ОГЛАВЛЕНИЕ ВВЕДЕНИЕ

ВВЕДЕНИЕ

Процессы функционирования различных систем и сетей связи могут быть представлены той или иной совокупностью систем массового обслуживания (СМО) - стохастических, динамических, дискретно-непрерывных математических моделей. Исследование характеристик таких моделей может проводиться либо аналитическими методами, либо путем имитационного моделирования [1-6].

Имитационная модель отображает стохастический процесс смены дискретных состояний СМО в непрерывном времени в форме моделирующего алгоритма. При его реализации на ЭВМ производится накопление статистических данных по тем атрибутам модели, характеристики которых являются предметом исследований. По окончании моделирования накопленная статистика обрабатывается, и результаты моделирования получаются в виде выборочных распределений исследуемых величин или их выборочных моментов. Таким образом, при имитационном моделировании систем массового обслуживания речь всегда идет о статистическом имитационном моделировании [5;6].

Сложные функции моделирующего алгоритма могут быть реализованы средствами универсальных языков программирования (Паскаль, Си), что предоставляет неограниченные возможности в разработке, отладке и использовании модели. Однако подобная гибкость приобретается ценой больших усилий, затрачиваемых на разработку и программирование весьма сложных моделирующих алгоритмов, оперирующих со списковыми структурами данных. Альтернативой этому является использование специализированных языков имитационного моделирования [5-7].

Специализированные языки имеют средства описания структуры и процесса функционирования моделируемой системы, что значительно облегчает и упрощает программирование имитационных моделей, поскольку основные функциии моделирующего алгоритма при этом реализуются автоматически. Программы имитационных моделей на специализированных языках моделирования близки к описаниям моделируемых систем на естественном языке, что позволяет конструировать сложные имитационные модели пользователям, не являющимся профессиональными программистами.

Одним из наиболее эффективных и распространенных языков моделирования сложных дискретных систем является в настоящее время язык

GPSS [1;4;7]. Он может быть с наибольшим успехом использован для моделирования систем, формализуемых в виде систем массового обслуживания. В качестве объектов языка используются аналоги таких стандартных компонентов СМО, как заявки, обслуживающие приборы, очереди и т.п. Достаточный набор подобных компонентов позволяет конструировать сложные имитационные модели, сохраняя привычную терминологию СМО.

На персональных компьютерах (ПК) типа IBM/PC язык GPSS реализован в рамках пакета прикладных программ GPSS/PC [8]. Основной модуль пакета представляет собой интегрированную среду, включающую помимо транслятора со входного языка средства ввода и редактирования текста модели, ее отладки и наблюдения за процессом моделирования, графические средства отображения атрибутов модели, а также средства накопления результатов моделирования в базе данных и их статистической обработки. Кроме основного модуля в состав пакета входит модуль создания стандартного отчета GPSS/PC, а также ряд дополнительных модулей и файлов.

В данном издании, состоящем из двух частей, излагаются основы моделирования систем и сетей связи с использованием пакета GPSS/PC. В первой части рассматриваются основные понятия и средства GPSS/PC, приемы конструирования GPSS-моделей и технология работы с пакетом. Изложение материала сопровождается небольшими учебными примерами. Относительно подробное рассмотрение языка GPSS/PC вызвано отсутствием в литературе учебного материала по данной версии языка.

Во второй части рассматриваются примеры GPSS-моделей различных систем и сетей массового обслуживания, используемых для формализации процессов функционирования систем и сетей связи. Приводится также ряд примеров моделирования систем и сетей связи с использованием GPSS/PC. Подробно комментируются тексты GPSS-моделей и результаты моделирования.

1. ОБЩИЕ СВЕДЕНИЯ О GPSS/PC

Исходная программа на языке GPSS/PC, как и программа на любом языке программирования, представляет собой последовательность операторов. Операторы GPSS/PC записываются и вводятся в ПК в следующем формате:

номер_строки имя операция операнды ; комментарии

Все операторы исходной программы должны начинаться с номера 0_строки - целого положительного числа от 1 до 9999999. После ввода операторов они располагаются в исходной программе в соответствии с нумерацией строк. Обычно нумерация производится с некоторым шагом, отличным от 1, чтобы иметь возможность добавления операторов в нужное место исходной программы.Некоторые операторы удобно вводить, не включая их в исходную программу. Такие операторы вводятся без номера строки. В настоящем издании при описании формата операторов и в примерах моделей номера строк будут опускаться для лучшей читаемости текста. Отдельные операторы могут иметь имя для ссылки на эти операторы в других операторах. Если такие ссылки отсутствуют, то этот элемент оператора не является обязательным. В поле операции записывается ключевое слово (название оператора), указывающее конкретную функцию, выполняемую данным оператором.

Это поле оператора является обязательным. У некоторых операторов поле операции включает в себя также вспомогательный операнд.

В полях операндов записывается информация, уточняющая и конкретизирующая выполнение функции, определенной в поле операции. Эти поля в зависимости от типа операции содержат до семи операндов, расположенных в определенной последовательности и обозначаемых обычно первыми буквами латинского алфавита от A до G. Некоторые операторы вообще не имеют операндов, а в некоторых операнды могут быть опущены, при этом устанавливаются их стандартные значения (по умолчанию). При записи операндов используется позиционный принцип: пропуск операнда отмечается запятой.

Необязательные комментарии в случае их присутствия отделяются от поля операндов точкой с запятой. Комментарии не могут содержать букв русского алфавита. Операторы GPSS/PC записываются, начиная с первой позиции, в свободном формате, т.е. отдельные поля разделяются произвольным количеством пробелов. При вводе исходной программы в интегрированной среде GPSS/PC размещение отдельных полей операторов с определенным количеством интервалов между ними производится автоматически.

Каждый оператор GPSS/PC относится к одному из четырех типов: операторы-блоки, операторы определения объектов, управляющие операторы и операторы-команды.

Операторы-блоки формируют логику модели. В GPSS/PC имеется около 50 различных видов блоков, каждый из которых выполняет свою конкретную функцию. За каждым из таких блоков стоит соответствующая подпрограмма транслятора, а операнды каждого блока служат параметрами этой подпрограммы.

Операторы определения объектов служат для описания параметров некоторых объектов GPSS/PC (о самих объектах речь пойдет дальше). Примерами параметров объектов могут быть количество каналов в многоканальной системе массового обслуживания, количество строк и столбцов матрицы и т.п.

Управляющие операторы служат для управления процессом моделирования (прогоном модели). Операторы-команды позволяют управлять работой интегрированной среды GPSS/PC. Управляющие операторы и операторы-команды обычно не включаются в исходную программу, а вводятся непосредственно с клавиатуры ПК в процессе интерактивного взаимодействия с интегрированной средой. После трансляции исходной программы в памяти ПК создается так называемая текущая модель, являющаяся совокупностью разного типа объектов, каждый из которых представляет собой некоторый набор чисел в памяти ПК, описывающих свойства и текущее состояние объекта.

Объекты GPSS/PC можно разделить на семь классов: динамические, операционные, аппаратные, статистические, вычислительные, запоминающие и группирующие.

Динамические объекты, соответствующие заявкам в системах массового обслуживания, называются в GPSS/PC транзактами. Они "создаются" и "уничтожаются" так, как это необходимо по логике модели в процессе моделирования. С каждым транзактом может быть связано произвольное число параметров, несущих в себе необходимую информацию об этом транзакте. Кроме того, транзакты могут иметь различные приоритеты.

Операционные объекты GPSS/PC, называемые блоками, соответствуют операторам-блокам исходной программы. Они, как уже говорилось, формируют логику модели, давая транзактам указания: куда идти и что делать дальше. Модель системы на GPSS/PC можно представить совокупностью блоков, объединенных в соответствии с логикой работы реальной системы в так называемую блок-схему. Блок-схема модели может быть изображена графически, наглядно показывая взаимодействие блоков в процессе моделирования.

Аппаратные объекты GPSS/PC - это абстрактные элементы, на которые может быть расчленено (декомпозировано) оборудование реальной системы. К ним относятся одноканальные и многоканальные устройства и логические переключатели. Многоканальное устройство иногда называют памятью.

Одноканальные и многоканальные устройства соответствуют обслуживающим приборам в СМО. Одноканальное устройство, которое для краткости далее будем называть просто устройством, может обслуживать одновременно только один транзакт. Многоканальное устройство (МКУ) может обслуживать одновременно несколько транзактов. Логические переключатели (ЛП) используются для моделирования двоичных состояний логического или физического характера. ЛП может нахо-

диться в двух состояниях: включено и выключено. Его состояние может изменяться в процессе моделирования, а также опрашиваться для принятия определенных решений.

Статистические объекты GPSS/PC служат для сбора и обработки статистических данных о функционировании модели. К ним относятся очереди и таблицы .

Каждая очередь обеспечивает сбор и обработку данных о транзактах, задержанных в какой-либо точке модели, например перед одноканальным устройством. Таблицы используются для получения выборочных распределений некоторых случайных величин, например времени пребывания транзакта в модели.

К вычислительным объектам GPSS/PC относятся переменные (арифметические и булевские) и функции. Они используются для вычисления некоторых величин, заданных арифметическими или логическими выражениями либо табличными зависимостями.

Запоминающие объекты GPSS/PC обеспечивают хранение в памяти ПК отдельных величин, используемых в модели, а также массивов таких величин. К ним относятся так называемые сохраняемые величины и матрицы сохраняемых величин.

К объектам группирующего класса относятся списки пользователя и группы. Списки пользователя используются для организации очередей с дисциплинами, отличными от дисциплины "раньше пришел – раньше обслужен". Группы в данном издании рассматриваться не будут.

Каждому объекту того или иного класса соответствуют числовые атрибуты, описывающие его состояние в данный момент модельного времени. Кроме того, имеется ряд так называемых системных атрибутов, относящихся не к отдельным объектам, а к модели в целом.
Значения атрибутов всех объектов модели по окончании моделирования Выводятся в стандартный отчет GPSS/PC. Большая часть атрибутов доступна программисту и составляет так называемые стандартные числовые атрибуты (СЧА), 0которые могут использоваться в качестве операндов операторов исходной программы. Все СЧА в GPSS/PC являются целыми числами.

Каждый объект GPSS/PC имеет имя и номер. Имена объектам даются в различных операторах исходной программы, а соответствующие им номера транслятор присваивает автоматически. Имя объекта представляет собой начинающуюся с буквы последовательность букв латинского алфавита, цифр и символа "подчеркивание". При необходимости имени любого объекта, кроме имени блока, можно поставить в соответствие любой номер с помощью оператора описания EQU, имеющего следующий формат:

имя EQU номер

Блокам присваиваются их порядковые номера в исходной программе (непутать с номерами строк!).

Для ссылки 0на какой-либо стандартный числовой атрибут некоторого объекта соответствующий операнд оператора исходной программы записывается одним из следующих способов:

СЧА $имя ;

СЧА j ,

где СЧА - системное обозначение (название) конкретного стандартного числового атрибута данного объекта; имя - имя объекта; j - номер объекта; $ - символ-разделитель.

Прогон текущей модели, т.е. собственно моделирование, выполняется с помощью специальной управляющей программы, которую называют симулятором (от английского **SIMULATE** - моделировать, имитировать). Работа GPSS-модели под управлением симулятора заключается в перемещении транзактов от одних блоков к другим, аналогично тому, как в моделируемой СМО перемещаются заявки, соответствующие транзактам.

В начальный момент времени в GPSS-модели нет ни одного транзакта. В процессе моделирования симулятор генерирует транзакты в

определенные моменты времени в соответствии с теми логическими потребностями, которые возникают в моделируемой системе. Подобным же образом транзакты покидают модель в определенные моменты времени в зависимости от специфики моделируемой системы. В общем случае в модели одновременно существует большое число транзактов, однако в каждый момент времени симулятор осуществляет продвижение только какого-либо одного транзакта.

Если транзакт начал свое движение, он перемещается от блока к блоку по пути, предписанному блок-схемой. В тот момент, когда транзакт входит в некоторый блок, на исполнение вызывается подпрограмма симулятора, соответствующая типу этого блока, а после ее выполнения, при котором реализуется функция данного блока, транзакт "пытается" войти в следующий блок. Такое продвижение транзакта продолжается до тех пор, пока не произойдет одно из следующих возможных событий:

1) транзакт входит в блок, функцией которого является удаление транзакта из модели;
2) транзакт входит в блок, функцией которого является задержка транзакта на некоторое определенное в модели время;
3) транзакт "пытается" войти в следующий блок, однако блок "отказывается" принять его. В этом случае транзакт остается в том блоке, где находился, и позднее будет повторять свою попытку войти в следующий блок. Когда условия в модели изменятся, такая попытка может оказаться успешной, и транзакт сможет продолжить свое перемещение по блок-схеме.

Если возникло одно из описанных выше условий, обработка данного транзакта прекращается, и начинается перемещение другого транзакта.

Таким образом, выполнение моделирования симулятором продолжается постоянно.

Проходя через блоки модели, каждый транзакт вносит вклад в содержимое счетчиков блоков. Значения этих счетчиков доступны программисту через СЧА блоков: **W** - текущее содержимое блока и **N** - общее количество входов в блок.

Каждое продвижение транзакта в модели является событием, которое должно произойти в определенный момент модельного времени. Для того, чтобы поддерживать правильную временную последовательность событий, симулятор имеет таймер модельного времени, который автоматически корректируется в соответствии с логикой, предписанной моделью.

Таймер GPSS/PC имеет следующие особенности:

1) регистрируются только целые значения (все временные интервалы в модели изображаются целыми числами);
2) единица модельного времени определяется разработчиком модели, который задает все временные интервалы в одних и тех же, выбранных им единицах;
3) симулятор не анализирует состояние модели в каждый следующий момент модельного времени (отстоящий от текущего на единицу модельного времени), а продвигает таймер к моменту времени, когда происходит ближайшее следующее событие.

Значения таймера доступны программисту через системные СЧА C1 (относительное время) и AC1 (абсолютное время).

Центральной задачей, выполняемой симулятором, является определение того, какой транзакт надо выбрать следующим для продвижения в модели, когда его предшественник прекратил свое продвижение. С этой целью симулятор рассматривает каждый транзакт как элемент некоторого списка. В относительно простых моделях используются лишь два основных списка: список текущих событий и список будущих событий.

Список текущих событий включает в себя те транзакты, планируемое время продвижения которых равно или меньше текущего модельного времени (к последним относятся транзакты, движение которых было заблокировано ранее). Он организуется в порядке убывания приоритетов транзактов, а в пределах каждого уровня приоритета - в порядке поступления транзактов.

Список будущих событий включает в себя транзакты, планируемое время продвижения которых больше текущего времени, т.е. события, связанные с продвижением этих транзактов, должны произойти в будущем. Этот список организуется в порядке возрастания планируемого времени продвижения транзактов.

