. 6.10).

Временная дискрета: 1 мин

Таблица 6.10

Объект GPSS/H	Объект в системе
Транзакты	
Фрагмент 1	Рабочие
Фрагмент 2	Хакт управления
Памяти	
CRANE	Тельфер

4. Модельный файл.

В МФ введены цифры в скобках, которые не являются частью МФ, а служат для пояснений по п. 2.

```
    Модуль описания
    SIMULATE                                Пример 6.5. Проблема
                                            использования оборудования
*                                           (3 тельфера и 15 доводочных
                                            станков)
*                                           Временная дискрета: 1 мин
    CRANES      STORAGE     3               3 тельфера
OPERATOR        STORAGE     15              использование 15 операторов
                                            (фиктивная память)
* Модуль исполнения
*               Фрагмент 1                  рабочий цикл
                GENERATE    0,,,15,10       1 оператор на один станок,
                                            приоритет 15
    NEXTCYCL    ADVANCE     60,10           исполнение процесса 1 (1)
                ENTER       OPERATOR        использование фиктивной
                                            памяти
                ENTER       CRANES          запрос на тельфер для этапа 2
                ADVANCE     15,5            перестановка детали
                                            на станке (2)
                LEAVE       CRANES          освобождение тельфера
                PRIORITY    5               уменьшение приоритета
                                            для следующего этапа
                ADVANCE     90,20           исполнение процесса 2 (3)
                ENTER       CRANES          запрос на тельфер для этапа 4
                ADVANCE     30,5            снятие детали и установка
                                            заготовки (4)
                LEAVE       CRANES          освобождение тельфера
                PRIORITY    10              повышение приоритета
                                            для следующего этапа
                LEAVE       OPERATOR        освобождение фиктивной
                                            памяти
                TRANSFER    ,NEXTCYCL       возвращение к этапу 1
                                            для новой детали
```

```
*              Фрагмент 2                временной таймер
               GENERATE   2400           Хакт управления,
                                         приходящий после 40 ч
               TERMINATE  1              уменьшение СС до нуля,
                                         конец движения Хакт
*  Модуль управления
               START      1              установка СС на 1,
                                         начало движения Хакт
               END                       окончание ИМ
```

5. Итоговый отчет.

Итоговый отчет имеет стандартный вид (см. прил. 4), в случае введения фиктивных памятей данные по ним также приводятся в отчете. Табл. 6.11 содержит систематизированные данные по загрузке тельферов и рабочих для разных соотношений числа станков и тельферов.

Таблица 6.11

Отношение станки / тельферы	3/1	4/1	5/1	6/1	10/2	15/3
Использование тельферов	0.64	0.802	0.94	0.972	0.954	0.99
Использование рабочих	0.946	0.875	0.853	0.712	0.85	0.857

Примечание. Значения, приведенные в табл. 6.11, могут использоваться как контрольные при самостоятельном исследовании, расхождения во втором знаке допустимы.

6. Выводы и обсуждение.

1. При увеличении числа станков от 3-х до 6 на один тельфер коэффициент его использования возрастает.

2. При возрастании числа станков на один тельфер коэффициент занятости рабочих снижается, так как им все больше времени придется тратить на ожидание освободившегося тельфера.

3. При увеличении числа тельферов коэффициенты использования тельферов и рабочих одновременно возрастают.

4. Применение фиктивных памятей позволяет оценить долю занятости рабочих и при проведении каждого процесса в отдельности. Предоставим такую возможность читателю.

6.3.5. Использование уровней приоритета для управления получением данных

В процессе ИМ уровни приоритета могут использоваться следующим образом.

1. Для организации порядка обслуживания. По умолчанию внутри каждого уровня приоритета принимается ДО FCFS.

2. Для управления порядком съема данных в заданный момент МВ. Это обстоятельство представляется весьма важным при корреляции различных элементов модели, соответствующих реальным компонентам системы.

Если первый пункт очевиден, то второй требует комментариев. Его применение требует достаточного навыка в использовании ЯИМ и может при неверном расположении ОБ PRIORITY привести к логическим ошибкам.

Рассмотрим это на простом примере работы парикмахерской, когда темп прихода клиентов и время их обслуживания одинаковы, а в парикмахерской нет ни одного кресла для ожидания обслуживания, поэтому клиент уходит в другую парикмахерскую, если он немедленно не попадает на обслуживание. МФ для такого случая представлен ниже.

```
                SIMULATE
                GENERATE    20
                TRANSFER    BOTH,,BYEBYE
                SEIZE       JOE
                PRIORITY     5
                ADVANCE     20
                RELEASE      JOE
                TERMINATE   1
BYEBYE          TERMINATE   0
                START       100
                END
```

Предположим, что первоначально в МФ ОБ PRIORITY отсутствует, тогда каждый четный клиент (в соответствии с условиями задачи) будет покидать парикмахерскую, и на 100 клиентов, прошедших обслуживание, будет 99 отказавшихся от обслуживания, так как в момент прихода 2-го, 4-го и так далее клиентов мастер будет оканчивать обслуживание 1-го, 3-го и так далее клиентов. Поскольку приход нового клиента и окончание обслуживания предыдущего происходят в одно и то же время, то приходящий транзакт (клиент) стоит в начале СТС, и после сканирования выясняется, что предыдущий транзакт не покинул еще устройство обслуживания, выйдя из СБС, и при одинаковом приоритете он оказывается в СТС ниже по списку. Для устранения такого неправильного порядка следует приписать обслуживаемому транзакту больший приоритет, нежели приходящему, введя ОБ PRIORITY 5, что ставит его в СТС выше по списку, и при сканировании обслуживаемый транзакт учитывается первым, а исполнение ОБ RELEASE приводит ко входу приходящего транзакта на обслуживание. Уходов без обслуживания в этом случае не будет.

Задание. Промоделируйте обе ситуации, с ОБ PRIORITY и без него, а также попробуйте поставить ОБ PRIORITY после ОБ ADVANCE. Напомним, что ОБ PRIORITY всегда должен предшествовать ОБ ADVANCE!.

Рассмотренная ситуация несколько искусственна, так как детерминированные времена прихода и обслуживания встречаются крайне редко, но она подчеркивает порядок обработки связанных транзактов, что очень важно в процессе ИМ.

§ 6.4. АВТОМАТИЗАЦИЯ ПРОЦЕССА ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ И СОКРАЩЕНИЕ ИТОГОВОГО ОТЧЕТА

6.4.1. Автоматизация процесса ИМ

В примере 6.4 был приведен самый простой способ получения нескольких реплик в одном прогоне. Однако желание получить много реплик наталкивается на рутинную, чисто механическую процедуру написания многих пар ОБ CLEAR— START. Естественно, возникает необходимость упрощения и автоматизации рутинных операций, и GPSS/H предоставляет такую возможность за счет создания петель управления, реализуемых введением АМП и ОУ DO — ENDDO. Вначале рассмотрим смысл этих действий, а затем проиллюстрируем их примером.

Амперпеременные представляют собой символические имена для размещения в памяти ЭВМ различных величин. Поскольку в GPSS/H эти переменные предваряются знаком амперсанда &, то их название включает первые буквы этого символа (подробнее см. п. 4.4.2). Далее будем рассматривать только целочисленные и действительные (с плавающей точкой) АМП. Эти переменные играют большую роль в увеличении вычислительных возможностей GPSS/H и, в частности, с их помощью создаются петли управления автоматизацией получения многих реплик. АМП, никогда не имеющие метки, задаются в виде ОО в модуле описания. В операционном поле может быть задано одновременно несколько АМП, отделяемых друг от друга запятыми без пробела, появление первого пробела воспринимается как конец записи. ОО отличаются от ОБ и ОУ тем, что они только информируют, но не исполняются в процессе ИМ. Напомним правила обращения с АМП и ОО:

1) при компиляции все АМП начинаются с 0;

2) все ОО (и АМП в том числе) располагаются в верхней части модуля описания для поддержания правила компиляции МФ сверху вниз;

3) исполнение ОУ CLEAR не изменяет значений АМП;

4) АМП приводятся в итоговом отчете и могут быть вызваны в тестовом режиме командой display amp.

Петля управления в GPSS/H создается за счет последовательного использования двух ОУ DO/ENDDO, пока не будут выполнены условия останова. Метка у этих операторов не обязательна, операнды ОУ показаны в п. 5.1.2 и представляют собой начальное и конечное значения с необязательным операндом C, указывающим на величину приращения, например:

```
        DO              &I=1,10,1
            START       1
            CLEAR
        ENDDO
```

Вложенные ОУ смещены вправо от главных операторов петли управления, такая запись модуля управления позволяет проще читать директивы МФ. При нескольких уровнях петли управления смещение вправо регулируется ОО OPERCOL (см. п. 5.1.3, М2). Напомним, что ОУ DO/ENDDO управляют процессом моделирования, но не процессом движения транзактов, поэтому петля управления включает в себя только ОУ и никогда не включает ОБ! Второе, что необходимо неукоснительно соблюдать, — это отсутствие пробелов при записи петли управления, так как любой пробел справа после запятой воспринимается программой как начало комментариев. Правда, в последних версиях ЯИМ для большей читаемости петли управления пробелы и слева от знака =, и справа перед запятой допускаются, но в пособии будем придерживаться правил нотации, описанных в § 4.4. Использование петли управления представим с помощью модифицированного примера 6.2а.

Пример 6.2а. Модифицированная модель инструментальной кладовой

Модельный файл.

```
*               Модуль описания
    SIMULATE                        Пример 6.2а. Инструмен-
                                    тальная кладовая ДО SPT
*                                   Временная дискрета: 1 с
    INTEGER  &I                     индекс петли управления
*               Модуль исполнения
*               Фрагмент 1          1-й тип механиков
                GENERATE 420,360,,,5 приход механиков 1-го типа
*               (Приоритет 5)
                QUEUE    TOOLWAIT   постановка в очередь
                SEIZE    CLERK      запрос на обслуживание
                DEPART   TOOLWAIT   выход из очереди
                ADVANCE  300,90     обслуживание
                RELEASE  CLERK      освобождение кладовщика
```

```
                TERMINATE 0                 уход из кладовой
*               Фрагмент 2                  2-й тип механиков
                GENERATE 360,240,,,10       приход механиков 2-го типа
                         (Приоритет 10)
                QUEUE     TOOLWAIT          постановка в очередь
                SEIZE     CLERK             запрос на обслуживание
                DEPART    TOOLWAIT          выход из очереди
                ADVANCE   100,30            обслуживание
                RELEASE   CLERK             освобождение кладовщика
                TERMINATE 0                 уход из кладовой
*               Фрагмент 3                  Хакт управления
                GENERATE 28800              время моделирования 8 ч
                TERMINATE1                  уменьшение CC на 1,
*                                           окончание движения транзактов
*               Модуль управления
                DO        &I=1,10,1         управление проведением
                                            десяти реплик
                START     1                 старт очередной реплики
                CLEAR                       очистка данных для очередной
                                            реплики
                ENDDO                       завершение петли управления
                END                         окончание процесса ИМ
```

Приведенный пример МФ дается без принятой в пособии схемы представления, так как отличия от примера 6.2 сводятся только к появлению АМП в модуле описания и петли управления в модуле управления, а вся содержательная часть исполняемого МФ сохранена без изменений.

Таким образом, процесс автоматизации достигается путем несложных операций, сводящихся к коррективам в модулях описания и управления.

6.4.2. Сокращение итогового отчета

При получении большого числа реплик в одном прогоне иногда возникает задача выбора и последующего выделения в итоговом отчете нескольких именованных реплик из общего большего числа реплик. В GPSS/H это можно осуществить, используя ОУ PUTPIC (PUT out a PICture), подробно описанный в п. 5.1.2, К5. ОУ PUTPIC размещается в модуле управления и может иметь две формы: *немедленного* появления, когда сообщение следует сразу за появлением ОУ, и *отложенного* появления, когда сообщение появляется в другом месте отчета. В пособии используется только форма немедленного появления. Пример использования ОУ PUTPIC будет приведен в гл. 8 при

рассмотрении варианта примера 6.4, в котором будет использовано несколько не рассматривающихся до этого ОУ.

§ 6.5. РАЗБОР ОШИБОК И УПРАЖНЕНИЯ

В этом параграфе используется та же самая логика, что и в § 5.6, т. е. вначале разбираются случаи неправильного применения операндов или формы записи по материалу гл. 6, а затем рассматриваются упражнения, ответы на которые приведены в прил. 5.

6.5.1. Разбор ошибок

1. Что произойдет, если в МФ примера 6.1 исключить ОБ RELEASE? Будет ли окончен процесс ИМ? Проведите моделирование в пакетном режиме и оцените получившийся результат.

Объяснение: Очевидно, что при изъятии ОБ RELEASE логика поведения модели нарушится, однако это не будет замечено программой. Первый клерк, попавший в очередь к менеджеру, никогда не выйдет от него, а три остальных клерка будут блокированы. Причем процесс моделирования окончится в указанное время.

Отсюда следует, что исследователь должен очень внимательно проверять написанный МФ, не доверяя принципу, что если исполнение модели произошло, то все в порядке.

2. Что произойдет, если в МФ примера 6.1 исключить ОБ SEIZE? Будет ли окончен процесс ИМ, появляется ли сообщение об ошибках? Проведите моделирование в пакетном режиме и оцените получившийся результат.

Объяснение: Когда первый клерк выходит из ОБ RELEASE без прохода через менеджера (устройство), появится сообщение об ошибке исполнения «Transaction being destroyed controls or is in contention for a Facility» — транзакт нарушил условия управления или конфликтует с устройством.

3. Проведите эксперимент с МФ примера 6.3, удалив по очереди вначале ОБ QUEUE, а затем во второй попытке ОБ DEPART. Что при этом произойдет и появятся ли сообщения об ошибках?

Объяснение: МФ компиляцию пройдет удовлетворительно, а затем начнется процесс исполнения. В первом случае, после того как первый Хакт попадет на терминирование, появится предупреждение «Transaction being destroyed belongs to the one or more Queues» — транзакт разрушен, оставаясь в одной или более очередей. Сообще-

ние об ошибке не появляется, так как информация о пребывании в очереди не влияет на логику ИМ и на данные итогового отчета. Во втором случае при исполнении появится предупреждение «DEPARTing transaction not a member of the Queue» — транзакт, выходящий из очереди, не является членом очереди, и следом за предупреждением появится сообщение об ошибке «Queue contents have been decremented to less than zero» — содержимое очереди уменьшено на величину, меньшую нуля, и процесс ИМ мгновенно останавливается.

4. Рассмотрим МФ для ресторана быстрого питания из п. 6.2.2. Поменяйте местами SEIZE WINDOW и DEPART WINDOW. Какое время проведут автомашины в очереди к окну? Проведите моделирование в пакетном режиме и убедитесь, насколько ваши предположения совпали с данными окончательного отчета.

Объяснение: В том случае, когда ОБ SEIZE не находится между ОБ QUEUE и DEPART, он становится последовательным ОБ МФ, поэтому Хакт проходят очередь без задержки, процент нулевых входов становится равным 100 и никакой информации не собирается.

5. Объясните, что произойдет с транзактами, движущимися в такие ОБ:

```
1) TARGET       TRANSFER    .375,TARGET,SALTMINE
2) AGAIN        TRANSFER    .375,,AGAIN
3) ENDLESS      TRANSFER    .375,ENDLESS,ENDLESS
```

Объяснение: В случае 1) транзакт следует в ОБ, определяемый операндом С. Проблема возникает в другом: ОБ, определяемый операндом В, является сам ОБ TRANSFER. Поскольку ОБ TRANSFER никогда не препятствует входу Хакт, то он мгновенно исполняется, и Хакт либо идет по пути С, либо опять исполняет ОБ TRANSFER и т. д. В случае 2) транзакт все время движется в ОБ TRANSFER, как в свой последовательный ОБ. Такой случай явно не имеет практического смысла. В случае 3) идет непрерывное исполнение ОБ TRANSFER, пока пользователь не прервет этот бессмысленный цикл.

6. Объясните, почему следующая запись не имеет практического смысла:

```
TRANSFER        .0,PATH5,PATH7
```

Объяснение: Такой ОБ TRANSFER известен как безусловный, и транзакт со 100% -й вероятностью отправляется по пути В. Никаких сообщений об ошибке при этом не поступит, поскольку программа не предусматривает таких неквалифицированных действий пользователя.

6.5.2. Упражнения[1]

10. В МФ примера 6.1 измените значение временной дискреты с 1 мин до 30 с (0,5 мин), сравните результаты.

Вопрос: Существует ли разница в полученных результатах, и если существует, то чем она вызвана?

11. Используя МФ примера 6.1, ответьте на следующие *вопросы*.

A. В какое время каждый из 4-х клерков окончит работу над его первым документом?

B. Сколько времени 2-й клерк будет ожидать приема у менеджера?

C. Встретятся ли за модельное время пары связанных по времени транзактов?

12. В примере 6.1 введем экономические показатели: зарплата менеджера 500 р. в день, зарплата каждого клерка 350 р., стоимость расходных материалов 100 р., стоимость оформленного документа 300 р.

Вопрос: Какова дневная прибыль нотариальной конторы?

13. Используя данные упражнения 12, подсчитайте прибыль конторы за 40-часовую рабочую неделю для 3-х и 5 клерков соответственно.

14. Используя МФ примера 6.2, поменяйте местами фрагменты 1 и 2 модуля исполнения, проведите моделирование в пакетном режиме. Сравните полученные результаты с результатами примера 6.2, попробуйте объяснить несовпадение результатов.

15. В примере 6.2 оба типа механиков учитываются одной очередью TOOLWAIT, так что в отчете получаются суммарные данные. Попробуйте создать очереди отдельно для механиков 1-го и 2-го типов так, чтобы это было отражено в итоговом отчете.

16. Создайте нижеприведенный МФ, в котором время поступления объектов меньше времени обслуживания; вычислите, не проводя моделирование, ожидаемое число объектов на конвейере. Проведите моделирование в пакетном режиме. Оцените среднее число объектов на конвейере; оцените разницу в ожидаемом числе и полученном после моделирования. Какое максимальное число объектов на конвейере?

```
      SIMULATE
*
      GENERATE    25,20       объекты прибывают один за одним
                              и размещаются на конвейере
      QUEUE       MOVING      создание очереди MOVING
```

[1] Номера упражнений идут по нарастающей нотации.

```
        ADVANCE     100         время обслуживания
        DEPART      MOVING      выход из очереди MOVING
        TERMINATE   1           выход из системы
*
        START       500         число обрабатываемых объектов
        END                     окончание процесса ИМ
```

17. Используя в качестве пособия пример 6.5, постройте модель для более сложного случая, описанного ниже.

В цеху производится окончательная обработка изделия из поступающей заготовки, этапы обработки таковы.

1. Поступление заготовки со склада в темпе 12±3.
2. Погрузка заготовки на финишный станок 10±4.
3. Проведение финишного процесса 1 80±20.
4. Перестановка детали на станке 15±7.
5. Проведение финишного процесса 2 110±30.
6. Снятие изделия со станка 10±4.
7. Транспортирование изделия на склад готовой продукции 12±3.
8. Возвращение к этапу 1.

Изделия настолько тяжелы, что для манипулирования с ними на этапах 1, 2, 4, 6, 7 требуется тельфер. В цеху существует только один тельфер, который может быть запрошен рабочими, работающими на других станках, с периодичностью 39±10, время занятости крана другими запросами равно 25±10. Временная дискрета равна 1 мин. Принимайте за устройство тельфер, а за транзакты — рабочих как финишной машины, так и других станков. Считается, что в первый момент времени тельфер находится в распоряжении рабочего финишного станка, т. е. работа начинается без задержки, а первый запрос от других рабочих поступает через 39±10. Время моделирования 400 ч.

Вопросы:

A. В МФ предусмотрите получение информации об использовании тельфера при перестановке изделия на станке (этап 4), использовании тельфера на этапах 6, 7, 1, 2, а также при требованиях других рабочих. Определите коэффициент использования тельфера и количество изготовленных деталей.

B. Введя фиктивное устройство, получите загрузку рабочего финишного станка непосредственно в итоговом отчете.

C. Измените дисциплину обслуживания так, чтобы рабочий финишного станка имел преимущество перед другими запросами. Оцените изменения в полученных характеристиках по сравнению с п. 1.

18. Используя в качестве пособия пример 6.2, постройте модель для оценки и сравнения разных характеристик кладовщика. Пусть механик обращается за запасными частями каждые 300±250 с, кладовщик работает со скоростью 280±150 с.

Вопросы:

A. Постройте модель для 8 ч непрерывной работы, полагая, что простой станка стоит 600 р./ч. Определите общие потери за рабочий день, включая время ожидания в кладовой.

B. Оплата кладовщика составляет 20 р./ч, менеджер решил нанять более квалифицированного кладовщика со скоростью работы 280±50 с за 30 р./ч. Промоделируйте эту ситуацию и оцените выигрыш за счет сокращения времени ожидания. Оцените факт замены кладовщика, учитывая возрастание заработной платы, но ускорение темпа работы; какой из вариантов использования кладовщика предпочтительнее?

Отметим при этом, что операнд A у обоих кладовщиков одинаков. Это объясняет, почему в п. 1 было включено время ожидания. Время моделирования также 8 ч.

19. Промоделируйте МФ примера 6.3 в пакетном режиме, сравните данные, приведенные в таблице результатов примера, с данными, полученными вами.

Вопросы:

A. Используя входные данные примера, оцените ожидаемый коэффициент использования буксиров и сравните с результатами ИМ.

B. Придайте судам типа A больший приоритет, чем судам типа B, проведите моделирование. Как изменились средние времена пребывания в порту судов обоего типа?

C. Измените время прихода судов: для судов типа A 1.8±0.9 ч, а для судов типа B 0.8±0.4 ч. Определите без моделирования минимальное число буксиров и причалов обоего вида, нужных для нормальной работы порта.

D. Продумайте пути изменения МФ для того, чтобы удовлетворить условиям п. 3. Для начала снимите ограничения по числу буксиров и причалов (поставьте звездочку перед ОУ STORAGE или сотрите их). Промоделируйте и используйте результаты моделирования для определения минимума буксиров и причалов, сравните результаты с цифрами п. 3.

20. Вернитесь к условиям упражнения 17, используйте все входные данные без изменения, но теперь полагайте, что в цеху используется пять финишных станков, на каждом из которых работают отдельные рабочие, и имеется два тельфера. Требования на тельферы со стороны других рабочих сохранены. Проведите моделирование в пакетном режиме и оцените результаты, сравнив их с результатами упражнения 17 (по коэффициенту загрузки рабочих и тельфера).

21. Рассмотрим пример 6.4, в котором телевизоры, прошедшие подрегулировку, возвращаются в общую очередь, где действует усло-

вие, что 12 % проверяемых телевизоров переходят на регулировку. Такое условие для отрегулированных и вернувшихся в очередь для контроля телевизоров вряд ли является правильным.

В связи с этим перестройте МФ примера 6.4 с учетом того, что уход на новую регулировку ранее отрегулированных телевизоров составляет 1 %. После моделирования сравните данные с примером 6.4.

22. Постройте МФ для следующей ситуации: в цеху ведется обработка деталей на станке с ЧПУ, среднее время поступления деталей на ЧПУ 10±5 мин, 30 % деталей типа А, остальные — типа В, оба типа деталей поступают на станцию 1, где обработка ведется в течение 20±10 мин. Детали типа А поступают затем на станцию 2, где они обрабатываются в течение 30±10 мин, затем они поступают на станцию 4, где проводится окончательная обработка в течение 25±10 мин. Детали типа В после станции 1 поступают на станцию 3, где они обрабатываются в течение 45±5 мин, затем они поступают на станцию 4, где окончательная обработка ведется 25±10 мин.

Вопросы:

А. Собрать информацию со всех рабочих станций, используя ДО FCFS, предполагая, что количество рабочих на каждой станции неограниченно (все очереди имеют нулевой вход). Моделирование провести для 100 деталей, определив среднее число рабочих на каждой станции.

В. По данным п. 1 определите для станций ближайшее целое число рабочих, установив соответственно такое же число идентичных серверов. Определите после моделирования все необходимые характеристики системы.

23. Постройте МФ для следующей ситуации: производственная линия состоит из двух рабочих станций РС1 и РС2, на каждой из которых работает по одному рабочему. Обрабатываемая деталь проходит последовательно через обе РС. Перед РС1 не существует ограничений на размещение любого числа приходящих заготовок, однако между РС1 и РС2 существует всего одно место для размещения незавершенной детали, прежде чем она поступит на РС2. Поэтому, если на месте хранения между РС1 и РС2 лежит деталь, рабочий РС1 вынужден ожидать, когда оно освободится, даже если он закончил операции со своей деталью. Заготовки прибывают на РС1 в интервале 30±20 мин, цикл обработки на РС1 25±10, а на РС2 30±10 мин. Будем полагать, что детали достаточно тяжелы и их надо перемещать с помощью кара (названного в МФ AGV — automated guided vehicle). В системе существует только один кар, вызов кара занимает 30±15 с, на каждое перемещение (на РС1, место хранения между РС1 и РС2, на РС2 и с РС2) тратится по 30 с. Если после окончания опера-

ций на РС1 РС2 свободна, то деталь сразу перемещается на РС2, что экономит операционное время. Поэтому больший приоритет должен быть приписан освобождению РС2, а не освобождению РС1. Рабочие не участвуют в процессе погрузки и разгрузки деталей с кара, однако они не могут начать следующую операцию, пока деталь после обработки не будет удалена с РС. Провести ИМ для 100 деталей.

Вопросы:

A. Когда окончится процесс ИМ?

B. Какие существуют линии раздела?

C. Как осуществить прямой переход к РС2?

D. Какова загрузка кара?

Глава 7
РАБОТА С ОТЛАДЧИКОМ ПРОГРАММ МОДЕЛИРОВАНИЯ (ТЕСТОВЫЙ РЕЖИМ)

§ 7.1. ОСОБЕННОСТИ РАБОТЫ С ОТЛАДЧИКОМ

Как уже было отмечено, одним из преимуществ GPSS/H является наличие эффективного отладчика программ моделирования (дебаггера), который позволяет пользователю работать с программой в диалоговом режиме, осуществляя не только тестовую проверку, но и введение необходимых изменений. Работа в пакетном режиме позволяет оценивать результаты моделирования лишь по окончании процесса ИМ. Поэтому констатация наличия ошибок в итоговом отчете мало что дает исследователю, удлиняя и удорожая процесс исследования. В связи с этим интерактивный режим оказывается весьма полезным, так как он дает возможность не только получать промежуточную информацию о той части МФ, которая вызывает наибольшие сомнения, но и вносить конструктивные изменения. В пакетном режиме очень трудно до начала ИМ выделить ту часть итогового отчета, которая бы представила наибольший интерес, а также оценить заранее характер и возможность возникновения ошибок. Тестовый режим в большинстве случаев снимает ограничения, присущие пакетному режиму. Техника и технология использования возможностей отладчика будут рассмотрены в нижеследующих параграфах. Подчеркнем еще раз, что последовательность глав второго раздела такова, что читать их надо по мере появления первого упоминания об отладчике. Так, первые упражнения гл. 6 требуют обращения к отладчику, а следовательно, к материалу гл. 7. Сведения об отладчике специально собраны в одной главе, чтобы не искать информацию о нем по всему пособию. Поэтому, встретив упоминание об отладчике, следует прочесть два первых параграфа гл. 7 и научиться работать в диалоговом режиме. Последующие параграфы можно изучить позже, когда понадобится активное воздействие на отработку вами созданной программы моделирования с помощью специальных команд отладчика.

7.1.1. Запуск отладчика

Запуск отладчика (тестовый режим, дебаггер) осуществляется записью (после появления приглашения в командной строке используемой оболочки или DOS) следующего вида:

< gpssh filename tv >

где filename — имя МФ с расширением .gps, вызываемое, в том числе, одновременным нажатием клавиш Ctrl — j, курсор находится на имени файла; tv(test vision) является командой запуска тестового режима. Вызов отладчика возможен только в случае успешной компиляции вашего МФ. Не пугайтесь, если отладчик не вызывается — вернитесь к вашему МФ и по F4, основываясь на данных об ошибках, содержащихся в листинге отчета (имя вашего файла с расширением .lis), внесите все необходимые коррективы и лишь затем повторите попытку вызова отладчика.

Кроме указанного формата запуска отладчика может использоваться и такой:

< gpssh filename tv type nowarn или
gpssh filename tvtnw >

type — представляет собой команду отобразить на экране дисплея отчет; nowarn — представляет собой команду запрета отображения на экране сообщений об ошибках. Вторая строка содержит ту же самую информацию в сокращенном виде, но адекватно воспринимаемую программой. Вторая форма записи в пособии не используется.

После исполнения команды вызова отладчика на экране дисплея появляется трехоконный отладчик, который представлен на рис. 7.1, *A* и *B*. Вариант *A* наблюдается на экране дисплея сразу после вызова отладчика, в качестве примера рассмотрен простейший вариант поступления деталей со склада, их перемещение по ленте транспортера, обработка на станке и уход из системы. Процесс ИМ длится до получения данных о 100 обработанных деталях.

Сразу после вызова окна пусты, а на рис. 7.1, *A* окна для удобства помечены звездочками, которых в действительности нет: одной звездочкой помечено окно исходного МФ (SOURCE FOR MODEL), двумя звездочками — статусное окно и тремя звездочками — диалоговое окно. На рис. 7.1, *B* представлены те же окна после инициализации МФ примера 7.1, приведенного в п. 7.4.7. Инициализация вызванного трехоконного отладчика производится двумя эквивалентными путями:

1) нажатием клавиши F10, что соответствует одному такту действия отладчика (каждое последующее последовательное нажатие клавиши продвигает процесс ИМ на один такт);

2) написанием в командной строке после приглашения (:) команды

< s[tep]_ n >

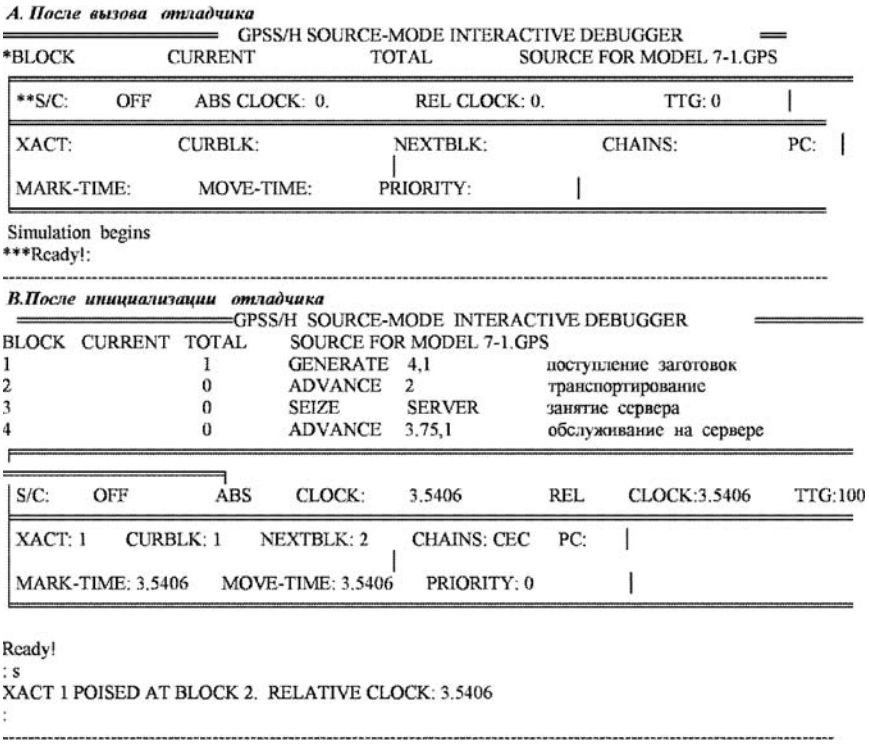


Рис. 7.1. *Вид трехоконного отладчика: А — в начальный момент вызова; В — после инициализации отладчика (первый шаг)*

где команда может быть написана в любом виде, начиная от четырех букв до одной первой буквы и через обязательный пробел число тактов, на которые продвинется процесс ИМ (по умолчанию 1), т. е. s = F10. При отсутствии пробела появится предупреждение об ошибке, после которого необходимо написать команду в правильном формате.

7.1.2. Содержание окон

Рассмотрим теперь более подробно содержание окон, руководствуясь рис. 7.1, *В*, поскольку все названия у окон обоих вариантов одинаковы, но в варианте *В* присутствует конкретное содержание.

Примечание. Окно диалога варианта *А* содержит запись — Simulation begins, которая исчезает после инициализации, кроме того, на

рисунке не показаны всегда существующие надписи об авторе программы (см. реальное окно отладчика).

Окно исходного модельного файла (окно источника)

Занимает одну треть экрана дисплея и является верхней частью экрана отладчика.

• Справа вверху первого окна отладчика следует общее название «Исходный файл интерактивного дебаггера GPSS/H».

Далее в окне следуют названия колонок:

• BLOCK — обозначает номер ОБ;

• CURRENT — показывает ИН Хакт в ОБ;

• TOTAL — показывает общее количество транзактов в ОБ за время ИМ;

• SOURCE CODE (основан на примере 7.1) — показывает 4 или 5 строк исходного МФ, а красная черта указывает на следующий ОБ, куда направится Хакт; если МФ длинный, то высекается только фрагмент, указывающий нахождение Хакт, следующий ОБ и 2–3 сопутствующих ОБ.

Окно текущего положения (статусное окно)

Занимает среднюю часть экрана и содержит информацию о состоянии модели, меняющуюся по мере продвижения Хакт по модели.

• S/C:OFF — положение флага изменения статуса (см. п. 5.4.3): выключено;

• ABS CLOCK — значение абсолютного МВ;

• REAL CLOCK — значение относительного МВ. Оба эти значения в отсутствии ОУ CLEAR или RESET имеют одинаковую величину;

• TTG — (termination to go) — значение СС. Поскольку на первом такте терминирование не происходит, то это значение равно значению операнда А ОБ START и уменьшается при каждом терминировании на величину операнда А ОБ TERMINATE;

• XACT — ИН транзакта, активного в данный момент;

• CURBLK — имя или номер ОБ, где находится Хакт;

• NEXTBLK — имя или номер ОБ, куда направляется транзакт;

• CHAINS:CEC — имя списка, в котором транзакт находится в настоящий момент, в данном примере — СТС;

• PC: — число захватов (в пособии не используется);

• MARK-TIME — время, отмечаемое при входе Хакт в модель;

• MOVE-TIME — время будущего начала движения, если Хакт находится на обслуживании;

• PRIORITY — уровень приоритета Хакт.

Окно диалога

Находится в нижней трети экрана и показывает команды, задаваемые пользователем, реакцию отладчика на эти команды, пригла-

шения разного вида (см. далее по тексту). Информация окна автоматически прокручивается вверх для освобождения пространства для новой информации.

• Ready! — сигнал, что отладчик готов к началу диалога;

• :s — приглашение в начале командной строки;

• XACT1 — запись после выполнения такта, гласящая, что «Хакт 1 располагается перед ОБ 2», т. е. вышел со склада и готов начать движение на транспортере. Выражение «poised at ...» всегда обозначает,что Хакт пытается двигаться в ОБ, номер которого указан в сообщении.

7.1.3. Выход из сеанса отладчика

Прерывание тестового режима и переход в командную оболочку можно осуществить двумя путями.

1. Команда QUIT, написанная в командной строке, прерывает сессию отладчика и возвращает управление DOS. Однако при этом программа вначале проверяет, нет ли какой незавершенной операции, которая должна попасть в листинг отчета, и если такая операция существует и она не завершена, то программа уточняет, действительно ли вы хотите прервать сессию. При этом у пользователя имеется два выбора: повторить запись quit и прервать сессию с потерей неоконченных операций или написать run или continue, позволяющие завершить все операции.

2. Команда QQ или равноценная — qq (Quit Quickly) сразу прерывает сессию отладчика и немедленно возвращается в командную оболочку без какой-либо проверки незавершенных операций.

Обе эти команды не имеют никаких операндов.

§ 7.2. ФУНКЦИОНАЛЬНЫЕ КЛАВИШИ И КОМАНДЫ ОТЛАДЧИКА

7.2.1. Функциональные клавиши

После запуска отладчика функциональные клавиши F1 –F10 и клавиши прокрутки выполняют специфические функции, которые указаны в табл. 7.1

Имеет смысл потренироваться с использованием всех клавиш, чтобы запомнить их назначение.

Таблица 7.1. *Роль функциональных клавиш*

Клавиша	Выполняемая функция
Стрелка вверх	Прокрутка окна диалога на 5 строк вверх
Стрелка вниз	Прокрутка окна диалога на 5 строк вниз
Стрелка влево	Прокрутка окна диалога влево на 20 символов
Стрелка вправо	Прокрутка окна диалога вправо на 10 символов
F1	Перемещение к началу окна диалога
F2	Удаление всех окон, кроме окна диалога
F3	Перемещение к концу окна диалога
F4	Возвращение к трехоконному виду
F5	Прокрутка окна диалога влево на 20 символов
F6	Прокрутка окна диалога вправо на 10 символов
F7	Прокрутка вверх на 1 линию окна МФ
F8	Не используется
F9	Прокрутка вниз на 1 линию окна МФ
F10	Продвижение процесса ИМ на один такт = step

Смысл выполняемых стрелками и клавишами функций может быть изменен при использовании команды SET.

7.2.2. Команды и коды объектов

Полный список команд, используемых в сеансе отладчика, приведен в табл. 7.2. Логика использования наиболее применяемых будет рассмотрена в § 7.4, эти команды отмечены *.

Таблица 7.2. *Команды отладчика (в алфавитном порядке)*[1]

Команда	Операнды	Пояснение
AT*	Имя или номер одного или нескольких ОБ	Предлагает список команд отладки, исполняемых каждый раз при достижении Хакт назначенного ОБ, прерывается командой END или для ранее установленных точек командой UNBREAK

[1] В списке классов и кодов объектов подчеркнуты допустимые сокращения

Продолжение табл. 7.2

Команда	Операнды	Пояснение
BREAK*	Имя или номер одного или нескольких ОБ	Вводит прерывание процесса для каждого установленного ОБ, глобальное прерывание снимается командой UNBREAK, локальное прерывание снимается командой CONTINUE
CHECKPOINT	Не имеет	Команда записывается в единственном специальном файле для определения точек контроля одна за другой
CONTINUE*	Нуль или несколько имен	Служит для возобновления ИМ или для снятия локальных точек прерывания
	SCAN, NEXT, SYSTEM	Устанавливает локальные запреты, подробнее см. в § 7.4
CX	Нуль или больше имен	Аналогична команде CONTINUE, за исключением запрета на создание дополнительного отчета
DISPLAY*	Статус или код объекта, см. прим. 1, 2	Выводит на экран дисплея статистики многих объектов и другую информацию, прерывается нажатием клавиш BREAK, CTRL-C, CTRL—BREAK
DS	Дополнительное имя ОБ	Показывает код МФ для рассматриваемого ОБ в окне источника отладчика
IGNORE	Имя или номер последующего ОБ	Используется для отмены глобальных прерываний, включая количество отмен
NEXT	Не имеет	Определяет время исполнения модели до прихода нового Хакт из СТС, редко применима
PRINT	Статус или код объекта, см. прим. 1, 2	Печатает информацию для многих объектов в итоговом отчете, а не на экране дисплея
QQ*	Не имеет	Немедленно прекращает процесс ИМ
QUIT*	Не имеет	Прекращает процесс ИМ при выполнении условий

Окончание табл. 7.2

Команда	Операнды	Пояснение
RUN*	Нуль или больше имен	Исполняет процесс ИМ, снимает все условия остановки процесса ИМ
	SCAN, SYSTEM, NEXT, CLOCK, XACT	Устанавливает глобальные условия запрета, см. § 7.4
RESTORE	Не имеет	Возвращает модель в положение до контроля
SET*	TIME =n,nS,nM	Устанавливает предел времени исполнения процесса в секундах или минутах
	TLOG[=]ON,OFF	Переключает логику действия терминала, при включении отображает на экране
	TV[=] ON,OFF	Управляет показом процесса отладки
STEP*	Число тактов	Задает темп продвижения по модели
STOP*	Не имеет	Прерывает на время процесс ИМ
TRAP*	SYSTEM	Определяет, что в каждый момент времени Хакт останавливается, контроль передается пользователю
	NEXT	Хакт снимается из СТС, появляется сообщение, контроль передается пользователю
	SCAN	Производит сканирование СТС, контроль передается пользователю
	CLOCK	Ставит ограничение на абсолютное время, когда оно достигает назначенной величины, контроль передается пользователю
	XACT	Ставит ограничение на Хакт с определенным ИН
UNBREAK	Одно или несколько имен ОБ	Снимает глобальные условия прерывания
UNTRAP*	SYSTEM, NEXT, CLOCK, XACT	Снимает условия запрета со всех точек модели

Приложение.

1. Коды состояния модели:

ATL blockname/ num	список АТ имен или номеров ОБ
ATP	все точки АТ
BREAKPOINTS	все глобальные точки прерывания
CEC	список текущих событий
CLOCKS	абсолютное и относительное время
COMMON	использование общей памяти
CPU	время ЦПУ
FEC	список будущих событий
INT	список прерываний
MAT	список соответствий
OUTPUT	стандартный отчет
STATUS	значения ИН Хакт, времени, значения СС
TRAPS	все глобальные запреты

2. Имена классов объектов:

AMP	отображение всех АМП
BLO	отображение данных ОБ
FAC	отображение данных об устройствах
QUE	отображение данных об очередях
RNO	выходные данные о БСВ
STO	отображение данных о памятях
TAB	отображение данных о таблицах

§ 7.3. ОСНОВЫ ИСПОЛЬЗОВАНИЯ ОТЛАДЧИКА

Наличие отладчика примечательно тем, что после успешно проведенной симуляции и исполнения команды ОУ START можно передать управление программой пользователю. Пользователь может применить одну или несколько команд отладчика, дальнейшая работа может вестись в диалоговом режиме, с последовательной передачей функций управления от программы к пользователю и обратно. Такой режим способствует отладке программы, уточнению сомнительных мест, выводу на дисплей любой интересующей пользователя информации. Процесс продолжается до подачи команды STOP или любой команды выхода из тестового режима, например QQ. Отметим, что в тестовом режиме (сеансе отладчика) *невозможно внести какие-либо изменения в МФ*! Для внесения изменений необходимо прервать сеанс отладчика, вернуться к редактору, с помощью которого вы создавали МФ, внести необходимые изменения и лишь после

этого вернуться в тестовый режим. Алгоритм проведения диалога между пользователем и GPSS/H представлен на рис. 7.2. Действия (в виде пронумерованных линий) следующие.

Линия 1. После обращения к тестовому режиму считается, что процесс компиляции успешно проведен, прошло исполнение ОУ и программа готова начинать исполнение модели по команде ОУ START. В этот момент программа прерывает исполнение, пишет «Ready» и передает управление пользователю, вызывая приглашение (:) в командной строке и ожидая команд пользователя.

Линия 2. Пользователь может написать одну или несколько команд, при исполнении которых исполнение программы прерывается и управление вновь передается пользователю. Затем подается команда по подведению итогов.

Линия 3. Программа начинает процесс ИМ и переходит к фазе движения транзактов, процесс ИМ продолжается до начала действия условий прерывания или запрета. В этот момент процесс ИМ прерывается, на экране дисплея появляется сообщение о поставленных условиях, передается пользователю и предлагается (:) пользователю совершать дальнейшие действия.

Линия 4. Пользователь пишет соответствующие команды для изменения условий прерывания и/или давая команду на выдачу информации.

Линии 3 и *4* повторяются несколько раз, до тех пор, пока не будут выполнены задачи пользователя.

Линия 5. Выход из интерактивного тестового режима.

Линия 6. Представляет собой путь при уменьшении СС до нуля или меньше, при выполнении этих условий программа выдает сооб-

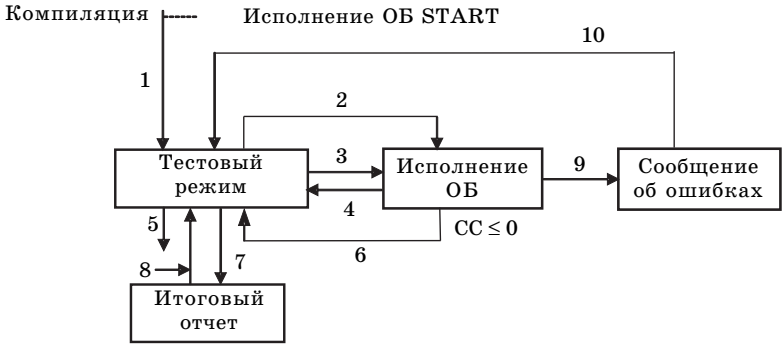


Рис. 7.2. *Блок-схема алгоритма диалога пользователя и программы в тестовом режиме*

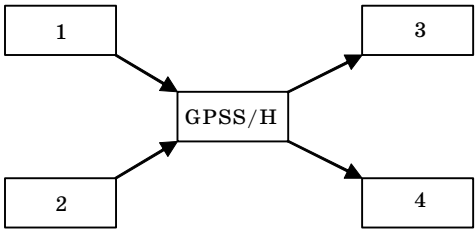


Рис. 7.3. *Архитектура входов и выходов программы:*
1 — входной МФ; 2 — входные команды диалогового режима; 3 — стандартный итоговый отчет; 4 — отчет в процессе диалога

щение о подготовке итогового отчета и передает управление пользователю.

Линии 7 и *8*. Если после сообщения о подготовке отчета пользователь подаст команду Step или нажмет функциональную клавишу F10, то программа выводит итоговый отчет на дисплей. После показа отчета по линии *8* управление возвращается пользователю. Чаще всего, получив сообщение по линии *6*, пользователь переходит по линии *5* к выходу из тестового режима.

Линии 9 и 10. Если в процессе ИМ произойдет ошибка исполнения, программа выдает сообщение об ошибках, которые появляются в диалоговом окне, передавая управление пользователю, который обычно должен прервать тестовый режим, вернуться в командную оболочку и откорректировать МФ.

Примечание. После прочтения этого параграфа имеет смысл вернуться к параграфам, содержащим проверку ошибок, и рассмотреть приведенные в них МФ с помощью тестового режима. Напомним, что всегда можно прервать тестовый режим и вернуться в DOS или командную оболочку.

Задание: Возьмите любой простой МФ с числом стартов не более 5 и проведите пошаговые операции, наблюдая за всеми записями в диалоговом окне и изменениями, проходящими в окне состояний.

Кроме алгоритма, представленного на рис. 7.2, рассмотрим те возможности, которые открывает GPSS/H в использовании архитектуры входа и выхода (рис. 7.3).

