МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ
ВОСТОЧНОУКРАИНСКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ
имени Владимира Даля

А.Г. КОРОЛЕВ

МОДЕЛИРОВАНИЕ СИСТЕМ НА ЯЗЫКЕ GPSS WORLD

ПРАКТИЧЕСКИЙ ПОДХОД В ПРИМЕРАХ И ЗАДАЧАХ.

Издательство ВНУ
Луганск – 2004

УДК 62. 529

А.Г. Королёв Моделирование систем на языке GPSS WORLD. Практический подход в примерах и задачах. Учебное пособие. Луганск: Изд-во Восточноукр. нац. ун-та, 2004.-143 с. Библиогр.14

ISBN

Рассматриваются методы и техника моделирования систем массового обслуживания, основанные на дискретно- событийном подходе. Изложение ведется на основе языка GPSS World. Изложение сопровождается большим количеством примеров.

Учебное пособие предназначено для студентов старших курсов по специальностям «Компьютерная инженерия», «Компьютерные системы и сети», «АСУ ТП», а также может быть использовано специалистами по технологиям обучения и инженерно-техническими работниками проектных организаций..

Рецензенты:

д-р техн. наук, проф. В.А. Святный (Донецкий национальный технический университет),
д-р техн. наук, проф. И.И.Стенцель (Восточноукраинский национальный университет).

Печатается по решению Ученого совета Северодонецкого технологического института и редакционной коллегии «Инженерия» Восточноукраинского национального университета им. В.Даля Министерства образования и науки Украины

Редактор: В.Я Сидоренко

ISBN

1. Введение.

Имитационное моделирование, как и моделирование вообще – мощное средство для изучения сложных систем. [1..14]. Для имитационного моделирования используется целый спектр языков, однако наибольшую популярность среди специалистов приобрел язык GPSS. Он подробно описан, например, в [4,5,8] . Но наибольшей популярностью до сих пор пользуется книга Шрайбера Т. Дж. Моделирование на GPSS [14], которая также известна, как «красная книга». Эта книга является фундаментальным самоучителем по языку GPSS и прекрасно описывает как проблематику задач моделирования систем массового обслуживания, так и методы их решения с помощью данного языка.

В последнее время, в связи с появлением новой версии языка GPSS, GPSS World, возникла настоятельная потребность в подготовке нового учебного пособия, описывающего особенности этой версии, и ее среды, обеспечивающей разработку моделей на GPSS. В этой книге собран многолетний опыт преподавания языка GPSS студентам. Она содержит большое количество примеров, иллюстрирующих особенности работы со средой языка, а также особенности применения различных блоков новой версии для решения конкретных задач моделирования.

Данное пособие предназначено как для студентов, изучающих компьютерное моделирование, так и для специалистов, использующих моделирование в практической работе.

Язык GPSS имеет довольно длинную историю, и пережил многие другие языки, которые разрабатывались для общения специалистов с ЭВМ. В первую очередь, упомянем, что этот язык предназначен для моделирования систем массового обслуживания, и, несмотря на многочисленные попытки сделать лучшие языки, выжил. Он прошел ряд процессов модернизации, и посейчас является наиболее популярным языком моделирования систем.

Концептуально, модель, положенная в основу языка, представляет собой следующее.

Имеется система, состоящая из одноканальных и многоканальных устройств, которые осуществляют обслуживание заявок (транзакций). Заявки перемещаются по системе, занимая эти устройства, как правило, они задерживаются в устройствах на обслуживание. Траектория движения заявок по системе может быть разной, в зависимости от вида заявок, и от значений случайных и расчетных параметров, которые рассчитываются заявками в процессе их движения по системе. В принципе, заявки вводятся в систему специальными блоками через заданные промежутки времени, которые могут быть и случайными. Заявки рано или поздно должны быть удалены из системы.

Этот язык имеет все основные вычислительные возможности традиционных языков программирования, но в отличие от них, вычисления идут не в порядке расположения операторов, а в том порядке, в котором заявки движутся через блоки языка. Все операции происходят в так называемом модельном времени, и язык хорошо приспособлен для моделирования процессов, проходящих одновременно. Заявки конкурируют между собой за ресурсы

системы, и в первую очередь за устройства. Поэтому в системе возникают очереди, из заявок, претендующих на один и тот же ресурс.

Если оценивать язык GPSS в целом, то можно сказать, что GPSS - это больше, чем язык программирования для имитационного моделирования систем массового обслуживания. Это также необычное явление в области программирования. Он появился в декабре 1961 года и был одним из самых удачных на то время проблемно-ориентированных языков.

Основой для создания GPSS послужил дискретно-событийный подход, разработанный Джеффри Гордоном. В своё время, GPSS произвёл настоящую революцию в области исследования систем массового обслуживания. Специалист в этой области, мог провести исследование работы сложнейшей системы в течение многих лет её реального функционирования, за считанные минуты и часы работы модели. Это было настолько быстро и необычно, что у специалистов в этой области как бы выросли крылья. И даже некоторая жесткость конструкций и недостаточная гибкость языка, не меняли общего восторженного отношения к GPSS .

Собственно систему общецелевого моделирования (General Purpose Systems Simulator) впервые создал Джеффри Гордон (Geoffrey Gordon) в 1961 году. Тогда она была реализована на больших ЭВМ . В первой версии было сравнительно немного блоков, всего чуть больше 20.Уже в первой версии GPSS было два типа обслуживающих аппаратов, «Устройства», обслуживающие одну заявку в данный момент времени и «Памяти» (STORE) (переименованный в следующих версиях на STORAGE), который мог обслуживать несколько заявок одновременно. Устройства могли быть заняты и освобождены блоками SEIZE и RELEASE, а памяти - соответственно блоками ENTER и LEAVE. Дальнейшее развитие системы шло за счёт расширения номенклатуры блоков, и за счет удаления тех блоков, которые оказывались ненужными в новых версиях.

Ситуация более или менее стабилизировалась к моменту появления версии GPSS III. Эта версия системы была уже достаточно близкой к системе GPSS World 2000. После этого, GPSS стал более языком моделирования, чем системным имитатором. Последующие версии GPSS развивались вначале на базе ЭВМ типа IBM/360 и ЕС ЭВМ, а затем, с появлением в начале 80-х годов персональных. компьютеров, новые версии стали создаваться и для них. Здесь ключевым этапом оказалось появление в 1984 году новой версии GPSS - GPSS /PC, разработанной фирмой Minuteman Software под руководством С. Кокса. Система GPSS /PC – это в основном не компилятор, а интерпретатор. Эта система работала под управлением MS-DOS, и когда возможности эксплуатации MS-DOS оказались исчерпанными, в 2000 году фирмой Minuteman Software была выпущена версия GPSS World 2000.

2. Достоинства GPSS World в сравнении с другими языками моделирования.

Новый продукт, GPSS World 2000 совместим сверху вниз с GPSS /PC, за исключением анимации. В GPSS World 2000 введено много новых функций. Введение новых 9 типов блоков увеличило их общее число до 53. Новый блок

INTEGRATION это средство для облегчения моделирования непрерывных и гибридных систем. Другие новые блоки позволяют гибко управлять файлами.

GPSS World имеет существенно отличающийся внешний вид по сравнению с GPSS /PC. Он имеет полноэкранный редактор традиционного для Windows типа. Для просмотра результатов используется свыше 20 стандартных окон. Хотя эта система недостаточно использует графику для представления движения заявок по блок диаграмме, однако она более удобна для отладки. Приложение также имеет новый более быстрый транслятор. СЧА могут принимать значения с плавающей точкой, в то время как в GPSS /PC были допустимы только целые значения.

GPSSWorld включает PLUS - язык программирования нижнего уровня для моделирования. Моделирование с использованием PLUS выражений может быть включено почти везде в GPSS World программы, в любом блоке или процедуре вызова, и таким образом, увеличивается мощность программ. Язык PLUS позволяет программно управлять не только вычислениями, но и размещением результатов. Система GPSS World разрешает многозадачность, позволяя нескольким имитационным процессам выполняться одновременно.

GPSS World является языком, проблемно ориентированным на имитационное моделирование систем массового обслуживания. Представление жизни модели как движения во времени заявок, перемещающихся в модели и обслуживающихся в устройствах очень естественно для основной части задач имитационного моделирования. Язык GPSS World , в своем последнем виде, очень легок для изучения. Студенты после короткого времени обучения могут создавать достаточно сложные модели. Автоматический сбор статистики - это огромная помощь для начинающих. Для многих реальных систем, моделирование на GPSS World гораздо легче, чем другими методами. Компактные программы и возможность использования графического интерфейса позволяют ускорить создание прототипов моделей. При этом на каждую из них можно получить быстрый отклик после улучшений проведенных пользователем. Этот последний фактор стал причиной широкого использования имитационного моделирования на практике.

3.Работа с системой GPSS World.

Приложение GPSS World имеет оболочку, достаточно традиционную для приложений WINDOWS. Основные возможности по управлению его работой предоставляет главное меню и инструментальные кнопки. Главное меню содержит следующие подменю.

File Edit Search View Command Window Help.

Подменю File .

Это подменю содержит следующие пункты.

New

создать новый объект. Им может быть модель или текстовый файл.

Open

открыть существующий объект, сохраненный на диске. Им может быть модель, компилированная модель, отчет, текстовый файл, WEB страничка, и произвольный файл.

Close
закрыть окно объекта
Save
сохранить новую версию объекта в файле на диске.
Save As
сохранить объект в файле на диске, изменив имя и, возможно, папку .
Print
вызов диалогового окна печати.
Print Setup
вызов диалогового окна настройки принтера.
Internet
вызов связи с разработчиком системы GPSS – World по интернету.
Exit
выход из приложения.

Эти пункты достаточно традиционны для оконных приложений, и вряд ли нуждаются в особых пояснениях.

Подменю Edit.

Это подменю также достаточно стандартно и содержит следующие пункты.

Undo
отмена правки.
Cut
вырезать выделенный фрагмент текста в буфер.
Copy
скопировать выделенный фрагмент текста в буфер.
Paste
вставить содержимое буфера.
Insert Line
вставить строку.
Delete Line
удалить строку.
Font
вызов диалога изменения шрифта.
Insert Block
вставить блок GPSS World. Этот пункт даёт доступ к диалоговому окну, позволяющему сформировать любой блок GPSS World в диалоге.
SettingsCommand
вызвать диалоговое окно настройки системы.

Некоторые другие пункты, этого подменю не рассматриваются, так как они предназначены для достаточно подготовленных пользователей системы.

Подменю Search

Содержит следующие команды, типичные для оконных приложений.
Find/Replace
вызывает диалог поиска и замены подстрок.
Go to Line
обеспечивает переход к строке с заданным номером.
Next Bookmark
обеспечивает переход к следующей невидимой отметке.
Mark
позволяет отметить текущую позицию невидимой меткой.
Unmark
позволяет удалить невидимую отметку в текущей позиции.
Unmark All
позволяет убрать все отметки.
Select to Bookmark
позволяет выбрать текст от текущей позиции до следующей отметки.
Next Error
позволяет перейти к следующей строке с ошибкой.
Previous Error
позволяет перейти к предыдущей позиции с ошибкой.

Подменю View
Содержит следующие команды.
Notices
выводит общие сведения о системе
Toolbar
отображает и скрывает панель инструментов.
Simulation clock
включает и выключает отображение системных часов в ходе моделирования. Этот пункт доступен только после компиляции модели.

Подменю Command.
Содержит следующие команды.
Create Simulation
создает компилированную модель путем трансляции текста модели.
Retranslate
повторяет трансляцию без создания новой компилированной модели (она создается в том же файле).
Repeat Last Command
повторяет последнюю успешно выполненную команду, посланную модели.
START
позволяет задать команду старт для начала моделирования.
STEP 1
обеспечивает пошаговое выполнение моделирования.
HALT
принудительно останавливает процесс моделирования.

CONTINUE
продолжает процесс моделирования после останова.
CLEAR
сбрасывает модель в начальное состояние для нового запуска процесса моделирования.
RESET
очищает модель от статистической информации.
SHOW
вызывает окно, позволяющее узнать значение любой метки или стандартного числового атрибута в нижней строке главного окна, и в окне компиляции.
Custom
вызывает окно, позволяющее ввести любую допустимую команду GPSS World.

Подменю Window .
Cascade
располагает окна каскадом.
Tile
располагает окна мозаикой.
Simulation Snapshot
предъявляет информацию о кадре процесса моделирования.
Simulation Window
предъявляет статистическую информацию по блокам, выражениям, одноканальным устройствам, логическим ключам, матрицам, очередям, ячейкам, многоканальным устройствам, таблицам, а также окно для вывода графика изменения параметров во времени. Позволяет следить за этой информацией в ходе моделирования.
Здесь же можно вызвать на передний план любое окно, открытое в приложении.

Подменю Help.
Help Topics
дает доступ к справочной системе по темам.
About
дает информацию о системе.

4. Общие сведения о языке GPSS World.

GPSS World – это высоко интегрированная компьютерная среда моделирования общего назначения, разработанная для профессионалов моделирования. Это - мощный инструмент моделирования, покрывающий, и дискретное и непрерывное компьютерное моделирование, с чрезвычайно высоким уровнем взаимодействия и визуализации. При использовании GPSS World, возможно предсказать результаты проектных решений по чрезвычайно сложным реальным системам.

GPSS World обеспечивает предсказание поведения сложных реальных систем.

Многие дорогостоящие проекты в прошлом потерпели неудачу, потому что конечный результат не был предсказан достаточно точно. Определение максимальной пропускной способности системы, ее стоимости, и многого другого, всё это необходимо детально знать о системе при её разработке, причём как можно раньше. Хотя хорошие математические модели чрезвычайно ценны, и они должны использоваться там, где это возможно, сложность реальных систем требует использования компьютерного моделирования. В этих случаях и необходима система GPSS World .

GPSS World основан на языке компьютерного моделирования, GPSS . Этот язык был разработан Джеффри Гордоном в IBM в 1960 году. Именно он ввел наиболее важные концептуальные понятия, которые важны для любого коммерческого языка моделирования событий.

Эта версия, GPSS World, - прямой потомок GPSS /PC, ранней реализации GPSS для персональных компьютеров. Введение в 1984 году, GPSS /PC, сохранило тысячам пользователей миллионы долларов. Теперь, реализация GPSS World расширяет их возможности .

Язык GPSS World разработан так, чтобы давать ответы быстро и надежно, с минимумом усилий. Прозрачность языка ценна по трем причинам.

Очень опасно полагаться на непрозрачное моделирование "В черном ящике", внутренние механизмы которого нельзя наблюдать. Трудно быть уверенным, что модель работает именно так, как должно.

Во вторых, удачные модели весьма ценны и имеют удивительно длительный срок жизни. Возможно, что новым работникам понадобится ознакомиться с ранее разработанными моделями. Это почти невозможная задача, если средство моделирования не было сделано сразу с высоким уровнем прозрачности.

Третьей, одной из наиболее важных причин, является понимание поведении системы когда, опытный профессионал может видеть внутреннюю динамику работы модели.

GPSS World был разработан, чтобы обеспечить прозрачность, позволяющую видеть внутренние механизмы моделей и зафиксировать результат. Взаимодействие модели с пользователем позволяет ему не только провести исследование, но и обеспечивает управление моделями. Встроенные средства анализа данных могут вычислять доверительные интервалы и выполнять дисперсионный анализ. Так что теперь, можно проводить эксперименты и оптимизацию автоматически, с относительно небольшими усилиями .

Моделирование большинства систем требует знания только малого подмножества блоков и команд.

Однако для сложных систем, нужно знакомства со всем тем, что может предложить GPSS World . Это пособие должно быть интересно, не только для студентов, но и для квалифицированных пользователей системы GPSS.

Язык GPSS World является объектно-ориентированным. Люди создают модели объектов, в первую очередь для того, чтобы создать сами объекты.

Проведение моделирования требуют выполнения нескольких шагов. Эти шаги обычно включают:

формирование модели и совокупность данных;
тестирование и проверку;
собственно моделирование;
экспериментирование;
анализ результатов.

В GPSS World, вы создадите и измените модель, с помощью полноэкранного текстового редактора. Если вы хотите, то можете вставлять инструкции блока GPSS World, используя специальный диалог, где вы только заполняете поля. Возникающие в результате блоки помещаются последовательно в позицию ввода в вашей модели. Вы затем создаете Объект моделирования, выбирая Command./Create Simulation в основном меню окна. После этого, вы можете использовать мощный набор команд для того, чтобы управлять ходом моделирования. Вы можете вводить команды в интерактивном режиме, или включать их в первоначальную модель. В ходе тестирования и проверки, доступно большое количество типов окон для интерактивного просмотра .

Язык моделирования GPSS World был значительно расширен за счёт специального языка PLUS (Programming Language Under Simulation) .

Этот простой, но мощный язык программирования снимает ограничения, которые существовали в прежних версиях GPSS. Данные внутри этой среды (GPSS World и PLUS) используются без контроля типов, с автоматическим преобразованием, когда это необходимо. Кроме того, многочисленные функции и библиотеки распределений вероятностей непосредственно доступны в PLUS выражениях. Написанные пользователем PLUS процедуры можно использовать наравне с библиотечными процедурами. Команда включения может вводить существующие библиотеки процедур пользователей, содержащие проверенные PLUS процедуры для использования в течение моделирования.

К определяемой пользователем PLUS процедуре можно обращаться где угодно в модели. Выражения, определенные в PLUS, могут включать элементы данных и Стандартные Числовые Атрибуты. Когда необходимо, PLUS Выражения может появляться и вне PLUS процедур, то есть в инструкциях GPSS .

Фактически, большинство параметров в GPSS - инструкциях могут теперь быть и в форме выражений.

В Версии 4 GPSS World, язык PLUS был расширен. Теперь он допускает описания экспериментов. Это мощное свойство разрешает программируемое управление ходом моделирования, и даже может быть основано на результатах моделирования. Таким образом, полностью автоматизируется операция исследования поверхностей поведения системы. Эти эксперименты могут быть созданы с минимальным участием с вашей стороны. GPSS World теперь включает два новых мощных генератора экспериментов, к которым можно обращаться через меню редактирования основного окна. Генератор экспериментов создаст эксперимент под вашим управлением, его PLUS исходный текст можно вставить в вашу модель. Точно так же генератор экспериментов с оптимизаций вставит в вашу модель сложный эксперимент исследования поверхности решения, который сам ищет минимум, или максимум и сообщает его вам.

GPSS World имеет многообразные дискретные и непрерывные возможности для моделирования.

Типы данных теперь включают целые и вещественные числовые значения, а также строки. Каждый тип - автоматически преобразуется в требуемую форму, если это необходимо. Матрицы были улучшены. Они могут теперь включать до 6 размерностей.

Когда вы создаете компилированную модель, то если возникли ошибки, по ним можно перемещаться последовательно, используя меню.

Очень просто и визуализировать управление моделированием. GPSS World может создавать стилизованные мультипликации для любого объекта. Соответствующие окна работают динамически – и они показывают изменения значений параметров модели.

Модели теперь могут связываться со внешним миром, используя текстовые файлы. Вы можете использовать их, чтобы обратиться к файлам данных, создавать файлы результата и заказные отчеты, обращаться к внутренним данным непосредственно. GPSS World может даже связываться с другими программами прямым обращением. PLUS - библиотека содержит обращение, Call_Integer (), Call_String (), и Call_Real (). Эти процедуры могут вызывать внешние функции, существующие в вашей системе в исполняемых файлах (то есть стало возможным использовать EXE и DLL файлы в ваших моделях).

5. Введение в предметную область языка GPSS.

Этот язык предназначен для изучения поведения систем массового обслуживания, в которых происходит конкуренция людей или заданий на обработку, за ограниченные ресурсы. И в этой связи, люди или задания выстраиваются в очереди, претендуя на обслуживание.

Простейшим примером системы массового обслуживания является система с одним устройством и очередью.

Рассмотрим систему, состоящую из одного человека, выполняющего обслуживание. Это может быть кассир, кладовщик, парикмахер и т.п. Клиенты приходят к такому обслуживающему устройству в случайные моменты времени, ждут очереди на обслуживание, а затем - обслуживаются. Как и в жизни, обслуживание идет по принципу: кто первым пришел, тот первым обслужен. Будем считать, что мы собираемся моделировать работу кассира.

Тогда наша система характеризуется двумя независимыми величинами: интервалом прихода клиентов и интервалом их обслуживания.

После их задания, можно исследовать поведение такой системы, и в частности узнать:

число клиентов, пришедших в течение заданного промежутка времени;
количество клиентов, сразу же попавших на обслуживание;
среднее время, проведенное клиентом в очереди;
среднюю длину очереди;
среднее время, затраченное клиентом на обслуживание;
максимальную длину очереди за всё время наблюдения;
степень занятости нашего кассира .

Можно получить и много другой информации о поведении такой системы, причем с минимальными усилиями.

Вначале приведем текст такой модели с минимальными пояснениями, а затем рассмотрим процесс моделирования для этого примера подробнее.

Пусть интервал прихода клиентов подчиняется экспоненциальному закону распределения, и в среднем равен 120 секундам. Этот закон распределения хорошо описывает совершенно не организованный поток клиентов, и на практике, многие реальные законы распределения близки к экспоненциальному закону.

Пусть на обслуживание клиента кассир тратит от 80 до 130 секунд, причем любое время на обслуживание является одинаково вероятным.

Мы будем считать, что нам нужно промоделировать работу системы при обслуживании 1000 клиентов.

Чтобы создать такую модель и провести моделирование, необходимо выполнить следующие действия.

Запустить программу GPSS World.

Выбрать в главном меню пункты File/New. В появившемся окне выбрать вариант Model и щелкнуть по OK.

Будет создано окно для ввода текста модели.

В этом окне следует набрать следующий код.

```
GENERATE        (exponential(1,0,120))
QUEUE  kassa
SEIZE    KASSA
DEPART        kassa
ADVANCE       105,25
RELEASE       kassa
TERMINATE     1
```

Это и будет собственно модель.

Затем нужно выбрать пункты Command/Create Simulation . Будет создана компилированная модель (моделирование) и результат компиляции отобразится в новом окне примерно в виде следующего текста.

```
09/08/03 07:22:33  Model Translation Begun.
09/08/03 07:22:33  Ready.
```

Далее нужно выбрать пункты Command/Start, в диалоговом окне ввести число 1000 вместо 1 и щелкнуть по OK

Текст окна JOURNAL изменится и приобретёт примерно такой вид:

```
10/11/03 21:35:52  Model Translation Begun.
10/11/03 21:35:52  Ready.
10/11/03 21:36:01  START 1000
10/11/03 21:36:01  Simulation in Progress.
10/11/03 21:36:01  The Simulation has ended.  Clock is 117107.274626.
10/11/03 21:36:02  Reporting in Untitled Model 1.1.1  -  REPORT Window.
```

Одновременно, по результатам моделирования будет создано окно итоговой статистики (REPORT), где будет собрана вся стандартная статистика по работе модели. Она будет выглядеть примерно следующим образом.

```
                    GPSS World Simulation Report - Untitled Model 1.1.1


                  Tuesday, May 04, 2004 13:12:29

          START TIME           END TIME  BLOCKS  FACILITIES  STORAGES
               0.000         117107.275     7        1          0

             NAME                       VALUE
          KASSA                      10000.000


 LABEL              LOC  BLOCK TYPE     ENTRY COUNT CURRENT COUNT RETRY
                    1    GENERATE          1007             0       0
                    2    QUEUE             1007             6       0
                    3    SEIZE             1001             1       0
                    4    DEPART            1000             0       0
                    5    ADVANCE           1000             0       0
                    6    RELEASE           1000             0       0
                    7    TERMINATE         1000             0       0


FACILITY         ENTRIES  UTIL.   AVE. TIME AVAIL. OWNER PEND INTER RETRY DELAY
 KASSA             1001    0.890     104.155  1     1001    0    0     0      6


QUEUE              MAX CONT. ENTRY ENTRY(0) AVE.CONT. AVE.TIME   AVE.(-0) RETRY
 KASSA              14    7   1007    106     3.162    367.696    410.954   0


CEC XN   PRI          M1      ASSEM  CURRENT  NEXT  PARAMETER    VALUE
  1001    0      116536.227   1001      3      4


FEC XN   PRI          BDT      ASSEM  CURRENT  NEXT  PARAMETER    VALUE
  1008    0      117639.933   1008      0      1
```

Теперь мы можем обсудить как саму модель, так и полученные результаты.

GENERATE (exponential(1,0,120))

Эта строка обеспечивает ввод в модель заявок, которые имитируют приход клиентов. Распределение интервалов прихода определяется экспоненциальным законом со средним интервалом 120 единиц модельного времени.

QUEUE kassa

Эта строка моделирует поступление клиентов в очередь, с тем, чтобы можно было собрать статистику по очереди к кассе (kassa)

SEIZE kassa

Эта строка моделирует начало работы кассира с клиентом. Клиент может попасть на обслуживание к кассиру, только тогда, когда кассир обслужит клиентов пришедших раньше. Когда с клиентом началась работа, кассир считается занятым.

DEPARTkassa

Эта строка моделирует удаление клиента из очереди, так как он уже обслуживается у кассира. Если использовался блок QUEUE, то должен

использоваться и блок DEPART. Только тогда будет собрана правильная статистика по очереди.

ADVANCE 105,25

Эта строка моделирует задержку клиента на обслуживание у кассира. Средний интервал задержки равен 105 секундам модельного времени, а отклонение интервала от среднего составляет по 25 модельных секунд в обе стороны. Любое значение времени задержки из этого интервал абсолютно равновероятно.

RELEASE kassa

Эта строка моделирует завершение работы кассира с клиентом. После этого, кассир вновь становится свободным и может начать обслуживание следующего клиента.(если он есть)

TERMINATE 1

Эта строка моделирует уход клиента из системы. А число 1 в ней обеспечивает завершение моделирования после того, как будет обслужено то число клиентов, которое указано в Start (то есть 1000).