Симулятор GPSS/PC помещает транзакты в зависимости от условий в модели в тот или иной список и переносит транзакты из списка в список, просматривает списки, выбирая следующий транзакт для обработки, корректирует таймер модельного времени после обработки всех транзактов в списке текущих событий.

2. ОСНОВНЫЕ БЛОКИ GPSS/PC И СВЯЗАННЫЕ С НИМИ ОБЪЕКТЫ

2.1. Блоки, связанные с транзактами

С транзактами связаны блоки создания, уничтожения, задержки транзактов, изменения их атрибутов и создания копий транзактов.

Для создания транзактов, входящих в модель, служит блок GENERATE (генерировать), имеющий следующий формат:

имя GENERATE A,B,C,D,E

В **поле A** задается среднее значение интервала времени между моментами поступления в модель двух последовательных транзактов. Если этот интервал постоянен, то **поле B** не используется. Если же интервал поступления является случайной величиной, то в **поле B** указывается модификатор среднего значения, который может быть задан в виде модификатора-интервала или модификатора-функции.

Модификатор-интервал используется, когда интервал поступления транзактов является случайной величиной с равномерным законом распределения вероятностей. В этом случае в **поле B** может быть задан любой СЧА, кроме ссылки на функцию, а диапазон изменения интервала поступления имеет границы **A-B**, **A+B**.

Например, блок

GENERATE 100,40

создает транзакты через случайные интервалы времени, равномерно распределенные на отрезке [60;140].

Модификатор-функция используется, если закон распределения интервала поступления отличен от равномерного. В этом случае в **поле B** должна быть записана ссылка на функцию (ее СЧА), описывающую этот закон, и случайный интервал поступления определяется, как целая часть произведения **поля A** (среднего значения) на вычисленное значение функции.

В **поле C** задается момент поступления в модель первого транзакта. Если это поле пусто или равно 0, то момент появления первого транзакта определяется операндами A и B.

Поле D задает общее число транзактов, которое должно быть создано блоком **GENERATE**. Если это поле пусто, то блок генерирует неограниченное число транзактов до завершения моделирования.

В **поле E** задается приоритет, присваиваемый генерируемым транзактам. Число уровней приоритетов неограничено, причем самый низкий приоритет - нулевой. Если поле E пусто, то генерируемые транзакты имеют нулевой приоритет.

Транзакты имеют ряд стандартных числовых атрибутов. Например, СЧА с названием PR позволяет ссылаться на приоритет транзакта. СЧА с названием M1 содержит так называемое резидентное время транзакта, т.е. время, прошедшее с момента входа транзакта в модель через блок **GENERATE**. СЧА с названием XN1 содержит внутренний номер транзакта, который является уникальным и позволяет всегда отличить один транзакт от другого. В отличие от СЧА других объектов, СЧА транзактов не содержат ссылки на имя или номер транзакта. Ссылка на СЧА транзакта всегда относится к активному транзакту, т.е. транзакту, обрабатываемому в данный момент симулятором.

Важными стандартными числовыми атрибутами транзактов являются значения их параметров. Любой транзакт может иметь неограниченное число параметров, содержащих те или иные числовые значения. Ссылка на этот СЧА транзактов всегда относится к активному транзакту и имеет вид Pj или P$ имя, где j и имя - номер и имя параметра соответственно. Такая ссылка возможна только в том случае, если параметр с указанным номером или именем существует, т.е. в него занесено какое-либо значение.

Для присваивания параметрам начальных значений или изменения этих значений служит **блок ASSIGN** (присваивать), имеющий следущий формат:

имя ASSIGN A,B,C

В **поле A** указывается номер или имя параметра, в который заносится значение **операнда B**. Если в **поле A** после имени (номера) параметра стоит знак + или -, то значение **операнда B** добавляется или вычитается из текущего содержимого параметра. В **поле C** может быть указано имя или номер функции-модификатора, действующей аналогично функции-модификатору в поле B блока **GENERATE**.

Например, блок

ASSIGN 5,0

записывает в параметр с номером 5 значение 0, а блок

ASSIGN COUNT+,1

добавляет 1 к текущему значению параметра с именем COUNT.

Для записи текущего модельного времени в заданный параметр транзакта служит **блок MARK** (отметить), имеющий следующий формат:

имя MARK A

В **поле A** указывается номер или имя параметра транзакта, в который заносится текущее модельное время при входе этого транзакта в блок MARK. Содержимое этого параметра может быть позднее использовано для определения транзитного времени 0пребывания транзакта в какой-то части модели с помощью СЧА с названием MP.

Например, если на входе участка модели поместить блок

MARK MARKER ,

то на выходе этого участка СЧА MP$MARKER будет содержать разность между текущим модельным временем и временем, занесенным в параметр MARKER блоком MARK.

Если **поле A** блока MARK пусто, то текущее время заносится на место отметки времени входа транзакта в модель, используемой при определении резидентного времени транзакта с помощью СЧА M1.

Для изменения приоритета транзакта служит **блок PRIORITY** (приоритет), имеющий следующий формат:

имя PRIORITY A,B

В **поле A** записывается новый приоритет транзакта. В **поле B** может содержаться ключевое слово BU, при наличии которого транзакт, вошедший в блок, помещается в списке текущих событий после всех остальных транзактов новой приоритетной группы, и список текущих событий просматривается с начала. Использование такой возможности будет рассмотрено ниже.

Для удаления транзактов из модели служит **блок TERMINATE** (завершить), имеющий следующий формат:

имя TERMINATE A

Значение **поля A** указывает, на сколько единиц уменьшается содержимое так называемого счетчика завершений при входе транзакта в данный блок TERMINATE. Если **поле A** не определено, то оно считается равным 0, и транзакты, проходящие через такой блок, не уменьшают содержимого счетчика завершений.

Начальное значение счетчика завершений устанавливается управляющим **оператором START** (начать), предназначенным для запуска прогона модели. **Поле A** этого оператора содержит начальное значение счетчика завершений (см. разд. 3). Прогон модели заканчивается, когда содержимое счетчика завершений обращается в 0. Таким образом, в модели должен быть хотя бы один блок TERMINATE с непустым **полем A**, иначе процесс моделирования никогда не завершится.

Текущее значение счетчика завершений доступно программисту через **системный СЧА TG1**.

Участок блок-схемы модели, связанный с парой блоков **GENERATE-TERMINATE**, называется **сегментом**. Простые модели могут состоять из одного сегмента, в сложных моделях может быть несколько сегментов.

Например, простейший сегмент модели, состоящий всего из двух блоков **GENERATE** и **TERMINATE** и приведенный на рис. 1, в совокупности с управлящим **оператором START** моделирует процесс создания случайного потока транзактов, поступащих в модель со средним интервалом в 100 единиц модельного времени, и уничтожения этих транзактов. Начальное значение счетчика завершений равно 1000. Каждый транзакт, проходящий через **блок TERMINATE**, вычитает из счетчика единицу, и таким образом моделирование завершится, когда тысячный по счету транзакт войдет в **блок TERMINATE**. При этом точное значение таймера в момент завершения прогона непредсказуемо. Следовательно, в приведенном примере продолжительность прогона устанавливается не по модельному времени, а по количеству транзактов, прошедших через модель.

```
GENERATE    100,40
TERMINATE   1
START       1000
```

Если необходимо управлять продолжительностью прогона по модельному времени, то в модели используется специальный сегмент, называемый сегментом таймера.

```
GENERATE    100,40
TERMINATE
GENERATE    100000
TERMINATE   1
START       1
```

Например, в модели из двух сегментов, приведенной на рис. 2, первый (основной) сегмент выполняет те же функции, что и в предыдущем примере. Заметим, однако, что **поле A блока TERMINATE** в первом сегменте пусто, т.е. уничтожаемые транзакты не уменьшают содержимого счетчика завершений. Во втором сегменте **блок GENERATE** создаст первый транзакт в момент модельного времени, равный 100000. Но этот транзакт окажется и последним в данном сегменте, так как, войдя в **блок TERMINATE**, он обратит в 0 содержимое счетчика завершений, установленное **оператором START** равным 1. Таким образом, в этой модели гарантируется завершение прогона в определенный момент модельного времени, а точное количество транзактов, прошедших через модель, непредсказуемо.

В приведенных примерах транзакты, входящие в модель через блок **GENERATE**, в тот же момент модельного времени уничтожались в **блоке TERMINATE**. В моделях систем массового обслуживания заявки обслуживаются приборами (каналами) СМО в течение некоторого промежутка времени прежде, чем покинуть СМО. Для моделирования такого обслуживания, т.е. для задержки транзактов на определенный отрезок модельного времени, служит **блок ADVANCE** (задержать), имеющий следующий формат:

```
имя     ADVANCE     A,B
```

Операнды в **полях A** и **B** имеют тот же смысл, что и в соот-

ветствующих полях блока GENERATE. Следует отметить, что транзакты, входящие в блок **ADVANCE**, переводятся из списка текущих событий в список будущих событий, а по истечении вычисленного времени задержки возвращаются назад, в список текущих событий, и их продвижение по блок-схеме продолжается. Если вычисленное время задержки равно 0, то транзакт в тот же момент модельного времени переходит в следующий блок, оставаясь в списке текущих событий.

Например, в сегменте, приведенном на рис. 3, транзакты, поступающие в модель из б**лока GENERATE** через случайные интервалы времени, имеющие равномерное распределение на отрезке [60;140], попадают в **блок ADVANCE**. Здесь определяется случайное время задержки транзакта, имеющее равномерное распределение на отрезке [30;130], и транзакт переводится в список будущих событий. По истечении времени задержки транзакт возвращается в список текущих событий и входит в блок **TERMINATE**, где уничтожается. Заметим, что в списке будущих событий, а значит и в **блоке ADVANCE** может одновременно находиться произвольное количество транзактов.

```
GENERATE    100,40
ADVANCE     80,50
TERMINATE   1
```

В рассмотренных выше примерах случайные интервалы времени подчинялись равномерному закону распределения вероятностей. Для получения случайных величин с другими распределениями в GPSS/PC используются вычислительные объекты: переменные и функции.

Как известно, **произвольная случайная величина** связана со случайной **величиной R**, имеющей равномерное распределение на отрезке [0;1], через свою обратную функцию распределения. Для некоторых случайных величин уравнение связи имеет явное решение, и **значение случайной величины с заданным распределением вероятностей** может быть вычислено через **R** по формуле. Так, например, значение случайной **величины E** с показательным (экспоненциальным) распределением с **параметром d** вычисляется по формуле:

E= -(1/d) * ln(R)

Напомним, что параметр d имеет смысл величины, обратной математическому ожиданию E, а, следовательно, 1/d - математическое ожидание (среднее значение) случайной величины E.

Для получения случайной величины R с равномерным распределением на отрезке [0;1] в GPSS/PC имеются встроенные генераторы случайных чисел. Для получения случайного числа путем обращения к такому генератору достаточно записать системный СЧА RN с номером генератора, например RN1. Правда, встроенные генераторы случайных чисел GPSS/PC дают числа не на отрезке [0;1], а целые случайные числа, равномерно распределенные от 0 до 999, но их нетрудно привести к указанному отрезку делением на 1000.

Проще всего описанные вычисления в GPSS/PC выполняются с использованием арифметических переменных. Они могут быть целыми и действительными. Целые переменные определяются перед началом моделирования с помощью оператора определения **VARIABLE** (переменная), имеющего следующий формат:

имя VARIABLE выражение

Здесь имя - имя переменной, используемое для ссылок на нее, а выражение - арифметическое выражение, определяющее переменную. Арифметическое выражение представляет собой комбинацию операндов, в качестве которых могут выступать константы, СЧА и функции, знаков арифметических операций и круглых скобок. Следует заметить, что знаком операции умножения в GPSS/PC является символ # (номер). Результат каждой промежуточной операции в целых переменных преобразуется к целому типу путем отбрасывания дробной части, и, таким образом, результатом операции деления является целая часть частного.

Действительные переменные 0определяются перед началом модели-

рования с помощью оператора определения **FVARIABLE**, имеющего тот же формат, что и **оператор VARIABLE**. Отличие действительных переменных от целых заключается в том, что в действительных переменных все промежуточные операции выполняются с сохранением дробной части чисел, и лишь окончательный результат приводится к целому типу отбрасыванием дробной части.

Арифметические переменные обоих типов имеют единственный СЧА с названием **V**, значением которого является результат вычисления арифметического выражения, определяющего переменную. Вычисление выражения производится при входе транзакта в блок, содержащий ссылку на СЧА V с именем переменной.

Действительные переменные могут быть использованы для получения случайных интервалов времени с показательным законом распределения. Пусть в модели из примера на рис. 3 распределения времени поступления транзактов и времени задержки должны иметь показательный закон. Это может быть сделано так, как показано на рис. 4.

```
TARR    FVARIABLE     -100#LOG((1+RN1)/1000)
TSRV    FVARIABLE     -80#LOG((1+RN1)/1000)
        GENERATE      V$TARR
        ADVANCE       V$TSRV
        TERMINATE     1
```

Рис. 4

Переменная с именем **TARR** задает выражение для вычисления интервала поступления со средним значением 100, вторая **переменная** с именем **TSRV** - для вычисления времени задержки со средним значением 80. Блоки GENERATE и ADVANCE содержат в **поле A** ссылки на соответствующие переменные, при этом **поле B** не используется, так как в поле A содержится случайная величина, не нуждающаяся в модификации.

Большинство случайных величин не может быть получено через случайную величину R с помощью арифметического выражения. Кроме того, такой способ является достаточно трудоемким, так как требует обращения к математическим функциям, вычисление которых требует десятков машинных операций. Другим возможным способом является использование вычислительных объектов GPSS/PC типа функция.

Функции используются для вычисления величин, заданных табличными зависимостями. Каждая функция определяется перед началом моделирования с помощью **оператора определения FUNCTION** (функция), имеющего следующий формат:

```
имя     FUNCTION    A,B
```

Здесь имя - имя функции, используемое для ссылок на нее; **A** – стандартный числовой атрибут, являющийся аргументом функции; **B** - тип функции и число точек таблицы, определяющей функцию.

Существует пять типов функций. Рассмотрим вначале непрерывные числовые функции, тип которых кодируется буквой **C**. Так, например, в определении непрерывной числовой функции, таблица которой содержит 24 точки, поле B должно иметь значение C24.

При использовании непрерывной функции для генерирования случайных чисел ее аргументом должен быть один из генераторов случайных чисел **RNj**. Так, оператор для определения функции показательного распределения может иметь следующий вид:

```
EXP     FUNCTION    RN1,C24
```

Особенностью использования встроенных генераторов случайных чисел **RNj** в качестве аргументов функций является то, что их значения в этом контексте интерпретируются как дробные числа от 0 до 0,999999.