§ 7.4. ПРИМЕНЕНИЕ ОСНОВНЫХ КОМАНД В ТЕСТОВОМ РЕЖИМЕ

Далее рассмотрим команды, отмеченные * в табл. 7.2. Команды рассматриваются не в алфавитном порядке, а по частоте использования и по важности. Первой рассмотрим команду DISPLAY, позволя-

ющую выводить на экран дисплея разнообразную информацию о состоянии модели и о данных различных объектов.

7.4.1. Команда DISPLAY

Команда используется для выдачи информации о членах классов объектов или о состоянии модели. Классы объектов и коды состояний приведены в комментариях 1, 2 табл. 7.2.

Информация о членах классов объектов имеет следующий формат:

< D [isplay] класс объекта(имя,номер) >

Приведем несколько примеров (обратите внимание на возможные варианты записи команды):

DISPLAY	FAC — вывести данные об устройствах
DIS	QUEUE(SAM,FRED) — данные об очередях SAM, FRED
D	BLO(1...10,20) — данные об ОБ с первого по десятый и о 20-м
d	AMP — показать все АМП

Если нет членов класса, удовлетворяющих записанной команде, то информация не отображается. Например, если запрашивается информация об устройствах, а ни одно устройство пока не захвачено или не прервано, то информация на дисплее не появляется.

Информация *о статусе модели* имеет следующий формат (впредь будем пользоваться только сокращенной формой записи команды):

< d код... >

а для транзактов

< d xact=n >

например:

d CPU COM — показать время исполнения ЦПУ и загрузку памяти
d STA — показать состояние модели (Хакт, время, состояние СС)
d CEC FEC — показать СТС и СБС
d ATL 12 — показать список АТ команд для ОБ 12
d XACT=100 — показать транзакт с ИН 100
d XACT=3 XACT=10 — показать Хакт с ИН 3 и 10

Информация, показываемая для каждого класса или состояния (за исключением информации об АМП), идентична данным выходного отчета для объектов. Информация о состоянии модели при введенных условиях запрета и прерывания специфична для отладчика.

Информация о транзактах идентична информации об активных транзактах в окне состояния отладчика; если нет текущих транзактов, то в ответ на запрос появляется сообщение об их отсутствии.

7.4.2. Команды TRAP и UNTRAP

Команда TRAP может применяться для установки запрета в самых разных вариантах, которые выбираются пользователем. В GPSS/H существует пять видов наложения условий запрета. В любом случае при введении запрета появляется сообщение и управление передается пользователю.

1. Условия запрета **SYSTEM** проявляются каждый раз, когда транзакт по какой-либо причине больше не может двигаться по модели. Среди этих причин можно назвать:

— транзакту запрещен вход в ОБ;

— транзакт попадает в СБС;

— транзакт размещается в списке пользователя (ОБ LINK);

— транзакт уничтожен (ОБ TERMINATE, ASSEMBLE).

Эта форма запрета используется как барьер при изучении движения транзакта, представляющего большой интерес для исследователя. Если вдруг транзакт неожиданно исчезает из поля зрения, то эти условия позволяют определить его судьбу и принять решение о других командах.

2. Условия запрета **NEXT** возникают всякий раз, когда симулятор вводит Хакт после сканирования СТС. При введении этого запрета Хакт располагается перед его следующим ОБ и появляется сообщение "XACT i POISED at BLOCK j ", где i и j могут быть номерами или именами.

Эти два условия в пособии не используются.

3. Условие запрета **CLOCK** проявляется, когда абсолютное время достигает или превышает заданное время, при этом процесс ИМ останавливается в это время. Если абсолютное время превысило заданное, то его величина фиксируется для продолжения процесса ИМ после снятия запрета со значения, превысившего заданное время.

4. Условия запрета **XACT** налагаются на определенный транзакт, и всякий раз, когда Хакт с этим номером снимается после сканирования из СТС, появляется сообщение и управление передается пользователю. Условия запрета налагаются заранее, и по умолчанию роль отмеченного Хакт прослеживается за весь жизненный цикл.

5. Условия запрета **SCAN** реализуются каждый раз, когда симулятор готов начать сканирование СТС, при этом появляется сообще-

ние «SCAN TRAP TAKEN (SYSTEM POISED TO BEGIN CEC SCAN)» — запрет по сканированию введен (система готова начать сканирование СТС) и управление передается пользователю.

Условия запрета не допускают использование команд IGNORE и AT!

Условия запрета всегда реализуются с помощью команды TRAP!

Формат команды TRAP для всех видов запретов имеет вид

< TRAP Y [Y ...] >, Y = (SYSTEM, NEXT, CLOCK,...)

например:

TRAP SYSTEM NEXT — наложены запреты SYSTEM и NEXT;

TR XACT=1 XACT=5 — наложены запреты на Хакт ИН1 и ИН5;

T CLOCK 1000 — наложено ограничение по времени — 1000 временных дискрет;

t SCAN — наложен запрет на сканирование СТС.

Для снятий условий запрета используется только команда UNTRAP, которая должна предварять любые команды продолжения процесса ИМ. Так, если будет применена одна из команд RUN, CONTINUE, STEP, то процесс ИМ при отсутствии команды UNTRAP будет продолжаться только до одной из ближайших точек прерывания или запрета. Формат команды UNTRAP аналогичен формату команды TRAP, поэтому приведем лишь примеры использования команды:

UNTRAP SYSTEM NEXT — снятие запретов SYSTEM и NEXT;

UNT XAC=1 XACT=5 — снятие запретов с Хакт ИН1 и ИН5.

Программа GPSS/H предусматривает два вида запретов: *глобальные и локальные*. Команда TRAP налагает только глобальные запреты, которые могут быть сняты только командой UNTRAP. Локальные же запреты, налагаемые командой CONTINUE, приводимой в п. 7.4.5, действуют только на протяжении одной команды отладчика.

7.4.3. Команды BREAK и UNBREAK

Условия прерывания также могут быть *глобальными* и *локальными* и применяться к любому количеству ОБ одновременно. Глобальные условия прерывания действуют до тех пор, пока они не будут специально сняты; локальные условия, устанавливаемые командой CONTINUE, действуют на протяжении одной команды. Имена или номера ОБ, на которые наложены условия прерывания, могут быть отображены на дисплее после команды < d BRE >. Когда Хакт достигает ОБ, к которому применены условия прерывания, появляется сообщение, содержащее ИН Хакт, ОБ, текущее значение времени,

после чего управление передается пользователю. Формат команд одинаков и имеет вид

```
< B[REAK]      имя блока >
< UNB[REAK]    имя блока >
```

например:

BREAK	5	— прерывание на ОБ 5
B	5 LINE	— прерывание на ОБ 5 и LINE
UNBREAK	5	— снятие прерывания с ОБ 5
UNB	5 LINE	— снятие прерывания с ОБ 5 и LINE

7.4.4. Команда AT

Команда AT обеспечивает в каждый момент времени возможность автоматизации отображения списка команд, относящихся к ОБ, в который пытается войти Хакт. Формат команды AT имеет вид

```
< AT имя или номер ОБ
@ команда 1
...
@команда n
@END
:                    >
```

Когда достигается точка прерывания на заданном ОБ, то при наличии команды AT список команд, ассоциирующихся с этой командой, автоматически выполняется программой. Таким образом, команда AT может быть использована как для задания условий прерывания, так и для запуска процесса выполнения списка команд. Обратите внимание на следующее обстоятельство: как только в командной строке появится команда AT, так сразу форма приглашения с (:) меняется на (@), которая предваряет каждую команду списка, после исполнения команды end форма приглашения в командной строке снова принимает привычный вид (:). Следует помнить, что выполнение любой команды из списка осуществляется после нажатия пользователем клавиши Enter. Приведем пример использования команды:

```
: at block6
@      d blo
@      r
@ end
:
```

Все вышеназванные команды используются для запуска или реализации процесса ИМ.

Команды, находящиеся в списке, для лучшей читаемости специально сдвинуты вправо. В п. 7.4.7 будет рассмотрена сессия отладчика (пример 7.1), где использованы многие из приведенных команд.

7.4.5. Команды RUN, CONTINUE, STEP

Эти команды могут использоваться для достижения точек запрета или прерывания.

1. В нормальных условиях (при отсутствии специальных условий) команда RUN обеспечивает проведение процесса ИМ в пакетном режиме с нормальной скоростью моделирования, что, естественно, исключает возможность диалога. При наличии условий запрета или прерывания команда RUN сразу доводит процесс ИМ до первой специальной точки. Формат команды имеет вид

< R[UN] Y > Y= (block,SCAN,XACT,...)

например:

RUN		— отсутствие специальных точек, процесс ИМ идет до конца
RUN	SAM 17	— движение до глобальных прерываний в ОБ SAM и 17
R	CLOCK 400	— запрет процесса ИМ после 400 дискрет МВ
R	XACT=5 3	— запрет на Хакт с ИН5, прерывание на Хакт с ИН3

2. Команда C[ONTINUE] аналогична команде RUN, т. е. доводит процесс ИМ до конца (при отсутствии специальных условий), а кроме того, с ее помощью можно установить локальные условия прерывания и локальные условия запрета только для запретов типа NEXT и SYSTEM. Локальность запрета заключается в том, что его действие распространяется только на период исполнения команды CONTINUE, появление первой глобальной специальной точки прерывает исполнение команды, а следовательно, отменяет локальные условия. Формат этой команды имеет такой же вид, как у команды RUN. Приведем примеры:

CONTINUE		— отсутствие ограничений, процесс ИМ идет до конца
CONTINUE	SAM 17	— локальные прерывания на ОБ SAM и 17
C	NEXT	— локальный запрет типа NEXT
c	12 SYSTEM	— прерывание на ОБ 12 и запрет типа SYSTEM

3. Команда STEP используется в так называемом тактовом или пошаговом режиме, служит для продвижения процесса либо на один

такт, что равносильно исполнению одного ОБ или нажатию функциональной клавиши F10, либо на то число тактов, которое задается после обязательного пробела за именем команды. Формат команды имеет вид

< S[TEP] n >

по умолчанию число тактов равно 1, например:

```
STEP      5      — исполнение пяти ОБ
S                — исполнение одного ОБ
s     2          — исполнение двух ОБ
```

Команда STEP не налагает ограничений на отдельные транзакты, поэтому необходимо использовать ограничения типа NEXT или SYSTEM. Переход в пошаговый режим исключает возможность использовать команду CONTINUE, которая применима только в нормальном режиме.

7.4.6. Команды STOP, SET

1. Команда STOP используется для прерывания сеанса диалога. Если в момент прерывания подготавливается итоговый отчет, то об этом появляется сообщение и необходимо еще раз написать ту же команду, тогда процесс останавливается. Формат команды имеет простой вид

< STO[P] >

пример применения очевиден.

2. Команда SET используется для установления предела времени исполнения программы. Когда процесс ИМ достигает предела МВ, заданного либо извне, либо с помощью операнда A ОБ SIMULATE, появляется сообщение, процесс останавливается и управление передается пользователю. Формат этой команды имеет вид

< SE[T] TIME [=] n [S,M] >

например:

```
SET TIME = 1.5M   — установить предел МВ, равный 1.5 мин
SE   TIME   90S   — то же самое в секундах
se   TIME   20    — появляется сообщение «Minutes assumed»
```

Ряд команд не рассмотрены в этом параграфе, поскольку они не нужны на начальном периоде освоения отладчика.

7.4.7. Пример использования введенных команд

Применение основных команд проиллюстрируем с помощью простого примера, который будет основанием для рассмотрения двух сеансов отладчика, использующего введенные выше команды.

Пример 7.1. Обработка деталей в механическом цеху

1. Постановка задачи.

Заготовки в темпе 4±1 мин поступают со склада на конвейерную линию, по которой они в течение 2 мин транспортируются до токарного станка, время обработки на станке составляет 3.75±1 мин, после чего обработанные детали выходят из системы, поступая на промежуточный склад.

2. Допущения, сделанные в модели.

Будем считать, что заготовки не ждут погрузки на конвейер. Система представляет собой одноканальную линию обслуживания. Задача решается для обработки 100 заготовок.

3. Определения.

Транзактами в этой модели являются заготовки, а устройством является токарный станок.

4. Модельный файл.

```
*   Модуль описания
    SIMULATE                        Пример 7.1. Модель токарного цеха
*   Модуль исполнения
    GENERATE    4,1                 поступление деталей
    ADVANCE     2                   транспортировка по конвейеру
    SEIZE       SERVER              занятие станка
    ADVANCE     3.75,1              обработка детали
    RELEASE     SERVER              освобождение станка
    TERMINATE   1                   уменьшение СС на 1
*   Модуль управления
    START       100                 обработка 100 деталей
    END                             окончание процесса ИМ
```

Сеанс 1. Произведем запуск отладчика командой

gpssh ex7-1.gps tv,

после появления трехоконного отладчика в командной строке пишем

step 1, s или нажимаем функциональную клавишу *F10*.

Установим глобальное прерывание на ОБ 4 (в примере этим ОБ является ОБ ADVANCE)

BREAK 4 или *b 4*,

для продолжения процесса ИМ используем одну из команд, печатаем в командной строке

RUN или *r*, либо аналогичную команду *CONTINUE* или *c*.

Моделирование продолжается до прихода Хакт в ОБ 4, после этого экран трехоконного отладчика приобретает вид, показанный на рис. 7.4. В окне состояния видно, что Хакт 1 достиг точки прерывания в ОБ 4 и располагается перед входом в ОБ, т. е. его текущим ОБ является SEIZE.

```
========== GPSS/H SOURCE-MODE INTERACTIVE DEBUGGER =======
BLOCK CURRENT  TOTAL  SOURCE FOR MODEL EX7-1.GPS
2                1  ADVANCE    2          транспортирование заготовки
3                1  SEIZE      SERVER     занятие сервера
4                0  ADVANCE    3.75,1     обслуживание на сервере
5                0  RELEASE    SERVER     освобождение сервера
6                0  TERMINATE  1          уменьшение СС на единицу
═══════════════════════════════════════════════════════════════
S/C: ON    ABS CLOCK: 5.5406      REL CLOCK: 5.5406      TTG: 100  |
═══════════════════════════════════════════════════════════════
XACT: 1          CURBLK: 3       NEXTBLK: 4      CHAINS: CEC PC:  |

MARK-TIME: 3.5406          MOVE-TIME: 5.5406    PRIORITY: 0       |
═══════════════════════════════════════════════════════════════
Ready!
: s 1
XACT 1 POISED AT BLOCK 2. RELATIVE CLOCK: 3.5406
: b 4
: r
XACT 1 HAS REACHED BREAKPOINT AT BLOCK 4. RELATIVE CLOCK: 5.5406
:
```

Рис. 7.4. *Вид окна отладчика после исполнения первых команд*

Продолжим процесс ИМ, напечатав *c*; в диалоговом окне появляется сообщение «XACT 2 HAS REACHED BREAKPOINT AT BLOCK 4» — Хакт 2 достиг точки прерывания в ОБ 4.

Что за этот отрезок времени произошло с Хакт 1? Для выяснения напечатаем

d XACT=1,

после чего следует сообщение: «XACT 1 no longer exist» — Хакт 1 больше не существует. Отсюда следует, что Хакт 1 уже терминирован.

Установим новую глобальную точку прерывания на ОБ 3 SEIZE командой *r 3*, нажатием Enter процесс ИМ продвигается до одной из точек прерывания. Для того чтобы выяснить, какие заданы точки прерывания, печатаем в командной строке *d BRE*, что позволяет отобразить заданные точки прерывания. Установим теперь локальную точку прерывания на ОБ 6 TERMINATE, для этого напечатаем *c 6*, с помощью этой команды одновременно задаются локальные ограничения и запускается процесс ИМ. Процесс ИМ прерывается при достижении Хакт любого из ОБ с номерами 3, 4, 6. Отобразим ОБ с наложенными ограничениями командой *d BRE*. Все команды проводите в сеансе работы с отладчиком, наблюдая за изменениями на экране. Отметьте, что в появившемся списке есть только ОБ 3 и 4, ОБ 6 не включен, так как точка прерывания исполняется только при окончании действия команды *c 6*.

Удалим действие прерывания на ОБ 3 командой *unb 3* и для проверки исполнения вызовем список ограничений командой *d bre*, в спис-

ке окажется только ограничение на ОБ 4, удалим также прерывание с ОБ 4 командой *unb 4*.

Предположим, что мы хотим получать информацию об использовании устройства каждый раз, когда Хакт оканчивает процесс обслуживания; для этого введем точку прерывания на ОБ 5 RELEASE командой *r 5*. Процесс ИМ останавливается в этом месте, получаем информацию об использовании устройства, напечатав *d FAC (SERVER)* и запустив процесс ИМ командой *c*, т. е. написав две последовательные команды.

Существует более компактный способ осуществить предыдущие операции.

Вначале снимем ограничение на ОБ 5 командой *unb 5* и установим глобальное ограничение на ОБ 5 командой *AT 5*, после чего появится приглашение вида (@), затем вводим команду *d FAC(SERVER)* и, дождавшись нового приглашения (@), либо печатаем команду *end*, либо нажимаем клавишу Enter, после чего появляется обычное приглашение (:). После осуществления этих операций программа каждый раз по достижении Хакт ОБ 5 будет в окне диалога выдавать информацию об использовании устройства. Напечатав команду *c*, окончим процесс ИМ, выходим из диалогового режима одним из возможных способов, например наиболее частым путем подачи команды быстрого выхода *QQ* или *qq*.

Проведите сеанс отладчика несколько раз, вводя свои точки прерывания, для того чтобы убедиться в эффективности отладчика и приобрести практические навыки.

Сеанс 2. В целях приобретения дальнейших навыков в работе с отладчиком ответим с его помощью на некоторые вопросы по примеру 7.1.

1. Чему равняется коэффициент использования устройства, оцениваемый по каждой заготовке из первой партии, равной 10 заготовкам, в какой момент пройдет 10-я заготовка?

2. Какова загрузка устройства в момент 100 дискрет МВ?

3. Когда 34-я заготовка покинет склад и когда она попадет на устройство?

4. Когда 35-я заготовка попадет в цех и остается ли в это время 34-я заготовка в модели?

5. Когда 50-я заготовка покинет устройство?

Очевидно, что можно задать любые вопросы, касающиеся моментов процесса ИМ, но и вышеприведенные должны продемонстрировать мощность и эффективность работы отладчика. Исходя из желания подвигнуть читателя на использование компьютера при отработке диалогового режима виды окон отладчика больше отображать

Таблица 7.3. *Коэффициент использования устройства*

Хакт	$K_и$	Хакт	$K_и$
1	0.433	6	0.746
2	0.566	7	0.778
3	0.655	8	0.793
4	0.696	9	0.812
5	0.720	10	0.829

не будем, а наиболее интересная информация, появляющаяся в диалоговом окне, будет приводиться в тексте.

1. Запускаем сеанс отладчика. Для ответа на первый вопрос введем прерывание на ОБ 6 TERMINATE командой *b 6* и запускаем процесс ИМ командой *c*. Для получения информации об использовании устройства запишем команду *d FAC(SERVER)*. Точка прерывания сработала в 9.7701 дискрет МВ, коэффициент использования устройства равен 0.433.

Для запуска процесса ИМ вводим команду *c*, при этом помним, что условия прерывания не сняты. При достижении точки прерывания следующим Хакт снова получаем информацию об использовании устройства командой *d FAC(SERVER)*. Такие операции надо произвести еще 8 раз.Очевидно, что экономичней это осуществить с помощью последовательности команд: *AT – d FAC(SERVER) end* и последующей команды *c*. Каждый раз при достижении ОБ 6 будет выдаваться информация о коэффициенте использования устройства $K_и$. Данные о десяти проходах Хакт через точку прерывания приведены в табл. 7.3.

Возрастание коэффициента использования объясняется тем, что первоначально предполагалось, что устройство свободно и заготовок в системе нет. Это переходное состояние через какое-то число заготовок перейдет в установившееся состояние (подробнее см. гл. 8). Десятая заготовка выйдет из системы в 45.802 дискрет МВ. Снимем прерывание командой *unb 6*.

2. Для ответа на второй вопрос используем условия запрета. Введем команду *t CLO=100*, а затем команду *c*, с помощью которой достигаем значения времени 100.5762, $K_и$ в этот момент равен 0.922.

3. Ответить на третий вопрос можно двумя путями: первый — узнать, когда Хакт с ИН34 появится на конвейере, и для этого ввести точку прерывания на ОБ 2 и дождаться появления транзакта; второй, более эффективный, — наложить запрет на появление Хакт с ИН34 командой *t XACT=34*, после чего появляется сообщение «XACT 34 HAS NOT YET BEEN CREATED» — Хакт 34 пока еще не определен. Введя одну из команд *c* или *s* или нажав клавишу *F10*,

запускаем процесс ИМ и узнаем время появления 34-й заготовки на конвейере, равное 133.7422 дискреты МВ. К устройству 34-я заготовка подъедет в 135.7422 (133.7422 + 2) дискрет МВ.

Обратите внимание, что простота рассматриваемого МФ, в частности, непосредственное поступление заготовки со склада на конвейер, позволяет получить одинаковые значения времени как для непрерывного, так и для шагового режима. В более сложных случаях влияние этих команд будет различным!

4. Для ответа на четвертый вопрос необходимо отменить прежнее условие запрета командой *unt 34* и наложить новый запрет командой *t 35* и запустить процесс командой *c*. Исполнение остановится, когда Хакт 35 окажется перед входом в ОБ 1, это произойдет в момент 137.1532 дискрет МВ.

Для того чтобы узнать, остался ли при этом Хакт 34 в модели, введем команду *d XACT=34*; в сообщении диалогового окна говорится, что Хакт 34 еще находится в модели, так как ему запрещен вход в ОБ 3, так как Хакт ожидает обслуживания.

5. Для ответа на пятый вопрос надо отменить предыдущий запрет командой *unt XACT=35* и ввести запрет Хакт с ИН50 командой *t XACT=50* и запустить процесс ИМ командой *c*. Процесс остановится, когда Хакт с ИН50 окажется на входе в модель; так как в модели его еще нет, то появится сообщение «REQUEST QUEUED» — запрашиваемый находится в очереди. Введя команду отмены запрета *unt XACT=50,* получаем возможность оценить время, когда Хакт 50 покинет модель. Для этого вводим команду прерывания на ОБ 6 TERMINATE *b 6* и команду запуска ИМ *c*; получим, что время выхода из модели равно 202.8926 дискрет МВ. Таким образом, мы получили ответы на все поставленные вопросы и можно прерывать диалог командой *qq* и выходить в командную оболочку. Учитывая, что многие сейчас используют на своих компьютерах ОС Windows 2000 или XP, можно все действия с программой проводить в рамках Windows Commander.

§ 7.5. ПРАКТИЧЕСКИЕ СОВЕТЫ ПО РАБОТЕ С ОТЛАДЧИКОМ

Прежде чем начинать работу в режиме отладки для практических целей, а не для тренировки навыков, необходимо четко представлять специфику работы моделируемой системы. Особенно важно при разработке новой системы уяснить для себя проблемные части будущей системы, поведение которых не до конца очевидно исследователю. Именно в целях практического применения диалогового режима можно дать несколько практических советов.

• Заранее определите количество и расположение критических точек модели.

• Установите режим прерывания в этих точках (break — b) и непременно пользуйтесь на каждом шаге процедурой отображения (display — d). В непосредственной близости от точки прерывания используйте шаговый режим (step — s), чтобы не пропустить каких-либо особенностей поведения модели. Удаляйте точки прерывания сразу после того, как задача исследования достигнута. Подобное поведение позволяет выявить возможные ошибки и верифицировать логику модели.

• Избегайте длительных диалоговых режимов, чтобы не утомляться и не наделать новых ошибок при длительной отладке. Выйдя из режима диалога, проанализируйте все ваши действия в процессе отладки и продумайте новые изменения, если они необходимы.

• Помните, что выявленная ошибка порой дает гораздо больше информации, позволяющей исключать подобные ошибки в дальнейшем. Появление второй аналогичной ошибки свидетельствует о том, что либо ваши усилия были не эффективны, либо заданы неверные начальные условия или режимы использования.

• Для выявления влияния поведения транзактов на возможные ошибки используйте условия запрета (trap — t), чтобы уточнить логику МФ.

• В случае невозможности определить причины ошибки, попробуйте применять стрессовые методы, например увеличить размер шага, чтобы локализовать место происхождения ошибки. Порой помогает изменение стратегии контроля, но все это приходит с опытом.

§ 7.6. УПРАЖНЕНИЯ

В настоящем параграфе нет раздела проверки ошибок, и это сделано сознательно, потому что большинство примеров, разобранных в пп. 5.5.1 и 6.5.1 (кроме примеров, решаемых без помощи компьютера), должны были быть разобраны с использованием диалогового режима (тестовый режим сеанса отладчика). Если это не было сделано, то на базе сведений, изложенных в гл. 7, проведите в тестовом диалоговом режиме разбор примеров из указанных выше пунктов.

Приводимые ниже упражнения необходимо разобрать с использованием сеанса отладчика, что должно способствовать не только закреплению материала гл. 7 и уяснению основных команд, но и пониманию логики внутренней деятельности программы в целом. (Напоминаем, что номера упражнений идут нарастающим итогом для удобства пользования ответами в прил. 5.)

24. Рассмотрите МФ:

```
            SIMULATE
BLOCK1      GENERATE    50,25
            TERMINATE   1
BLOCK3      GENERATE    10,5
            TERMINATE   1
            START       5
            END
```

Используя команды *t[rap] scan, run, d[isplay] cec fec*, ответьте на вопросы (при этом помните, что, когда списки СТС и СБС пусты, программа не реагирует на команду их показа и не выдает никакого сообщения).

Вопросы:

A. Каковы ИН Хакт, выходящих из генераторов?

B. В каких списках находятся транзакты сразу после инициализации?

C. Каков порядок расположения транзактов в списке?

25. Рассмотрите МФ:

```
            SIMULATE
BLOCK1      GENERATE    40,15
BLOCK2      ADVANCE     25,10
BLOCK3      TERMINATE   1
BLOCK4      GENERATE    30,20
BLOCK5      ADVANCE     20,5
BLOCK6      TERMINATE   1
            START       5
            END
```

Используя последовательно команды *t clock 50 — r; set tv off — d cec fec; set tv on — t clock 75 — r; set tv off — d cec fec*, внимательно проанализируйте информацию, появляющуюся на экране отладчика.

Вопросы:

A. Определите реакцию отладчика на две последовательно записанные команды *trap* без промежуточной команды *run*.

B. Определите реакцию программы на серию команд *t clock 75 — r — t clock 75*.

C. Оцените совпадение и различие команд *t scan* и *t clock*.

26. Рассмотрите МФ для моделирования процесса обработки заготовок в механическом цеху, представленный ниже, все данные приведены непосредственно в МФ. На этом примере изучите, как работает ФИС и ИС транзактов (для выполнения этого упражнения необходимо прочесть п. 5.4.3*). Используйте для этих целей сеанс отладчика. Обратите внимание, что время поступления меньше времени

обслуживания, такие системы называются системами с накоплением и на практике не используются. Однако примем эти условия для уяснения работы ФИС модели и ИС.

```
            SIMULATE
BLOCK1      GENERATE    15,,,2      заготовки поступают каждые
                                    15 мин
BLOCK2      ADVANCE     10          время транспортировки
 BLOCK3       SEIZE          DRILL  занятие сверлильного станка
 BLOCK4       ADVANCE        30     время сверловки
 BLOCK5       RELEASE        DRILL  освобождение сверлильного
                                    станка
 BLOCK6       TERMINATE      1      готовая деталь покидает
                                    модель
              START          2      количество обработанных
                                    деталей
 END                                окончание процесса ИМ
```

Задание: Используя последовательность 9 серий нижеприведенных команд, изучите изменения, происходящие на экране отладчика.

A. *t clock 40 — r; t system — r.*

B. *d fac.*

C. *d cec.*

D. *t next — r — d cec*

E–F. s — *d cec*

J–I. *s.*

K. *r – r.*

27. Создайте нижеприведенный МФ:

```
            SIMULATE
            GENERATE    50,20
BLOCK2      QUEUE       WAITAREA
BLOCK3      SEIZE       SPRAYER
            DEPART      WAITAREA
            ADVANCE     48,15
BLOCK6      RELEASE     SPRAYER
            TERMINATE   1
            START       100
            END
```

Вопросы:

A. Используйте команду *at* для создания списка команд *d blo6 — r.* Какое наибольшее число Хакт ожидает обслуживания при исполнении блока 6 RELEASE?

B. В какие моменты МВ отмечено появление максимума ожидающих транзактов?

28. Используя МФ примера 6.2 (модель инструментальной кладовой), ответьте в тестовом режиме на ряд вопросов.

Вопросы:

A. Отметьте время первых пяти появлений механиков первого типа.

B. Отметьте время появления первых пяти механиков любого типа

C. Отметьте время окончания обслуживания пяти механиков любого типа.

D. Найдите самое раннее время, когда точно один механик любого типа ожидает обслуживания при занятости кладовщика.

E. Найдите самое раннее время, когда два механика (не зависимо от типа) ожидают обслуживания при занятости кладовщика.

F. Найдите самое раннее время, когда по одному механику каждого типа ожидают обслуживания.

J. Найдите самое раннее время, когда при ДО SPT механик с большим приоритетом находится раньше механика с меньшим приоритетом.

Глава 8
СТАТИСТИЧЕСКИЕ ВОЗМОЖНОСТИ ЯЗЫКА ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ GPSS/H

§ 8.1. ГЕНЕРАЦИЯ СЛУЧАЙНЫХ ПЕРЕМЕННЫХ С ЗАДАННОЙ ФУНКЦИЕЙ РАСПРЕДЕЛЕНИЯ

Имитационное моделирование по своей сути обладает рядом недостатков, которые присущи всем ЯИМ:

1) возможность получения только средних статистических оценок;

2) сложность получения оптимального варианта моделируемой системы.

Именно эти недостатки представляют для нас интерес (общее сравнение аналитического и компьютерного моделирования см. § 1.4) при рассмотрении статистических аспектов оценки результатов ИМ. Поэтому в настоящей главе уделим внимание только вопросам повышения точности оценок и изучению путей приближения к оптимальному (или, точнее, к рациональному) варианту построения исследуемой системы.

Повысить точность оценок можно за счет:

— выбора параметров входных потоков транзактов и потоков их обслуживания, адекватных функциям распределения случайных величин в реальной системе;

— планирования проведения вычислительного эксперимента;

— уменьшения дисперсии оценок за счет специальных мероприятий.

Приближение к рациональному варианту построения системы можно осуществить при выполнении следующих процедур:

— выяснении установившегося значения показателей за счет исключения переходного периода;

— выборе наилучшей альтернативы в паретовском множестве.

Все эти вопросы будут рассмотрены в данной главе.

8.1.1. Обоснование выбора функции распределения случайных величин

Главной задачей в процессе ИМ является получение последовательности независимых и одинаково распределенных случайных величин. Первое, что при этом интересует исследователя, — какому закону распределения подчиняются случайные потоки заявок и обслуживания в проектируемой системе.

1. Естественно, что вначале пытаются использовать данные по функционированию аналогичных или похожих систем, накопленных за какой-то период времени. Трудности, которые подстерегают на этом пути, очевидны:

— количество случайных чисел для моделирования даже не очень сложных систем весьма велико, поэтому статистические данные о функционировании систем-аналогов надо собирать в течение большого периода времени;

— собранные данные не будут достоверными, так как и в процессах функционирования, и в идеологии построения систем, и в параметрах законов распределения за период наблюдения произошли значительные, если не революционные, изменения.

Даже если изменений не произошло и нас не интересуют возможные изменения в будущем, при моделировании придется опираться только на одну выборку данных из возможного множества.

2. Следующая возможность для получения исходных данных для ИМ — аппроксимация статистических данных, полученных при проведении процесса пилотных вычислительных экспериментов. Для этого можно использовать полученную гистограмму, не забывая при этом, что никогда не будет учитываться величина большая или меньшая, чем размах гистограммы. Кроме того, если с большой степенью достоверности известно, что функции распределения потоков случайных величин непрерывны, то гистограмма дает представление о дискретных значениях плотности вероятности. Существует много статистических методов по сглаживанию гистограмм и получению эмпирических непрерывных функций плотности вероятности, которые не рассматриваются в пособии.

3. Используется обобщенный метод «бутстреп» по улучшению статистических данных (см. §. 8.2), разработанный Э. Дадевичем [15].

4. Наиболее часто основой для целей моделирования реальной системы выбираются теоретические функции распределения случайных величин. При этом полагают, что при определенных допущениях частные распределения хорошо аппроксимируются известными теоретическими распределениями. Особенно хорош этот метод при оценке чувствительности моделируемой системы к виду функций распределения.

Общая идея принципов моделирования случайных чисел, в том числе и двухэтапный процесс получения случайных чисел с заданной функцией распределения, рассмотрены в п. 3.3.2. В настоящем параграфе рассмотрим практические методы получения случайных чисел и функций для некоторых наиболее применяемых распределений.

Можно дать несколько рекомендаций.

1. При исследованиях степени приближения системы к установившемуся режиму следует пользоваться теоретическими распределениями, которые также полезны, если практические данные хорошо совпадают с теоретическими функциями распределения.

2. Для практических целей можно пользоваться данными, полученными в результате пилотного прогона верифицированной модели.

3. Старайтесь непосредственно не пользоваться данными о функционировании систем-аналогов или выборочными данными, полученными из гистограмм, а применяйте обобщенный метод «бутстреп».

8.1.2. Генерация БСВ в GPSS/H

В GPSS/H имеется ряд ГСЧ, обозначаемых при необходимости в МФ как RNj (random number — j). Каждый из них может создать более двух миллионов равномерно распределенных случайных чисел в открытом интервале (0, 1). По умолчанию стартовая позиция каждого ГСЧ не одинакова, что позволяет получать с разных генераторов не коррелированные потоки БСВ. По идее, эти ГСЧ производят псевдослучайные числа, но так как апериодичность их весьма велика и составляет 56 двоичных значений, то для практических задач БСВ принято считать действительно случайными (см. п. 3.4.2). Каждая БСВ имеет 16 десятичных разрядов. Рассмотрим способ использования равномерно распределенных БСВ для получения времени поступления транзактов — ОБ GENERATE, времени обслуживания транзактов в устройствах и памятях — ОБ ADVANCE и вероятности перехода в следующий ОБ в статистическом варианте — ОБ TRANSFER. Для первых двух случаев используется выражение

$$T_j = (\mathrm{A} - \mathrm{B}) + y \cdot 2\mathrm{B}. \qquad (8.1)$$

Поскольку значения $y = \mathrm{RN}$ (или ГСЧ) лежат в интервале $0.0 < \mathrm{RN} < 1.0$, то меньшее значение времени T_j из (8.1) равняется A − B, а большее A + B. Если B = 0, то получаем детерминированное значение времени и умножение на значение RN не производится.

В качестве примера приведем первые восемь 6-разрядных БСВ (не 16-разрядных, как в действительности) для 1-, 2- и 3-го ГСЧ (табл. 8.1).

Предположим, что ОБ GENERATE или ОБ ADVANCE имеют значения операндов A и B 420, 360 соответственно, первое значение ГСЧ1 в табл. 8.1 равно 0.270295, тогда $T_j = (420 - 360) + 0.270295 \cdot 720 = 254.6$, второе значение $T_j = 60 + 0.499353 \cdot 720 = 419.7$ и т. д.

Таблица 8.1. *Первые 10 чисел трех ГСЧ*

ГСЧ	Показатель							
ГСЧ1	.270295	.499353	.739770	.204092	.120667	.768434	.440394	.859458
ГСЧ2	.985170	.445582	.675116	.454705	.834154	.559414	.634527	.617560
ГСЧ3	.214159	.791699	.792091	.132872	.377848	.943256	.271380	.393878

Если рассматривается несколько входных потоков, как в примере 6.2, то второе БСВ используется для получения времени прихода механиков второго типа, тогда время прибытия механиков второго типа, имеющего операнды A и B 360 и 240 соответственно:

$$T_j = (360 - 240) + 0.499353 \cdot 480 = 359.7.$$

Второе время прихода механиков первого типа $T_j = 60 + 0.739770 \times$ $\times$ 720 = 592.6 и т. д. Затем, поскольку транзакт, представляющий механиков первого типа, попадает на обслуживание, четвертое случайное число из табл. 8.1 используется для моделирования времени обслуживания. Этот процесс хорошо прослеживается в тестовом режиме.

Времена прибытия механиков обоих типов будут складываться, давая непрерывное изменение. Так, для механиков первого типа за два прихода оно равно 254.6 + 592.6 = 847.2, что отмечается на оси MB с помощью модификатора времени.

Следует отметить, что если не заданы специальные условия получения случайных чисел с помощью ОУ RMULT, то всегда для времен прихода и обслуживания используется только ГСЧ1!

Существуют еще два случая применения БСВ, взятых с ГСЧ1. Оба они используются для представления использования ОБ TRANSFER в статистическом варианте и варианте PICK, который используется для случайного выбора одного из нескольких ОБ (этот вариант в пособии не используется).

Рассмотрим ОБ TRANSFER в статистическом варианте:

TRANSFER .250,BLOCK1,BLOCK2. БСВ, взятые с ГСЧ1, используются для выбора пути движения транзакта; так, если три первых разряда ГСЧ меньше числа 0.250, то Хакт движется в BLOCK2, в противном случае — в BLOCK1.

Таким образом, в явных или *транспарентных* случаях, когда это специально не оговорено, для представления времен прихода и обслуживания, а также для логического выбора используются всегда БСВ, взятые с ГСЧ1.

8.1.3. Получение случайных чисел с заданными функциями распределения

Предположим, что для исследуемых случайных параметров потоков выбрано теоретическое распределение, тогда необходимо генерировать независимые случайные переменные, каждая из которых имеет такую же функцию распределения. Если функция распределения $F_x(.)$ — непрерывная возрастающая функция, то для нее всегда найдется обратная или инверсная функция $F_x^{-1}(.)$, называемая также инверсной функцией распределения. Метод обратных функций широко применяется для генерации случайных переменных $X_1, X_2, \ldots$.

Теорема 8.1. Предположим, что $F_x(.)$ — функция распределения, которая имеет обратную функцию $F_x^{-1}(.)$, а U — равномерно распределенная БСВ в интервале (0, 1). Тогда $F_x(U)$ имеет функцию распределения $F_x(.)$.

Доказательство: Случайная переменная $F_x^{-1}(U)$ имеет функцию распределения

$$P[F_x^{-1}(U) \le x] = P[U \le F_x(x)] = F_x(x). \qquad (8.2)$$

Первое равенство в (8.2) является следствием того, что обратная функция распределения в U не будет больше x, если и только если U не будет больше значения функции распределения в x. Второе равенство в (8.2) следует из того, что вероятность появления БСВ не может быть больше чем $z = z$ для всех значений z между 0 и 1, а $F_x(x)$ принимает значения также между 0 и 1.

Следствие. Если $U_1, U_2, \ldots$ — независимые равномерно распределенные БСВ в интервале (0, 1), тогда

$$X_1 = F_x^{-1}(U_1), \;\; X_2 = F_x^{-1}(U_2), \ldots \qquad (8.3)$$

являются независимыми случайными переменными, каждая из которых имеет функцию распределения $F_x(.)$.

Пользуясь этой теоремой, подробно рассмотрим, как получаются случайные числа, имеющие экспоненциальную и нормальную функцию распределения.

Экспоненциальная функция распределения

Функция плотности вероятности для экспоненты с параметром потока (или средним значением) λ имеет вид $f(x) = 1/\lambda e^{-x/\lambda}$, и следовательно, функция распределения запишется как

$$F(x) = \int_{-\infty}^{x} f(z)\mathrm{d}z = 1 - e^{-x/\lambda}. \qquad (8.4)$$

Для того чтобы найти обратную функцию распределения для любого $y = \mathrm{RN} = U$ (или ГСЧ) между 0 и 1, воспользуемся решением уравнения $y = F(x)$ для x, выраженных через y. Для решения уравнения $y = 1 - e^{-x/\lambda}$ воспользуемся тем, что

$$e^{-x/\lambda} = 1 - y;$$

$$-x/\lambda = \ln(1 - y);$$

$$x = -\lambda \ln(1 - y). \qquad (8.5)$$

Следовательно, для y в интервале (0, 1)

$$F^{-1}(y) = -\lambda \ln(1 - y). \qquad (8.6)$$

Воспользовавшись следствием теоремы 8.1, заключаем, что если U_i — независимые равномерно распределенные БСВ, тогда $X_i = -\lambda \ln(1 - U_i)$ являются случайными числами с экспоненциальной функцией распределения и параметром потока (средним значением) λ.

Иногда в литературе вместо разности $(1 - U_i)$ пользуются непосредственно значением U_i. Этого делать нельзя по ряду соображений:

— довольно часто (см. § 8.3) для увеличения точности моделирования вводят положительную корреляцию между случайными переменными. Если же использовать U_i, то корреляция может стать отрицательной и привести к снижению точности моделирования;

— сокращение до U_i может привести к ошибкам распознавания машинных кодов;

— при использовании инверсного метода усечение до U_i может привести к неверным вычислениям, производимым пользователем.

Пользователи GPSS/H застрахованы от этих неприятностей, так как все рассматриваемые функции распределения находятся в программе и пользователь задает только входные параметры. Так, задание экспоненциального распределения осуществляется следующим образом: вместо операнда A в ОБ GENERATE и ОБ ADVANCE пишется RVEXPO(RNj, λ), где RVEXPO (Random Variate EXPOnential) — название функции распределения; RNj — номер ГСЧ, с которого берется БСВ, а λ — параметр потока или среднее значение. Операнд B по умолчанию равен нулю. Экспоненциальное распределение имеет большую степень изменчивости; это объясняется тем, что дисперсия равна квадрату среднего значения, а показатель экспоненты асимптотически стремится к бесконечности при приближении БСВ к 1. Чем больше значение показателя, тем меньше вероятность его появления (табл. 8.2).

При желании более развернутую информацию об экспоненциальном распределении можно почерпнуть из специальной литературы.

Таблица 8.2. Значение показателя и вероятность появления

Показатель	Вероятность	Показатель	Вероятность
0	1	2.0	0.135
0.1	0.904	3.0	0.049
0.5	0.606	5.0	0.006
1.0	0.367	8.0	0.0002

Функция нормального распределения

Нормальное распределение любимо большинством инженеров за его наглядность и понятные параметры, кроме того, со студенческой скамьи они запомнили, что с помощью нормального распределения можно описать большинство непрерывных случайных величин. Автор не разделяет этой эйфории, считая, что нормальное распределение — это мираж, не достижимый в большинстве практических случаев, однако дискуссия на эту тему выходит за рамки пособия. Поэтому вернемся на проторенную дорогу и выясним, как создаются случайные числа, подчиняющиеся нормальному распределению.

Пусть необходимо получить независимые случайные переменные X_i, имеющие нормальное распределение с параметрами математического ожидания μ и дисперсией σ^2. Такое распределение чаще всего обозначается как $N(\mu, \sigma^2)$. Случайная переменная X имеет нормальное распределение $N(\mu, \sigma^2)$, если его функция распределения имеет вид

$$F_x(x) = \int_{-\infty}^{x} fx(z)\mathrm{d}z, \tag{8.7}$$

где

$$f_x(z) = 1/\sqrt{2\pi}\sigma e^{-0.5(z-\mu)^2/\sigma^2}. \tag{8.8}$$

Известно, что не существует простого конечного выражения для $F_x(.)$.

Существуют приближения, полученные разными авторами, одно из которых [24] — при $\mu = 0$ и $\sigma^2 = 1$, что приводит к стандартному нормальному распределению, позволяет получить выражение

$$F_x(x) = x(4.4 - x)/10 + 0.5, \tag{8.9}$$

дающее ошибку не более 0.005 для значений x в интервале от 0 до 2.2.

Поскольку мы имеем дело с симметричным распределением, а, по определению, время не может быть отрицательным, то при получе-

нии случайных переменных это следует иметь в виду. Для этого необходимо выполнять два условия.

1. Ожидаемое время по крайней мере в три раза должно превышать стандартное отклонение (знаменитое правило «трех сигм»). Как известно, вероятность попадания в такой интервал равна примерно 99.7 %, следовательно, не более 0.15 % попадет на отрицательную часть.

2. Преобразовать значения, попавшие в отрицательную часть, в положительные значения. Применение абсолютных значений приводит к очень небольшим ошибкам, порядок которых определен выше.

В GPSS/H используется встроенная функция ABS (ABSolute-value function), которая конвертирует отрицательные значения в положительные, оставляя неизменными положительные значения. Появление записи о нормальности функции распределения автоматически подключает функцию ABS.

Нормальное распределение задается записью на месте операнда А ОБ GENERATE, ADVANCE выражения RVNORM (RNj, μ, σ) (в скобках указаны: на первом месте — номер ГСЧ, с которого берутся БСВ; на втором и третьем — параметры нормального распределения). Операнд В по умолчанию равен 0. Пример записи:

ADVANCE RVNORM(1,3,0.5),,4,,5.

что читается как «поступает транзакт, имеющий нормальное распределение с параметрами 3, 0.5; приходящий в 4-ю дискрету МВ и имеющий приоритет 5».

Язык ИМ GPSS/H имеет целый ряд встроенных функций распределения, в профессиональной версии ЯИМ их более 30. В студенческой версии возможно использовать:

— встроенные математические функции COS-ACOS — косинус-арккосинус; SIN-ASIN — синус-арксинус; TAN-ATAN — тангенс-арктангес; LOG — логарифм; SQRT — корень квадратный; ABS — абсолютное значение. Формат задания этих функций прост, например:

ABS (выражение), SQRT (число), SIN (угол или выражение) и т. д.;

— функции распределения RVBIN — дискретное биномиальное распределение; RVERL — распределение Эрланга; RVEXPO — экспоненциальное распределение; RVGAMA — гамма-распределение; RVGEO — геометрическое распределение; RVLNOR — логнормальное распределение; RVNORM — нормальное распределение; RVPSSN — распределение Пуассона; RVPT — распределение Пирсона; RVWEIB — распределение Вейбула. Рассмотрим без доказательств распределения Пуассона и Эрланга, имеющие важное значение для СМО.