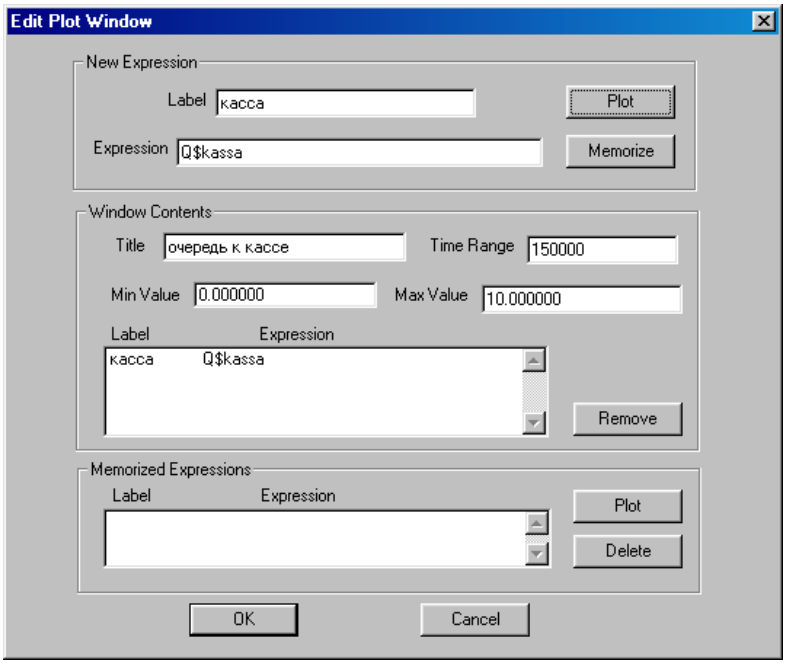
Анализируя выходную статистику, можно определить следующие результаты моделирования. На обслуживание 1000 клиентов системе понадобилось 117107.275 секунд модельного времени. Всего поступило на обслуживание 1007 клиентов. Кассир был занят на 89%. Среднее время обслуживания клиента составило 104.155 секунды. Максимальная величина очереди составила 14 клиентов. 106 клиентов вообще не стояли в очереди. Средняя величина очереди была 3.162. А среднее время, которое клиент провёл в очереди, равно 367.696 секунд.

Общий вывод – при такой скорости работы, система не очень хорошо справляется с работой, что может привести к потере клиентов. После нескольких экспериментов с различными временами обслуживания, нетрудно установить, что заменив блок ADVANCE на

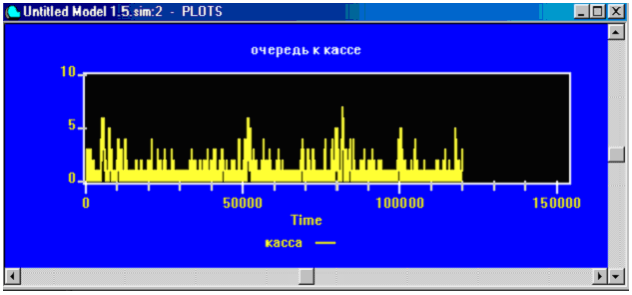
ADVANCE 80,25, мы получим следующие результаты.

Средняя занятость кассира - 0.678. Среднее время обслуживания - 79.609, Средняя длина очереди - 0.644. Среднее время в очереди- 75.504 что уже гораздо лучше.

Если есть желание посмотреть, как себя ведёт, например, очередь к кассе в процессе моделирования, то, вызвав пункт меню Window/Simulation Window/Plot Window , получим окно для задания вида графиков. В нём надо заполнить поля в New Expression и щёлкнуть по Plot. Затем в Window Contents заполнить поля, как указано в окне. Затем щелкнуть по OK.



Тогда после запуска модели по Start 1000, получим следующее окно, отображающее изменение очереди к кассиру во времени.



Не вдаваясь в подробный анализ полученных результатов, попробуем глубже вникнуть в суть процесса моделирования этой простой системы. Представим себе логическую схему реализации модели этой системы на компьютере.

При моделировании систем массового обслуживания (СМО), ключевым понятием является событие.

В системе с одним обслуживающим элементом (кассиром) и очередью такие изменения, как приход клиента, начало обслуживания, конец обслуживания, называются событиями. Каждое событие в системе вызывают изменения состояния системы. Для построения модели, нужно для каждого события определить, как реализовать это событие и как корректировать состояние системы в связи ним. Среди всех событий ключевую роль при моделировании играют основные события.

Рассмотрим события, которые еще не возникли, но должны возникнуть. Эти события могут возникнуть немедленно (в текущий момент модельного времени) или попозже, если нужный момент для события еще не наступил. Такие события называются будущими событиями, и их наступление обычно планируется. Когда наступит время для будущего события, оно непременно произойдёт.

Все события в системе разделим на две группы, основных и вспомогательных.

Основное - это такое событие, время возникновения которого можно запланировать заранее. Остальные события назовём вспомогательными, так как время их возникновения заранее запланировать нельзя. Вспомогательные события возникают как следствие основных событий, и поэтому любое вспомогательное событие по времени возникновения обязательно совпадает с одним из основных событий.

В системе с кассиром и клиентами есть две группы основных событий – приход очередного клиента, и завершение обслуживания очередного клиента. Времена прихода клиентов должны соответствовать распределению интервалов прихода клиентов, а времена завершения обслуживания клиентов должны соответствовать распределению времен их обслуживания.

Предположим, что к кассиру пришёл клиент. Необходимо запланировать время прихода следующего клиента. Для этого:

1. Разыгрывают случайное число в соответствии с интегральным законом распределения интервалов прихода клиента:

T<прихода>

2. Эта величина прибавляется к текущему времени в модели (к текущему значению модельного таймера). Результат и определяет момент, когда в будущем придет следующий клиент. Время прихода следующего клиента предсказывается, но оно предсказывается точно, так как определяется на основе известного закона распределения случайных величин. После планирования, система работает так, что клиент придет именно тогда, когда она запланировала его приход.

При достижении моделью времени прихода клиента, он появляется на входе системы, и вновь планируется приход следующего клиента. Время прихода первого клиента обычно планируется, таким образом, как если бы в нулевой момент времени пришел клиент, но на обслуживание он не поступил.

Рассмотрим событие завершения обслуживания. Завершение планируется, когда обслуживание заявки началось и ведется аналогично

планированию времени прихода. Когда момент времени завершения обслуживания будет, достигнут, может разыгрываться следующее время завершения обслуживания (если заявка есть в очереди или заявка только что поступила).

В общем случае возникновение одного основного события может вызвать планирование нескольких новых основных событий.

В системе с одним обслуживающим элементом, к вспомогательным событиям относятся:

поступление заявки на обслуживание. Оно возникает, например, когда обслуживающий элемент свободен и поступила заявка в систему.

Это вспомогательное событие вызывает переход обслуживающего элемента из свободного состояния в занятое и планирование времени окончания обслуживания.

Это же самое вспомогательное событие может возникнуть и как реакция на завершение обслуживания, если в очереди есть заявки.

Таким образом, нельзя планировать время поступления заявок на обслуживание. Дальше будем называть устройством, оборудование, способное обрабатывать не более одной заявки одновременно.

6. Схема обработки основных событий.

Приход заявки вызывает

планирование следующего прихода

проверку состояния устройства - свободно? Да или нет.

По нет, происходит поступление заявки в очередь.

По да – происходит поступление заявки на обслуживание.

Это в свою очередь вызывает переход устройства в занятое состояние и планирование окончания обслуживания.

Окончание обслуживания вызывает

проверку состояния очереди - есть заявка? Да или нет.

По нет, происходит переход устройства в свободное состояние.

По да - поступление заявки на обслуживание.

Это в свою очередь вызывает продвижение заявок в очереди и планирование окончания обслуживания

Можно считать, что все выполняемые в ходе моделирования действия, кроме планирования, являются вспомогательными событиями. В списке операций, вызываемых событиями, у нас нет тех, которые обеспечивают сбор статистики. На практике эти операции входят в работу модели: например, если нас интересует максимальная длина очереди, то обработка события поступления заявки в очередь должна быть расширена, с тем, чтобы обеспечивать сбор всей статистики по очереди.

7. Модельный таймер и завершение моделирования.

Обычно в начале моделирования, значение таймера равно нулю и разработчик модели сам должен решить, какому времени этот нуль соответствует. Затем нужно выбрать единицу модельного времени - сутки, час, минута и т.п.

Предположим, что моделируется система, и в данный момент ее состояние изменилось. Следующий логический шаг – это увеличение значения таймера.

Чисто теоретически, существуют два подхода:

1) увеличить его на некоторую единицу, проверить состояние системы и определить те из запланированных событий, которые должны произойти при новом значении таймера.

Если такие события есть, то нужно выполнить операции, реализующие события, и снова изменить значение таймера на эту единицу и т.д. Если событий на данный момент не запланировано, то значение таймера просто должно увеличиваться на эту единицу.

2) этот подход используют концепцию переменного приращения значения таймера. В соответствии с ним, время в модели наращивается до ближайшего основного события.

Очевидно, что выгоднее второй подход, т.к. при этом не обрабатываются те моменты времени, на которые не запланировано событий.

Реальный выигрыш во многом зависит как от вида модели, так и от качества реализации системы.

Ближайшее событие определяется на основе списка будущих событий. События в нем стоят в порядке наступления. Когда обрабатываются основные события, то список будущих событий может быть скорректирован в соответствии со временем наступления каждого будущего события.

В принципе, рассмотренный подход обеспечивает бесконечное время моделирования системы. В действительности, рано или поздно, моделирование нужно прекращать. Завершение моделирование не всегда одномоментный процесс, например, может быть желательно не принимать новых заявок в систему, а выполнить обслуживание оставшихся заявок.

Для завершения моделирования, в GPSS предусмотрен так называемый счетчик завершений. Команда Start задаёт начальное значение этого счётчика, а каждая заявка, покидающая систему, может уменьшать значение счетчика на некоторую величину (обычно на единицу), или не менять его значения. Как только в ходе моделирования содержимое счётчика окажется равным нулю или отрицательным, моделирование немедленно останавливается и выдаётся стандартная итоговая статистика. При желании, разработчик модели может использовать значение счётчика, как и другие системные параметры, для управления процессом моделирования. В GPSS принято завершать моделирование либо после обработки заданного количества заявок, либо после истечения определенного времени моделирования.

8. Одновременные события.

События одновременны, если они происходят при одном значении модельного времени.

Пусть одновременно завершается обслуживание и поступает заявка. Тогда выбор ближайшего события неоднозначен, и в моделировании возникают тонкости. Зависимость от порядка обработки событий может повлиять на правильность моделей. Для нашего примера можно убедиться, что порядок обработки событий безразличен.

1. Пусть приходит заявка и очередь не пуста. Тогда в любом случае заявка попадает в очередь.

2. Пусть очередь пуста в момент прихода заявки. Тогда:

а) пусть сначала обрабатывается завершение обслуживания. Так как очередь пуста, то устройство станет свободным. Когда будет обрабатываться приход заявки - устройство будет свободно, и заявка попадает на обслуживание;

б) пусть сначала обрабатывается приход заявки. Она попадает в очередь. Затем происходит завершение обслуживания. Далее заявка из очереди попадает на обслуживание. Т.к. все это происходит в один момент модельного времени, то результат тот же самый.

В принципе 1 вариант обработки лучше, т.к. заявка попадает на обслуживание, минуя очередь.

Если задачу видоизменить и отказывать в обслуживании тем заявкам, которые приходят, когда устройство занято, то порядок обработки заявок станет важен.

Управлять порядком обработки можно с помощью приоритетов, которые также важны и при определении степени важности принятия на обслуживание тех или иных заявок .

9. Выводы.

Рассмотрение основных вопросов моделирования в простейшем случае , позволяет представить: какие средства нужно включить в язык моделирования Систем Массового Обслуживания (СМО).

Здесь нужны:

1) генераторы случайных чисел;

2) средства преобразования законов распределения случайных чисел в стандартные законы (экспоненциальный, нормальный и др.);

3) средства задания эмпирических (экспериментальных) законов распределения;

4) встроенный таймер модельного времени;

5) автоматическое выполнение таких логических операций, как:

проверка состояния очереди при завершении обслуживания,

определение наличия заявок для обслуживания и т.п.;

6) автоматическое продвижение таймера к следующему основному событию;

7) автоматическая передача управления в ту часть модели, где находится схема обработки нужного события;

8) возможность присвоения приоритета заявкам, чтобы управлять последовательностью обработки событий;

9) средства автоматического сбора статистики в тех местах модели, которые интересуют разработчика. Обычно собирают статистику по очередям и по устройствам, которые есть в модели;

10) возможность обслуживания заявок в том порядке, в каком предпочитает пользователь;

11) автоматическая выдача итоговой статистики по модели.

Фактически все эти средства включены в язык GPSS.

Запись на языке GPSS очень компактна по числу операторов. Поэтому многие детали моделирования исчезают из поля зрения. Следовательно, разработка моделей может потребовать большей тщательности, чем то может показаться на первый взгляд.

10. Основные концепции моделирования на GPSS.

Предварительные суждения.

G P S S - General Purpose Simulation System (общецелевая система моделирования).

Это система, воспринимает текст модели, и позволяет пользователю производить эксперименты с моделью на ЭВМ.

Модель на GPSS составляется из блоков, входящих в язык, и в этом виде поступает на моделирование.

Элементы модели GPSS.

Модель строится на основе объектов из 4-х классов:

динамических,

аппаратно-ориентированных,

статистических,

операционных.

Динамические объекты - это элементы потока обслуживания (заявки). Они создаются или уничтожаются в модели специальными операторами. Каждой заявке сопоставляется набор параметров (P параметров), с помощью которых разработчик модели может задать нужные характеристики заявок.

Работа системы отображается в модели в виде перемещения заявок от блока GENERATE к блокам TERMINATE через другие блоки модели. Заявки являются абстрактными подвижными элементами, которые могут моделировать объекты реального мира (людей, сообщения, программы, транспортные средства и т.д.). Перемещаясь между блоками модели, заявки вызывают и испытывают различные воздействия. Возможны их задержки в некоторых местах модели, изменение маршрута в зависимости от условий, расщепление заявки на несколько копий и т.п. Каждая заявка перемещается вместе с набором своих параметров. Набор включает: номер заявки; номер блока, где сейчас находится заявка; номер блока, куда она должна перейти; время начала движения; приоритет, который определяет порядок обработки, а также её личный набор параметров, которые задают желаемые характеристики моделируемого подвижного объекта. Некоторые другие параметры заявки мы рассмотрим позднее.

11. Списки в GPSS .

Любая из заявок хранится в GPSS в списках. Каждая заявка может быть:

в списке текущих событий,

в списке будущих событий,

в списке пользователя,

в списке прерываний,

в списке синхронизируемых заявок.

В списке текущих событий заявки расположены в порядке убывания приоритетов, а при равном приоритете - в порядке поступления в список.

Каждая заявка в этом списке может быть в потенциально активном состоянии, когда она подлежит просмотру в данный момент модельного времени, в состоянии движения, либо в состоянии задержки.

Если заявка находится в состоянии движения, то GPSS пытается продвинуть ее к следующим блокам, вплоть до выхода из системы. Если вход в очередной блок невозможен, то заявка переходит в состояние задержки. Если заявка попадает в блок ADVANCE, то она переходит из списка текущих событий в список будущих событий и для неё планируется время начала движения (после задержки). Части списка текущих событий для заявок ожидающих изменения состояния аппаратуры, являются, на самом деле, списками задержки. Когда изменится состояние соответствующей аппаратуры, например, заявка, освободит устройство, заявки из списка задержки этой аппаратуры перейдут в потенциально активное состояние. Когда завершится обработка движущейся заявки, очередная заявка, из числа потенциально активных заявок, получат возможность двигаться. Когда окажется, что ни одна заявка из списка текущих событий не может быть продвинута дальше, то считается, что настало время продвинуть таймер к очередному основному событию и его значение наращивается. Тогда из списка будущих событий в список текущих событий переходят те заявки, чьё время начала движения уже наступило.

Список будущих событий содержат заявки, у которых еще не настало время начала движения. Эти заявки располагаются строго в порядке возрастания времени начала движения. В этом списке приоритеты не действуют. В нем хранятся заявки, находящиеся в блоке ADVANCE или запланированные к приходу из блока GENERATE.

Список пользователя содержит заявки, временно удаленные из списка текущих событий с помощью блока LINK. Их возвращение в список текущих событий возможно с помощью блока UNLINK. Количество списков пользователя может быть произвольным.

Список прерываний содержит заявки, обслуживание которых прервано на устройстве. В этот список заявки заносятся по мере поступления, а извлекаются по мере надобности. Они никогда не обрабатываются по дисциплине списка. В списке для каждой заявки имеется ссылка на ту заявку, обслуживание которой прервала эта заявка. Поэтому возможно выстраивание цепочек прерываний.

Список синхронизируемых заявок. Он содержит заявки, которые ожидают комплектования в блоках ASSEMBLE и GATHER или ждут парной заявки в блоке MATCH.

12. Основные виды объектов в GPSS.

В системе GPSS имеются системные объекты, с которыми связаны следующие стандартные числовые атрибуты (СЧА).

RNj – случайное число, выдаваемое j датчиком случайных чисел. Все датчики генерируют последовательности равномерно распределенных случайных чисел из интервала от 0 до 999 включительно. В случае использования датчика в

качестве аргумента функции, он генерирует случайные числа из интервала от 0 до 0.999999;

C1 - текущее значение условного времени моделирования. Оно автоматически изменяется программой и устанавливается в 0 управляющими командами CLEAR или RESET;

AC1 - текущее значение абсолютного времени моделирования. Оно автоматически изменяется программой. Эта величина не меняется под действием управляющей команды RESET и устанавливается в 0 лишь под действием команды CLEAR;

TG1 - число, равное текущему значению счетчика завершений. Заявки, вошедшие в блоки TERMINATE с ненулевым операндом A, уменьшают значение этого счетчика на число, равное значению операнда A. Начальное значение счетчика определяет команда START. При значении счетчика равном 0 или отрицательном, моделирование останавливается.

Z1 - возвращает размер свободной оперативной памяти в байтах.

Наиболее важным объектом системы являются заявки. Они появляются из блока Generate и проходять через различные блоки модели, а затем уничтожаются блоками Terminate. С заявками связан ряд стандартных числовых атрибутов, которые можно использовать для организации логики работы модели.

Стандартные числовые атрибуты (СЧА) заявки:

XN1 - возвращает номер активной заявки;

M1 - время пребывания заявки в модели. Эта величина может изменяться блоком MARK. Время пребывания вычисляется следующим образом: M1 равно разнице текущего значения абсолютного времени и отметки времени рождения активной заявки;

PR - приоритет активной в данный момент заявки. Эта величина может изменяться блоком PRIORITY. По умолчанию приоритет равен 0.

A1 - номер ансамбля, к которому принадлежит заявка.

Pj или *j, или *<имя>, или P$<имя> - значение параметра j текущей заявки или значение параметра с именем <имя> текущей заявки ;

Косвенное обращение к стандартному числовому атрибуту формируется как начальная часть СЧА, затем *, а затем имя или номер P параметра, в котором находится номер или имя для уточнения основного СЧА. Например, FC*1 – это количество заявок, вошедших в устройство, номер или имя которого занесено в P1, SC*NUM – это количество заявок, вошедших в многоканальное устройство, номер или имя которого занесено в P$NUM. Для косвенного обращения можно использовать только P - параметры.

MPj - значение времени, равное разности абсолютного модельного времени и содержимого j-го параметра текущей заявки;

MBj - флаг синхронизации: 1 , если заявка, находящаяся в блоке j, принадлежит тому же семейству, что и текущая заявка; 0 - в противном случае.

Со стандартными числовыми атрибутами других объектов GPSS, можно ознакомиться в Приложении 1.

Аппаратно-ориентированные объекты соответствуют элементам оборудования, которое управляет заявками. Применительно к ЭВМ - это устройства, накопители (многоканальные устройства) и переключатели (ключи).

Устройство – это оборудование, которое может одновременно обслуживать одну заявку, многоканальное устройство - это оборудование, которое может одновременно обслуживать некоторое число заявок. Устройства моделируют объекты, в которых может происходить обработка заявки. Как правило, обработка связана с затратами времени. Существует полная аналогия между устройством и каналом системы массового обслуживания.

В GPSS можно моделировать прерывания на устройствах. Имеются средства логической проверки состояния устройств. Каждое из действий в устройстве отображается в модели определенным блоком. Занятие и освобождение устройства моделируются блоками: SEIZE <ном. устройства >, RELEASE <ном. устройства>. Для проверки состояния устройств используются блоки: GATE, TEST. Прерывания моделируют блоки: PREEMPT, RETURN, и так далее.

Внимание!!!

Если устройство или другой объект задан именем (см. EQU), то между СЧА этого объекта и именем указывается $.

(например, F$USTR1 - состояние устройства USTR1.)

Проверку состояний элементов оборудования выполняет блок GATE, который пропускает заявку, если проверяемое условие истинно и отсылает её по метке или задерживает, если оно ложно.

Многоканальные устройства служат для моделирования объектов обладающих емкостью.

Вход и выход в них моделируют блоки: ENTER и LEAVE. Входящая в блок ENTER заявка занимает часть емкости многоканального устройства, а выходящая через блок LEAVE заявка освобождает часть емкости. Емкость многоканального устройства указывается с помощью команды STORAGE.

Для представления в модели коммутационных объектов, которые могут быть включены или выключены, используются логические ключи. Срабатывание ключей моделирует блок: LOGIC, проверку состояния выполняет GATE. Блок LOGIC может установить ключ в состояние: S - включено(1), R - выключено(0), I - переключено.

Статистические объекты - это очереди и таблицы. Они используются для наблюдения за поведением системы и не влияют на ее работу. В каждой точке модели может возникнуть список задержанных заявок. Заявка задерживается перед блоками, вход в которые в данных условиях не возможен. Они могут задерживаться перед блоками SEIZE, ENTER, GATE, TEST и т.д.

Для сбора статистики об очередях, в местах задержки ставят блоки QUEUE и DEPART. При входе в первый блок текущая длина очереди обычно увеличивается на 1, при выходе - уменьшается.

Для сбора статистики по значениям какого-либо СЧА и представления ее в виде стандартной таблицы используют блоки: TABLE, TABULATE.

TABLE - описывает какую именно переменную надо табулировать и как именно.

TABULATE - собирает информацию для формирования таблицы.

Переменные (выражения).

Ранее в блоках GPSS нельзя было использовать логические и арифметические выражения, поэтому когда такая необходимость возникала, то использовались стандартные числовые атрибуты типа: "переменная". При этом, когда было нужно, происходило обращение к выражению, которое вычислялось, а результат вычисления затем использовался в блоке модели. Сейчас надобность в использовании переменных стала заметно меньше.

Для записи числовых и строковых значений в системе, используются ячейки. Запись ведет блок SAVEVALUE. Данные, записанные в ячейки, доступны любой заявке. Кроме простых ячеек, имеются матрицы ячеек.

Для временного хранения заявок, которые заведомо не должны пока двигаться по модели, используются списки пользователя. Заявки заносятся в списки блоками Link, а извлекаются из списков блоками Unlink.

13. Строки GPSS. Блоки и команды.

Каждый блок или команда модели характеризуется местоположением, операцией и операндом.

Каждый блок занимает в тексте модели определенное положение и имеет свой номер.

Аналогично, имеет свой номер (чаще метку) функция, очередь, многоканальное устройство, логические переключатели, таблицы, переменные и т.п.

Каждому блоку можно сопоставить символьную метку, которая располагается перед ключевым словом блока, отделяясь от него, по крайней мере, одним пробелом. Символьную метку, которая не сопоставлена блоку, можно также связать с числовым или строковым значением с помощью команды EQU.

Ее формат

<метка> EQU <число>.

В принципе, метке можно сопоставить PLUS – выражение, однако в нем не должна встречаться эта метка.

На символьную метку можно ссылаться непосредственно в выражениях и операндах. Обычно метки имеют те блоки и команды, на которые ссылаются другие блоки программы.

Операция блока - это глагол, определяющий основную роль блока.

GENERATE - генерирует

TEST - проверяет

QUEUE - ставит в очередь

ADVANCE - задерживает

Операнды задают уточняющую информацию для работы блока. Всего операндов может быть до 7. Типично когда в блоке 1 или 2 операнда.

Пример:

Mm1 ADVANCE 300,20; комментарий

Символ “; “ определяет начало комментария.

Блок ADVANCE задерживает заявку на 300+-20 единиц времени.

В поле операндов могут быть:
метки, которые эквивалентны числам;
стандартные числовые атрибуты и PLUS- выражения в круглых скобках ;
ключевые сокращения и метки блоков;
косвенно - адресуемые элементы вида.
СЧА1*<ном> или
СЧА1*<имя>

Здесь СЧА1 - это начальная часть некоторого стандартного числового атрибута без имени и номера, а имя или номер уточняется с помощью значение P – параметра заявки, то есть здесь используется атрибут P<ном> или P$<имя> .

Операнды обычно разделяются запятыми и недопустимы пробелы в поле операндов . Комментарии следует писать после ";".

Если строка начинается с *, то вся строка является комментарием.

Если используются числовые имена таблиц, накопителей и т.п., то нужно предварительно определить их с помощью команды эквивалентности EQU.

Пример:

```
Метка \          / Номер сопоставления.
     OC1 EQU 1
    USTR EQU 3
```

Тогда будут равноценными строки

```
QUEUE OC1
ENTER USTR
```

и

```
QUEUE 1
ENTER 3
```

Все эквивалентности должны быть описаны до их использования в исполнительной части модели.

С блоками модели связаны следующие СЧА:

Nj - общее число заявок, которое вошло в j-й блок. Подсчет ведется программой автоматически. Например, N$MET1 - счетчик числа входов в блок с меткой MET1. Этот счетчик изменяется при каждом входе заявки в блок;

Wj - текущее число заявок, которые находятся в блоке j. Значение этого счетчика также подсчитывается автоматически. Например, W$MET2 - счетчик текущего числа заявок в блоке с меткой MET2.

14. Основные команды и блоки GPSS.

Команда START - показывает, что все входные данные уже получены, и можно начинать моделирование. Команда START при работе с моделью может выдаваться неоднократно. После прекращения моделирования, можно изменить модель с помощью команд и вновь запустить ее с помощью команды START.

Команда может иметь до 4 параметров, то есть в общем виде выглядит как START A,B,C,D . Однако реально полезны только параметры A и B.

Здесь A - определяет содержимое счетчика завершений. Операторы TERMINATE могут вычитать определенное значение из счетчика завершений. Когда в счетчике завершений будет 0 или отрицательная величина, то моделирование завершится.

B – определяет нужно ли формировать стандартный отчёт, если параметр имеет значение NP , то отчет не создаётся. По умолчанию, параметр должен отсутствовать.