Таблица с координатами точек функции располагается в строках, следующих непосредственно за **оператором FUNCTION**. Эти строки не должны иметь поля нумерации. Каждая точка таблицы задается парой **Xi** (значение аргумента) и **Yi** (значение функции), отделяемых друг от друга запятой. Пары координат отделяются друг от друга символом "/" и располагаются на произвольном количестве строк. Последователь-

ность значений аргумента **Xi** должна быть строго возрастающей.

Как уже отмечалось, при использовании функции в **поле B** блоков **GENERATE** и **ADVANCE** вычисление интервала поступления или времени задержки производится путем умножения операнда A на вычисленное значение функции. Отсюда следует, что функция, используемая для генерирования случайных чисел с показательным распределением, должна описывать **зависимость y=-ln(x)**, представленную в табличном виде. **Оператор FUNCTION** с такой таблицей, содержащей 24 точки для обеспечения достаточной точности аппроксимации, имеет следующий вид:

```
 EXP    FUNCTION    RN1,C24
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915
.7,1.2/.75,1.38/.8,1.6/.84,1.85/.88,2.12/.9,2.3
.92,2.52/.94,2.81/.95,2.99/.96,3.2/.97,3.5/.98,3.9
.99,4.6/.995,5.3/.998,6.2/.999,7/.9998,8
```

Вычисление непрерывной функции производится следующим образом. Сначала определяется интервал (Xi;Xi+1), на котором находится текущее значение СЧА-аргумента (в нашем примере - сгенерированное значение RN1). Затем на этом интервале выполняется линейная интерполяция с использованием соответствующих значений Yi и Yi+1. Результат интерполяции усекается (отбрасыванием дробной части) и используется в качестве значения функции. Если функция служит операндом B блоков **GENERATE** или **ADVANCE**, то усечение результата производится только после его умножения на значение операнда A.

Использование функций для получения случайных чисел с заданным распределением дает хотя и менее точный результат за счет погрешностей **аппроксимации**, но зато с меньшими вычислительными затратами (несколько машинных операций на выполнение линейной интерполяции). Чтобы к погрешности аппроксимации не добавлять слишком большую погрешность усечения, среднее значение при использовании показательных распределений должно быть достаточно большим (не менее 50). Эта рекомендация относится и к использованию переменных.

Функции всех типов имеют единственный СЧА с названием **FN**, значением которого является вычисленное значение функции. Вычисление функции производится при входе транзакта в блок, содержащий ссылку на СЧА FN с именем функции.

Заменим в примере на рис. 4 переменные TARR и TSRV на функцию EXP (рис. 5).

Поскольку в обеих моделях используется один и тот же генератор RN1, интервалы поступления и задержки, вычисляемые в блоках GENERATE и ADVANCE, должны получиться весьма близкими, а может быть и идентичными. При большом количестве транзактов, пропускаемых через модель (десятки и сотни тысяч), разница в скорости вычислений должна стать заметной.

```
 EXP    FUNCTION    RN1,C24
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915
.7,1.2/.75,1.38/.8,1.6/.84,1.85/.88,2.12/.9,2.3
.92,2.52/.94,2.81/.95,2.99/.96,3.2/.97,3.5/.98,3.9
.99,4.6/.995,5.3/.998,6.2/.999,7/.9998,8
        GENERATE    100,FN$EXP
        ADVANCE     80,FN$EXP
        TERMINATE   1
```

Рис. 5

Особенностью непрерывных функций является то, что они принимают "непрерывные" (но только целочисленные) значения в диапазоне от Y1 до Yn , где n - количество точек таблицы. В отличие от них дискретные числовые функции, тип которых кодируется буквой **D** в **операнде B** оператора определения функции, принимают только отдельные (дискретные) значения, заданные координатами Yi в строках, следующих за оператором определения **FUNCTION**. При вычислении дискретной

функции текущее значение СЧА-аргумента, указанного в поле A оператора **FUNCTION,** сравнивается по условию <= последовательно со всеми значениями упорядоченных по возрастанию координат Xi до выполнения этого условия при некотором i. Значением функции становится целая часть соответствующего значения Yi.

Если последовательность значений аргумента таблицы с координатами точек функции представляет числа натурального ряда (1,2,3,...,n), то такую дискретную функцию с целью экономии памяти и машинного времени удобно определить как списковую числовую функцию (тип L).

Пусть в модели на рис. 5 заявки, моделируемые транзактами, с равной вероятностью 1/3 должны относиться к одному из трех классов (типов) 1,2 и 3, а среднее время задержки обслуживания заявок каждого типа должно составлять соответственно 70, 80 и 90 единиц модельного времени. Это может быть обеспечено способом, показанным на рис. 6.

В блоке ASSIGN в параметр TYPE каждого сгенерированного транзакта заносится тип заявки, получаемый с помощью **дискретной функции CLASS**. Аргументом функции является **генератор случайных чисел RN1**, а координаты ее таблицы представляют собой обратную функцию распределения дискретной случайной величины "класс заявки" с одинаковыми вероятностями каждого из трех значений случайной величины.

Поле A блока ADVANCE содержит ссылку на **списковую функцию MEAN**, аргументом которой служит **параметр TYPE** входящих в блок транзактов. В зависимости от значений этого параметра (типа заявки) среднее время задержки принимает одно из трех возможных значений **функции MEAN**: 70, 80 или 90 единиц.

```
 EXP     FUNCTION    RN1,C24
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915
.7,1.2/.75,1.38/.8,1.6/.84,1.85/.88,2.12/.9,2.3
.92,2.52/.94,2.81/.95,2.99/.96,3.2/.97,3.5/.98,3.9
.99,4.6/.995,5.3/.998,6.2/.999,7/.9998,8
 CLASS   FUNCTION    RN1,D3
.333,1/.667,2/1,3
 MEAN    FUNCTION    P$TYPE,L3
1,70/2,80/3,90
        GENERATE    100,FN$EXP
        ASSIGN      TYPE,FN$CLASS
        ADVANCE     FN$MEAN,FN$EXP
        TERMINATE   1
```

Рис. 6

Следует отметить, что в данном примере можно было бы не использовать параметр TYPE и обойтись одной дискретной функцией, возвращающей с равной вероятностью одно из трех возможных значений среднего времени задержки. Однако использование параметров дает некоторые дополнительные возможности, которые будут рассмотрены позже.

Транзакты могут входить в модель не только через блок GENERATE, но и путем создания копий уже существующих транзактов в **блоке SPLIT** (расщепить), имеющем следующий формат:

```
имя     SPLIT    A,B,C
```

В **поле A** задается число создаваемых копий исходного транзакта (родителя), входящего в блок SPLIT. После выхода из **блока SPLIT** транзакт-родитель направляется в следующий блок, а все транзакты-потомки поступают в блок, указанный в **поле B**. Если **поле B** пусто, то все копии поступают в следующий блок.

Транзакт-родитель и его потомки, выходящие из **блока SPLIT**, могут быть пронумерованы в параметре, имя или номер которого указаны в **поле C**. Если у транзакта-родителя значение этого параметра при входе в блок SPLIT было равно k, то при выходе из блока оно станет равным k+1, а значения этого параметра у транзактов-потомков ока-

жутся равными k+2, k+3 и т.д.
Например, блок

SPLIT 5,MET1,NUM

создает пять копий исходного транзакта и направляет их в блок с именем MET1. Транзакт-родитель и потомки нумеруются в параметре с именем NUM. Если, например, перед входом в блок значение этого параметра у транзакта-родителя было равно 0, то при выходе из блока оно станет равным 1, а у транзактов-потомков значения параметра NUM будут равны 2, 3, 4, 5 и 6.

2.2. Блоки, связанные с аппаратными объектами

Все примеры моделей, рассматривавшиеся выше, пока еще не являются моделями систем массового обслуживания, так как в них не учтена основная особенность СМО: конкуренция заявок на использование некоторых ограниченных ресурсов системы. Все транзакты, входящие в эти модели через блок **GENERATE**, немедленно получают возможность "обслуживания" в блоке **ADVANCE**, который никогда не "отказывает" транзактам во входе, сколько бы транзактов в нем не находилось.

Для моделирования ограниченных ресурсов СМО в модели должны присутствовать аппаратные объекты: одноканальные или многоканальные устройства. **Одноканальные устройства создаются в текущей модели** при использовании блоков **SEIZE** (занять) и **RELEASE** (освободить), имеющих следующий формат:

имя SEIZE A
имя RELEASE A

В **поле A** указывается номер или имя устройства. Если транзакт входит в **блок SEIZE**, то устройство, указанное в **поле A**, становится занятым и остаётся в этом состоянии до тех пор, пока этот же транзакт не пройдёт соответствующий **блок RELEASE**, освобождая устройство. Если устройство, указанное в **поле A блока SEIZE**, уже занято каким-либо транзактом, то никакой другой транзакт не может войти в этот блок и остаётся в предыдущем блоке. Транзакты, задержанные (заблокированные) перед **блоком SEIZE**, остаются в списке текущих событий и при освобождении устройства обрабатываются с учетом приоритетов и очередности поступления.

Каждое устройство имеет следующие СЧА: F - состояние устройства (0 - свободно,1 - занято); **FR** - коэффициент использования в долях 1000; **FC** - число занятий устройства; **FT** - целая часть среднего времени занятия устройства.

Воспользуемся блоками SEIZE и RELEASE для моделирования одноканальной СМО с ожиданием (рис. 7). Теперь блок ADVANCE находится между блоками SEIZE и RELEASE, моделирующими занятие и освобождение устройства с именем SYSTEM, и поэтому в нем может находиться только один транзакт. Транзакты, выходящие из блока GENERATE в моменты занятости устройства, не смогут войти в блок SEIZE и будут оставаться в блоке GENERATE, образуя очередь в списке текущих событий.

```
 EXP    FUNCTION    RN1,C24
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915
.7,1.2/.75,1.38/.8,1.6/.84,1.85/.88,2.12/.9,2.3
.92,2.52/.94,2.81/.95,2.99/.96,3.2/.97,3.5/.98,3.9
.99,4.6/.995,5.3/.998,6.2/.999,7/.9998,8
       GENERATE    100,FN$EXP
       SEIZE       SYSTEM
       ADVANCE     80,FN$EXP
       RELEASE     SYSTEM
       TERMINATE   1
```

Рис. 7

Для моделирования захвата (прерывания) одноканального устройства вместо блоков SEIZE и RELEASE используются соответственно **блоки PREEMPT** (захватить) и **RETURN** (вернуть). Блок **PREEMPT** имеет

следующий формат:

имя PREEMPT A,B,C,D,E

В **поле A** указывается имя или номер устройства, подлежащего захвату. В **поле B** кодируется условие захвата. Если это поле пусто, то захват возникает, если обслуживаемый транзакт сам не является захватчиком. Если же в поле B записан операнд PR, то захват возникает, если приоритет транзакта-захватчика выше, чем приоритет обслуживаемого транзакта.

Поля C, D и E определяют поведение транзактов, обслуживание которых было прервано. **Поле C** указывает имя блока, в который будет направлен прерванный транзакт. **В поле D** может быть указан номер или имя параметра прерванного транзакта, в который записывается время, оставшееся этому транзакту до завершения обслуживания на устройстве. При отсутствии операнда **в поле E** прерванный транзакт сохраняет право на автоматическое восстановление на устройстве по окончании захвата. Если же **в поле E** указан **операнд RE**, то транзакт теряет такое право.

Блок RETURN имеет единственный **операнд A**, содержащий имя или номер устройства, подлежащего освобождению от захвата.

Блоки PREEMPT и RETURN могут быть использованы для моделирования СМО с абсолютными приоритетами. В простейших случаях, при одном уровне захвата, в блоке PREEMPT используется единственный операнд **A**. При этом прерванный транзакт переводится симулятором из списка будущих событий в так называемый список прерываний устройства, а по окончании захвата устройства возвращается в список будущих событий с предварительно вычисленным временем занятия устройства для продолжения обслуживания.

Для создания в модели многоканальных устройств (МКУ) они должны быть предварительно определены с помощью **операторов определения STORAGE** (память), имеющих следующий формат:

имя STORAGE A

Здесь имя - имя МКУ, используемое для ссылок на него; A – емкость (количество каналов обслуживания) МКУ, задаваемая константой.

Для занятия и освобождения каналов обслуживания МКУ используется пара **блоков ENTER** (войти) и **LEAVE** (покинуть), имеющих следующий формат:

имя ENTER A,B

имя LEAVE A,B

В **поле A** указывается номер или имя МКУ, в **поле B** число каналов МКУ, занимаемых при входе в блок **ENTER** или освобождаемых при входе в **блок LEAVE.** Обычно **поле B** пусто, и в этом случае по умолчанию занимается или освобождается один канал.

При входе транзакта в блок ENTER текущее содержимое МКУ увеличивается на число единиц, указанное в **поле B** . Если свободная емкость МКУ меньше значения **поля B**, то транзакт не может войти в **блок ENTER** и остается в предыдущем блоке, образуя очередь в списке текущих событий.

При входе транзакта в **блок LEAVE** текущее содержимое МКУ уменьшается на число единиц, указанное в поле B. Не обязательно освобождается такое же число каналов МКУ, какое занималось при входе данного транзакта в блок ENTER, однако текущее содержимое МКУ не должно становиться отрицательным.

Многоканальные устройства имеют следующие СЧА: S - текущее содержимое МКУ; **R** - свободная емкость МКУ; **SR** - коэффициент использования в долях 1000; **SA** - целая часть среднего содержимого МКУ; **SM** - максимальное содержимое МКУ; **SC** - число занятий МКУ; **ST** - целая часть среднего времени занятия МКУ.

Воспользуемся блоками ENTER-LEAVE и оператором STORAGE для моделирования двухканальной СМО с ожиданием (рис. 8). Если текущее содержимое МКУ с именем STO2 меньше 2, т.е. в блоке ADVANCE нахо-

дится один или ни одного транзакта, то очередной транзакт, поступающий в модель через блок GENERATE, может войти в блок ENTER и затем в блок ADVANCE. Если же текущее содержимое МКУ равно 2, то очередной транзакт остается в блоке GENERATE, образуя очередь в списке текущих событий. По истечении задержки одного из двух обслуживаемых транзактов в блоке ADVANCE и после входа его в блок LEAVE первый из заблокированных транзактов сможет войти в блок ENTER.

```
 STO2   STORAGE     2
 EXP    FUNCTION    RN1,C24
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915
.7,1.2/.75,1.38/.8,1.6/.84,1.85/.88,2.12/.9,2.3
.92,2.52/.94,2.81/.95,2.99/.96,3.2/.97,3.5/.98,3.9
.99,4.6/.995,5.3/.998,6.2/.999,7/.9998,8
        GENERATE    100,FN$EXP
        ENTER       STO2
        ADVANCE     160,FN$EXP
        LEAVE       STO2
        TERMINATE   1
```

Рис. 8

К аппаратным объектам относятся также логические переключатели (ЛП), которые могут находиться в двух состояниях: "включено" и "выключено". В начале моделирования все ЛП находятся в состоянии "выключено". **Отдельные переключатели могут быть установлены в начальное состояние "включено"** с помощью **оператора INITIAL** (инициализировать), имеющего следующий формат:

```
INITIAL    LS$ имя

INITIAL    LS j
```

Здесь имя и j - соответственно имя и номер ЛП, устанавливаемого в начальное состояние "включено".