Дискретное распределение Пуассона

Экспоненциальное распределение позволяет оценить время поступления или обслуживания и является непрерывным по определению. Однако можно рассматривать темп или скорость поступления или обслуживания. Так, если время поступления равно 20 мин, то темп поступления равен 3 ед./ч. Обычно пользуются временем поступления, поскольку оно позволяет прогнозировать будущее поведение, однако бывают ситуации, когда представляет интерес именно темп поступления. Распределение темпа поступления дискретно и целочисленно по определению. В непрерывных распределениях имеют дело с функцией плотности вероятности, которую мы обозначали $f(x)$. В дискретных распределениях обычно пользуются понятием частости $p(x)$. Не трудно показать [1], что если время поступления экспоненциально, то темп поступления имеет распределение Пуассона. Показатель распределения Пуассона μ обратно пропорционален параметру потока λ при экспоненциальном распределении, т. е. $\mu = 1/\lambda$. Так, если среднее значение времени поступления равно 5 мин, то темп равен 0.2 поступления/мин или 12 поступлениям/ч. В подтверждение пуассоновского распределения темпа поступления говорят следующие утверждения.

1. Вероятность поступления транзактов в течение небольшого интервала времени пропорциональна длине этого интервала.

2. Вероятность одновременного поступления двух или более транзактов пренебрежимо мала.

3. Времена поступления независимы одно от другого [1].

Задание случайных переменных, имеющих распределение Пуассона, производится аналогично другим функциям распределения:

GENERATE RVPSSN (RNj, μ)

Далее в тексте это распределение не используется, при получении входных данных в виде темпа поступления их следует преобразовать во времена поступления, так как контролируемым параметром при моделировании является модельное время.

Распределение Эрланга

Иногда при моделировании необходимо учитывать не только время обслуживания, но и время простоя или вынужденной остановки устройства или памяти. В этом случае приходится иметь дело с несколькими выборками из различных экспоненциальных распределений. Напомним, что такое распределение называется распределением Эрланга k-го порядка, а экспоненциальное распределение является его частным случаем при $k = 1$.

В GPSS/H учет двух экспоненциальных распределений реализуется либо написанием подряд нескольких ОБ ADVANCE, либо

включением арифметического выражения в поле операнда A, например:

ADVANCE RVEXPO (3,0.45) + RVEXPO (3,0.15)

Использование распределения Эрланга будет проиллюстрировано в примере 8.2 после введения понятий о представлении функций.

Рассмотрим пример использования случайных переменных, имеющих распределение, отличное от равномерного.

Пример 8.1. Модель приемного покоя

1. Постановка задачи.

Темп прихода пациентов подчиняется распределению Пуассона, среднее время их появления равно 30 мин. Затем они подходят в регистрацию, 45 % из них в состоянии ожидать приема у врача, обозначим их CW (can wait), а 55 % требуется немедленная помощь, их обозначим NIA (need immediate attention).

Регистратор тратит какое-то время на заполнение карты больного и выяснение обстоятельств, время распределено по экспоненте со средним значением 10 мин.

В приемном покое принимают два доктора, пациенты NIA имеют больший приоритет по сравнению с CW. На осмотр пациентов NIA доктор тратит 45 мин (экспоненциально распределенных), 75 % осмотренных вынуждены сдавать анализы и ждать результата исследований, а 25 % уходят домой. После получения анализов (экспоненциальное распределение со средним 30 мин) пациенты NIA снова должны пройти к любому доктору, но уже имея более низкий приоритет по сравнению с пациентами CW, время вторичного ожидания распределено экспоненциально со средним 15 мин, после чего пациенты уходят. Пациенты CW проходят к доктору один раз, время их осмотра распределено по экспоненте со средним значением 30 мин, и они имеют приоритет меньший, чем у пациентов NIA, идущих на прием первый раз, но больший, чем у тех же пациентов, приходящих второй раз с результатами анализов.

Проведите моделирование за период 240 ч, полагая, что за это время не произойдет изменения каких-либо параметров, хотя, конечно, такие предположения весьма идеалистичны (см. § 8.4). Создайте очереди во всех четырех возможных точках ожидания и оцените полученные результаты.

2. Допущения, сделанные в модели.

Логика предлагаемой модели весьма прямолинейна, транзакты (пациенты) поступают с одного генератора с приоритетом 10, проходят через классификацию, а затем разделяются на два потока, у пациентов NIA приоритет возрастает до 15, а затем этот поток делится на два, идущих на сдачу анализов и уходящих домой, при этом у первых приоритет снижается до 5 и они вновь идут к доктору, после чего покидают систему. Пациенты CW не меняют своего приоритета за все время моделирования.

3. Таблица обозначений (табл. 8.3).

Временная дискрета: 1 мин.

Таблица 8.3

Объект GPSS/H	Объект в системе
Транзакты	
Фрагмент 1 модели	Приход пациентов на регистрацию
Фрагмент 2 модели	Пациенты
Фрагмент 3 модели	Пациенты
Фрагмент 4 модели	Транзакт управления
Устройства	
NURSE	Регистратор
Очереди	
CWLINE	Очередь пациентов CW
NIALINE1	Очередь пациентов NIA
NIALINE2	Вторичная очередь пациентов NIA
NURSE	Очередь к регистратору
Памяти	
DOCTORS	Доктора, ведущие прием

4. Модельный файл.

```
*       Модуль описания
SIMULATE                                  Пример 8.1. Модель приемного
                                          покоя
*                                         Временная дискрета: 1 мин
DOCTORS     STORAGE      2                доктора в приемном покое
*  Модуль исполнения
*                Фрагмент 1               приход пациентов
GENERATE    RVEXPO(1,30),,,,10            приход пациентов с приоритетом 10
   QUEUE     NURSE                        создание очереди к регистратору
   SEIZE     NURSE                        начало регистрации
   DEPART    NURSE                        выход из очереди к регистратору
   ADVANCE   RVEXPO(1,10)                 процесс регистрации
   RELEASE   NURSE                        освобождение регистратора
   TRANSFER. 450,,CANWAIT                 45 %, могущих ждать
*                Фрагмент 2               пациенты, не могущие ждать
   NIAPATH   PRIORITY     15              повышение приоритета до 15
   QUEUE     NIALINE1                     создание очереди к докторам
   ENTER     DOCTORS                      заход на осмотр
   DEPART    NIALINE1                     выход из очереди к докторам
   ADVANCE   RVEXPO(1,45)                 время первичного осмотра
   LEAVE     DOCTORS                      уход от доктора
   TRANSFER  .250,,NIAHOME                25 %, уходящих домой
   PRIORITY  5                            снижение приоритета до 5
   ADVANCE   RVEXPO(1,30)                 проведение лабораторных анализов
```

```
      QUEUE    NIALINE2                 создание очереди
                                        на вторичный осмотр
      ENTER    DOCTORS                  начало вторичного осмотра
               DEPART   NIALINE2        выход из очереди
                                        вторичного осмотра
               ADVANCE  RVEXPO(1,15)    проведение вторичного
                                        осмотра
               LEAVE    DOCTORS         уход от доктора
NIAHOME        TERMINATE 0              уход из системы
 *             Фрагмент 3               пациенты, могущие ждать
CANWAIT        QUEUE    CWLINE1         создание очереди
               ENTER    DOCTORS         начало осмотра
               DEPART   CWLINE1         уход из очереди
               ADVANCE  RVEXPO(1,30)    осмотр у доктора
               LEAVE    DOCTORS         освобождение доктора
CWHOME         TERMINATE 0              уход из системы
*              Фрагмент 4               управление процессом ИМ
               GENERATE 14400           Хакт управления за 240 ч
               TERMINATE 1              уменьшение CC на1,
                                        окончание движения Хакт
*              Модуль управления
               START    1               установка CC на 1,
                                        начало движения Хакт
               END                      окончание процесса ИМ
```

5. Итоговый отчет.

Краткие итоги ИМ сведены в табл. 8.4.

6. Обсуждение.

Отметим следующие особенности:

— в процессе ИМ использовались случайные переменные, полученные только с ГСЧ1 для всех ОБ — GENERATE, ADVANCE, TRANSFER;

Таблица 8.4

Наименование	Название	Использование	Число входов	Содержание	
				максимальное	среднее
Устройство	NURSE	0.302	453	–	–
Памяти	DOCTORS	0.705	628	–	–
Очереди	NURSE	–	453	5	0.11
	NIALINE1	–	251	4	0.282
	NIALINE2	–	176	17	1.726
	CWLINE1	–	202	8	0.814

Примечание. Полные данные см. в листинге программы.

— приоритет пациентов NIA менялся дважды, вначале повышаясь до 15, а перед вторым посещением доктора снижаясь до 5, причем снижение производилось до ухода транзактов в СБС;

— несмотря на то что коэффициент занятости докторов равен 0.75, среднее время ожидания довольно большое. У пациентов CW время ожидания в среднем равно 58 мин. Таким образом, индивидуальное время ожидания имеет большой разброс.

8.1.4. Получение непрерывных и дискретных функций

До настоящего момента мы имели дело только со случайными переменными в виде БСВ, непосредственно получаемых с ГСЧ, или со случайными переменными с заданной функцией распределения, основой для которых также являются БСВ. Однако входные сигналы в реальной системе могут быть случайными функциями или случайными векторами, поэтому имеет смысл рассмотреть способы задания внешних функций.

D-функция — выборка из дискретного распределения

В качестве иллюстрации рассмотрим работу сверлильного станка. Время на обработку одной детали будет зависеть от числа отверстий, их диаметра, вида материала детали. В табл. 8.5 приведена зависимость времени сверления от числа отверстий в детали и относительные частоты проведения каждой операции.

На рис. 8.1 представлены различные распределения, в том числе, построенное по данным табл. 8.5; масштаб вертикальных шкал различен.

Таблица 8.5. *Зависимость времени сверления*

Показатель	Данные				
Число отверстий	1	2	3	4	5
Время сверления	4.0	7.5	10.5	13.5	16.5
Частота операций	0.20	0.35	0.15	0.10	0.20
Накопленная частота	0.20	0.55	0.70	0.80	1.0

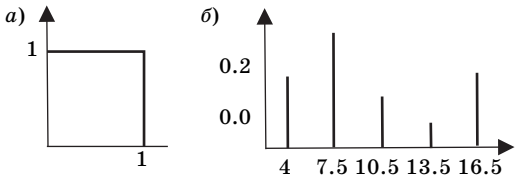


Рис. 8.1. *Вид равномерного (а) и дискретного (б) распределений*

В отличие от ряда теоретических распределений, встроенных в программу, дискретные функции типа представленных на рис. 8.1, *б* необходимо задавать в модуле определений самим пользователем. Задание функции осуществляется с помощью ОУ FUNCTION, формат которого содержит пять обязательных составляющих и имеет вид

< label FUNCTION argument,type_n

x_1,y_1/.../x_n,y_n >

label — обязательная метка, выбираемая пользователем для идентификации;

FUNCTION — название ОУ в операционном поле;

argument — операнд А содержит выбираемый пользователем номер ГСЧ (RNj);

type_n — операнд В содержит два символа: первый — тип функции (в GPSS/H возможно задание пяти типов функций, подробнее см. п. 5.1.2), второй — целое число точек задаваемой функции; x_i, y_i — пары точек функции: x_i — значение аргумента функции; y_i — значение К2), принимаемое функцией. Пары значений функции записываются во второй строке, начиная с первой колонки, внутри пары отделяются запятой без пробела, пары друг от друга отделяются слэшем без пробела. Появление пробела воспринимается программой как начало комментариев. В начале первой пары чисел (слева) и в конце последней пары чисел (справа) не ставится никаких символов. Длина записи пар чисел не должна заходить за 72-ю колонку редактора; если число точек велико, то занимается третья строка с начала какой-то пары и т. д. Для задания функции может использоваться более одного описания функции, тогда они пишутся друг под другом. В качестве примера запишем функцию табл. 8.5 и рис. 8.1:

```
DRILTIME FUNCTION RN7,D5
.2,4/.55,7.5/.7,10.5/.8,13.5/1,16.5
```

или в виде

```
.2,4/.55,7.5          первое описание
.7,10.5               второе описание
.8,13.5/1,16.5        третье описание
```

Отметим, что в отличие от задания теоретических функций типа RV..., где номер ГСЧ задается только цифрой в скобках, при задании внешних функций номер ГСЧ задается сочетанием RNj. Порядок получения значений функции таков: вначале ГСЧ вырабатывает БСВ в интервале (0,1), полученное значение сравнивается с накопленной частотой; если значение меньше 0.2, то выбирается величина 4.0, если в интервале $0.2 < \text{БСВ} \le 0.55$, то выбирается величина 7.5 и т. д. Необходимо отметить, что левая граница любого интервала открытая, а правая — закрытая для всех, кроме последнего, так как граница 1.0 является открытой (рис. 8.2).

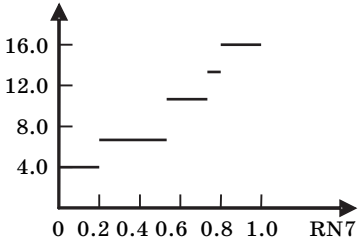


Рис. 8.2. *Графическое представление D-функции*

Естественно, что значения функции будут появляться не так, как это задано в табл. 8.5, а так, как это будет предписано появлением БСВ. Часто для экономии машинного времени табл. 8.5 можно перестроить в порядке убывания относительных частот, тогда сверление двух отверстий будет наиболее частым событием.

Оператор управления FUNCTION исполняется после того, как успешно завершится компиляция МФ, а размещается он в модуле описания до начала движения транзактов, так как программа вначале исполняет все ОУ, стоящие в МФ выше ОУ START. Задание функции в операнде А ОБ GENERATE и ADVANCE проиллюстрируем примером рассматриваемой D-функции:

ADVANCE FN(DRILTIME)

Запись в операнде А—FN(DRILTIME) говорит о том, что используется функция, заданная в модуле описания с именем, приводимым в скобках, соответственно операнд В по умолчанию равен нулю.

Дискретной функцией можно интерпретировать разные события, например число используемых касс в торговом зале, количество отказов оборудования, число отсутствующих по разным причинам сотрудников и т. д. Дискретную функцию также можно использовать в ОБ TRANSFER в вариантах безусловной адресации или в статистическом виде. Другие, более сложные варианты использования функции, например для преобразования одной переменной в другую, определения емкости памяти, содержания очереди и т. п., в пособии не рассматриваются, хотя опытный пользователь может использовать любые варианты применения функций.

C-функция — выборка из непрерывного распределения

Случайная переменная может быть непрерывной, и при этом нельзя подобрать соответствующего теоретического распределения, так как вид непрерывной кривой может представлять собой сплайн-распределение, когда соединяются несколько разных непрерывных распределений. Запись непрерывной функции отличается от дискретной появлением литеры С, а графическое воплощение представляет собой набор прямых наклонных отрезков, соединяемых между собой в непрерывную ломаную линию.

Представим, что описывается время обслуживания клиента банковским служащим (табл. 8.6).

Таблица 8.6. *Время обслуживания*

Показатель	Данные						
Время обслуживания, с	≤ 60	120	180	240	300	360	≥ 360
Относительная частота	0.0	0.07	0.16	0.50	0.19	0.08	0.0
Накопленная частота	0.0	0.07	0.23	0.73	0.92	1.0	–

У каждого интервала после первого левый конец открыт, а правый закрыт, у последнего интервала открыты оба конца. На основании данных таблицы построим кривую (рис. 8.3).

Из рисунка ясно, как получать значение С-функции на основании БСВ. Пример получения значения функции показан штриховыми стрелками, где значению БСВ, взятого с RN4, соответствует значение времени обслуживания. С-функция задается аналогично дискретной функции, для нашего примера получим

```
BANKSERV FUNCTION RN4,C6
0,60/.07,120/.23,180/.73,240
.92,300/1,360
```

С помощью С-функции можно осуществлять выборку и из равномерного распределения. Предположим, что ОБ GENERATE имеет операнды А, В 360 и 240 соответственно, тогда первая точка прямой равна 360 – 240 = 120, а вторая точка 360 + 240 = 600. Через полученные точки проходит прямая, точки которой будут соответствовать равномерному распределению при задании непрерывной С-функции в следующем виде:

```
TYPET FUNCTION RN3,C2
0,120/1,600
```

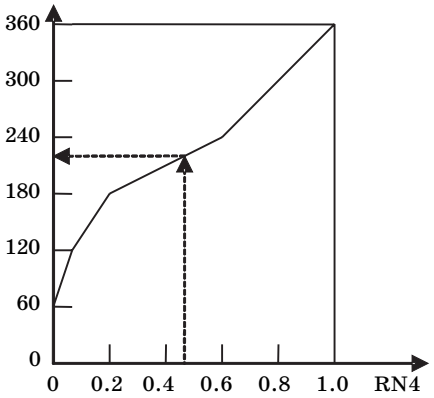


Рис. 8.3. *Графическое представление С-функции*

Поскольку по умолчанию данные для БСВ операндов A и B ОБ GENERATE или ADVANCE берутся с RN1, то для создания С-функции необходимо использовать другой ГСЧ, например RN3.

До настоящего момента в качестве независимой переменной (аргумента) использовались БСВ, с помощью которых находились зависимые значения накопленной частоты, ассоциируемые со значением функции. Однако возможен и другой, *инверсный метод*, когда за независимую переменную принимается накопленная частота, выбираемая случайно из равномерного распределения в интервале (0,1), а затем определяется зависимое значение функции. Инверсный метод описан во многих монографиях по моделированию, его преимуществом является возможность использования только одной выборки из равномерного распределения, этот метод применительно к конкретным задачам будет использован в § 8.3.

Для иллюстрации использования функций приведем пример производства с разными группами станков.

Пример 8.2. Модель производства с разными группами станков

1. Постановка задачи.

Механическое производство оснащено пятью видами станков (табл. 8.7).

Таблица 8.7

Номер вида	Число станков	Номер вида	Число станков
1	3	4	3
2	2	5	1
3	4		

Внутри каждого вида станки идентичны, поэтому не существенно, на каком из станков данного вида производится операция. На производстве одновременно обрабатывается три типа различных изделий, каждое из которых требует своей последовательности обработки на станках. Общее число видов и загрузка используемого станка для каждого типа изделий приведены в табл. 8.8.

Таблица 8.8

Тип изделия	Число занятых станков	Последовательность обработки	Время обработки, ч
1	4	3 1 2 5	0.5 0.6 0.85 0.5
2	3	4 1 3	1.1 0.8 0.75

Окончание табл. 8.8

Тип изделия	Число занятых станков	Последовательность обработки	Время обработки, ч
3	5	2 5 1 4 3	1.2 0.25 0.7 0.9 1.0

Расположение станков и порядок движения изделий представлены на рис. 8.4.

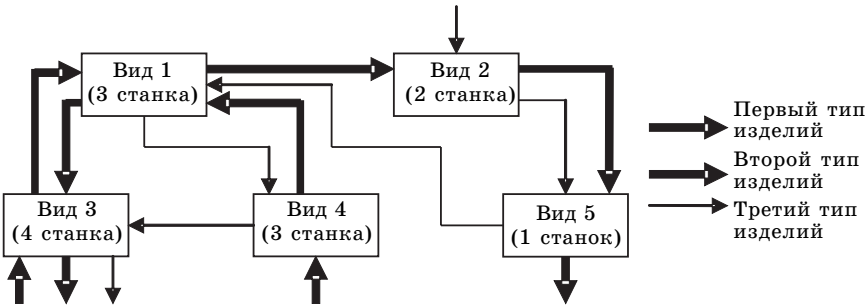


Рис. 8.4. *Схема движения изделий разного типа*

Установлено, что за 200 ч работы изделия первого типа составляют 30 % от общего числа, изделия второго типа — 50 %, а изделия третьего типа — 20 %.

Время поступления изделий задано в табл. 8.9, интервалы времени поступления изделий поделены на интервалы 0.2 ч, вероятность поступления внутри интервала принимается одинаковой. Интервалы, кроме первого и последнего слева, открыты, справа закрыты, последний интервал справа открыт.

Необходимо построить модель, собирая информацию с каждого вида станков. Будем считать, что в начальный момент времени 0.0 в цеху находится 8 изделий, не прошедших ни одной операции. Определить параметры работы цеха за 25 8-часовых рабочих дня. Все возможные изменения в

Таблица 8.9

Показатель	Данные						
Время поступления, ч	0.0	0.2	0.4	0.6	0.8	< 1.0	≥ 1.0
Относительная частота	0.0	0.40	0.30	0.15	0.10	0.05	0.0
Накопленная частота	0.0	0.40	0.70	0.85	0.95	1.0	–

системе не учитываются. Определить, какой вид станков является критичным для производства.

2. Допущения, сделанные в модели.

В модели для оценки поступления транзактов в модель используется C-функция, заданная в табл. 8.9, для оценки типов изделий вводится D-функция (30, 50, 20 %). ОБ TRANSFER, применяемый в безусловном варианте, служит для выбора направления движения транзактов. Каждый фрагмент модели моделирует поведение одного из типов обрабатываемых изделий в соответствии со схемой рис. 8.4. Наличие в системе первых восьми изделий реализуется ОБ GENERATE и TRANSFER, использующим дискретную функцию.

3. Таблица определений (табл. 8.10).

Временная дискрета: 1 мин

Таблица 8.10

Объект GPSS/H	Объект в системе
Транзакты	
Фрагмент 1 модели	8 изделий, находящихся в цеху
Фрагмент 2 модели	Поступление изделий
Фрагмент 3 модели	Изделия первого типа
Фрагмент 4 модели	Изделия второго типа
Фрагмент 5 модели	Изделия третьего типа
Фрагмент 6 модели	Транзакт управления
Функции	
JOBPATH	D-функция типов изделий
JOBIAT	C-функция времени поступления
Очереди	
GROUP1	Очереди для всех типов изделий, ожидающих обслуживания на соответствующих станках
GROUP2	
GROUP3	
GROUP4	
GROUP5	
Памяти	
GROUP1	Памяти, моделирующие виды используемых станков, с 1-го вида по 5-й
GROUP2	
GROUP3	
GROUP4	
GROUP5	

4. Модельный файл.

```
* Модуль описания
      SIMULATE
*
```

Пример 8.2. Модель производства с разными типами станков
Временная дискрета: 1 мин

```
JOBPATH   FUNCTION   RN1,D3 D                  функция типов изделий
.3,JOBTYPE1/.8,JOBTYPE2/1,JOBTYPE3
JOBIAT    FUNCTION        RN1,C6              С-функция поступления
                                              изделий
0,0/.4,.2/.7,.4/.85,.6/.95,.8/1,1
GROUP1    STORAGE                             3 станка 1-го вида
GROUP2    STORAGE                             2 станка 2-го вида
GROUP3    STORAGE                             4 станка 3-го вида
GROUP4    STORAGE                             3 станка 4-го вида
GROUP5    STORAGE                             1 станок 5-го вида
*    Модуль исполнения
*         Фрагмент 1                          8 изделий, находящихся
                                              в цеху
          GENERATE ,,,8                       8 изделий в МВ 0.0
          TRANSFER ,FN(JOBPATH)               пересылка к виду
                                              изделий
*         Фрагмент 2                          прибытие изделий
          GENERATE      FN(JOBIAT)            прибытие изделий одно
                                              за другим
          TRANSFER      ,FN(JOBPATH)          пересылка изделий
*         Фрагмент 3                          изделия 1-го типа
JOBTYPE1  QUEUE         GROUP3                создание очереди
                                              к станкам третьего вида
          ENTER         GROUP3                занятие станков
                                              третьего вида
          DEPART        GROUP3                выход из очереди
                                              к станкам третьего вида
          ADVANCE       RVEXPO(1,.25)         время обработки
                                              с распределением
                                              Эрланга 2-го рода
          LEAVE         GROUP3                освобождение станков
                                              третьего вида
*
          QUEUE         GROUP1                создание очереди
                                              к станкам первого вида
          ENTER         GROUP1                занятие станков первого
                                              вида
          DEPART        GROUP1                выход из очереди
                                              к станкам первого вида
          ADVANCE       RVEXPO(1,.3)          время обработки
                                              с распределением
                                              Эрланга 2-го рода
          LEAVE         GROUP1                освобождение станков
                                              первого вида
*
```

```
                QUEUE       GROUP2      создание очереди
                                        к станкам второго вида
                ENTER       GROUP2      занятие станков второго
                                        вида
                DEPART      GROUP2      выход из очереди
                                        к станкам второго вида
                ADVANCE     RVEXPO(1,.425) время обработки
                                        с распределением
                                        Эрланга 2-го рода
                LEAVE       GROUP2      освобождение станков
                                        второго вида
*
                QUEUE       GROUP5      создание очереди
                                        к станкам пятого вида
                ENTER       GROUP5      занятие станков
                                        пятого вида
                DEPART      GROUP5      выход из очереди
                                        к станкам пятого вида
                ADVANCE     RVEXPO(1,.25) время обработки
                                        с распределением
                                        Эрланга 2-го рода
                ADVANCE     RVEXPO(1,.25)
                LEAVE       GROUP5      освобождение станков
                                        пятого вида
*
                TERMINATE   0           изделие 1-го типа
                                        покидает цех
  *             Фрагмент 4              изделия 2-го типа
JOBTYPE2        QUEUE       GROUP4      создание очереди
                                        к станкам четвертого вида
                ENTER       GROUP4      занятие станков
                                        четвертого вида
                DEPART      GROUP4      выход из очереди
                                        к станкам четвертого вида
                ADVANCE     RVEXPO(1,.55) время обработки
                                        с распределением
                                        Эрланга 2-го рода
                LEAVE       GROUP4      освобождение станков
                                        четвертого вида
*
                QUEUE       GROUP1      создание очереди
                                        к станкам первого вида
                ENTER       GROUP1      занятие станков первого
                                        вида
                DEPART      GROUP1      выход из очереди
                                        к станкам первого вида
```

```
                ADVANCE     RVEXPO(1,.4)    время обработки
                                            с распределением
                                            Эрланга 2-го рода
                LEAVE       GROUP1          освобождение станков
                                            первого вида
*
                QUEUE       GROUP3          создание очереди
                                            к станкам третьего вида
                ENTER       GROUP3          занятие станков
                                            третьего вида
                DEPART      GROUP3          выход из очереди
                                            к станкам третьего вида
                ADVANCE     RVEXPO(1,.375)  время обработки
                                            с распределением
                                            Эрланга 2-го рода
                LEAVE       GROUP3          освобождение станков
                                            третьего вида
                TERMINATE   0               изделие 2-го типа
                                            покидает цех
*               Фрагмент 5                  изделия 3-го типа
JOBTYPE3        QUEUE       GROUP2          создание очереди
                                            к станкам второго вида
                ENTER       GROUP2          занятие станков второго
                                            вида
                DEPART      GROUP2          выход из очереди
                                            к станкам второго вида
                ADVANCE     RVEXPO(1,.6)    время обработки
                                            с распределением
                                            Эрланга 2-го рода
                LEAVE       GROUP2          освобождение станков
                                            второго вида
*
                QUEUE       GROUP5          создание очереди
                                            к станкам пятого вида
                ENTER       GROUP5          занятие станков пятого
                                            вида
                DEPART      GROUP5          выход из очереди
                                            к станкам пятого вида
                ADVANCE     RVEXPO(1,.125)  время обработки
                                            с распределением
                                            Эрланга 2-го рода
                LEAVE       GROUP5          освобождение станков
                                            пятого вида
*
                QUEUE       GROUP1          создание очереди
                                            к станкам первого вида
```

```
         ENTER     GROUP1        занятие станков первого
                                 вида
         DEPART    GROUP1        выход из очереди
                                 к станкам первого вида
         ADVANCE   RVEXPO(1,.35) время обработки
                                 с распределением
                                 Эрланга 2-го рода
         LEAVE     GROUP1        освобождение станков
                                 первого вида
*
         QUEUE     GROUP4        создание очереди
                                 к станкам четвертого вида
         ENTER     GROUP4        занятие станков
                                 четвертого вида
         DEPART    GROUP4        выход из очереди
                                 к станкам четвертого вида
         ADVANCE   RVEXPO(1,.45) время обработки
                                 с распределением
                                 Эрланга 2-го рода
         LEAVE     GROUP4        освобождение станков
                                 четвертого вида
*
         QUEUE     GROUP3        создание очереди
                                 к станкам третьего вида
         ENTER     GROUP3        занятие станков
                                 третьего вида
         DEPART    GROUP3        выход из очереди
                                 к станкам третьего вида
         ADVANCE   RVEXPO(1,.5)  время обработки
                                 с распределением
                                 Эрланга 2-го рода
         LEAVE     GROUP3        освобождение станков
                                 третьего вида
         TERMINATE 0             изделие 3-го типа
                                 покидает цех
*        Фрагмент 6               временной таймер
         GENERATE  200           Хакт управления
                                 приходит через 200 ч
         TERMINATE 1             уменьшение CC на 1,
                                 прекращение движения
                                 транзактов
*        Модуль управления
         START     1             установка CC на 1,
                                 начало движения
                                 транзактов
         END                     окончание процесса ИМ
```

5. Итоговый отчет

Краткие итоги моделирования сведены в табл. 8.11.

Таблица 8.11

Наименование	Среднее время на один отказ	Число входов	Максимум	Среднее число
Памяти				
GROUP1	0.756	630	3	0.72
GROUP2	0.796	308	2	1.033
GROUP3	0.573	624	4	0.735
GROUP4	0.789	450	3	1.052
GROUP5	0.620	307	1	0.404
Очереди				
GROUP1	0.243	630	8	0.764
GROUP2	1.203	308	13	1.85
GROUP3	0.052	624	5	0.164
GROUP4	0.554	457	10	1.267
GROUP5	0.398	307	7	0.611

Примечание. Подробный листинг изучите, проведя моделирования примера 8.2.

6. Обсуждение.

Использование памятей (видов станков) не превышает 80 %, среднее число изделий, ожидающих обслуживания, мало и не превышает в среднем 1.85, однако при пиковой нагрузке максимальное число изделий в очереди ко второму виду станков сравнительно велико и равно 13. Этот результат объясняется большим разбросом времени обслуживания с распределением Эрланга 2-го рода.

Поэтому можно думать о добавлении станков второго вида. Более подробно концепция выбора наилучшей альтернативы будет рассмотрена в § 8.5.

§ 8.2. ПЛАНИРОВАНИЕ ПРОЦЕССА ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ

В § 8.1 мы уточнили, как обеспечивается начало и протекание процесса ИМ, основанное на генерации БСВ чисел и функций с разными функциями распределения. В настоящем параграфе рассмотрим вопросы, связанные с планированием ИМ:

— продолжительность процесса ИМ;

— выбор статистических характеристик исследуемого процесса,

— выбор методов обработки накопленных выходных данных.

Для решения этих вопросов используются методы планирования эксперимента, рассмотренные рядом авторов [11, 15, 20].

8.2.1. Продолжительность процесса ИМ

Ответ на этот вопрос представляется довольно простым правилом: *моделировать так долго, чтобы достигнуть поставленных целей*. Очевидно, что продолжительность ВЭ является одним из важных параметров процесса ИМ. Поэтому надо выбирать между решением проведения одного прогона модели или проведения многих независимых прогонов для получения высокой доверительной вероятности. В первом случае можно получить уверенность в работоспособности модели и определить один набор значений выходных характеристик, во втором случае можно проводить моделирование неоправданно долго. Истина, как и в любой практической ситуации, лежит где-то между этими границами. Прежде всего, необходимо понять, по какому критерию можно оценивать длительность ИМ. В числе основных вариантов можно назвать оценку:

— среднего значения;

— разности между средними значениями;

— по наилучшему варианту.

Рассмотрим эти варианты более подробно.

Оценка среднего значения

Очень часто необходимо произвести оценку каких-то средних значений, например длину очереди к расчетному узлу супермаркета в разные дни и разные периоды рабочего времени. Промоделируем подобную ситуацию n раз и получим набор независимых случайных значений $X_1, X_2, ..., X_n$, каждое из которых имеет одинаковое нормальное распределение с неизвестным средним значением μ и дисперсией σ^2. Используя центральную предельную теорему, можно считать, что оценка

$$\bar{X} = 1/n \sum_{i=1}^{n} Xi \qquad (8.10)$$

примерно соответствует нормальному распределению $N(\mu, \sigma^2)$.

Необходимо найти число прогонов n, которое и определит продолжительность испытаний, при этом разность между определяемым средним значением и искомым средним будет меньше или равна наперед заданному малому числу d:

$$P(\left|\bar{X} - \mu\right| \le d) = P^* \text{ при } d = 0.01,\ P^*=0.95. \qquad (8.11)$$

Поскольку $(X - \mu)/(\sigma/\sqrt{n})$ соответствует нормализованному распределению $N(0,1)$, то левая часть (8.11) равна

$$P\left\{-d(\sigma/\sqrt{n}) \le (\bar{X}-\mu)/(\sigma/\sqrt{n}) \le d/(\sigma/\sqrt{n})\right\} =$$

$$= \Phi(d/(\sigma/\sqrt{n})) - \Phi(-d/(\sigma/\sqrt{n})) = 2\Phi(d/(\sigma/\sqrt{n})) - 1 \qquad (8.12)$$

тогда и только тогда, когда

$$2\Phi(d/(\sigma/\sqrt{n})) - 1 = P^* \text{ или } (d\sqrt{n}/\sigma = \Phi^{-1}((1+P^*)/2). \qquad (8.13)$$

Таким образом, чтобы удовлетворить (8.11), число прогонов

$$n = [\{\sigma/d\ \Phi^{-1}((1 + P^*)/2)\}^2], \qquad (8.14)$$

где значение [.] округляется до ближайшего целого числа, например: при заданных $P^* = 0.95$, $d = 0.01$ и $\sigma = 0.5$ число прогонов $n = 10000$.

В том случае, если по соображениям экономии машинного времени возможно провести только 400 прогонов, то точность d при этих же данных равна 0.05.

Если $X_1, \ldots, X_n$ не подчиняются нормальному распределению, то необходимо применять специальные методы преобразования этих переменных в переменные, имеющие приблизительно нормальное распределение, например с помощью преобразования Бокса, применяя новую переменную Y и коэффициент мощности преобразования λ. В выражении (8.14) предполагается, что значение стандартного отклонения σ известно, однако часто значение дисперсии не известно. Значение выборочной дисперсии s^2 не может послужить отправной точкой, так как не известно число прогонов n.

Теорема 8.2. Если $X_1, X_2, \ldots, X_n$ независимые случайные величины с распределением $N(\mu, \sigma^2)$ и неизвестными μ и σ^2, тогда не существует одношаговой статистической процедуры $T(X_1, X_2, \ldots, X_n)$ такой, чтобы

$$P(T(X_1, X_2, \ldots, X_n) - d \le \mu \le T(X_1, X_2, \ldots, X_n) + d) = P^*. \qquad (8.15)$$

Утверждение, что μ будет находиться в интервале между $\bar{X} - d$ и $\bar{X} + d$ для любого выбранного n, не будет корректным, так как это утверждение справедливо только с вероятностью

$$P(\bar{X} - d \le \mu \le \bar{X} + d) = P(-d \le X - \mu \le d) = 2\Phi(d/(\sigma/\sqrt{n})) - 1,$$

которая при больших значениях стандартного отклонения может быть намного меньше, чем заданная доверительная вероятность. Стремление использовать в этом случае распределение Стьюдента не может привести к правильным результатам, так как при этом точность будет зависеть от числа прогонов и невозможно будет уложиться в требуемый интервал $2d$.

Для случая неизвестной дисперсии Дадевичем [15] предложена 7-шаговая процедура, названная методом HM (heteroscedastic method — метод, отличный от существующих).

Метод HM

HM1 — назначим положительное целое значение $n_0 \geq 2$. Будем считать, что переменная ω, необходимая для дальнейших расчетов, имеет распределение Стьюдента — t-распределение (что строго доказывается, но в пособии не рассмотрено) с ν степенями свободы.

Примечание. Напомним, что число степеней свободы равняется числу независимых свободных наблюдений минус число статистик, определенных на основе этих наблюдений. Для t-распределения такая статистика одна — среднее значение, поэтому $\nu = n - 1$.

Значение ω определится следующим образом:

$$\omega = t^{-1}_{n_0-1}((1+P^*)/2)/d. \qquad (8.16)$$

HM2 — получим значения переменных $X_1, \ldots, X_{n_0}$.

HM3 — вычислим

$$\bar{X}(n_0) = (X_1 + \ldots + X_{n_0})/n_0,\ s^2 = \sum_{i=1}^{n_0}(X - \bar{X}(n_0))^2/(n_0 - 1).$$

HM4 — положим, что

$$n = \max\{n_0 + 1, [(t^{-1}_{n_0-1}((1+P^*)/2))^2 s^2/d^2]\}, \qquad (8.17)$$

где $[x]$ обозначает наименьшее целое число, которое $\geq x$.

HM5 — получим значения переменных $X_{n_0+1}, \ldots, X_n$.

HM6 — вычислим

$$\bar{\bar{X}} = X_1 + \ldots + (X_{n_0} + X_{n_0+1} + \ldots + X_n)/n.$$

HM7 — со 100% -й достоверностью утверждаем, что

$$\bar{\bar{X}} = -d \leq \mu \leq \bar{\bar{X}} + d.$$

Возникает вопрос, какое начальное значение n_0 следует выбирать при проведении этой процедуры. В табл. 8.12 приведены значения n_0

Таблица 8.12. *Значения* ω *для разных значений* n_0

Показатель	Значение				
n_0	2	5	15	60	∞
$t^{-1}_{n_0-1}(0.975)$	12.706	2.776	2.145	2.00	1.96
ω	161.44	7.71	4.60	4.00	3.84

для $P^* = 0.95$, значения для других величин вероятности легко рассчитать, используя формулы, приведенные в НМ1–НМ7.

Оценка разности между средними значениями

Довольно часто в процессе ИМ приходится оценивать эффективность функционирования двух разных вариантов реальной системы, и тогда представляется интересной оценка разницы между средними значениями выходных характеристик этих вариантов.

Будем считать, что при первом способе имеем ряд переменных X_1, X_2, ... с неизвестными средним μ_1 и дисперсией σ_1^2, а при втором — переменные Y_1, Y_2, ... с неизвестными μ_2 и σ_2^2.

Целью исследования является получение разности $\mu_1 - \mu_2$ с точностью $\pm d$.

Решение этой задачи сводится к следующему.

1. Для переменных X_1, X_2, ..., используя какое-то фиксированное ω, по методике НМ (этапы НМ1–НМ4) рассчитываем среднее $\bar{X}$, а затем вычисляются значения

$$T_1 = (\tilde{X} - \mu_1)/1/\omega,$$

где

$$\tilde{X} = b\bar{X}(n_0) + (1+b)\bar{X}(n - n_0);$$
$$b = n_0 / n(1 + \sqrt{1 - n/n_0(1-(n-n_0)}/(\omega s)^2). \qquad (8.18)$$

2. Для переменных Y_1, Y_2, ..., используя то же самое ω и ту же процедуру, рассчитываем среднее $\bar{Y}$ и значение T_2 по схеме (8.18) для Y. Получаем два независимых случайных числа, имеющих распределение Стьюдента (t-распределение). Таким образом, получаем интервал $\tilde{X} - \tilde{Y} - d \le \mu_1 - \mu_2 \le \tilde{X} - \tilde{Y} + d$, вероятность покрытия которого P равняется

$$P(\tilde{X} - \tilde{Y} - d \le \mu_1 - \mu_2 \le \tilde{X} - \tilde{Y} + d) =$$
$$= P(-d/ \le 1/\omega(\tilde{X} - \mu_1)/1/\omega - (\tilde{Y} - \mu_2)/1/\omega \le d \cdot 1/\omega) =$$
$$= P(-\omega d \le T_1 + T_2 \le \omega d) = F_{T_1+T_2}(\omega d) - F_{T_1+T_2}(-\omega d) = 2F_{T_1+T_2}(\omega d) - 1. \qquad (8.19)$$

Отметим, что поскольку разность T_i и их сумма имеют одинаковое распределение, это позволяет произвести замену во второй строке выражения (8.19), чтобы не иметь дело с отрицательным значением ω. Из (8.19) можно определить значение ω:

$$\omega = ((F^{-1}_{T_1+T_2}(1 + P^*/2))/d. \qquad (8.20)$$

Таблица 8.13. Значения величины выборки n_0

Показатель	Значение		
$F^{-1}_{T_1-T_2}(0.975)$	25.42	3.94	2.95
n_0	2	5	20

Для доверительной вероятности $P^* = 0.95$ значение функции F берется равным 0.975 (двусторонний интервал), и тогда можно получить данные, приведенные в табл. 8.13.

В теории математической статистики эта задача называется задачей Беренса—Фишера.

Оценка по наилучшему варианту

Довольно часто одна и та же задача может быть выполнена одной и той же системой разными способами, и надо определить лучший способ. Вначале рассмотрим случай, когда эффективность системы оценивается одной выходной характеристикой и качество системы определяется по максимуму среднего значения этой выходной характеристики.

Предположим, что рассматривается несколько способов функционирования:

Способ 1 $X_{11}, X_{12}, \ldots$ со средним μ_1 и дисперсией σ_1^2

Способ 2 $X_{21}, X_{22}, \ldots$ со средним μ_2 и дисперсией σ_2^2

.

.

.

Способ k $X_{k1}, X_{k2}, \ldots$ со средним μ_k и дисперсией σ_k^2

На основе полученных выборок X_{ij} ($i = 1, 2, \ldots, k$, $j = 1, 2, \ldots, n$) определяются средние выборочные значения $\bar{X}_i$ и определяется наибольшее значение среди всех способов. Основным вопросом при выбранном значении объема выборки n является выяснение того, максимален ли выбранный способ или насколько он приблизился к максимуму. Тогда вероятность того, что разность между максимумом и выбранным способом не превысит наперед заданную точность $\delta > 0$, равна

$$P(X_{\max} - X_i \le \delta) \ge P^*. \qquad (8.21)$$

Необходимо отметить, что объем выборки n не может быть определен в рамках одношаговой статистической процедуры (на основе простого правила, что нельзя использовать данные до их получения). Следовательно, надо прибегнуть к двухэтапной процедуре НМ-метода. Выше описана одноэтапная процедура. Для перехода к двухэтапной процедуре используем первые четыре шага НМ1–НМ4 и вычислим с помощью (8.18) независимые значения $T_i = (\tilde{X}_i - \mu_i)/1/\omega$ для

каждого $i = 1, 2, ..., k$, которые подчиняются распределению Стьюдента. После определения $\tilde{X}_i$ для каждого способа найдем наибольшее значение $\tilde{X}_j$ в лучшем способе, тогда вероятность выбора максимального значения в лучшем способе определится как

$$P(\tilde{X}_j \to \max) = P[\tilde{X}_{(i)} < \tilde{X}_{(k)} \text{ aeu anao } i = 1, 2, \ldots, k-1]. \quad (8.22)$$

Разместим средние значения выходных характеристик разных способов в порядке возрастания:

$\mu_{[1]}, \mu_{[2]}, ..., \mu_{[k]}$, а так как $T_{(1)}, ..., T_{(k)}$ — независимые случайные переменные, подчиняющиеся t-распределению с $n-1$ степенями свободы, то нетрудно показать, что вероятность P (8.22) определится как

$$P(\tilde{X}_j \to \max) = \int_{-\infty}^{\infty} \left(\prod_{i=1}^{k-1} F_{n_0}(z + (\mu_{[k]} - \mu_i)\omega) \right) f_{n_0}(z)\mathrm{d}z, \quad (8.23)$$

где соответственно $F_{n_0}(.)$ и f_{n_0} — функция распределения и плотность распределения вероятности распределения Стьюдента с $n-1$ степенями свободы. Эта вероятность будет равна заданной доверительной вероятности P^*, если положить $\omega = h_{n_0}(k, P^*)/\delta$, где $h_{n_0}(k, P^*)$ является решением для h в выражении

$$\int_{-\infty}^{\infty} (F_{n_0}(z+h))^{k-1} f_{n_0}(z)\mathrm{d}z = P^*. \quad (8.24)$$

Значение $h_{n_0}(k, P^*)$ табулировано рядом авторов, и в табл. 8.14 приведено несколько значений для разного числа способов при $n_0 = 15$, $P^* = 0.95$.

Сформулируем этапы проведения описанной процедуры.

1. Выбирается начальный объем выборки $n_0 \geq 2$ и значение $\omega = h_{n_0}(k, P^*)/\delta$, где h является решением уравнения (8.24).

2. Определяются члены первой выборки $X_{i1}, X_{i2}, ..., X_{in_0}$ и вычисляется выборочное среднее $\bar{X}_i(n_0)$ и выборочная дисперсия s_i^2.

3. Определяется максимальное целое значение объема добавленной выборки $n_i = \max\{n_0 + 1, |(\omega si)|^2\}$, а затем получаются значения новых членов выборки $X_{in_0+1}, ..., X_{in}$.

4. Вычисляется среднее значение для новой выборки $\bar{X}_i(n_i - n_0)$.

Таблица 8.14. *Значения h для разных значений k*

Показатель	Значение				
k	3	4	5	10	15
$h_{15}(k, 0.95)$	2.94	3.17	3.34	3.79	4.02

5. Вычисляется $\tilde{X}_i = b_i \bar{X}_i(n_0) + (1 - b_i)\bar{X}_i(n_i - n_0)$, где b_i определяется из (8.18).

6. Подобные операции производятся для каждого способа функционирования реальной системы, т. е. для $i = 1, 2, ..., k$.

7. Выбирается как наилучший способ, в котором имеется максимальное значение $\tilde{X}_i$, и затем определяется вероятность того, что выбранный способ является либо лучшим, либо лежит в пределах заданной точности при заданной доверительной вероятности:

$$P(\tilde{X}_{\max} - \tilde{X}_i \le \delta) \ge P^*. \qquad (8.25)$$

Рассмотренный случай демонстрирует логику выбора лучшего способа среди ряда альтернатив при учете одной выходной характеристики. На практике приходится учитывать несколько выходных характеристик, например точность и быстродействие; прибыль и объем продаж и т. д. Предположим, что m=1, 2, ..., p выходных характеристик i-й системы подчиняются нормальному распределению со многими переменными, с вектором средних значений $\mu_i = (\mu_{i1}, ..., \mu_{ip})$ и ковариационной матрицей Σ для $i = 1, 2, ..., k$.

В данном случае можно использовать несколько стратегий.

- Выбрать систему, для которой максимален вектор средних значений.
- Выбрать систему, у которой максимальна сумма средних значений, каждое из которых умножено на коэффициент значимости.
- Выбрать систему, у которой максимальна сумма квадратов средних значений.
- Выбрать систему, у которой максимально произведение вектора средних значений на ковариационную матрицу.

Указанные стратегии хорошо разработаны, имеются в книге по статистике пакета MATLAB, а также в пакете «Статистика». Выбор функции предпочтения зависит как от конкретной задачи, так и от выбора пользователя.

Поэтому, обозначив возможность решения задачи выбора лучшего варианта, остановимся на этом и отошлем читателя к § 8.5, где будет рассмотрен пример выбора наилучшего варианта.

8.2.2. Статистическое планирование и регрессионный анализ

Довольно часто при исследовании малоизученных объектов (задачи теории технической и экономической кибернетики, теории распознавания образов и т. д.) используется метод «черного ящика».