Моделирование можно приостановить без выдачи итоговой статистики, введя команду HALT, или задав предварительно команду STOP, которая может приостановить моделирование при достижении условия останова. Если после этого нужен отчет, то его можно получить, введя команду REPORT в окне CUSTOM.

Пример простой модели на GPSS.

GENERATE 100,7; генерируются заявки через 100 +/- 7 (93-107 единиц времени.)

SEIZE 1; занимается 1-ое устройство

ADVANSE 90,50; заявки задерживаются в устройстве на 90 +/- 50 единиц времени.

RELEASE 1; происходит выход из устройства 1.

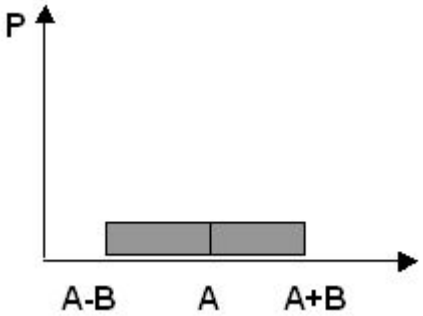
TERMINATE 2; содержимое счетчика завершения уменьшается на 2

START 200; завершить моделирование после обслуживания 200/2 заявок

Для генерирования заявок используется блок GENERATE, имеющий вид:

GENERATE A,B,C,D,E

С помощью операндов A и B, задаваемых как вещественные или целые положительные числа (возможно, выражения), определяется равномерное распределение интервалов при генерации заявок. Интервалы между заявками могут быть от A-B до A+B включительно. Плотность вероятности в интервале равна 1/(2*B).



Здесь A должно быть больше или равно B. Параметр C - если он указан, определяет момент появления 1-ой заявки. Например.

GENERATE 100,20,3000

Для оператора

GENERATE 100,20

момент появления первой заявки равномерно распределен в интервале от 80 до 120. Последующие заявки также будут появляться через интервал от 80 до 120 друг от друга.

Параметр D – определяет предельное число генерируемых заявок. Блок

GENERATE 100,20,,300

не сможет выпустить более 300 заявок. Если параметр опущен, то генератор может выпустить любое число заявок.

Параметр E - задает приоритет заявок. По умолчанию заявки имеют 0-ой, то есть самый низкий приоритет. Блок

GENERATE 100,20,,300,3

будет выпускать заявки с приоритетом 3.

Если 2-ой параметр (B) в блоке GENERATE – ссылка на функцию, то интервал времени определяется как произведение A*B . Ссылка на функцию имеет вид FN$имя_функции или Fnномер_функции. Блок

GENERATE 100,FN$spec

будет определять интервал между заявками как 100*FN$spec.

В GPSS- World нет смысла пользоваться этой возможностью, более того, лучше всячески избегать использования ссылки на функцию в качестве второго параметра в этом блоке.

Ввод заявок в систему ведет блок GENERATE, а вывод заявок из системы ведет блок TERMINATE. Заявка, попавшая в TERMINATE, удаляется из системы навсегда. Одновременно, если в блоке TERMINATE указано число, то при прохождении через него заявки, из счетчика завершений вычитается это число. Когда в счетчике завершений будет 0 или отрицательная величина, моделирование останавливается и выдается итоговая статистика.

Общий вид блока – следующий:

TERMINATE A

Моделирование задержки заявки ведет блок ADVANCE, который имеет 1 или 2 параметра. Блок очень похож на GENERATE и его параметры A и B имеют тот же смысл. Они определяют интервал задержки заявки. Здесь так же, если B является ссылкой на функцию, то интервал задержки определяется как произведение A на B. Общая форма блока:

ADVANCE A,B

Войти в ADVANCE и находиться в нем одновременно, может любое число заявок. Обычно блок ADVANCE моделирует обслуживание заявок в устройстве или в многоканальном устройстве и поэтому он обрамляется блоками входа/выхода в них.

В случае устройств, занятие устройства отображается блоком

SEIZE A

а освобождение устройства отображается блоком

RELEASE A

Здесь A – это номер или метка (имя) устройства.

Когда устройство занято, то в него не могут входить другие заявки, а когда оно свободно, то в него входит первая по очереди заявка.

Эти номер или имя устройства в блоках SEIZE и RELEASE должны быть одинаковыми. Устройство перестанет быть занятым, когда занявшая его заявка. пройдет через блок RELEASE этого устройства.

Заявка может занять любое число устройств .

Заявки, стоящие в очереди перед SEIZE, обслуживаются в порядке поступления, с учетом приоритетов. Заявка, занимающая устройство, не может покинуть систему.

С устройствами связаны следующие СЧА:

Fномер или F$имя - текущее состояние устройства с данным номером или именем. Эта величина равна 0, если устройство свободно, и 1 - во всех остальных случаях. Этот атрибут изменяется блоками SEIZE, RELEASE, PREEMPT и RETURN. Например, F$ACPU - состояние устройства ACPU;

FIномер или FI$имя – флаг прерывания устройства с данным номером или именем. Эта величина равна 1, если устройство находится в состоянии прерывания, 0 - в противном случае;

FVномер или FV$имя – флаг готовности к использованию устройства с данным номером или именем. Эта величина равна 1, если устройство готово к использованию, 0 - в противном случае;

FRномер или FR$имя - коэффициент использования устройства в тысячных долях, т.е., если коэффициент равен 0.88, то FRj равен 880;

FCномер или FC$имя - общее число входов в устройство.

FTномер или FT$имя - среднее время использования устройства одной заявкой.

Команда START в ходе моделирования может быть введена несколько раз, и каждый раз будет запущен процесс продолжения моделирования из того состояния, в котором застала модель выдача итоговой статистики по нулевому значению счетчика моделирования.

Перед запуском, модель можно подправить, введя новые команды управления моделью, в частности, указав команды EQU, RESET и CLEAR.

Команда RESET сбрасывает в ноль статистику, накопленную по устройствам, очередям, спискам пользователя, таблицам и блокам. Однако она не удаляет заявок из модели.

Подробнее действие оператора RESET можно описать следующим образом:

значение условного относительного времени (C1) устанавливается в ноль;

значение условного абсолютного времени (AC1) остается неизменным;

все датчики псевдослучайных чисел остаются неизменными;

значения ячеек и матриц, а также состояния логических ключей не изменяются.

Счетчики блоков (Nj) полагаются равными (Wj). Временные интегралы устройств устанавливаются в ноль.

Временные интегралы содержимого многоканальных устройств устанавливаются в ноль. Счетчики числа входов в многоканальное устройство

(SCj) и максимальное содержимое многоканального устройства(SMj) устанавливаются в соответствии с текущим числом занятых каналов в данный момент времени. Временные интегралы всех очередей сбрасываются в ноль.

Счетчики входов в очередь (QCj) и максимальное содержимое очередей (QMj) полагаются равными текущей длине очереди. В таблицах стираются накопленные статистические данные.

Временные интегралы списков пользователя сбрасываются в ноль. Счетчик числа входов в список (CCj) и максимальное содержимое (CMj) устанавливаются равным текущей длине списка.

Команда CLEAR фактически приводит модель в то состояние, которое было в ней сразу после компиляции.

В формате CLEAR OFF, команда обеспечивает сохранение в модели без изменений всех значений логических ключей, ячеек и элементов матриц, иначе они также обнуляются.

Для слежения за очередями, а совсем не для их создания используются блоки

QUEUE A,B

и

DEPART A,B

Здесь A – это номер или имя очереди.

A B- это количество занимаемых или освобождаемых единиц очереди.

Эти блоки следят за очередями, которые возникают независимо от блоков слежения. Если Вы поставили эти блоки, то в итоговой статистике будет информация о соответствующей очереди.

При входе в блок QUEUE текущая длина очереди увеличивается на B, при выходе через DEPART, уменьшается на B. По умолчанию, значение B равно 1.

С очередями связаны следующие СЧА:

Qномер или Q$имя - длина соответствующей очереди . Эта величина может изменяться блоками QUEUE и DEPART. Например, Q2 соответствует очереди 2, а Q$FAC соответствует очереди FAC;

QAномер или QA$имя - средняя длина очереди.

QMномер или QM$имя - максимальная длина очереди.

QCномер или QC$имя - общее число вхождений в очередь.

QZномер или QZ$имя - число нулевых вхождений в очередь. То есть число заявок, которые вошли в очередь, но в ней не задержались. Число нулевых вхождений подсчитывается как число заявок, которые, войдя в QUEUE, тут же прошли в другой блок, вне зависимости оттого, что это за блок.

QTномер или QT$имя - среднее время пребывания заявки в очереди (включая нулевые вхождения);

QXномер или QX$имя - среднее время пребывания заявки в очереди (без нулевых вхождений).

Пример:

Пусть заявки приходят на обслуживание с средним интервалом 100 и отклонением 80 единиц времени. Распределение интервалов прихода – равномерное. Затем они обслуживаются в устройстве по очереди за среднее время 70 с отклонением в 20 единиц времени. Интервалы обслуживания распределены равномерно. Промоделировать работу такой системы в течение 100000 единиц времени. Организовать наблюдение за очередью.

Текст такой модели выглядит следующим образом.

```
GENERATE 100,80
QUEUE 1
SEIZE num1
DEPART 1
ADVANCE 70,20
RELEASE num1
TERMINATE
GENERATE 100000
TERMINATE 1
START 1
```

В данном примере, заявки приходят в систему с интервалом 100+- 80 единиц. Поступают в очередь с номером 1, затем занимают устройство с именем num1. Далее покидают очередь с номером 1 . Задерживаются в устройстве на 70+- 20 единиц времени. Затем покидают устройство num1 и удаляются из системы. Когда одна из заявок освобождает устройство, то его может занять первая по очереди заявка, и она пройдет тот же путь. Когда наступит момент времени 100000, то второй генератор выдаст первую заявку, которая будет тут же уничтожена и содержимое счетчика завершений станет равным 0, а значит, моделирование остановится и будет выдана итоговая статистика.

В данном случае она будет выглядеть следующим образом.

```
                GPSS World Simulation Report - Untitled Model 1.3.1

                     Thursday, December 05, 2002 17:49:53

           START TIME            END TIME  BLOCKS  FACILITIES  STORAGES
                0.000          100000.000     9        1          0

              NAME                       VALUE
           NUM1                        10000.000

 LABEL              LOC  BLOCK TYPE     ENTRY COUNT CURRENT COUNT RETRY
                    1    GENERATE           992             0       0
                    2    QUEUE              992             0       0
                    3    SEIZE              992             0       0
                    4    DEPART             992             0       0
                    5    ADVANCE            992             1       0
                    6    RELEASE            991             0       0
                    7    TERMINATE          991             0       0
                    8    GENERATE             1             0       0
                    9    TERMINATE            1             0       0

FACILITY          ENTRIES  UTIL.   AVE. TIME AVAIL. OWNER PEND INTER RETRY DELAY
 NUM1               992    0.692      69.713  1      993    0    0     0       0

QUEUE              MAX CONT. ENTRY ENTRY(0) AVE.CONT. AVE.TIME   AVE.(-0) RETRY
 1                   3    0    992    601     0.145     14.619     37.089   0

FEC XN   PRI         BDT      ASSEM  CURRENT  NEXT  PARAMETER    VALUE
```

```
993    0      100003.419    993      5      6
994    0      100055.299    994      0      1
995    0      200000.000    995      0      8
```

Пусть в предыдущем примере, мы организуем многократное моделирование обслуживания по 500 групп заявок, каждый раз сбрасывая статистку. Всего разрешим генератору выпустить 1200 заявок. В конце, когда моделирование завершится по ошибке, выполним команду Report

Текст модели может выглядеть, например, следующим образом:

```
    GENERATE 100,80,,1200
    QUEUE 1
    SEIZE num1
    depart 1
    ADVANCE 98,70
    RELEASE num1
    TERMINATE  1
start 500
reset
start 500
reset
start 500
report
```

Текст в окне компиляции будет выглядеть следующим образом.

10/16/03 16:29:14 Model Translation Begun.
10/16/03 16:29:14 Ready.
10/16/03 16:29:14 Simulation in Progress.
10/16/03 16:29:14 The Simulation has ended. Clock is 49810.294860.
10/16/03 16:29:14 Reporting in Untitled Model 1.15.1 - REPORT Window.

10/16/03 16:29:14 Simulation in Progress.
10/16/03 16:29:14 The Simulation has ended. Clock is 99749.876560.
10/16/03 16:29:14 Reporting in Untitled Model 1.15.2 - REPORT Window.

10/16/03 16:29:14 Simulation in Progress.
10/16/03 16:29:14 Error Stop.
10/16/03 16:29:14 Halt.
10/16/03 16:29:14 Clock:120022.213940.
10/16/03 16:29:14 There are no Transactions. Check Transaction limits and blocking.
10/16/03 16:29:14 Reporting in Untitled Model 1.15.3 - REPORT Window.

Нетрудно убедиться, что здесь моделирование действительно прервалось по ошибке.

Три выходных статистики этого моделирования, имеют вид:

Thursday, October 16, 2003 16:29:14

START TIME	END TIME	BLOCKS	FACILITIES	STORAGES
0.000	49810.295	7	1	0

NAME	VALUE
NUM1	10000.000

LABEL	LOC	BLOCK TYPE	ENTRY COUNT	CURRENT COUNT	RETRY
	1	GENERATE	502	0	0
	2	QUEUE	502	1	0
	3	SEIZE	501	1	0
	4	DEPART	500	0	0
	5	ADVANCE	500	0	0
	6	RELEASE	500	0	0
	7	TERMINATE	500	0	0

FACILITY	ENTRIES	UTIL.	AVE. TIME	AVAIL.	OWNER	PEND	INTER	RETRY	DELAY
NUM1	501	0.982	97.592	1	501	0	0	0	1

QUEUE	MAX	CONT.	ENTRY	ENTRY(0)	AVE.CONT.	AVE.TIME	AVE.(-0)	RETRY
1	13	2	502	21	4.993	495.392	517.021	0

CEC XN	PRI	M1	ASSEM	CURRENT	NEXT	PARAMETER	VALUE
501	0	49690.818	501	3	4		

FEC XN	PRI	BDT	ASSEM	CURRENT	NEXT	PARAMETER	VALUE
503	0	49834.158	503	0	1		

GPSS World Simulation Report - Untitled Model 1.15.2

Thursday, October 16, 2003 16:29:14

START TIME	END TIME	BLOCKS	FACILITIES	STORAGES
49810.295	99749.877	7	1	0

NAME	VALUE
NUM1	10000.000

LABEL	LOC	BLOCK TYPE	ENTRY COUNT	CURRENT COUNT	RETRY
	1	GENERATE	500	0	0
	2	QUEUE	501	1	0
	3	SEIZE	501	1	0
	4	DEPART	500	0	0
	5	ADVANCE	500	0	0
	6	RELEASE	500	0	0
	7	TERMINATE	500	0	0

FACILITY	ENTRIES	UTIL.	AVE. TIME	AVAIL.	OWNER	PEND	INTER	RETRY	DELAY
NUM1	501	0.987	98.386	1	1001	0	0	0	1

QUEUE	MAX	CONT.	ENTRY	ENTRY(0)	AVE.CONT.	AVE.TIME	AVE.(-0)	RETRY
1	14	2	502	16	5.613	558.427	576.812	0

CEC XN	PRI	M1	ASSEM	CURRENT	NEXT	PARAMETER	VALUE
1001	0	99540.920	1001	3	4		

FEC XN	PRI	BDT	ASSEM	CURRENT	NEXT	PARAMETER	VALUE
1003	0	99796.720	1003	0	1		

GPSS World Simulation Report - Untitled Model 1.15.3

Thursday, October 16, 2003 16:29:14

```
          START TIME           END TIME  BLOCKS  FACILITIES  STORAGES
           99749.877         120022.214     7        1          0

             NAME                        VALUE
            NUM1                       10000.000

 LABEL              LOC  BLOCK TYPE     ENTRY COUNT CURRENT COUNT RETRY
                    1    GENERATE           198             0       0
                    2    QUEUE              199             0       0
                    3    SEIZE              200             0       0
                    4    DEPART             200             0       0
                    5    ADVANCE            200             0       0
                    6    RELEASE            200             0       0
                    7    TERMINATE          200             0       0

FACILITY          ENTRIES  UTIL.   AVE. TIME AVAIL. OWNER PEND INTER RETRY DELAY
 NUM1                200    0.929      94.215  1        0    0     0     0      0

QUEUE              MAX CONT. ENTRY ENTRY(0) AVE.CONT. AVE.TIME    AVE.(-0) RETRY
 1                   5    0    200     38     0.975     98.869    122.060   0
```

По выходной статистике данного примера видно, что генератор действительно выпустил 1200 заявок.

15. Техника построения модели и проведения моделирования.

Эта работа состоит из следующих этапов.

Вначале необходимо разработать детальное описание задачи, и сформулировать ее, используя понятия языка GPSS-World, а также указать назначение всех объектов модели. Здесь же нужно указать, что принято за единицу модельного времени.

После этого нужно сформировать текст модели, желательно с комментариями, используя для этого пункты File/New. В появившемся окне следует выбрать вариант Model и щелкнуть по OK. Будет создано окно для ввода текста модели.

Когда текст будет набран, нужно выбрать пункты Command/Create Simulation. Будет создана компилированная модель (моделирование) и результат компиляции отобразится в новом окне. Если компиляция пройдет без ошибок, то модель можно будет запускать на выполнение. Иначе следует исправить ошибки в тексте или напрямую, или, используя пункты подменю Search Next Error и Previous Error.

После исправления ошибок можно запустить модель на выполнение, например командой Command/Start. Когда моделирование будет завершено, то сформируется итоговая статистика и при необходимости можно будет продолжить процесс моделирования, видоизменив параметры модели или без их изменений, выполнив новую команду Start.

Подробности истолкования итоговой статистики можно узнать из приложения 2.

Если сохранить окно компиляции, то для моделирования станет не обязательным располагать текстом модели. Достаточно будет открыть это окно, и можно будет сразу вести моделирование. Это говорит о том, что компилированная модель сама по себе является программой и имеет самостоятельную ценность. В

частности, она может быть настроена нужным образом с помощью команд GPSS-World, или путем использования текстовых файлов, которые компилированная модель может прочитать. В процессе моделирования можно наблюдать за работой модели с помощью специальных окон из подменю Window. Подробнее о работе с ними можно прочитать в приложении 3.

Рассмотрим способ управления компилированной моделью с помощью команд Equ.

Прежде всего, заменим все числа - метками, и зададим их значения с помощью эквивалентностей. Получим, например, следующий текст модели.

```
Inter_in equ 100
Inter_work equ 70
Del_in equ 80
Del_work equ 20
Time equ 100000
 GENERATE Inter_in,Del_in
 QUEUE 1
 SEIZE num1
 DEPART 1
 ADVANCE Inter_work,Del_work
 RELEASE num1
 TERMINATE
 GENERATE time
 TERMINATE 1
START 1
```

Проведя моделирование, нетрудно убедиться, что результат этого моделирования будет точно таким же, как и у исходной модели.

Впредь мы, не всегда будем приводить итоговую статистику. Мы будем приводить ее в тех случаях, когда это особенно важно для правильного понимания работы той или иной модели.

Очевидно что, перед выполнением команды Start, мы можем задать новые значения одной или нескольким меткам модели. Например, мы можем задать следующие значения меток с помощью пункта Command/Custom

```
Inter_work equ 90.35
Time equ 200000
```

А затем выполнить пункты

Clear

Start 1

Получим другие результаты в файле итоговой статистики.

Аналогичным образом можно управлять другими параметрами модели. Однако в случае использования в команде Equ выражений, их нужно задавать в пункте Command/Custom по одной команде. В выражении нельзя напрямую использовать переопределяемую метку. Значения меток можно переопределять в процедурах языка PLUS простым присваиванием. Конечно, если они не являются формальными параметрами процедуры. (точнее функции), или ее внутренними (временными) переменными .

В качестве законов распределения для интервалов прихода заявок, и интервалов обслуживания, кроме равномерного распределения довольно часто используется экспоненциальное и нормальное распределения интервалов. Для их задания используют встроенные функции Exponential(A,B,C) и Normal(A,B,C). Здесь в Exponential(A,B,C)

A – это номер генератора случайных чисел, чаще всего 1

B – это минимальное значение интервала, обычно 0.

C – это среднее значение величины, распределенной по указанному закону. Среднее значение интервала при этом составляет B+C.

A в Normal(A,B,C)

A – это номер генератора случайных чисел, чаще всего 1

B – это среднее значение величины, распределенной по указанному закону.

C – это среднеквадратичное отклонение интервала. Отложив подробное обсуждение других функций и выражений на потом, отметим пока, что выражения в этой версии GPSS необходимо записывать в круглых скобках. Например, если в предыдущей задаче интервал поступления заявок подчиняется экспоненциальному закону, интервал обслуживания – нормальному, то текст модели может приобрести следующий вид.

```
Inter_in equ 100
Inter_work equ 70
Del_in equ 80
Del_work equ 20
Time equ 100000
 GENERATE (exponential(1,0,Inter_in))
 QUEUE 1
 SEIZE num1
 DEPART 1
 ADVANCE (normal(1,Inter_work,Del_work))
 RELEASE num1
 TERMINATE
 GENERATE time
 TERMINATE 1
START 1
```

Рассмотрим расширенный пример использования блоков работы с устройствами.

Пусть на одном компьютере работают два пользователя.

Один из них использует его со средним интервалом 4 часа, который распределен по экспоненциальному закону. Работа на компьютере продолжается в течении 1.5 +-0.5 часа и включает в себя обращение к интернету через сервер в течении 20+-10 минут в начале работы.

Другой использует его со средним интервалом 3 часа, распределенным по экспоненциальному закону. Работа на компьютере продолжается в течении 1.3 +-0.75 часа и включает в себя обращение к интернету через сервер в течении 15+-7 минут в начале работы.

На сервер поступают другие задания со средним интервалом 30 минут, требующие обслуживания в течении 15 минут. Эти интервалы распределены по экспоненциальному закону.

Промоделировать работу системы в течение 120 месяцев по 22 рабочих дня, то есть в течение 120*22*8 часов.

Собрать статистику по очередям, и по занятости устройств каждым пользователем.

Для сбора статистики по занятости устройств будем использовать дополнительные очереди, так как по очередям собирается наиболее обширная статистика.

Текст такой модели может выглядеть, например, следующим образом.

```
GENERATE        (exponential(1,0,(6#60)))
QUEUE  pc
SEIZE    pc
DEPARTpc
QUEUE USER1_PC
QUEUE SERV
SEIZE SERV
DEPART SERV
QUEUE USER1_SERV
ADVANCE 20,10
DEPART USER1_SERV
RELEASE SERV
ADVANCE         (1.5#60),(0.5#60)
DEPART USER1_PC
RELEASE         pc
TERMINATE

GENERATE        (exponential(1,0,(4#60)))
QUEUE  pc
SEIZE    pc
DEPARTpc
QUEUE USER2_PC
QUEUE SERV
SEIZE SERV
DEPART SERV
QUEUE USER2_SERV
ADVANCE 15,7
DEPART USER2_SERV
RELEASE SERV
ADVANCE         (1.3#60),(0.75#60)
DEPART USER2_PC
RELEASE         pc
TERMINATE

GENERATE        (exponential(1,0,30))
```

```
QUEUE  SERV
SEIZE    SERV
DEPART  SERV
ADVANCE (exponential(1,0,15))
RELEASE          SERV
TERMINATE

GENERATE (22#8#60)
TERMINATE 1
START 100
```

В этой модели используется 2 устройства pc – персональный компьютер, и serv – сервер. Используются также очереди к устройствам Serv и pc. А для сбора статистики по занятости персонального компьютера и сервера каждым из пользователей, используются очереди USER1_PC, USER1_SERV, USER2_PC, USER2_SERV.

Результат моделирования может выглядеть, например, следующим образом.

```
            GPSS World Simulation Report - ex1s.11.1

                 Tuesday, December 17, 2002 14:16:24

          START TIME           END TIME  BLOCKS  FACILITIES  STORAGES
               0.000        1056000.000    41        2          0

             NAME                      VALUE
         PC                          10001.000
         SERV                        10000.000
         USER1_PC                    10004.000
         USER1_SERV                  10005.000
         USER2_PC                    10002.000
         USER2_SERV                  10003.000


LABEL              LOC  BLOCK TYPE     ENTRY COUNT CURRENT COUNT RETRY
                     1  GENERATE          2922             0       0
                     2  QUEUE             2922             0       0
                     3  SEIZE             2922             0       0
                     4  DEPART            2922             0       0
                     5  QUEUE             2922             0       0
                     6  QUEUE             2922             0       0
                     7  SEIZE             2922             0       0
                     8  DEPART            2922             0       0
                     9  QUEUE             2922             0       0
                    10  ADVANCE           2922             0       0
                    11  DEPART            2922             0       0
                    12  RELEASE           2922             0       0
                    13  ADVANCE           2922             1       0
                    14  DEPART            2921             0       0
                    15  RELEASE           2921             0       0
                    16  TERMINATE         2921             0       0
                    17  GENERATE          4297             0       0
                    18  QUEUE             4297             0       0
                    19  SEIZE             4297             0       0
                    20  DEPART            4297             0       0
                    21  QUEUE             4297             0       0
                    22  QUEUE             4297             0       0
                    23  SEIZE             4297             0       0
                    24  DEPART            4297             0       0
                    25  QUEUE             4297             0       0
                    26  ADVANCE           4297             0       0
```

```
    27    DEPART          4297          0       0
    28    RELEASE         4297          0       0
    29    ADVANCE         4297          0       0
    30    DEPART          4297          0       0
    31    RELEASE         4297          0       0
    32    TERMINATE       4297          0       0
    33    GENERATE       34982          0       0
    34    QUEUE          34982          0       0
    35    SEIZE          34982          0       0
    36    DEPART         34982          0       0
    37    ADVANCE        34982          0       0
    38    RELEASE        34982          0       0
    39    TERMINATE      34982          0       0
    40    GENERATE         100          0       0
    41    TERMINATE        100          0       0


FACILITY         ENTRIES  UTIL.   AVE. TIME AVAIL. OWNER PEND INTER RETRY DELAY
 SERV             42201    0.615      15.391  1        0    0     0     0      0
 PC                7219    0.782     114.418  1    42303    0     0     0      0


QUEUE              MAX CONT. ENTRY ENTRY(0) AVE.CONT. AVE.TIME   AVE.(-0) RETRY
 SERV               12    0  42201  17093     0.754     18.870     31.716   0
 PC                 22    0   7219   1603     1.595    233.374    299.987   0
 USER2_PC            1    0   4297      0     0.438    107.550    107.550   0
 USER2_SERV          1    0   4297      0     0.061     15.001     15.001   0
 USER1_PC            1    1   2922      0     0.345    124.517    124.517   0
 USER1_SERV          1    0   2922      0     0.055     20.018     20.018   0


FEC XN   PRI         BDT      ASSEM  CURRENT  NEXT  PARAMETER    VALUE
 42297    0     1056000.943   42297      0     17
 42302    0     1056039.264   42302      0     33
 42304    0     1056056.420   42304      0      1
 42303    0     1056065.319   42303     13     14
 42305    0     1066560.000   42305      0     40
```

В результате анализа статистики работы пользователей в течение 10 лет, мы получили следующие результаты. Хотя на персональном компьютере работало всего 2 пользователя, максимальная очередь по заданиям, которые они выполняли, составила 22 задания, а средняя очередь – 1.59 задания. Сам персональный компьютер имел занятость 0.782. Сервер имел максимальную очередь в 12 заданий, а среднюю – в 0.754 задания. Первый пользователь выполнил 2921 задание, а второй - 4297 заданий. Среднее время выполнения задания пользователями составило 114.4 минуты. А среднее время выполнения задания сервером, составило 15.39 минуты. Пользователям удалось сразу сесть за персональный компьютер всего 1603 раза. В то время, как всего было выполнено 7219 заданий. Компьютер был занят обслуживанием заданий первого пользователя в среднем 124.517 минут, а обслуживанием заданий второго пользователя в среднем 107.55 минут. Занятость компьютера первым пользователем составила 0.345, а вторым пользователем - составила 0.438. Загрузка сервера первым пользователем составила 0.055, а вторым пользователем составила 0.061. И это далеко не вся информация, полученная в ходе моделирования такой простой модели!