Для включения, выключения и инвертирования логических переключателей в процессе моделирования служит **блок LOGIC** (установить ЛП), имеющий следующий формат:

```
имя    LOGIC X    A
```

В поле A указывается имя или номер ЛП. Вспомогательный **операнд X** указывает вид операции, которая производится с логическим переключателем при входе транзакта в блок: **S** - включение, **R** - выключение, **I** - инвертирование. Например:

```
LOGIC S    9
LOGIC R    FLAG
```

Логические переключатели имеют единственный СЧА с названием **LS**. Значение СЧА равно 1, если ЛП включен, и 0, если он выключен.

2.3. Блоки для сбора статистических данных

Два последних примера в предыдущем параграфе представляют собой законченные модели одноканальной и многоканальной СМО с ожиданием. Однако такие модели разрабатываются обычно для исследования различных характеристик, связанных с ожиданием заявок в очереди: длины очереди, времени ожидания и т.п., а в приведенных примерах очередь транзактов образуется в списке текущих событий и недоступна исследователю. Для регистрации статистической информации о процессе ожидания транзактов в модели должны присутствовать статистические объекты: очереди или таблицы.

Объекты типа очередь создаются в модели путем использования блоков - регистраторов очередей: **QUEUE** (стать в очередь) и **DEPART** (уйти из очереди), имеющих следующий формат:

```
имя    0QUEUE     A,B
имя    DEPART     A,B
```

В **поле A** указывается номер или имя очереди, а в **поле B** - число единиц, на которое текущая длина очереди увеличивается при входе транзакта в блок QUEUE или уменьшается при входе транзакта в блок DEPART. Обычно поле B пусто, и в этом случае его значение по умолчанию принимается равным 1.

Для сбора статистики о транзактах, заблокированных перед каким-либо блоком модели, блоки QUEUE и DEPART помещаются перед и после этого блока соответственно. При прохождении транзактов через **блоки QUEUE и DEPART соответствующим образом изменяются следующие СЧА очередей: Q** - текущая длина очереди; **QM** - максимальная длина очереди; **QA** - целая часть средней длины очереди; **QC** - общее число транзактов, вошедших в очередь; **QZ** - число транзактов, прошедших через очередь без ожидания (число "нулевых" входов); **QT** - целая часть среднего времени ожидания с учетом "нулевых" входов; **QX** - целая часть среднего времени ожидания без учета "нулевых" входов.

Дополним приведенную на рис. 7 модель одноканальной СМО блоками QUEUE и DEPART (рис. 9). Теперь транзакты, заблокированные перед блоком SEIZE из-за занятости устройства SYSTEM, находятся в блоке QUEUE, внося свой вклад в статистику о времени ожидания, накапливаемую в статистическом объекте типа "очередь" с именем LINE. При освобождении устройства первый из заблокированных транзактов войдет в блок SEIZE и одновременно в блок DEPART, прекращая накопление статистики об ожидании этого транзакта.

```
       EXP    FUNCTION    RN1,C24
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915
.7,1.2/.75,1.38/.8,1.6/.84,1.85/.88,2.12/.9,2.3
.92,2.52/.94,2.81/.95,2.99/.96,3.2/.97,3.5/.98,3.9
.99,4.6/.995,5.3/.998,6.2/.999,7/.9998,8
       GENERATE    100,FN$EXP
       QUEUE       LINE
       SEIZE       SYSTEM
       DEPART      LINE
       ADVANCE     80,FN$EXP
       RELEASE     SYSTEM
       TERMINATE   1
```

Рис. 9

Очень часто исследователя интересует не только среднее значение времени ожидания в очереди, но и дисперсия этого времени, а также статистическое распределение выборки времени ожидания, представляемое обычно графически в виде гистограммы. Имея такое распределение, можно оценить вероятность того, что время ожидания превысит или не превысит некоторое заданное значение. **Для сбора и обработки данных о выборочном распределении времени ожидания в очереди** служат **статистические объекты типа Q-таблица.**

Для создания в модели такой **таблицы** она должна быть предварительно определена с помощью **оператора определения QTABLE** (**Q**-таблица), имеющего следующий формат:

```
имя    QTABLE    A,B,C,D
```

Здесь имя - имя таблицы, используемое для ссылок на нее; **A** - номер или имя очереди, распределение времени ожидания в которой необходимо получить; **B** - верхняя граница первого частотного интервала таблицы; **C** - ширина частотных интервалов; **D** - количество частотных интервалов.

Диапазон всевозможных значений времени ожидания в очереди, указанной в **поле A**, разбивается на ряд частотных интервалов, количество которых указано в **поле D**. Первый из этих интервалов имеет ширину от минус бесконечности до величины, указанной в **поле B**, включительно. Второй интервал включает значения, большие, чем величина первой границы в **поле B**, но меньшие или равные **B+C**, и т.д. Все

промежуточные интервалы имеют одинаковую ширину, указанную в **поле C**. Наконец, последний интервал включает все значения, большие, чем последняя граница. Значения операндов B, C и D должны задаваться целыми константами. Операнд B может быть неположительным, хотя для **Q-таблицы** это не имеет смысла, так как время не может быть отрицательным. **Операнды C и D должны быть строго положительными.**

При прохождении транзакта через блоки QUEUE и DEPART его время ожидания фиксируется, и к счетчику частотного интервала таблицы, в который попало это время, добавляется 1. Одновременно в таблице накапливается информация для вычисления среднего значения и среднеквадратического отклонения (корня из дисперсии) времени ожидания. По окончании моделирования среднее значение и среднеквадратическое отклонение времени ожидания, а также счетчики попаданий в различные частотные интервалы выводятся в стандартный отчет GPSS/PC.

Таблицы, как и другие объекты GPSS/PC, имеют СЧА: TC – общее число транзактов, вошедших в очередь, связанную с таблицей; **TB** - целая часть среднего времени ожидания в очереди; **TD** - целая часть среднеквадратического отклонения времени ожидания в очереди.

Дополним модель из примера на рис. 9 оператором QTABLE для получения распределения времени ожидания в очереди с именем LINE (рис. 10).

```
 WTIME  QTABLE     LINE,50,50,10
 EXP    FUNCTION   RN1,C24
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915
.7,1.2/.75,1.38/.8,1.6/.84,1.85/.88,2.12/.9,2.3
.92,2.52/.94,2.81/.95,2.99/.96,3.2/.97,3.5/.98,3.9
.99,4.6/.995,5.3/.998,6.2/.999,7/.9998,8
        GENERATE   100,FN$EXP
        QUEUE      LINE
        SEIZE      SYSTEM
        DEPART     LINE
        ADVANCE    80,FN$EXP
        RELEASE    SYSTEM
        TERMINATE  1
```

Рис. 10

Оператор определения таблицы с именем WTIME разбивает ось времени на 10 частотных интервалов. Первый интервал включает значения от 0 до 50, второй - от 50 до 100, третий - от 100 до 150 и т.д. Последний, десятый, интервал включает значения, превышающие 450. Если, например, время ожидания некоторого транзакта в очереди составило 145 единиц модельного времени, то к счетчику третьего частотного интервала будет добавлена 1. Следует заметить, что информация в таблицу с именем WTIME заносится автоматически, при входе транзактов в блоки QUEUE и DEPART, и никаких специальных мер для этого принимать не требуется.

Таблицы в GPSS/PC могут использоваться в более общем случае не только для табулирования времени ожидания в очереди, но и для получения выборочных распределений произвольных СЧА любых объектов модели. **Для определения таблиц** служит **оператор TABLE** (таблица), формат которого совпадает с форматом оператора QTABLE. Отличие состоит лишь в том, что в **поле A оператора TABLE записывается стандартный числовой атрибут**, выборочное распределение которого необходимо получить, а **операнды B, C и D** определяют разбиение на частотные интервалы диапазона всевозможных значений этого СЧА.

Занесение информации в таблицу, определяемую оператором TABLE, уже не может быть выполнено симулятором автоматически, как в случае Q-таблиц. **Для этого используется специальный блок TABULATE** (табулировать), имеющий следующий формат:

```
имя     TABULATE     A
```

В **поле A** указывается номер или имя таблицы, определенной соответствующим оператором TABLE.

При входе транзакта в блок TABULATE текущее значение табулируемого аргумента таблицы, указанного в поле A оператора TABLE, заносится в нее в соответствии с заданным в операторе TABLE разбиением области значений аргумента на частотные интервалы. **Одновременно корректируются текущие значения СЧА таблицы:** счетчик входов в таблицу **TC**, среднее время ожидания **TB** и среднеквадратическое отклонение времени ожидания **TD**.

Пусть, например, в модели многоканальной СМО, приведенной на рис. 8, надо получить распределение времени пребывания заявок в системе, включающего время ожидания в очереди и время обслуживания. Это может быть обеспечено способом, показанным на рис. 11.

Оператор TABLE определяет таблицу с именем TTIME, аргументом которой служит СЧА M1 - время пребывания транзакта в модели. В рассматриваемой модели значение СЧА M1 одновременно будет являться временем пребывания транзакта в СМО в том случае, если занесение информации в таблицу производить перед выходом транзакта из модели. Поэтому блок TABULATE, заносящий информацию о времени пребывания каждого транзакта в модели в таблицу TTIME, располагается перед блоком TERMINATE. Диапазон возможных значений времени пребывания транзакта в модели разбит в операторе TABLE на 12 частотных интервалов, ширина которых (кроме последнего) равна 100 единицам модельного времени.

```
TTIME   TABLE       M1,100,100,12
STO2    STORAGE     2
EXP     FUNCTION    RN1,C24
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915
.7,1.2/.75,1.38/.8,1.6/.84,1.85/.88,2.12/.9,2.3
.92,2.52/.94,2.81/.95,2.99/.96,3.2/.97,3.5/.98,3.9
.99,4.6/.995,5.3/.998,6.2/.999,7/.9998,8
        GENERATE    100,FN$EXP
        ENTER       STO2
        ADVANCE     160,FN$EXP
        LEAVE       STO2
        TABULATE    TTIME
        TERMINATE   1
```

Рис. 11

2.4. Блоки, изменяющие маршруты транзактов

В приведенных выше примерах транзакты, выходящие из любого блока, всегда поступали в следующий блок. В более сложных моделях возникает необходимость направления транзактов к другим блокам в зависимости от некоторых условий. Эту возможность обеспечивают блоки изменения маршрутов транзактов.

Блок TRANSFER (передать) служит для передачи входящих в него транзактов в блоки, отличные от следующего. Блок имеет девять режимов работы, из которых рассмотрим здесь лишь три наиболее часто используемых. В этих трех режимах блок имеет следующий формат:

```
имя     TRANSFER    A,B,C
```

Смысл операндов в полях A, B и C зависит от режима работы блока.
В режиме безусловной передачи поля A и C пусты, а в **поле B** указывается имя блока, к которому безусловным образом направляется транзакт, вошедший в блок TRANSFER. Например:

```
        TRANSFER    ,FINAL
```

В режиме статистической передачи операнд A определяет вероятность, с которой транзакт направляется в блок, указанный в **поле C**. С вероятностью **1-A** транзакт направляется в блок, указанный в **поле B** (в следующий, если поле B пусто).

Вероятность в поле A может быть задана непосредственно десятичной дробью, начинающейся с точки. Например, блок

```
        TRANSFER    .75,THIS,THAT
```

с вероятностью 0,75 направляет транзакты в блок с именем THAT, а с вероятностью 0,25 - в блок с именем THIS.

Если же поле A начинается не с десятичной точки и не содержит одного из ключевых слов - признаков других режимов работы блока, то его значение рассматривается как количество тысячных долей в вероятности передачи. Например, предыдущий блок TRANSFER можно записать также в следующем виде:

```
TRANSFER    750,THIS,THAT
```

В режиме логической передачи в поле A записывается **ключевое слово BOTH** (оба). Транзакт, поступающий в блок TRANSFER, сначала пытается войти в блок, указанный в поле B (или в следующий блок, если **поле B пусто**), а если это не удается, т.е. **блок B** отказывает транзакту во входе, то в блок, указанный в поле C. Если и эта попытка неудачна, то транзакт задерживается в блоке TRANSFER до изменения условий в модели, делающего возможным вход в один из блоков B или C, причем при одновременно возникшей возможности предпочтение отдается блоку B. Например:

```
TRANSFER    BOTH,MET1,MET2
```

Блок TEST (проверить) служит для задержки или изменения маршрутов транзактов в зависимости от соотношения двух СЧА. Он имеет следующий формат:

```
имя     TEST X    A,B,C
```

Вспомогательный **операнд X** содержит условие проверки соотношения между СЧА и может принимать следующие значения: **L** (меньше); **LE** (меньше или равно); **E** (равно); **NE** (не равно); **GE** (больше или равно); **G** (больше). **Поле A** содержит первый, а поле B - второй из сравниваемых СЧА. Если проверяемое условие A X B выполняется, то блок TEST пропускает транзакт в следующий блок. Если же это условие не выполняется, то транзакт переходит к блоку, указанному в поле C, а если оно пусто, то задерживается перед блоком TEST.

Например, блок

```
TEST LE     P$TIME,C1
```

не впускает транзакты, у которых значение параметра с именем TIME больше текущего модельного времени. Блок

```
TEST L      Q$LINE,5,OUT
```

направляет транзакты в блок с именем OUT, если текущая длина очереди LINE больше либо равна 5.

Для задержки или изменения маршрута транзактов в зависимости от состояния аппаратных объектов модели служит **блок GATE** (впустить), имеющий следующий формат:

```
имя     GATE X    A,B
```

Вспомогательный **операнд X** содержит код состояния проверяемого аппаратного объекта, а в **поле A** указывается имя или номер этого объекта. Если проверяемый объект находится в заданном состоянии, то блок GATE пропускает транзакт к следующему блоку. Если же заданное в блоке условие не выполняется, то транзакт переходит к блоку, указанному в **поле B**, а если это поле пусто, то задерживается перед блоком GATE.

Операнд X может принимать следующие значения: U (устройство занято); **NU** (устройство свободно); **I** (устройство захвачено); **NI** (устройство не захвачено); **SE** (МКУ пусто); **SNE** (МКУ не пусто); **SF** (МКУ заполнено); **SNF** (МКУ не заполнено); **LS** (ЛП включен), **LR** (ЛП выключен).