Суть его сводится к описанию входных воздействий-факторов, определению выходных характеристик-откликов, установлению взаимосвязей «фактор—отклик» и на основе их анализа — определению внутренней структуры исследуемого объекта. Процесс ИМ является прекрасным средством для решения задачи установления зависимости выходной характеристики Y от ряда входных воздействий x_1, x_2, ..., x_k для некоторого $k \geq 1$. При такой постановке задачи вначале задается ряд числовых значений входных воздействий, затем проводится моделирование и анализируется полученная выходная характеристика. Затем определяется среднее значение выходной характеристики в виде функции от входных воздействий. Примеров подобного подхода можно привести сколь угодно много (количество машин и водителей на станции «Скорой помощи», использование компьютеров в локальной сети, операционистов в банке и т. д.). В качестве примера рассмотрим наличие на станции «Скорой помощи» x_1 водителей и x_2 спецавтомашин, при этом важно знать время Y от получения вызова до выезда автомашины с бригадой врачей. Тогда среднее время $M(Y)$ определится как функция $f(x_1, x_2)$ от x_1 и x_2 при условии получения наименьших потерь C (т. е. количества водителей и автомашин при сохранении эффективности медицинского обслуживания). При этом $M(Y) \leq C$ оценивается как время обслуживания. Автомашины могут ремонтироваться, а часть водителей быть больны или в отпуске, поэтому совершенно очевидно, что равенство $x_1 = x_2$ практически никогда не существует. Обычно $M(Y)$ является непрерывной монотонной функцией входных воздействий и может быть представлена полиномом любого порядка, а в нашем случае — полиномом четвертой степени (при $k = 4$)

$$M(Y) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_{12} x_1 x_2 + \beta_{13} x_1 x_3 + \ldots + \\ + \beta_{1234} x_1 x_2 x_3 x_4 + \ldots \qquad (8.26)$$

На практике принято считать, что члены уравнения (8.26), имеющие порядок более двух, пренебрежимо малы и оказывают небольшое влияние на среднее значение, поэтому в результате отсеивающего анализа их отбрасывают.

Таким образом, при планировании имитационного эксперимента стараются сократить размер полинома без потери точности, тем более, что для описания эксперимента требуется, чтобы все коэффициенты β были определены количественно. Поясним процедуру простым примером при $k = 1$. Более сложные случаи можно изучить в любой монографии по регрессионному анализу.

А. Использование планирования для k = 1

Пример 8.3. Вычислительная система общего доступа (обоснования и модель)

В данном примере несколько отойдем от схемы представления примеров, принятой в пособии.

Рассмотрим стандартную схему одного ЦПУ и n терминалов. Предположим, что вначале производится подготовка задания на терминале, после подготовки на терминале следует обращение к ЦПУ. На ЦПУ организуется очередь на обслуживание с ДО FCFS. Если заявка выполнена за оговоренное время, она поступает на свой терминал, в противном случае она становится в конец очереди с уменьшенным временем обслуживания. Подобная задача описана Лоу и Келтоном еще в 1982 г. [16]. Однако она представляет интерес для описания процесса планирования ИМ. Главным вопросом при этом является определение времени Y-ухода заявки с терминала и возвращения на него полностью исполненной на ЦПУ, назовем это время *циклом*. Число x_1 при этом представляет собой количество терминалов. Сформулируем вопрос следующим образом: «Сколько терминалов может быть соединено с ЦПУ, чтобы $M(Y)$ не превышало 6 с?»

В случае одного варьируемого параметра x_1 уравнение регрессии (8.26) будет сокращено до следующего вида:

$$M(Y) = \beta_0 + \beta_1 x_1 + \beta_2 x_2.$$

Необходимо определить коэффициенты β, для этого надо получить значения $M(Y)$ для разных значений переменной. Воспользуемся данными моделирования (см. ниже), которые сведены в табл. 8.15.

Из таблицы видно, что искомое время лежит между значениями 25 и 35 терминалов. Используя методы теории регрессионного анализа (см. ППП «Статистика» или книгу по статистике ППП MATLAB) находим, что обычная оценка (наименьших квадратов) в нашем случае будет иметь вид

$$\hat{\tilde{\beta}} = (\tilde{X}\tilde{X})^{-1}\tilde{X}\tilde{Y}; \qquad (8.27)$$

$$\hat{\tilde{\beta}} = \begin{pmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \\ \hat{\beta}_2 \end{pmatrix}, \quad \tilde{X} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 15 & 25 & 35 & 45 \\ 225 & 625 & 1225 & 2025 \end{pmatrix}, \quad \tilde{Y} = \begin{pmatrix} 1.637 \\ 3.125 \\ 7.613 \\ 16.518 \end{pmatrix}. \qquad (8.28)$$

Таблица 8.15. *Зависимость среднего времени цикла от числа терминалов*

Показатель	Значение			
Число терминалов x	15	25	35	45
Среднее время $M(Y)$	1.637	3.125	7.613	16.518

Таблица 8.16. *Сравнение данных моделирования и предсказания по (8.29)*

Показатель	Значение			
Число терминалов	15	25	35	45
Результаты ИМ	1.637	3.125	7.613	16.518
Предсказание из (8.29)	1.721	2.962	7.866	16.433

Используя один из вышеназванных статистических пакетов (или применяя прямые вычисления), подставив приведенные в (8.28) значения в (8.27), получим уравнение регрессии в виде

$$M(Y) = 6.72955 - 0.60865x_1 + 0.0183177x_2^2. \qquad (8.29)$$

Выражение (8.29) можно использовать для предсказания времени цикла при разных значениях числа терминалов, эти данные приведены в табл. 8.16.

Видим, что совпадение данных очень высокое, а так как можно выбирать только целое число терминалов, то расхождения практически нет, что свидетельствует о правильности выбора уравнения регрессии. Для того чтобы найти искомое число терминалов, приравняем выражение (8.29) заданному времени:

$6 = 6.72955 - 0.60865x_1 + 0.0183177x_2^2$, решая относительно x_1, получим, что $x_1 = 31.98$.

Отметим, что этот ответ получен при ряде сделанных допущений:

— не уточнено, когда начался сбор данных, т. е. в системе мог еще не окончиться переходной режим (см. § 8.3);

— не рассмотрена возможность проведения на терминалах нескольких заданий;

— не рассмотрены более сложные дисциплины обслуживания на ЦПУ.

Оставим эти варианты рассмотрения для заинтересованных читателей.

Подчеркнем, что прямое решение вопроса (примитивная оптимизация), сформулированного выше, приводит к необходимости проведения более 30 прогонов для всех значений числа терминалов от 15 до 45, что превысит затраты машинного времени для четырех прогонов более чем на 700 %. Поэтому примитивная оптимизация в принципе не должна использоваться при планировании имитационных экспериментов.

После сделанных вычислений проведем построение модели системы общего доступа, используя уже привычную схему.

1. Постановка задачи.

Имеется система общего доступа с одним центральным процессором и терминалами (из проведенных выше расчетов число терминалов положим равным 32). Время подготовки задания на терминале распределено по экспоненциальному закону с параметром 25 с. После подготовки задания оно обращается к ЦПУ, время обращения также распределено по экспоненциальному закону с параметром 0.8 с. Все задания выстраиваются в очередь с дисциплиной обслуживания FCFS. Обслуживание каждого задания происходит в течение 0.1 с плюс 0.015 с, которые возникают при

постановке и выходе задания из очереди. Если задание не завершается за 0.1 с, оно становится в конец очереди к ЦПУ, время выполнения задания при этом уменьшается на 0.1 с. При завершении за 0.1 с задание возвращается на тот терминал, с которого оно поступило. Необходимо оценить среднее время цикла при заданном числе терминалов. За временную дискрету примем значение 1 мс.

2. Допущения, сделанные в модели.

Для того чтобы проверить расчеты, сделанные по (8.29), в МФ сразу введем значение терминалов, равное 32. При этом будем считать, что:

— с одного терминала за моделируемое время поступает только одно задание;

— в момент начала моделирования все терминалы начинают работать с нуля;

— очередь заданий, выстраивающихся к ЦПУ, реализуется по принципу FCFS, т. е. ни одно задание не имеет приоритета.

Теперь дадим пояснения к некоторым, впервые применяемым в данном примере, операторам. В модуле управления и описания используются ОУ INITIAL для введения начальных значений, FVARIABLE — для задания переменной с плавающей точкой, TABLE — для задания параметров таблицы.

В модуле исполнения используются ОБ ASSIGN — для присвоения получаемых значений параметрам транзакта, MARK — для записи абсолютного времени, TEST — для проверки логических условий, TABULATE — для занесения данных в таблицу, SAVEVALUE — для подсчета завершенных заданий. Описание роли этих ОБ будет приведено в п. 6 данного примера.

3. Таблица определений (табл. 8.17).

Таблица 8.17

Объект GPSS/H	Объект в системе
Транзакты	
NOTERM	Количество терминалов
STRT	Транзакт управления
Устройства	
CPU	Центральный процессор
Переменные	
OHEAD	Время организации очереди
SLICE	Время исполнения задания
SWAP	Суммарное время ЦПУ
NOJOBS	Неоконченные задания
CRT	Оконченные задания
LSTTIME	Время ЦПУ к последнему заданию
Очереди	
CPUQ	Очередь заданий к ЦПУ
Таблицы	
RSPTIME	Итоговая таблица

4. Модельный файл.

```
*                                          Пример 8.3. Модель
                                           системы общего доступа
*                                          Временная дискрета: 1 мс
*  Модуль 1
           SIMULATE
           INITIAL     X$OHEAD,15/
                       X$SLICE,100         время организации очереди/
                                           время исполнения ЦПУ
           INITIAL     X$SWAP,115          полное время занятости
                                           ЦПУ на одно задание
           INITIAL     X$CTR,0/
                       X$NOJOBS,
                       6000                счетчик заданий/
                                           окончание заданий
LSTTIME    FVARIABLE   P1+X$OHEAD          время ЦПУ на последнем
                                           цикле
RSPTIME    TABLE       M1,600,600,25       таблица времени ЦПУ
*  Модуль 2
           GENERATE    ,,,32               начало работы всех
                                           терминалов
STRT       ADVANCE     RVEXPO
                       (1,25000)           режим подготовки заданий
           ASSIGN      2,RVEXPO
                       (1,800)             P2= полному времени
                                           работы ЦПУ
           ASSIGN      1,P2                P1= времени на задание
           MARK                            запись времени
LOP        QUEUE       CPUQ                создание очереди
                                           ожидания ЦПУ
           SEIZE       CPU                 начало работы на ЦПУ
           DEPART      CPUQ                выход из очереди к ЦПУ
           TEST GE     P1,X$SLICE,
                       LAST                проверка условия,
                                           что обращение последнее
           ASSIGN      1-,X$SLICE          если не последнее,
                                           уменьшение времени
           ADVANCE     X$SWAP              обслуживание на ЦПУ
           RELEASE     CPU                 освобождение ЦПУ
           TRANSFER ,LOP                   возвращение в очередь
                                           к ЦПУ
LAST       ADVANCE     V$LSTTIME           последнее обслуживание
                                           на ЦПУ
           RELEASE     CPU                 освобождение ЦПУ
           ASSIGN      1,0                 уменьшение потребного
                                           времени до нуля
```

TABULATE	RSPTIME	внесение данных в таблицу
SAVEVALUE	CTR+,1	увеличение на единицу числа заданий
TEST GE	XCTR, XNOJOBS, STRT	при достижении числа заданий — остановка
TERMINATE	1	выполнение условия остановки
START	1	
END		окончание процесса ИМ

5. Итоговый отчет.

Наиболее интересные результаты сведены в табл. 8.18

Таблица 8.18. *Основные результаты моделирования*

Наименование	Использование	Число входов		Среднее значение	Максимальное
		общее	нулевое		
Устройство	0.966	42336	–	108.93	–
Очередь	–	42336	37542	4.95	17
Примечание. Для получения полного листинга необходимо провести моделирование представленного МФ.					

Для 32 терминалов получено значение цикла 5.65 с, для 33 терминалов — уже 6.26 с, поэтому для выполнения условия задачи окончательно принимается 32 терминала, что было предсказано при решении выражения (8.29).

6. Обсуждение МФ и результатов ИМ.

В представленном МФ следует отметить, что сигналом к окончанию испытания является число завершенных заданий, которое невозможно включить в операнд А ОУ START, поскольку задания могут возвращаться на ЦПУ несколько раз. Для проверки логических условий окончания использования ЦПУ и равенства числа завершенных заданий числу терминалов в МФ поставлены два ОБ TEST, первый из которых проверяет условие того, что поступление задания на ЦПУ будет последним. Это происходит путем сравнения параметра транзакта P1 со временем реализации на ЦПУ: если это время оказывается меньше или равно 0.1 с, то это поступление задания последнее, если больше, то задание отправляется в конец очереди на ЦПУ. Второй ОБ сравнивает число выполненных заданий с незавершенными и служит основой для подачи команды на завершение испытаний.

Время ИМ весьма мало и поэтому намного проще провести несколько прогонов при разных значениях числа терминалов и определить алгоритм построения уравнения регрессии, чем проводить примитивную оптимизацию.

Довольно часто при планировании имитационного эксперимента для нахождения зависимости среднего значения выходной характеристики от нескольких факторов приходится балансировать между двумя граничными ситуациями:

с одной стороны, увеличение числа факторов позволяет находить большее число коэффициентов выражения (8.29).

с другой стороны, уменьшение числа факторов дает возможность сократить время испытаний.

Б. Использование планирования для k = 2, центральное композитное планирование

Дадевич [15] предложил метод *центрального композитного планирования* (ЦКП)(Central Composite Design), который рассмотрим на примере планирования для $k=2$. Этот метод позволяет определить коэффициенты полного квадратического уравнения

$$M(Y) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_{11} x_1^2 + \beta_{22} x_2^2 + \beta_{12} x_1 x_2. \qquad (8.30)$$

Для решения (8.30) необходимо провести 9 прогонов ($2^k + 2k + 1$) с входными данными, приведенными в табл. 8.19. В процессе моделирования получаются четыре факториальные точки при значениях факторов $x_1 x_2$, принимающих значения ± 1, четыре «звездные» точки, лежащие на осях со значениями $\pm\alpha$, и одна центральная точка с координатами (0,0). Значение «звездной» точки α показывает, насколько далеко мы отходим от центра. Так, при значении $\alpha = 2$ мы находимся вдвое дальше от центра, чем уровень, принятый за ± 1. Из кодирования таблицы ± 1 не следует, что значение уровней должно быть одинаковым, например: некодированное нижнее положительное значение для x_1 может быть равно 3 – кодированный уровень (−1), а верхнее значение 13 представляет собой кодированный уровень (+1), тогда центральная точка равна (3 + 13)/2 = 8. Предполо-

Таблица 8.19. *Значения для моделирования по методу ЦКП*

Название прогонов	Некодированные x_1 x_2	Кодированные x_1 x_2
Факториальные точки	(3, 5) (3, 20) (13, 5) (13, 20)	(−1, −1) (−1, 1) (1, −1) (1, 1)
«Звездные» точки	(1, 12.5) (15, 12.5) (8, 2) (8, 23)	(−α, 0) (α, 0) (0, −α) (0, α)
Центральная точка	(8, 12.5)	(0, 0)

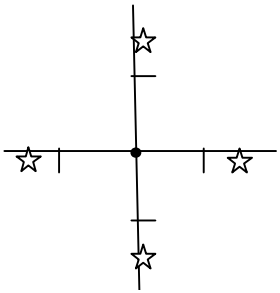


Рис. 8.5. Иллюстрация применения ЦКП

жим, что $\alpha = 1.4$, тогда для x_1 $8 - 3 = 13 - 8 = 5$. При $\alpha = 1.4 \cdot 5 = 7$ и для отрицательных $8 - 7 = 1 \cdot (-\alpha)$, и для положительных значений $8 + 7 = 15 \cdot \alpha$ результаты одинаковы. Примем некодированные значения x_2 равными 5 — кодированный уровень (-1) и верхнее значение 20 $(+1)$, центральное значение равно $(5 + 20)/2 = 12.5$. Традиционно при использовании ЦКП значение α выбирается одинаковым для всех k факторов. Тогда для x_2 получим $12.5 - 5 = 20 - 12.5 = 7.5$, а «звездная» точка равна $7.5 \cdot 1.4 = 10.5$, а ее положение на оси равно $12.5 - 10.5 = 2$ и $12.5 + 10.5 = 23$. Все полученные данные сведены в табл. 8.19 и рис. 8.5.

При использовании ЦКП необходимо иметь в виду следующее:

— при необходимости определить целочисленные значения (число операторов в банке, число станков и т. п.) планирование эксперимента должно учитывать это ограничение;

— иногда приходится выбирать значение α разным, для того чтобы хорошо выбрать планируемую область;

— при попадании точек в область нежелательных значений выходной характеристики необходимо их преобразовать.

В данном параграфе были рассмотрены лишь некоторые вопросы планирования имитационного эксперимента, не нашедшие отражения в русскоязычной литературе, заинтересованный читатель может обратиться к литературе [8–11], где представлен ряд других методов. В заключение рассмотрим применение метода «бутстреп», практически не представленного в литературе по ИМ.

8.2.3. Имитационное моделирование и метод «бутстреп»

Имитационное моделирование является не только мощным инструментом исследования и оптимизации реальных систем, но может успешно конкурировать с аналитическими методами при решении статистических проблем.

Появившийся в последние десятилетия метод «бутстреп» (ВМ — bootstrap method) служит именно для этих целей. Вначале рассмотрим метод в его первоначальном классическом виде и, отметив недостатки, обратимся к обобщенному методу, предложенному Дадевичем [15].

Предположим, что нас интересует случайная переменная $\theta = \theta(F)$, но функция распределения $F(.)$ не известна. Однако мы располагаем

случайной выборкой $X_1, X_2, ..., X_n$ из $F(.)$. Вопрос сводится к тому, чтобы, используя имеющуюся выборку, определить $\theta(F)$.

Этапы применения ВМ выглядят следующим образом.

ВМ1 — берется первая случайная выборка (с замещением) объемом n из множества $\{X_1, X_2, ..., X_n\}$ и производится оценка среднего значения $\hat{\theta}_1$ по первой выборке.

ВМ2 — берется вторая случайная выборка (с замещением) объемом n из множества $\{X_1, X_2, ..., X_n\}$ и производится оценка среднего значения $\hat{\theta}_2$ по второй выборке.

.
.
.

ВМN — берется N-я случайная выборка (с замещением) объемом n из множества $\{X_1, X_2, ..., X_n\}$ и производится оценка среднего значения $\hat{\theta}_N$ по N-й выборке.

Затем вычисляются значения выборочного среднего и выборочной дисперсии, найденные на этапах ВМ1–ВМN – 1:

$$\bar{\theta} = (1/N)\sum_{i=1}^{N}\hat{\theta}_i; \quad \hat{\sigma}^2 = (1/N-1)\sum_{i=1}^{N}(\hat{\theta}_i - \bar{\theta})^2, \tag{8.31}$$

и с доверительной вероятностью $100(1 - \alpha)$ % можно утверждать, что оценка для θ находится внутри интервала

$$\hat{\theta} = \hat{\theta} \pm \Phi^{-1}(1-\alpha/2)\hat{\sigma}. \tag{8.32}$$

Таким образом, ВМ позволяет получать оценку, основанную на реальной выборке, но восполняемость выборки приводит к вариабельности оценки. Это обстоятельство является большим недостатком метода «будстреп», так как всегда могут появиться в рассматриваемом множестве другие величины, которые приведут к возникновению значений больших, нежели максимальное значение в первоначальной выборке. Поэтому приходится находить эмпирическую плотность вероятности и использовать ее на каждом этапе, что часто оказывается весьма сложным. Мнения, существующие о ВМ, весьма полярны: некоторые считают, что ВМ — хорошая идея; другие считают, что ВМ не может быть использован на практике; третьи убеждены, что ВМ дает возможность получить хорошие результаты для конкретного набора данных, но не может служить основой для создания теории.

Обобщенный ВМ (ОВМ) позволяет получать хорошие статистические результаты, исходя из обобщенного лямбда-распределения или эмпирической плотности вероятности.

Процедура ОВМ при проведении обычной оценки θ через $\hat{\theta} = \hat{\theta}(X_1, ..., X_n)$ сводится к следующему.

ОВМ1 — оцениваем F через $\hat{F}$, оценка $\hat{F}$ всегда должна обладать теми же свойствами, что и истинная функция. Так, например, если известно, что F непрерывна, то и ее оценка должна быть непрерывной.

ОВМ2 — берем N независимых случайных выборок объемом n из $\hat{F}$, тогда:

из выборки $Y_1, Y_2, \ldots, Y_n$ получаем оценку $\hat{\theta}_1$;

из выборки $Y_{n+1}, Y_{n+2}, \ldots, Y_{2n}$ получаем оценку $\hat{\theta}_2$;

.

.

.

из выборки $Y_{(N-1)n+1}, Y_{(N-1)n+2}, \ldots, Y_{Nn}$ получаем оценку $\hat{\theta}_N$.

ОВМ3 — используем оценки, полученные на этапе ОВМ2, для определения средней оценки из выражений (8.31), (8.32).

Исследования, проведенные рядом авторов, показали, что ОВМ всегда дает оценки лучше, чем классический ВМ, так как ВМ является частным случаем ОВМ и практически дает только эмпирическое распределение, используемое на этапе ОВМ1, число реализаций при ИМ не должно быть меньше 500.

Подводя итоги § 8.2, необходимо отметить, что для планирования имитационного эксперимента необходимо иметь начальные представления об основных положениях теории планирования эксперимента, которые не рассмотрены в данном пособии (библиография по этим вопросам весьма обширна и может удовлетворить самым разнообразным требованиям читателя). Это сделано, во-первых, из методических соображений, а во-вторых, из-за нежелания увеличивать объем пособия.

§ 8.3. ОСОБЕННОСТИ ПРОЦЕССОВ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ

Рассмотрим некоторые особенности процесса имитационного моделирования, имеющие непосредственное отношение к статистическим аспектам ИМ. Процессы ИМ можно классифицировать, используя различные признаки классификации, например:

— задание начальных условий прекращения ИМ — условия заданы (прекращаемые испытания) или процесс длится бесконечно долго;

— характер протекания процесса ИМ — переходной или установившийся;

— цель использования процесса ИМ — контрольная (пилотная) или основная.

Задачи, решаемые в процессе:

— исследовательские (может ли в принципе функционировать вновь создаваемая моделируемая система);

— утилитарные (определение средних значений интересующих выходных характеристик имеющейся системы);

— оптимизационные (выбор рационального варианта из ряда альтернатив);

— статистические (уменьшение числа испытаний, повышение точности).

Можно предложить и другие признаки классификации, но названные представляются наиболее значимыми и будут рассмотрены ниже и частично уже рассмотрены в § 8.2. Автор заранее признателен тем, кто предложит дополнительные признаки классификации и вступит в конструктивную дискуссию.

8.3.1. Типы процессов ИМ

Задание начальных условий

Все примеры, рассмотренные до настоящего момента, относятся к типу *прекращаемых* испытаний. По определению А. Лоу и Д. Келтона, данному в книге, считающейся классикой литературы по ИМ [16], прекращаемыми испытаниями называются такие, при которых определение выходных характеристик производится на интервале $[0, T_c]$, где T_c является моментом времени появления обусловленного события. Обусловленными событиями могут быть:

— случайные моменты времени, соответствующие наступлению 100 выходов (терминирований) транзактов из системы (см. пример 6.4);

— заданные моменты времени, в течение которых определяются характеристики системы (см. пример 6.3);

— заданное число транзактов, обслуженных по одному разу для определения случайного времени загрузки ЦПУ (см. пример 8.3).

Необходимо понимать разницу между оговоренным (физическим) и заданным прекращением. Физическое прекращение оговаривается расписанием работы моделируемой системы (работа магазина с 10.00 до 20.00, обслуживание клиентов банка с 9.00 до 14.00 и т. д.). Ряд систем работает круглосуточно. Однако и первые, и вторые могут моделироваться как прекращаемые испытания при задании начальных условий, описанных выше. При необходимости сбора статистических данных по нескольким независимым прогонам необходимо четко оговаривать начальные условия моделирования прекращаемых испытаний реальной системы. Так, например, при получении сред-

него значения времени, затраченного на контроль 100 телевизоров (см. пример 6.4), необходимо оговорить идентичные начальные условия: начало работы системы при отсутствии телевизоров, оставшихся непроверенными с предыдущего дня; отсутствие телевизоров на посту контроля и регулировки и т. п.

Необходимо также различать модели с изменяемыми и с неизменными входными данными. Например, задав неизменной функцию распределения появления транзактов (экспоненциальная функция распределения), можно варьировать темп прихода транзактов в разные моменты реального времени (функция распределения Пуассона). Так, моделируя работу банка, необходимо учитывать пиковые нагрузки и спады в течение рабочего дня. Такое варьирование входных данных потребует переосмысления работы модели и внесения в МФ дополнительных операторов ЯИМ. Для компенсации изменения темпа появления транзактов необходимо будет предусмотреть введение дополнительных ресурсов в виде устройств или памятей.

Задаваясь коэффициентом использования ресурсов, исследователь может прогнозировать наиболее рациональное поведение системы. Очевидно, что моделирование систем с изменяемыми входными данными и меняющимися ресурсами требует от исследователя не только освоения правил моделирования, но и ясного представления о процессах, которые необходимо моделировать. Поэтому ограничимся сказанным и предоставим читателю свободу действий самостоятельно или с привлечением консультантов.

Характер протекания процесса ИМ

В тех случаях, когда условия прекращения процесса ИМ не заданы, можно говорить о двух стадиях процесса ИМ — неустановившейся и установившейся. По Лоу и Келтону: «Установившийся процесс ИМ — это такой процесс, при котором определение выходных характеристик происходит в течение времени моделирования, стремящегося к бесконечности» [16]. Поскольку обусловленные события при этом не используются, то реально моделирование длится до того момента, когда выходные характеристики стабилизируются, т. е. их изменение во времени находится в интервале значений, определяющих заданную точность. При этом считается, что и ресурсы системы не меняются. Можно назвать достаточно много примеров подобного моделирования. Диаграмма процесса отладки и установившейся работы какого-либо нового производства представлена на рис. 8.6.

В начале моделирования (начало отладки реальной системы) производительность системы равна нулю. После того как система начнет функционировать, производительность нарастает. Очевидно, что

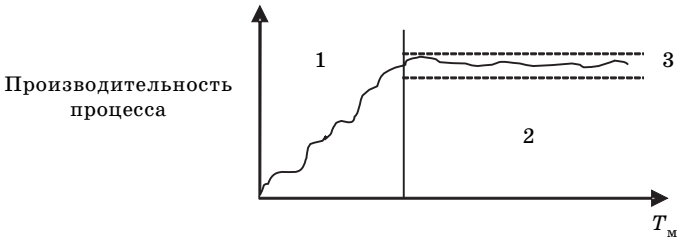


Рис. 8.6. *Стадии производственного процесса:*
1 — область переходного процесса; 2 — область установившегося процесса; 3 — допуск на изменение выходной характеристики

производительность не может быть представлена плавной кривой, так как воздействующие внешние и внутренние факторы приведут к стохастическому изменению мгновенных характеристик производительности, в том числе и в области установившегося процесса. Процесс считается установившимся, когда выходные значения производительности окажутся на уровне ожидаемых значений и не будут выходить за нижнюю границу допуска. Уход за верхнюю границу говорит о потенциальных возможностях системы, которые надо выявить и закрепить. Область переходного процесса надо пытаться минимизировать, но это относится к сфере конкретных управленческих и инженерных решений, и пути их реализации в пособии не рассматриваются. Более того, изучение особенностей установившегося состояния:

— будет ли система находиться сколь угодно долго в установившемся состоянии;

— есть ли причины, могущие вывести систему из этого состояния (отказы оборудования, изменение внешних условий и т. п.);

— как оценить границу между переходным и установившимся состоянием, — тоже не проводится, хотя на все эти вопросы можно ответить с помощью процесса ИМ конкретной системы при задании конкретных начальных условий.

Цели использования процесса ИМ

Достаточно подробно рассмотрены вопросы оценки продолжительности процесса ИМ (см. § 8.2). На практике, тем не менее, часто используется разделение процесса ИМ на две стадии: пилотного (контрольного) прогона и основного процесса ИМ. Пилотный прогон позволяет, во-первых, судить об адекватности и работоспособности модели, а во-вторых, оценить примерное значение стандартного отклонения. Напомним, что при объеме выборки менее 30 реализаций мы имеем дело с распределением Стьюдента, которое сходится к нормальному распределению при объеме выборки, стремящемся к 30. Поэто-

му принято выборку объемом менее 30 считать малой, к членам выборки предъявлять требование случайности, независимости и нормальности распределения; выборку объемом 30 и более считать большой, а ее члены могут иметь любую функцию распределения при сохранении требований случайности и независимости.

Далее алгоритм поведения сводится к следующему: если пилотный прогон не выявил никаких замечаний к логике работы модели, то его используют для улучшения статистических оценок. При этом по реализациям пилотного прогона ($n \leq 10$) определяются начальные оценки и необходимый объем выборки, а затем проводятся дополнительные реализации, число которых определяется заданными правилами, и по сумме обоих прогонов оценивается стандартное отклонение, удовлетворяющее заданной точности. Необходимым условием при этом является независимость первой и второй выборок. Естественно, что при этом приходится переназначать исходную позицию ГСЧ, чтобы исключить корреляцию потоков БСВ. Для этих целей в GPSS/H используется ОУ RMULT (см. п. 5.1.2, К7 и прил. 2). Положение ОУ RMULT в МФ зависит от того момента времени, когда пользователь предусматривает его исполнение. Так, если некоторые ГСЧ должны быть установлены в какое-то положение до начала ИМ, то ОУ RMULT ставится в модуле 1 управления и описания, чтобы его исполнение произошло до начала моделирования; если же положение ГСЧ переназначается в процессе ИМ, то ОУ RMULT ставится в модуле 3 управления, между ОУ START, запускающими новый прогон. Для того чтобы проиллюстрировать логику использования пилотного прогона, обратимся вновь к примеру 6.2, введя в него все элементы учета статистических оценок.

Пример 8.4. Использование пилотного прогона модели инструментальной кладовой

1. Постановка задачи.

В примере 6.2а оценивается ожидаемое число механиков с точностью ±0.5 механика с доверительной вероятностью 90 %. Данные пилотного прогона, состоящего из 10 реализаций, полученные с помощью МФ примера 6.2а, приведены в табл. 8.20.

Первая оценка предсказывает необходимость проведения 44 реплик (на самом деле до округления 43.25), т. е. во второй выборке необходимо провести еще 34 реплики. Изменим МФ примера 6.2а для получения 34 независимых реплик. Значение ГСЧ1 на 10-й реплике было равно 102 956, слегка изменим это значение и в операнде А ОУ RMULT введем значение 105 000. Поскольку пилотный прогон показал адекватность модели, то второй прогон из 34 реплик можно осуществлять в пакетном режиме, а окончательную оценку производить на основе всех 44 реплик (10 + 34).

Таблица 8.20. *Среднее число механиков в очереди при разных ДО*

Номер реплики	Среднее число механиков	
	ДО FCFS	ДО SPT
1	1.705	1.501
2	3.627	2.033
3	2.780	2.456
4	5.502	7.584
5	3.494	2.890
6	4.114	3.060
7	4.704	0.967
8	4.326	5.536
9	1.362	2.489
10	3.737	4.163
Выборочное среднее $\bar{X}$	3.535	3.268
Выборочное отклонение s	1.286	1.99

2. Изменения, введенные в МФ.

1. В модуль 1 введен ОУ RMULT, операнд А которого задает новое начальное положение ГСЧ1 еще до начала моделирования.

2. Введен ОУ PUTPIC (см. п. 5.1.2 и прил. 2), позволяющий выделить отдельные строки итогового отчета.

3. Определения.

Определения идентичны примеру 6.2 и поэтому не повторяются.

4. Модельный файл.

```
*    Модуль описания
          SIMULATE                  Пример 8.4 (модификация
                                    примера 6.2а). Инструменталь-
                                    ная кладовая
                                    ДО SPT
*                                   Временная дискрета: 1 с
          INTEGER   &I              индекс петли управления
          RMULT     105000          установка ГСЧ1 в положение
                                    105000
*    Модуль исполнения
*                   Фрагмент 1      1-й тип механиков                 *
          GENERATE  420,360,,,5     приход механиков 1-го типа
*                   (Приоритет 5 )
          QUEUE     TOOLWAIT        постановка в очередь
          SEIZE     CLERK           запрос на обслуживание
          DEPART    TOOLWAIT        выход из очереди
          ADVANCE   300,90          обслуживание
          RELEASE   CLERK           освобождение кладовщика
          TERMINATE 0               уход из кладовой
*                   Фрагмент 2      2-й тип механиков                 *
          GENERATE  360,240,,,10    приход механиков 2-го типа
```

```
*                       (Приоритет 10 )
          QUEUE         TOOLWAIT    постановка в очередь
          SEIZE         CLERK       запрос на обслуживание
          DEPART        TOOLWAIT    выход из очереди
          ADVANCE       100,30      обслуживание
          RELEASE       CLERK       освобождение кладовщика
          TERMINATE     0           уход из кладовой
*                       Фрагмент 3  Хакт управления                 *
          GENERATE      28800       время моделирования 8 ч
          TERMINATE     1           уменьшение СС на 1,
*                                   окончание движения транзактов
*    Модуль управления
          DO            &I=1,34,1   управление проведением 34 реплик
          START         1           старт очередной реплики
          PUTPIC        LINE=3,FILE=SYSPRINT,(&I)
0═══════════════════════════════════════════════════════════════
The Postsimulation Report above is for supplementary replication *.
═══════════════════════════════════════════════════════════════
          CLEAR                     очистка данных для очередной
                                    реплики
     ENDDO                          завершение петли управления
     END                            окончание процесса ИМ
```

5. Итоговый отчет.

Отчет во многом повторяет данные примера 6.2, поэтому приведем только значения среднего и стандартного отклонения по 44 репликам: $\bar{X} = 2.601$, $s = 1.62$, что полностью соответствует заданной доверительной вероятности 90 %.

6. Обсуждение.

Найденное по суммарным репликам значение стандартного отклонения на 20 % меньше полученного в пилотном прогоне (1.62 против 1.99). Полученная точность оценки также повышена, задана 0.5, получена 0.4, а именно также на 20 %, что совпадает с теоретическими предположениями.

8.3.2. Использование потоков, дополняющих БСВ (антитез)

Общая идея метода антитез

При проведении процесса ИМ одним из мощных методов уменьшения дисперсии при одинаковом объеме выборки или уменьшения объема выборки без потери точности является использование встречных или дополняющих потоков БСВ, в дальнейшем будем называть их *антитезами*.

Теория математической статистики жестко связывает уменьшение стандартного отклонения с объемом выборки; так, увеличение

объема выборки в 4 раза приводит лишь к двукратному уменьшению стандартного отклонения.

Однако использование антитез позволяет нарушать каноны теории. Существует ряд способов понижения дисперсии при проведении ИМ, перечислим их без проведения сравнительного анализа (заинтересованному читателю можно рекомендовать работу [11] или монографию И. М. Соболя «Метод Монте-Карло»).

Стратифицированная выборка — когда данные разбиваются на непересекающиеся страты, попадание средней оценки в одну из страт позволяет не рассматривать другие. Мощность оценки будет зависеть от абсолютной разности между средними значениями страт. Безотносительно к ИМ, автором исследовано применение метода Махаланобиса к оценке расстояний на множестве кластеров. Очевидно, что эта методика может быть применена и к уменьшению дисперсии процесса ИМ.

Значимая выборка — когда вводится новая функция плотности, задающая большие веса значениям тех параметров, которые вносят наибольший вклад в значение выходных характеристик на основе принципа Парето 80:20. Автор в ряде своих публикаций [3, 4] ввел понятие коэффициента значимости, которое с успехом применяется при аналитических расчетах и может быть полностью использовано в процессе ИМ.

Начальная выборка, когда исследователь располагает данными пилотных прогонов, данными по испытаниям аналогичных систем и т. д. Такое моделирование называется моделированием, управляемым по предыстории.

Комбинированная выборка — когда используются аналитические модели, например формула Литтла $L = \lambda T$, связывающая ожидаемое число заявок в системе (транзактов) L со временем их пребывания в системе T при параметре потока заявок λ. Средством понижения дисперсии в данном случае является использование известных значений параметра потока.

Однако самым эффективным и удобно применяемым в GPSS/H, по мнению автора, является метод антитез. Название введено Т. Шрайбером [17].

Рассмотрим суть метода антитез. Формируются пары реплик и по значениям этих реплик вычисляется среднее значение пары. Вместо обычного требования независимости реплик используется отрицательная корреляция, т. е. максимально возможному значению одного члена пары соответствует минимально возможное значение, допустимое в этой серии опытов, и наоборот. Проиллюстрируем эту идею на примере бросания двух игральных кубиков (табл. 8.21). Очевид-

Таблица 8.21. Пары антитез при бросании двух кубиков.

Прямой результат	Антитеза	Частота	Среднее пар
2	12	1/36	7
3	11	2/36	7
4	10	3/36	7
5	9	4/36	7
6	8	5/36	7
7	7	6/36	7
8	6	5/36	7
9	5	4/36	7
10	4	3/36	7
11	3	2/36	7
12	2	1/36	7

но, что максимальным значением в данном примере является 12 (выпадание двух граней по 6), а минимальным — 2 (выпадание двух граней по 1), и тогда эти значения составят первую пару. Логика представления других пар аналогична.

В силу симметрии частота появления результатов в паре одинакова, сумма результатов в паре также одинакова, а следовательно, и среднее значение пар одинаково и равняется 7. Теперь предположим, что мы хотим промоделировать бросание кубиков на ЭВМ. Можно ли заменить 10 реплик проведением 5 реплик с использованием антитез? Ответ очевиден из табл. 8.21. Более того, можно утверждать, что в этом симметричном случае (при условии правильности кубиков) вполне достаточно проведения только одной реплики с использованием антитез! Табл. 8.21 должна привести грамотного статистика в состояние тихой ярости — статистические испытания проведены, а дисперсия равна нулю! Ведь при проведении независимых испытаний дисперсия равна 35/6, т. е. почти 6. Очевидно, что в реальных ситуациях, а не в таком рафинированном примере, дисперсия не может быть равна нулю по определению, так как добиться абсолютной отрицательной корреляции вряд ли удастся. Лоу и Келтон показали [16], что возможно снижение дисперсии на 60–65%, однако и такого снижения вполне достаточно для практических целей, учитывая сокращение машинного времени и снижение дисперсии при одновременном уменьшении объема необходимой выборки. Отсюда следует, что можно использовать входные данные с большей дисперсией, чем это допустимо в случае независимости членов выборки. Интересно сравнить результаты моделирования с использованием антитез с результатами использования обычного метода при независимости членов выборки. Для заданного числа реплик объем выборки с использованием антитез вдвое меньше стандартного метода.

Причем машинное время на создание 20 независимых БСВ и 10 пар антитез одинаково, а сам процесс моделирования почти вдвое короче.

Однако вспомним постулаты теории математической статистики.

1. В случае малых выборок дальнейшее уменьшение выборки приводит к возрастанию t-статистики и пропорциональному расширению доверительного интервала.

2. Ширина доверительного интервала обратно пропорциональна корню квадратному из размера выборки, поэтому уменьшение объема выборки вдвое вроде бы никак не может привести к снижению дисперсии.

Рассматривая уменьшение выборки (в случае малых объемов выборки) вдвое при использовании антитез, можно заметить, что увеличение t-статистики оказывает влияние на уменьшение дисперсии, но не такое значительное, как при применении независимых членов выборки. Этот вопрос является предметом изучения специалистов по статистике и не рассматривается в пособии, однако читатель должен сделать важный для него вывод: применение антитез способствует снижению дисперсии, а насколько, — зависит от конкретных систем. Ниже мы покажем использование метода антитез на примере модели инструментальной кладовой специально для того, чтобы можно было сравнить результаты.

Теперь рассмотрим идею использования антитез на примере одноканальной, однофазной СМО. Предположим, что время поступления следующего транзакта меньше среднего времени, а время обслуживания предшествующего транзакта больше среднего, тогда поступающий транзакт будет ожидать обслуживания больше среднего времени ожидания. Тогда в паре антитез для второго члена все будет наоборот: время поступления больше среднего, время обслуживания предыдущего меньше среднего, а время ожидания становится меньше среднего. Следовательно, время ожидания больше среднего будет уравновешиваться временем ожидания меньше среднего и среднее время ожидания в паре будет близко к предсказываемому времени ожидания. В результате выборочная дисперсия таких пар уменьшается и можно рассчитывать на сужение доверительного интервала. Точное математическое доказательство приведено А. Лоу [16].

Способ получения антитез

Задачей выработки антитез является стремление получить абсолютную отрицательную корреляцию выходных значений при использовании отрицательной корреляции БСВ в интервале (0, 1). Будем полагать с высокой степенью достоверности, что БСВ отрицательно коррелированы с их дополнением до 1. Например, дополнением для 0.25 является 0.75, для 0.03 соответственно 0.97 и т. д. Значение,

полученное с оговоренного ГСЧ, используется как прямое значение, а затем оно же преобразуется в дополнение, чтобы использоваться в обратном потоке инверсированных чисел. Очевидно, что получение выходных значений с отрицательной корреляцией при отрицательной корреляции БСВ справедливо для симметричных функций распределения; для несимметричных, к каким относится экспоненциальное распределение, коэффициент корреляции равен −0.64. Отсюда следует заключение, что выходные значения не обладают абсолютной отрицательной корреляцией даже при абсолютной отрицательной корреляции БСВ. Преобразование входных отрицательно коррелированных БСВ в выходные отрицательно коррелированные значения достаточно длительный и сложный процесс, который нами не рассматривается. Рассмотрим только внешние команды управления, с помощью которых GPSS/H производит получение антитез. Случайное число-антитеза получается с любого номера ГСЧ при использовании ОУ RMULT (см. § 5.2 и прил. 2) записью в одном из операндов числа со знаком минус, например:

а) прямые числа RMULT 25000, 50000;

б) антитезы RMULT −25000, −50000.

В примере заданы установки для ГСЧ1 и ГСЧ2 в случае а) и для антитез с тех же генераторов в случае б).

Необходимо подчеркнуть, что использование прямого потока БСВ и антитез должно быть обязательно согласовано. Так, если какое-либо значение с любого ГСЧ используется для получения времени поступления транзакта в прямом потоке БСВ, то эквивалент (антитеза) должен использоваться для описания этого же транзакта во встречном потоке. В случае независимости реплик мы использовали один и тот же генератор для получения и времени поступления и времени обслуживания. Если то же самое сделать при использовании антитез, получится несинхронизированное наступление событий и задача становится некорректной (попробуйте сами разобрать эту ситуацию). Поэтому при использовании метода антитез обязательным условием является применение разных ГСЧ для получения времени поступления и времени обслуживания, при наличии нескольких ОБ GENERATE и ОБ ADVANCE для каждого из них ***обязательно*** используется свой ГСЧ. Выполнение этого непременного условия позволяет осуществить синхронизацию потоков событий.

Подобное разделение ГСЧ необходимо неукоснительно соблюдать при задании встроенных функций типа RVEXPO и аналогичных, а также задаваемых ОО FUNCTION дискретных и непрерывных функций. Следует учитывать, что назначение генератора не всегда может решить все проблемы. Представим, например, вполне реальный слу-

чай, когда транзакт не может дожидаться в очереди и покидает систему не обслуженным. Метод антитез в этом случае может привести к появлению ошибки, и такую возможность необходимо учитывать в структуре модели.

При использовании метода антитез стартовая позиция ГСЧ для прямого и обратного потока должна быть одинаковой. Однако в случае прекращаемых испытаний число входов в систему бывает больше, чем контрольное число терминированных транзактов, и таким образом, итоговое значение ГСЧ не будет являться стартовым для новой пары антитез. Решить эту проблему можно, установив стартовое значение ГСЧ до начала каждой реплики (пример 8.5). Это достигается путем введения в поле операндов ОУ RMULT арифметических выражений, которые для антитез заключаются в скобки, перед скобкой ставится знак минус.

Напомним, что при постоянстве использования ОУ RMULT после запятой операнда ставится знак (_), свидетельствующий о продолжении записи, тогда при компиляции ищется следующая запись для этого же ОУ. Это объясняется и длиной записи в поле операндов, чтобы не делать строку слишком большой и оставить место для комментариев, а также с целью достижения большей наглядности записи. Составление арифметических выражений в поле операнда не оговорено какими-либо правилами и зависит от опыта и предпочтений пользователя, а также от навыка написания программ в других языках программирования, поскольку правила написания арифметических выражений идентичны Фортрану, Паскалю, СИ++, Бейсику и т. п.

Пример 8.5. Применение антитез в модели инструментальной кладовой

1. Постановка задачи.

Пример инструментальной кладовой используется уже в четвертый раз, что отвечает методическим целям сравнения эффективности применяемых методов: выбора оптимальной дисциплины обслуживания (пример 6.2); автоматизации процесса ИМ при получении ряда последовательных реплик в одном прогоне (пример 6.2а); использования пилотного прогона для получения величины выборки (пример 8.3) и, наконец, использования метода антитез. В рассматриваемом примере будем использовать 22 пары антитез вместо 44 независимых реплик (пример 8.4). Генераторы случайных чисел ГСЧ1 и ГСЧ2 (в модельном файле RN1, RN2 соответственно) используются для времени прихода и времени обслуживания механиков 1-го типа, а ГСЧ3 и ГСЧ4 — для таких же времен механиков 2-го типа. Для задания указанных времен будем пользоваться непрерывными четырьмя функциями, описываемыми в модуле 1.

В результате моделирования необходимо сравнить полученный результирующий доверительный интервал с интервалом примера 8.4.

2. Допущения, сделанные в модели.

В модели, по сравнению с предыдущими, вводится описание двухточечных непрерывных функций, описывающих времена прихода и обслуживания механиков обоего типа. Также вводится раздельное использование ГСЧ и переназначение их стартовых позиций с помощью целочисленной АМП. Учитывая, что в модели существуют два потока чисел, вводятся соответственно два ОУ PUTPIC.

3. Определения, вводимые в модели.

Поскольку кроме функций никаких других нововведений нет, таблица определений не приводится.