Для моделирования работы с многоканальными устройствами используют блоки ENTER и LEAVE. А для их описания – команду STORAGE.

Команда описания емкости устройства STORAGE имеет вид

метка STORAGE A

где A – число каналов в устройстве.

число каналов должна быть целым числом. Оно обычно определяет число заявок, которые могут одновременно обслуживаться в многоканальном устройстве. Метка определяет имя устройства. Если понадобится определить устройство с помощью номера, то метке должно быть сопоставлено целое число с помощью эквивалентности, причем до описания емкости самого устройства.

Занятие многоканального устройства выполняется с помощью блока ENTER.

ENTER A,B

Освобождение выполняется с помощью блока LEAVE.

LEAVE A,B

Здесь A – это имя или номер устройства, а B это количество единиц емкости, на которое претендует заявка. По умолчанию, B равно 1, то есть заявка претендует на один канал.

Заявка может войти в устройство, если в нем есть то число свободных каналов, на которое она претендует. Иначе заявка ждет, пока нужное количество каналов освободится. При освобождении устройства, заявка может освободить любое число каналов, хотя обычно она освобождает то число каналов, которое она заняла. Недопустимо освобождать больше каналов, чем их занято в данный момент.

Если емкость многоканального устройства исчерпана или недостаточна для вхождения в него заявки, то заявка остается на выходе предыдущего блока.

С многоканальными устройствами связаны следующие СЧА:

Sномер или S$имя - текущее число занятых каналов в многоканальном устройстве. Величина изменяется блоками ENTER и LEAVE. Например, S$OPER - текущее число занятых каналов многоканального устройства OPER;

Rномер или R$имя - число свободных каналов многоканального устройства. Эта величина также изменяется блоками ENTER и LEAVE. Например, R$MACH - свободный объем многоканального устройства MACH;

SRномер или SR$имя - коэффициент использования многоканального устройства в тысячных долях, т.е., если коэффициент равен 0.65, то SR равно 650;

SAномер или SA$имя - среднее содержимое многоканального устройства ;

SMномер или SM$имя - максимальное содержимое многоканального устройства ;

SCномер или SC$имя - общее число входов в многоканальное устройство;

STномер или ST$имя - среднее время пребывания одной заявки в многоканальном устройстве.

SEномер или SE$имя - флаг не занятости (пустоты) многоканального устройства : 1 - свободно, 0 - занято;

SFномер или SF$имя - флаг занятости всех каналов многоканального устройства : 1 - заполнено, 0 - не заполнено;

SVномер или SV$имя - флаг готовности многоканального устройства : 1 - готово , 0 - не готово;

Для своего обслуживания, заявка может одновременно занимать любое число одноканальных и многоканальных устройств.

Простейший пример применения многоканальных и одноканальных устройств совместно:

Пусть заявки первого типа поступают на обслуживание в систему с интервалом 100+-50 секунд. Они проходят вначале через 5 канальное устройство nac, где обслуживаются в течение 450+- 100 сек. А затем они проходят через одноканальное устройство 3, где обслуживаются в течение 90+- 20 сек и покидают модель.

Пусть также заявки второго типа поступают на обслуживание в систему с интервалом 1000+-100 секунд. Они проходят через 5 канальное устройство nac, где обслуживаются в течение 500+- 200 сек, занимая по 2 канала. Затем они покидают модель. Промоделировать обслуживание 1000 заявок. Текст такой модели, мог бы иметь, например, следующий вид:

```
nac STORAGE 5
GENERATE 100,50
QUEUE NAC
ENTER nac
DEPART NAC
ADVANCE 450,100
LEAVE nac
QUEUE 3
SEIZE 3
DEPART 3
ADVANCE 90,20
RELEASE 3
TERMINATE 1

GENERATE 1000,100
QUEUE NAC1,2
ENTER nac,2
DEPART NAC1,2
ADVANCE 500,200
 LEAVE nac,2
 TERMINATE
 START 1000
```

Результаты моделирования могут выглядеть, например, следующим образом:

```
    GPSS World Simulation Report - Untitled Model 1.3.1

         Wednesday, December 18, 2002 15:10:13

 START TIME           END TIME  BLOCKS  FACILITIES  STORAGES
      0.000         100249.641    19        1          1

    NAME                      VALUE
NAC                         10000.000
```

```
          NAC1                       10001.000

 LABEL              LOC  BLOCK TYPE     ENTRY COUNT CURRENT COUNT RETRY
                     1    GENERATE          1008             0       0
                     2    QUEUE             1008             0       0
                     3    ENTER             1008             0       0
                     4    DEPART            1008             0       0
                     5    ADVANCE           1008             5       0
                     6    LEAVE             1003             0       0
                     7    QUEUE             1003             2       0
                     8    SEIZE             1001             1       0
                     9    DEPART            1000             0       0
                    10    ADVANCE           1000             0       0
                    11    RELEASE           1000             0       0
                    12    TERMINATE         1000             0       0
                    13    GENERATE           101             0       0
                    14    QUEUE              101            53       0
                    15    ENTER               48             0       0
                    16    DEPART              48             0       0
                    17    ADVANCE             48             0       0
                    18    LEAVE               48             0       0
                    19    TERMINATE           48             0       0

FACILITY          ENTRIES  UTIL.   AVE. TIME AVAIL. OWNER PEND INTER RETRY DELAY
 3                  1001    0.899      90.014  1     1103    0    0     0      2

QUEUE               MAX CONT. ENTRY ENTRY(0) AVE.CONT. AVE.TIME   AVE.(-0) RETRY
 3                    6    3   1003    132     1.922    192.076    221.186   0
 NAC                 51    0   1008     26    18.775   1867.280   1916.719   0
 NAC1               106  106    202      0    36.816  18271.466  18271.466   0

STORAGE             CAP. REM. MIN. MAX.  ENTRIES AVL.  AVE.C. UTIL. RETRY DELAY
 NAC                  5    0   0     5     1104   1    4.972  0.994    0   53

CEC XN   PRI          M1      ASSEM  CURRENT  NEXT  PARAMETER    VALUE
  1103    0       99559.739   1103      8      9

FEC XN   PRI          BDT      ASSEM  CURRENT  NEXT  PARAMETER    VALUE
  1111    0      100267.030   1111      0      1
  1104    0      100271.944   1104      5      6
  1106    0      100273.148   1106      5      6
  1109    0      100440.757   1109      5      6
  1107    0      100582.197   1107      5      6
  1110    0      100643.369   1110      5      6
  1108    0      100899.989   1108      0     13
```

Однако, если ввести последовательно команды
Nac storage 7
Clear
Start 1000

То получим следующую выходную статистику, в которой емкость многоканального устройства будет уже 7, а очереди к устройствам окажутся заметно меньше.

```
              GPSS World Simulation Report - Untitled Model 1.3.2

                   Wednesday, December 18, 2002 15:13:15

          START TIME           END TIME  BLOCKS  FACILITIES  STORAGES
               0.000         101882.427    19        1          1

             NAME                     VALUE
          NAC                       10000.000
          NAC1                      10001.000

 LABEL              LOC  BLOCK TYPE     ENTRY COUNT CURRENT COUNT RETRY
```

```
                    1    GENERATE          1006             0       0
                    2    QUEUE             1006             0       0
                    3    ENTER             1006             0       0
                    4    DEPART            1006             0       0
                    5    ADVANCE           1006             6       0
                    6    LEAVE             1000             0       0
                    7    QUEUE             1000             0       0
                    8    SEIZE             1000             0       0
                    9    DEPART            1000             0       0
                   10    ADVANCE           1000             0       0
                   11    RELEASE           1000             0       0
                   12    TERMINATE         1000             0       0
                   13    GENERATE           101             0       0
                   14    QUEUE              101             0       0
                   15    ENTER              101             0       0
                   16    DEPART             101             0       0
                   17    ADVANCE            101             0       0
                   18    LEAVE              101             0       0
                   19    TERMINATE          101             0       0

FACILITY          ENTRIES  UTIL.   AVE. TIME AVAIL. OWNER PEND INTER RETRY DELAY
 3                  1000    0.884      90.056  1        0    0    0     0      0

QUEUE               MAX CONT. ENTRY ENTRY(0) AVE.CONT. AVE.TIME   AVE.(-0) RETRY
 3                    4    0   1000     252     0.570     58.023     77.571   0
 NAC                  2    0   1006     875     0.057      5.743     44.100   0
 NAC1                 2    0    202     178     0.016      8.304     69.896   0


STORAGE             CAP. REM. MIN. MAX.  ENTRIES AVL.  AVE.C. UTIL. RETRY DELAY
 NAC                  7    1   0     7     1208   1    5.418  0.774    0    0

FEC XN   PRI         BDT      ASSEM  CURRENT  NEXT  PARAMETER    VALUE
  1105    0      101945.426   1105      5      6
  1102    0      101970.637   1102      5      6
  1109    0      101993.030   1109      0      1
  1104    0      102042.379   1104      5      6
  1106    0      102197.803   1106      5      6
  1108    0      102221.671   1108      5      6
  1107    0      102279.501   1107      5      6
  1103    0      102338.769   1103      0     13
```

16. Арифметические выражения в GPSS World.

В ранних версиях GPSS, были запрещено использовать выражения в качестве операндов, и поэтому там приходилось пользоваться командами

VARIABLE, FVARIABLE и BVARIABLE, которые описывали выражения целого, вещественного и логического типа соответственно.

В GPSS World можно использовать выражения в качестве операндов, однако в этом случае они должны быть заключены в круглые скобки. В принципе, теперь можно использовать как выражения числового, так и строкового типов, а приведение типов, как правило, берет на себя система. На практике, использование строк ограничено операторами работы с файлами, а также операторами ASSIGN и SAVEVALUE. Мы же ограничимся пока числовыми выражениями.

Вид выражений во многом аналогичен выражениям существующих языков программирования. В выражении используют стандартные числовые атрибуты, метки, константы, знаки операций, вызовы библиотечных функций, и круглые скобки. Выражения должны составляться по правилам элементарной алгебры. Выражения вычисляются слева направо с учетом приоритетов операций.

Вычисления ведутся с двойной точностью. Ниже приводятся операции и библиотечные функции, используемые в выражениях.

- знак минус;

^ - оператор возведения в степень;

'NOT' - логическое отрицание: 1 (TRUE), если операнд 0;

0 (FALSE), если не ноль;

'AND' - оператор логического умножения: 1 (TRUE), если оба операнда не нулевые,0(FALSE) - в противном случае;

'OR' - оператор логического сложения: 1 (TRUE), если один из операндов не нулевой, 0 (FALSE) - в противном случае;

'G' или > - оператор отношения "больше";

'L' или < - оператор отношения "меньше";

'E' или = - оператор отношения "равно";

'NE' или /= - оператор отношения "не равно";

'LE' или <= - оператор отношения "меньше или равно";

'GE' или >= - оператор отношения "больше или равно";

ОБРАТИТЬ ВНИМАНИЕ !!!

- оператор арифметического умножения;

/ - оператор арифметического деления;

\ - оператор деления нацело;

@ - оператор деления по модулю;

+ - оператор арифметического сложения;

- - оператор арифметического вычитания.

Элементарные функции.

ABS( ) - абсолютное значение операнда;

ATN( ) - арктангенс операнда, результат выражен в радианах;

COS( ) - косинус операнда в радианах;

INT( ) - целая часть;

EXP( ) - экспонента операнда;

LOG( ) - натуральный логарифм операнда;

SIN( ) - синус операнда в радианах;

SQR( ) - квадратный корень из операнда;

TAN( ) - тангенс операнда в радианах;

Порядок действий соответствует следующим приоритетам:

Возведение в степень.

Умножение, деление, остаток.

Сложение и вычитание.

Отношения

Логическое и.

Логическое или.

Кроме выше приведенных функций, имеется большая группа встроенных функций, связанных со статистикой и функции типа запросов.

17. Арифметические переменные VARIABLE FVARIABLE и BVARIABLE.

Для того, чтобы упростить запись операторов, содержащих сложные выражения, можно использовать команды VARIABLE FVARIABLE и BVARIABLE. Эти команды имеют вид:

имя переменной FVARIABLE выражение
имя переменной VARIABLE выражение
имя переменной BVARIABLE выражение

Пример задания выражений:

vv1 FVARIABLE q$och+3.2#(normal(1,80,10)

Для того, чтобы вычислить выражение, нужно записать обращение к выражению, то есть использовать конструкцию типа V$имя , которая заставляет вычисляться указанное выражение.

Например:

ADVANCE V$vv1

Команды FVARIABLE и VARIABLE на самом деле означают одно и то же. Они вычисляют выражение как вещественное число двойной точности. Команда BVARIABLE - преобразует результат вычисления к логическому типу, то есть возвращает 0, если выражение равно 0, иначе ее значение равно 1. Сами выражения в любом случае должны иметь числовой результат. Для обращения к логическим выражениям, используется конструкция типа BV$имя.

При использовании эквивалентностей, вместо имени выражения можно указывать его номер.

В выражениях можно использовать любые стандартные числовые атрибуты (СЧА) и числа, в частности, в выражении можно ссылаться на другие выражения.

Пример: пусть в систему поступают заявки, со средним интервалом 100, распределенным по экспоненциальному закону. Эти заявки обслуживаются в одноканальном устройстве с интервалом, который определяется величиной, 120-10*<текущая длина очереди>. Промоделировать обслуживание заявок в течение 100000 единиц времени.

Текст такой модели может выглядеть следующим образом:

```
generate (exponential(1,0,100))
queue och
seize unit1
depart och
advance (120-10#q$och)
release unit1
terminate
generate 100000
terminate 1
start 1
```

Пример:

Пусть в аналогичной задаче, время обслуживания равно 120, если очередь меньше 5, и равно 90, если очередь больше 4.

Тогда текст модели приобретет, например, вид:

```
tt table m1,0,30,10
llog bvariable q$och<5
 generate (exponential(1,0,100))
 queue och
 seize unit1
 depart och
mark
 advance (120#bv$llog+90#('not'bv$llog))
tabulate tt
 release unit1
 terminate
 generate 100000
 terminate 1
 start 1
```

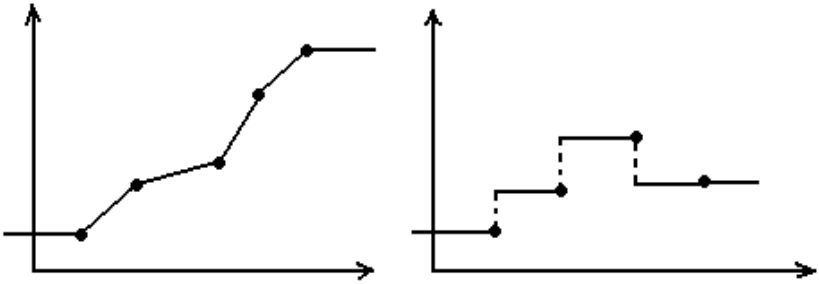
18. Функции в GPSS.

Для задания пользовательских функций с помощью точек или выражений, и некоторых других целей, используются функции. Напоминаем, что если в блоках ADVANCE A,B и GENERATE A,B второй параметр функция, то время обслуживания вычисляется в соответствии с выражением: T=A*B

Вызов функции выполняется с помощью следующей конструкцией: FNномер или FN$имя.

В GPSS имеются несколько разновидностей функции. Чаще всего используются непрерывная и дискретная функции.

График непрерывной функции: График дискретной функции:



Функция описывается следующим образом:

имя_функции FUNCTION аргумент, <тип>количество точек

аргумент1,ф-ция1/аргумент2,ф-ция2/...

Для непрерывных функций тип будет C, а для дискретных тип будет D. Каждая пара чисел описывает одну точку графика. Точки должны разделяться символом / или переводом строки. Аргументы в описании функции непременно должны быть числами, а значения аргументов должны возрастать от точки к

точке. Число точек в заголовке функции должно соответствовать числу описанных точек.

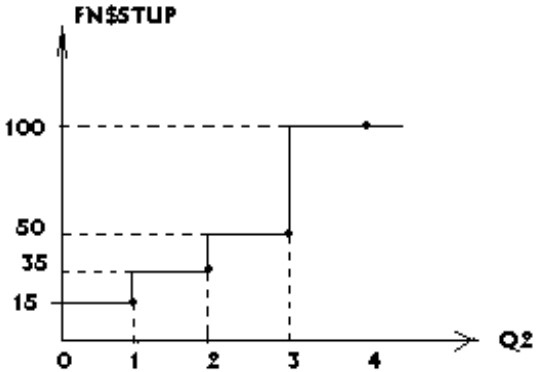
```
10  stup FUNCTION Q2 D4
1,15/2,35/3,50/4,100
```

здесь аргументом служит Q2 -вторая очередь.

Пример применения функции:

```
ADVANCE FN$stup
```

График к примеру:



Значения функций вычисляется следующим образом. Если аргумент имеет значение, меньшее, чем в первой точке, то функция имеет то же значение, что и в первой точке. Если аргумент имеет значение, большее, чем в последней точке, то функция имеет то же значение, что и в последней точке. В остальных случаях значение функций вычисляется по-разному, в зависимости от их вида.

Для непрерывных функций, если аргумент лежит в интервале между двумя точками, то значение функции определяется с помощью линейной интерполяции.

Для дискретных функций, если значение аргумента лежит в интервале между двумя точками, то значение функции будет равно значению в правом конце интервала. Причем правый конец входит в интервал, а левый - нет.

Часто функции используются таким образом, что в качестве аргумента используется генератор случайных чисел: RN1, RN2, … RN7 и так далее. Если он используется как аргумент функции, то он генерирует вещественные числа из диапазона (0..1). В остальных случаях генератор случайных чисел генерирует целые числа из диапазона 0..999 .

Для задания случайной величины, распределенной по известному закону, всегда используется интегральная функция распределения. Так, например.

Пусть интервал поступления заявок в одноканальное устройство равен в среднем 110 и распределен по экспоненциальному закону.

Пусть время обслуживания для 10% заявок равно 100, для 20% заявок оно равно 80, для 40% заявок оно равно 120, а для остальных 30% - время обслуживания равно 60.

Тогда модель работы такого устройства может иметь вид.

```
tt table m1,0,10,20
inter function RN7,D4
0.1,100/0.3,80/0.7,120/1,60
generate (exponential(1,0,110))
queue och
seize unit1
depart och
mark
advance fn$inter
tabulate tt
release unit1
terminate
generate 100000
terminate 1
start 1
```

Рассматривая тексте этой модели можно заметить, что функция inter задает интегральное распределение в соответствии с заданием, и сама функция является дискретной.

Другие возможные виды функций – следующие:

- дискретная атрибутивная - E;
- табличная числовая - L;
-табличная атрибутивная - M.

Функция типа E. В этих функциях, как и в функциях типа D, определяется интервал, к которому принадлежит значение аргумента. Вторым значением для описания конечной точки интервала - должно быть в общем случае некоторое выражение, которое вычисляется, и считается значением функции на интервале.

Например, пусть интервал поступления заявок в одноканальное устройство равен в среднем 100 и распределен по экспоненциальному закону.

Пусть время обслуживания заявок зависит от длины очереди, причем, для очереди, длина которой меньше либо равна 2, время обслуживания выражается формулой, 120 - <длина очереди>#5.

Если длина очереди меньше либо равна 7, время обслуживания выражается формулой, 110 - <длина очереди>#8.

Если длина очереди меньше либо равна 11, время обслуживания выражается формулой, 100 - <длина очереди>#6. Очевидно, в соответствии с описанием функции, при длине очереди более 11, так же действует последняя формула.

Модель работы такого устройства может иметь вид.

```
tt table m1,0,10,20
tt1 table q$och,0,1,20
inter function Q$och,E3
2,(120-Q$och#5)/ 7,(110-Q$och#8)/ 11,(100-Q$och#6)
generate (exponential(1,0,100))
queue och
```

```
seize unit1
depart och
mark
tabulate tt1
advance fn$inter
tabulate tt
release unit1
terminate

generate 100000
terminate 1
start 1
```

Табличная числовая функция – тип L .

Функции типа L и M вычисляются быстрее своих аналогов. В функции типа L , аргумент для точек должен быть набором целых чисел, идущих подряд, начиная с 1. Значения функции для точек должны быть числами или метками. Если аргумент функции окажется меньше 1 или слишком большим, то возникает ошибка времени исполнения модели.

Пусть в предыдущей задаче, время обслуживания для каждой длины очереди описывается таблицей.

Длина	Время
0	105
1	100
2	90
3	80
4	70
5	40
6	20

Тогда текст модели может выглядеть следующим образом:

```
 tt table m1,0,10,20
tt1 table q$och,0,1,20
inter function (Q$och+1),L7
1,105/2,100/3,90/4,80/5,70/6,40/7,20
generate (exponential(1,0,100))
queue och
seize unit1
depart och
mark
tabulate tt1
advance fn$inter
tabulate tt
release unit1
terminate
```

```
generate 100000
terminate 1
start 1
```

Табличная атрибутивная функция – типа M.

Эта функция сочетает в себе свойства функции типа L и функции типа E. В этой функции, аргументом для точек должен быть набор целых чисел, идущих подряд, начиная с 1. Значения функции для точек должны быть в общем случае, выражениями. Если аргумент функции окажется меньше 1 или слишком большим, то возникает ошибка времени исполнения модели.

Функции двух последних типов, не могут быть со случайными аргументами.

19. Табулирование переменных.

В GPSS – World можно табулировать значения стандартных числовых атрибутов и выражений.

Например: табулировать распределение длины очереди, табулировать распределение времени жизни заявки и так далее.

Пусть имеется некоторое выражение. Возможные значения этого выражения разобьем на ряд отрезков одинаковой длины. Получим интервал от [-бесконечности до 1точки] и от [последней точки до +бесконечности] - эти интервалы называются интервалами переполнения.

Все остальные интервалы будут одинаковыми.

При проходе заявки через блок tabulate, который наблюдает за значениями данного выражения, определяется, в какой интервал попадает значение выражения. При этом корректируется таблица, которая фиксирует, сколько раз значение попало в каждый из интервалов, включая два бесконечные. Собранная информация выводится в итоговой статистике. Фактически при табулировании получается ненормированная гистограмма табулируемой величины.

Блок tabulate имеет следующий вид.

Tabulate <имя таблицы>,<добавляемая величина>

В блоке Tabulate указывается номер или имя таблицы, которая выполняет табулирование и, возможно, добавляемое в интервал значение. (По умолчанию в интервал добавляется единица.)

Сама таблица описывается командой TABLE. Вид этой команды.

имя TABLE A,B,C,D

здесь A – это табулируемая величина;

B - это первое граничное значение для интервалов;

C- ширина интервала;

D - количество интервалов, включая два бесконечных.

Примеры использования таблиц мы уже рассматривали. С помощью пунктов меню Window, Simulation window, Table Window – можно просмотреть любую таблицу как гистограмму.

Нижняя граница частотного класса всегда включается в предыдущий частотный класс.

Другим примером использования табулирования может быть команда QTABLE, которая обеспечивает табулирование времени, проводимого заявкой в очереди. Команда имеет вид:

Имя_таблицы QTABLE A,B,C,D

Здесь A – имя или номер очереди, а параметры B,C и D – аналогичны в командах TABLE и QTABLE.

В отличие от команды TABLE, которая является описательной, команда QTABLE сама по себе вызывает формирование соответствующей таблицы, и не требует блоков TABULATE. В принципе, команда QTABLE не является обязательной. Ее работу можно выполнить, используя обычные таблицы.