Например, блок

```
GATE SNE    BUF3
```

отказывает во входе транзактам, поступающим в моменты, когда в МКУ с именем BUF3 все каналы обслуживания свободны. Блок

```
GATE LR     4,BLOK2
```

направляет транзакты в блок с именем BLOK2, если в момент их поступления ЛП с номером 4 включен.

Блоки рассматриваемой группы используются при моделировании различных СМО с потерями заявок. Воспользуемся, например, блоками

TRANSFER для моделирования двухканальной СМО с отказами и повторными попытками (рис. 12).

```
STO2    STORAGE     2
EXP     FUNCTION    RN1,C24
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915
.7,1.2/.75,1.38/.8,1.6/.84,1.85/.88,2.12/.9,2.3
.92,2.52/.94,2.81/.95,2.99/.96,3.2/.97,3.5/.98,3.9
.99,4.6/.995,5.3/.998,6.2/.999,7/.9998,8
        GENERATE    100,FN$EXP
ENT1    TRANSFER    BOTH,,REFUS
        ENTER       STO2
        ADVANCE     160,FN$EXP
        LEAVE       STO2
        TERMINATE   1
REFUS   TRANSFER    .1,,OUT
        ADVANCE     250,FN$EXP
        TRANSFER    ,ENT1
OUT     TERMINATE   1
```

Рис. 12

Транзакты, поступающие в модель, попадают в блок TRANSFER с именем ENT1, работающий в логическом режиме. Если в момент поступления транзакта в МКУ STO2 хотя бы один канал свободен, то блок TRANSFER направит транзакт в следующий блок, т.е. в блок ENTER. Если же в момент поступления оба канала МКУ заняты, и поэтому блок ENTER отказывает во входе, то транзакт будет направлен в блок TRANSFER с именем REFUS, работающий в статистическом режиме. С вероятностью 0,9 транзакты из этого блока передаются в следующий блок, задерживаются в нем на случайное время и с помощью блока TRANSFER, работающего в безусловном режиме, передаются вновь на вход модели в блок с именем ENT1. С вероятностью 0,1 транзакты из блока с именем REFUS передаются в блок TERMINATE с именем OUT для уничтожения.

Следует заметить, что для уничтожения транзактов, получивших отказ в обслуживании, понадобился отдельный блок TERMINATE для фиксации в стандартном отчете количества потерянных транзактов с помощью счетчика блока с именем OUT (СЧА N$OUT).

Для моделирования той же СМО может быть использован также блок TEST (рис. 13). В этом варианте модели транзакт проходит в блок ENTER, если текущее число занятых каналов (СЧА S$STO2) меньше 2.

```
STO2    STORAGE     2
EXP     FUNCTION    RN1,C24
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915
.7,1.2/.75,1.38/.8,1.6/.84,1.85/.88,2.12/.9,2.3
.92,2.52/.94,2.81/.95,2.99/.96,3.2/.97,3.5/.98,3.9
.99,4.6/.995,5.3/.998,6.2/.999,7/.9998,8
        GENERATE    100,FN$EXP
ENT1    TEST L      S$STO2,2,REFUS
        ENTER       STO2
        ADVANCE     160,FN$EXP
        LEAVE       STO2
        TERMINATE   1
REFUS   TRANSFER    .1,,OUT
        ADVANCE     250,FN$EXP
        TRANSFER    ,ENT1
OUT     TERMINATE   1
```

Рис. 13

При использовании блока GATE модель принимает вид, показанный на рис. 14. В этом варианте транзакт проходит в блок ENTER, если

условие "МКУ STO2 не заполнено" истинно.

```
 STO2   STORAGE     2
 EXP    FUNCTION    RN1,C24
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915
.7,1.2/.75,1.38/.8,1.6/.84,1.85/.88,2.12/.9,2.3
.92,2.52/.94,2.81/.95,2.99/.96,3.2/.97,3.5/.98,3.9
.99,4.6/.995,5.3/.998,6.2/.999,7/.9998,8
        GENERATE    100,FN$EXP
 ENT1   GATE SNF    STO2,REFUS
        ENTER       STO2
        ADVANCE     160,FN$EXP
        LEAVE       STO2
        TERMINATE   1
 REFUS  TRANSFER    .1,,OUT
        ADVANCE     250,FN$EXP
        TRANSFER    ,ENT1
 OUT    TERMINATE   1
```

Рис. 14

2.5. Блоки, работающие с памятью

Для хранения в памяти отдельных числовых значений и массивов таких значений используются сохраняемые величины и матрицы сохраняемых величин.

Сохраняемые величины могут использоваться в модели для хранения исходных данных, которые надо изменять при различных прогонах модели, промежуточных значений и результатов моделирования. В начале моделирования все сохраняемые величины устанавливаются равными 0. **Для установки отличных от 0 начальных значений сохраняемых величин** используется **оператор INITIAL**, имеющий следующий формат:

```
INITIAL    X$ имя,значение
INITIAL    Xj     ,значение
```

Здесь имя и j - соответственно имя и номер сохраняемой величины, а значение - присваиваемое ей начальное значение (константа).

Для изменения сохраняемых величин в процессе моделирования служит блок SAVEVALUE (сохранить величину), имеющий следующий формат:

```
имя     SAVEVALUE     A,B
```

В **поле A** указывается номер или имя сохраняемой величины, в которую записывается значение **операнда B**. Если в поле A после имени (номера) сохраняемой величины стоит знак + или -, то значение **операнда B** добавляется или вычитается из текущего содержимого сохраняемой величины. Например:

```
SAVEVALUE     5,Q$LINE

SAVEVALUE     NREF+,1
```

Сохраняемые величины имеют единственный СЧА с названием X, значением которого является текущее значение соответствующей сохраняемой величины.

Изменим пример на рис. 14 таким образом, чтобы исходные данные модели (средний интервал поступления транзактов и среднее время обслуживания) были заданы сохраняемыми величинами, а результат моделирования (количество потерянных транзактов) фиксировался также в сохраняемой величине. Такая модель будет иметь вид, показанный на рис. 15.

Матрицы сохраняемых величин дают возможность упорядочить сохраняемые значения в виде матриц m*n, где m - число строк, n - число столбцов матрицы. **Каждая матрица должна быть перед началом моделирования определена с помощью оператора MATRIX** (определить матрицу), имеющего следующий формат:

```
имя     MATRIX    A,B,C
```

Поле A оператора не используется и сохранено в GPSS/PC для совместимости со старыми версиями GPSS. **В полях B и C** указываются соответственно число строк и столбцов матрицы, задаваемые константами, причем общее число элементов, равное произведению B на C, не должно превышать 8191. Например, оператор

```
MTAB    MATRIX    ,10,2
```

определяет матрицу с именем MTAB, содержащую десять строк и два столбца.

```
        INITIAL     X$TARR,100
        INITIAL     X$TSRV,160
STO2    STORAGE     2
EXP     FUNCTION    RN1,C24
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915
.7,1.2/.75,1.38/.8,1.6/.84,1.85/.88,2.12/.9,2.3
.92,2.52/.94,2.81/.95,2.99/.96,3.2/.97,3.5/.98,3.9
.99,4.6/.995,5.3/.998,6.2/.999,7/.9998,8
        GENERATE    X$TARR,FN$EXP
ENT1    GATE SNF    STO2,REFUS
        ENTER       STO2
        ADVANCE     X$TSRV,FN$EXP
        LEAVE       STO2
OUT     TERMINATE   1
REFUS   TRANSFER    .1,,COUT
        ADVANCE     250,FN$EXP
        TRANSFER    ,ENT1
COUT    SAVEVALUE   NREF+,1
        TRANSFER    ,OUT
```

Рис. 15

В начале моделирования элементы всех определенных матриц устанавливаются равными 0. **Для установки отличных от 0 начальных значений отдельных элементов матриц используется оператор INITIAL**, имеющий следующий формат:

```
INITIAL    MX$ имя (a,b),значение
INITIAL    MXj     (a,b),значение
```

Здесь **имя** и **j** - соответственно имя и номер матрицы; **a** и **b** - номера соответственно строки и столбца, задаваемые константами; значение - присваиваемое элементу матрицы начальное значение, задаваемое также константой.

Для изменения значений элементов матриц в процессе моделирования служит **блок MSAVEVALUE** (сохранить значение элемента матрицы), имеющий следующий формат:

```
имя     MSAVEVALUE    A,B,C,D
```

В **поле A** указывается имя или номер матрицы, после которого, как и в блоке SAVEVALUE, может стоять знак + или -. В **полях B и C** указываются номера соответственно строки и столбца, определяющие изменяемый элемент матрицы. В **поле D** указывается величина, используемая для изменения заданного элемента матрицы. Например:

```
MSAVEVALUE    5,3,2,X1
MSAVEVALUE    MTAB+,P$ROW,P$COL,1
```

Матрицы имеют единственный СЧА с названием MX, ссылка на который записывается в следующем виде:

```
MX$ имя (a,b)
MXj (a,b)
```

Здесь **имя** и **j** - соответственно имя и номер матрицы; **a** и **b** - номера соответственно строки и столбца, задаваемые константами или ссылками на СЧА параметров транзактов. Например:

```
MX5(2,1)

MX$MTAB(P$ROW,P$COL)
```

2.6. Блоки для работы со списками пользователя

Так как заблокированные транзакты находятся в списке текущих событий, то при большом количестве таких транзактов симулятор расходует слишком много времени на просмотр этого списка с целью выбора очередного транзакта для продвижения. Для экономии машинного времени заблокированные транзакты целесообразно помещать в так называемые списки пользователя и оставлять их там до тех пор, пока не выполнятся условия, позволяюшие дальнейшее продвижение этих транзактов. Кроме того, размещение ожидающих транзактов в списках пользователя позволяет организовать различные дисциплины очередей, отличные от дисциплины "раньше пришел - раньше обслужен", реализованной в списке текущих событий.

Списки пользователя представляют собой некоторые буферы, куда могут временно помещаться транзакты, выведенные из списка текущих событий. В отличие от списков текущих и будущих событий транзакты вводятся в списки пользователя и выводятся из них не автоматически, а в соответствии с логикой модели с помощью специальных блоков.

Для ввода транзактов в список пользователя служит блок LINK (ввести в список), который может быть использован в двух режимах: условном и безусловном. Ограничимся рассмотрением лишь безусловного режима, в котором блок LINK имеет следующий формат:

имя LINK A,B

В **поле A** задается имя или номер списка пользователя, в который безусловным образом помещается транзакт, вошедший в блок. **Поле B** определяет, в какое место списка пользователя следует поместить этот транзакт. **Если в поле B записано ключевое слово FIFO**, то транзакт помещается в конец списка, если **LIFO** - в начало списка. В других случаях транзакты упорядочиваются в соответствии с вычисленным значением поля B, где обычно записывается **один из СЧА транзактов**, таких как **PR, M1 или P**. Если **поле B содержит СЧА PR**, то транзакты упорядочиваются по убыванию приоритета. В остальных случаях производится упорядочение по возрастанию указанного СЧА.

Например, блок

LINK 5,FIFO

помещает транзакты в список пользователя с номером 5 в порядке их поступления в блок. Блок

LINK BUFER,P$ORDER

помещает транзакты в список пользователя с именем BUFER, упорядочивая их по возрастанию параметра с именем ORDER.

Условия, при которых транзакт помещается в список пользователя, в безусловном режиме проверяются средствами, предусмотренными разработчиком модели. Например, направить транзакт в список пользователя в случае занятости устройства можно так, как показано на рис. 16. Если устройство с именем FAC4 занято, то блок GATE не впускает транзакт в блок SEIZE, а направляет его в блок LINK с именем WAIT, и транзакт вводится в конец списка пользователя с именем BUFER.

```
                .....................
                GATE NU     FAC4,WAIT
                SEIZE       FAC4
                .....................
        WAIT    LINK        BUFER,FIFO
                .....................
```

Рис. 16

Для вывода одного или нескольких транзактов из списка пользователя и помещения их обратно в список текущих событий служит блок UNLINK (вывести из списка), имеющий следующий формат:

имя UNLINK X A,B,C,D,E,F

В **поле A** указывается имя или номер списка пользователя. **Поле B** содержит имя блока, в который переходят выведенные из списка пользователя транзакты. В **поле C** указывается число выводимых транзактов или ALL для вывода всех находящихся в списке транзактов.

Операнды в полях D и E вместе со вспомогательным **операндом X** определяют способ и условия вывода транзактов из списка пользователя. Если поля D и E пусты, то и операнд X не используется, а транзакты выводятся с начала списка пользователя. Если поле D содержит **ключевое слово BACK**, то поле E и вспомогательный операнд X не используются, а транзакты выводятся с конца списка. В остальных случаях значение поля D интерпретируется как номер параметра транзактов, находящихся в списке пользователя, а из списка выводится заданное число тех транзактов, у которых значение этого параметра по отношению к значению операнда в поле E удовлетворяет условию, заданному вспомогательным операндом X. Операнд X принимает те же значения, что и в блоке TEST.

В поле F указывается имя блока, куда переходит транзакт, выходящий из блока UNLINK, если из списка пользователя не выведен ни один транзакт. Если это поле пусто, то выводящий транзакт переходит в следующий блок независимо от количества выведенных транзактов.

Например, блок

```
UNLINK      5,NEXT,1
```

выводит из списка пользователя с номером 5 один транзакт с начала списка и направляет его в блок с именем NEXT. Блок

```
UNLINK      BUFER,ENT1,1,BACK
```

выводит из списка пользователя с именем BUFER один транзакт с конца списка и направляет его в блок с именем ENT1. Блок

```
UNLINK E    P$UCH,MET2,ALL,COND,P$COND,MET3
```

выводит из списка пользователя, номер которого записан в параметре UCH выводящего транзакта, и направляет в блок с именем MET2 все транзакты, содержимое параметра COND которых равно содержимому одноименного параметра выводящего транзакта. Если таких транзактов в списке не окажется, то выводящий транзакт будет направлен в блок с именем MET3, в противном случае - к следующему блоку.

Следует отметить следующие особенности выполнения блока UNLINK. Во-первых, если поля D и E содержат ссылки на СЧА транзактов, то поле D вычисляется относительно транзактов в списке пользователя, а поле E - относительно активного транзакта. Во-вторых, после вывода транзактов из списка симулятор продолжает или начинает продвижение транзакта с наивысшим приоритетом, а при равенстве приоритетов отдает предпочтение транзакту-инициатору вывода.

Каждый список пользователя имеет следующие СЧА: CH - текущая длина списка; CA - средняя длина списка (целая часть); CM - максимальная длина списка; CC - общее число транзактов, вошедших в список; CT - целая часть среднего времени пребывания транзакта в списке.