4. Модельный файл.

```
* Модуль описания
                SIMULATE                Пример 8.5 (модификация
                                        примера 8.4). Инструмен-
                                        тальная кладовая
                                        ДО SPT
*                                       Временная дискрета: 1 с
                INTEGER     &I          индекс петли управления
                UNLIST      CSECHO      не отображать операторы
                                        управления
  TYPE1IAT      FUNCTION    RN1,C2      время прибытия механиков
                                        1-го типа
  0,60/1,780
  TYPE1ST       FUNCTION    RN2,C2      время обслуживания
                                        механиков 1-го типа
  0,210/1,390
  TYPE2IAT      FUNCTION    RN3,C2      время прибытия механиков
                                        2-го типа
  0,120/1,600
  TYPE2ST       FUNCTION    RN4,C2      время обслуживания
                                        механиков 2-го типа
  0,70/1,130
* Модуль исполнения
*               Фрагмент 1              1-й тип механиков *
                GENERATEFN(TYPE1IAT),_
                            ,,,5        приход механиков 1-го типа
                * ( Приоритет 5 )
                QUEUE       TOOLWAIT    постановка в очередь
                SEIZE       CLERK       запрос на обслуживание
                DEPART      TOOLWAIT    выход из очереди
                ADVANCE     FN(TYPE1ST) обслуживание механиков
                                        1-го типа
                RELEASE     CLERK       освобождение кладовщика
                TERMINATE   0           уход из кладовой
                * Фрагмент 2            2-й тип механиков      *
```

```
                GENERATE   FN(TYPE2IAT),_
                           ,,,10            приход механиков
                                            2-го типа
                           * ( Приоритет 10 )
                QUEUE      TOOLWAIT         постановка в очередь
                SEIZE      CLERK            запрос на обслуживание
                DEPART     TOOLWAIT         выход из очереди
                ADVANCE    FN( TYPE2ST)     обслуживание
                RELEASE    CLERK            освобождение кладовщика
                TERMINATE  0                уход из кладовой
                * Фрагмент 3                Хакт управления *
                GENERATE   28800            время моделирования 8 ч
                TERMINATE  1                уменьшение СС на 1,
*                                           окончание движения
                                            транзактов
*  Модуль управления
                DO         &I=1,22,1        управление проведением
                                            22 антитез
                  RMULT    99000+ 1000*&I,_ I-я прямая реплика
                                            с RN1
                           199000+1000*&I,_ I-я прямая реплика
                                            с RN2
                           299000+1000*&I,_ I-я прямая реплика
                                            с RN3
                           399000+1000*&I   I-я прямая реплика
                                            с RN4
                  START    1                старт очередной прямой
                                            реплики
                  PUTPIC LINE=3,FILE=SYSPRINT,(&I)
0=================================================================
 The Report above is for non-antithetic replication number * .
=================================================================
                  CLEAR                     очистка данных для
                                            реплик-антитез
                  RMULT    -(99000+1000*&I), _I-я реплика-антитеза
                                            с RN1
                           -(199000+1000*&I),_ I-я реплика-антитеза
                                            с RN2
                           -(299000+1000*&I), _ I-я реплика-антитеза
                                            с RN3
                           -(399000+1000*&I)  I-я реплика-антитеза
                                             с RN4
                  START    1
                  PUTPIC LINE=3,FILE=SYSPRINT,(&I)
0=================================================================
 The Report above is for antithetic replication number * .
```

CLEAR	очистка данных для прямых реплик
ENDDO	завершение петли управления
END	окончание процесса ИМ

5. Итоговый отчет.

Весь листинг отчета занимает много места и поэтому не приводится, в табл. 8.22 из итогового отчета приведены величины пар антитез и их среднее значение, а в табл. 8.23 сведены итоговые данные по примерам 8.4 и 8.5.

Таблица 8.22. *Значения прямых и обратных потоков чисел*

№ пар чисел	Прямой поток	Обратный поток	Среднее пары
1	0.973	3.586	2.280
2	2.125	1.049	1.587
3	0.856	4.935	2.896
4	2.244	2.364	2.304
5	3.612	1.163	2.388
6	5.878	0.880	3.379
7	1.888	1.101	1.495
8	0.635	5.939	3.287
9	1.216	2.462	1.839
10	1.710	1.920	1.815
11	1.569	1.826	1.698
12	3.392	1.802	2.597
13	1.104	3.644	2.374
14	2.067	1.340	1.704
15	1.883	2.649	2.266
16	0.789	5.776	3.283
17	3.063	0.973	1.018
18	1.088	2.933	2.011
19	2.715	1.368	2.042
20	2.422	1.131	1.777
21	1.450	2.800	2.125
22	2.060	1.281	1.671

Таблица 8.23. *Сравнение данных двух примеров*

Пример	Выборочное среднее	Выборочное отклонение	Размах 90 % интервала
Пример 8.4	2.601	1.62	(2.2; 3.0)
Пример 8.5	2.219	0.564	(2.0; 2.4)

6. Обсуждение.

Можно отметить следующие отличительные особенности МФ примера 8.5.

1. Для уменьшения итогового отчета в МФ введен ОО UNLIST, который запрещает вывод на печать ОУ (см. прил. 3).

2. Операнды А ОБ GENERATE, ADVANCE представляют собой непрерывные функции, описываемые в модуле 1 и используемые для представления времен прибытия и обслуживания.

3. ОУ RMULT размещен в петле управления для того, чтобы задавать стартовые позиции для ГСЧ1–ГСЧ4 для всех 22 пар прямых чисел и их антитез.

Символ (_) указывает на протяженность использования условия на все 4 ГСЧ, такой же символ используется в ОБ GENERATE для сокращения длины записи.

Сравнение данных (табл. 8.23) позволяет заключить, что при применении метода антитез стандартное отклонение составляет всего 35 % от значения примера 8.4, и это при условии сокращения реплик с 44 до 22. Доверительный интервал при сохранении значения доверительной вероятности сократился при этом вдвое.

Такое улучшение статистических характеристик получено при минимальных усилиях со стороны пользователя и минимальных затратах машинного времени на создание антитез, причем суммарное время моделирования на пилотный прогон и исполнение 34 реплик оказалось на 30 % больше времени моделирования модели примера 8.5 и в целом составило 0.36 с на ПК Р-4.

§ 8.4. ВЫБОР НАИЛУЧШЕЙ АЛЬТЕРНАТИВЫ В ПАРЕТО-ОПТИМАЛЬНОМ МНОЖЕСТВЕ

Решение проблемы рационального построения исследуемой системы является сложной, многоэтапной и многокритериальной задачей. Многие авторы занимались и продолжают заниматься решением ее отдельных аспектов. В принципе, для решения оптимальной задачи необходимо иметь неограниченные ресурсы, тогда можно решать двуединую задачу оптимизации: либо максимизировать значения выходных характеристик, либо, сохраняя выходные значения на заданном уровне, минимизировать ресурсы. Чаще всего, на практике невозможно располагать неограниченными ресурсами и приходится решать задачу максимизации выходных характеристик системы при ограниченных ресурсах R. Такие решения не корректно называть оптимальными. Достаточно вспомнить строки А. Вознесенского: «Это почти неподвижности мука / Мчаться куда-то со скоростью звука / Зная, наверно, что есть уже где-то,/ Некто летящий со скоростью света». Поэтому, имея ограниченные ресурсы, правильнее говорить о рациональных, или субоптимальных, решениях, которые и будем рассматривать. Применяя методы ИМ, можно с помощью метода бенчмаркинга селектировать альтернативные варианты построения системы, оценивая их по возрастанию выходных характеристик. Под методом бенчмаркинга понимается процесс сравнительно-

го анализа разных (чаще всего двух) концепций, компонентов, подсистем, процессов. Цель бенчмаркинга — количественно оценить самый лучший вариант среди рассмотренных альтернатив. В основе любого сравнения лежит принцип попарного сопоставления, поэтому в скобках, фразой выше, подчеркнуто, что альтернатив две, худшая отвергается, а лучшая сравнивается со следующей и т. д. Наконец, выбрав рациональный вариант, пытаться улучшать уже именно его за счет проектирования параметров и допусков на них. На улучшение какого-либо параметра расходуется определенный ресурс, при большом числе параметров чаще всего выбирают методику, основанную на методах теории планирования эксперимента или робастного проектирования. При этом меняют какой-либо параметр до исчерпания ресурса R или до физически допустимого предела изменения этого параметра при неизменных других. Каждому варианту сопоставляется значение выходного параметра. Назовем эту вектор-характеристику качеством целевого функционирования Q_f [3, 4], тогда возрастание Q_f отвечает цели проектирования. Если проводить сравнение двух альтернатив, то альтернатива Q_f^1 доминирует над альтернативой Q_f^2, если превышено значение хотя бы по одному параметру Q_f. Отношения доминирования Θ могут быть нескольких типов:

— отношение Слейтера (строгое доминирование), когда $Q_f \Theta R$ выполняется тогда и только тогда, когда $Q_f^i > Q_f^j$ при всех значениях $i, j = 1, 2, ..., n$;

— отношение Парето, когда $Q_f \ \Theta \ R$ выполняется тогда и только тогда, когда $Q_f^i \geq Q_f^j$ при всех значениях $i, j = 1, 2, ..., n$.

Чаще всего используют отношение Парето. Очевидно, что изменение разных параметров никогда не приведет вектор Q_f в одну точку пространства, в котором в результате многих попыток образуется множество субоптимальных точек, составляющих Парето-оптимальное множество. Попадание в это множество позволяет проводить дальнейшее отыскание рационального варианта методами, уже рассмотренными в гл. 8.

Напомним, что процесс ИМ не позволяет давать точечных оценок, а оценивает только средние значения, поэтому выбор лучшего варианта в Парето-оптимальном множестве должен основываться на статистических оценках, а именно на оценивании дисперсии.

8.4.1. Выбор из двух альтернатив в процессе ИМ

В примере 6.2 производился выбор между двумя дисциплинами обслуживания, в результате которого число ожидающих обслуживания механиков было минимальным. Однако при этом получались

не абсолютные оценки, а средние. Один из путей решения задачи нахождения различия между несколькими ДО статистическим путем — это построение доверительного интервала для разности длин ожидаемых очередей при разных дисциплинах обслуживания.

Сравнение двух альтернатив с некоррелированными t-парами

Прямой метод сравнения двух некоррелированных альтернатив, когда оценивается разность ожидаемых значений, состоит в следующем:

1) получается равное число реплик n для обеих альтернатив;

2) составляются пары реплик одного номера из каждой альтернативы;

3) вычисляется разность для каждой пары, а затем определяются среднее значение и стандартное отклонение разности;

4) вычисляется доверительный интервал для разности при разных значениях доверительной вероятности. Если доверительный интервал не включает точку нуля, то можно предполагать (при выбранной значимости в процессе распознавания гипотез), что выходные величины будут отличаться от ожидаемых величин. Если интервал включает точку нуля, то есть основание полагать, что выходные значения не отличаются от ожидаемых величин.

Будем считать, что реплики независимы не только в рамках одной альтернативы, но и между альтернативами. Если на основе шага 4 выбираем доверительную вероятность 95 % , то это значит, что гипотеза о том, что разницы между двумя ожидаемыми значениями нет, будет отвергаться на 5% -м уровне значимости. Дальнейших комментариев не делаем, считая, что читатели знакомы с основами теории распознавания статистических гипотез. Кроме идентичности реплик необходимо помнить, что должно выполняться требование идентичности функций распределения. Напомним, что при малых выборках разность между парами должна иметь нормальное распределение, и тогда для вычисления доверительного интервала можно применить распределение Стьюдента. В нашем случае сравнение двух некоррелированных выборок можно заменить оценкой одной выборки, включающей разность этих выборок.

Проиллюстрируем построение доверительного интервала для некоррелированных пар при разных ДО, подчиняющихся t-распределению (табл. 8.24). Характеристикой, представляющей интерес, является среднее число механиков, ожидающих обслуживания.

Данные получены для примера 6.2; поскольку БСВ получались с одного ГСЧ, то для избежания корреляции 10 реплик ДО FCFS исполнялись с помощью ОБ RMULT 125000, а 10 реплик ДО SPT — с помощью ОБ RMULT 150000. Внизу таблицы приведены данные для разных значений доверительной вероятности. Первые два значения

Таблица 8.24. Среднее число механиков в очереди

№ реплики	Средняя длина очереди при разных ДО		Разность
	FCFS	SPT	
1	3.826	1.326	2.500
2	1.529	3.668	−2.139
3	1.805	0.737	1.068
4	4.140	1.619	2.521
5	2.116	3.572	−1.456
6	5.010	1.290	3.720
7	5.411	2.723	2.688
8	1.974	2.229	−0.255
9	6.585	1.395	5.190
10	2.550	1.346	1.204
Выборочное среднее		1.504	
Выборочное стандартное отклонение		2.293	
Выборочный интервал 80 %		[0.50, 2.50]	
Выборочный интервал 90 %		[0.18, 2.83]	
Выборочный интервал 95 %		[−0.14, 3.14]	

не включают нуль, подтверждая уверенность, что очередь по ДО SPT будет короче. Последний 95%-й интервал включает нуль, что означает, что на уровне значимости 5 % гипотеза об отсутствии разницы между длинами очередей будет отвергнута. Эта ситуация будет рассмотрена в упражнениях (см. п. 8.4.2 и § 8.5).

Существует много различных методов построения доверительного интервала для разности ожидаемых значений двух независимых, а следовательно, не коррелированных выборок из разных генеральных совокупностей. Применение того или иного метода зависит от конкретной постановки задачи. Рассмотрим кратко, не вдаваясь в аналитические подробности, три возможных метода:

1) классическое представление двух t-распределенных выборок;

2) классическое представление двух z-распределенных выборок;

3) доверительный интервал Вельша.

Для сравнения этих методов можно предложить два критерия:

1) должны ли выборки из двух генеральных совокупностей иметь одинаковый объем;

2) должны ли быть равны дисперсии двух генеральных совокупностей?

Метод некоррелированных пар, рассмотренный выше, требует равенства объема выборок, не предусматривая при этом никаких ограничений на величину дисперсии. Очевидно, что равенство объемов выборки достаточно просто осуществляется на практике, когда моделируются обе альтернативы. Предположим, однако, что одна аль-

тернатива относится к уже существующей системе, данные по которой получить трудно, дорого или долго, а вторая относится к изучению изменений в существующей системе и моделируется непосредственно. В этом случае выборка по первой альтернативе является весьма малой и невозможно набрать информацию для корректного принятия решения.

При классическом представлении *двух t-распределенных выборок* не требуется равенства объемов выборок, но обязательно требуется равенство дисперсий при малых выборках ($n \leq 30$) и нормальное распределение выходных величин. Требование равенства дисперсий ограничивает применение этого метода. В любом случае приходится прибегать к ухищрениям типа комбинирования двух выборочных дисперсий и нормировки их относительно объема выборок. Центр доверительного интервала будет находиться посередине значения разности выборочных дисперсий.

При значении объема каждой выборки хотя бы 30 используется классическое представление *двух z-распределенных выборок*, когда не требуется равенства дисперсии двух генеральных совокупностей и нормальности распределения выходных величин. Это представление хорошо работает для больших выборок, когда исследователь не связан необходимостью уменьшать объем выборки.

Наконец, Велшем предложено *приблизительное решение проблемы* Беренса—Фишера, когда дисперсии двух генеральных совокупностей не равны, объемы выборок меньше 30 и также могут быть не равны. В этом случае по выборочной дисперсии определяется число степеней свободы, задающее верхнее значение t-статистики для заданного доверительного уровня [16].

Указанные варианты представляют самостоятельный интерес, автором не рассматриваются, а для практических целей вполне достаточно сведений, излагаемых далее в этом пункте.

Сравнение двух альтернатив с коррелированными t-парами

Иногда, в целях увеличения контраста при сравнении двух альтернатив, используется идея рассмотрения положительно коррелированных пар. Особенно это важно при изучении влияния на функционирование моделируемой системы измеримых и неизмеримых параметров. Тогда контролируемые параметры выделяются в одну группу, а неконтролируемые замораживаются, чтобы они не оказывали маскирующее влияние.

В качестве примера рассмотрим стартовую зарплату выпускников технических специальностей. Ожидается, что женщины получают зарплату ниже, чем мужчины, и этот факт необходимо проверить сравнением с помощью оценки выборки. Один из способов — получение

некоррелированных пар, как рассмотрено выше. Однако здесь не учитывается такой фактор как пол, который будем считать предметом исследования. Кроме того, необходимо учитывать такие факторы как качество самого вуза, уровень жизни в рассматриваемом регионе, среднюю зарплату по региону, суммарный балл выпускника, принимаемые за неконтролируемые признаки. Поэтому стартовая зарплата является случайной величиной, и трудно оценить влияние полового признака на фоне других неконтролируемых признаков. Один из способов избежать влияния неконтролируемых факторов — использование *связанных* или подобранных пар. Предположим, взята независимая выборка по женщинам и каждому члену выборки подбирается значение зарплаты мужчины, окончившего тот же или одинаковый по рейтингу вуз с примерно одинаковым значением среднего балла. Таким образом создаются связанные пары, для которых заблокировано воздействие уровня жизни в регионе и средней зарплаты в регионе. При вычислении разности зарплат влияние заблокированных факторов будет нивелировано. При использовании таких связанных пар вводится положительная корреляция между членами каждой связанной пары. Главной задачей при этом является уменьшение вариабельности получаемых разностей, что позволит более качественно произвести оценку исследуемых альтернатив. Использование антитез (см. п. 8.3.2) позволяет уменьшить значение дисперсии, но при этом реализуется отрицательная корреляция членов одной генеральной совокупности. В данном случае необходимо получить положительную корреляцию пар, принадлежащих разным генеральным совокупностям. В случае одноканальной системы мы имели дело со средними значениями пар, в рассматриваемом случае двух систем необходимо работать с разностями значений пар. Идею положительной корреляции рассмотрим на примере 6.2, ставшем сквозным для иллюстрации различных методов, модели инструментальной кладовой с разными ДО. Двумя контролируемыми факторами являются ДО FCFS и SPT, а неконтролируемыми факторами — времена прибытия и ожидания. Для того чтобы не возникал вопрос, почему эти времена являются неконтролируемыми факторами, каждое время должно задаваться со своего ГСЧ на основании непрерывной функции. Подобный подход особенно интересен тогда, когда нет возможности получить большую выборку. Сравнение двух ДО производится последовательно для одной ДО, например FCFS, в течение 8 ч за один рабочий день, а затем за то же время для другой ДО, что позволяет оценить разницу в ДО, принимаемую за контролируемый фактор. При этом используется четыре неконтролируемых фактора: два типа механиков и два времени прихода и обслуживания для каждого типа.

Пример 8.6. Использование положительной корреляции в модели инструментальной кладовой

Используя все данные примера 8.5 (на базе примера 6.2) и не повторяя постановку задачи, приведем МФ для случая положительной корреляции, подчеркнув введенные особенности. Для удобства обращения к ним в МФ дана нумерация строк курсивными цифрами, не имеющими прямого отношения к МФ.

1. Введено четыре непрерывные функции, каждая из которых получается со своего ГСЧ (цифры *2–5*).

2. Приоритет механиков 1-го типа устанавливается равным 5 (*6*). Приоритет механиков 2-го типа устанавливается целочисленной АМП &TYPE2PR (*7*), это первый пример задания операнда ОБ непосредственно через АМП. Впоследствии этот операнд используется в качестве индекса петли управления (*8*), меняющей приоритет с 5 на 10 и обратно. Вначале приоритеты обоих механиков равны и реализуется ДО FCFS для всех 10 реплик, затем приоритет возрастает до 10 и реализуется ДО SPT, также для 10 реплик.

3. Вложенная петля управления (*9, 17*) служит для проведения 10 реплик. Сама петля для наглядности сдвинута вправо, и это заставило переназначить начало записи операндов с помощью ОБ OPERCOL (*1*) до 30-й колонки.

4. Для каждой реплики начальное положение ГСЧ1–ГСЧ4 вычисляется в виде функции номера реплики &I (*10*).

5. Реплики помечаются для каждой ДО с помощью ОУ PUTPIC (*12, 14*). Правильность фиксации выходных данных обеспечивается условной петлей IF/ELSE/ENDIF (*11, 13, 15*). При приоритете 5 записываются данные по ДО FCFS и исполняется ОУ (*12*), при невыполнении условия исполняется ОУ (*14*).

6. Исполнение ОУ (*18*) приводит к началу моделирования следующей альтернативы.

1. Модельный файл.

```
* Модуль описания
  SIMULATE                                  Пример 8.6 (модификация
                                            примера 8.5)
*                                           Общие случайные числа
*                                           Сравнение ДО FCFS и SPT
*                                           Временная дискрета: 1 с
      INTEGER   &I                          индекс петли управления
      INTEGER   &TYPE2PR                    индекс изменения приоритета
                                            механиков 2-го типа
 1    OPERCOL   30                          сканирование операнда A
                                            с 30-й колонки
      UNLIST    CSECHO                      не отображать ОУ
 2    TYPE1IAT  FUNCTION  RN1,C2            время прибытия
                                            механиков 1-го типа
      0,60/1,780
```

```
3       TYPE1ST   FUNCTION   RN2,C2      время обслуживания
                                          механиков 1-го типа
        0,210/1,390
4       TYPE2IAT  FUNCTION   RN3,C2      время прибытия
                                          механиков 2-го типа
        0,120/1,600
5       TYPE2ST   FUNCTION   RN4,C2      время обслуживания
                                          механиков 2-го типа
        0,70/1,130
*  Модуль исполнения
*                 Фрагмент 1                1-й тип механиков   *
6                 GENERATE   FN(TYPE1IAT),,,,5 приход
                                          механиков 1-го типа
*                       (Приоритет 5)
                  QUEUE      TOOLWAIT     постановка в очередь
                  SEIZE      CLERK        запрос на обслуживание
                  DEPART     TOOLWAIT     выход из очереди
                  ADVANCE    FN(TYPE1ST ) обслуживание
                                          механиков 1-го типа
                  RELEASE    CLERK        освобождение
                                          кладовщика
                  TERMINATE  0            уход из кладовой
*                 Фрагмент 2                2-й тип механиков   *
7                 GENERATE   FN(TYPE2IAT),_
                             ,,,&TYPE2PR  приход механиков
                                          2-го типа
*                 (Приоритет = &TYPE2PR)
                  QUEUE      TOOLWAIT     постановка в очередь
                  SEIZE      CLERK        запрос на обслуживание
                  DEPART     TOOLWAIT     выход из очереди
                  ADVANCE    FN( TYPE2ST) обслуживание
                  RELEASE    CLERK        освобождение
                                          кладовщика
                  TERMINATE  0            уход из кладовой
*                 Фрагмент 3              Хакт управления *
                  GENERATE   28800        время моделирования 8 ч
                  TERMINATE  1            уменьшение СС на 1,
*                                         окончание движения
                                          транзактов
*       Модуль управления
8                 DO         &TYPE2PR=5,10,5 вначале ДО FCFS,
                                          а затем ДО SPT
9                 DO         &I=1,10,1 10 реплик для текущей
                                          альтернативы
10              RMULT   99000+1000*&I,_   I-я текущая
                                          реплика с RN1 (ГСЧ1)
                        199000+1000*&I,_  I-я текущая
                                          реплика с RN2
```

```
                                299000+1000*&I,_ I-я текущая реплика
                                                 с RN3
                                399000+1000*&I   I-я текущая реплика
                                                 с RN4
                    START       1                старт &I текущей
                                                 реплики
 11                 IF          &TYPE2PR=5       истинно => присваива-
                                                 ется метка FCFS
 12                             PUTPIC LINE=3,FILE=SYSPRINT,(&I)
0=================================================================
 The Report above is for replication * for the FCFS service order .
=================================================================
 13 ELSE                                  в противном случае метка SPT
 14                             PUTPIC LINE=3,FILE=SYSPRINT,(&I)
0=================================================================
 The Report above is for replication * for the SPT service order .
 15                    ENDIF    окончание условной петли
 16                    CLEAR    очистка данных для следующей реплики
 17              ENDDO          следующее значение &I
 18  ENDDO                      следующая альтернатива
     END                        окончание исполнения МФ
```

2. Результаты моделирования (табл. 8.25).

Таблица 8.25. *Среднее число механиков в очереди при положительной корреляции*

№ реплики	Средняя длина очереди при разных ДО		Разность
	FCFS	SPT	
1	1.219	0.973	0.246
2	3.065	2.125	0.940
3	1.057	0.856	0.210
4	3.124	2.244	0.880
5	5.084	3.612	1.472
6	8.654	5.878	2.776
7	2.638	1.888	0.750
8	0.745	0.635	0.110
9	1.592	1.216	0.376
10	2.405	1.710	0.695
Выборочное среднее		0.845	
Выборочное стандартное отклонение		0.796	
Доверительный интервал 80 %		[0.498, 1.192]	
Доверительный интервал 90 %		[0.384, 1.306]	
Доверительный интервал 95 %		[0.276, 1.414]	
Доверительный интервал 99 %		[0.024, 1.662]	

Из таблицы следует, что гипотеза о том, что ДО SPT приводит к меньшему значению очереди, поддерживается до уровня доверительной вероятности 99 %, так как значение нуля еще не попало в доверительный ин-

тервал. Применение положительной корреляции позволило, по сравнению с данными табл. 8.24, намного повысить статистическую точность: так, стандартное отклонение уменьшилось более чем на 60 %, среднее значение также снизилось, однако снижение среднего вызвано, скорее, применяемыми БСВ, а не методикой.

8.4.2. Выбор лучшей из *k* сравниваемых альтернатив

Выше мы рассмотрели выбор лучшей из двух альтернатив, когда выбор ограничен, как в случае двух дисциплин обслуживания, и трудно предложить еще какое-либо альтернативное решение. Однако на практике таких альтернатив может быть значительно больше, и всегда возникает желание найти решение, близкое к оптимальному или лежащее в Парето-оптимальном множестве. Аналитическая идея двухэтапной процедуры поиска лучшей альтернативы из *k* существующих была доложена Дадевичем и Дадалом (Д/Д-процедура) в 1975 г. на математическом симпозиуме в Цинцинати, а затем более подробно представлена в монографии[1], однако, по непонятным причинам, не применена им в его более поздней монографии [15] при описании возможностей ИМ на языке GPSS PC, хотя процедура Д/Д прекрасно используется при моделировании на GPSS/H [17].

Рассмотрим вначале аналитическую схему Д/Д-процедуры, а затем пример ИМ в случае четырех переменных.

Схема двухэтапной Д/Д-процедуры

Рассмотрим основные шаги Д/Д-процедуры.

1. Получается два и более независимых значений по каждой из сравниваемых альтернатив. БСВ, получаемые с одного ГСЧ, в этом случае не применимы. У Лоу [16] рекомендовано на первом шаге получать не менее 15 реплик. Это является первым этапом Д/Д-процедуры.

2. Вычисляются различные статистики, но в обязательном порядке — выборочные среднее и стандартное отклонения.

3. Для каждой альтернативы проводятся дополнительные независимые испытания, количество испытаний варьируется от типа исследуемой задачи и может меняться от альтернативы к альтернативе. Это является вторым этапом Д/Д-процедуры.

4. Для каждой альтернативы по результатам этапов 1 и 2 ищутся взвешенные статистики, причем объемы первой и второй выборки не обязательно совпадают.

---

[1] *Dudewicz E. J.* Modern Mathematical Statistics. NY: John Wiley, 1988.

5. Альтернатива с наибольшим или наименьшим (в зависимости от условий задачи) значением статистики признается лучшей.

Д/Д-процедура оговаривает нормальность распределения выходных значений, но, что весьма важно, не требует равенства дисперсий выходных популяций. Опишем факторы, влияющие на второй этап Д/Д-процедуры, — определение размера второй выборки.

1. *Влияние дисперсии выборки первого этапа.*

Чем выше выборочная дисперсия на первом этапе, тем больше должен быть объем выборки второго этапа при прочих равных условиях. Поскольку выборочная дисперсия различных альтернатив различна, то объем выборки второго этапа для каждой альтернативы будет различным и прямо пропорциональным выборочной дисперсии первого этапа для рассматриваемой альтернативы.

2. *Вероятность принятия правильного решения.*

Поскольку мы имеем дело со случайными векторами в Парето-оптимальном множестве и пользуемся псевдослучайными числами, всегда существует вероятность неверного выбора. Поэтому получаемые решения должны оцениваться задаваемым уровнем доверительной вероятности от 90 % и выше. При этом очевидно, что чем больше уровень задаваемой доверительной вероятности, тем больший объем выборки второго этапа необходимо выбирать.

3. *Уровень безразличия.*

Исследователю необходимо задать уровень ошибки, ниже которого все результаты будут признаваться аналогичными. Так, выходными характеристиками может быть стоимость, производительность, потери, процент брака и т. д. Если, например, задать процент брака 0.5 %, то уровень безразличия позволит считать хорошими системы с выходными характеристиками 99.5 % и выше. Естественно, что стремление понизить уровень безразличия будет приводить к увеличению объема выборки. Вообще, чтобы быть точным, объем выборки второго этапа обратно пропорционален квадрату значения безразличия.

Из перечисленных факторов очевидно, что определение объема выборки второго этапа является достаточно сложной проблемой. Приведем основные уравнения, которые используются на втором и четвертом шагах Д/Д-процедуры и будут использованы в примере 8.7.

Рассмотрим гипотетический пример ткацкой фабрики, на которой работает какое-то количество собственных станков, которые могут отказывать в процессе эксплуатации. Для поддержания объема производства и для устранения дефектов, во-первых, арендуется дополнительный станок и, во-вторых, имеется несколько ремонтников. В табл. 8.26 представлены значения выборочных среднего $\bar{x}$ и

Таблица 8.26. *Значение статистик для четырех альтернатив, выраженных в стоимости (в рублях)*

y		x	
		0	1
1	$\bar{x}$	12833	14140
	s	1227	1439
2	$\bar{x}$	12490	12845
	s	242	555

стандартного отклонения s для x арендуемых станков и y ремонтных рабочих на этапе первоначальной выборки.

Среди указанных альтернатив необходимо выбрать такую, которая приводит к минимальной стоимости за день. Данные табл. 8.26 получены на основе модели примера 8.7 при реализации 15 реплик для каждой из альтернатив, идея построения прослеживается в самой таблице. Полученные данные являются основой для вычисления размера выборки второго этапа при принятом уровне доверительной вероятности 95 % и уровне безразличия 300 р. в день. Положим, что n_0 — начальный объем выборки первого этапа; N — общий объем выборки после добавления выборки второго этапа для каждой рассматриваемой альтернативы; $N - n_0$ — объем выборки, добавляемой на втором этапе. Тогда значение N определится из следующего выражения:

$$N = \max \{n_0 + 1; [(h_1 s/d)^2]\}, \qquad (8.33)$$

где s — выборочное стандартное отклонение рассматриваемой альтернативы; d — уровень безразличия, одинаковый для всех альтернатив; h_1 — коэффициент, зависящий от:

— размера первоначальной выборки n_0;

— принятого уровня доверительной вероятности P, %;

— числа рассматриваемых альтернатив $k \geq 2$.

В табл. 8.27 приведены значения коэффициента h_1 с учетом всех вышеназванных факторов для восьми значений альтернатив k (более подробные таблицы приведены в монографии [17]).

В уравнении (8.33) прямые скобки [.] использованы для указания, что берется наименьшее целое значение, которое превышает или равно m. Например, $[0.05] = 1$, $[31.8] = 32$ и т. д. Таким образом, в минимальном случае N больше n_0 на 1 и равняется $[(h_1 s/d)^2]$, если это значение превышает $n_0 + 1$. Например, если на первом этапе объем выборки принят равным 15, а $[(h_1 s/d)^2] = 31.9$, тогда объем добавляемой выборки равен $32 - 15 = 17$; а если $[(h_1 s/d)^2] = 7.1$, тогда добавляется выборка второго этапа величиной 1 и суммарная выборка = 16.

Таблица 8.27. *Значения коэффициента h_1 при разных факторах*

P, %	n_0	Количество рассматриваемых альтернатив k							
		2	3	4	5	6	7	8	9
90	15	1.93	2.39	2.63	2.81	2.93	3.04	3.12	3.20
90	20	1.90	2.34	2.58	2.75	2.87	2.97	3.05	3.12
90	25	1.88	2.32	2.55	2.72	2.84	2.93	3.01	3.08
90	30	1.87	2.30	2.54	2.69	2.81	2.91	2.98	3.05
95	15	2.50	2.94	3.17	3.34	3.46	3.57	3.65	3.72
95	20	2.45	2.87	3.10	3.26	3.38	3.47	3.55	3.62
95	25	2.42	2.84	3.06	3.21	3.33	3.42	3.50	3.56
95	30	2.41	2.81	3.03	3.18	3.30	3.39	3.46	3.53
99	15	3.64	4.04	4.27	4.43	4.55	4.64	4.73	4.80
99	20	3.54	3.92	4.13	4.28	4.39	4.48	4.55	4.62
99	25	3.48	3.85	4.05	4.20	4.30	4.39	4.46	4.53
99	30	3.45	3.81	4.01	4.14	4.25	4.33	4.40	4.46

Используя данные табл. 8.27, вычислим объемы выборки для четырех альтернатив табл. 8.26 при $P = 95\ \%$, $k = 4$, $n_0 = 15$, $d = 300$. Данные сведены в табл. 8.28, где числа внутри таблицы представляют значение выражения (8.33) — $[(h_1 s/d)^2]$, через / — добавляемый объем выборки второго этапа $N - n_0$, x — число арендуемых станков, y — число ремонтников.

Как видно из таблицы, объем выборки второго этапа варьируется от 1 до 203 в зависимости от стандартного отклонения выборки первого этапа. Объем выборки второго этапа может быть уменьшен при уменьшении значения доверительной вероятности и / или увеличении уровня безразличия. После проведения испытаний с увеличенным объемом выборки подсчитывается среднее взвешенное значение (стандартное отклонение по выборке второго этапа не используется и поэтому может не рассчитываться).

Для каждой альтернативы, по выборкам первого и второго этапов, подсчитываются средние значения, которые затем взвешивают-

Таблица 8.28. *Объем выборки второго этапа*

Показатель	Данные по номеру альтернативы			
	1	2	3	4
Комбинация x, y	0, 1	1, 1	0, 2	1, 2
Значения $[(h_1 s/d)^2]/N - n_0$	158.12/144	217.4/203	6.16/1	32.34/18
Средняя стоимость, р.	13120	14235	12160	12920

Таблица 8.29. Окончательные результаты выбора

Показатель	Данные по номеру альтернативы			
	1	2	3	4
Комбинация x, y	0, 1	1, 1	0, 2	1, 2
Значения W_0/W_1	0.116/ 0.184	0.082/ 0.918	1.243/ −0.243	0.526/ 0.474
Взвешенная стоимость, р.	13090	14277	12565	12878

ся и складываются. Вес W_0 для выборки первого этапа для каждой альтернативы подсчитывается на основе выражения

$$W_0 = (n_0/N)\left[1+\sqrt{1-\left(N/n_0\right)\left[1-\left(N-n_0\right)/\left(h_1 s/d^2\right)\right]}\right]. \quad (8.34)$$

Значение весового коэффициента W_1 для выборки второго этапа каждой из альтернатив определится как $W_1 = 1 - W_0$.

В табл. 8.29 приведены значения весовых коэффициентов, подсчитанные из выражения (8.34), и значения средних стоимостей для каждой из альтернатив.

Отметим, что для альтернативы № 3 коэффициент первого этапа оказался больше единицы, что привело к отрицательному коэффициенту на втором этапе, а это именно та альтернатива, объем выборки для которой увеличился всего на единицу. С доверительной вероятностью 95 % наименьшая стоимость относится как раз к этой альтернативе, которая имеет стоимость 12565 р. в день и при заданном уровне безразличия должна быть выбрана как лучшая. Кстати, эта же альтернатива оказалась лучшей и по результатам оценки выборки первого этапа, однако этот факт совершенно не обязателен, чаще всего происходит как раз обратная ситуация, когда лучшая альтернатива на первом этапе может оказаться далеко не лучшей после добавления выборки второго этапа.

Пример 8.7. Выбор лучшего варианта из четырех возможных (модель ткацкой фабрики)

Д/Д-процедура будет проиллюстрирована с помощью описания имитационной модели ткацкой фабрики.

1. Постановка задачи.

На ткацкой фабрике работает 10 ткацких станков, 5 дней в неделю, по 8 ч в день. На каждом станке работает отдельный рабочий. Поскольку существует возможность случайного отказа станка, то для обеспечения 400 ч оперативной работы в неделю возникают разные возможности:

— иметь один дополнительный станок в рабочем состоянии для немедленной замены отказавшего станка;

— восстановить имеющийся дополнительный станок (если он находится не в рабочем состоянии) как можно быстрее;

— поместить отказавший станок в ремонтную зону и начать его восстановление.

Эпюра возможных состояний показана на рис. 8.7.

Итак, на эпюре изображено 12 станков, из них 10 работают, 1 восстанавливается, но очереди на восстановление нет (хотя она и возможна), 1 в полностью исправном состоянии находится в резервном режиме. Целью менеджмента фабрики является обеспечение 400 ч продуктивной работы в неделю. Час простоя станка стоит 560 р. Среднее число работающих машин равно 8, т. е. 64 ч в день, вместо ожидаемых 80, тогда сумма возможных потерь может достигнуть $16 \times 560 = 8960$ р. На фабрике есть один мастер по ремонту станков, но руководство готово нанять второго, если это окажется эффективным. Оплата рабочего составляет 3600 р. в день (пример гипотетический, поэтому не надо удивляться уровню оплаты труда, тем более, что она составляет лишь 35 % стоимости простоя). При этом руководство не хочет нанимать еще операторов для работы на станках и приобретать в собственность дополнительно 11 станков из-за их дороговизны, но готово арендовать станок на долгосрочной основе. Стоимость аренды составляет 3000 р. независимо от того, используется станок или нет. Время восстановления отказавшего станка составляет 24±8 ч и не может быть сокращено вдвое даже при наличии двух ремонтников. Время безотказной работы станка равняется 200±100 ч, время замены отказавшего станка на исправный пренебрежимо мало и не учитывается. Время восстановления и наработка на отказ принимается одинаковой для всех станков, также считается, что не существует разницы в квалификации рабочих и ремонтников. Возможны четыре стратегии поведения руководства:

— не делать ничего нового — альтернатива (0, 1);

— нанять второго ремонтника, но не брать в аренду станок — альтернатива (0, 2);

— не нанимать второго ремонтника, а арендовать 1 станок — (1, 1);

— нанять второго ремонтника и арендовать 1 станок — (1, 2).

Нужно составить модель для описания процесса функционирования фабрики и, используя Д/Д-процедуру, определить лучшую стратегию поведения руководства из четырех названных в смысле минимизации ожидаемой дневной стоимости. Положим уровень доверительной вероятности равным 95 %, а уровень безразличия равным 300 р. в сутки. На

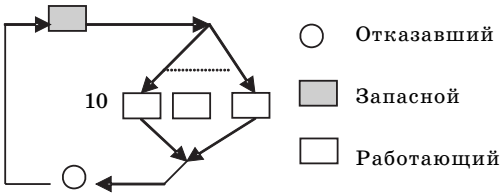


Рис. 8.7. Эпюра возможных состояний

первом этапе провести 15 реплик для всех альтернатив. В определении дневной стоимости включать зарплату одного или двух ремонтников (3600 р. в день), арендную плату (3000 р. в день), стоимость простоя равна 540 р. в час или 4480 р. за рабочий день.

Будем считать, что в начале моделирования все 10 станков находятся в работоспособном состоянии. Оставшаяся работоспособность каждого станка определяется значением 150±140 ч, т. е. лежит в интервале от 10 часов до 290. Другие собственные станки, в том числе и арендованный, обладают полным ресурсом.

2. Допущения, сделанные в модели.

Оговорим ограничения, принимаемые в модели, поскольку они могут повлиять на функционирование отдельных операторов ЯИМ. Эти ограничения касаются числа ремонтников,числа рабочих-операторов, полного числа станков в системе. Первые два ограничения удобно представить ОУ STORAGE, а третье — транзактами. Это объясняется тем, что число ремонтников и рабочих задается, а число станков может быть переменным и двигаться в процессе функционирования от этапа эпюры к этапу.

Рассмотрим особенности функционирования модели. Предположим, что станок готов к использованию, но в действии находятся 10 исправных станков и память OPRATORS (напомним, что имя не может иметь больше 8 символов) полна и имеющийся станок не может исполнить следующий ожидаемый ОБ ENTER OPRATORS. Как только один из станков откажет, оператор освободится и сможет начать работу на резервном станке, который начнет функционирование со временем полностью исправного станка, т. е. начинает расходовать ресурс сначала в интервале (100, 300), а не в интервале (10, 290). Отказавший станок, пройдя все свои ОБ, возвращается к точке ожидания входа в систему. Очевидно, что эту логику можно представить двумя фрагментами модуля исполнения, со своим ОБ GENERATE,, 1,1 который вводит 1 станок в момент 1 (момент отказа очередного станка). Еще один фрагмент модуля исполнения вводит при необходимости арендованный станок с помощью ОБ GENERATE,,1,& LEASED.

3. Таблица определений (табл. 8.30).

Таблица 8.30

Объект GPSS/H	Объект в системе
Транзакты	
Фрагмент 1	Арендованный станок
Фрагмент 2	Основные станки (вначале 10)
Фрагмент 3	Резервный станок
Фрагмент 4	Транзакт управления
Амперпеременные	
FIXERS	Число ремонтников (1, 2)
I	Счетчик реплик
LEASED	Число арендуемых станков (0, 1)
Памяти	
FIXSHOP	Память для ремонтников (1, 2)
OPRATORS	Память числа операторов

4. Модельный файл.

```
*            Модуль 1 Управления и описания
             SIMULATE                  Пример 8.7. Модель ткацкой
                                       фабрики
*                                      Временная дискрета: 1 ч
             INTEGER   &FIXERS         число ремонтников
             INTEGER   &I              &I — индекс петли
                                       управления
             INTEGER   &LEASED         число арендованных
                                       машин
             OPERCOL   30              считывание первого операнда
                                       с 30-й колонки
             UNLIST    CSECHO          запрет на показ ОУ
OPRATORS     STORAGE   10              10 операторов
*            Модуль 2 Исполнения
             Фрагмент 1                Арендованный станок
                                       (при необходимости )
             GENERATE  ,,1,&LEASED     ввод 1-го арендуемого
                                       станка в момент 1
             TRANSFER  ,REPEAT         передача в основную линию
 *           Фрагмент 2                Станки в основной линии
             GENERATE  0,,,10          функционирование
                                       10 основных станков
             ENTER     OPRATORS        занятие оператора
                                       (без задержки)
             ADVANCE   150,140         использование остаточного
                                       ресурса
             TRANSFER  ,BROKEN         переход на восстановление
*            Фрагмент 3                Восстановление станка
             GENERATE  ,,1,1           ввод восстановленного станка
                                       в момент 1
REPEAT       ENTER     OPRATORS        занятие оператора
             ADVANCE   200,100         использование полного
                                       ресурса
BROKEN       LEAVE     OPRATORS        при отказе освобождение
                                       оператора
             ENTER     FIXSHOP         занятие ремонтника
             ADVANCE   24,8            время восстановления
             LEAVE     FIXSHOP         освобождение ремонтника
             TRANSFER  ,REPEAT         передача на использование
*            Фрагмент 4                Временной таймер
             GENERATE  1000            25 рабочих недель
             TERMINATE 1               сигнал на окончание
                                       процесса ИМ
```

```
*              Модуль 3 Управления
               DO          &FIXERS=1,2,1   вначале 1, затем
                                           2 ремонтника
FIXSHOP        STORAGE     &FIXERS         изменение емкости
                                           памяти FIXSHOP
                 DO        &LEASED=0,1,1   вначале 0 , затем
                                           1 арендованный станок
                   DO      &I=1,15,1       осуществление 15 реплик
                         START 1           запуск &I-й реплики
                         PUTPIC LINES=6,FILE=SYSPRINT,
                                           (&I,&FIXERS,&LEASED)
0=============================================================
 Shown above is Replication Report * for this configuration:

                 Number of repairpersons : *
                 Number of leased machines : *
=============================================================
                       CLEAR               очистка для проведения
                                           следующей реплики
                     ENDDO                 проведение следующей реплики
                   ENDDO                   изменение числа
                                           арендованных станков
                 ENDDO                     изменение числа ремонтников
                 END                       окончание процесса ИМ
```

5. Итоговый отчет.

Здесь (надеясь, что читатель не только промоделировал работу ткацкой фабрики, но и сам построил МФ, возможно, сняв ряд введенных в начале упрощающих ограничений) не приводим полного листинга отчета, а даем лишь табл. 8.31 основных результатов, главными из которых являются выборочные значения среднего числа продуктивно используемых станков и стандартного отклонения от этого числа по результатам 15 реплик первого этапа для каждой из альтернатив. Данные по объему дополнительной выборки использованы из примера 8.4.

Таблица 8.31. *Выборочные статистики четырех альтернатив*

Показатель	Данные по номеру альтернативы			
	1	2	3	4
Комбинация x, y	0, 1	1, 1	0, 2	1, 2
Среднее значение	7.9584	8.3601	8.8687	9.4850
Отклонение	0.2748	0.3223	0.0542	0.1243
Данные на основе выборки второго этапа				
Среднее значение 2	7.8939	8.339	8.938	9.4667
Отклонение 2	0.2705	0.3189	н/о	0.0802

На основании табл. 8.31 можно оценить потери, получаемые при работе с числом станков меньше 10, не забывая при этом учитывать стоимость аренды и/или зарплату 2-го ремонтника. Для второго этапа требуются только средние значения, но во второй части таблицы приведены значения и стандартных отклонений. Стандартное отклонение для альтернативы 3 не вычислялось, потому что выборка второго этапа для нее равна 1 (см. табл. 8.28). Поскольку интересна оценка средней стоимости для каждой альтернативы, то учет арендной платы и зарплаты не проводился.

6. Обсуждение.

В примере 8.7 АМП используются как значение операндов (второй раз в примерах пособия), что указывает на гибкость моделирования на GPSS/H — одна из прежних версий ЯИМ этого не позволяет. В МФ использованы три вложенные петли управления числом ремонтников, числом арендованных станков и числом реплик. Удобство применения петель управления в ЯИМ вообще хорошо иллюстрируется возможностями рассматриваемого примера.

Отметим, что ОУ STORAGE FIXSHOP расположен в модуле 3, а не в модуле 1, и его емкость определяется АМП. Поскольку количество ремонтников в начальный момент моделирования ничего не определяет, то размещение в модуле 3 не влияет на логику работы, а читается удобней.

В начальный момент времени число арендованных станков равно нулю, а следовательно, операнд D ОБ GENERATE фрагмента 1 также равен нулю, т. е. генератор не инициализирован и транзакты с него не поступают.

Для осуществления реплик второго этапа были введены некоторые дополнения, не отмеченные в представленном МФ.