Пусть в предыдущей задаче, необходимо протабулировать время, проводимое заявками в очереди och. Тогда текст модели приобретет вид:

```
tt Qtable och,0,10,20
inter function (Q$och+1),L7
1,105/2,100/3,90/4,80/5,70/6,40/7,20
generate (exponential(1,0,100))
queue och
seize unit1
depart och
mark
advance fn$inter
release unit1
terminate

generate 100000
terminate 1
start 1
```

А итоговая статистика будет иметь вид:

```
    GPSS World Simulation Report - Untitled Model 1.2.3

        Saturday, April 05, 2003 07:04:23

 START TIME          END TIME  BLOCKS  FACILITIES  STORAGES
      0.000        100000.000    10        1          0
    NAME                      VALUE
INTER                       10002.000
OCH                         10001.000
TT                          10000.000
UNIT1                       10003.000

LABEL             LOC  BLOCK TYPE     ENTRY COUNT CURRENT COUNT RETRY
                    1  GENERATE           983             0       0
                    2  QUEUE              983             1       0
                    3  SEIZE              982             0       0
                    4  DEPART             982             0       0
                    5  MARK               982             0       0
                    6  ADVANCE            982             1       0
                    7  RELEASE            981             0       0
                    8  TERMINATE          981             0       0
                    9  GENERATE             1             0       0
                   10  TERMINATE            1             0       0


FACILITY         ENTRIES  UTIL.   AVE. TIME AVAIL. OWNER PEND INTER RETRY DELAY
 UNIT1              982    0.894      91.002  1      983    0    0     0      1
```

```
QUEUE              MAX CONT. ENTRY ENTRY(0) AVE.CONT. AVE.TIME   AVE.(-0) RETRY
 OCH                 7    1    983    111     1.654    168.223    189.636   0

TABLE              MEAN    STD.DEV.       RANGE            RETRY FREQUENCY CUM.%
 TT              168.390  108.649                           0
                                          -        0.000          111    11.30
                                   0.000  -       50.000           63    17.72
                                  50.000  -      100.000          118    29.74
                                 100.000  -      150.000          146    44.60
                                 150.000  -      200.000          143    59.16
                                 200.000  -      250.000          144    73.83
                                 250.000  -      300.000          130    87.07
                                 300.000  -      350.000           88    96.03
                                 350.000  -      400.000           36    99.69
                                 400.000  -      450.000            3   100.00


FEC XN   PRI         BDT      ASSEM  CURRENT  NEXT  PARAMETER    VALUE
   983    0      100065.704    983      6      7
   985    0      100169.156    985      0      1
   986    0      200000.000    986      0      9
```

Обратите внимание, что в итоговой статистике есть таблица, табулирующая время, проведенное заявками в очереди, хотя в тексте модели нет блока TABULATE.

20. Блоки присваивания в GPSS.

В GPSS имеется широкий набор блоков присваивания, так как каждый тип данных, имеет свой блок присваивания.

Для задания приоритета заявки используется следующий блок.

PRIORITY A

Здесь A – это новый приоритет активной заявки.

После присваивания активной заявке нового приоритета, ее продвижение по модели прекращается, она устанавливается последней в своем приоритетном классе, и просмотр списка текущих событий начинается сначала.

Заявки с более высоким приоритетом, во всех устройствах принимаются на обслуживание раньше заявок с более низким приоритетом. Минимальный приоритет (по умолчанию) равен 0. Приоритет должен быть целым и положительным. Не рекомендуются слишком большие значения приоритетов, в особенности с пропусками.

Пример

Пусть имеется два потока заявок, обслуживаемых в двухканальном устройстве. Заявки первого типа требуют для обслуживания одного канала, а второго – двух. Интервалы поступления и обслуживания заявок распределены по экспоненциальному закону со средними значениями равными.

Для заявок первого типа – интервал поступления – 145 мс., обслуживания – 25мс.

Для заявок второго типа – интервал поступления – 100 мс., обслуживания – 100мс. Заявки второго типа обслуживаются с приоритетом.

Промоделировать работу системы в течение 100000 мс.

Текст такой модели может иметь вид:

```
int1 equ 45
Wint1 equ 25
```

```
ust storage 2
generate (exponential(1,0,int1))
queue ust
enter ust
depart ust
queue ust1
advance (exponential(1,0,Wint1))
depart ust1
leave ust
terminate

int2 equ 100
Wint2 equ 100
generate (exponential(1,0,int2))
priority 1
queue ust
enter ust,2
depart ust
queue ust2
advance (exponential(1,0,Wint2))
depart ust2
leave ust,2
terminate

generate 100000
terminate 1
start 1
```

Здесь для сбора более подробной статистики, используются формальные очереди ust1 и ust2.

В GPSS имеются параметры заявки, которые доступны только самой заявке (ее P – параметры). Для присваивания этим параметрам значений используются блоки

ASSIGN A,B,C

Здесь A – имя или номер P – параметра.

B – числовое или строковое значение.

C – номер или имя функции.

Если параметр C отсутствует, то выполнение блока происходит следующим образом.

Вычисляется значение B и присваивается данному P –параметру. Параметр A может заканчиваться значком + или - , тогда значение B добавляется к P – параметру, или вычитается из него.

Если имеется параметр C, то значение B умножается на значение соответствующей функции, после чего, результат используется для изменения значения P - параметра по рассмотренным выше правилам.

Обратите внимание, что и функция, и P – параметр задаются не своими обозначениями, а именами или номерами.

Например,

ASSIGN num+,(Q$och+30),3

Означает, что нужно вычислить величину (Q$och+30) и умножить ее на FN3, то есть на 3 – ю функцию, а затем результат добавить к значению P$num.

Если некоторый P- параметр не существовал, то при первом выполнении блока ASSIGN, он создается с нулевым значением, причем это значение тут же может использоваться в выражении.

Примечание: не существующий P- параметр нельзя использовать в выражениях. Значение P – параметра может быть и строковым.

В блоках ASSIGN третий параметр используется крайне редко и оставлен, в основном, для совместимости со старыми версиями GPSS.

Получение значения P- параметра возможно с помощью конструкции Pномер или P$имя.

Кроме параметров заявки, в GPSS имеются параметры системы, так называемые X –параметры, или ячейки.

Ячейки доступны любой заявке. Для присваивания ячейкам значений используются блоки

SAVEVALUE A,B

В этих блоках, параметры A и B – аналогичны параметрам A и B в блоках ASSIGN.

Получение значения ячейки возможно с помощью конструкции Xномер или X$имя. Ячейка может иметь и строковое значение. Рассмотрим примеры использования блоков ASSIGN и SAVEVALUE.

Пусть в систему поступают заявки со средним интервалом 100 мс., распределенным по экспоненциальному закону. Заявка может равновероятно занять любое из 5 устройств, где она будет обслуживаться за время, равное 490+-90 мс.

Промоделировать работу системы, в течение 10000000 мс. Определить суммарное время задержки заявок в очереди, к каждому устройству.

Для задания номера устройства и очереди, которые выбраны для обслуживания, будем использовать P – параметры.

Тогда можно получить следующую модель.

```
int1 equ 100
Wint equ 490
generate (exponential(1,0,int1))
assign num_u,(duniform(1,1,5))
queue P$num_u
seize P$num_u
depart P$num_u
savevalue P$num_u+,m1
advance Wint,90
release P$num_u
```

```
terminate

generate 10000000
terminate 1
start 1
```

Итоговая статистика модели будет иметь вид.

```
            GPSS World Simulation Report - Untitled Model 1.7.1

                  Saturday, April 05, 2003 19:23:56

             START TIME            END TIME  BLOCKS  FACILITIES  STORAGES
                  0.000       10000000.000    11        5          0

                NAME                        VALUE
            INT1                          100.000
            NUM_U                       10002.000
            WINT                          490.000

 LABEL              LOC  BLOCK TYPE     ENTRY COUNT CURRENT COUNT RETRY
                     1    GENERATE          99660             0       0
                     2    ASSIGN            99660             0       0
                     3    QUEUE             99660           114       0
                     4    SEIZE             99546             0       0
                     5    DEPART            99546             0       0
                     6    SAVEVALUE         99546             0       0
                     7    ADVANCE           99546             5       0
                     8    RELEASE           99541             0       0
                     9    TERMINATE         99541             0       0
                    10    GENERATE              1             0       0
                    11    TERMINATE             1             0       0


FACILITY          ENTRIES  UTIL.   AVE. TIME AVAIL. OWNER PEND INTER RETRY DELAY
 1                 19883    0.974     489.868  1    99559    0     0     0     23
 2                 19790    0.969     489.416  1    99521    0     0     0     27
 3                 19949    0.976     489.337  1    99535    0     0     0     22
 4                 19978    0.979     490.061  1    99511    0     0     0     25
 5                 19946    0.977     490.014  1    99590    0     0     0     17


QUEUE              MAX CONT. ENTRY ENTRY(0) AVE.CONT. AVE.TIME   AVE.(-0) RETRY
 1                  64   23  19906    527    14.202   7134.708   7328.732   0
 2                  52   27  19817    633    11.691   5899.475   6094.136   0
 3                  99   22  19971    453    24.928  12482.249  12771.954   0
 4                  71   25  20003    412    17.625   8811.313   8996.616   0
 5                  66   17  19963    472    13.366   6695.233   6857.367   0


SAVEVALUE               RETRY       VALUE
 1                        0   141893983.067
 2                        0   116698712.232
 3                        0   249145247.795
 4                        0   176075738.905
 5                        0   133596092.206


FEC XN   PRI         BDT      ASSEM  CURRENT  NEXT  PARAMETER    VALUE
 99590    0    10000016.466   99590      7      8
                                                   NUM_U          5.000
 99535    0    10000018.926   99535      7      8
                                                   NUM_U          3.000
 99521    0    10000060.295   99521      7      8
                                                   NUM_U          2.000
 99662    0    10000100.517   99662      0      1
 99559    0    10000198.798   99559      7      8
                                                   NUM_U          1.000
```

```
 99511    0    10000239.275   99511      7      8
                                                    NUM_U         4.000
 99663    0    20000000.000   99663      0     10
```

Нетрудно увидеть, что заявки действительно равновероятно поступают в каждое из устройств. Однако итоговая статистика достаточно неустойчива по средней величине очереди. Это происходит по причине высокой загрузки устройств. Если средний интервал поступления заявок увеличить до 120, то получим следующую статистику.

```
                GPSS World Simulation Report - Untitled Model 1.8.1

                     Saturday, April 05, 2003 19:32:55

           START TIME           END TIME  BLOCKS  FACILITIES  STORAGES
                0.000       10000000.000    11         5          0

              NAME                      VALUE
          INT1                        120.000
          NUM_U                     10002.000
          WINT                        490.000

 LABEL              LOC  BLOCK TYPE     ENTRY COUNT CURRENT COUNT RETRY
                     1    GENERATE         82942             0       0
                     2    ASSIGN           82942             0       0
                     3    QUEUE            82942            15       0
                     4    SEIZE            82927             0       0
                     5    DEPART           82927             0       0
                     6    SAVEVALUE        82927             0       0
                     7    ADVANCE          82927             5       0
                     8    RELEASE          82922             0       0
                     9    TERMINATE        82922             0       0
                    10    GENERATE             1             0       0
                    11    TERMINATE            1             0       0

FACILITY          ENTRIES  UTIL.   AVE. TIME AVAIL. OWNER PEND INTER RETRY DELAY
 1                 16636    0.814     489.052  1    82919    0     0     0      4
 2                 16637    0.815     489.646  1    82881    0     0     0     10
 3                 16512    0.809     490.007  1    82941    0     0     0      0
 4                 16470    0.808     490.406  1    82939    0     0     0      1
 5                 16672    0.816     489.709  1    82938    0     0     0      0


QUEUE              MAX CONT. ENTRY ENTRY(0) AVE.CONT. AVE.TIME   AVE.(-0) RETRY
 1                  17    4  16640   3076     1.803   1083.454   1329.157   0
 2                  23   10  16647   3064     1.750   1051.022   1288.107   0
 3                  16    0  16512   3115     1.728   1046.707   1290.082   0
 4                  23    1  16471   3114     1.736   1053.813   1299.495   0
 5                  28    0  16672   3121     1.956   1173.150   1443.344   0


SAVEVALUE               RETRY       VALUE
 1                        0   18020302.930
 2                        0   17470221.213
 3                        0   17283232.504
 4                        0   17357129.563
 5                        0   19558753.972


FEC XN   PRI         BDT      ASSEM  CURRENT  NEXT  PARAMETER    VALUE
 82944    0    10000031.600   82944      0      1
 82881    0    10000071.877   82881      7      8
                                                    NUM_U         2.000
 82919    0    10000142.439   82919      7      8
                                                    NUM_U         1.000
 82941    0    10000273.776   82941      7      8
                                                    NUM_U         3.000
 82938    0    10000521.644   82938      7      8
```

```
                                                NUM_U          5.000
 82939    0    10000557.382   82939      7      8
                                                NUM_U          4.000
 82945    0    20000000.000   82945      0     10
```

Видоизменим модель, с тем, чтобы получить среднеквадратичную величину времени в очереди для каждого устройства.

Тогда модель приобретет следующий вид.

```
int1 equ 120
Wint equ 490
generate (exponential(1,0,int1))
assign num_u,(duniform(1,1,5))
queue P$num_u
seize P$num_u
depart P$num_u
savevalue P$num_u+,(m1^2)
assign 1,(P$num_u+10),
savevalue p1+,1
savevalue (P$num_u+20),(sqr(X*num_u/X*1))
advance Wint,90
release P$num_u
terminate

generate 10000000
terminate 1
start 1
```

Данная модель нуждается в серьезных пояснениях.

Здесь в строке

```
savevalue P$num_u+,(m1^2)
```

происходит суммирование квадратов времен в каждой из 5 очередей.

В строке

```
assign 1,(P$num_u+10),
```

происходит занесение в параметр p1 величины, равной номеру очереди плюс 10, для того, чтобы можно было в ячейках от X11 до X15 накапливать число заявок, отстоявших в каждой из пяти очередей. В строке

```
savevalue p1+,1
```

происходит собственно накопление числа заявок для каждой из очередей. И, наконец, в строке

```
savevalue (P$num_u+20),(sqr(X*num_u/X*1))
```

в ячейках от X21 до X25 происходит вычисление среднеквадратичного времени в очереди для каждой из очередей.

Здесь X*Num_u и X*1 это значения X – параметров с номерами P$Num_u и P1.

Выходная статистика для этой модели будет иметь следующий вид.

```
    GPSS World Simulation Report - Untitled Model 1.13.1

         Saturday, April 05, 2003 19:51:49

 START TIME          END TIME  BLOCKS  FACILITIES  STORAGES
```

```
          0.000       10000000.000    14        5          0

          NAME                        VALUE
       INT1                          120.000
       NUM_U                       10002.000
       WINT                          490.000

 LABEL              LOC  BLOCK TYPE     ENTRY COUNT CURRENT COUNT RETRY
                     1   GENERATE          82942             0       0
                     2   ASSIGN            82942             0       0
                     3   QUEUE             82942            15       0
                     4   SEIZE             82927             0       0
                     5   DEPART            82927             0       0
                     6   SAVEVALUE         82927             0       0
                     7   ASSIGN            82927             0       0
                     8   SAVEVALUE         82927             0       0
                     9   SAVEVALUE         82927             0       0
                    10   ADVANCE           82927             5       0
                    11   RELEASE           82922             0       0
                    12   TERMINATE         82922             0       0
                    13   GENERATE              1             0       0
                    14   TERMINATE             1             0       0


FACILITY         ENTRIES  UTIL.   AVE. TIME AVAIL. OWNER PEND INTER RETRY DELAY
 1                16636    0.814     489.052  1    82919    0     0     0      4
 2                16637    0.815     489.646  1    82881    0     0     0     10
 3                16512    0.809     490.007  1    82941    0     0     0      0
 4                16470    0.808     490.406  1    82939    0     0     0      1
 5                16672    0.816     489.709  1    82938    0     0     0      0


QUEUE              MAX CONT. ENTRY ENTRY(0) AVE.CONT. AVE.TIME   AVE.(-0) RETRY
 1                  17    4  16640   3076     1.803   1083.454   1329.157   0
 2                  23   10  16647   3064     1.750   1051.022   1288.107   0
 3                  16    0  16512   3115     1.728   1046.707   1290.082   0
 4                  23    1  16471   3114     1.736   1053.813   1299.495   0
 5                  28    0  16672   3121     1.956   1173.150   1443.344   0


SAVEVALUE               RETRY       VALUE
Суммы квадратов времен в очереди для каждой из 5 очередей.

 1                       0   44706746176.019
 2                       0   42308134314.083
 3                       0   40530952188.711
 4                       0   45326635347.821
 5                       0   61574910893.441
Число заявок простоявших в каждой из 5 очередей.

 11                      0      16636.000
 12                      0      16637.000
 13                      0      16512.000
 14                      0      16470.000
 15                      0      16672.000
Среднеквадратичное время пребывания заявок в каждой из 5 очередей.

 21                      0       1639.314
 22                      0       1594.683
 23                      0       1566.728
 24                      0       1658.937
 25                      0       1921.799


FEC XN   PRI         BDT      ASSEM  CURRENT  NEXT  PARAMETER    VALUE
 82944    0    10000031.600   82944      0      1
 82881    0    10000071.877   82881     10     11        1         12.000
                                                    NUM_U         2.000
 82919    0    10000142.439   82919     10     11        1         11.000
                                                    NUM_U         1.000
```

```
82941    0    10000273.776   82941     10     11       1          13.000
                                                  NUM_U          3.000
82938    0    10000521.644   82938     10     11       1          15.000
                                                  NUM_U          5.000
82939    0    10000557.382   82939     10     11       1          14.000
                                                  NUM_U          4.000
82945    0    20000000.000   82945      0     13
```

Даже предыдущий пример, показывает, что использование ячеек не всегда удобно. Более удобным способом работы с нумерованными значениями являются матрицы. В принципе, для многих задач больше подошли бы линейные массивы, но таковые в GPSS не предусмотрены.

Описание матриц выполняет команда:

имя матрицы MATRIX A,B,C

Каждая матрица должна иметь команду, ее описывающую.

Поле A не используется (оставлено для совместимости с более старыми версиями GPSS).

В поле B задается число строк матрицы, в поле C - число столбцов. Операнды B и C могут быть положительными целыми числами.

Например, оператор описания

PRINT MATRIX ,4,3

определяет матрицу размером в 4 строки и 3 столбца. Полученная матрица определяется как матрица PRINT .Команда MATRIX создает матрицу в текущей модели.

Матрица не может быть удалена из текущей модели. Матрица может быть переопределена или инициализирована повторно другой командой MATRIX с той же меткой. Переопределение, при котором размер матрицы изменяется, вызывает выделение памяти под новую матрицу.

В принципе, количество измерений в матрице может быть и больше, (до шести) но последующие индексы будут доступны только при использовании процедур на языке PLUS. А в блоке MSAVEVALUE, может быть ровно два индекса, остальные индексы, если они есть, полагаются равными 1. Начальные значения всех элементов матрицы, первоначально, равны 0.

Обращение к элементам матрицы происходит с помощью конструкции MXномер(m,n) или MX$имя(m,n). Здесь m и n - это индексы в матрице.

Блок MSAVEVALUE используется для записи значений в матрицы, а также для увеличения или уменьшения значений, записанных в матрицах. Он имеет следующий вид:

MSAVEVALUE A,B,C,D

В операнде A задается имя матрицы. Крайним правым символом операнда может быть знак "+", если блок MSAVEVALUE работает в режиме добавления и "-" – для режима уменьшения.

В операнде B задается номер строки матрицы,

В операнде C - номер столбца.

Таким образом, операнды B и C определяют относительное расположение ячеек матрицы, содержимое которых изменяется. Номера ячеек матрицы начинаются с 1.

Аргумент операнда D определяет значение, которое должно сохраниться в соответствующем элементе матрицы. Это значение, может быть добавлено (режим добавления) или вычтено (режим уменьшения) из элемента матрицы.

Например, блок

MSAVEVALUE INV,2,3,4

записывает константу 4 в матрицу INV (строка 2, столбец 3).

Когда заявка входит в блок MSAVEVALUE, то анализируется операнд A и используется матрица с указанным именем. Если матрица не создана, то возникает ошибка.

Соответствующий элемент матрицы определяется содержимым операндов B и C. Если такого элемента не существует, возникает ошибка.

Пример: решение предыдущей задачи с использованием матрицы.

```
matr  matrix ,3,5
int1 equ 120
Wint equ 490
generate (exponential(1,0,int1))
assign num_u,(duniform(1,1,5))
queue P$num_u
seize P$num_u
depart P$num_u
msavevalue matr+,1,P$num_u,(m1^2)
msavevalue matr+,2,P$num_u,1
msavevalue
matr,3,P$num_u,(sqr(mx$matr(1,P$num_u)/mx$matr(2,P$num_u)))
advance Wint,90
release P$num_u
terminate

generate 10000000
terminate 1
start 1
```

Итоговая статистика по модели будет иметь вид:

```
    GPSS World Simulation Report - Untitled Model 1.4.1

         Sunday, April 06, 2003 20:28:41

 START TIME           END TIME  BLOCKS  FACILITIES  STORAGES
      0.000       10000000.000    13        5          0

    NAME                      VALUE
INT1                        120.000
MATR                      10000.000
NUM_U                     10003.000
WINT                        490.000

LABEL             LOC  BLOCK TYPE     ENTRY COUNT CURRENT COUNT RETRY
                  1    GENERATE         82942             0       0
                  2    ASSIGN           82942             0       0
```

```
                    3    QUEUE            82942            15       0
                    4    SEIZE            82927             0       0
                    5    DEPART           82927             0       0
                    6    MSAVEVALUE       82927             0       0
                    7    MSAVEVALUE       82927             0       0
                    8    MSAVEVALUE       82927             0       0
                    9    ADVANCE          82927             5       0
                   10    RELEASE          82922             0       0
                   11    TERMINATE        82922             0       0
                   12    GENERATE             1             0       0
                   13    TERMINATE            1             0       0

FACILITY         ENTRIES  UTIL.   AVE. TIME AVAIL. OWNER PEND INTER RETRY DELAY
 1                 16636    0.814     489.052  1    82919    0     0     0      4
 2                 16637    0.815     489.646  1    82881    0     0     0     10
 3                 16512    0.809     490.007  1    82941    0     0     0      0
 4                 16470    0.808     490.406  1    82939    0     0     0      1
 5                 16672    0.816     489.709  1    82938    0     0     0      0


QUEUE              MAX CONT. ENTRY ENTRY(0) AVE.CONT. AVE.TIME   AVE.(-0) RETRY
 1                  17    4  16640   3076     1.803   1083.454   1329.157   0
 2                  23   10  16647   3064     1.750   1051.022   1288.107   0
 3                  16    0  16512   3115     1.728   1046.707   1290.082   0
 4                  23    1  16471   3114     1.736   1053.813   1299.495   0
 5                  28    0  16672   3121     1.956   1173.150   1443.344   0


MATRIX             RETRY     INDICES             VALUE
 MATR              0
                               1 1               44706746176.019
                               1 2               42308134314.083
                               1 3               40530952188.710
                               1 4               45326635347.821
                               1 5               61574910893.441
                               2 1               16636
                               2 2               16637
                               2 3               16512
                               2 4               16470
                               2 5               16672
                               3 1               1639.313
                               3 2               1594.683
                               3 3               1566.727
                               3 4               1658.937
                               3 5               1921.799


FEC XN   PRI         BDT      ASSEM  CURRENT  NEXT  PARAMETER    VALUE
 82944    0    10000031.600   82944      0      1
 82881    0    10000071.877   82881      9     10
                                                    NUM_U          2.000
 82919    0    10000142.439   82919      9     10
                                                    NUM_U          1.000
 82941    0    10000273.776   82941      9     10
                                                    NUM_U          3.000
 82938    0    10000521.644   82938      9     10
                                                    NUM_U          5.000
 82939    0    10000557.382   82939      9     10
                                                    NUM_U          4.000
 82945    0    20000000.000   82945      0     12
```

В этой модели первая строка матрицы используется для накопления суммы квадратов времен в очереди для каждой из очередей. Вторая строка – используется для определения числа заявок, покинувших каждую из очередей. Третья строка - используется для определения среднеквадратичных значений времени в каждой из очередей. Очевидно, что такая модель выглядит более логичной и понятной.

Здесь, правда, надо иметь в виду, что по умолчанию, значения элементов матрицы не попадают в выходную статистику. Чтобы они туда попали, нужно войти в пункты меню Edit, Settings, в появившемся окне перейти на вкладку Reports и в ней поставить флажок в строке Matrixes, затем щелкнуть по кнопкам Применить и ОК.

В GPSS имеются логические ключи, которые с одной стороны, могут рассматриваться как ячейки, со значениями 0 или 1, а с другой стороны, как элементы оборудования.

Блоком присваивания для ключей является блок

LOGIC режим A

Здесь A – это номер или имя ключа.

Режим может быть S, R, или I.

Режим S обеспечивает запись в ключ значения 1.

Режим R обеспечивает запись в ключ значения 0.

Режим I обеспечивает запись в ключ инвертированного значения.

Примеры использования ключей мы рассмотрим позднее.

Доступ к значению ключа можно получить с помощью конструкции

LSномер или LS$имя .

Ячейки, матрицы и ключи первоначально имеют нулевые значения, но с помощью команды INITIAL любому из этих элементов можно сопоставить другое начальное значение.

Команда INITIAL имеет следующий формат:

INITIAL A,B

Здесь A – это обращение к ячейке, элементу матрицы или ключу, а B – начальное значение соответствующего элемента.

Примеры использования команды.

INITIAL X88,12000

Величина 12000 записывается в ячейку с номером 88.

INITIAL MX3(2,4),-33

Величина -33 записывается в строку 2, столбец 4 матрицы 3.

Для обнуления текущего времени жизни заявки, или для записи текущего абсолютного времени моделирования в P – параметр, используется блок MARK.

Этот блок имеет вид:

MARK

Или

MARK A

Здесь A – это номер или имя P – параметра.

В первой версии, блок просто обнуляет время жизни заявки, то есть параметр M1. А во второй, он заносит в соответствующий P – параметр значение текущего абсолютного времени моделирования, то есть параметр AC1.

Фактически, вторая форма могла бы быть заменена эквивалентным блоком ASSIGN.

Для доступа к интервалу времени, прошедшем с момента прохождения заявкой соответствующего блока MARK, можно использовать конструкцию:

MPномер или MP$имя . Здесь имя или номер определяет P –параметр. Фактически, значения этой величины, может быть определено с помощью выражения:

AC1- Pномер или AC1- P$имя

Чаще всего значение времени жизни используется для табулирования времени обслуживания заявок в устройствах.

Примером использования блока MARK могла бы служить следующая задача.

Заявка проходит последовательно 3 устройства, протабулировать время, проведенное заявкой в очереди к каждому устройству.

Пусть интервал поступления заявок определяется величиной 100+-90 единиц времени. Время обслуживания в каждом из устройств составляет 90+-80, 75+-70, 60+-55.