Воспользуемся рассмотренными блоками для моделирования многоканальной СМО с ожиданием транзактов в списке пользователя (рис. 17). Если МКУ с именем STO2 не заполнено, блок GATE впускает вновь прибывший транзакт в блок ENTER, и в МКУ занимается один канал. Если же МКУ заполнено, то блок GATE направляет транзакт в блок LINK с именем WAIT, помещающий транзакт в конец списка пользователя с именем BUFER, моделирующего очередь к МКУ. Каждый транзакт, покидающий МКУ по завершении обслуживания и освобождающий один канал, проходит блок UNLINK и выводит один транзакт с начала списка (если список не пуст), направляя его в блок с именем ENT1 на занятие канала в МКУ.

```
 STO2   STORAGE     2
 EXP    FUNCTION    RN1,C24
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915
.7,1.2/.75,1.38/.8,1.6/.84,1.85/.88,2.12/.9,2.3
.92,2.52/.94,2.81/.95,2.99/.96,3.2/.97,3.5/.98,3.9
.99,4.6/.995,5.3/.998,6.2/.999,7/.9998,8
```

```
        GENERATE    100,FN$EXP
        GATE SNF    STO2,WAIT
ENT1    ENTER       STO2
        ADVANCE     160,FN$EXP
        LEAVE       STO2
        UNLINK      BUFER,ENT1,1
        TERMINATE   1
WAIT    LINK        BUFER,FIFO
```

Рис. 17

Заметим, что для изменения дисциплины обслуживания на "позже пришел - раньше обслужен" достаточно или заменить в поле B блока LINK FIFO на LIFO, или записать в поле D блока UNLINK операнд BACK. Следует также обратить внимание на то, что блоки QUEUE-DEPART для сбора статистики об ожидающих транзактах не используются, так как почти все те же данные можно получить из статистики о списке пользователя.

Рассмотрим еще один пример, иллюстрирующий использование списков пользователя для организации нестандартных дисциплин обслуживания. Пусть в одноканальной СМО с ожиданием требуется организовать такую дисциплину, при которой приоритет отдается заявкам с наименьшим временем обслуживания. Такая модель будет иметь вид, показанный на рис. 18.

В параметр TSRV поступающих в модель транзактов в блоке ASSIGN записывается случайное время обслуживания, вычисляемое с использованием функции EXP. Если устройство SYSTEM свободно, то блок GATE впускает транзакт в блок SEIZE, и устройство занимается на время P$TSRV. Если же в момент поступления транзакта устройство занято, то блок GATE направляет транзакт в блок LINK, который вводит тран-

```
EXP     FUNCTION    RN1,C24
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915
.7,1.2/.75,1.38/.8,1.6/.84,1.85/.88,2.12/.9,2.3
.92,2.52/.94,2.81/.95,2.99/.96,3.2/.97,3.5/.98,3.9
.99,4.6/.995,5.3/.998,6.2/.999,7/.9998,8
        GENERATE    100,FN$EXP
        ASSIGN      TSRV,80,EXP
        GATE NU     SYSTEM,WAIT
 SFAC   SEIZE       SYSTEM
        ADVANCE     P$TSRV
        RELEASE     SYSTEM
        UNLINK      LINE,SFAC,1
        TERMINATE   1
WAIT    LINK        LINE,P$TSRV
```

Рис. 18

закт в список пользователя LINE, упорядочивая транзакты по возрастанию времени обслуживания, записанного в параметре P$TSRV. Блок UNLINK по освобождении устройства выводит с начала списка транзакт с наименьшим временем обслуживания, обеспечивая тем самым заданную дисциплину.

3. УПРАВЛЯЮЩИЕ ОПЕРАТОРЫ GPSS/PC

Для управления прогоном модели используются управляющие операторы GPSS/PC. С одним из них - оператором START - мы уже сталкивались при рассмотрении блока TERMINATE. **Оператор START** (начать) имеет следующий формат:

```
START    A,B,C,D
```

Поле A содержит константу, задающую начальное значение счетчика завершений. В **поле B** может быть записано **ключевое слово NP** - признак подавления формирования стандартного отчета по завершении

моделирования. Если поле B пусто, то по окончании прогона модели формируется отчет со стандартной статистической информацией о всех объектах модели (см. разд. 5). Поле C не используется и сохранено для совместимости со старыми версиями GPSS. Поле D может содержать 1 для включения в отчет списков текущих и будущих событий. Если поле D пусто, то выдача в отчет содержимого этих списков не производится.

Оператор SIMULATE (моделировать) устанавливает предел реального времени, отводимого на прогон модели. Если прогон не завершится до истечения этого времени, то он будет прерван принудительно с выдачей накопленной статистики в отчет.

Оператор SIMULATE имеет единственный операнд A, содержащий предельное время моделирования в минутах, задаваемое константой. Оператор размещается перед оператором START, начинающим лимитированный прогон.

Оператор RMULT (установить значения генераторов) позволяет перед началом прогона установить начальные значения генераторов случайных чисел RN, определяющие генерируемые ими последовательности. **Поля A-G** оператора могут содержать начальные значения генераторов соответственно RN1-RN7, задаваемые константами. Начальные значения генераторов, не установленные операторами RMULT, совпадают с номерами генераторов.

Оператор RESET (сбросить) сбрасывает всю статистическую информацию, накопленную в процессе прогона модели. При этом состояние аппаратных, динамических и запоминающих объектов, а также генераторов случайных чисел сохраняется, и моделирование может быть возобновлено с повторным сбором статистики. Оператор не имеет операндов.

С оператором RESET связано различие между относительным (СЧА C1) и абсолютным (СЧА AC1) модельным временем. Таймер относительного времени C1 измеряет модельное время, прошедшее после последнего сброса статистики оператором RESET, а таймер абсолютного времени AC1 - модельное время, прошедшее после начала первого прогона модели. Если не использовалось ни одного оператора RESET, то значения этих таймеров совпадают. Оператор RESET устанавливает таймер C1 в ноль и не влияет на таймер AC1.

Оператор RESET используется обычно при моделировании нестационарных процессов, когда требуется собрать статистику по отдельным интервалам стационарности или исключить влияние переходного периода на собираемую статистическую информацию.

Пусть, например, в модели, приведенной на рис. 18, необходимо отбросить статистику, собираемую на первой тысяче транзактов. Это может быть сделано способом, показанным на рис. 19.

Первый оператор START начинает прогон модели длиной 1000 транзактов (переходный период). Поскольку статистика, накопленная на этом периоде, не используется, в поле B оператора указан признак подавления формирования отчета NP. Оператор RESET сбрасывает накопленную статистику, не изменяя состояния модели. Второй оператор START начинает основной прогон модели с формированием отчета по завершении прогона.

```
EXP     FUNCTION    RN1,C24
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915
.7,1.2/.75,1.38/.8,1.6/.84,1.85/.88,2.12/.9,2.3
.92,2.52/.94,2.81/.95,2.99/.96,3.2/.97,3.5/.98,3.9
.99,4.6/.995,5.3/.998,6.2/.999,7/.9998,8
        GENERATE    100,FN$EXP
        ASSIGN      TSRV,80,EXP
        GATE NU     SYSTEM,WAIT
 SFAC   SEIZE       SYSTEM
        ADVANCE     P$TSRV
        RELEASE     SYSTEM
```

```
        UNLINK      LINE,SFAC,1
        TERMINATE   1
WAIT    LINK        LINE,P$TSRV
        START       1000,NP
        RESET
        START       10000
```

Рис. 19

Оператор CLEAR (очистить) очищает модель, подготавливая ее к повторному прогону. При этом сбрасывается вся накопленная в предыдущем прогоне статистика, из модели удаляются все транзакты, и она приводится к исходному состоянию, как перед первым прогоном. Устанавливаются в ноль сохраняемые величины и матрицы, что следует учитывать при использовании этих объектов для хранения исходных данных. Исключение составляют генераторы случайных чисел, которые не возвращаются к своим начальным значениям, что позволяет повторить прогон модели на новой последовательности случайных чисел. Оператор не имеет операндов.

Оператор CLEAR используется обычно для организации нескольких независимых прогонов модели на разных последовательностях случайных чисел. Перед повторением прогона можно при необходимости переопределить отдельные объекты модели, например емкости многоканальных устройств.

Пусть, например, требуется повторить прогон модели, приведенной на рис. 17, три раза при емкости МКУ, равной 1, 2 и 3. Это может быть выполнено так, как показано на рис. 20. После каждой очистки модели оператором CLEAR оператор STORAGE устанавливает новое значение емкости МКУ с именем STO2.

Оператор END (закончить) завершает сеанс 0работы с GPSS/PC и возвращает управление в операционную систему. Оператор не имеет операндов.

```
STO2    STORAGE     1
EXP     FUNCTION    RN1,C24
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915
.7,1.2/.75,1.38/.8,1.6/.84,1.85/.88,2.12/.9,2.3
.92,2.52/.94,2.81/.95,2.99/.96,3.2/.97,3.5/.98,3.9
.99,4.6/.995,5.3/.998,6.2/.999,7/.9998,8
        GENERATE    100,FN$EXP
        GATE SNF    STO2,WAIT
ENT1    ENTER       STO2
        ADVANCE     160,FN$EXP
        LEAVE       STO2
        UNLINK      BUFER,ENT1,1
        TERMINATE   1
WAIT    LINK        BUFER,FIFO
        START       10000
        CLEAR
STO2    STORAGE     2
        START       10000
        CLEAR
STO2    STORAGE     3
        START       10000
```

Рис. 20

Как правило, управляющие операторы не включаются в исходную программу, т.е. не имеют номеров строк, а вводятся пользователем непосредственно с клавиатуры ПК.

4. НЕКОТОРЫЕ ПРИЕМЫ КОНСТРУИРОВАНИЯ GPSS-МОДЕЛЕЙ

4.1. Косвенная адресация

В рассматривавшихся до сих пор примерах моделей ссылки на различные объекты GPSS/PC производились исключительно по данным им произвольным именам. Такая адресация объектов удобна, когда речь идет о небольшом числе объектов каждого типа. Если же число объектов некоторого типа велико, то во избежание пропорционального роста количества блоков в модели используют ссылки на эти объекты по их номерам с использованием так называемой косвенной адресации.

Идея косвенной адресации заключается в том, что каждый транзакт в некотором своем параметре содержит номер того или иного объекта, а в полях блоков, адресующихся к объектам, записывается ссылка на этот параметр транзакта. Проиллюстрируем применение **косвенной адресации** на примере следующей модели.

```
EXP     FUNCTION    RN1,C24
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915
.7,1.2/.75,1.38/.8,1.6/.84,1.85/.88,2.12/.9,2.3
.92,2.52/.94,2.81/.95,2.99/.96,3.2/.97,3.5/.98,3.9
.99,4.6/.995,5.3/.998,6.2/.999,7/.9998,8
 CLASS  FUNCTION    RN1,D3
.333,1/.667,2/1,3
 MEAN   FUNCTION    P$TYPE,L3
1,70/2,80/3,90
 PRIOT  VARIABLE    4-P$TYPE
 STO2   STORAGE     2
 WTIME  QTABLE      LINE,50,50,10
 TTIME  TABLE       M1,100,100,12
        GENERATE    100,FN$EXP
        ASSIGN      TYPE,FN$CLASS
        PRIORITY    V$PRIOT
        QUEUE       LINE
        QUEUE       P$TYPE
        ENTER       STO2
        DEPART      P$TYPE
        DEPART      LINE
        ADVANCE     FN$MEAN,FN$EXP
        LEAVE       STO2
        TABULATE    TTIME
        TERMINATE   1
```

Рис. 21

Пусть на вход моделируемой многоканальной СМО с двумя каналами обслуживания поступает пуассоновский поток заявок со средним интервалом поступления 100 единиц модельного времени. Каждая заявка с равной вероятностью 1/3 относится к одному из трех классов: 1, 2 или 3, а среднее время обслуживания заявок каждого типа составляет соответственно 70, 80 и 90 единиц модельного времени. Чем меньше среднее время обслуживания заявки, тем выше ее приоритет. Необходимо построить модель, позволяющую оценить средние значения времени ожидания заявок каждого типа, а также распределения общего времени ожидания в очереди и общего времени пребывания в системе. Такая модель имеет вид, показанный на рис. 21.

Переменная PRIOT служит для вычисления приоритета транзакта как функции его класса, содержащегося в параметре с именем TYPE. Транзакты класса 1 (P$TYPE=1) получат приоритет 3, транзакты класса 2 - приоритет 2 и транзакты класса 3 - приоритет 1.

В блоке ASSIGN в параметр TYPE транзактов записывается класс заявки, разыгрываемый с помощью функции CLASS. В следующем блоке PRIORITY с помощью переменной PRIOT определяется приоритет транзактов, первоначально равный 0 (отсутствует поле E в блоке GENERATE).

Далее каждый транзакт "отмечается" в блоках QUEUE в двух очередях. Очередь с именем LINE является общей для транзактов всех

классов. Входя в следующий блок QUEUE, транзакт отмечается в очереди с номером 1, 2 или 3 в зависимости от класса заявки, записанного в параметре TYPE. Аналогичным образом фиксируется уход из очередей в блоках DEPART. Таким образом, в модели создается четыре объекта типа "очередь": одна очередь с именем LINE и три с номерами 1, 2 и 3. При этом три последние очереди создаются одной парой блоков QUEUE-DEPART! В этом и заключается эффект косвенной адресации.

Как уже отмечалось ранее, каждому имени объекта симулятор сам ставит в соответствие некоторый номер. При ссылках на объекты одного и того же типа одновременно по именам и номерам, как это имеет место в рассматриваемом примере, существует опасность параллельной адресации к одному и тому же объекту вместо двух разных или, наоборот, к двум разным объектам вместо одного. Так, в рассматриваемой модели мы, вообще говоря, не знаем, какой именно номер поставит симулятор в соответствие имени очереди LINE. Если этот номер будет от 1 до 3, то это приведет к ошибке, так как в модели окажется не четыре очереди, а три, причем в одну из них будет заноситься информация как обо всех транзактах, так и дополнительно о транзактах одного из трех классов. Как избежать такой ситуации?

К счастью, в большинстве случаев об этом можно не заботиться, поскольку симулятор ставит в соответствие именам объектов достаточно большие номера, начиная с 10000. При необходимости же можно воспользоваться оператором EQU, о котором уже говорилось выше, и самостоятельно сопоставить имени объекта желаемый номер. Например, в рассматриваемой модели для того, чтобы очередь с именем LINE имела номер 4, достаточно записать оператор:

```
LINE    EQU    4
```

4.2. Обработка одновременных событий

Так как модельное время в GPSS целочисленно, то оказывается вполне вероятным одновременное наступление двух или более событий, причем вероятность этого тем больше, чем крупнее выбранная единица модельного времени. В некоторых случаях одновременное наступление нескольких событий, или так называемый временной узел, может существенно нарушить логику модели.