1. В модуль 1 после ОУ OPERCOL был введен ОУ RMULT в виде

RMULT 200000 для альтернативы (1, 0), после чего для других альтернатив операнд A менялся следующим образом: 300000, 400000, 500000 соответственно для альтернатив (1, 1), (0, 2), (1, 2).

2. Для осуществления каждой из альтернатив приходится изменять операнды АМП в петлях управления (кроме числа реплик), например, для альтернативы (1,0)

DO &FIXERS=1,1,1

DO &LEASED=0,0,1

3. АМП в петле управления числом реплик должна быть изменена на требуемое число дополнительных реплик, например, для альтернативы (1, 0)

DO &I=1,144,1

В силу своей уникальности все четыре отдельных МФ были промоделированы в пакетном режиме. Статистические данные были получены с помощью ПО «Статистика» вне тела программы.

Лучшим выбран вариант, дающий наибольшее использование станков.

§ 8.5. УПРАЖНЕНИЯ

Как и в гл. 7, разбор ошибок не включен, так как большинство примеров требуют применения пакетного режима.

29. Используя данные табл. 8.1, выражение (8.1) и данные примеров 6.2, 6.4, ответьте в тестовом режиме на следующие ***вопросы***.

A. Проверьте утверждение, что первое время прихода механика 1-го типа равно приблизительно 254.6 временных дискрет (пример 6.2).

B. Предположим, что фрагменты 1 и 2 поменялись местами. Чему будет равно время первого прихода механика 1-го типа (пример 6.2)?

C. Когда первый и второй телевизоры войдут в систему и когда окончится контроль первого телевизора (пример 6.4)?

30. Используя пример 6.3, введите в него новые ОУ INTEGER, UNLIST CSECHO, DO/ENDO, PUTPIC. Проведите процесс получения 20 реплик. Используйте данные для вычисления выборочного среднего и стандартного отклонения для четырех вариантов: среднее время нахождения судов типа A и B в гавани порта, среднее время ожидания причала для судов каждого типа. (Воспользуйтесь подсказкой на базе примеров 8.4–8.6.) Определение статистик производите с помощью внешних программ.

31. Используя данные табл. 8.20, подсчитайте значения доверительных интервалов для 80 и 95 % доверительной вероятности.

Вопрос: Насколько полученные значения соответствуют вашим первоначальным представлениям?

32. Предположим, что кто-то выскажет сомнения в действительной независимости данных по двум ДО табл. 8.20. Как вы прокомментируете эти сомнения?

33а. Используя МФ примера 8.6, ответьте в тестовом режиме на следующие ***вопросы***.

A. Когда при ДО FCFS в первый день механик 1-го типа придет 10-й раз?

B. Когда придет механик 2-го типа 10-й раз?

C. В какое время начнется обслуживание этих механиков?

D. Как долго они будут обслуживаться?

Ответы на эти вопросы найдите сами, положив значение приоритета механика 2-го типа равным приоритету механика 1-го типа и помня, что ИН транзактов механиков 1-го типа имеет нечетный номер.

33б. Используя пример 8.6 и пакетный режим, ответьте на следующие ***вопросы***.

A. Почему используются четыре разных ГСЧ и будет ли последовательность событий одинакова в случае одного и нескольких ГСЧ?

B. Почему в случае связанных пар чисел можно обойтись тремя ГСЧ?

C. Почему показания позиции ГСЧ в петле управления не могут быть представлены в виде неизменных чисел, например 100000, 200000 и т. д.

Часть 3
РАСШИРЕНИЕ ВОЗМОЖНОСТЕЙ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ НА ЯЗЫКЕ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ GPSS/H

Настоящий раздел предназначен только для информации читателя еще о двух ПП Wolverine SW Corp. Поэтому в разделе не приводятся конкретные данные об упоминаемых программных продуктах, а дается их общее описание и конспективное изложение логики действия. Очевидно, что заинтересованные читатели могут непосредственно обратиться в корпорацию. Стоимость студенческой версии Proof Animation составляет всего 35 $.

Глава 9
АНИМАЦИОННЫЙ ПАКЕТ PROOF ANIMATION

§ 9.1. ФИЛОСОФИЯ АНИМАЦИИ

Любой заказчик результатов имитационных исследований или непосредственный исследователь редко удовлетворяются сухими средними цифрами, получаемыми в итоговом отчете, всегда хочется увидеть процесс функционирования наглядно, а еще лучше — в динамике. Известно, что скорость и достоверность зрительного восприятия на порядки выше, чем слухового или анализа цифровых значений. Поэтому анимация на сегодняшний день рассматривается как обязательная компонента коммерческих имитационных систем. В самом деле, при проектировании нового цеха интересно воочию представить расположение оборудования, персонала, материальных потоков, складирование полуфабрикатов и готовой продукции и т. д.

Примеры, естественно, можно множить без конца, что подчеркивает важность создания средств анимации.

Многие системы ИМ имеют встроенные средства анимации, позволяющие отобразить и сам процесс ИМ, и его результаты. Но они не позволяют решить многие задачи по визуализации исследований, поскольку имеют выраженную академическую направленность, уровень которой является недостаточным для выполнения конкретных задач.

Задача привлечения к моделированию более широкого круга пользователей может успешно решаться привлечением новых про-

граммных средств анимации, позволяющих перейти от обычных абстрактных объектов ИМ к анимационным картинкам, адекватно описывающим процессы реального мира.

Существует много прикладных пакетов для создания анимации имитационных моделей. Одни представляют собой объектно-ориентированные коммерческие пакеты ИМ, основанные на принципах визуального моделирования. Из них можно выделить: Arena, AutoMod, Extend, ProModel, QUEST, SIMFACTORY II.5, SIMPLE++ (eM-Plant), Taylor ED и др.

Другие являются самостоятельными средствами для создания анимации имитационной модели на основании определенных входных данных. В этом плане представляет конкретный практический интерес в применении для исследований и в обучении пакет Proof Animation корпорации Wolverine Software (1993). Недаром этот ПП используется создателями других версий GPSS, например GPSS World, являющейся развитием GPSS PC. Причем авторы подчеркивают, что Proof не является *объектно-ориентированным ПП*.

Известно, что ЯИМ GPSS вводит данные и выводит результаты как ASCII-файлы. Это обеспечивает легкость использования этих файлов как интерфейса между другим программным обеспечением и системой GPSS. А поскольку ASCII-код является входом в Proof Animation, то эти пакеты можно использовать в сочетании друг с другом. Proof Animation представляет собой универсальный инструмент для создания анимации на основе трассировочных файлов, представленных в ASCII. Трассировочные файлы можно делать в разных средах, используя различные языки программирования. Единственным условием будет лишь соответствие «стандартам» трассировочных файлов Proof Animation.

Это является одним из существенных преимуществ Proof Animation по сравнению с сегодняшними аналогами.

В последнее время наблюдается тенденция по выводу в мировую сеть технологий ИМ на основе специализированных языков, подобных HTML. Так возник VRML (Virtual Reality Modeling Language — Язык Моделирования Виртуальной Реальности). Этот формат файлов используется для описания трехмерных объектов и миров для сети Internet. Первая реализация VRML (VRML 1.0 спецификация) была создана компанией Silicon Graphics и представляла собой формат описания статических миров. Во второй реализации VRML (VRML 2.0 спецификация) добавились более сложные интерактивные возможности и анимация. Она была разработана компанией Silicon Graphics в сотрудничестве с компаниями Sony и Mitra.

Есть много инструментов для создания продуктов компьютерной визуализации, но внимание обращают именно на VRML по следующим причинам.

1. Этот язык, как и вся технология разработки графических моделей, является открытым, так что для начального процесса обучения не нужно будет покупать дорогие пакеты по созданию VRML-анимации, а для демонстрации продуктов достаточно иметь стандартный Internet-browser.

2. Графические модели имеют вполне хорошее качество изображения, совершенно достаточное для отображения процессов в технике (имеется возможность, например, показать целый цех, начиная от отдельной заклепки на корпусе станка и кончая «панорамой», на которой видны 100 работающих станков одновременно).

3. Во многих областях техники (например, в автомобильной промышленности) именно VRML-модели становятся «де-факто» стандартной формой создания изображений объектов и средств производства, которая последовала за CAD-чертежами; имеется много «конвертеров», которые позволяют быстро создавать трехмерные VRML-модели на базе CAD-чертежей; многие современные симуляторы предоставляют возможность «встраивать» готовые VRML-модели в собственные «анимации»; уже создаются «отраслевые библиотеки» с наборами VRML-моделей для соответствующих изделий и средств производства; в развиваемой сегодня концепции Digitale Fabrik (Virtual Factory) VRML-модели занимают ведущее место.

Слабая сторона VRML-моделей состоит в том, что вопросы взаимосвязи таких моделей с имитационными моделями разработаны очень слабо. Это объясняется тем, что VRML-модели чаще всего выглядят как законченные модели, на которые очень трудно влиять извне. Однако это возможно, так как любая VRML-модель, в конечном итоге, — это код ASCII.

§ 9.2. КРАТКОЕ ОПИСАНИЕ ПРОГРАММНОГО ПРОДУКТА PROOF ANIMATION

9.2.1. Добавление анимации к моделированию с использованием Proof

Общий обзор

Proof Animation представляет собой семейство программных продуктов (ПП), дополняющих анимацией процесс моделирования дискретных событий. ПП выпускается в виде различных версий, вклю-

чая сокращенную студенческую версию, версию среднего размера, коммерческую версию неограниченного размера и демоверсию свободного распространения. ПП построен на базе ASCII-кодов для имитации на IBM-совместимых компьютерах систем широкого класса. Векторное построение обеспечивает большие возможности графики и позволяет менять размеры изображения. ПП включает встроенные элементы построения четких изображений и систему ввода и вывода данных. Открытая архитектура ПП делает его идеальным инструментом для отображения результатов моделирования, проведенного с помощью различных ЯИМ. Вне зависимости от величины, сложности и области применения модели, ПП обеспечивает возможность отображения результатов в виде движущихся изображений. ПП использует возможности интерфейса DirectDraw, имеющегося во всех версиях Windows 98, NT4, 2000, XP, а также может быть легко добавлен к Windows 95.

В ПП используются два входных потока ASCII-кодов. Первый из них создает статическое изображение объекта, в котором происходит движение динамических элементов — транзактов (аэропорт, госпиталь, банк и т. д.), а второй отображает временную последовательность транзактов рождающихся, движущихся, уничтожаемых и т. п. Оба эти потока представляют собой ASCII-тексты свободного формата, создаваемые достаточно просто различными способами. В ПП могут быть использованы выходные данные модели, записанные в отдельном файле, или файлы из другой программы. Файлы, имитирующие движение транзактов, представляются с помощью какого-либо ЯИМ или специального пакета типа SLX (см. описание ниже), GPSS/H, Extend, Slam, Siman, Simscript, GPSS и т. д. Файлы движения могут быть записаны на языке общего назначения типа C, а в простых случаях — с помощью текстового редактора. Файлы статического изображения создаются с помощью средств, встроенных в ПП Proof. ПП позволяет выполнять задачи САПР, импортируя .DXF-файлы. Когда структура создается один раз, она может быть создана непосредственно. Когда ПП непосредственно сопряжен с ЯИМ, то с помощью подпрограммы входной поток вводится построчно в каждый момент времени. ПП Proof может напрямую управляться любой C-совместимой программой, находящейся в Windows DLL (Dynamic Link Library). ПП Proof представляет собой открытую, не зависящую от вида ЯИМ-структуру, применимую как для технических, так и экономических задач. Естественно, что по мере изменений и улучшений в GPSS/H и SLX в ПП также могут вноситься изменения.

Хотя ПП Proof, разработанный Wolverine Corp., предназначен, в первую очередь, для работы с продукцией этой же корпорации SLX и

GPSS/H, тем не менее он может использоваться с другими ЯИМ и программами, что являлось основной целью разработчиков. Таким образом, ПП используется как выходной интерфейс для широкого класса различных видов входного интерфейса.

Уточнение деталей

• Входные ASCII-потоки управляют работой ПП. Любое ПО, допускающее форматирование в ASCII-кодах, может быть использовано в целях анимации с помощью Proof. Один из потоков управляет построением геометрических деталей пространства, в котором движутся динамические элементы (транзакты), а также задает путь их продвижения. Второй поток задает последовательность движения транзактов с помощью команд CREATE, DESTROY, PLACE, PLOT, MOVE, SET SPEED, SET COLOR и некоторых других. Эти команды просты, легко осваиваются и обеспечивают простоту и гибкость интеграции с логикой МФ.

• Использование результатов моделирования. В этом случае потоки структуры и движения, прежде чем поступить на вход Proof, должны быть представлены в виде файлов. Можно отметить два преимущества: 1) аппаратное обеспечение одинаково как для моделирования, так и для анимации; 2) обеспечивается возможность при анимации перемещаться во времени назад и вперед, изменять скорость просмотра или рассматривать наиболее интересную часть структуры.

• Конкурирующая версия (DLL). При использовании любого С-совместимого ПО можно пользоваться DLL-версией Proof. Эта версия обеспечивает следующие функции:

ProofDLL	инициализация — остановка
ProofOpenLayout	окрытие файла структуры
ProofSendTraceLine	посылка команд трассировки
ProofSuspend	временная остановка
ProofResume	итоги анимации
ProofStatus	восстановление статуса

DLL обеспечивает необходимую синхронизацию между ПП и посылаемыми командами. Число команд невелико, что позволяет быстро их освоить и экономить время моделирования.

• Векторная геометрия. В ПП оба потока основываются на векторной геометрии. Одним из преимуществ такого подхода является возможность создания сложного изображения, меняющего свои размеры. Второе достоинство — возможность использовать элементы стандартного САПР, представленные в виде DXF-файлов.

• Плавное движение. Proof обеспечивает плавное движение многих объектов. Все время поддерживаются два вида изображений — видимых и невидимых. Ввод данных производится в виде невиди-

мых изображений; когда ввод данных завершен, мгновенно появляются оба вида изображений. Изображения отображаются за счет ПО вертикальной развертки, происходящей 60–70 раз в секунду, что гораздо выше, чем у других видов ПО, за счет этого и обеспечивается четкость и плавность.

• Цвет и разрешение. Все версии ПП обеспечивают передачу 256 цветов с разрешением экрана от 640×400 до 1280×1024 пикселей в зависимости от размера памяти применяемой видеокарты (от 2 Мбайт и выше).

Создание анимаций и презентаций

• Изображение структуры. Если в системе существует САПР в виде DXF-файлов, то пользователь импортирует их в ПП, используя встроенные утилиты САПР. При импортировании используется послойный или полинейный ввод в зависимости от типа отображаемого объекта. При отсутствии элементов САПР можно пользоваться внутренним построителем ПП, позволяющим вводить данные с помощью мыши или клавиатуры, численно задавая длину линии и угол наклона.

• Определение классов объектов. После того как создана структура, вторым шагом является определение одного или нескольких классов объектов. Создание движущихся объектов с их отличительным видом, цветом и скоростью движения является одной из важных особенностей ПП. Движение объекта начинается только после его идентификации как динамического объекта, его размеры зависят от созданной структуры, а разные цвета обозначают освобождение или занятость ресурсов и состояние транзакта.

• Определение траектории.

9.2.2. Структура ПП Proof

Меню ПП Proof включает 6 типов функций: движение, отладка, рисование, класс объекта, траектория движения, презентация. Кратко рассмотрим каждую из этих функций, не рассматривая, как она реализуется программно, так как получение навыков возможно лишь при наличии самого ПП Proof, поставленного на ваш компьютер.

Функция движения. Позволяет запускать и останавливать процесс анимации, управляя при этом скоростью и временем движения.

Функция отладки. Аналогична функции движения, а кроме того, дает возможность задерживать процесс анимации и изучать передвижение какого-то класса объектов.

Функция рисования. Позволяет создавать и редактировать структуру неподвижных частей (здания, дороги, оборудование).

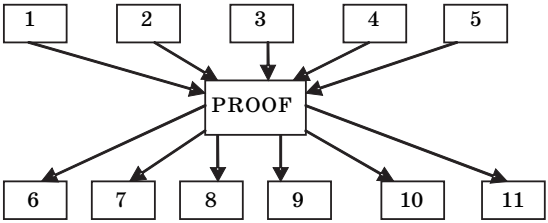


Рис. 9.1. *Входные и выходные файлы ПП Proof*

Функция траектории. Позволяет создавать траектории, по которым будет двигаться объект. Траектории создаются пользователем на основе сочетания линий, дуг и вводимых координат.

Функция класса объекта. Позволяет выбрать вид и форму движущегося объекта, причем вид выбирается в начале и не меняется в процессе анимации.

Функция презентации. Позволяет контролировать сценарий презентации, которая является одной из частей процесса анимации. Она допускает просмотр статических слайдов, применение специальных эффектов и / или управление меню, созданного пользователем.

На рис. 9.1 представлен набор входных и выходных файлов, применяемых в ПП Proof.

Входные файлы

1. Обязательные файлы структуры неподвижных объектов. Эти файлы содержат основополагающий текст и все графические изображения, необходимые для анимации. В них также заносятся определители классов объектов, сообщения переменного объема, панели управления графиками, траекториями, чертежами для организации движения. Файлы структуры обычно создаются и сохраняются в рамках самого ПП Proof, однако могут передаваться из других программ или генерироваться с их помощью, например все необходимые CAD-программы. В том случае, если возникает желание ввести данные из собственной программы, необходимо помнить, что формат ввода требует ASCII-кодов. Расширение этих файлов .LAY-Layout.

2. Обязательные файлы трассировки анимации. Файлы используются после создания среды анимации и выбора объектов. С их помощью задается временная последовательность движения (моменты поворотов, изменения направления и т. п.) и другие команды, позволяющие осуществлять анимацию. Файлы трассировки записываются непосредственно в модель или программу с соблюдением правил синтаксиса, принятого в ПП Proof. Когда используются возможности динамической библиотеки (файлы .DLL), команды подаются из

программы запуска и не требуется создание файлов трассировки. Синтаксис файлов трассировки описывается в руководстве к ПП Proof, а часть информации содержится в HELP. Расширение этих файлов имеет вид .ATF — Animation Trace.

3. Не обязательный файл описания презентации. Он содержит последовательность команд, которые позволяют выводить на дисплей слайды. Файл создается пользователем с помощью текстового редактора. Расширение этих файлов имеет вид .PSF — Presentation Script.

4. Не обязательные файлы растровых изображений. Эти файлы содержат статические изображения для презентации. Источником этих файлов может быть либо ПП Proof, либо любая программа, поддерживающая форматы .PCX, .BMP. Расширение этих файлов и *9* имеет вид .PCX — Bitmapped Image (PaintBrush).

5. Не обязательные CAD-файлы (.DXF). Стандартные файлы .DXF вводятся извне и сохраняются во всех версиях ПП Proof, кроме студенческой.

Выходные файлы

6. Файлы структуры аналогичны входным файлам структуры *1*. В любой момент времени эти файлы могут быть сохранены или использоваться в качестве входных для любого графопостроителя. При сохранении файлов в них сохраняются определения и вся начальная информация. Расширение, так же как у *1*, .LAY-Layot.

7. Файл связей. При выполнении функций рисования или создания траектории ПП Proof производит файл связей, который содержит описание параметров траекторий. Поэтому модель или программа может извлекать из него любую необходимую информацию, чтобы убедиться, что значения расстояния и времени не противоречат первоначально заданным. С другой стороны, открывается возможность изменять геометрию траектории или скорость движения по ней непосредственно в процессе анимации. Расширение этих фалов имеет вид .LKG — Linkage.

8. Файлы оповещения. Время от времени ПП Proof выдает на экран предупреждения, сигналы изменения состояния или сообщения об ошибке. Более подробная информация о причинах оповещения содержится в файлах с расширением .LOG и указанием имени траектории.

9. Файлы изображений. Имеется возможность в любой точке траектории сохранить изображение на дисплее для последующего использования.

10. Файлы CAD. Имеется возможность экспорта файлов в формате .DXF для дальнейшего использования в CAD-программах.

11. Специальные файлы трассировки. Позволяют выделять часть файла трассировки для создания демо-версий. Расширение этих файлов имеет вид .PTF — Proof Professional Demo Trace File.

ПП Proof обладает HELP, позволяющим уточнить формат необходимой команды, кнопка подсказки всегда присутствует в основном меню.

9.2.3. Выполнение основных действий

При инсталляции ПП Proof к основному меню Windows автоматически добавляется меню Proof. До начала процесса анимации некоторые кнопки остаются серыми. Рассмотрим главные возможности ПП Proof.

Модификация пространства

В ПП Proof для представления геометрической информации используются действительные числа с плавающей точкой. Анимация может быть представлена в любом масштабе, и пользователь сам выбирает масштаб и четкость изображения. На рис. 9.2 представлена логика преобразования пространства. Блок *1* задает начальные координаты изображения в виде пары 2D-чисел (x, y), затем блок *2* производит математическое преобразование размещения, масштаба, угла поворота, ориентации, затем блок *3* производит управление четкостью изображения: 640×350, 640×480, 1024×768 точек.

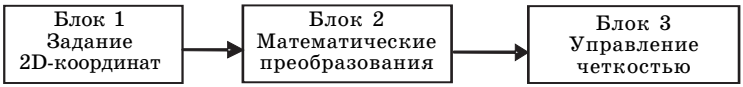


Рис. 9.2. *Схема модификации пространства*

По умолчанию размер начального изображения равен 80 единицам по ширине и при необходимости размер меняется самим пользователем. Причем в данном случае пользователь обладает весьма широкими возможностями по заданию положения объекта и его масштаба. Пункты главного и вспомогательных меню предоставляют достаточно широкий выбор.

Модификация времени

При использовании ПП Proof пользователь имеет дело с тремя видами времени:

1) реальным временем системы, на которое пользователь не оказывает влияния;

2) масштабируемым временем анимации, которое может растягиваться или сжиматься (сравните со временем моделирования гл. 4)

в зависимости от процессов функционирования исследуемого объекта;

3) временем машинной реализации процесса анимации.

Обычно анимация проходит в выбранном для процесса масштабе времени, называемом «единицей времени анимации» (ЕВА), при этом скорость отображается на экране дисплея в виде «ЕВА в секундах машинной реализации». Например, если ВД моделирования и время анимации заданы в секундах, то ЕВА = 1.0. ЕВА можно изменить в любой момент. Причем для ПП Proof не имеет значения, в каких единицах измеряется время моделирования (секунды, минуты, сутки и т. д.), а выбор скорости анимации зависит от пользователя. По умолчанию скорость анимации равна 6 ЕВА, но это не имеет никакого значения, поскольку скорость легко меняется с помощью меню изменения скорости. Масштабы могут быть уточнены в любой момент вызовом скрытых экранов.

Выбор объектов

Зачастую от выбора объекта движения зависит успех процесса анимации или презентации. Поэтому в ПП Proof эта функция достаточно хорошо поддерживается. Пакет позволяет производить следующие операции с объектами движения: создавать — CREATE; разрушать — DESTROY; размещать в координатной сетке — PLACE (x, y); изменять форму — SHAPE, цвет — COLOR, скорость движения — SPEED; поворачивать — ROTATE; перемещать по определенной траектории — PATH или из одной точки в другую — POINT. Объект определяется один раз до начала процесса анимации. Имя объекта создается на основе простых правил: имя должно начинаться с буквы, длина имени не превышает 16 символов (после первой буквы может быть любая комбинация символов). Разное написание делает объекты различными; так, например: CAR, car, Car — три разных объекта, в то же время написание команд типа TIME, time,Time соответствует одной и той же команде. Объект в случае разрушения не может попасть в итоговый отчет. Если по какой-либо причине необходимо передать информацию о разрушенном объекте, то следует запомнить его одновременно с разрушением в новом файле с расширением .ZZZ. Набор команд основного и всплывающих меню позволяет осуществлять все манипуляции с объектом просто и оперативно.

Задание траекторий и движения

Задавать траектории можно непосредственно в модели или ПП Proof или извне, используя возможности Basic, C++, Fortan, Pascal, GPSS/H, GPSS PC. Траектории задаются в виде прямых и дуг. Плавность движения объектов осуществляется за счет большой частоты повторения кадров (до 70 кадров в секунду). В ПП Proof используют-

ся два типа движения: направляемые и не направляемые. Направляемые движения основываются на начальной информации, содержащейся в файлах траектории, и объект движется по заданной физически осуществимой траектории (например, конвейер, железная дорога, очередь перед сервером). Не направляемые движения задаются пользователем и могут возникать между двумя точками в пространстве, которые не могут быть заданы изначально в файлах траектории, а вычисляются непосредственно в процессе анимации. Примерами таких движений могут быть размещение детали в случайно освобождающуюся ячейку вертикального штабелера или осуществление случайного подключения на телефонном коммутаторе. Задание имени траектории привязывает осуществление движения именно к ней, в случае не направляемого движения надо задавать каждую пару координатных точек и время перехода к другой координате. Необходимо отметить, что ПП Proof позволяет запоминать координаты движения и останавливать движение объекта, если конечный пункт его траектории занят предыдущим динамическим объектом. После освобождения траектории остановленный объект продолжает свое движение.

Дополнительные возможности

ПП Proof позволяет осуществлять много других функций. Необходимо отметить, что реализация функций постоянно совершенствуется, появляются новые возможности, которые надо не описывать, а рассматривать непосредственно на ПП. Однако о некоторых важных функциях кратко упомянем в этом разделе. На экране дисплея кроме траекторий и движущихся объектов отображаются различные ***сообщения***, содержащие статистические данные о процессе анимации (время, скорость движения, расстояние), описание объекта (например, для самолета — название авиакомпании, номер рейса), статус объекта (занятость, загрузка). Сообщения можно помечать разным цветом, чтобы выделить их приоритетность. Работа с сообщениями требует определенного навыка, позволяющего избежать корреляции между командами и сообщениями разного уровня. Весь процесс анимации можно оснастить звуковым сопровождением (музыка, спецэффекты).

Подводя итоги, следует отметить, что именно рассматриваемый ПП Proof на сегодняшний день является наиболее эффективным, гибким, открытым, что позволяет решать задачи повышения наглядности и эффективности процесса ИМ на GPSS/H.

Глава 10
ИНТЕГРИРОВАННЫЙ ПАКЕТ SLX

§ 10.1. ПРЕДСТАВЛЕНИЕ О ЯЗЫКЕ SLX

Знакомство с языком SLX предваряю словами благодарности президенту Wolverine Software Corp. Джеймсу Хенриксену, который предоставил в распоряжение автора ряд ценных материалов, на основе которых написана эта глава. Она не является пересказом «Инструкции пользователя» и содержит лишь описание некоторых отличительных особенностей языка SLX, его возможностей и небольшого примера использования [14].

10.1.1. Введение в SLX (Simulation Language with Extensibility)

Язык SLX относится к новому поколению ПП Wolverine Corp. и основывается на широких возможностях ЯИМ GPSS/H. Традиционный подход версий GPSS, базирующийся на построении модели из блоков, обладает целым рядом преимуществ: позволяет достаточно просто и быстро создавать МФ, применять ЯИМ для широкого класса разнообразных систем, обеспечивать единую основу для математического представления. Вместе с тем, используя традиционный подход, приходится сталкиваться с рядом проблем.

1. Набор операторов ЯИМ имеет тенденцию к разрастанию по мере того, как усложняются моделируемые системы и возрастает их номенклатура. Простое добавление новых операторов в известной мере начинает разрушать концептуальную целостность начальной идеи ЯИМ.

2. Не вполне очевидно, как выбирать оптимальную комбинацию операторов, способную описать новые возникающие ситуации, отображение которых не вполне отвечает начальным возможностям операторов.

3. Для многих практических пользователей уровень абстракции выше уровня его понимания, так, например, производственный мастер с большой для себя пользой мог бы применять ИМ, но не знает, как описать модель.

Учитывая эти и ряд других обстоятельств, Wolverine Corp. на протяжении ряда лет вела разработку языка, развивающего возможности традиционного ЯИМ, построенного на базе операторов. При разработке языка SLX решались три глобальные проблемы: введение

уровневого (послойного) построения (в оригинале layer — уровень, слой); создание механизма организации связей и управления между уровнями; создание внешнего интерфейса, обеспечивающего доступ к уровням. Для решения этих проблем пришлось решать следующие технические задачи.

• Построить такую иерархию хорошо организованных уровней, которая позволяла бы пользователю SLX большую часть времени уделять решению практических задач на верхних уровнях иерархии, но при необходимости осуществлять простой переход на нижние уровни.

• Создать механизм перехода вверх и вниз между уровнями.

• Создать интерфейс, ориентированный на ОС Windows, который: 1) был бы скрыт на высших уровнях; 2) немедленно был готов к использованию при запросе; 3) мог осуществлять эффективную работу на нижнем уровне.

На рис. 10.1 представлена пирамида SLX.

Язык обеспечивает большие возможности для моделирования современных систем, описываемых СИ-подобными языками. Язык представляет собой многоуровневую (послойную) структуру с ядром SLX в основании пирамиды, далее в середине — традиционный ЯИМ (GPSS/H) и на вершине — специализированные и прикладные диалекты языка.

Наиболее важной характеристикой языка является его многоуровневая архитектура. Эффективность применения такой структуры зависит от ряда обстоятельств.

• Уровни должны быть хорошо продуманы и четко выражены. На каждом уровне содержится минимально необходимое число функций. Развитию языка способствовала многолетняя успешная практика работы с GPSS/H, который и является основой нового языка. В некоторых случаях коды начального файла GPSS/H могут быть перенесены для использования в SLX. В других случаях можно сократить, упростить или модифицировать алгоритмы GPSS/H или даже устранить дефекты программы.

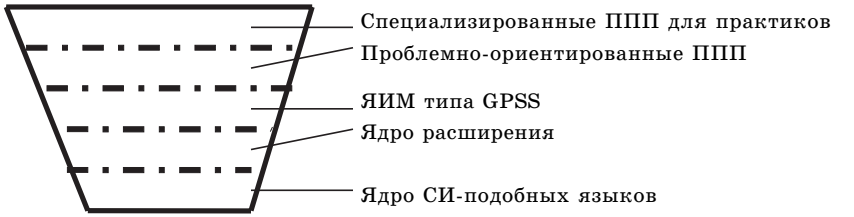


Рис. 10.1. *Пирамида SLX*

• Уровни не должны находиться далеко друг от друга. Во многих языках программирования используется многоуровневая структура, но обычно между уровнями существует большой разрыв. Например, язык может обеспечивать в качестве начальной концепции объектно-ориентированное построение, но не учитывать его в процедурно-ориентированном описании, как это допускают языки типа Фортран или СИ. Проблема состоит в том, что в этом случае рассматриваются лишь два уровня, далеко отстоящие друг от друга. На одном рассматривается большое число деталей реализации СИ, которые могут быть использованы в ЯИМ и расширить возможности СИ. С другой стороны, система контроля ошибок ЯИМ не может быть использована в СИ. Ядро SLX, написанное на СИ-подобном языке, позволяет пользователю не отбрасывать детали более низкого уровня, делая язык более мощным. Следовательно, SLX включает возможность комплексной проверки предполагаемого возникновения ошибок, таких как ссылки в конце массива данных или использование неверных величин. Переход с уровня на уровень производится весьма просто.

• Механизм перехода с уровня на уровень весьма эффективен и базируется на абстрактном представлении. Более высокие уровни обеспечивают более абстрактное описание, чем нижние. Детали описания нижних уровней скрыты в описании более высоких уровней. Язык обеспечивает представление как данных, так и абстрактных процедур. Подобно СИ, SLX обеспечивает возможность определять новые типы данных и агрегировать их в создаваемые объекты. Механизм процедурной абстракции весьма эффективен, включает макросы и возможность введения новых определений, позволяя создавать петли, дополнительные аргументы, расширение логики и т. п.

• Язык SLX позволяет осуществлять связь с другими языками и программами. Если у пользователя возникает необходимость обратиться к коллекции функций СИ/СИ++ непосредственно из SLX, то следует поместить эти функции в динамическую библиотеку (DLL) и обеспечить аргументами, отвечающими на вызов. SLX автоматически генерирует хедер-файлы СИ/СИ++(.h), которые определяют объекты SLX в синтаксисе СИ. Дадим краткое описание динамической библиотеки.

10.1.2. Особенности языка SLX

В компиляторах традиционных языков программирования *модули* программы преобразуются в объектный код, который непосредственно исполняется компьютером или преобразуется интерпретатором (рис. 10.2).

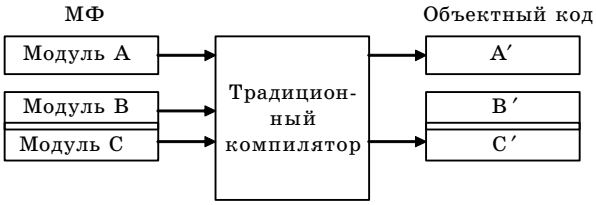


Рис. 10.2. *Архитектура традиционного компилятора*

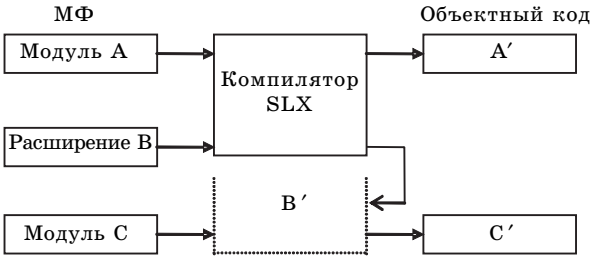


Рис. 10.3. *Архитектура компилятора SLX*

В SLX логика работы компилятора расширена. В процессе компиляции расширение осуществляется за счет внутренних возможностей компилятора. На рис. 10.3 представлен пример расширения модуля С за счет расширения модуля В, причем этот процесс может повторяться неоднократно.

Высказав эти предварительные соображения, рассмотрим некоторые особенности построения языка.

Особенности построения

• *Уровни пирамиды SLX*

Рис. 10.1 достаточно хорошо представляет разделение уровней SLX. Ядро SLX достаточно мало, но содержит все необходимые элементы ИМ. В результате тщательного отбора разработчики выделили из GPSS/H наиболее важные, корневые операторы. Ядро имеет следующие основные характеристики:

— число потребных операторов весьма мало, благодаря чему базовые операторы могут комбинироваться таким образом, чтобы описать наибольшее число возникающих ситуаций;

— ядро имеет непосредственный доступ. В традиционном построении GPSS операторы не допускают прямого доступа до тех пор, пока они не будут переведены в надлежащее состояние (логические переключатели, очереди и т. п.). Такая функция как wait until сложна для реализации, поэтому не удивительно, что она очень редко ис-

пользуется в других ЯИМ. Парадоксально, что команды, трудные для реализации, наиболее легко понимаются пользователями. Эти сложности исключены появлением в SLX директивы *wait until*;

— доступ к основным элементам ядра гарантирует точность и адекватность модельного представления. В пирамиде SLX выполняется правило: «чем ниже вы опускаетесь, тем более точная модель получается». На уровне ядра точность виртуально не ограничена;

— ядро обеспечивает вычислительные возможности при работе с СИ-подобными программами. Большинство ЯИМ меньше внимания уделяют вычислительным возможностям, а больше обращают внимание на управление параллелизмом процесса ИМ;

— все элементы ядра, включая взятые из ядра СИ, активно контролируются, возникновение ошибок типа неполного запоминания массивов, неверного синтаксиса диагностируется в процессе исполнения и приводит к его прерыванию.

Каждый более высокий уровень шире нижнего (см. рис. 10.1). Это означает, что более высокие уровни включают в себя все особенности нижних уровней. И каждая комбинация приспособлена для выполнения частных требований. Сравним, например, использование ОБ SEIZE в GPSS/H, который моделирует поведение одноканального сервера, и SEIZE в SLX, который является подпрограммой 9-линейного исполняемого кода. Наиболее важные функции этого кода — исполнение команды *wait until*, реализующей ожидание освобождения ОБ, и сбор статистики между уровнями GPSS/H и ядра SLX. Размер кода продиктован соображениями уменьшения величины и возможности контроля в процессе исполнения.

• *Механизм связей между уровнями*

Создание уровней — это решение только первой проблемы. Вторая, не менее важная — создание механизма связей, который позволял бы расширять существующие уровни или создавать новые поверх старых. Более высокие уровни обеспечивают больший уровень абстракции, чем нижние, поэтому детали нижних уровней скрыты при переходе на верхний уровень. SLX обеспечивает представление как данных, так и процедур. Подобно СИ, SLX обеспечивает определение новых типов данных и строит *объекты*, которые включают в себя эти данные.

Слово «объект» вызывает много эмоций и ожиданий благодаря широко распространенному пониманию объектно-ориентированного программирования (ООП). На создание SLX повлияли идеи ООП, тем не менее он не является полностью объектно-ориентированным языком. Такое утверждение звучит странно в наши дни, так как ряд ПП, претендуя на объектную ориентированность, на самом деле достаточно

далеки от этого. SLX же, не претендуя на объектную ориентированность, имеет больше элементов архитектуры ООП, чем некоторые другие ООП-языки. В SLX используются объекты двух видов. Первый вид — пассивные объекты, используемые для моделирования единиц, подобных геометрическим построениям в Proof, т. е. когда поведение объекта не меняется, или, например, операторы Facilites, Queues, Storages ЯИМ GPSS/H, неизменяемые ресурсы типа автостоянки и т. п. Второй вид — активные объекты, поведение которых меняется в процессе исследования. Примером таких объектов являются покупатели супермаркета. Таким образом, активные объекты аналогичны транзактам GPSS/H. Ряд единиц могут рассматриваться либо как пассивный, либо как активный объект. Так, при одноканальном обслуживании с дисциплиной типа FIFO сервер воспринимается как пассивный объект, если же сервер может выполнять ряд действий, то он моделируется как активный объект (например, мясник, продающий мясо покупателям, стоящим в очереди, — пассивный объект, он же — разделывающий тушу, складывающий мясо в холодильник, делающий заказ поставщику и т. д., — активный объект).

На более высоком уровне процедурной абстракции SLX использует макросы и новые предписания, содержащие расширенную логику, создание петель, дополнительных аргументов, которые достаточно просто вводятся в структуру SLX.

• *Возможности интерфейса SLX*

Третьей проблемой было создание внешнего интерфейса, позволяющего проводить отладку и контроль процесса моделирования. Возможности ОС Windows позволяют наблюдать в процессе исполнения за данными модели, кодами МФ, списком событий и т. д. При желании коды МФ могут рассматриваться с любым уровнем детализации. Рассмотрим, например, ОБ SEIZE GPSS/H: при разных уровнях рассмотрения его можно принимать за неделимый элемент, а на нижнем уровне рассматривать его более подробно. На рис. 10.4 представлено окно SLX Calls & Expansions, которое отображает, в каком месте модели находится пользователь и путь его прихода в эту точку. На рисунке зафиксировано текущее положение. Нажимая на различные уровни дерева Calls & Expansions, можно переходить с уровня на уровень, отвечая на вопрос, как в действительности пользователь приходит к точке останова процесса. Окно может быть открыто как для общих данных, так и для индивидуального элемента движения.

Правильно обращаясь к соответствующим окнам SLX, можно получить развернутую информацию о данных и/или МФ. На рис.10.5 показано окно представления данных, а на рис. 10.6 — окно установки точек прерывания отладчика и /или показа МФ.

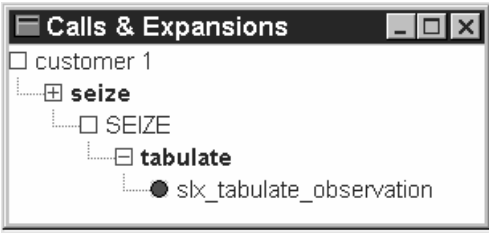


Рис. 10.4. *Окно Calls & Expansions*

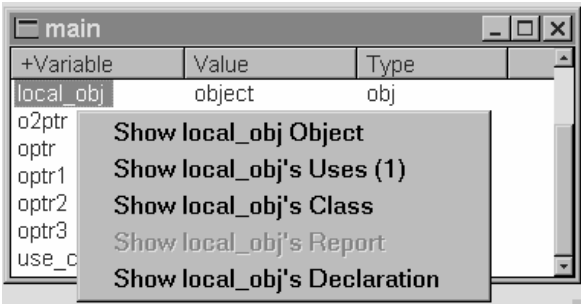


Рис. 10.5. *Представление данных варианта (Sample Data Display Options)*

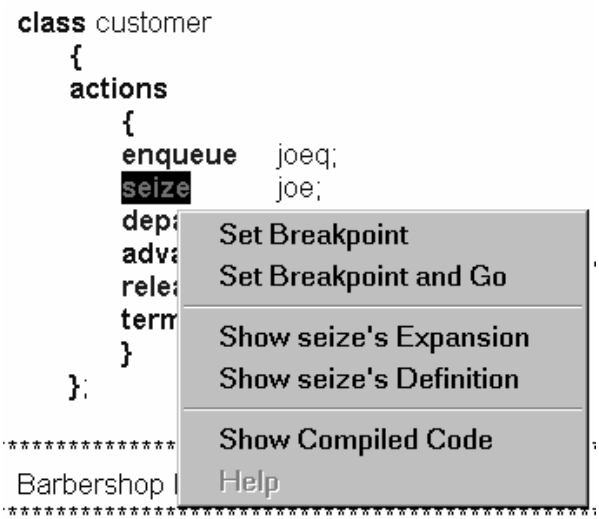


Рис. 10.6. *Окно установки точек прерывания отладчика*

Сочетание трех описанных выше характеристик делает SLX весьма эффективным инструментом исследования.

• *Запись команд и директив SLX.*

Одной из наиболее применяемых форм расширения возможностей SLX можно назвать использование новых команд и директив, являющихся, по сути дела, макросами (в понимании традиционных языков). Отличие состоит в том, что команды чаще работают на уровне утверждений, чем на уровне определений.

Для команд и директив языка SLX можно выделить четыре особенности:

— прототип, определяющий синтаксис утверждения или команды;

— дополнительную логику и возможность создания петель управления внутри утверждения;

— возможность введения в утверждение или команду текста, различаемого компилятором SLX;

— наличие диагностических утверждений, позволяющих выдавать исчерпывающие сообщения при появлении ошибок.

Описание прототипов команд и директив производится с использованием метаязыка, который позволяет определить следующие виды компонентов команд и директив:

выражения, задаваемые пользователем;

ключевые слова, определяемые пользователем;

вспомогательные (не обязательные) компоненты;

список объектов, т. е. повторяемые компоненты;

символы пунктуации.

Основные особенности SLX обеспечивают возможности создания разнообразной логики, петель управления, расширения, диагностики. Большинство языков, использующих макросы, требуют наличия специальных подпрограмм или команд для их определения. Обычно эти подпрограммы отличаются от тех, которые используются в основной программе. Например, команды *if, else, endif* в языке СИ плохо используются для условного описания макросов, а их синтаксис отличается от принятого в СИ. В отличие от СИ, язык SLX не имеет специальных команд, используемых для определения утверждений, а сами новые команды *time delays, fork, wait until* позволяют решать многие задачи моделирования. Причем информация считывается из МФ и запоминается в структурах данных, определяемых пользователем. В дополнение к новым директивам SLX поддерживает традиционные макросы и модули расширения, применимые как при компиляции, так и в процессе исполнения.

Особенности ядра SLX

Объекты SLX могут иметь ряд стандартных команд и директив, которые включают в явном виде идентификаторы исполняемого кода. Начальные директивы исполняются при инициализации объекта, конечные — при уничтожении объекта, атрибуты создания отчета — при получении директивы, атрибуты действия — при описании активного объекта. Число элементов, поддерживающих процесс ИМ, не велико, однако понимание особенностей этих элементов затруднительно. К числу таких элементов относится обобщенная форма директивы *wait until,* которая не является изобретением SLX и встречается в других языках, но в SLX она выполняет особую роль. Поэтому ниже рассмотрим более подробно некоторые особенности ядра SLX, которые призваны подчеркнуть легкость обучения и простоту их использования.

• *Объекты и указатели объектов*

Выше уже было определено, что в SLX имеется два рода объектов. Пассивные, у которых не существует собственной стратегии поведения и управление которыми осуществляется другими объектами (детали, перемещаемые от станка к станку, согласно воле менеджера). Для читателей, знакомых с языком СИ, эти объекты аналогичны команде *structs*. Второй тип объектов — активные, т. е. определяемые своей линией поведения (рабочий в цеху). Подчеркивалось, что объекты в разной ситуации могут быть либо пассивными, а потом активными, либо сохранять свое состояние во время всего процесса ИМ (пример с мясником в супермаркете).

Начальные свойства записываются так же, как это делается в языке СИ.

Для введения нового объекта используется оператор *new*, который переводит указатель на новый объект:

```
pointer (customer) cust;
cust = new customer
```

Активные объекты имеют директиву действия, когда используется команда *activate,* инициализируется атрибут *puck,* помещая объект в СТС, т. е. в состояние, готовое к исполнению:

```
activate new butcher;
```

• *Определение пака (puck)*

Puck (эльф, озорной) является программным элементом SLX, определяющим время задержки или ожидания освобождения сервера. Манипуляция с паками позволяет осуществлять в процессе ИМ квазипараллелизм и является основным механизмом моделирования появления событий.

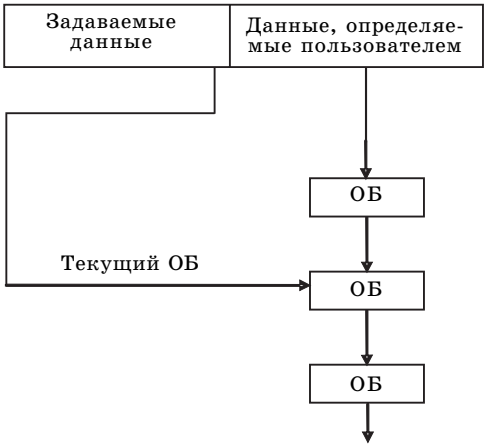


Рис. 10.7. *Традиционное представление потока транзактов*

Язык ИМ GPSS/H, впрочем, как и все остальные версии, построен на представлении потока транзактов, и все внимание сосредотачивается на этих единицах движения, проходящих от ОБ к ОБ и использующих ресурсы. При этом используется два вида данных (рис. 10.7): задаваемые пользователем и определяющие единицу движения; задаваемые для определения траектории, состояния и размещения транзакта (текущий ОБ, следующий ОБ).

При рассмотрении в GPSS/H модели супермаркета транзактами можно представить покупателей, для которых пользователем могут быть заданы вероятность посещения того или иного отдела, количество выбираемого товара и т. п. К задаваемым данным можно отнести следующую исполняемую директиву, следующее состояние ОБ, приоритет и т. д. Приоритет относится к задаваемым данным потому, что его значение может быть изменено программой непосредственно в процессе ИМ. Все данные, определяемые пользователем, должны считываться и записываться с помощью программы, доступной пользователю.