Тогда модель этой системы могла бы выглядеть следующим образом.

```
tt1 table mp$t1,0,50,100
tt2 table mp$t2,0,50,100
tt3 table mp$t3,0,50,100
generate 100,90
mark t1
queue uu1
seize uu1
depart uu1
tabulate tt1
advance 90,80
release uu1
mark t2
queue uu2
seize uu2
depart uu2
tabulate tt2
advance 75,70
release uu2
mark t3
queue uu3
seize uu3
depart uu3
tabulate tt3
advance 60,55
release uu3
terminate 1
start 1000
```

Тогда итоговая статистика будет иметь вид:

```
GPSS World Simulation Report - Untitled Model 1.10.1
```

Wednesday, August 27, 2003 18:08:48

START TIME	END TIME	BLOCKS	FACILITIES	STORAGES
0.000	97629.911	23	3	0

NAME	VALUE
T1	10003.000
T2	10005.000
T3	10007.000
TT1	10000.000
TT2	10001.000
TT3	10002.000
UU1	10004.000
UU2	10006.000
UU3	10008.000

LABEL	LOC	BLOCK TYPE	ENTRY COUNT	CURRENT COUNT	RETRY
	1	GENERATE	1002	0	0
	2	MARK	1002	0	0
	3	QUEUE	1002	0	0
	4	SEIZE	1002	0	0
	5	DEPART	1002	0	0
	6	TABULATE	1002	0	0
	7	ADVANCE	1002	0	0
	8	RELEASE	1002	0	0
	9	MARK	1002	0	0
	10	QUEUE	1002	1	0
	11	SEIZE	1001	0	0
	12	DEPART	1001	0	0
	13	TABULATE	1001	0	0
	14	ADVANCE	1001	1	0
	15	RELEASE	1000	0	0
	16	MARK	1000	0	0
	17	QUEUE	1000	0	0
	18	SEIZE	1000	0	0
	19	DEPART	1000	0	0
	20	TABULATE	1000	0	0
	21	ADVANCE	1000	0	0
	22	RELEASE	1000	0	0
	23	TERMINATE	1000	0	0

FACILITY	ENTRIES	UTIL.	AVE. TIME	AVAIL.	OWNER	PEND	INTER	RETRY	DELAY
UU1	1002	0.925	90.154	1	0	0	0	0	0
UU2	1001	0.777	75.799	1	1001	0	0	0	1
UU3	1000	0.606	59.197	1	0	0	0	0	0

QUEUE	MAX	CONT.	ENTRY	ENTRY(0)	AVE.CONT.	AVE.TIME	AVE.(-0)	RETRY
UU1	12	0	1002	121	2.591	252.408	287.075	0
UU2	9	1	1002	386	0.844	82.252	133.794	0
UU3	4	0	1000	621	0.166	16.200	42.745	0

TABLE	MEAN	STD.DEV.	RANGE			RETRY	FREQUENCY	CUM.%
TT1	252.408	215.035				0		
				-	0.000		121	12.08
			0.000	-	50.000		90	21.06
			50.000	-	100.000		103	31.34
			100.000	-	150.000		89	40.22
			150.000	-	200.000		76	47.80
			200.000	-	250.000		69	54.69
			250.000	-	300.000		77	62.38
			300.000	-	350.000		70	69.36
			350.000	-	400.000		62	75.55
			400.000	-	450.000		58	81.34
			450.000	-	500.000		46	85.93
			500.000	-	550.000		39	89.82
			550.000	-	600.000		38	93.61
			600.000	-	650.000		19	95.51
			650.000	-	700.000		14	96.91
			700.000	-	750.000		7	97.60

```
                              750.000  -      800.000             8    98.40
                              800.000  -      850.000             5    98.90
                              850.000  -      900.000             3    99.20
                              900.000  -      950.000             1    99.30
                              950.000  -     1000.000             4    99.70
                             1000.000  -     1050.000             1    99.80
                             1050.000  -     1100.000             0    99.80
                             1100.000  -     1150.000             1    99.90
                             1150.000  -     1200.000             1   100.00
 TT2              82.303  135.004                          0
                                       -        0.000           386    38.56
                                 0.000  -       50.000           213    59.84
                                50.000  -      100.000           149    74.73
                               100.000  -      150.000            88    83.52
                               150.000  -      200.000            47    88.21
                               200.000  -      250.000            26    90.81
                               250.000  -      300.000             8    91.61
                               300.000  -      350.000             9    92.51
                               350.000  -      400.000            19    94.41
                               400.000  -      450.000            12    95.60
                               450.000  -      500.000            14    97.00
                               500.000  -      550.000            12    98.20
                               550.000  -      600.000             9    99.10
                               600.000  -      650.000             4    99.50
                               650.000  -      700.000             2    99.70
                               700.000  -      750.000             2    99.90
                               750.000  -      800.000             1   100.00
 TT3              16.200   29.584                           0
                                       -        0.000           621    62.10
                                 0.000  -       50.000           245    86.60
                                50.000  -      100.000           105    97.10
                               100.000  -      150.000            27    99.80
                               150.000  -      200.000             2   100.00


FEC XN    PRI         BDT      ASSEM  CURRENT  NEXT  PARAMETER    VALUE
  1001     0       97679.139   1001      14     15
                                                      T2        97513.442
                                                      T1        97464.998
  1003     0       97723.509   1003       0      1
```

Для присваивания значений P – параметрам заявки, можно использовать блок PLUS, который имеет вид:

PLUS A,B

Здесь A- выражение, значение которого записывается в P параметр, заданный как B. Фактически, блок PLUS ничем не лучше, чем блок ASSIGN. Однако в этом блоке второй параметр может быть опущен, и тогда блок фактически просто выполняет вызов выражения A для выполнения определенной работы, в стиле вызова функций на языке C или C++ .

Для перемещения текущей заявки в конец списка текущих событий для данного приоритетного класса, используют блок

BUFFER.

Иначе говоря, после него текущая заявка пропускает вперед все заявки с данным приоритетом. Пример применения блока можно найти в разделе, где описываются блоки проверки условий. Фактически, блок BUFFER эквивалентен блоку PRIORITY PR.

21. Работа с прерываниями.

Прерывания возможны только для одноканальных устройств. Если устройство в данный момент свободно, то занятие его по прерыванию не лучше, чем обычное занятие устройства. Если устройство обслуживает заявку, то прерывание возможно, и тогда будет обслуживаться заявка, пришедшая по прерыванию.

Когда обслуживание прерывания завершено, то устройство может продолжить обслуживание прерванной заявки.

Занятие устройства по прерыванию выполняет блок

PREEMPT A,B,C,D,E

Завершение обслуживания по прерыванию выполняет блок

RETURN A

Здесь A - номер устройства, которое занято по прерыванию.

B - или пусто, или PR

Если B равно PR, то занятие разрешено, только если по прерыванию пришла заявка с более высоким приоритетом. В подобных случаях может возникнуть цепочка прерываний из заявок с возрастающими приоритетами.

Если B отсутствует, то прерывается обработка простых заявок, но не прерывается обслуживание заявок, пришедших по прерыванию.

C - это метка блока, куда пошлют прерванную заявку для обслуживания. Тогда она, возможно, не попадает в список прерываний. Если ее там не примут, то ее обслуживание прерывается, и она по-прежнему претендует на обслуживание в устройстве.

Если C опущено, то заявка всегда претендует на обслуживание в устройстве, то есть попадает в список прерываний.

D - номер P - параметра прерванной заявки, в который заносится время, оставшееся до окончания ее обслуживания в устройстве (предполагается, что заявка находится в некотором блоке ADVANCE).

E - если E=RE, то заявка непременно теряет право обслуживания в устройстве, и будет обслуживаться там, куда ее послали. Только в этом случае, заявка, перемещенная в другой блок не должна освобождать устройство, в котором она обслуживалась. Во всех остальных случаях перемещения, заявка должна освободить устройство.

Если E опущено, то заявка сохраняет право на обслуживание в устройстве.

Примечание: устройство остается занятым прерванной заявкой во всех случаях, кроме случая, когда E=RE.

Пример:

Пусть заявки поступают в систему со средним интервалом 100 единиц, распределенным по экспоненциальному закону. Приоритет заявок равномерно распределен в интервале 1.. 10. Время обработки любой заявки составляет 95+-15 единиц. Обслуживание заявок идет с прерываниями и учетом приоритета. Промоделировать обслуживание 1000 заявок. Протабулировать время обслуживания каждой заявки. Модель такой системы может выглядеть следующим образом:

```
tt table m1,0,80,100
```

```
generate (exponential(1,0,100))
priority (duniform(1,1,10))
queue ust1
preempt ust,pr
mark
depart ust1
advance 95,15
tabulate tt
return ust
terminate 1
start 1000
```

Итоговая статистика такой модели имеет следующий вид, который подтверждает наличие многоуровневых прерываний

```
            GPSS World Simulation Report - Untitled Model 1.2.2
                 Monday, September 08, 2003 11:14:38

          START TIME           END TIME  BLOCKS  FACILITIES  STORAGES
               0.000         101296.423    10        1          0

              NAME                     VALUE
          TT                         10000.000
          UST                        10002.000
          UST1                       10001.000

 LABEL              LOC  BLOCK TYPE     ENTRY COUNT CURRENT COUNT RETRY
                     1    GENERATE          1004             0       0
                     2    PRIORITY          1004             0       0
                     3    QUEUE             1004             2       0
                     4    PREEMPT           1002             0       0
                     5    MARK              1002             0       0
                     6    DEPART            1002             0       0
                     7    ADVANCE           1002             2       0
                     8    TABULATE          1000             0       0
                     9    RETURN            1000             0       0
                    10    TERMINATE         1000             0       0


FACILITY          ENTRIES  UTIL.   AVE. TIME AVAIL. OWNER PEND INTER RETRY DELAY
 UST                1002    0.941      95.175  1     1002    0    1     0      2


QUEUE               MAX CONT. ENTRY ENTRY(0) AVE.CONT. AVE.TIME   AVE.(-0) RETRY
 UST1                19    2   1004    497     5.475    552.434   1093.972   0


TABLE               MEAN    STD.DEV.       RANGE           RETRY FREQUENCY CUM.%
 TT               151.283  108.006                          0
                                    80.000  -      160.000          669    66.90
                                   160.000  -      240.000          189    85.80
                                   240.000  -      320.000           81    93.90
                                   320.000  -      400.000           29    96.80
                                   400.000  -      480.000           12    98.00
                                   480.000  -      560.000            7    98.70
                                   560.000  -      640.000            5    99.20
                                   640.000  -      720.000            3    99.50
                                   720.000  -      800.000            2    99.70
                                   800.000  -      880.000            0    99.70
                                   880.000  -      960.000            1    99.80
                                   960.000  -     1040.000            2   100.00


FEC XN   PRI         BDT      ASSEM  CURRENT  NEXT  PARAMETER    VALUE
  1002    3      101319.600   1002      7      8
```

```
1005    0      101389.871   1005      0      1
```

Другой пример:

Пусть основные заявки поступают в систему со средним интервалом 200 единиц, распределенным по экспоненциальному закону. Пусть также с интервалом 300+-200 в систему поступают заявки, которые занимают устройство по прерыванию. Основные заявки этом случае отправляются на обслуживание в другое устройство, которое работает втрое медленнее основного. Основное устройство обслуживает основные заявки за интервал времени равный 120+-20, а приоритетные заявки – за время 150+-100 единиц. Промоделировать обслуживание 1000 заявок, пришедших по прерыванию. Протабулировать времена обслуживания основных заявок, и заявок, пришедших по прерыванию. Модель такой системы может выглядеть следующим образом:

```
tt table m1,0,80,100
tt1 table m1,0,10,100
generate (exponential(1,0,200))
queue ust
seize ust
depart ust
mark
advance 120,20
release ust
tabulate tt
terminate

generate 300,200
queue ust1
preempt ust,,next,2
mark
depart ust1
advance 150,100
tabulate tt1
return ust
terminate 1
next release ust
seize ustd
advance (3#p2)
release ustd
tabulate tt
terminate
start 1000
```

Итоговая статистика этой модели имеет вид:

```
    GPSS World Simulation Report - Untitled Model 1.8.1

         Monday, September 08, 2003 11:26:21

 START TIME           END TIME  BLOCKS  FACILITIES  STORAGES
     0.000         345380.811    24        2          0
```

NAME	VALUE
NEXT	19.000
TT	10000.000
TT1	10001.000
UST	10002.000
UST1	10003.000
USTD	10004.000

LABEL	LOC	BLOCK TYPE	ENTRY COUNT	CURRENT COUNT	RETRY
	1	GENERATE	1763	0	0
	2	QUEUE	1763	1	0
	3	SEIZE	1762	1	0
	4	DEPART	1761	0	0
	5	MARK	1761	0	0
	6	ADVANCE	1761	0	0
	7	RELEASE	904	0	0
	8	TABULATE	904	0	0
	9	TERMINATE	904	0	0
	10	GENERATE	1161	0	0
	11	QUEUE	1161	0	0
	12	PREEMPT	1161	0	0
	13	MARK	1161	0	0
	14	DEPART	1161	0	0
	15	ADVANCE	1161	0	0
	16	TABULATE	1000	0	0
	17	RETURN	1000	0	0
	18	TERMINATE	1000	0	0
NEXT	19	RELEASE	1018	0	0
	20	SEIZE	1018	0	0
	21	ADVANCE	1018	0	0
	22	RELEASE	1018	0	0
	23	TABULATE	1018	0	0
	24	TERMINATE	1018	0	0

FACILITY	ENTRIES	UTIL.	AVE. TIME	AVAIL.	OWNER	PEND	INTER	RETRY	DELAY
UST	2923	0.930	109.867	1	2922	0	0	0	1
USTD	1018	0.556	188.754	1	0	0	0	0	0

QUEUE	MAX	CONT.	ENTRY	ENTRY(0)	AVE.CONT.	AVE.TIME	AVE.(-0)	RETRY
UST	21	2	1763	116	3.859	756.068	809.319	0
UST1	1	0	1161	1161	0.000	0.000	0.000	0

TABLE	MEAN	STD.DEV.	RANGE			RETRY	FREQUENCY	CUM.%
TT	217.315	131.273				0		
			80.000	-	160.000		1009	52.50
			160.000	-	240.000		241	65.04
			240.000	-	320.000		282	79.71
			320.000	-	400.000		207	90.48
			400.000	-	480.000		90	95.16
			480.000	-	560.000		47	97.61
			560.000	-	640.000		20	98.65
			640.000	-	720.000		16	99.48
			720.000	-	800.000		7	99.84
			800.000	-	880.000		2	99.95
			880.000	-	960.000		1	100.00
TT1	141.505	55.936				0		
			50.000	-	60.000		71	7.10
			60.000	-	70.000		52	12.30
			70.000	-	80.000		47	17.00
			80.000	-	90.000		56	22.60
			90.000	-	100.000		57	28.30
			100.000	-	110.000		50	33.30
			110.000	-	120.000		52	38.50
			120.000	-	130.000		70	45.50
			130.000	-	140.000		56	51.10
			140.000	-	150.000		50	56.10
			150.000	-	160.000		70	63.10

```
                                160.000  -      170.000            47    67.80
                                170.000  -      180.000            44    72.20
                                180.000  -      190.000            49    77.10
                                190.000  -      200.000            46    81.70
                                200.000  -      210.000            32    84.90
                                210.000  -      220.000            43    89.20
                                220.000  -      230.000            30    92.20
                                230.000  -      240.000            41    96.30
                                240.000  -      250.000            37   100.00


CEC XN    PRI          M1       ASSEM  CURRENT  NEXT  PARAMETER    VALUE
  2922     0      345242.715    2922       3       4


FEC XN    PRI          BDT      ASSEM  CURRENT  NEXT  PARAMETER    VALUE
  2924     0      345450.826    2924       0      10
  2926     0      345540.857    2926       0       1
```

По итоговой статистике, можно сделать вывод, что заявки, пришедшие по прерыванию, обслуживаются заметно быстрее.

22. Перенаправление потоков заявок.

Безусловное перенаправление заявок выполняет блок

TRANSFER ,метка.

При выполнении этого блока заявка отправляется на блок с указанной меткой или номером .

Блок TRANSFER в режиме BOTH.имеет вид:

TRANSFER BOTH,метка1,метка2

Такой TRANSFER пытается отправить заявку на блок с первой меткой, если ее там не принимают, то на блок со второй меткой. Если ее и там не принимают, то заявка задерживается, пытаясь войти в блок TRANSFER позднее.

Примечание : во всех случаях, когда говорится о метке блока, с равным успехом это может быть и выражение, определяющее номер блока.

Пример. Пусть заявки поступают в систему с интервалом 100+-50 и должны обслуживаться одним из устройств, 1 или 2. В устройстве 1 обслуживание занимает 150+-50 единиц времени, а в устройстве 2 – 170+-40. В первую очередь делается попытка занять устройство 1. После обслуживания в 1 или 2 устройстве, заявка обслуживается в устройстве 3 за время 90+-30 и покидает систему. Промоделировать обслуживание 1000 заявок.

Текст такой модели может выглядеть следующим образом:

```
GENERATE 100,50
QUEUE OCH1
TRANSFER BOTH,Y1,Y2
Y1 SEIZE 1
DEPART OCH1
ADVANCE 150,50
RELEASE 1
TRANSFER ,Y3
Y2 SEIZE 2
DEPART OCH1
```

```
ADVANCE 170,40
 RELEASE 2
Y3  QUEUE 3
SEIZE 3
DEPART 3
ADVANCE 90,30
RELEASE 3
TERMINATE 1
START 1000
```

Результат моделирования имеет следующий вид:

```
    GPSS World Simulation Report - Untitled Model 1.5.1

         Wednesday, September 10, 2003 06:59:30

 START TIME           END TIME  BLOCKS  FACILITIES  STORAGES
      0.000         100785.025    18        3          0


    NAME                     VALUE
OCH1                       10000.000
Y1                             4.000
Y2                             9.000
Y3                            13.000


 LABEL              LOC  BLOCK TYPE     ENTRY COUNT CURRENT COUNT RETRY
                    1    GENERATE          1002             0       0
                    2    QUEUE             1002             0       0
                    3    TRANSFER          1002             0       0
Y1                  4    SEIZE              538             0       0
                    5    DEPART             538             0       0
                    6    ADVANCE            538             1       0
                    7    RELEASE            537             0       0
                    8    TRANSFER           537             0       0
Y2                  9    SEIZE              464             0       0
                   10    DEPART             464             0       0
                   11    ADVANCE            464             1       0
                   12    RELEASE            463             0       0
Y3                 13    QUEUE             1000             0       0
                   14    SEIZE             1000             0       0
                   15    DEPART            1000             0       0
                   16    ADVANCE           1000             0       0
                   17    RELEASE           1000             0       0
                   18    TERMINATE         1000             0       0


FACILITY          ENTRIES  UTIL.   AVE. TIME AVAIL. OWNER PEND INTER RETRY DELAY
 1                   538    0.788     147.591  1     1002    0     0     0       0
 2                   464    0.785     170.556  1     1001    0     0     0       0
 3                  1000    0.888      89.528  1        0    0     0     0       0


QUEUE              MAX CONT. ENTRY ENTRY(0) AVE.CONT. AVE.TIME   AVE.(-0) RETRY
 3                   5    0   1000     306     0.424     42.780     61.643   0
 OCH1                2    0   1002     739     0.096      9.644     36.743   0


FEC XN   PRI         BDT      ASSEM  CURRENT  NEXT  PARAMETER    VALUE
  1001    0      100798.069   1001     11     12
  1003    0      100814.928   1003      0      1
  1002    0      100839.653   1002      6      7
```

По итоговой статистике видно, что первое устройство занято больше второго, так как заявки пытаются вначале войти именно в него.

Блок TRANSFER в статистическом режиме имеет вид:

TRANSFER вероятность,метка1,метка 2

Этот блок с заданной вероятностью отправляет заявку по второй метке, а с дополняющей вероятностью – по первой метке.

Пример:

TRANSFER .250,MM1,MM2

Или

TRANSFER p1,MM1,MM2

Вероятность должна быть вещественным числом.

В случае использования СЧА или выражения, в качестве вероятности используется 1/1000 вычисленного значения.

Пример:

Пусть заявки поступают в систему с интервалом 100+-70 и обслуживаются одним из двух устройств. В первом устройстве – за 150+-90, а во втором – за 200+-80. вероятность попасть в первое устройство - 0.4 – а во второе – 0.6 . тогда модель может выглядеть следующим образом.

```
GENERATE 100,70
TRANSFER .600,MM1,MM2
MM1   QUEUE 1
SEIZE 1
DEPART 1
ADVANCE 150,90
RELEASE 1
TERMINATE 1
MM2 QUEUE 2
SEIZE 2
DEPART 2
ADVANCE 200,80
RELEASE 2
TERMINATE 1
START 1000
```

Выходная статистика этой модели имеет вид:

```
    GPSS World Simulation Report - Untitled Model 1.2.1

         Friday, September 12, 2003 16:38:33

 START TIME           END TIME  BLOCKS  FACILITIES  STORAGES
      0.000         110146.374    14        2          0

    NAME                        VALUE
MM1                              3.000
MM2                              9.000

 LABEL             LOC  BLOCK TYPE     ENTRY COUNT CURRENT COUNT RETRY
                   1    GENERATE          1097             0       0
                   2    TRANSFER          1097             0       0
MM1                3    QUEUE              443             0       0
                   4    SEIZE              443             0       0
```

```
                    5    DEPART             443             0       0
                    6    ADVANCE            443             0       0
                    7    RELEASE            443             0       0
                    8    TERMINATE          443             0       0
MM2                 9    QUEUE              654            96       0
                   10    SEIZE              558             1       0
                   11    DEPART             557             0       0
                   12    ADVANCE            557             0       0
                   13    RELEASE            557             0       0
                   14    TERMINATE          557             0       0


FACILITY          ENTRIES  UTIL.   AVE. TIME AVAIL. OWNER PEND INTER RETRY DELAY
 1                  443    0.598     148.638  1        0    0    0     0       0
 2                  558    0.999     197.114  1      942    0    0     0      96

QUEUE               MAX CONT. ENTRY ENTRY(0) AVE.CONT. AVE.TIME   AVE.(-0) RETRY
 1                    4    0    443    224     0.243     60.478    122.337   0
 2                   99   97    654      2    46.129   7769.095   7792.927   0

CEC XN   PRI          M1      ASSEM  CURRENT  NEXT  PARAMETER    VALUE
   942    0       95373.782    942     10     11

FEC XN   PRI          BDT     ASSEM  CURRENT  NEXT  PARAMETER    VALUE
  1098    0      110234.724   1098      0      1
```

Здесь можно видеть, что второе устройство действительно более занято, чем первое.

Если модель модифицировать следующим образом,

```
GENERATE 100,70
assign go,600
TRANSFER  p$go,MM1,MM2
MM1   QUEUE 1
SEIZE 1
DEPART 1
ADVANCE 150,90
RELEASE 1
TERMINATE 1
MM2 QUEUE 2
SEIZE 2
DEPART 2
ADVANCE 200,80
RELEASE 2
TERMINATE 1
START 1000
```

то результат моделирования будет практически тем же.

Другой пример использования Transfer в вероятностном режиме.

Пусть заявки поступают в одноканальное устройство с интервалом 100 единиц распределенным по экспоненциальному закону. Заявки обслуживаются в нем с интервалом 80+-30. Пусть обслуживание 15% заявок оказывается неудачным, и заявки нуждаются в повторном обслуживании. Промоделировать обслуживание 5000 заявок. Тогда, модель системы может выглядеть следующим образом.

```
TT1 TABLE M1,0,150,50
GENERATE (exponential(1,0,100))
```

```
MM2   QUEUE 1
      SEIZE 1
       DEPART 1
       ADVANCE 80,30
       RELEASE 1
       TRANSFER .15,MM1,MM2
MM1    TABULATE TT1
       TERMINATE 1
   START 5000
```

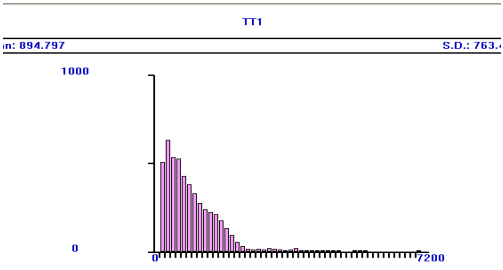
Итоговая статистика имеет вид:

```
    GPSS World Simulation Report - Untitled Model 1.6.1
         Sunday, September 14, 2003 08:15:06

 START TIME          END TIME  BLOCKS  FACILITIES  STORAGES
      0.000        498578.562     9        1          0

   NAME                      VALUE
MM1                           8.000
MM2                           2.000
TT1                       10000.000

 LABEL            LOC  BLOCK TYPE     ENTRY COUNT CURRENT COUNT RETRY
                   1   GENERATE          5012             0       0
MM2                2   QUEUE             5898            11       0
                   3   SEIZE             5887             1       0
                   4   DEPART            5886             0       0
                   5   ADVANCE           5886             0       0
                   6   RELEASE           5886             0       0
                   7   TRANSFER          5886             0       0
MM1                8   TABULATE          5000             0       0
                   9   TERMINATE         5000             0       0

FACILITY         ENTRIES  UTIL.   AVE. TIME AVAIL. OWNER PEND INTER RETRY DELAY
 1                 5887    0.945      80.004  1     5006    0    0     0     11


QUEUE             MAX CONT. ENTRY ENTRY(0) AVE.CONT. AVE.TIME   AVE.(-0) RETRY
 1                 31   12   5898    320     8.056    680.994    720.061   0

TABLE             MEAN    STD.DEV.       RANGE            RETRY FREQUENCY CUM.%
 TT1            894.797  763.410                           0
                                  0.000  -      150.000          509    10.18
                                150.000  -      300.000          635    22.88
                                300.000  -      450.000          535    33.58
                                450.000  -      600.000          526    44.10
                                600.000  -      750.000          431    52.72
                                750.000  -      900.000          384    60.40
                                900.000  -     1050.000          330    67.00
                               1050.000  -     1200.000          278    72.56
                               1200.000  -     1350.000          240    77.36
                               1350.000  -     1500.000          224    81.84
                               1500.000  -     1650.000          215    86.14
                               1650.000  -     1800.000          180    89.74
                               1800.000  -     1950.000          137    92.48
                               1950.000  -     2100.000           98    94.44
                               2100.000  -     2250.000           57    95.58
                               2250.000  -     2400.000           35    96.28
                               2400.000  -     2550.000           17    96.62
                               2550.000  -     2700.000           12    96.86
                               2700.000  -     2850.000           18    97.22
                               2850.000  -     3000.000           14    97.50
                               3000.000  -     3150.000           20    97.90
                               3150.000  -     3300.000           16    98.22
                               3300.000  -     3450.000           15    98.52
                               3450.000  -     3600.000            8    98.68
                               3600.000  -     3750.000           12    98.92
```

3750.000	-	3900.000	21	99.34
3900.000	-	4050.000	9	99.52
4050.000	-	4200.000	4	99.60
4200.000	-	4350.000	5	99.70
4350.000	-	4500.000	2	99.74
4500.000	-	4650.000	1	99.76
4650.000	-	4800.000	1	99.78
4800.000	-	4950.000	1	99.80
4950.000	-	5100.000	2	99.84
5100.000	-	5250.000	0	99.84
5250.000	-	5400.000	0	99.84
5400.000	-	5550.000	2	99.88
5550.000	-	5700.000	2	99.92
5700.000	-	5850.000	3	99.98
5850.000	-	6000.000	0	99.98
6000.000	-	6150.000	0	99.98
6150.000	-	6300.000	0	99.98
6300.000	-	6450.000	0	99.98
6450.000	-	6600.000	0	99.98
6600.000	-	6750.000	0	99.98
6750.000	-	6900.000	0	99.98
6900.000	-	7050.000	0	99.98
7050.000	-	7200.000	0	99.98
7200.000	-	_	1	100.00

CEC XN	PRI	M1	ASSEM	CURRENT	NEXT	PARAMETER	VALUE
5006	0	497437.935	5006	3	4		

FEC XN	PRI	BDT	ASSEM	CURRENT	NEXT	PARAMETER	VALUE
5013	0	498590.684	5013	0	1		



По результатам моделирования можно заметить, что некоторые заявки действительно многократно обслуживаются устройством.