Рассмотрим, например, еще раз модель на рис. 14. Здесь может образоваться временной узел между событиями "поступление транзакта на вход модели" и "завершение обслуживания в МКУ". Если непосредственно перед завершением обслуживания были заняты оба канала МКУ, то обработка временного узла зависит от последовательности транзактов, соответствующих событиям, в списке текущих событий.

Предположим, что первым в списке расположен транзакт, освобождающий канал МКУ. Тогда вначале будет обработан этот транзакт, т.е. событие "завершение обслуживания в МКУ", причем условие "МКУ STO2 не заполнено", проверяемое в блоке GATE, станет истинным. Затем будет обработан транзакт, поступивший на вход модели, в блок GATE с именем ENT1, из блока GENERATE или из блока TRANSFER в безусловном режиме. При этом транзакт будет впущен в блок ENTER, и МКУ в тот же момент модельного времени снова окажется заполненным. Такая ситуация при обработке временного узла представляется естественной.

Предположим теперь, что первым в списке текущих событий расположен транзакт, поступающий на вход модели. Так как условие "МКУ STO2 не заполнено" ложно, то блок GATE направит этот транзакт в блок с именем REFUS. Таким образом, в модели будет зафиксирован отказ в обслуживании, хотя в этот же момент модельного времени, после обработки транзакта, освобождающего канал, МКУ станет доступным.

Порядок расположения транзактов, соответствующих рассматриваемым событиям, в списке текущих событий случаен, и в среднем в половине случаев временной узел будет обрабатываться не так, как нужно. В результате статистика, связанная с отказами, окажется искаженной.

Для правильной обработки временного узла надо обеспечить такой порядок расположения транзактов в списке текущих событий, чтобы транзакт, освобождающий МКУ, всегда располагался первым. Этого мож-

но добиться, управляя приоритетами транзактов (рис. 22).

```
 STO2   STORAGE     2
 EXP    FUNCTION    RN1,C24
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915
.7,1.2/.75,1.38/.8,1.6/.84,1.85/.88,2.12/.9,2.3
.92,2.52/.94,2.81/.95,2.99/.96,3.2/.97,3.5/.98,3.9
.99,4.6/.995,5.3/.998,6.2/.999,7/.9998,8
        GENERATE    100,FN$EXP
 ENT1   GATE SNF    STO2,REFUS
        ENTER       STO2
        PRIORITY    1
        ADVANCE     160,FN$EXP
        LEAVE       STO2
        TERMINATE   1
 REFUS  TRANSFER    .1,,OUT
        ADVANCE     250,FN$EXP
        TRANSFER    ,ENT1
 OUT    TERMINATE   1
```

Рис. 22

Транзакты, поступающие в модель через блок GENERATE, имеют нулевой приоритет. Такой же приоритет имеют транзакты, получившие отказ в обслуживании, направленные в блок с именем REFUS и затем повторно поступающие в блок с именем ENT1. Те же транзакты, что поступают на обслуживание, повышают приоритет до 1 в блоке PRIORITY, и после выхода из блока ADVANCE возвращаются из списка будущих в список текущих событий, располагаясь в начале списка. Таким образом, нужный порядок транзактов обеспечивается, и временной узел будет обработан правильно.

Опасность неверной обработки временных узлов характерна для моделей со списками пользователя. Рассмотрим, например, еще раз модель на рис. 18. Здесь также возможен временной узел между событиями "приход транзакта" и "завершение обслуживания транзакта".

Пусть первым в списке текущих событий располагается вновь пришедший транзакт. Так как устройство с именем SYSTEM занято, то блок GATE направит этот транзакт в блок LINK, и он будет введен в список пользователя с именем LINE. Затем будет обработан транзакт, освобождающий устройство. Проходя через блок UNLINK, он выведет транзакт с начала списка пользователя и направит его в список текущих событий, где тот продвинется в блок SEIZE, занимая устройство SYSTEM.

Если же первым в списке текущих событий располагается транзакт, освобождающий устройство, то он выведет первый из ожидающих транзактов из списка пользователя в список текущих событий, где тот расположится после вновь пришедшего транзакта. Поэтому первым будет обработан вновь пришедший транзакт, который пройдет через блок GATE и займет устройство "без очереди". Транзакт-очередник, который был выведен из списка пользователя, "застрянет" перед блоком SEIZE и после очередного освобождения устройства займет его, нарушая, в свою очередь, логику работы модели.

Проведенный анализ показывает, что для правильной обработки временного узла необходимо обеспечить такой порядок расположения транзактов в списке текущих событий, чтобы первым всегда располагался вновь пришедший транзакт. В рассматриваемом случае этого можно добиться, используя блок PRIORITY с операндом BU (рис. 23).

Перед освобождением устройства обслуженный транзакт проходит через блок PRIORITY, который, оставляя неизменным приоритет транзакта PR, переводит его в конец списка текущих событий. При новом просмотре списка в случае наличия временного узла начинает обрабатываться вновь поступивший транзакт. Так как устройство еще занято, он направляется блоком GATE в список пользователя. При повторной обработке обслуженного транзакта тот освобождает устройство и выво-

дит очередной транзакт из списка пользователя. Таким образом, правильная обработка временного узла обеспечивается и в этом случае.

```
EXP     FUNCTION    RN1,C24
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915
.7,1.2/.75,1.38/.8,1.6/.84,1.85/.88,2.12/.9,2.3
.92,2.52/.94,2.81/.95,2.99/.96,3.2/.97,3.5/.98,3.9
.99,4.6/.995,5.3/.998,6.2/.999,7/.9998,8
        GENERATE    100,FN$EXP
        ASSIGN      TSRV,80,EXP
        GATE NU     SYSTEM,WAIT
 SFAC   SEIZE       SYSTEM
        ADVANCE     P$TSRV
        PRIORITY    PR,BU
        RELEASE     SYSTEM
        UNLINK      LINE,SFAC,1
        TERMINATE   1
 WAIT   LINK        LINE,P$TSRV
```

Рис. 23

5. КОМАНДЫ GPSS/PC И ТЕХНОЛОГИЯ РАБОТЫ С ПАКЕТОМ

5.1. Загрузка интегрированной среды

Пакет GPSS/PC включает в себя два основных модуля: модуль GPSSPC.EXE, представляющий интегрированную среду, в которой производится ввод, редактирование, отладка и выполнение модели, и модуль GPSSREPT.EXE, предназначенный для получения стандартного отчета GPSS/PC. Загрузка обоих модулей производится обычным образом из командной строки MS DOS или из программы-оболочки Norton Commander.

После загрузки интегрированной среды на экране появляется "заставка" с названием пакета: начинается так называемый сеанс работы с GPSS/PC. Затем заставка гасится, и появляется экран, разделенный на две части: большая верхняя часть содержит так называемое окно данных, меньшая нижняя часть - окно команд. Окно данных в начальный момент пусто, в окне команд в верхней командной строке высвечен символ "приглашения" >, сигнализирующий о готовности системы принимать команды.

5.2. Ввод новой модели

Если исходная программа с моделью еще не введена и не записана на диске, то необходимо ввести ее с клавиатуры. Ввод производится в командную строку. Сначала вводится номер строки очередного оператора и нажимается клавиша Пробел. Курсор автоматически перемещается к началу следующего поля - поля имени, и в позиции курсора высвечивается символ L , сигнализирующий о том, что вы находитесь в поле имени (LABEL - метка). Если оператор имеет имя, необходимо ввести его и нажать клавишу Пробел, в противном случае - сразу нажать клавишу Пробел. В любом случае курсор переходит к началу следующего поля - поля операции, о чем сигнализирует символ V (VERB - глагол) в позиции курсора. Необходимо ввести название оператора и нажать клавишу Пробел. Очень удобным является то, что название оператора не обязательно вводить полностью: как только транслятор распознает оператор по нескольким первым буквам, он после нажатия клавиши Пробел сам дополнит его до полного названия.

При синтаксической ошибке в операторе под командной строкой появляется указатель на место ошибки, причем ошибочный символ не вводится. Необходимо в этом случае повторить ввод символа.

Аналогичным образом вводятся поля операндов, при этом в позиции курсора высвечивается обозначение текущего поля (A,B, ... ,G). Для перехода к следующему полю операндов вводится запятая, для пе-

рехода к полю комментариев - Пробел. При переходе курсора в поле комментариев в позиции курсора высвечивается символ ; , сигнализирующий о возможности начать ввод комментария.

По окончании ввода последнего поля операндов или комментария следует нажать клавишу Enter, при этом введенный оператор транслируется и отображается в окне данных, а командная строка очищается, и в ее первой позиции снова появляется символ "приглашения".

По мере ввода новых операторов окно данных заполняется, и по окончании ввода в нем находится вся исходная программа в последовательности ввода, необязательно совпадающей с последовательностью нумерации строк. Для отображения в окне данных исходной программы в последовательности нумерации строк необходимо ввести в командную строку команду DISPLAY (отобразить). Эта команда, как и все остальные команды GPSS/PC, вводится без номера строки. С помощью команды DISPLAY можно также вывести в окно данных отдельную строку, указав ее номер в поле A команды, или последовательность строк, указав начальный и конечный номера в полях A и B соответственно.

5.3. Редактирование текста модели

Удалить строки из исходной программы можно командой DELETE (удалить), указав в полях A и B начальный и конечный номера удаляемой последовательности. Для удаления одной строки достаточно ввести лишь поле A.

При необходимости вставить в текст новый оператор, поместив его между уже введенными операторами, достаточно ввести его с промежуточным номером строки. Вы можете перенумеровать строки, введя команду RENUMBER (перенумеровать), в поле A которой указывается номер первой строки, а в поле B - шаг перенумерации.

Отредактировать содержимое строки можно с помощью команды EDIT (редактировать), в поле A которой указывается номер редактируемой строки. При вводе такой команды в командной строке появляется редактируемая строка. Подводя курсор к нужным позициям строки, вы можете внести в нее необходимые изменения. По окончании редактирования следует нажать клавишу Enter, и отредактированная строка перенесется в окно данных, заменив в исходной программе первоначальную строку с этим номером. Вы можете убедиться в этом, введя команду DISPLAY 2.

Если редактируемый оператор короткий, а изменений в нем много, то редактирование удобнее произвести, введя измененный оператор с тем же номером строки.

5.4. Запись и считывание модели с диска

Если работа с моделью предполагается и по окончании данного сеанса, то после ввода и редактирования исходную программу имеет смысл записать на диск. Для этого необходимо ввести команду SAVE (сохранить), в поле A которой указывается имя файла, в который будет записана модель. Файл должен иметь расширение .GPS.

Записав модель в файл, вы сможете в следующем сеансе работы с GPSS/PC не вводить ее заново с клавиатуры, а считать с диска, введя команду @ спецификация_файла, где спецификация_файла - полное имя файла, которое вы дали исходной программе в команде SAVE, включающее расширение .GPS. При выполнении команды @ операторы исходной программы по мере их считывания из файла транслируются и выводятся в окно данных.

5.5. Прогон модели и наблюдение за моделированием

После того, как исходная программа модели введена с клавиатуры или считана с диска и оттранслирована, в памяти ПК создалась текущая модель, и теперь можно выполнить ее прогон. Для этого в командную строку необходимо ввести управляющий оператор START, указав в поле A соответствующее начальное значение счетчика завершений.

После нажатия клавиши Enter оператор START переносится в окно данных, и прогон модели начинается. Об этом сигнализирует сообщение

Simulation in Progress ,

появляющееся в нижней строке командного окна - строке состояния, а также так называемый индикатор моделирования, мигающий в правой стороне нижней части окна данных.

Если прогон модели достаточно длинный, то можно наблюдать за процессом моделирования, открывая те или иные графические окна. Это производится путем нажатия клавиши Alt одновременно с символьной клавишей с первой буквой названия окна.

Например, после нажатия клавиш Alt+B в верхней части экрана на месте окна данных появляется окно блоков (BLOCKS), изображающее динамику продвижения транзактов через блок-схему модели. Рядом с каждым блоком выводится текущее число транзактов в нем, которое обновляется в процессе моделирования. Нажав клавиши Alt+N, вы можете заменить эту информацию на общее число транзактов, прошедших через каждый блок. Блок, в котором находится активный транзакт, выделен повышенной яркостью (на цветных мониторах - другим цветом).

Нажав клавиши Alt+F, вы можете наблюдать окно устройств (FACILITIES), в котором наглядно отображена информация о текущем состоянии каждого устройства модели: его использовании, занятости, очереди к нему.

Аналогичную информацию о многоканальных устройствах можно получить, нажав Alt+S и открыв окно памятей (STORAGES).

Если в модели используются статистические таблицы, то, нажав клавиши Alt+T, вы откроете окно таблиц (TABLES) с гистограммой распределения соответствующего атрибута модели, обновляющейся в процессе моделирования. Над гистограммой выводятся также текущие значения среднего и среднеквадратического отклонения табулируемого атрибута.

Если в модели используются матрицы, то, нажав клавиши Alt+M, вы откроете окно матриц (MATRICES), в котором можно наблюдать обновляющиеся в процессе моделирования значения элементов матриц.

Находясь в любом из перечисленных окон, вы можете путем нажатия клавиш Alt+L включить трассировку активного транзакта. При этом в верхней части окна появляется строка, содержащая информацию о текущем модельном времени, номере активного транзакта и его продвижении через блок-схему модели. Отключить трассировку можно повторным нажатием этих же клавиш.

Перемещение внутри окна любого типа к тому или иному объекту этого типа осуществляется путем нажатия клавиш управления курсором PgUp, PgDn и End. Возвращение в окно данных производится путем нажатия клавиш Alt+D.

Следует заметить, что наблюдение графических окон и особенно строки трассировки существенно замедляет моделирование, и при длинных прогонах моделей этой возможностью не следует злоупотреблять.

Открытие того или иного окна может быть выполнено также с помощью команды WINDOW (окно), в поле A которой указывается имя окна, однако удобнее это делать так, как описано выше.

Кроме графических окон внутри любого из них, кроме окна данных, может быть открыто до четырех микроокон. Микроокна открываются и закрываются командой MICROWINDOW (микроокно), имеющей следующий формат:

```
MICROWINDOW    A,B,C    ; комментарий
```

В поле A указывается номер микроокна - константа 1, 2, 3 или 4. Поле B содержит наблюдаемую величину - любой СЧА модели. Поле C определяет состояние микроокна в результате выполнения команды: ON - открыто, OFF - закрыто. Если поле C пусто, то по умолчанию команда открывает заданное микроокно. В поле комментария может быть задано название микроокна длиной до восьми символов.