В SLX функциональность транзактов скрыта на нижнем уровне, и в ядре нет транзактов как таковых. Роль транзактов в данных, определяемых пользователем, играет класс объектов (*object class*). Носителем задаваемых данных в SLX является пак (*puck*). Каждый объект SLX является членом какого-то класса и имеет копию в данных объекта. Директива, исполняемая объектом, содержится в свойствах действия (*actions property*) класса, и любая процедура нижнего уровня подчиняется этим свойствам действия. В SLX для выбранного объекта можно иметь несколько паков (рис. 10.8).

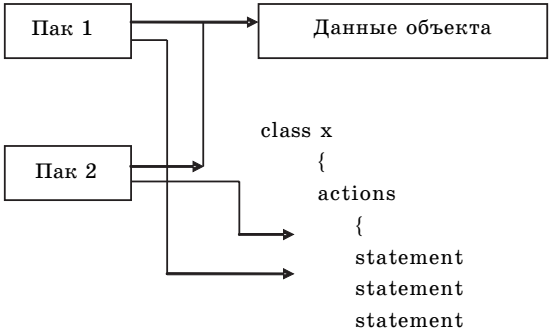


Рис. 10.8. *Пример применения двух паков*

• *Внешний и внутренний параллелизм*

В SLX параллелизм реализуется двумя путями: осуществлением связей между объектами (внешний параллелизм) и многократным обеспечением действия самого объекта (внутренний параллелизм). При внешнем параллелизме поровну (1:1) осуществляются связи между объектами и паками, что функционально эквивалентно потоку транзактов. Внутренний параллелизм достигается определением более одного пака на активный объект (см. рис. 10.8).

Введение нескольких паков достигается применением директивы *fork*. Предположим, что при разработке модели цеха необходимо описать работу станка, способного производить три операции одновременно, причем некоторые узлы станка участвуют во всех трех операциях. Данные, описывающие эти узлы, должны быть легко доступны внутри фрагмента модели для всех трех операций.

На рис. 10.9 показана схема описания активного объекта с использованием директивы *fork*. Каждая директива *fork* определяет новый пак для станка. Каждый потомок пака размещается в СТС, готовый для исполнения и заключенный в фигурные скобки { }, предшествующие директиве *fork*. Начальный (родительский) пак продолжит исполнение при следующей директиве. После исполнения второго пака объект (станок) обслуживает три пака, каждый из которых имеет прямой доступ к общим данным, описывающим станок в целом. Таким образом, станок моделирует три одновременных действия.

Если в языке существует только возможность создания клонов, а определение директивы *fork* отсутствует, объединение трех одновременных операций весьма затруднительно, но не невозможно. Рассмотрим, например, ОБ SPLIT в GPSS/H, позволяющий создать ряд клонов. Главная трудность возникнет при попытке запоминания

```
class machine
        {
        "Declarations for variables local to the machine"

        actions
                {
                fork
                        {
                        "actions for operation 1"
                        }

                fork
                        {
                        "actions for operation 2"
                        }

                "actions for operation 3"
                }
        };
```

Рис. 10.9. *Пример одновременного действия трех паков*

данных для трех образованных транзактов, так как каждый транзакт имеет идентичную информацию. Поэтому придется создавать глобальные переменные. Причем надо помнить, что в GPSS/H изменить собственные атрибуты активного транзакта можно просто, но изменить атрибуты других практически невозможно. Поэтому придется создавать набор глобальных переменных для каждого станка, если их несколько. При изменении состава станков сбор и обработка статистических данных будут весьма затруднительны. Поэтому введение директивы *fork* упрощает процесс моделирования, позволяя без труда оперировать с несколькими паками.

• *Время ожидания wait until*

Как только динамические объекты начинают движение по модели, могут возникнуть задержки двух видов: задаваемые задержки (типа ОБ ADVANCE в GPSS/H) и вызванные состоянием модели. В SLX задаваемые задержки определяются директивой *wait until*, а задержки, определяемые состоянием модели, — командой *control*. Последняя команда используется как ключевое слово в переменных декларациях SLX:

```
control integer     count;
control boolean     repair_completed;
```

Ключевое слово *control* сообщает компилятору SLX, что при каждом появлении этого слова значение управляемой величины изменяется. Контроль осуществления этой операции производится с помо-

щью наблюдения за паками, ожидающими изменения уровня или значения переменной. Такие ожидания описываются директивой *wait until*:

```
wait until (count > 10);
wait until (repair_completed);
```

можно задать и более сложные условия типа:

```
wait until (count >= 10
    or repair_completed
    and not repairman_busy);
```

SLX также поддерживает неопределенные (*indefinite*) ожидания, задаваемые пользователем. Для их задания необходимо реализовать три этапа:

1) пак, приходящий в точку ожидания, должен быть доступен для другого пака, что достигается помещением его в набор (*set*);

2) пак исполняет предписание директивы *wait*, не дожидаясь выполнения команды *until*;

3) в соответствующий момент МВ исполнение другого пака приводит к реализации директивы *reactivate*, что делает активным ожидающий пак:

```
optimistic_event_time = "some expression"

wait until    (time == optimistic_event_time
               or "some other condtion");
```

Если пак ждет больше оговоренного предписанием времени, то он покидает систему, не выполняя своих функций (состояние *renege*). Во многих языках моделирования эта операция реализуется достаточно сложно, поэтому рениджинг осуществляется путем встраивания специальных блоков. В SLX эта операция осуществляется достаточно просто и тогда, когда это необходимо.

• *Комплекты (Sets)*

SLX включает возможности для определения и манипулирования с комплектами объектов. Комплекты могут быть однородными, т. е. состоящими из одинаковых объектов, и универсальными, включающими в себя разнородные объекты. Определение объектов производится с помощью одного или нескольких атрибутов, задающих либо дисциплину обслуживания, либо номер члена комплекта. В универсальных комплектах один из атрибутов задает тип объекта.

Особенности расширения

SLX представляет собой платформу, на базе которой могут быть построены другие средства имитации. Механизм расширения позволяет придать известным средствам новые особенности. Это очевидно проявляется в поисковых системах. Обычно в большинстве языков программирования для укрупнения операций используют макросы, структура которых ограничена или отличается по синтаксису от языка к языку. В этом смысле SLX сам является макросом, содержащим небольшое число основных определений и позволяющим чтение файла даже в процессе выполнения предписания.

В языке активно используются методы контроля процесса моделирования. Сочетание известных и вновь предлагаемых методов является еще одним преимуществом языка. Следует отметить два нововведения. Первое — возможность непосредственного использования особенностей среднего уровня языка. Обычно при использовании GPSS/H исследователи пользуются особенностями верхнего уровня, редко прибегая к включению возможностей среднего уровня. Второе — язык позволяет при контроле использовать особенности его предписаний, что интересно, в первую очередь, для тех, кто создает имитационные пакеты верхнего уровня. Объединенные вместе, они позволяют значительно расширить возможности языка. Поэтому SLX хорошо адаптируется к разным целям разных исследователей.

Архитектура SLX хорошо приспособлена для целей обучения моделированию. Обычно обучение моделированию начинается с понимания набора блоков и тренировки на хорошо определенных примерах. Набив таким образом руку, начинают строить модели, игнорируя ряд возможностей языка и неправильно используя другие возможности. Так, например, при начальном изучении GPSS/H чаще всего запоминают формулу: «активный объект — пассивный сервер», хотя язык допускает и такое построение, как «пассивный объект — активный сервер». Для пользователей пакетов высокого уровня, особенно созданных на основе графического построения, изучение основ построения языка практически не интересно. Хорошо это или плохо — вопрос спорный. Приверженцы языков высокого уровня считают, что это хорошо, их более консервативные оппоненты считают, что нельзя принимать решения при низком уровне знаний. В SLX мало число предписаний среднего уровня, поддерживающих процесс моделирования.

Несмотря на то, что SLX вполне самодостаточен, обладает высокой мощностью и гибкостью, тем не менее его возможности могут быть расширены за счет связи с другим ПО, хорошо сочетающимся с SLX. Например, если вы располагаете набором функций языка СИ, они могут быть вызваны непосредственно из SLX.

• *SLX и динамическая библиотека*

Для вызова функций СИ/СИ$^{++}$, находящихся в динамической библиотеке, необходимо в SLX задать аргументы этих функций и имя файла в DLL. Интерфейс SLX имеет меню, обращаясь к которому можно создавать СИ-подобные файлы. Как уже упоминалось, SLX содержит скрытую информацию, и при манипуляции с функциями СИ ее можно принимать во внимание. Как только происходит первое обращение к DDL, проверяется наличие в DLL функции с именем *connect,* при обнаружении она вызывается первой и помечается указателем, позволяющим обращаться к ней внутри SLX. Эта функция позволяет без труда вызывать необходимые функции из DLL. Вызовом функции *disconnect* можно закрыть все открытые файлы DLL.

• *SLX и возможности анимации*

Wolverine Software разработала интерфейс для связи между SLX и Proof Animation, используя способности директив SLX. Proof (см. гл. 9) применяет поток ASCII-команд, которые создают и разрушают объекты, меняют параметры их движения, форму, цвет и т. д. Директивы SLX способны генерировать общие с Proof команды, обладающие одинаковым синтаксисом, например:

```
place 27 on loop,
```

команда Proof может иметь вид

```
PA_place objectID on “loop”;
```

В примере “27” и “loop” — переменные компоненты команды Proof *place on*. В SLX имя “27” воспринимается как переменная величина объекта ID и представляет “loop” как константу потока.

Интерфейс SLX-Proof позволяет записывать команды Proof в файлы для последующей анимации или непосредственно передавать их в DLL для анимации в реальном времени. И, наконец, в SLX существует возможность чтения входных файлов Proof, запоминания их в структуре данных SLX и перезаписи новых данных во входные файлы. Так, например, геометрические характеристики входных файлов, модифицированных при применении Proof, доступны для программ SLX и, более того, могут быть модифицированы непосредственно программами SLX.

• *SLX и HLA(High Level Architecture)*

Интерфейс для SLX может быть использован для соединения моделей SLX с инфраструктурой моделирования (run-time infrastructure — RTI), позволяющей осуществлять расширенное моделирова-

ние на уровнях HLA (Strassburger, Schulze, Klein and Henriksen 1998). Интеграция заключается в соединении функций C++, которые располагаются между SLX и RTI. Объединение SLX и HLA обладает эффектом синергизма, который придает архитектуре SLX новые свойства, которые позволят еще больше расширить возможности SLX. Для специалистов, знакомых с HLA и желающих развивать такой способ моделирования, возникает новая альтернатива использовать все преимущества SLX и развивать методы моделирования на основе языков высокого уровня, таких как C++ или ADA.

§ 10.2. ПРИМЕР ИСПОЛЬЗОВАНИЯ SLX

В качестве примера рассмотрим гипотетическую конвейерную систему (пример Д. Хенриксена [14]). На базе этого примера проиллюстрируем все основные особенности SLX.

Описание конвейерной системы

Для конвейерной системы (рис. 10.10) надо составить модель на языке SLX.

Четыре сборщика работают в изолированных секциях длиной 7,5 м, изделия поступают со склада и упаковываются в картонные коробки размером $35 \times 50 \times 70$ см (поступление коробок подчиняется равномерному закону). Время упаковки меняется от 0 до 10 с в зависимости от расположения сборщика, который первоначально располагается посередине секции. Как только коробка упакована, она располагается перпендикулярно оси ленточного конвейера. Коробка может быть поставлена на ленту только при наличии свободного места, большего, по крайней мере, длины коробки сантиметров на 20. Лента движется со скоростью 0,3 м/с и может останавливаться при необходимости. Сборщики ленивы и не делают лишних шагов, чтобы отыскать свободное место на ленте. Лента конвейера на три метра длиннее линии упаковки. Лента конвейера соединена с 10-метровой лентой накопителя, которая также движется со скоростью 0,3 м/с. Коробки образуют очередь в конце накопителя, ожидая погрузки на кольцевой конвейер. Инфракрасный датчик размещен посередине накопителя, каждый раз коробка, проходя мимо датчика, прерывает луч; если прерывание превысит 3 с, то ленточный конвейер останавливается и стоит до тех пор, пока не появится луч датчика. Кольцевой конвейер движется со скоростью 0,6 м/с. Для размещения коробки на кольцевом конвейере требуется по одному метру свободного пространства с каждого конца коробки. Для размещения коробки на кольцевой конвейер требуется 2,5 с, т. е. время ожидания коробок

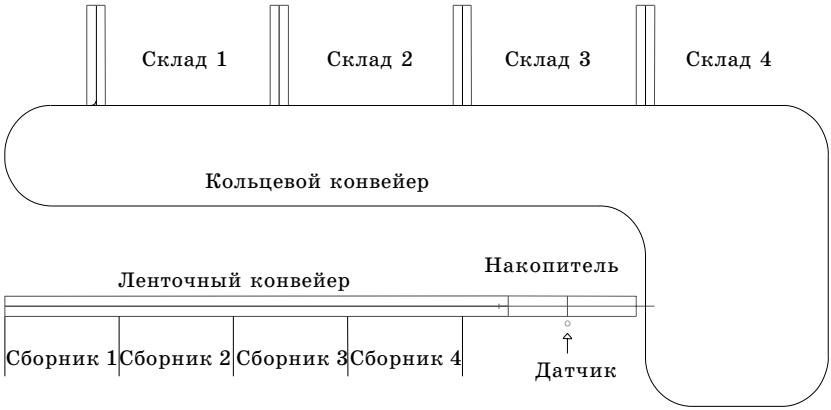


Рис. 10.10. *Пример производственного конвейера*

в накопителе не должно превышать этого времени. Каждая коробка имеет адрес назначения в одном из четырех складов, расположенных вдоль кольца через 12 м.

Длина кольцевого конвейера равна 140 м. Когда коробка достигает нужного места, она перемещается на 6-метровый погрузочный конвейер, движущийся со скоростью 0,15 м/с. Приход коробки к концу погрузочного конвейера означает выход из системы. Перегрузка коробки с кольцевого на погрузочный конвейер занимает минимум 10 с. Если вторая коробка с тем же адресом приходит раньше 10 с, то она направляется на второй круг для очередной попытки, таких циркуляций может быть 2, 3, 4.

Создание параллелизма

Как упомянуто в § 10.1, в языке SLX используются два типа объектов. Пассивные — без собственной стратегии поведения. В нашем примере коробки, действия над которыми выполняются другими объектами, являются пассивными объектами. Эти объекты похожи на структуры в языке C (struct). Активные объекты в примере — упаковщики и лица, перегружающие коробки с конвейера на конвейер. Активные объекты определяются введением нового оператора — пака (puck), введение нового активного объекта предваряется записью *activate new,* превращая его в пак и помещая в список текущих событий, например:

Activate new Picker(25.0,50.0)

Паки являются членами СТС и определяют времена задержки и нахождения в установленном состоянии. Манипуляция с паками является основным механизмом, позволяющим отслеживать возник-

новение событий во времени и обеспечивать имитацию параллелизма. Симулятор SLX может рассматриваться как менеджер паков. Обычно во всех версиях GPSS элементы движения носят название транзактов. Транзакты определяются двумя видами данных: задаваемыми пользователем и представляющие собой атрибуты. В рассматриваемом примере упаковщики представляют собой транзакты, имеющие атрибуты в виде левого и правого конца зоны упаковки, текущего положения и т. д. Изменяемыми данными могут быть приоритет, следующая исполняемая директива и т. п. Все определяемые пользователем данные могут считываться и записываться с помощью кода пользователя. В SLX функциональность транзактов распадается на независимые компоненты нижнего уровня, которые, по существу, не являются транзактами. Роль данных транзактов, определяемых пользователем, играет класс объекта (*object class*), роль задаваемых данных в SLX играет пак (*puck*). Директивы, исполняемые объектами, содержатся в свойствах действия (*actions property*) класса объекта, а все процедуры скрыты на нижнем уровне.

В модели конвейера представлены оба вида параллелизма:

— внешнего, так, четыре упаковщика используют, хотя и косвенно, возможность перемещения в пространстве вдоль ленты конвейера, что является ресурсом с позиции GPSS, основанного на потоке транзактов;

— внутреннего, когда возможно одновременно оперировать с несколькими паками, а в нашем примере, когда активный объект модели (упаковщик) имеет возможность проведения действий в течение 10 с, пока другая коробка снята с ленты и расположена на упаковочном столе, и 40 с, пока коробка достигнет правой границы и не выйдет из секции. Это достигается с помощью директивы *fork*:

```
fork
        {
        advance 40.0;           // box travel time;
        "Remove load from loading dock."
        terminate;
        }
advance 10.0;                   // Cycle time
```

В указанном примере директива *fork* определяет второй пак на столе упаковки. Оба пака расположены в СТС, но один из них (потомок) выделен фигурными скобками { }. Основной пак (родительский) продолжает исполнение с 10,0-секундной задержкой, оговоренной директивой. Таким образом, первые 10 с 40-секундного пребывания

в секции пака-потомка совпадают с 10-секундной задержкой родительского пака.

Большинство ЯИМ поддерживают только внешний параллелизм, во многих из них допустимо клонирование транзактов, что похоже на директиву *fork* SLX. Однако преимущество директивы SLX состоит в том, что допускается запись разной информации в произведенных паках (см. § 10.1). Использование директивы *fork*, даже в этом простом примере, подчеркивает большую гибкость и эффективность ее применения. Когда родительский пак задерживается на 10 с, то с ним можно производить необходимые операции без опасения насчет расположения на ленте конвейера пака-потомка. В более сложных ситуациях возможность осуществления внутреннего параллелизма становится насущно необходимой.

Применение директивы wait until

Директива *wait until* также поддерживает все условия, связанные с заданием времени. Модельное время в SLX задается командой **time.** Рассмотрим 6 возможных случаев задания времени ожидания:

```
wait until (time < expression);
wait until (time <= expression);
wait until (time == expression);
wait until (time!= expression);
wait until (time > expression);
wait until (time >= expression);
```

Первые два вида не могут использоваться в процессе ИМ, так как время моделирования не может принимать отрицательных значений. Третий случай определяет скрытое событие, возникающее в оговоренное время, что эквивалентно применению ОУ ADVANCE, хотя это не создает дополнительных трудностей при использовании директивы *wait until*. Если выражение позволяет времени уменьшаться, то появляется сообщение об ошибке исполнения, так как время не может идти в обратную сторону. Четвертый случай вводит возможность обращения со временем как с управляемой величиной, в связи с этим рассмотрим возможности пятого и шестого случаев. Пятый случай подобен третьему, за исключением того, что время, по определению, уменьшаться не может, и поэтому ошибка исполнения не появляется. Шестой случай распадается на два этапа: вначале скрытое событие появляется в оговоренное время, а затем, когда это время достигнуто, начинает работать механизм *wait until*, используя время как управляемую величину.

Директива *wait until* позволяет объединять определения, основанные на времени, с другими определениями. В примере конвейерной

системы эта возможность широко используется. Рассмотрим упрощенную версию подпрограммы, созданную для описания перемещения коробки из точки A в точку B по ленте конвейера. В модели объект LoadStatus описывает каждую коробку на конвейере. Объект LoadStatus содержит целочисленные переменные, называемые ChangeCount. В каждый момент времени другой объект осуществляет действия, которые влияют на рассматриваемый объект, увеличивая LoadStatus ChangeCount.

Например, если объект LoadStatus располагается впереди другого объекта LoadStatus, то ChangeCount объекта должен быть увеличен. На практике это правило осуществляется очень просто: во всех случаях, когда появляются изменения для конкретного объекта LoadStatus, они касаются только соседнего объекта. Приведем фрагмент подпрограммы, описывающей перемещение объекта LoadStatus из точки A в точку B:

```
pointer(LoadStatus)Load;
"set Load's current position to B."
forever
        {
        wait until (ConveyorSpeed > 0.0);
        Distance = B - CurrentPosition(Load);
        ArrivalTime = Distance / ConveyorSpeed;
        LocalCount = Load -> ChangeCount;
        wait until (time == ArrivalTime
        or Load -> ChangeCount!= LocalCount);

        if (Load -> ChangeCount == LocalCount)
                break;  // exit the forever loop
        }
```

В этом фрагменте движение происходит по петле, пока объект LoadStatus не достигнет точки B. Рассмотрим внешние причины, которые могут повлиять на движение объекта LoadStatus внутри петли. Во-первых, может изменяться скорость движения конвейера, когда скорость изменяется, то атрибуты ChangeCount всех коробок на конвейере получают приращение. В приведенном фрагменте директива *wait until* предусматривает, что скорость конвейера должна быть больше нуля. Это исключает деление на нуль в теле петли. Во-вторых, в каждый момент времени при нахождении в петле вычисляется время поступления коробки и задается с помощью функции CurrentPosition ее текущее положение. Директива *wait until* испол-

няется либо для случая, когда достигнуто заданное значение времени, либо когда появляются изменения, действующие на объект LoadStatus. Подпрограмма защищена от случайных сбоев, что позволяет корректно осуществлять приращения в ChangeCount для любого объекта LoadStatus, в котором происходят какие-либо изменения.

Особенности расширения

SLX предоставляет возможности расширения для построения программ на более высоких слоях. Проиллюстрируем это утверждение рассматриваемым примером.

В процессе создания программы выяснилась необходимость построения иерархии объектов и подпрограмм, позволяющих описывать ленточный конвейер, накопительную область, круговой конвейер и датчики. В принципе, выполняя заказ, на этом можно было бы и остановиться, передав заказчику необходимые подпрограммы и инструкции пользователя. Однако в примере хочется показать технологию создания подпрограмм и интерфейса, достаточно простого для применения. Набор необходимых для нашего примера директив будет рассмотрен далее, а здесь рассмотрим пример создания одной из директив, а именно CVR_Send (аббревиатурой CVR будем обозначать директивы, относящиеся к нашему примеру). Директива CVR_Send используется для размещения объекта LoadStatus на конвейере. Если никаких дополнительных условий не предусматривается, то под ограничением движения подразумевается крайняя правая точка конвейера. Пак, который исполняет директиву CVR_Send, не ожидает, когда объект LoadStatus достигнет этого ограничения, вместе с тем он может последовательно проверять, достиг ли объект LoadStatus этого ограничения. Альтернативная директива CVR_Ride, наоборот, заставляет ожидать, когда объект LoadStatus достигнет конца ленточного конвейера. Тогда будем считать,что директива CVR_Send является асинхронной, а директива CVR_Ride — синхронной формой транспортирования объекта LoadStatus.

Формат записи директивы CVR_Send имеет вид (рис. 10.11).

Отметим, что вид конвейера, на который размещается объект LoadStatus, никак не обозначен. Это сделано из тех соображений, что по умолчанию объект первоначально размещается на ленточном конвейере. Первая линия задает компоненты директивы CVR_Send; имя, предваряемое знаком #, вводится пользователем при каждом применении директивы. Прямые скобки [] используются для выделения группы необязательных определений. Символ @ впереди ключевого слова “to” сообщает SLX о том, что можно игнорировать обычный смысл “to” и воспринимать его как ключевое слово директивы

```
pointer(LoadStatus)    Load;
CVR_Send Load to 47.5;
CVR_Send Load;
      statement CVR_Send                    #Load [@to #Destination];
       definition
          if (#Destination!= "")
               expand (#Load, #Destination)   "ConveyLoad(#, #);\n";
          else
               expand (#Load, #Load)   "ConveyLoad(#, (#) -> LoadConveyor ->
                                        ConveyorLength);\n";
```

Рис. 10.11. *Определение директивы CVR_Send*

CVR_Send. Таким образом, в SLX "to" является резервным словом. Секция definition определяет расположение директивы CVR_Send в нижних слоях директив SLX. Внутри секции definition используется директива *expand*, применяемая для определения кода нижних слоев SLX. Директива *expand* определяет одну или несколько линий выходов, которые входят в начальный поток данных компилятора SLX. Список выражений может пополняться и редактироваться внутри создаваемых выходных линий. Все записи, помеченные символом #, считаются редактируемыми значениями.

Подобный подход аналогичен использованию макросов во многих языках программирования, однако при этом существуют ограничения по организации внутренней логики и синтаксису основного языка. В SLX язык макросов органично связан и не налагает никаких ограничений. Более того, в SLX возможно чтение файла, являющегося частью расширенной директивы. В директиве CVR_Send применяемая форма расширения зависит от того, применяется или нет условие "to". В противном случае вызов директивы ConveyLoad генерирует подпрограммы нижних уровней.

Создание слоев

При создании модели конвейера были приняты следующие условия.

1. Структура данных создана для конвейеров, датчиков и коробок (нагрузки). Пока не определена структура данных, необходимо ввести понятие статуса объектов — *status*, который должен быть представлен в виде отдельных независимых объектов LoadStatus, доступных пользователю (в нашем случае — коробки). Такой подход имеет ряд преимуществ:

позволяет скрыть детали нижнего слоя от кода более высоких слоев;

позволяет записывать подпрограммы нижних слоев в виде, не зависящем от типа нагрузки;

допускает множественные связи между объектами LoadStatus и способами их транспортирования. Так, при перегрузке с одного конвейера на другой (если это происходит не мгновенно) объект LoadStatus присутствует на обоих конвейерах. Например, если круговой конвейер останавливается до того, как объект попадает на него с ленточного конвейера, последний тоже должен быть остановлен.

2. Функции нижнего уровня должны быть записаны для определения позиции объекта, например:

CurrentPosition(Load)

3. Функции должны быть записаны таким образом, чтобы при сканировании данных конвейера можно было определить расположение нагрузки в зонах (*zones*) конвейера, что достигается записью

FindFirstLoadInZone and
FindLastLoadInZone

4. Для определения состояния зон конвейера было разработано четыре функции:

WaitForZoneFull
WaitForZoneNotFull
WaitForZoneEmpty
WaitForZoneNotEmpty

Функция WaitForZoneFull не позволяет объекту LoadStatus быть погруженным на конвейер и наоборот, функция WaitForZoneEmpty переводит указатель объекта LoadStatus в режим доступности к размещению на конвейере, эта функция позволяет располагать объект на ленте случайным образом. Правда, при этом необходимо придерживаться заданных расстояний перед и за коробкой.

Функция WaitForZoneFull служит в том числе и для введения в модель датчиков.

Датчики должны моделироваться для очень узкого диапазона свободного правого конца конвейера; как только этот диапазон будет нарушен, начинается действие функции WaitForZoneFull, указатель переключается на занятость зоны и идентифицирется нагрузка, которая нарушила условие.

5. Для моделирования работы конвейера были определены директивы:

```
CVR_Conveyor
CVR_Sensor
CVR_LoadClass
CVR_ConveyorSpeed(c) = s;
CVR_LoadSpeed(l) = s;
CVR_Place ... on
CVR_Remove ... from
CVR_Send
CVR_Ride
```

Первые три директивы определяют компоненты конвейерной системы, следующие две определяют скорости конвейера и нагрузки (коробок). Последние четыре служат для описания размещения, удаления и транспортировки нагрузки.

ЗАКЛЮЧЕНИЕ

Широкое внедрение информационных технологий во все сферы жизни человеческого общества (CALS, CASE, CIDM, IDEF, ERP-технологий) заставляет с еще большей интенсивностью обращаться к вопросам ИМ систем, становящихся все более сложными. На этом фоне ЯИМ GPSS/H и его развитие в виде языка SLX не только не потеряли актуальность, но и приобретают все больше сторонников во всем мире. В меньшей степени это относится к России, так как среди ряда специалистов бытует мнение, что ЯИМ GPSS (имеется в виду его первая версия) потерял свою актуальность и не может соперничать с другими ЯИМ. Целью настоящего пособия было не только желание опровергнуть эту в принципе неверную точку зрения, но и постараться показать возможности и преимущества ЯИМ GPSS/H.

Здесь уместно напомнить, как начали расходиться пути разработчиков различных версий GPSS. К первой версии GPSS V ее разработчик Д. Гордон (G. Gordon — IBM Corp.) не обращался последние 30 лет, что и дало основание специалистам во всем мире считать, что этот ЯИМ вышел в тираж. Однако возникли два направления развития ЯИМ. Первое развивалось специалистами Minuteman Computer Corp. А. Коксом, Т. Шпрингером (A. Cox, T. Springer), которые не стремились сохранять преемственность с ЯИМ IBM и пошли по пути развития пользовательского интерфейса типа Windows, в меньшей степени в GPSS PC и в значительной степени в последней версии GPSS World. Второе, развиваемое специалистами Wolverine Software Corp. Дж. Хенриксеном, Р. Крейном (J. Henriksen, R. Crain), не отошло от классического представления Д. Гордона, сохранило текстово-ориентированное описание, но усилило все преимущества ЯИМ, в то же время практически исключило все недостатки, присущие другим версиям. Перечень этих преимуществ достаточно подробно изложен в гл. 4.

Вместе с тем с огорчением следует констатировать, что ЯИМ GPSS/H мало известен в России. Благодаря усилиям специалистов российской фирмы «Элина-компьютер» и американской корпорации Minuteman Computer в России и СНГ продвигается ЯИМ GPSS World, являющийся развитием GPSS PC, в котором, кстати, для целей анимации используется ПП корпорации Wolverine SW Proof Animation. По ряду характеристик GPSS World уступает ЯИМ GPSS/H, но в силу малой известности в России ЯИМ GPSS/H не может конкурировать с ПП «Элина-компьютер».

Поэтому пособием автор в какой-то мере решил исправить определенную несправедливость по отношению к ЯИМ GPSS/H, который

весьма удобен для целей преподавания в вузах и эффективен при решении практических задач. Многочисленные примеры, приведенные во второй части, в известной степени иллюстрируют широкий спектр применения ЯИМ GPSS/H, хотя можно назвать гораздо больше областей его применения.

Сегодняшний, сузившийся благодаря Интернету мир позволяет достаточно быстро приобрести описанные в ПП непосредственно у разработчика (координаты приведены в главе 4).

При возникновении у читателя каких-либо вопросов можно обратиться к автору по e-mail < bnm@aanet.ru > или < overall@list.ru> .

Автор заранее благодарен за любые конструктивные замечания и открыт для сотрудничества.

ПРИЛОЖЕНИЯ

Приложение 1

Операторы блоков GPSS/H[1]

Название	Операнды и их значение	Пункт	По умолчанию	Примечание
ADVANCE (продвинуть)	A — среднее время обслуживания B — модификатор времени	5.1.1, A3	0 0	Возможна функция
ALTER (изменять)	A — имя объекта, меняющего атрибут B — число объектов, меняющих атрибут C — приоритет объекта D — изменяемое значение E-F — ограничения на изменяемые величины	Нет	Нет Нет Нет Нет Нет	Возможно использование дополнительного кода
ASSEMBLE (соединять)	A — определяет число Хакт, соединяемых в единый набор	5.1.1, B2	Нет	Разрушаются после исполнения
ASSIGN (назначать)	A — определяет имя назначаемого параметра B — заменяемая величина, модифицирующая A C — имя оцениваемой функции B D — тип параметра	5.1.1, E1	Нет Нет Нет Нет	Может иметь знак ±
ATNWAIT (запрет)	Нет, дает возможность ожидать следующего обращения к тестовому режиму	Нет	Нет	
BCALL BCLEAR BCLOSE BFILEDEF BGETLIST BGETSTRING BPUTPIC BPUTSTRING BRESET BRMULT BSTORAGE	Группа ОБ, выполняющая функции, аналогичные ОУ, непосредственно в модуле исполнения МФ. Появление литеры B делает их применение очень удобным, так как их можно применять в процессе исполнения программы	5.1.1 З		Перевод и значения операндов см. в прил. 2, в списке ОУ без первой литеры B

Продолжение прил. 1

Название	Операнды и их значение	Пункт	По умолчанию	Примечание
BUFFER (демпфер)	Операндов нет, останавливает исполнение Хакт ОБ и запускает сканирование СТС	Нет	Нет	После сканирования процесс продолжается
CHANGE (изменять)	A — имя переопределяемого ОБ B — имя ОБ, встающее на замену в A	Нет	Нет	Имя или число ОБ, меняющих название
COUNT[2] (считать)	A — имя подсчитываемого параметра B — нижний предел проверяемого атрибута C — верхний предел проверяемого атрибута D — выражение для сравнения с атрибутами E — определяет подсчитываемый атрибут	Нет	Нет Нет Нет Нет Нет	Операнд C применяется для устройств, памятей, логики с их идентификаторами
DEPART (покидать)	A — имя или номер покидаемой очереди B — число единиц уменьшения очереди	5.1.1, Ж1	Нет 1	Хакт может занимать несколько единиц
ENTER (входить)	A — имя или номер памяти B — число единиц занимаемой памяти	5.1.1, Б3	Нет 1	Хакт может занимать несколько единиц
EXAMINE (проверять)	A — имя или номер проверяемой группы B — проверка числовых значений B = 0 – проверка, является ли Хакт членом группы C — имя ОБ при невыполнении условия B	Нет	Нет Нет Нет Нет	Хакт направляется по адресу C, если проверка не верна
EXECUTE (исполнять)	A — определяет имя исполняемого ОБ, что позволяет Хакт идти непоследовательно	Нет	Нет	При запрете входа Хакт остается в ОБ EXECUTE

Название	Операнды и их значение	Пункт	По умолчанию	Примечание
FAVAIL (доступно)	A — определяет доступное устройство или освобожденное после исполнения FUNAVAIL	Нет	Нет	Определяет класс доступных устройств
FUNAVAIL[3] (не доступно)	A — определяет недоступные устройства B — прерывание до освобождения C — имя ОБ, где остается Хакт [D] — определяет группу изменяемых параметров E — определяет прерванные транзакты F — имя ОБ, куда идут прерванные транзакты G — кольцо обращений к занятому ОБ [H] — только если операнд G имеет код RE	Нет	Нет Нет Нет Нет Нет Нет Нет Нет	Операнды B, E, G могут кодироваться обнулением либо “CO”, “RE” (см. прим. 2)
GATE[4] (клапан)	A — определяет имя и состояние рассматриваемых объектов B — имя непоследовательного ОБ	5.1.1, Б4	Нет Нет	Подробнее см. в тексте пособия
GATHER (собирать)	A — число Хакт, собираемых в набор, первый Хакт исполняется ОБ	5.1.1, В3	Нет	ОБ может содержать любое число Хакт
GENERATE (производить)	A — среднее время или функция типа RV (см. прил. 3) A′ — любой СЧА или СЛА B — модификатор времени C — время прихода первого транзакта D — число создаваемых Хакт с конкретного ГСЧ E — приоритет Хакт F-I — параметры Хакт всех типов	5.1.1, А1	0 Нет 0 0 ∞ 0 12РН	Если операнд A задан функцией RV, то операнд B не используется

Название	Операнды и их значение	Пункт	По умолчанию	Примечание
INDEX (указать)	A — имя параметра, к которому прибавляется B B — значение, прибавляемое к A	Нет	Нет Нет	Результат запоминается в Хакт
JOIN (объединить)	A — имя группы, к которой присоединяется Хакт B — значение определяется операндом A	Нет	Нет Нет	Возможен транзактный или числовой вид
LEAVE (покинуть)	A — имя освобождаемой памяти B — число единиц освобождаемой памяти	5.1.1, Б3	Нет 1	Хакт может занимать несколько единиц
LINK (связь)	A — определяет имя списка пользователя B — если ДО FIFO, то в конец списка [C] — логический переключатель	Нет	Нет Нет Нет	Наличие C переводит в условную форму
LOGIC (логика)	A — имя логического ключа, переключаемого в соответствии с кодом	5.1.1, Б4	Нет	Дополнительный код “S, R, I” после имени ОБ
LOOP (петля)	A — имя меняемого на единицу параметра B — имя ОБ петли, если Хакт больше нуля	5.1.1, Г3	Нет Нет	Значение параметра сравнивается с нулем
MARK (отметить)	A — отсутствует, значение АС1 запоминается в MB A — присутствует, АС1 запоминается в Хакт	Нет	Нет	AC! — абсолютное время
MATCH (пара)	A — имя ОБ, которому подбирается пара, что синхронизует поток Хакт одного набора	5.1.1, В4	Нет	Хакт должен быть членом набора
MSAVEVALUE (сохранение значения матрицы)	A — имя значения матрицы B — номер строки матрицы C — номер столбца матрицы	Нет	Нет Нет Нет	При наличии знака + или − у операнда A значение складывается или вычитается

Название	Операнды и их значение	Пункт	По умолчанию	Примечание
	D — замещаемое значение E — тип слова матрицы (MX, MH, MB, ML)		Нет MX	
PREEMPT (прерывать)	A — имя управляемого устройства B — определение уровня прерывания C – имя ОБ, куда направляется прерванный Хакт D — параметр запоминания времени обслуживания [E] — логическое условие	5.1.1, Б1	Нет Нет Нет Нет Нет	При E, кодируемом RE, проверяется то же устройство
PRINT (печатать)	A — детализация печати B — верхний порог детализации C — тип выводимых объектов	Нет	1 Задан Тип	Уровень вывода задается пользователем
PRIORITY (приоритет)	A — задает новый уровень приоритета [B] — при коде BUFFER запускает сканирование	5.1.1, Е2	Нет Нет	Приоритет меняется в процессе ИМ
QUEUE (очередь)	A — имя очереди, в которую встает Хакт B — число единиц очереди, добавляемых Хакт	5.1.1, Б1	Нет 1	Хакт может добавить несколько единиц
RELEASE (освободить)	A — имя освобождаемого устройства, исполнение ОБ освобождает сервер	5.1.1, Б1	Нет	Используется только в паре с ОБ SEIZE
REMOVE (перемещать)	A — имя группы, из которой объект перемещается B — максимальное число перемещаемых Хакт C — при наличии ОБ работает в числовом виде D — атрибут Хакт, используемый для сравнения E — выражение для сравнения при отборе F — адрес альтернативного ОБ	Нет	Нет Нет Нет Нет Нет Нет	Возможно задание арифметических или минимаксных условий

Название	Операнды и их значение	Пункт	По умолчанию	Примечание
RESHEDULE (перезапись)	A — определяет ИН Хакт в СБС B — определяет АС1, заменяя текущее МВ Хакт в A [C] — имя заменяемого ОБ D — имя альтернативного ОБ	Нет	Нет Нет Нет Нет	ИН меняется с помощью СЧА. Операнд C не обязателен
RETURN (возвращать)	A — имя устройства, освобождаемое после прерывания	5.1.1, Б2	Нет	Парный ОБ с ОБ PREEMPT
SAVAIL (доступная память)	A — имя доступной для использования памяти или емкости, доступной для использования	Нет	Нет	Память вводится в ОБ SUNAVAILE в недоступность
SAVEVALUE (сохранение значения)	A — имя сохраняемого значения (+ или −) B — значение, модифицирующее A C — тип сохраняемой величины (XF, XH, XB, XL)	Нет	Нет Нет X	Значение B может иметь любой вид
SCAN (сканировать)	A — имя группы, из которой сканируется Хакт B — определяет атрибуты Хакт C — сравниваемая величина D — определяет первый удовлетворяющий атрибут E — определяет параметр, куда заносится D F — имя альтернативного ОБ	Нет	Нет Нет Нет Нет Нет Нет	Возможны арифметические и минимаксные условия
SEIZE (захватить)	A — имя устройства, захваченного Хакт. При невыполнении условий препятствует входу Хакт	5.1.1, Б1	Нет	Парный ОБ с ОБ RELEASE
SELECT[5] (выбрать)	A — имя параметра, в котором запоминается число объектов, удовлетворяющих условию	5.1.1, Д2	Нет	Операнд D использует дополнительные коды

Название	Операнды и их значение	Пункт	По умолчанию	Примечание
	B — нижний предел сравниваемых объектов C — верхний предел сравниваемых объектов D — число для сравнения при наличии кода E — определяет СЧА, сравниваемые с B и C F — имя альтернативного ОБ		Нет Нет Нет Нет Нет	(см. прим. 4). Операнд E сравнивает арифметические выражения
SUNAVAIL (не доступна)	A — имя памяти, которая не доступна. Хакт ожидает изменения состояния	Нет	Нет	Может определяться и часть емкости
TABULATE (табулировать)	A — имя таблицы для записи данных B — взвешивающий коэффициент	5.1.1, Ж2	Нет 1	Таблица должна быть задана в модуле 1
TERMINATE (разрушить)	A — величина, вычитаемая из заданного числа в счетчике свершений (см. § 5.3)	5.1.1, А2	0	Значение A может быть любым
TEST (проверить)	A — выражение для сравнения (левая часть) B — выражение для сравнения (правая часть) C — адрес направления Хакт при невыполнении	5.1.1, Г2	Нет Нет Нет	Сравнение производится каждый раз в СТС
TRACE (след)	Операнда нет, исполнение ОБ переключает флаг разряда и определяет траекторию	Нет	Нет	Применение отладчика эффективней этого ОБ
TRANSFER (перемещение)	A — имеет ряд видов (см. п. 5.1.1) B — определяет следующий ОБ C — определяет альтернативный ОБ	5.1.1, Г1	Нет Нет Нет	Широко применяется в безусловном и статистическом виде
UNLINK[6] (прекращение связи)	A — имя списка пользователя B — имя ОБ, куда возвращается Хакт	Нет	Нет Нет	Работа со списком (-ами) пользователя экономит ма-

Продолжение прил. 1

Название	Операнды и их значение	Пункт	По умолчанию	Примечание
	C — максимальное число вызываемых Хакт D — правило изъятия из списка E — проверка условия соответствия F — адрес альтернативного ОБ		Нет Нет Нет Нет	шинное время, но требует внимания
UNTRACE (отмена следа)	Нет операнда, исполнение ОБ выключает флаг разряда и трассировка отсутствует	Нет	Нет	Лучше пользоваться отладчиком

Примечание 1. Операторы приведены в алфавитном порядке, а не по смысловому признаку, как в тексте пособия. Отмеченным цифрами ОБ требуются дополнительные коды после имени ОБ, если присутствует операнд, отмеченный в тексте примечаний. В [ ] заключены не обязательные операнды или не обязательная информация. Графа «Пункт» содержит ссылку на номер пункта в пособии.

Примечание 2.

Операнд C

Устройства

U — занято или прервано
NU — не занято и не прервано
I — постоянно прерывается
NI — не постоянно прерывается
FV — всегда доступно
FNV — не всегда доступно

Памяти

SE — постоянно пустая
SNE — не постоянно пустая
SF — постоянно полная
SNF — не постоянно полная
SV — постоянно доступная
SNV — не постоянно доступная

Ключи

LR — установлен
LS — переустановлен

Операнд D

Отношения

G — больше
GE — больше или равно

L — меньше
LE — меньше или равно
E — равно
NE — не равно

Примечание 3.

У ***операнда B*** может быть три варианта:

отсутствует — Хакт постоянно контролирует устройство, в противном случае — перерыв до доступности устройства

CO — контроль транзакта, в противном случае — продолжается использование устройства

RE — использование устройства, в противном случае — возврат к ОБ C

Операнд D используется только тогда, когда B кодируется указанным образом.

Операнд E (три варианта):

отсутствует — Хакт, прерванный в устройстве, ожидает его освобождения

CO — кодируется так, только если операнд B имеет такой же код, Хакт продолжает бороться за вход в прерванное устройство

RE — продолжает соревнование за использование устройства при выполнении одного из трех условий: 1) находиться в ОБ ADVANCE, 2) быть прерванным, 3) прерывать при использовании именно этого устройства

Примечание 4.

Операнд A — коды устройств, памятей и ключей такие же, как и в прим. 2, появляется дополнительно условие сбора.

Условие сбора

M — условие сбора выполнено (ОБ MATCH, ASSEMBLE, GATHER)

MN — условие сбора не выполнено

Примечание 5.

Операнд D — коды устройств, памятей, ключей, арифметических условий такие же, как в прим. 2, 4, добавляется условие минимакса.

Операнд E — только арифметические и минимаксные условия.

Операнд F — при отсутствии операнда Хакт направляется строго в следующий ОБ, при наличии операнда (все виды дополнительных кодов) и невыполнении условия значение параметра определяется операндом A.

Примечание 6.