Имеются также следующие версии блока Transfer.

Блок TRANSFER в режиме All имеет следующий вид.

TRANSFER ALL,первая метка, последняя метка, шаг

В этом режиме заявка пытается войти в блок с первой меткой, если ей это удается, то она туда и уходит. Если же нет, то увеличивается номер блока на величину шага, и делается попытка войти в блок с этим номером.

Дальше все происходит аналогично входу по первой метке. Последняя попытка войти делается по последней метке, или в блок, лежащий до нее, если величина шага не обеспечивает попадания на последнюю метку. В случае неудачи, заявка задерживается на входе в блок TRANSFER до появления возможности войти в один из блоков.

Примечание: номер блока можно посмотреть в выходной статистике.

Блок TRANSFER в режиме Pick имеет следующий вид.

TRANSFER PICK,ПерваяМетка,ПоследняяМетка

В этом режиме равновероятно выбирается блок, номер которого лежит между Первой и Последней меткой включительно. Именно туда и отправляется заявка. Польза от блока в этом режиме сомнительна. Фактически, он оставлен только для совместимости со старыми версиями GPSS.

Блок TRANSFER в режиме Function имеет следующий вид.

TRANSFER FN,ИмяФункции,Приращение

Здесь вычисляется значение функции, к нему добавляется приращение. Полученная величина определяет номер блока, в который отправляется заявка . Польза от блока в этом режиме сомнительна. Фактически он может быть заменен блоком безусловного перенаправления заявок, где номер блока задается выражением.

Блок TRANSFER в режиме Parameter имеет следующий вид.

TRANSFER P,ИмяPпараметра,Приращение

Здесь выбирается значение соответствующего P параметра и складывается с приращением. Полученная величина определяет номер блока, в который отправляется заявка. Польза от блока в этом режиме сомнительна. Фактически он также может быть заменен блоком безусловного перенаправления заявок, где номер блока задается выражением. Иногда его используют для возврата из фрагмента модели, используемого как подпрограмма.

Блок TRANSFER в режиме Subroutine имеет следующий вид.

TRANSFER SBR, метка перехода,имяPпараметра

В этом режиме заявка переходит по метке, а в P- параметр – записывается номер самого этого блока . Для возврата из процедуры, используется блок TRANSFER в режиме Parameter с приращением в 1, что обеспечивает возврат на следующий блок после блока перехода в стиле подпрограммы.

TRANSFER в режиме Simultaneous на сегодня вообще лишен смысла, и не используется.

Примеры по последним версиям блока не приводятся, ввиду их тривиальности.

23. Блоки проверки условий.

Для явной проверки условий в GPSS используются блоки GATE и TEST.

Блок проверки GATE имеет следующий формат:

GATE условие A,B

Блок GATE управляет потоком заявок на основе проверки значения условия. Блок GATE, как и блок TEST, определяет номер следующего блока, к

которому должна перейти заявка. Если условие, верно, то заявка проходит через блок GATE и идет дальше по модели, иначе заявка уходит по метке B. Блок GATE может задержать заявку на входе, если в нем не задан параметр B. Тогда, заявка будет многократно пытаться войти в блок GATE.

В поле условия может задаваться один из следующих логических операторов.

1) для устройств:

NU - устройство , заданное в поле A, свободно;

U - устройство , заданное в поле A, занято (в результате выполнения заявкой блока SEIZE или PREEMPT);

NI – устройство , заданное в поле A, не прервано;

I - устройство , заданное в поле A, обслуживает прерывание;

FV - устройство , заданное в поле A, доступно;

FNV - устройство , заданное в поле A, не доступно;

2) для многоканальных устройств:

SE - многоканальное устройство , заданное в поле A, пусто;

SNE - многоканальное устройство , заданное в поле A, не пусто;

SF - многоканальное устройство , заданное в поле A, заполнено;

SNF - многоканальное устройство , заданное в поле A, не заполнено;

SV - многоканальное устройство , заданное в поле A, доступно;

SNV - многоканальное устройство , заданное в поле A, не доступно;

3) для логических ключей:

LS - логический ключ , заданный в поле A, включен (1);

LR - логический ключ , заданный в поле A, выключен(0);

4) логические операторы, связанные с заявками:

M - в блоке, номер которого задан в поле A блока GATE, находится в состоянии синхронизации заявка, принадлежащая к тому же семейству, что и заявка, пытающаяся войти в блок GATE;

NM - в блоке , номер которого задан в поле A блока GATE, в состоянии синхронизации нет ни одной заявки, принадлежащей к тому же семейству, что и заявка, пытающаяся войти в блок GATE.

Для проверок M или NM имеет смысл указывать только номера или метки блоков ASSEMBLE, GATHER, или MATCH.

Поле A содержит имя или номер объекта, для которого проводится проверка. Всего возможны проверки для 4 типов объектов.

Поле B содержит номер следующего блока для входящей заявки, когда логический оператор имеет значение "ложь". Если поле B определено, то оно должно содержать номер блока, допустимый для текущей модели.

Блоки GATE очень мощные, но они могут повлечь значительные расходы процессорного времени на безуспешные попытки заявки войти в блок. Чтобы уменьшить частоту безуспешных попыток вхождения в блок, можно поместить заявки в список пользователя, используя блоки LINK и UNLINK.

Пример:

Пусть имеется железнодорожный переезд со шлагбаумом. Автомобили с каждой стороны шлагбаума подъезжают со средним интервалом в 20 секунд, распределенным по экспоненциальному закону. Через шлагбаум одновременно могут переезжать два автомобиля. Время переезда – 15+-3 секунды. В среднем через 3600+- 900 секунд проходит поезд, и шлагбаум закрывается на 300+-60 секунд. Промоделировать работу переезда в течение суток.

Текст модели может выглядеть следующим образом.

```
INITIAL LS$SHLAG,1
PERE   STORAGE 2
TT1 TABLE M1,0,10,50
TT2 TABLE Q1,0,3,50

 GENERATE (exponential(1,0,20))
 QUEUE 1
 GATE LS SHLAG
 ENTER PERE
 DEPART 1
 ADVANCE 15,3
 LEAVE PERE
 TABULATE TT1
 TERMINATE

 GENERATE (exponential(1,0,20))
 QUEUE 1
 GATE LS SHLAG
 ENTER PERE
 DEPART 1
 ADVANCE 15,3
 LEAVE PERE
 TABULATE TT1
 TERMINATE

GENERATE 3600,900
LOGIC R SHLAG
ADVANCE 300,60
LOGIC S  SHLAG
TABULATE TT2
TERMINATE

GENERATE (3600#24)
TERMINATE 1
    START 1
```

В принципе, другие варианты использования блоков GATE достаточно понятны, кроме вопросов синхронизации, которые будут рассмотрены позже.

Блок TEST заметно мощнее блока GATE, и полностью покрывает его возможности. Он имеет следующий формат:

TEST отношение A,B,C

Здесь отношение проверяется между значениями из полей A и B. Блок TEST определяет номер следующего блока для вошедшей в него заявки в зависимости от того, выполняется требуемое отношение или нет.

Отношением может быть

L - меньше.

LE - меньше или равно.

E - равно.

NE - не равно.

G - больше.

GE - больше или равно.

Если отношение истинно, заявка переходит к следующему блоку. Если отношение ложно, заявка переходит к блоку, номер которого задан меткой (возможно номером) блока.

Блок TEST может работать в двух режимах:

1) в режиме безусловного входа. Тогда метка должна быть задана.

в режиме условного входа. Тогда метка отсутствует, и заявка не может войти в блок TEST до тех пор, пока значения не изменятся таким образом, что отношение будет истинно.

Использование таких блоков TEST может значительно увеличить время работы модели. Рассмотрим несколько примеров блоков TEST.

TEST L C1,500,SNA

Пока значение условного относительного времени не достигнет 500, заявка от блока TEST будет переходить к следующему по номеру блоку. Как только значение условного времени станет равным 500 (и более), заявка будет переходить к блоку, номер которого определяется меткой SNA.

TEST GE N$PATH1,N$PATH2

Если счетчик числа входов в блок PATH1 (N$PATH1) больше или равен счетчику числа входов в блок PATH2 (N$PATH2), заявки входят в блок TEST и переходят к следующему по номеру блоку. Если счетчик блока PATH1 меньше счетчика блока PATH2, заявки не могут войти в блок TEST.

Пример: пусть имеется 2 устройства обслуживающих заявки. Один обслуживает за 11+-5 с,другой-за11+-7с. Заявки становятся в очередь к тому устройству, где очередь короче.

Тогда модель может выглядеть следующим образом.

```
GENERATE 5.6,5
TEST L Q1,Q2,LAB
QUEUE 1
SEIZE 1
DEPART 1
ADVANCE 11,5
RELEASE 1
TERMINATE 1
LAB QUEUE 2
```

```
SEIZE 2
DEPART 2
ADVANCE 11,7
RELEASE 2
TERMINATE 1
 START 200
```

Пример: система должна обслуживать заявки, если длина очереди меньше 10. Если она равна 10,то последующие заявки теряются.

```
 Решение:
GENERATE 6,5
TEST L Q1,10,MM
 QUEUE 1
 SEIZE 1
 DEPART 1
 ADVANCE 10,5
 RELEASE 1
 TERMINATE 1
 MM      TERMINATE
 START 200
```

Пример:

Пусть в систему поступают заявки с интервалом 100, распределенным по экспоненциальному закону. Пусть обслуживание заявок ведется набором устройств, каждое из которых имеет время обслуживания, равное в среднем 500 + 40#номер устройства. Отклонение времени обслуживания от среднего составляет 30%. Для обслуживания выбирается первое попавшееся свободное устройство. Определить общее и среднее время обслуживания 1000 заявок.

Текст модели может выглядеть следующим образом.

```
GENERATE (exponential(1,0,100))
assign num,0
rret assign num+,1
test e f*num,0,rret
 SEIZE p$num
 ADVANCE (500+40#p$num),((500+40#p$num)#0.3)
 RELEASE p$num
ll savevalue sum+,m1
savevalue sr_sum,(X$sum/n$ll)
 TERMINATE 1
 START 1000
```

В данном случае, выходная статистика представляет интерес. Она, в частности, свидетельствует, что разные устройства имеют разное время задержки, и что в данном случае понадобилось иметь не менее 16 устройств.

```
GPSS World Simulation Report - Untitled Model 1.10.1


Monday, September 15, 2003 16:05:07

START TIME           END TIME  BLOCKS  FACILITIES  STORAGES
     0.000         103261.300    10       16          0
```

NAME	VALUE
LL	8.000
NUM	10000.000
RRET	3.000
SR_SUM	10002.000
SUM	10001.000

LABEL	LOC	BLOCK TYPE	ENTRY COUNT	CURRENT COUNT	RETRY
	1	GENERATE	1006	0	0
	2	ASSIGN	1006	0	0
RRET	3	ASSIGN	4800	0	0
	4	TEST	4800	0	0
	5	SEIZE	1006	0	0
	6	ADVANCE	1006	6	0
	7	RELEASE	1000	0	0
LL	8	SAVEVALUE	1000	0	0
	9	SAVEVALUE	1000	0	0
	10	TERMINATE	1000	0	0

FACILITY	ENTRIES	UTIL.	AVE. TIME	AVAIL.	OWNER	PEND	INTER	RETRY	DELAY
1	159	0.831	540.000	1	0	0	0	0	0
2	146	0.816	577.194	1	1004	0	0	0	0
3	126	0.754	617.577	1	1003	0	0	0	0
4	114	0.722	654.277	1	1006	0	0	0	0
5	97	0.657	699.208	1	1000	0	0	0	0
6	87	0.622	738.408	1	1001	0	0	0	0
7	74	0.552	770.942	1	1005	0	0	0	0
8	62	0.492	820.000	1	0	0	0	0	0
9	45	0.375	860.000	1	0	0	0	0	0
10	35	0.305	900.000	1	0	0	0	0	0
11	24	0.218	940.000	1	0	0	0	0	0
12	21	0.199	980.000	1	0	0	0	0	0
13	7	0.069	1020.000	1	0	0	0	0	0
14	4	0.041	1060.000	1	0	0	0	0	0
15	4	0.043	1100.000	1	0	0	0	0	0
16	1	0.011	1140.000	1	0	0	0	0	0

SAVEVALUE	RETRY	VALUE
SUM	0	690920.000
SR_SUM	0	690.920

FEC XN	PRI	BDT	ASSEM	CURRENT	NEXT	PARAMETER	VALUE
1000	0	103338.116	1000	6	7		
						NUM	5.000
1007	0	103379.619	1007	0	1		
1001	0	103399.834	1001	6	7		
						NUM	6.000
1003	0	103566.636	1003	6	7		
						NUM	3.000
1004	0	103670.975	1004	6	7		
						NUM	2.000
1006	0	103913.751	1006	6	7		
						NUM	4.000
1005	0	103931.604	1005	6	7		
						NUM	7.000

Построим модель аналогичной системы, у которой допускаются очереди до 4-х заявок. Такая модель может иметь вид:

GENERATE (exponential(1,0,100))
assign num,0
rret assign num+,1
test l q*num,4,rret

```
queue  p$num
 SEIZE p$num
depart p$num
 ADVANCE (500+40#p$num),((500+40#p$num)#0.3)
 RELEASE p$num
ll savevalue sum+,m1
savevalue sr_sum,(X$sum/n$ll)
 TERMINATE 1
 START 1000
```

В итоговой статистике, число устройств, необходимых для этого случая уменьшается до 8.

```
            GPSS World Simulation Report - Untitled Model 1.17.1

                 Monday, September 15, 2003 16:23:13

          START TIME           END TIME  BLOCKS  FACILITIES  STORAGES
               0.000         105287.189    12        8          0

             NAME                      VALUE
          LL                          10.000
          NUM                      10000.000
          RRET                         3.000
          SR_SUM                   10002.000
          SUM                      10001.000

 LABEL              LOC  BLOCK TYPE     ENTRY COUNT CURRENT COUNT RETRY
                     1    GENERATE          1023             0       0
                     2    ASSIGN            1023             0       0
RRET                 3    ASSIGN            3506             0       0
                     4    TEST              3506             0       0
                     5    QUEUE             1023            16       0
                     6    SEIZE             1007             1       0
                     7    DEPART            1006             0       0
                     8    ADVANCE           1006             6       0
                     9    RELEASE           1000             0       0
LL                  10    SAVEVALUE         1000             0       0
                    11    SAVEVALUE         1000             0       0
                    12    TERMINATE         1000             0       0


FACILITY          ENTRIES  UTIL.   AVE. TIME AVAIL. OWNER PEND INTER RETRY DELAY
 1                  192    1.000     548.281  1     1004    0     0     0      4
 2                  181    0.991     576.238  1     1006    0     0     0      3
 3                  170    0.988     611.827  1     1007    0     0     0      4
 4                  154    0.981     670.924  1     1001    0     0     0      2
 5                  145    0.940     682.687  1     1002    0     0     0      2
 6                  109    0.782     755.160  1     1003    0     0     0      1
 7                   47    0.344     770.383  1     1005    0     0     0      0
 8                    9    0.069     812.573  1        0    0     0     0      0


QUEUE              MAX CONT. ENTRY ENTRY(0) AVE.CONT. AVE.TIME   AVE.(-0) RETRY
 1                   4    4    196       1     3.810   2046.712  2057.208   0
 2                   4    3    184       1     3.688   2110.269  2121.801   0
 3                   4    4    174       1     3.583   2167.832  2180.363   0
 4                   4    2    156       1     3.359   2266.956  2281.581   0
 5                   4    3    147       9     2.642   1892.319  2015.731   0
 6                   4    1    110      10     1.791   1714.156  1885.571   0
 7                   4    0     47      10     0.638   1429.604  1815.983   0
 8                   3    0      9       3     0.082    955.457  1433.185   0


SAVEVALUE               RETRY       VALUE
 SUM                      0    2668159.507
```

```
 SR_SUM                    0       2668.160


CEC XN    PRI          M1       ASSEM  CURRENT  NEXT  PARAMETER    VALUE
  1002     0       102505.508   1002       6      7
                                                      NUM            5.000


FEC XN    PRI          BDT      ASSEM  CURRENT  NEXT  PARAMETER    VALUE
  1004     0       105344.748   1004       8      9
                                                      NUM            1.000
  1024     0       105452.506   1024       0      1
  1006     0       105690.218   1006       8      9
                                                      NUM            2.000
  1001     0       105705.546   1001       8      9
                                                      NUM            4.000
  1005     0       105732.303   1005       8      9
                                                      NUM            7.000
  1007     0       105748.464   1007       8      9
                                                      NUM            3.000
  1003     0       106079.712   1003       8      9
                                                      NUM            6.000
```

Пример применения блока BUFFER в сочетании с GATE и TEST.

Пусть в систему поступают заявки, со средним интервалом 100 единиц, распределенным по экспоненциальному закону. Они должны группироваться по 10, и поступать на обслуживание 11 канальное устройство. Заявка, которая обнаружила 10 заявок, должна впустить их в устройство всех вместе, а сама остаться для формирования новой группы. Время обслуживания в многоканальном устройстве равно 900+-300. Если в момент попытки войти в многоканальное устройство, в нем останутся заявки, то все 10 заявок группы, должны быть удалены из системы. Промоделировать успешное обслуживание 1000 заявок.

Текст этой модели может быть, например, таким.

```
stor storage 11
generate (exponential(1,0,100))
test e X$num,10,next
logic s key
buffer
logic r key
next savevalue num+,1
gate ls key
GATE SE STOR,OUT
buffer
enter stor
savevalue num-,1
advance 900,300
leave stor
terminate 1
out savevalue num-,1
 terminate
start 1000
```

В данной задаче блоки BUFFER нужно применить как для заявки, которая является инициатором впуска других заявок, так и для впущенных заявок,

для корректной работы блока Gate se stor,out. Когда заявка - инициатор впуска обрабатывается, то вначале она устанавливает ключ KEY в положение включено. Затем она становится последней в списке подвижных заявок, и поэтому задержанные заявки имеют возможность пройти через два блока Gate. Они также ставятся в конец списка потенциально подвижных заявок. Поэтому дальше движется заявка – инициатор, и выключает ключ. Она в результате оказывается первой задержанной заявкой в новой группе. А заявки прошедшие через два Gate, получают возможность войти все одновременно в многоканальное устройство. Остальные детали работы модели представляются достаточно очевидными, и поэтому не обсуждаются.

Выходная статистика этой достаточно не простой задачи, имеет вид:

```
            GPSS World Simulation Report - Untitled Model 1.23.1

                 Monday, October 06, 2003 10:49:19

          START TIME           END TIME  BLOCKS  FACILITIES  STORAGES
               0.000         172966.730    16        0          1

             NAME                      VALUE
         KEY                        10002.000
         NEXT                           6.000
         NUM                        10001.000
         OUT                           15.000
         STOR                       10000.000


 LABEL              LOC  BLOCK TYPE     ENTRY COUNT CURRENT COUNT RETRY
                    1    GENERATE          1669             0       0
                    2    TEST              1669             0       0
                    3    LOGIC              166             0       0
                    4    BUFFER             166             0       0
                    5    LOGIC              166             0       0
NEXT                6    SAVEVALUE         1669             9       0
                    7    GATE              1660             0       0
                    8    GATE              1660             0       0
                    9    BUFFER            1000             0       0
                   10    ENTER             1000             0       0
                   11    SAVEVALUE         1000             0       0
                   12    ADVANCE           1000             0       0
                   13    LEAVE             1000             0       0
                   14    TERMINATE         1000             0       0
OUT                15    SAVEVALUE          660             0       0
                   16    TERMINATE          660             0       0


STORAGE            CAP. REM. MIN. MAX.  ENTRIES AVL.  AVE.C. UTIL. RETRY DELAY
 STOR               11   11   0    10     1000   1    5.224  0.475    0    0


LOGICSWITCH          VALUE        RETRY
 KEY                   0            9


SAVEVALUE               RETRY       VALUE
 NUM                      0          9.000


FEC XN   PRI         BDT      ASSEM  CURRENT  NEXT  PARAMETER    VALUE
  1670    0      173069.965   1670      0      1
```

24. Блоки работы с файлами.

Эти блоки предназначены для работы с текстовыми файлами в GPSS.

Для открытия текстового файла используется блок

OPEN A,B,C

Здесь A – это имя файла, которое записывается в кавычках. В качестве имени лучше указать полный путь к файлу и его расширение. B – это номер файла, по умолчанию, он равен 1. Номер файла используется в других операторах работы с файлами. Имейте в виду, что строковая константа – это выражение, а поэтому его следует записывать в круглых скобках. Каждый открытый файл должен иметь уникальный номер. C – это метка или номер блока, куда уйдет заявка, если при открытии файла возникнут ошибки, например будет сделана попытка открыть уже открытый файл.

Если файла с данным именем не существует, то создается пустой файл.

Пример:

OPEN (“TestFile.txt”),2,LabError

Блок открывает файл и для чтения и для записи. То есть в открытый файл можно как записывать данные, так и читать их. Однако удобнее использовать файл или только для чтения, или только для записи.

После открытия, файл установлен на первую строку. Запись в файл и чтение из файла ведется строками. В этих строках допускаются только символы первое половины кодовой таблицы, то есть символы, с кодом не более 127. Поэтому, в частности, в файлах и строковых константах, недопустимы символы кириллицы.

В GPSS предусмотрено автоматическое преобразование числовых и строковых данных друг в друга, в соответствии с контекстом, или с помощью специальной функции.

Строковые данные могут сохраняться в метках, X –параметрах, и в P – параметрах.

Возможно использование файла на диске, или файла непосредственно в памяти. Файлу в памяти соответствует пустое имя. На самом деле, любой файл сохраняется в буферной памяти, до выполнения блока или операции CLOSE.

Блок CLOSE закрывает открытый файл, и имеет следующий вид:

CLOSE A,B,C

Здесь

A – номер или имя параметра заявки для записи кода ошибки.

B – номер файла, по умолчанию,1.

C – метка или номер блока, куда уйдет заявка по ошибке, то есть если код ошибки не нулевой. Даже если само закрытие файла не вызывает ошибки, возможно, были ошибки в ходе работы с файлом. Тогда код первой ошибки с ненулевым номером сохраняется системой и доступен блоку CLOSE. Последующие ошибки – игнорируются, даже если они были.

Пример:

CLOSE Error_P_Param,2

Фактическое сохранение файла на диске происходит только в момент выполнения блока или процедуры CLOSE.

Блок READ имеет следующий вид:

READ A,B,C

Здесь A – номер или имя параметра заявки для записи в него строки из файла. То есть чтение строки происходит в P – параметр.

B – номер файла, по умолчанию,1.

C – метка или номер блока, куда уйдет заявка по ошибке, то есть если код ошибки не нулевой.

Пример:

READ P_Param,1,Error_Lab

Текстовая строка файла - определяется номером текущей строки, который отслеживается в файле. После блока чтения, номер текущей строки увеличивается на 1. Если файл прочитан до конца, то в P параметр ничего не записывается, и заявка уходит по метке.

Примечание: в текстовом файле могут быть только символы с кодами от 32 до 127.

Блок SEEK используется для установки номера текущей строки в файле. Он имеет вид:

SEEK A,B

Здесь:

A – новый номер строки.

B – номер файла, по умолчанию 1.

Пример:

SEEK 20,3

Не исключено, что строки с нужным номером в файле не окажется.

Тогда чтение из файла окажется невозможным, а результат записи зависит о способа записи. (параметр D блока WRITE).

Блок WRITE записывает строку в файл. Он имеет вид:

WRITE A,B,C,D

Здесь A – строка текста, возможно в виде выражения.

B – номер файла, по умолчанию 1.

C – метка или номер блока, куда заявка уходит по ошибке.

D – режим записи строки в файл.

Возможен режим вставки или замены. Номер текущей строки по ON или по умолчанию, используется в режиме вставки, а по OFF, в режиме замены.

Режим вставки.

Если номер текущей строки соответствует одной их строк файла, то текущая строка и все последующие строки сдвигаются вниз. Если номер текущей строки больше номера последней строки, номер текущей строки становится на 1 больше номера последней строки. Новая строка помещается в текущую строку. Номер текущей строки увеличивается на 1.

Режим замены.

Если номер текущей строки соответствует одной их строк файла, то текущая строка получает новое значение. Если номер текущей строки больше номера последней строки, то файл пополняется пустыми строками. Строка с

текущим номером заменяется на новую строку. . Номер текущей строки увеличивается на 1.

Рассмотрим примеры работы с файлами.

Пусть в систему поступают заявки со средним интервалом 100 единиц, распределенным по экспоненциальному закону. Далее они обслуживаются одноканальным устройством за среднее время, равное 99.1, распределенное также по экспоненциальному закону. Вывести в файл "my_test.txt" ,номера заявок и времена их обслуживания группами по 5 заявок, открывая и закрывая файл, после вывода сведений о 5 заявках.