При открытии любого окна заданные микроокна с обновляющейся в процессе моделирования информацией появляются в правой части соответствующего окна. Микроокно имеет форму прямоугольника с названием над рамкой, если оно было задано в комментарии к команде

MICROWINDOW.

В процессе моделирования можно также наблюдать одновременно до двух графиков зависимостей любых СЧА модели от модельного времени. Для этого необходимо до запуска модели ввести одну или две команды PLOT (начертить), имеющие следующий формат:

```
PLOT    A,B,C,D    ; комментарий
```

В поле A указывается аргумент зависимости - любой СЧА модели. Поле B должно содержать максимальное значение этого СЧА, определяющее масштаб изображения по оси Y. Операнд B задается константой, значение которой должно быть не менее 13. Поля C и D определяют начальное и конечное значения модельного времени, определяющие масштаб изображения по оси X. Эти операнды также задаются константами. В поле комментария может быть задан заголовок графика длиной до 34 символов.

График обновляется при каждом изменении модельного времени, если оно попадает в диапазон, заданный операндами C и D. Указанный в поле A СЧА-аргумент вычисляется относительно первого транзакта, обрабатываемого после изменения модельного времени.

Процесс моделирования можно прервать, нажав одну из клавиш Esc или Home. При этом в строке состояния командного окна появляется сообщение о номере активного транзакта, обрабатываемого симулятором в момент прерывания. Вы можете узнать значения интересующих вас стандартных числовых атрибутов модели в момент прерывания, введя команду SHOW (показать), операндом которой служат отдельные СЧА или выражения из них. Значение заданного в команде СЧА или выражения выводится в окно данных или другое активное окно. Введя команду EVENTS (события), можно увидеть в окне данных содержимое списков текущих и будущих событий. Команда USERCHAINS (списки пользователя) позволяет просматривать в окне данных содержимое списков пользователя. Обе последние команды не имеют операндов.

Инициировать прерывание моделирования можно также с помощью команды STOP (остановить), имеющей следующий формат:

```
STOP    A,B,C
```

В поле A указывается номер транзакта, вызывающего прерывание, задаваемый константой. Если это поле пусто, то прерывание вызывается любым транзактом. В поле B задается имя или номер блока, при входе в который происходит прерывание. Если этот операнд опущен, то прерывание происходит при входе в любой блок. В поле C указывается ON для установки условия прерывания и OFF для снятия этого условия (по умолчанию ON).

Например, команда

```
STOP    100,MET1
```

устанавливает условие прерывания моделирования при входе транзакта с номером 100 в блок с именем MET1. Команда

```
STOP    2
```

будет вызывать прерывание при каждом продвижении транзакта с номером 2, а команда

```
STOP    ,CHAIR
```

при каждом входе любого транзакта в блок с именем CHAIR. Наконец, команда

```
STOP
```

без операндов будет вызывать прерывание при каждом продвижении любого транзакта, а команда

```
STOP    ,,OFF
```

снимает все условия прерывания, установленные ранее другими командами STOP.

Прервав моделирование, можно также воспользоваться командой STEP (выполнить шаг) для пошагового выполнения модели с целью ее отладки. Операнд в поле A команды задает количество входов активного транзакта в блоки, которое производится при каждом выполнении команды. Обычно этот операнд равен 1, и каждое выполнение команды STEP приводит к продвижению активного транзакта к следующему блоку. Отладку с использованием команды STEP удобно проводить, находясь в окне блоков.

Для продолжения моделирования после прерывания следует ввести в командную строку команду CONTINUE (продолжить).

Команды STEP и CONTINUE могут не только вводиться в командную строку с клавиатуры, но и выбираться из меню команд, появляющемся в командном окне при активизации любого графического окна. Выбор производится подводом крестообразного курсора в прямоугольную область нужной команды и нажатием клавиши Ins. В окне блоков меню команд предоставляет также некоторые дополнительные возможности [8].

Команды STEP, CONTINUE, а также любые другие часто используемые команды удобно загрузить на функциональные клавиши F1-F10. Для этого после ввода загружаемой команды с клавиатуры необходимо нажать клавиши Ctrl+Fn, где n - номер выбранной функциональной клавиши. После загрузки команды на функциональную клавишу для ее выполнения достаточно нажатия этой клавиши.

5.6. Получение и интерпретация стандартного отчета

По завершении прогона модели раздается звуковой сигнал, и в строке состояния появляются сообщения

Writing REPORT.GPS Simulation Complete Reporting ... ,

сигнализирующие о том, что моделирование закончено и в данный момент производится создание отчета о прогоне модели. Затем система переходит в состояние ожидания дальнейших команд.

Отчет, создаваемый по завершении моделирования, записывается в файл со стандартным именем REPORT.GPS. Это имя может быть изменено командой REPORT (создать отчет), имеющей следующий формат:

REPORT A,B

В поле A указывается спецификация файла, в который должен быть выведен отчет. Если поле B содержит ключевое слово NOW, то отчет создается немедленно после ввода команды.

Необходимо иметь ввиду, что отчет, создаваемый автоматически по завершении прогона модели или командой REPORT, является 1неформатированным, т.е. непригодным для непосредственного просмотра. Для форматирования и создания стандартного отчета GPSS/PC необходимо завершить сеанс и выполнить программу форматирования отчета. Выход из интегрированной среды (завершение сеанса) производится путем ввода управляющего оператора END (закончить). При этом производится выход в MS DOS или в программу-оболочку Norton Commander.

Для форматирования отчета необходимо загрузить модуль форматирования GPSSREPT.EXE. После его загрузки на экране появляется "заставка" с названием модуля, двумя окнами в нижней части экрана и сообщениями-подсказками. В левом окне выведено имя файла, в котором находится неформатированный отчет (по умолчанию это файл REPORT.GPS). В правом окне выведено обозначение устройства, куда должен быть выведен форматированный отчет (по умолчанию это экран дисплея SCRN:). Форматированный отчет может быть также выведен на печать или на диск. Для этого в правое окно надо ввести обозначение PRN: или имя файла на диске соответственно. Для переключения окон используется клавиша Enter. Для создания отчета на выбранном устройстве следует нажать клавишу Пробел, для выхода из программы - клавишу Esc.

Если содержимое окон по умолчанию оставлено без изменения, то после нажатия клавиши Пробел на экране появляется отчет о последнем прогоне модели, выполненном перед завершением сеанса работы с модулем GPSSPC.EXE. Отчет содержит следующую информацию:

1) общие сведения о модели и ее прогоне, включающие модельное время начала (START_TIME) и конца (END_TIME) прогона, количество блоков в модели (BLOCKS), количество устройств (FACILITIES), количество многоканальных устройств (STORAGES), объем памяти, остававшейся свободной при прогоне модели (FREE_MEMORY);

2) сведения об именах объектов модели, включающие для каждого имени идентификатор (NAME), присвоенное ему числовое значение (VALUE) и тип имени: 0, если числовое значение имени присвоено пользователем с помощью оператора EQU; 1, если числовое значение

имени присвоено системой; 2, если имя является именем блока;

3) сведения о блоках модели, включающие для каждого блока номер строки исходной программы (LINE), номер или имя блока (LOC), название блока (BLOCK_TYPE), количество транзактов, прошедших через блок (ENTRY_COUNT), текущее количество транзактов в блоке в момент завершения моделирования (CURRENT_COUNT), количество транзактов, заблокированных перед блоком в момент завершения моделирования (RETRY);

4) сведения об устройствах модели, включающие для каждого устройства его имя или номер (FACILITY), количество занятий устройства (ENTRIES), коэффициент использования (UTIL.), среднее время на одно занятие (AVE._TIME) и ряд других данных;

5) сведения о многоканальных устройствах модели, включающие для каждого МКУ его имя или номер (STORAGE), емкость (CAP.), количество свободных каналов в момент завершения моделирования (REMAIN.), наименьшее (MIN.) и наибольшее (MAX.) количество занятых каналов в процессе моделирования, количество занятий МКУ (ENTRIES), среднее количество занятых каналов (AVE.C.), коэффициент использования (UTIL.) и ряд других данных;

6) сведения об очередях модели, включающие для каждой очереди ее имя или номер (QUEUE), максимальную длину очереди в процессе моделирования (MAX.), текущую длину очереди в момент завершения моделирования (CONT.), общее количество транзактов, вошедших в очередь в процессе моделирования (ENTRIES), и количество "нулевых" входов в очередь (ENTRIES(0)), среднюю длину очереди (AVE.CONT.), среднее время ожидания в очереди с учетом всех транзактов (AVE.TIME) и без учета "нулевых" входов (AVE.(-0));

7) сведения о статистических таблицах модели, включающие для каждой таблицы ее имя или номер (TABLE), среднее значение (MEAN) и среднеквадратическое отклонение (STD.DEV.) табулируемой величины, границы частотных интервалов (RANGE), частоты (FREQUENCY) и накопленные частоты в процентах (CUM.%) попадания наблюдений в эти интервалы;

8) сведения о списках пользователя модели, включающие для каждого списка его имя или номер (USER_CHAIN), количество транзактов в списке в момент завершения моделирования (CHAIN_SIZE), среднее количество транзактов в списке (AVE.CONT), общее количество транзактов, вошедших в список в процессе моделирования (ENTRIES), максимальное количество транзактов, находившихся в списке (MAX), среднее время пребывания транзакта в списке (AVE.TIME);

9) сведения о логических переключателях модели, включающие для каждого ЛП его имя или номер (LOGICSWITCH) и состояние ЛП в момент завершения моделирования: 1 - "включен", 0 - "выключен";

10) сведения о сохраняемых величинах модели, включающие для каждой сохраняемой величины ее имя или номер (SAVEVALUE) и значение в момент завершения моделирования (VALUE);

11) сведения о матрицах модели, включающие для каждой матрицы ее имя или номер (MATRIX), а также список всех элементов матрицы в формате: "строка" (ROW), "столбец" (COLUMN), "значение" (VALUE).

Если в операторе START задан вывод в отчет списков текущих и будущих событий, то отчет включает в себя также сведения о транзактах, находившихся в момент завершения моделирования в этих списках. Сведения о транзактах размещаются в отчете в соответствии с размещением транзактов в каждом списке.

Информация о списке текущих событий включает в себя для каждого транзакта его номер (XACT_NUMBER), приоритет (PRI), резидентное время транзакта (M1), номер текущего блока (CURRENT), номер следующего блока (NEXT), а также перечень всех параметров транзакта в формате: "параметр" (PARAMETER), "значение" (VALUE).

Информация о списке будущих событий включает для каждого транзакта те же данные, однако вместо резидентного времени транзакта (M1) выводится запланированное время выхода транзакта из списка будущих событий (BDT).

Разумеется, сведения об объектах того или иного типа появля-

ются в отчете только в том случае, если в модели присутствует хотя бы один объект данного типа. Кроме того, включением в отчет сведений об объектах разных типов можно управлять с помощью так называемого установочного файла SETTINGS.GPS [8]. В отчетах о прогоне моделей, включающих в себя другие, не рассматривавшиеся здесь объекты GPSS/PC, появляется соответствующая информация и об этих объектах.

На рис. 24 приведен отчет о прогоне модели примера на рис. 21.

```
START_TIME   END_TIME  BLOCKS    FACILITIES  STORAGES   FREE_MEMORY
       0       14617     12           0          1        274320

LINE   LOC     BLOCK_TYPE     ENTRY_COUNT   CURRENT_COUNT   RETRY
 80     1       GENERATE           150             0          0
 90     2       ASSIGN             150             0          0
 100    3       PRIORITY           150             0          0
 110    4       QUEUE              150             0          0
 120    5       QUEUE              150             0          0
 130    6       ENTER              150             0          0
 140    7       DEPART             150             0          0
 150    8       DEPART             150             0          0
 160    9       ADVANCE            150             0          0
 170    10      LEAVE              150             0          0
 180    11      TABULATE           150             0          0
 190    12      TERMINATE          150             0          0

QUEUE   MAX  CONT. ENTRIES ENTRIES(0) AVE.CONT. AVE.TIME  AVE.(-0)
  1       1     0      54        48     0.02        6.07     54.67
  2       1     0      42        35     0.01        4.14     24.86
  3       1     0      54        49     0.02        6.22     67.20
 LINE     2     0     150       132     0.06        5.59     46.56

STORAGE        CAP. REMAIN. MIN.  MAX.  ENTRIES AVL.  AVE.C. UTIL.
 STO2             2     2     0     2      150   1     0.66  0.328

TABLE     MEAN    STD.DEV. RETRY  RANGE           FREQUENCY  CUM.%
 WTIME     5.59     25.23    0
                                -        50         144      96.00
                             50 -       100           3      98.00
                            100 -       150           1      98.67
                            150 -       200           2     100.00

 TTIME    69.48     70.88    0
                                -       100         117      78.00
                            100 -       200          23      93.33
                            200 -       300           8      98.67
                            300 -       400           2     100.00
```

Рис. 24

Отчет выводится на экран постранично. Для вывода очередной страницы необходимо нажать клавишу Пробел, для прекращения вывода отчета - клавишу Esc. По окончании вывода отчета на экране появляется сообщение

[SPACE] for another report Any other key to end

Для создания отчета на другом устройстве или другого отчета надо нажать клавишу Пробел, для выхода из программы GPSSREPT - любую другую клавишу.

Помимо отчета отдельные результаты моделирования могут быть также выведены в базу данных GPSS/PC [8] с помощью команд RESULT. Однофакторный дисперсионный анализ и получение доверительных интервалов для выведенных в базу данных характеристик модели могут быть выполнены с помощью команды ANOVA. Рассмотрение этих команд выходит за рамки данного издания.

СПИСОК ЛИТЕРАТУРЫ

1. Шакин В.Н., Воробейчиков Л.А., Шибанов С.Е., Семенова Т.И. Моделирование систем и сетей связи: Учебное пособие/МИС.- М., 1988.

2. Игельник Б.М., Лившиц В.М., Шибанов С.Е. Аналитическое моделирование систем связи: Учебное пособие/МИС. - М., 1989.

3. Шакин В.Н., Лившиц В.М. Принципы построения локальных сетей и анализ их характеристик: Учебное пособие для слушателей ФПКП/ МИС. - М., 1990.

4. Методические указания по использованию средств имитационного моделирования систем и сетей связи для слушателей ФПКП/ Л.А.Воробейчиков, В.Н.Шакин, С.Е.Шибанов/МИС. - М., 1990.

5. Шеннон Р. Имитационное моделирование систем - искусство и наука: Пер. с англ. - М.: Мир, 1978.

6. Максимей И.В. Имитационное моделирование на ЭВМ. - М.: Радио и связь, 1988.

7. Шрайбер Т.Дж. Моделирование на GPSS: Пер. с англ. - М.: Машиностроение, 1980.

8. GPSS/PC general purpose simulation. Reference Manual. - Minuteman software. P.O. Box 171. Stow, Massachusetts 01775, 1986.