Операнд D:

отсутствует — Хакт берется из начала списка

[BACK] — Хакт берется из конца списка

$\{BV_j\}$ — список сканируется от начала до конца и выбирается Хакт, для которого выполняется булево условие

Операторы управления и описания

Название	Функции и операнды	Пункт	Примечания
BVARIABLE (булева переменная)	Метка — обязательное имя переменной A — булевское выражение; 1 — истинно, 0 — ложно	Нет	Оценка логических выражений
CALL (вызов)	A — &имя подпрограммы (подпрограмм) [B] — [(аргумент вызываемой подпрограммы)]	Нет	Для связи с внешними ПП
CHAR*N (символьные &)	A — имя &переменной [(B)] — [(размер)]	Нет	Операции с символьными АМП
CLEAR (очистить)	[A] — исключаемые из очистки данные	5.1.2, К1	Позволяет сохранять данные прогона
CLOSE (закрыть)	A — имя файла. При закрытии неоткрытого файла — ошибка	Нет	Закрывает файлы входа/выхода
DO (исполнить)	A — &индекс = начальному значению B — диапазон изменений [C] — [шаг приращения] — по умолчанию 1	5.1.2, Л1	Задает параметры петли управления
ELSE (иначе)	Без операндов. Используется в сочетании с ОУ IF в группе директив при невыполнении условий	Нет	Элемент условных директив
ELSEIF (иначе, если)	A — булевское выражение. Запускает группу директив, проверяющих новое условие	Нет	При ложности начальной проверки
END (окончание)	Без операндов. Обязательный ОУ, исполнение которого прерывает процесс ИМ	5.1.2, И3	Отсутствие — ошибка исполнения
ENDDO (остановка петли)	Без операндов. Прекращает действие петли управления или одной из петель	5.1.2, Л1	Количество в МФ по числу петель
ENDIF (конец условий)	Без операндов. Прекращает действие группы условных ОУ	Нет	Соединяется с ОУ IF

Название	Функции и операнды	Пункт	Примечания
ENDMACRO (конец макроса)	Без операндов. Выдает команду об окончании макроса	Нет	Макрос содержит символ окончания
EQU[1] (эквивалент)	Метка — обязательное имя или символ A — имя одного или нескольких классов объектов B — количество объектов	Нет	Символ может быть одинаковым для разных объектов
EXTERNAL (внешний)	A — &имя внешних подпрограмм. Определяет внешние подпрограммы	4.2; 5.1.3, M1	Несколько имен отделяются запятыми
FILEDEF (определение файла)	Метка — имя файла A — текущее значение или выражение	Нет	Исполнение ОУ для открытого файла — ошибка
FUNCTION[2] (функция)	Метка — имя функции A — независимая переменная B — тип функции	5.1.2, К2	См. п. 4.4.2 и примеры в тексте
FVARIABLE (переменная)	Метка — имя переменной A — выражение с плавающей точкой	4.4.2	См. примеры в пособии
GETLIST[3] (показать список)	A — форма обращения B — одно или несколько определений входов	Нет	Исполнение ОУ производят данные входа/ выхода
GETSTRING[3] (показать ряд)	A — форма обращения B — скалярные переменные, заключенные в скобки	Нет	Применяется, когда не нужен полный набор
GOTO (направить)	A — адрес следующего исполняемого ОУ или директивы отчета	Нет	Применим не для всех ОУ
HERE (сюда)	Метка — имя объекта, куда надо внедрить ОУ. Операндов нет	Нет	Аналогичен директиве Фортрана CONTINUE
IF (если)	A — проверяемое условие. Обеспечивает исполнение ОУ в цепочке условий	5.1.2, Л1	Необходимы только ОУ IF / ENDIF, остальные ELSE / ELSEIF дополнительны

Название	Функции и операнды	Пункт	Примечания
INITIAL[4] (начальный)	A — имя величины	5.1.2, К2	Не допускает сложных скобок
INTEGER (целый)	A — &имя величины [необязательный размер]	5.1.3, М1	Можно задавать через запятую несколько АМП
LET (присвоить)	A — имя АМП = выражению или числу	5.1.2, Л4	Иногда может иметь метку
LIST[5] (список)	A — форма обращения	5.1.3, М4	Позволяет выдавать отчет разного вида
MACRO (макрос)	Метка — имя макроса A — имя файла, куда внедряется макрос [B] — числовая характеристика	Нет	Макрос может содержать 1 пробел, поэтому комментарий — минимум 2
MATRIX[6] (матрица)	Метка — имя матрицы A — тип матрицы B — число строк C — число столбцов	Нет	Этот ОУ является исполняемым
NODUMP (не включать)	A — имя объекта, не включаемого в сообщение об ошибках	Нет	Действует только на отчет об ошибках
NOXREF (исключить ссылки)	Без операндов. Исключает перекрестные ссылки	Нет	Включение ОУ сокращает объем отчета
OPERCOL (переназначить)	A — номер колонки редактора, с которой начинается сканирование первого символа первого операнда	5.1.3, М2	Не может быть меньше 10 и больше 60
PAGE (страница)	Без операндов. Усиливает эффект введения страниц в окончательный отчет	Нет	ОУ не печатается в итоговом отчете
PICTURE (отображение)	Метка — имя отображения [A] — число линий отображения, по умолчанию 1	Нет	Могут быть немедленное и отложенное отображение
PUTPIC[5] (показать вид)	A — форма вызова [B] — список объектов	5.1.2, К5	См. примеры гл. 6

Название	Функции и операнды	Пункт	Примечания
PUTSTRING (отобразить ряд)	A — FILE = имя объекта (B) — (текст сообщения)	Нет	Наиболее простая форма отчета
QTABLE (таблица очереди)	Метка — имя таблицы A — имя очереди B — верхняя граница нижнего разряда C — размах таблицы D — число интервалов	5.1.2	Таблица пополняется каждый раз при исполнении ОБ DEPART
READ (чтение)	[A] — число образов, перезапоминаемых после сохранения с помощью ОУ SAVE	Нет	По умолчанию число образов равно 0
REAL (действительный)	A — &переменная [рамер]. Служит для определения действительных АМП	Нет	Можно записывать несколько АМП
REALLOCATE (переназначить)	[A] — код класса объектов (устройства, памяти и т. д.) [B] — число объектов	5.1.3, М3	В GPSS/H происходит автоматически
REPORT (отчет)	Без операндов. Служит сигналом для подготовки отчета	Нет	ОУ стоит перед последним ОУ END
RESET (возврат)	A — определяет исключения. Исключения сохраняют данные предыдущего прогона	5.1.2, К6	Существует аналог ОБ с литерой B
RMULT (переустановить)	A — новое значение позиции ГСЧ	5.1.2, К7	См. примеры гл. 6 и 8
SAVE (сохранить)	A — имя сохраняемого образа	Нет	По умолчанию сохраняется весь образ
SIMULATE (моделировать)	[A] — указание на предел времени ЦПУ, при S в секундах [B] — указание на сохранение	5.1.2, И1	При отсутствии ОУ в МФ ИМ блокируется
SKIP (обход)	[A] — число пустых линий в отчете для лучшей читаемости	Нет	Не путать ОО SPACE внутри отчета

Название	Функции и операнды	Пункт	Примечания
START[7] (начало)	A — устанавливает начальное значение СС, при временном таймере равен 1	5.1.2, И2	Может иметь операнды B (см. прим.)
STARTMACRO (запуск макроса)	Метка — имя макроса. Операндов нет	Нет	Тело макроса содержит метки начала и конца
STORAGE (память)	Метка — имя памяти A — емкость	5.1.2, К8	Возможен второй формат
SYN (преобразовать)	Метка — определяемый символ A — значение	Нет	Определяемый символ должен быть новым
TABLE[8] (таблица)	Метка — имя таблицы A — форма представления B — верхняя граница нижнего класса C — ширина интервала D — число интервалов	5.1.2, К9	Примеры таблиц см. примеры в гл. 8
UNLIST[9] (остановка отчета)	A — форма представления	Нет	ОУ LIST, UNLIST являются парными
VARIABLE (переменная)	Метка — имя переменной A — целочисленное выражение	Нет	Оценивает часто используемую переменную
VCHAR*N (символьная АМП)	A — &переменная [(размер)]	Нет	Можно определять несколько АМП

Примечание 1. Операторы приведены в алфавитном порядке, операторы, выделенные серой заливкой, являются операторами описания, все остальные — операторы управления. Отмеченные цифрами операторы требуют дополнительную информацию после имени оператора. В [ ] заключены не обязательные операнды или не обязательная информация.

Примечание 2.

Операнд B — в поле операнда применяются следующие коды:

B	булева переменная
C	список пользователя
F	устройство
G	группа
H	(XH) полусловная сохраняемая величина
L	логический переключатель

M(MX) полнословная матричная сохраняемая величина
MB байтовая матричная сохраняемая величина
MH(Y) полусловная матричная сохраняемая величина
ML матричная сохраняемая величина с плавающей точкой
P параметр (тип не задан)
PB байтовый параметр
PF полнословный параметр
PH полусловный параметр
PL параметр с плавающей точкой
Q очередь
RN случайная величина (БСВ)
S память
T таблица
V переменная
X(XF) полнословная сохраняемая величина
XB байтовая сохраняемая величина
XH полусловная сохраняемая величина
XL сохраняемая величина с плавающей точкой
Z функция

Примечание 3.

Операнд B — в поле операнда приняты обозначения :

C непрерывная числовая величина
D дискретная числовая величина
L список числовых значений
E дискретная атрибутивная величина
M список атрибутивных значений
S селектор объектов

Примечание 4.

Операнд A — в поле операнда может быть одно из трех ключевых слов:

FILE = имя файла — определяет принадлежность ко входному файлу
END = метка — определяет имя ОУ, которому передается управление
ERR = появление ошибки — определяет имя ОУ, при исполнении которого появляется ошибка

Примечание 5.

ОУ INITIAL используется для представления сохраняемых величин, матричных сохраняемых величин и логических ключей. Формат записи сохраняется для всех названных объектов , отличаясь только названиями, например:

INITIAL X&SAM,25/XL3,1.5 — для сохраняемых величин
INITIAL MX&REOR(1,1),100 — для матричных величин
INITIAL LS10/LS1-LS5 — для ключей

Примечание 6.

Операнд A — используются три вида записи:

ABS	— для печати всего отчета
CSECHO	— для печати ОУ
MACX	— для печати линий макроса

Примечание 7.

Операнд B — использует следующие коды :

MB	— байт
MH	— полуслово
ML	— плавающая точка
MX	— слово

Примечание 8.

В ОУ START предусмотрены, но практически не используются, операнды [B, C , D]:

B — если стоит индекс NP, то производится отчет. Индекс NP имеет наибольший приоритет по сравнению со всеми другими командами о производстве отчета

C — дает команду по выдаче отчета сразу после обнуления CC

D — если операнд имеет код 1, выдается отчет по всем спискам

Примечание 9.

Операнд A — возможны четыре вида таблиц:

— кодируется как выражение для сбора статистики

— кодируется как выражение, но имеет знак (−), тогда создается дифференциальная таблица

— если имеет код IA, то таблица создается в интервальном виде

— если имеет код RT, то создается интегральная таблица

Приложение 3

Перечень функций, встроенных в GPSS/H

В отличие от прежних версий, язык имеет:

1. Встроенные математические функции:

COS-ACOS, SIN-ASIN, TAN-ATAN, EXP, LOG, SORT, ABS

(косинус-арккосинус и т. д., экспонента, натуральный логарифм, квадратный корень, абсолютное значение — см. гл. 5).

2. Встроенные законы распределения, формат которых покажем на примере бета-распределения. Как известно, в бета-распределении используется бета-функция и два постоянных числа p, q, называемых в языке $\alpha 1$, $\alpha 2$, формат записи

< RVBETA (rns, $\alpha 1$, $\alpha 2$ *)* > (rns — номер генератора БСВ).

Форматы других распределений аналогичны, изменяются только параметры законов. Ниже приведен ряд известных законов. Другие законы могут быть получены с использованием встроенных распределений и СЧА. В профессиональной версии GPSS/H в настоящее время встроено более 30 различных законов. Перечислим некоторые из них:

RVBETA	бета-распределение
RVBIN	дискретное биномиальное распределение

RVERL	распределение Эрланга k-го порядка
RVEXPO	экспоненциальное распределение
RVGAMA	гамма-распределение
RVGEO	геометрическое распределение
RVLAP	распределение Лапласа
RVLNOR	логнормальное распределение
RVNORM	нормальное распределение
RVPSSN	распределение Пуассона
RVPT	распределение Пирсона
RVWEIB	распределение Вейбула

Приложение 4

Описание эха МФ и граф итогового отчета

1. Эхо модельного файла (пример 6.1)

```
STUDENT GPSS/H RELEASE 2.0 (AY130) 24 Nov 2002 11:15:54 FILE: model61.gps
  (1)       (2)    (3) (4)    (5)        (6)        (7)              (8)              (9)
 LINE# STMT# IF DO BLOCK# *LOC OPERATION A,B,C,D,E,F,G COMMENTS

1  1      SIMULATE     Пример 6.1
2  2    * Нотариальная контора
3  3    * Временная дискрета 1 мин
4  4  ********************************************************************
5  5    *         Фрагмент 1 *
6  6  ********************************************************************
7  7    *
8  8    1         GENERATE       0,,,4              число клерков
9  9    2 REPEAT ADVANCE        30,5               время подготовки документа
10 10   3         SEIZE          MANAGER            переход к менеджеру
11 11   4         ADVANCE        8,2                проведение регистрации
12 12   5         RELEASE        MANAGER            уход от менеджера
13 13   6         TRANSFER       ,REPEAT            начало новой операции
14 14   *
15 15 ********************************************************************
16 16   *         Фрагмент 2 (Временной таймер) *
17 17 ********************************************************************
18 18   *
19 19   7         GENERATE       2400       моделирование в течение 40 ч
20 20   8         TERMINATE      1          сигнал на СС,
21 21 *                                     окончание движения транзактов
22 22 *
23 23 ********************************************************************
24 24   *         Модуль управления *
25 25  ************************************************************
```

```
26 26 *
27 27          START  1        установка СС
28 28 *
29 29          END             окончание моделирования
```

Примечание. Цифры, приведенные в скобках, в отчете не печатаются, а приведены лишь для удобства пояснений.

. Колонка LINE указывает на номер строки МФ.

. Колонка STMT указывает на число всех утверждений, в том числе и комментариев

. Колонка IF указывает на наличие ОУ IF (в пособии не используется).

. Колонка DO указывает на глубину использования ОУ.

. Колонка BLOCK# указывает на место, занятое ОБ.

. Колонка *LOC определяет расположение с 1-й по 72-ю колонку.

. Колонка OPERATION содержит название кода операции.

. Колонка A,B,C, ... соответствует операндам A,B,....

. Колонка COMMENTS представляет собой поле комментария.

Примечание. Колонки LINE, DO, IF приведены лишь в силу того, что они используются программой, но в пособии они не применяются.

2. Итоговый отчет

Предупреждение!

1. Комментарии даются непосредственно в тексте отчета и с целью различения их от текста пишутся курсивом.

2. Пояснения к содержанию отчета по устройствам, памятям, очередям даются после описания общей части отчета.

ENTITY DICTIONARY (IN ASCENDING ORDER BY ENTITY NUMBER; "*" => VALUE CONFLICT.)

*(Словарь объектов в порядке возрастания номеров объектов , * $\Longrightarrow$ значимость конфликта)*

Facilities: 1=MANAGER

(Устройство:1= MANAGER)

SYMBOL VALUE EQU DEFNS CONTEXT REFERENCES BY STATEMENT NUMBER

Обозначение Номер Строка Наименование Ссылка на номер утверждения

REPEAT	2	9	Block	13	
MANAGER	1		Facility	10	12

STORAGE REQUIREMENTS (BYTES)

Потребность в памяти (байтов)

COMPILED CODE: 244

COMPILED DATA: 80

MISCELLANEOUS: 0

ENTITIES: 264
COMMON: 10000
————————————
TOTAL: 10588

Simulation begins *(ИМ началось)*
RELATIVE CLOCK: 2400.0000 ABSOLUTE CLOCK: 2400.0000
Относительное время *Абсолютное время*

BLOCK CURRENT		TOTAL
Текущий	*ОБ*	*Общее число*
1		4
REPEAT	3	243
3		240
4	1	240
5		239
6		239
7		1
8		1

—AVG-UTIL-DURING— *Смотри комментарий 2.1*

FACILITY	TOTAL TIME	AVAIL TIME	UNAVL TIME	ENTRIES TIME/XACT	AVERAGE STATUS	CURRENT AVAIL	PERCENT XACT	SEIZING XACT	PREEMPTING XACT
OVEN	0.800	240	8.000	AVAIL	5				

RANDOM STREAM *№ ГСЧ*	ANTITHETIC VARIATES *Антитезы*	INITIAL POSITION *Начальное значениеГСЧ*	CURRENT POSITION *Текущее значениеГСЧ*	SAMPLE COUNT *Объем выборки*	CHI-SQUARE UNIFORMITY χ^2-*вероятность равномерности*
1	OFF	100000	100483	483	0.90

STATUS OF COMMON STORAGE
Состояние общей памяти

9232	BYTES AVAILABLE	*доступно*
768	IN USE	*используется*
880	USED (MAX)	*максимально использовано*

ИМ закончено *Абсолютное время*
Simulation terminated. Absolute Clock: 2400.0000
Общее число исполненных ОБ
Total Block Executions: 1207

Blocks / second: 1207000

Microseconds / Block: 0.83
Elapsed Time Used (SEC)

```
PASS2: 0.06
EXECUTION: 0.00
————————————
TOTAL: 0.06
```

2.1. Описание отчета по устройствам

Для удобства комментариев графам отчета присвоим номера, заключенные в круглые скобки. Все 10 номеров в итоговом отчете даются единой по длине строкой. Если устройств несколько, то для каждого устройства приводится своя информация в порядке появления устройств в МФ.

```
   (1 )        (2)         (3)         (4)         (5)         (6)

                    —AVG-UTIL-DURING—
FACILITY    TOTAL       AVAIL       UNAVL       ENTRIES     AVERAGE
OVEN        TIME        TIME        TIME                    TIME/XACT
            0.800                               240         8.000
                 (7)         (8)         (9)         (10)
            CURRENT     PERCENT     SEIZING     PREEMPTING
            STATUS      AVAIL       XACT        XACT
            AVAIL       100.0       5
```

(1) — наименование объекта GPSS/H, в данном случае устройство.

(2)–(4) — AVG-UTIL-DURING — среднее использование за указанные интервалы времени:

(2) — доля использования устройства за все время (% занятости), вычисляется в долях тысячи, в примере равна 0.800, что соответствует 80 %;

(3) — вычисляет долю доступности прибора, в пособии не рассматриваются варианты профилактики и восстановления, поэтому устройство считается доступным в течение всего МВ, см (8);

(4) — вычисляет долю недоступности устройства и в силу сказанного в (3) обе колонки остаются пустыми.

(5) — число входов Хакт в устройство. Если число входов в устройство равно 0, то отчет для такого устройства не создается.

(6) — показывает среднее время занятости устройства Хакт.

(7) — показывает текущее состояние устройства, в пособии эта графа всегда указывает на доступность устройства.

(8) — процент доступности, в нашем случае всегда 100 %.

(9) — показывает ИН захваченного Хакт в момент производства отчета, если устройство свободно, то эта колонка пуста.

(10) — показывает ИН Хакт, прервавшего обслуживание в силу его большего приоритета.

2.2. Описание отчета по очередям

На этот пункт распространяются все вводные замечания п. 2.1 применительно к очередям.

(1)	(2)	(3)	(4)	(5)	(6)
QUEUE	MAXIMUM CONTENTS	AVERAGE CONTENTS	TOTAL ENTRIES	ZERO	PERCENT ZEROS
TOOLWAIT	6	1.501	146	26	17.8

(7)	(8)	(9)	(10)
AVERAGE TIME/UNIT	$AVERAGE TIME/UNIT	QTABLE NUMBER	CURRENT CONTENTS
296.093	360.246		

(1) — имя очереди, для которой собирается информация.

(2) — максимальное число членов очереди.

(3) — среднее число членов очереди. Приведем пример расчета этого показателя: пусть MB равно 100 дискрет, имеются всего два транзакта, один из которых пробыл в очереди 20 дискрет, а второй 30 (общая сумма 50). В момент времени 40 отношение равно 50/40 =1.25, в момент 50 равно 50/50 =1, в момент 100 равно 50/100=0.5.

(4) — полное число членов очереди.

(5) — число нулевых входов, т. е. проход очереди без задержки и исполнение ОБ QUEUE/DEPART происходит в одно время.

(6) — процент нулевых входов, когда Хакт проходят без задержки.

(7) — указывает, какое время в среднем Хакт проводит в очереди. В отличие от (3), расчет этой величины проводится следующим образом: подсчитывается среднее время пребывания транзактов, пусть эти цифры 20.0, 0.0, 30.0, сумма времен делится на 3, т. е. 50/3 =16.667 независимо от модельного времени.

(8) — среднее время без учета нулевых проходов, в числе транзактов (7) не учитывается 0.0, тогда 50/2 = 25.0.

(9) — указывает идентификатор таблицы в случае ее наличия.

(10) — указывает на число членов очереди при создании отчета.

2.3. Описание отчета по памятям

На этот пункт распространяются все вводные замечания п. 2.1 применительно к памятям .

(1)	(2)	(3)	(4)	(5)	(6)
	—AVF-UTIL-DURING—				
STORAGE	TOTAL TIME	AVAIL TIME	UNAVL TIME	ENTRIES	AVERAGE TIME/UNIT
TUGBOATS	.439			374	.411

(7)	(8)	(9)	(10)	(11)	(12)
CURRENT STATUS	PERCENT AVAIL	CAPACITY	AVERAGE CONTENTS	CURRENT CONTENTS	MAXIMUM CONTENTS
AVAIL	100.0	3	1.316	0	3

(1) — имя или номер памяти.

(2) — часть общего времени, когда память занята.

(3) — часть времени, когда память доступна (в пособии 100 %).

(4) — часть времени, когда память не доступна (в пособии 0 %).

(5) — число памятей, задействованных в процессе ИМ, число занятых памятей воспринимается как число входов, при отсутствии входов отчет не создается.

(6) — среднее время занятости памяти.

(7) — состояние занятых памятей, в пособии всегда AVAIL.

(8) — процент доступности.

(9) — емкость памяти.

(10) — среднее содержание, равное произведению емкости и занятости.

(11) — текущая занятость памятей.

(12) — максимальная занятость

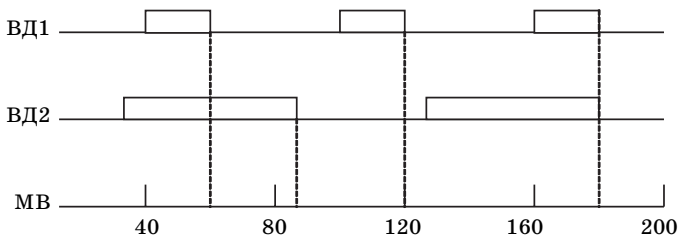
Приложение 5

Ответы к упражнениям

Вводные замечания

Напомним, что МВ никогда не может быть отрицательным и не может изменяться в сторону уменьшения. Следовательно, приход Хакт с любого из ОБ GENERATE всегда отмечается по оси выходных данных конкретного генератора и одновременно сносится на ось МВ (см. примеры к § 5.3). В простых случаях, когда времена детерминированы (примеры отмечены тильдой ~), удобно пользоваться временной диаграммой, которая позволяет понять физические основы происходящих процессов, например:

```
SIMULATE
GENERATE     40
ADVANCE      20
TERMINATE    1
GENERATE     30
ADVANCE      60
TERMINATE    2
START        5
END
```



Из рисунка видно, как происходит процесс ИМ (сопоставьте с результатами работы с отладчиком в пошаговом режиме). В момент 180 времен-

ных дискрет показания СС изменятся до −2, что послужит сигналом к окончанию процесса ИМ.

В более сложных случаях, когда выходные данные отдельных Хакт меняются по какому-либо закону распределения, будут использованы табличные данные, полученные в тестовом режиме. Транзакту, вышедшему первым с генератора, стоящего на первом месте в МФ, присваивается ИН1 и т. д. Отметим, что при случайном законе распределения Хакт, вошедшие в модель позже, т. е. имеющие большие ИН, могут обгонять ранее вошедшие Хакт, что объясняется различием времен обслуживания.

Нумерация ответов привязана к номеру пунктов, а также для простоты поиска имеет сквозную нумерацию.

Пункт 5.5.2

1. Результаты сведены в табл. П.5.1.

Таблица П.5.1

ИН	№ генератора	Время появления
2	2	35.0
1	1	50.0
3	2	70.0

2. Результаты сведены в табл. П.5.2.

Таблица П.5.2

ИН	№ генератора	Время появления
2	2	35.0
1	1	50.0
3	2	70.0
4	1	100.0
5	2	105.0
7	2	140.0

3. Результаты сведены в табл. П.5.3.

Таблица П.5.3

ИН	№ генератора	Время появления
1	1	46.2
2	2	49.9
4	2	85.1

Значение СС в момент окончания ИМ равно 0.

4. Результаты сведены в табл. П.5.4.

Таблица П.5.4

ИН	1	4	5	2	3
Время ухода	185.1	302.2	310.3	349.6	519.8

Из таблицы видно, что Хакт с ИН4 и 5 опережают при терминировании Хакт с ИН2 и 3.

5. Результаты сведены в табл. П.5.5.

Таблица П.5.5

ИН	1	4	5	2	3
Время ухода	322.5	348.4	429.2	537.9	693.4

В силу случайности времен обслуживания их нельзя объединять вместе, так как результаты не будут совпадать (проверьте это обстоятельство в тестовом режиме).

6. Результаты сведены в табл. П.5.6.

Таблица П.5.6

ИН	2	1	4	5
ВД	75.0	100.0	150.0	150.0
СС	2	1	0	–

Процесс ИМ окончится в 150 временных дискрет, Хакт с ИН5 движение не начнет, так как показания СС уже обнулятся.

7. Результаты сведены в табл. П.5.7.

Таблица П.5.7

ИН	1	2	3	4	5	6
ВД	25	25	50	50	75	75

Хакт с первого генератора нумеруются 1, 3, 5, и соответственно, они начинают движение первыми в связанные моменты времени.

8. Результаты сведены в табл. П.5.8.

Таблица П.5.8

№	2	2	1	2	2	1
ИН	2	3	1	4	6	5
ВД	25.0	50.0	50.0	75.0	100.0	100.0

В таблице подчеркивается значение приоритета при движении связанных по времени Хакт; так, транзакты с большим приоритетом движутся первыми.

9. Результаты сведены в табл. П. 5.9.

Таблица П.5.9

ИН	1	3	4	5	1	3	5	4	1	3
МВ	10.0	20.0	30.0	40.0	45.4	59.9	74.0	74.8	77.8	100

Из таблицы видно, что количество транзактов, стоящих в очереди, достигает 5, а порядок выхода из ОБ ADVANCE меняется.

Пункт 6.5.2

10. В результате уменьшения дискреты вдвое среднее время на один транзакт равно 16.001, что незначительно больше прежнего времени, равного 8 мин. Это объясняется тем, что интервал БСВ, из которого берется выборка, хотя и велик (2 147 483 846, см. § 3.3), но конечен, поэтому при уменьшении дискреты и соответственном увеличении времени обслуживания с 30.5 до 61.0 выборка, из которой берутся значения, расширяется вдвое. Это увеличение выборки и соответственно изменение разницы в получаемых значениях приводит к незначительному статистическому расхождению, не учитываемому при проведении практических расчетов.

11. A. Результаты сведены в табл. П.5.10.

Таблица П.5.10

ИН	1	3	4	5
МВ 1-го документа	27.7	29.9	32.3	27.0

B. Второй клерк приходит к менеджеру в 33.5.

C. За время 480.0 нет связанных по времени транзактов.

12. Среднее число N подготовленных документов равно 47.8, тогда:

$$\text{Прибыль} = N \times 300 - \{(500 + 4 \times 350) + N \times 100\} = \\ = 14340 - 6680 = 7760.$$

13. За неделю три клерка подготовят 184 документа:

$$\text{Прибыль} = 184 \times 300 - \{(2500 + 3 \times 1750) + 184 \times 100\} = 29050.$$

За неделю пять клерков подготовят 285 документов:

$$\text{Прибыль} = 285 \times 300 - \{(2500 + 5 \times 1750) + 285 \times 100\} = 45750.$$

По результатам моделирования, увеличение числа клерков увеличивает прибыль.

14. Поскольку для создания всех четырех потоков БСВ используется один и тот же ГСЧ, то перестановка фрагментов приведет к изменению

порядка получения чисел в выборке и это повлияет на окончательные результаты. Как добиться независимости потоков, рассматривается в гл. 8.

15. Ниже представлен МФ, включающий две разные очереди для механиков обоих типов, дополнительные ОБ имеют метку NEWx, а прежние ОБ — метку OLD.

```
        SIMULATE                              временная дискрета: 1 с
*
            GENERATE    420,360,,,5           приход механиков 1-го типа
*                   (Приоритет 5)
  OLD1      QUEUE       TOOLWAIT              создание очереди TOOLWAIT
  NEW1      QUEUE       TYPE1Q                создание очереди TYPE1Q
            SEIZE       CLERK                 начало обслуживания
                                              у кладовщика
  NEW2      DEPART      TYPE1Q                выход из очереди TYPE1Q
  OLD2      DEPART      TOOLWAIT              выход из очереди TOOLWAIT
            ADVANCE     300,90                время обслуживания
            RELEASE     CLERK                 освобождение кладовщика
            TERMINATE   0                     уход из кладовой
*
            GENERATE    360,240,,,10          приход механиков 2-го типа
*                   ( Приоритет 10 )
  OLD3      QUEUE       TOOLWAIT              создание очереди TOOLWAIT
  NEW3      QUEUE       TYPE2Q                создание очереди TYPE2Q
            SEIZE       CLERK                 начало обслуживания
                                              у кладовщика
  NEW4      DEPART      TYPE2Q                выход из очереди TYPE2Q
  OLD4      DEPART      TOOLWAIT              выход из очереди TOOLWAIT
            ADVANCE     100,30                время обслуживания
            RELEASE     CLERK                 освобождение кладовщика
            TERMINATE   0                     уход из кладовой
*
            GENERATE    28800                 транзакт управления после
                                              8 ч
            TERMINATE   1                     уменьшение CC на 1,
                                              конец движения Хакт
*
            START       1                     установка CC = 1, начало
                                              движения Хакт
            END                               окончание процесса ИМ
```

Порядок написания ОБ очередей в МФ может быть произвольным: OLD – NEW или наоборот, так как ОБ очередей никогда не препятствуют входу Хакт.

16. Ожидаемое число объектов на конвейере равно 4, в результате ИМ оно равно 3.971, а максимальное число 8.

17. А. Правильно построенная модель содержит 27 ОБ и результаты ИМ таковы:

Коэффициент использования тельфера	— 0.864
Среднее время ожидания на 4-м этапе	— 12.624
Среднее время ожидания на этапах 6, 7, 1, 2	— 8.724
Среднее время ожидания других требований	— 11.533
Число других требований	— 616

Принимайте эти данные за контрольные, несмотря на вероятностный характер БСВ, значительных расхождений в правильно построенной модели не может быть.

B. Коэффициент загрузки рабочего равен 0.922 .

C. При изменении дисциплины обслуживания надо иметь в виду , что в момент запроса тельфер может быть занят другим вызовом и необходимо ждать окончания работ, такая ситуация возникает не часто и составляет менее 10% случаев потребности рабочего в использовании тельфера . Данные при изменении дисциплины обслуживания таковы :

Коэффициент использования тельфера	— 0.874
Коэффициент загрузки рабочего	— 0.924
Среднее время ожидания на 4-м этапе	— 10.93
Среднее время ожидания на других этапах	— 9.7
Среднее время ожидания других требований	— 11.773

Сравнение с п. A показывает незначительное изменение характеристик.

18. A. Среднее число механиков, ожидающих обслуживания, равно 2.543. Стоимость их простоя за 8-часовой рабочий день равна

$$600 \times 8 \times 2.543 = 12206.4 \text{ р.}$$

B. При стоимости нового кладовщика, равной 30 р./ч, среднее число ожидающих механиков равно 1.297. Стоимость простоя с учетом увеличения зарплаты кладовщика за 8-часовой рабочий день составит

$$600 \times 8 \times 1.297 + 8 \times 10 = 6305.6.$$

Разница в потерях производства уменьшится почти на 6 тыс., что свидетельствует об эффективности найма нового кладовщика.

19. A. Ожидаемый коэффициент использования буксиров равен 0.519. Это подсчитывается следующим образом.

В среднем суда типа A прибывают каждые 3.2 ч и занимают 2 ч работы буксиров: $0.5 \times 3 = 1.5$ ч при швартовке и $2 \times 0.25 = 0.5$ при отшвартовке, это время составляет 0.172 от суммарного времени прихода и разгрузки $(2 / (3.2 + 8.4)) = 0.172$.

Суда типа B занимают 1.25 ч работы буксиров ($2 \times 0.5 + 0.25 = 1.25$) и это время составляет 0.347 от суммарного времени прихода и разгрузки. Эти два коэффициента можно сложить, так как буксиры используются в разное время, и тогда суммарный коэффициент равен 0.519.

B. Среднее время пребывания в порту судов типа A при их большем приоритете уменьшается с 10.732 до 9.623 ч, среднее время для судов типа B увеличится с 3.302 до 4.551 ч.

D. Среднее число буксиров и причалов по результатам моделирования равно 2.67, 5.08, 2.85 соответственно, при ожидаемой оценке эти цифры равны 3, 6, 3.

20. Для правильно построенной модели итоговые цифры:

Коэффициент использования тельфера	— 0.87
Среднее время ожидания на 4-м этапе	— 10.171
Среднее время ожидания на других этапах	— 8.988
Среднее время ожидания других требований	— 9.71
Количество других требований	— 613

При значительном отклонении данных проверьте созданный вами МФ.

21. МФ в значительной степени повторяет МФ примера 6.4, но введены 7 новых ОБ, имеющих метку NEWx, учитывающих переход на вторичную регулировку со значением 1 %.

```
            SIMULATE                    временная дискрета: 1 мин
TESTERS     STORAGE     2               число контролеров
*
            GENERATE    5.5,2           телевизоры поступают один
                                        за другим
RETEST      QUEUE       LASTTEST        создание очереди LASTTEST
            ENTER       TESTERS         начало контроля
            DEPART      LASTTEST        выход из очереди LASTTEST
            ADVANCE     10,3            время контроля
            LEAVE       TESTERS         выход с поста контроля
            TRANSFER    .120,,ADJUSTIT  12% уходит на регулировку
            TERMINATE   1               уход на упаковку
*
ADJUSTIT    QUEUE       ADJUSTQ         создание очереди ADJUSTQ
            SEIZE       ADJUSTER        переход на регулировку
            DEPART      ADJUSTQ         выход из очереди ADJUSTQ
            ADVANCE     30,10           время регулировки
            RELEASE     ADJUSTER        уход с регулировки
*
NEW1        QUEUE       LASTTEST        создание очереди LASTTEST
NEW2        ENTER       TESTERS         занятие контролера
NEW3        DEPART      LASTTEST        выход из очереди LASTTEST
NEW4        ADVANCE     10,3            время контроля
NEW5        LEAVE       TESTERS         уход с поста контроля
*
NEW6        TRANSFER    .010,,ADJUSTIT  регулировка 1%
*
NEW7        TERMINATE   1               уход на упаковку
*
START       100                         CC=100, начало движения
                                        Хакт
END                                     окончание процесса ИМ
```

22. Для решения этой задачи применяется МФ, состоящий из 35 ОБ, причем поток деталей можно разделить сразу, используя ОБ TRANSFER 0.300,,APATH, либо после того, как все детали пройдут станцию 1, производить разделение на два отдельных потока. Хотя оба подхода одинаково корректны, результаты могут несколько отличаться за счет выбираемых БСВ.

A. Процесс ИМ окончится в 1117.5, среднее число рабочих на станциях с первой по четвертую равно 1.837, 0.705, 3.111, 2.295.

B. Процесс ИМ окончится в 1213.5, что несколько больше, чем в п. А. В табл. П.5.11 приведены данные расчета.

Таблица П.5.11

№ станции	Число ожидающих рабочих	
	среднее	максимальное
1	1.907	8
2	3.564	8
3	0.010	1
4	0.105	2

23. Номера блоков на месте метки взяты из распечатки итогового отчета и соответствуют строкам эха МФ. Никакого логического смысла для исполнения МФ они не имеют, поэтому набраны курсивом.

```
                SIMULATE                временная дискрета: 1 мин
*
                GENERATE  30,20         поступление заготовок
                                        с приоритетом 0
                ADVANCE   0             исключение искажения
                                        времени поступления
*
  BLOCK3        SEIZE     STATION1      занятие РС1
  BLOCK4        SEIZE     AGV           занятие кара
                ADVANCE   0.5,0.25      вызов кара
                ADVANCE   0.5           транспортирование до РС1
                RELEASE   AGV           освобождение кара
                ADVANCE   25,10         время работы на РС1
*
  BLOCK9        TRANSFER  BOTH,,WAIT    попытка занять РС2
                                        или место хранения
*
                SEIZE     STATION2      занятие РС2
                SEIZE     AGV           занятие кара
                ADVANCE   0.5,0.25      вызов кара на РС1
BLOCK13         RELEASE   STATION1      уход с РС1
```

```
REST      ADVANCE   0.5          переезд на PC2
          RELEASE   AGV          освобождение кара
BLOCK16   PRIORITY  5            повышение приоритета при
                                 уходе с PC2
          ADVANCE   30,10        время работы на PC2
          SEIZE     AGV          вызов кара
          ADVANCE   0.5,0.25     прибытие кара на PC2
          RELEASE   STATION2     освобождение PC2
          ADVANCE   0.5          перевоз детали на склад
          RELEASE   AGV          освобождение кара
          TERMINATE 1            уход детали из системы
*
WAIT      SEIZE     THESPACE     занятие места ожидания
          SEIZE     AGV          вызов кара
          ADVANCE   0.5,0.25     прибытие кара к PC1
BLOCK27   RELEASE   STATION1     освобождение PC1
          ADVANCE   0.5          транспортировка
                                 к месту ожидания
          RELEASE   AGV          освобождение кара
*
          SEIZE     STATION2     занятие PC2
          SEIZE     AGV          занятие кара
          ADVANCE   0.5,0.25     подъезд кара к месту
                                 ожидания
BLOCK33   RELEASE   THESPACE     освобождение места ожидания
          TRANSFER  ,REST        освобождение PC2
          START     100          обработка 100 деталей
          END                    окончание процесса ИМ
```

Итоговые значения сведены в табл. П.5.12.

Таблица П.5.12 *(ответы на вопросы A, D)*

Наименование	Загрузка	Число входов	Среднее время
PC1	0.991	102	31.561
PC2	0.985	101	31.68
Место ожидания	0.828	98	27.45
Кар	0.123	400	0.998

B. Заготовка не может занять кар до тех пор, пока не будет определено место, к которому надо отвезти деталь (PC1, PC2, место ожидания). При ИМ может создаться ситуация, когда детали, находящейся на каре, надо попасть на PC1, в то же самое время детали, прошедшей обработку на PC1, требуется кар. Для решения этой проблемы в МФ вводится линия раздела, такой линией раздела является *BLOCK3*.

C. Вопрос немедленного перемещения детали с PC1 на PC2 решается введением ОБ TRANSFER вида BOTH, деталь сразу пытается перейти на PC2 и только при ее занятости попадает на место ожидания.

Пункт 7.6

24. A. При инициализации генераторов с блоков 1 и 3 выйдут Хакт с ИН1 и ИН2.

B. Оба транзакта имеют время больше нуля и, следовательно, располагаются в СБС.

C. Поскольку у Хакт 2 время меньше, он располагается в начале СБС.

25. В обоих случаях 1-го и 2-го вопросов появляется сообщение, что прежнее условие запрета на 75 дискрет MB заменено на новый запрет с таким же временем.

A. Обе команды приводят к началу этапа сканирования. Разница в том, что команда "t scan" каждый раз приводит к началу этапа сканирования, в отличие от нее команда "trap clock " приводит к началу сканирования только тогда, когда время достигнет или превысит значение времени запрета.

B. Команда "trap scan" действует до тех пор, пока она не будет отменена командой "untrap scan", команда "trap clock" действует, пока не выполнится условие запрета по времени (равно или больше) либо не поступит команда "untrap clock".

C. Обычно считается, что СТС пуст в момент MB 0.0. Команда "trap scan" в этот момент уже может действовать, а команда "trap clock" не может, так как она действует, когда транзакты уже находятся с СТС.

26. Рассмотрим последовательно события, происходящие в модели.

A. Поскольку времена детерминированы, то Хакт 2 в момент 40 не может попасть на обслуживание, так как устройство занято обслуживанием Хакт 1. Для того чтобы выяснить, где и почему задержан Хакт 2, вводится системный запрет, в результате которого выясняется, что Хакт 2 уникально блокирован, а следовательно, находится в СТС, показатель ФИС, который в окне состояния обозначен S/C, находится в положении «выключено».

B. Команда отображения информации об устройстве показывает, что устройство занято обслуживанием Хакт 1, ФИС выключен.

C. Команда отображения СТС показывает, что ИС находится в положении «включено», для этого смотрим на колонку SDPGFT, каждая буква несет информацию о соответствующем транзакте (DPGFT в пособии не рассматриваются) и буква S указывает на положение ИС. Когда соответствующее место в строке транзакта не занято, это означает, что данная опция выключена. В нашем случае наличие буквы S говорит о включенном ИС. ФИС по-прежнему выключен.

D. Продолжаем движение к следующему сканированию, когда Хакт 1 исполнит ОБ RELEASE и вернется из СБС в СТС в момент 55 мин (что совпадает с нашими предварительными расчетами: 15 + 10 + 30 = 55).

E. Хакт 2 проснулся и расположится в СТС выше Хакт 1.

F. Для продвижения Хакт 1 в блок 6 используем шаговый режим, исполнение блока 5 (ОБ RELEASE) переключает ФИС в положение «включено» — это первое изменение состояния модели.

G. Состояние Хакт 2 изменилось (пропала буква S), он проснулся.

H. Следующий шаг выводит Хакт 1 из системы.

J. Все внимание сосредоточено на Хакт 2, командой "r" ФИС выключается, перезапуская сканирование, после следующей команды "r" происходит новое переключение ФИС и Хакт 2 поступает на обслуживание; так как введен системный запрет, то появляется сообщение об обслуживании Хаст 2.

27. A. Наибольшее число транзактов, ожидающих обслуживания, равно 4.

B. Интервалы времени, в которые достигается максимум ожидающих транзактов, равны: 4563.0; 4621.8; 4684.0; 4717.8; 4760.1; 4819.5; 4901.3; 5096.9.

28. A. Ответ на этот вопрос получается при установлении точки прерывания на ОБ 5 ADVANCE для механиков 1-го типа и последовательности команд "r". Данные приведены в табл. П.5.13.

Таблица П.5.13. *Появление первых пяти механиков*

ИН Хакт типа 1	1	2	3	4	5
Время появления	254.6	847.2	1477.7	2163.1	2773.6

B. Условия прерывания наложены на оба ОБ ADVANCE. Результаты сведены в табл. П.5.14.

Таблица П.5.14. *Появление пяти механиков любого типа*

Тип механиков	1	2	2	1	2
Время появления	254.6	501.3	617.4	847.2	1073.9

C. Условия прерывания наложены на оба ОБ RELEASE. Результаты сведены в табл. П.5.15.

Таблица П.5.15. *Время окончания обслуживания пяти механиков*

Тип механиков	1	2	2	1	2
Время обслуживания	501.3	617.4	739.0	1073.9	1164.0

D. Условия прерывания наложены на оба ОБ RELEASE, и при первой передаче управления подаются команды "d blo — d cec". Обслуживается

механик 2-го типа, а ожидает обслуживания механик 1-го типа в 501.3 дискрет МВ.

E. В момент МВ 1073.9 механик 1-го типа освободит кладовщика, в этот момент времени ожидают обслуживания два механика 2-го типа.

F. В момент МВ 3067.4 механик 1-го типа освободит кладовщика, в этот момент времени ожидают обслуживания по одному механику каждого типа.

G. В момент МВ 5599.0 механик 2-го типа встанет в очереди перед механиком 1-го типа.

Пункт 8.5

29. A. $T_A = (420 - 360) + (0.270295)(360) = 254.1$.

B. $T_A = (420 - 360) + (0.4993530(360) = 419.5$.

C. Появление первого и второго телевизора приблизительно равно 4.58, 10.08.

30. Постройте модельный файл, используя начальный МФ примера 6.3 и как подсказку МФ примеров 8.4–8.6. Данные моделирования и расчета, округленные до первого десятичного знака, сведены в табл. П.5.16.

Таблица П.5.16. *Данные упражнения*

№	Время пребывания		Время ожидания	
	Тип А	Тип В	Тип А	Тип В
1	10.7	3.3	2.1	0.5
2	11.7	5.0	3.1	2.0
3	11.3	4.6	2.3	1.7
4	9.0	3.9	0.4	1.1
5	12.9	3.8	4.3	0.9
6	10.4	4.5	1.3	1.6
7	9.4	4.0	0.6	1.1
8	9.3	3.8	0.6	0.9
9	10.7	3.4	1.8	0.5
10	9.7	5.7	0.7	2.9
11	13.2	3.9	4.8	1.0
12	9.7	4.8	1.1	1.9
13	9.6	5.4	0.6	2.5
14	11.6	6.8	2.7	3.9
15	9.7	4.1	0.8	1.2
16	12.8	4.4	3.9	1.6
17	9.1	3.4	0.5	0.5
18	13.6	3.9	4.8	1.0
19	10.0	3.7	1.1	0.7
20	9.9	3.8	1.2	0.9
Среднее	10.7	4.31	1.94	1.42
Отклонение	1.46	0.888	1.50	0.879
95 %	[10.1, 11.3]	[4.0, 4.7]	[1.2, 2.5]	[1.1, 1.8]

31. При прочих равных условиях более значимый доверительный интервал всегда шире менее значимого, так, для табл. 8.20:

95 % [2.62, 4.45]

90 % [2.79, 4.28]

80 % [2.97, 4.10].

32. Данные по ДО табл. 8.20 не являются абсолютно независимыми, так как БСВ получаются с одного ГСЧ, а по умолчанию начальная позиция ГСЧ всегда одинакова, и поэтому, пока не вмешиваются значения времени обслуживания, результаты не являются независимыми. Сделать потоки независимыми можно, беря БСВ строго с разных ГСЧ или переназначая стартовые позиции в каждой реплике (см. примеры 8.4–8.6).

33а. A. Ответ на первый вопрос содержится в упражнении 32.

B. Можно использовать 3 ГСЧ, потому что независимый порядок, в котором механики приходят в кладовую, не влияет на очередность их обслуживания

C. Стартовая позиция ГСЧ должна меняться в начале каждой независимой реплики, а не только при изменении ДО.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Вагнер Г.* Основы исследования операций. М.: Мир, 1973.

2. *Варжапетян А. Г.* Техническая эффективность судовых систем управления. Л.: Судостроение, 1969.

3. *Варжапетян А. Г., Коршунов Г. И.* Обеспечение качества технических средств автоматизации. Л.: Машиностроение, 1984.

4. *Варжапетян А. Г., Глущенко В. В.* Системы управления. Исследование и компьютерное проектирование. М.: Вуз. книга, 2000.

5. *Геловани В. А., Юрченко В. В.* Проблемы компьютерного моделирования / МНИИ проблем управления. М., 1990.

6. *Коробейников В. П.* Принципы математического моделирования. Владивосток: Дальнаука, 1996.

7. *Леонов Г. Н.* Введение в математическое моделирование. Барнаул, 1998.

8. *Максимей И. В.* Имитационное моделирование на ЭВМ. М.: Радио и связь, 1988.

9. *Павловский Ю. Н.* Имитационное моделирование и системы / ВЦ РАН. М., 2000.

10. *Советов Б. Я., Яковлев С. А.* Моделирование систем. М.: Высш. шк., 1998.

11. *Харин Ю. С.* и др. Основы имитационного и статистического моделирования. Минск: ДизайнПро, 1997.

12. *Banks J., Carson J., Ngo Sy.* Getting Started with GPSS / Wolverine SW Corp., 1995.

13. *Henriksen J., Crain G.* GPSS / Reference Manual. Wolverine SW Corp., 1996.

14. *Henriksen J.* An Introduction to SLX: Proc. of the * Winter Simulation Conference // IEEE. 1999–2001.

15. *Karian Z., Dudewicz E.* Modern Statistical, Systems and GPSS Simulation. CRC Press, 1999.

16. *Law A. M., Kelton W. D.* Simulation Modeling and Analysis. NY.: McGraw-Hill, 1990.

17. *Schriber T.* Introduction to Simulation Using GPSS. NY.: John Willey & Sons, 1991.

Учебное издание

Варжапетян Артемий Георгиевич

ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ НА GPSS/H

Учебное пособие

Редактор *А. Г. Ларионова*
Верстальщик *Т. М. Каргапольцева*

Сдано в набор 02.11.06. Подписано к печати 06.03.07.
Формат 60×84 1/16. Бумага офсетная. Печать офсетная. Усл. печ. л. 24,1.
Уч.-изд. л. 25,0. Тираж 100 экз. Заказ № 120.

Редакционно-издательский центр ГУАП
190000, Санкт-Петербург, Б. Морская ул., 67