Текст такой модели может выглядеть, например, следующим образом.

```
generate 0,0,0,1
open ("my_test.txt"),1
terminate

 generate (exponential(1,0,99.1))
 queue ust1
 seize ust1
 depart ust1
 advance (exponential(1,0,99.1))
 release ust1
 write xn1,1
 write m1,1
 test e tg1,1,next
 close 1
 open ("my_test.txt"),1
next terminate 1
 start 5
 start 5
 start 5
```

Информация в файле "my_test.txt", выглядит несколько неожиданно.

```
12
562.5770308048654
13
492.5630312536567
14
476.4209258322892
15
239.5292685256686
16
217.1971056297193
7
250.0998758428123
8
246.5221416684337
9
```

```
360.2383020406099
10
375.8231442702685
11
462.932742408133
2
133.3072675944952
3
497.1623262176024
4
457.7763524183413
5
292.5509959257575
6
294.4114104210099
```

Дело в том, что после открытия файла, информация вставляется в начало файла. Чтобы выводимая информация выглядела более традиционно, текст модели можно изменить, например, следующим образом:

```
generate 0,0,0,1
open ("my_test.txt"),1
terminate

generate (exponential(1,0,100))
queue ust1
seize ust1
depart ust1
advance (exponential(1,0,99.1))
release ust1
write xn1,1
write m1,1
test e tg1,1,next
close 1
open ("my_test.txt"),1
seek 1000,1
next terminate 1
start 5
start 5
start 5
```

Текст файла тогда будет выглядеть более традиционно:

```
2
133.3072675944952
3
497.1623262176024
4
457.7763524183413
5
```

292.5509959257575
6
294.4114104210099
7
250.0998758428123
8
246.5221416684337
9
360.2383020406099
10
375.8231442702685
11
462.932742408133
12
562.5770308048654
13
492.5630312536567
14
476.4209258322892
15
239.5292685256686
16
217.1971056297193

Файл в таком виде, все равно выглядит не слишком хорошо, так как в каждой строке выведено только одно число, да еще и без всяких пояснений. Чтобы он выглядел лучше, нужно воспользоваться функциями, предназначенными для работы со строками, например.

```
generate 0,0,0,1
open ("my_test.txt"),1
terminate

generate (exponential(1,0,100))
queue ust1
seize ust1
depart ust1
advance (exponential(1,0,99.1))
release ust1
write (PolyCatenate("xn1= ",xn1," m1= ",m1)),1
test e tg1,1,next
close 1
open ("my_test.txt"),1
seek 1000,1
next terminate 1
start 5
start 5
start 5
```

Тогда выходной файл имеет вид:

xn1= 2 m1= 133.3072675944952
xn1= 3 m1= 497.1623262176024
xn1= 4 m1= 457.7763524183413
xn1= 5 m1= 292.5509959257575
xn1= 6 m1= 294.4114104210099
xn1= 7 m1= 250.0998758428123
xn1= 8 m1= 246.5221416684337
xn1= 9 m1= 360.2383020406099
xn1= 10 m1= 375.8231442702685
xn1= 11 m1= 462.932742408133
xn1= 12 m1= 562.5770308048654
xn1= 13 m1= 492.5630312536567
xn1= 14 m1= 476.4209258322892
xn1= 15 m1= 239.5292685256686
xn1= 16 m1= 217.1971056297193

Если использовать операторы записи в режиме замены, то получим следующие результаты. Текст модели.

```
generate 0,0,0,1
open ("my_test.txt"),1
terminate

generate (exponential(1,0,100))
queue ust1
seize ust1
depart ust1
advance (exponential(1,0,99.1))
release ust1
write (PolyCatenate("xn1= ",xn1," m1= ",m1)),1,,off
test e tg1,1,next
close 1
open ("my_test.txt"),1
next terminate 1
start 5
start 5
start 5
```

Здесь не нужно использовать Seek, и происходит двукратная перезапись данных в файл поверх первоначальных, выведенных при первом Start.

Текст файла.

xn1= 12 m1= 562.5770308048654
xn1= 13 m1= 492.5630312536567
xn1= 14 m1= 476.4209258322892
xn1= 15 m1= 239.5292685256686
xn1= 16 m1= 217.1971056297193

Операторы чтения могут использоваться, например, для задания исходных данных. Пусть в том же текстовом файле заданы времена для двух

последовательных задержек в одноканальном устройстве. Тогда текст такой модели мог бы, например, иметь такой вид:

```
initial x3,100
generate 0,0,0,1,1
open ("my_test.txt"),1
read 1,1
savevalue 1,p1
read 1,1
savevalue 2,p1
close 1
terminate

generate (exponential(1,0,X3))
queue ust1
seize ust1
depart ust1
advance X1,(0.2#X1)
advance (exponential(1,0,X2))
release ust1
terminate 1
start 1000
```

Содержимое файла могло бы иметь, например, следующий вид:

```
50
49.1
```

Выходной файл в этом случае будет иметь следующий вид:

```
    GPSS World Simulation Report - Untitled Model 1.23.1

         Friday, September 20, 2013 20:37:15

 START TIME           END TIME  BLOCKS  FACILITIES  STORAGES
      0.000         104044.672    16        1          0


    NAME                     VALUE
UST1                       10000.000


LABEL             LOC  BLOCK TYPE     ENTRY COUNT CURRENT COUNT RETRY
                    1  GENERATE             1             0       0
                    2  OPEN                 1             0       0
                    3  READ                 1             0       0
                    4  SAVEVALUE            1             0       0
                    5  READ                 1             0       0
                    6  SAVEVALUE            1             0       0
                    7  CLOSE                1             0       0
                    8  TERMINATE            1             0       0
                    9  GENERATE          1009             0       0
                   10  QUEUE             1009             8       0
                   11  SEIZE             1001             1       0
                   12  DEPART            1000             0       0
                   13  ADVANCE           1000             0       0
                   14  ADVANCE           1000             0       0
                   15  RELEASE           1000             0       0
                   16  TERMINATE         1000             0       0
```

```
FACILITY          ENTRIES  UTIL.   AVE. TIME AVAIL. OWNER PEND INTER RETRY DELAY
 UST1               1001    0.971     100.922  1     1002    0    0     0      8


QUEUE              MAX CONT. ENTRY ENTRY(0) AVE.CONT. AVE.TIME   AVE.(-0) RETRY
 UST1               24    9   1009     25     7.534    776.896    796.634   0


SAVEVALUE               RETRY       VALUE
 1                        0         "50"
 2                        0         "49.1"
 3                        0         100.000


CEC XN   PRI          M1      ASSEM  CURRENT  NEXT  PARAMETER    VALUE
  1002    0      103095.429   1002     11     12


FEC XN   PRI          BDT      ASSEM  CURRENT  NEXT  PARAMETER    VALUE
  1011    0      104072.616   1011      0      9
```

25. Функции работы со строками.

Эти функции могут использоваться для манипуляций со строками при вводе – выводе данных, а также для формирования значений X и P – параметров. Для работы со строками можно использовать исключительно функции. В строках можно использовать исключительно символы с кодами от 32 до 127.

Имеется следующий набор функций для работы со строками:

1. Align(строка1,строка2,номер)

Выравнивание. Строка1 накладывается на строку2. Конец строки1 накладывается на строку2 правым краем в позиции номер. Например, выполнив

Align(“TEPA”,“Stepanov”,5)

Получим

“STEPAnov”

2. Catenate(строка1,строка2)

Сцепление строки1 и строки2. Например, выполнив

Catenate(“Ivan”,“ov”)

Получим

“Ivanov”

3. Copies(строка, количество)

Строка дублируется указанное количество раз и копии сцепляются. Например, выполнив

Copies(“book_”,4)

Получим

“book_book_book_book_”

4. DataType(данное)

Функция проверяет тип данных и возвращает строку, содержащую название типа данных. Эти названия – следующие:

“INTEGER” – данное - 32 битовое целое.

“REAL” – данное – вещественное число двойной точности.
“STRING” - данное – строка символов.
“UNSPECIFIED” – данное не имеет никакого значения.
Пример
DataType(“TABLE”)
Возвращает
“STRING”

5. Find(строка1,строка2)
В строке2 отыскивается первое вхождение строки1. Возвращается позиция, где обнаружено начало строки1. При неуспехе поиска – возвращается 0.
Пример
Find(“dog”,“The dog”)
Результат
5

6. Left(строка, число)
Из строки выделяется не более, чем указанное число символов, начиная с первого, то есть, слева.
Пример
Left(“The_dog”,4)
Результат
“The_”

7. Length(строка)
Определяется число символов в строке.
Пример
Length(“The_dog__”)
Результат - число 9

8. LowerCase(строка)
Приводит буквы латинского алфавита в строке к нижнему регистру.
Пример
Lowercase(“The Dog –1998.”)
Результат
“the dog –1998.”

9. Place(строка1,строка2,позиция)
Накладывает строку1 поверх строки2, начало наложения определяет позиция.
Пример
Place(“Flat”,“The Dog”,4)
Результат
“The Flat”

10. PolyCatenate(сторка1,строка2 ...)

Сцепляются строки указанные в качестве аргументов. Число аргументов может быть любым.

Пример

PolyCatenate(“The ”,“time ”,“is ”,M1,” !”)

Результат

“The time is 323.567 !”

В этом примере происходит автоматическое преобразование значения M1 в строку и сцепление строк.

11. Right(строка,число)

Выделяет в строке максимум указанное число символов справа, то есть с конца строки.

Пример

Right(“The Dog”,4)

Результат

“ Dog”

12. String(данные)

Преобразует произвольные данные в строку явным образом.

Пример

String(12345+10)

Результат

“12355”

13. StringCompare(строка1, строка2)

Сравнивает строку1 и строку2.

Возвращает -1, 0, или 1 если строк1 меньше, равна, или больше строки2.

Пример

StringCompare(“The dog”,“the dog”)

Результат

-1

Эта функция часто используется в выражениях, где нужно сравнить строки, так как иначе происходит сравнение строк после их неявного преобразования в числа.

14. SubString(строка,позиция,число)

Выделяет подстроку в строке, начиная с указанной позиции. Выделяется указанное число символов.

Пример

Substring(“The dog is sleep”,9,5)

Результат

“is sl”

15. Trim(сторка)

Удаляет ведущие и хвостовые пробельные символы.

Пример

Trim(“ C A LL ”)
Результат
“C A LL”

16. UpperCase(SourceString)
Приводит латинские буквы к верхнему регистру.
Пример
Uppercase(“The Dog Is Sleep ”)
Результат
“THE DOG IS SLEEP”

17. Value(строка)
Преобразует строку – к числу. Начальная часть строки интерпретируется как вещественное число.
Пример
value(polycatenate(SQR(ac1),"time"))
Результат, например.
54.77225575051661

18. Word(SourceString, WordNumber)
Выделяет в строке слово с заданным номером. Слова разделяются пробелами и табуляциями.
Пример
Word(“My country Ukraina.”,2)
Результат
“country”
Пример применения строковых функций.
Рассмотрим следующую задачу, которую мы уже рассматривали при изучении блока GATE. Пусть имеется железнодорожный переезд со шлагбаумом. Автомобили с каждой стороны шлагбаума подъезжают со средним интервалом в 20 секунд, распределенным по экспоненциальному закону. Через шлагбаум одновременно могут переезжать два автомобиля. Время переезда – 15+-3 секунды. В среднем через 3600+- 900 секунд проходит поезд, и шлагбаум закрывается на 300+-60 секунд. Промоделировать работу переезда в течение суток. Для обозначения состояния шлагбаума, использовать слова OPEN и CLOSE.
Текст модели может выглядеть следующим образом.

```
INITIAL X$SHLAGB,"OPEN"
PERE   STORAGE 2
TT1 TABLE M1,0,10,50
TT2 TABLE Q1,0,3,50

 GENERATE (exponential(1,0,20))
 QUEUE 1
TEST E (StringCompare(X$SHLAGB,"OPEN")),0
 ENTER PERE
 DEPART 1
```

```
 ADVANCE 15,3
 LEAVE PERE
 TABULATE TT1
 TERMINATE

 GENERATE (exponential(1,0,20))
QUEUE 1
TEST E (StringCompare(X$SHLAGB,"OPEN")),0
 ENTER PERE
 DEPART 1
 ADVANCE 15,3
 LEAVE PERE
 TABULATE TT1
 TERMINATE

GENERATE 3600,900
SAVEVALUE  SHLAGB,"CLOSE"
 ADVANCE 300,60
SAVEVALUE SHLAGB,"OPEN"
TABULATE TT2
TERMINATE

GENERATE (3600#24)
TERMINATE 1
   START 1
```

```
    GPSS World Simulation Report - Untitled Model 1.2.1


          Sunday, September 21, 2003 16:01:07

 START TIME           END TIME  BLOCKS  FACILITIES  STORAGES
      0.000          86400.000    26        0          1


    NAME                     VALUE
PERE                       10001.000
SHLAGB                     10000.000
TT1                        10002.000
TT2                        10003.000

LABEL              LOC  BLOCK TYPE     ENTRY COUNT CURRENT COUNT RETRY
                     1  GENERATE          4174              0       0
                     2  QUEUE             4174              0       0
                     3  TEST              4174              0       0
                     4  ENTER             4174              0       0
                     5  DEPART            4174              0       0
                     6  ADVANCE           4174              0       0
                     7  LEAVE             4174              0       0
                     8  TABULATE          4174              0       0
                     9  TERMINATE         4174              0       0
                    10  GENERATE          4280              0       0
                    11  QUEUE             4280              0       0
                    12  TEST              4280              1       0
                    13  ENTER             4279              0       0
```

14	DEPART	4279	0	0
15	ADVANCE	4279	2	0
16	LEAVE	4277	0	0
17	TABULATE	4277	0	0
18	TERMINATE	4277	0	0
19	GENERATE	22	0	0
20	SAVEVALUE	22	0	0
21	ADVANCE	22	0	0
22	SAVEVALUE	22	0	0
23	TABULATE	22	0	0
24	TERMINATE	22	0	0
25	GENERATE	1	0	0
26	TERMINATE	1	0	0

QUEUE	MAX	CONT.	ENTRY	ENTRY(0)	AVE.CONT.	AVE.TIME	AVE.(-0)	RETRY
1	44	1	8454	2386	5.030	51.403	71.615	0

STORAGE	CAP.	REM.	MIN.	MAX.	ENTRIES	AVL.	AVE.C.	UTIL.	RETRY	DELAY
PERE	2	0	0	2	8453	1	1.467	0.734	0	1

TABLE	MEAN	STD.DEV.	RANGE			RETRY	FREQUENCY	CUM.%
TT1	66.420	84.250				0		
			10.000	-	20.000		3237	38.30
			20.000	-	30.000		1578	56.98
			30.000	-	40.000		749	65.84
			40.000	-	50.000		422	70.83
			50.000	-	60.000		230	73.55
			60.000	-	70.000		154	75.38
			70.000	-	80.000		134	76.96
			80.000	-	90.000		99	78.13
			90.000	-	100.000		87	79.16
			100.000	-	110.000		86	80.18
			110.000	-	120.000		103	81.40
			120.000	-	130.000		83	82.38
			130.000	-	140.000		68	83.19
			140.000	-	150.000		96	84.32
			150.000	-	160.000		61	85.04
			160.000	-	170.000		71	85.88
			170.000	-	180.000		82	86.85
			180.000	-	190.000		69	87.67
			190.000	-	200.000		73	88.53
			200.000	-	210.000		61	89.26
			210.000	-	220.000		57	89.93
			220.000	-	230.000		83	90.91
			230.000	-	240.000		84	91.91
			240.000	-	250.000		78	92.83
			250.000	-	260.000		78	93.75
			260.000	-	270.000		72	94.60
			270.000	-	280.000		66	95.39
			280.000	-	290.000		82	96.36
			290.000	-	300.000		75	97.24
			300.000	-	310.000		59	97.94
			310.000	-	320.000		40	98.41
			320.000	-	330.000		41	98.90
			330.000	-	340.000		31	99.27
			340.000	-	350.000		18	99.48
			350.000	-	360.000		16	99.67
			360.000	-	370.000		14	99.83
			370.000	-	380.000		3	99.87
			380.000	-	390.000		9	99.98
			390.000	-	400.000		2	100.00
TT2	30.364	6.060				0		
			15.000	-	18.000		1	4.55
			18.000	-	21.000		0	4.55
			21.000	-	24.000		1	9.09
			24.000	-	27.000		6	36.36
			27.000	-	30.000		4	54.55

```
                                30.000  -       33.000            3    68.18
                                33.000  -       36.000            3    81.82
                                36.000  -       39.000            2    90.91
                                39.000  -       42.000            1    95.45
                                42.000  -       45.000            1   100.00


SAVEVALUE               RETRY       VALUE
 SHLAGB                   0         "OPEN"


FEC XN    PRI         BDT      ASSEM  CURRENT  NEXT  PARAMETER    VALUE
  8480     0       86400.401   8480       0     10
  8477     0       86401.050   8477      15     16
  8478     0       86407.101   8478      15     16
  8476     0       86429.893   8476       0      1
  8351     0       87894.490   8351       0     19
  8481     0      172800.000   8481       0     25
```

26. Блоки работы с семействами заявок.

В GPSS имеются блоки, обеспечивающие работу с семействами.

Главный среди них - блок SPLIT, который имеет следующий формат:

SPLIT A,B,C

Блок SPLIT создает копии вошедшей в него заявки.

В поле A задается число создаваемых копий.

После создания копий заявка пытается перейти к следующему по номеру блоку.

Поле B задает номер блока, к которому переходят копии.

В поле C может быть задан номер P - параметра, используемого для присвоения копиям последовательных номеров.

Если, например, задан параметр 3, и в родительской заявке он равен X, то параметру P3 копий будут присвоены значения X+1, X+2, X+3 и т.д.

Помимо значений параметров, в каждую копию записывается значение приоритета и отметка времени исходной заявки.

Каждая копия становится членом семейства заявок, порожденного исходной заявкой, которая была создана блоком GENERATE. Сама эта заявка также входит в семейство.

Если копия заявки входит в блок SPLIT, то вторичная копия становится членом того же семейства, что и первичная копия.

В модели одновременно может существовать произвольное число семейств, оно все время меняется, поскольку каждая генерируемая блоком GENERATE заявка создает новое семейство из одной заявки.

Узнать номер семейства активной заявки можно с помощью стандартного числового атрибута A1.

Любую заявку можно принудительно включить в семейство с данным номером, с помощью блока ADOPT, который имеет вид:

ADOPT A

Здесь A – это номер семейства. В блоке SPLIT в качестве номера семейства используется номер родительской заявки.

Понятие семейства используется блокам ASSEMBLE, GATHER, MATCH.

Блок ASSEMBLE имеет следующий формат:

ASSEMBLE A

Он объединяет заданное число заявок, принадлежащих к одному семейству, в одну заявку (т.е. осуществляет сборку заданного числа заявок). После накопления, из блока ASSEMBLE выходит только одна заявка – инициатор сборки, которая переходит в следующий по номеру блок. В одном и том же блоке ASSEMBLE возможна одновременная сборка нескольких семейств.

Поле A задает число заявок, участвующих в сборке.

Пример использования блоков SPLIT и ASSEMBLE

Пусть в систему передачи данных поступают информационные сообщения со средним интервалом 100, распределенным по экспоненциальному закону. Каждое сообщение разбивается на 3..17 блоков. Количество блоков распределено равномерно. Блок передается за 300+-290 единиц времени. На приемном конце заявки вновь собираются из блоков. Вывести в файл информацию о времени конца передачи сообщения, времени передачи, числе блоков, и номере семейства.

Тогда модель такой системы может иметь вид:

```
generate (exponential(1,0,100))
assign num,(duniform(1,3,17))
split (P$num-1),next
next advance 300,290
assemble P$num
write (polycatenate("ac1=",ac1," m1=",m1," p=",P$num," a1=",a1)),1
terminate
generate 0,0,0,1
open ("my_test.txt"),1
advance 2000
close 1
terminate 1
```

Выходная статистика модели имеет вид:

```
    GPSS World Simulation Report - Untitled Model 1.1.1
         Friday, September 26, 2003 06:31:50

 START TIME           END TIME  BLOCKS  FACILITIES  STORAGES
      0.000           2000.000    12        0          0


    NAME                      VALUE
NEXT                           4.000
NUM                        10000.000


 LABEL             LOC  BLOCK TYPE     ENTRY COUNT CURRENT COUNT RETRY
                   1    GENERATE            24             0       0
                   2    ASSIGN              24             0       0
                   3    SPLIT               24             0       0
NEXT               4    ADVANCE            218            37       0
                   5    ASSEMBLE           181             8       0
                   6    WRITE               16             0       0
                   7    TERMINATE           16             0       0
                   8    GENERATE             1             0       0
                   9    OPEN                 1             0       0
```

10	ADVANCE	1	0	0
11	CLOSE	1	0	0
12	TERMINATE	1	0	0

FEC XN	PRI	BDT	ASSEM	CURRENT	NEXT	PARAMETER	VALUE
155	0	2007.061	144	4	5		
						NUM	9.000
195	0	2007.235	181	4	5		
						NUM	5.000
218	0	2015.751	196	4	5		
						NUM	8.000
149	0	2024.501	144	4	5		
						NUM	9.000
192	0	2026.389	181	4	5		
						NUM	5.000
156	0	2028.871	144	4	5		
						NUM	9.000
217	0	2030.349	196	4	5		
						NUM	8.000
176	0	2030.401	157	4	5		
						NUM	8.000
131	0	2031.926	131	4	5		
						NUM	7.000
191	0	2037.081	191	4	5		
						NUM	17.000
143	0	2042.393	131	4	5		
						NUM	7.000
213	0	2054.634	213	0	1		
162	0	2057.046	148	4	5		
						NUM	16.000
212	0	2072.234	191	4	5		
						NUM	17.000
167	0	2076.152	148	4	5		
						NUM	16.000
170	0	2080.078	148	4	5		
						NUM	16.000
207	0	2101.222	191	4	5		
						NUM	17.000
179	0	2107.021	157	4	5		
						NUM	8.000
188	0	2121.186	173	4	5		
						NUM	10.000
173	0	2122.656	173	4	5		
						NUM	10.000
189	0	2130.673	173	4	5		
						NUM	10.000
180	0	2146.606	157	4	5		
						NUM	8.000
157	0	2152.610	157	4	5		
						NUM	8.000
174	0	2157.065	157	4	5		
						NUM	8.000
198	0	2157.551	191	4	5		
						NUM	17.000
181	0	2160.991	181	4	5		
						NUM	5.000
205	0	2167.973	191	4	5		
						NUM	17.000
199	0	2187.599	191	4	5		
						NUM	17.000
203	0	2222.210	191	4	5		
						NUM	17.000
216	0	2227.871	196	4	5		
						NUM	8.000
211	0	2237.269	191	4	5		
						NUM	17.000
215	0	2273.898	196	4	5		
						NUM	8.000
197	0	2354.235	191	4	5		

```
                                                    NUM          17.000
208    0        2361.985    191      4      5
                                                    NUM          17.000
220    0        2408.954    196      4      5
                                                    NUM           8.000
210    0        2419.853    191      4      5
                                                    NUM          17.000
219    0        2514.902    196      4      5
                                                    NUM           8.000
214    0        2549.316    196      4      5
                                                    NUM           8.000
```

Информация в выходном файле имеет вид:

ac1=573.2089743218381 m1=519.56538 p=4 a1=9
ac1=577.3374279850588 m1=560.18336 p=6 a1=1
ac1=598.6287548612744 m1=568.8039 p=12 a1=3
ac1=758.2207229764802 m1=460.5788000000001 p=4 a1=25
ac1=825.3908393645681 m1=570.1065800000001 p=14 a1=21
ac1=901.8539412426507 m1=568.10384 p=14 a1=39
ac1=926.1783832939443 m1=504.2011799999999 p=3 a1=43
ac1=1002.427044887319 m1=421.90846 p=3 a1=57
ac1=1226.415078620566 m1=465.04712 p=8 a1=80
ac1=1306.019681088619 m1=555.2052199999999 p=12 a1=63
ac1=1310.391011843049 m1=564.5762800000001 p=17 a1=60
ac1=1665.998153464734 m1=569.0689600000001 p=11 a1=92
ac1=1688.816930248672 m1=485.2514200000001 p=13 a1=111
ac1=1753.572018768225 m1=564.9219600000001 p=7 a1=100
ac1=1878.636938458339 m1=519.4447399999999 p=6 a1=118
ac1=1952.848146648731 m1=467.8694 p=4 a1=137

Судя по информации из выходного файла, некоторые позднее поступившие на вход сообщения, на выходе появились раньше, что соответствует предполагаемым свойствам блока ASSEMBLE.

Рассмотрим модель системы с передачей информационных сообщений.

Пусть сообщения поступают в систему со средним интервалом времени в 750 единиц. Время распределено по экспоненциальному закону. Пусть число сегментов, на которые разбивается сообщение равномерно распределено в интервале от 3 до 12. Пусть на передачу одного сегмента по линии связи требуется 90+-20 единиц времени. После передачи каждого сегмента по линии связи, все они последовательно обрабатываются устройством, объединяясь в одно сообщение. Обработка в устройстве занимает 80+-50 единиц времени. Провести моделирование передачи и обработки 1000 сообщений. Текст такой модели может выглядеть следующим образом.

```
GENERATE (EXPONENTIAL(1,0,750))
QUEUE OCH
SEIZE    LINE1
DEPART OCH
ASSIGN   1,(DUNIFORM(1,3,12))
ASSIGN   2,P1
```

```
ADVANCE  90,20
TRANSFER  ,BBE2
BBC2 SPLIT    1,BG2
ADVANCE  90,20
BBE2  LOOP    1,BBC2
RELEASE  LINE1
BG2   QUEUE UST
SEIZE UST
DEPART UST
ADVANCE 80,50
RELEASE UST
ASSEMBLE  P2
TERMINATE 1
START 1000
```

Итоговая статистика этой модели будет иметь вид:

```
   GPSS World Simulation Report - Untitled Model 1.11.1

         Monday, September 29, 2003 18:11:32

  START TIME           END TIME  BLOCKS  FACILITIES  STORAGES
       0.000         759428.885    19        2          0

     NAME                       VALUE
BBC2                            9.000
BBE2                           11.000
BG2                            13.000
LINE1                       10001.000
OCH                         10000.000
UST                         10002.000

 LABEL              LOC  BLOCK TYPE     ENTRY COUNT CURRENT COUNT RETRY
                     1   GENERATE          1005             0       0
                     2   QUEUE             1005             4       0
                     3   SEIZE             1001             0       0
                     4   DEPART            1001             0       0
                     5   ASSIGN            1001             0       0
                     6   ASSIGN            1001             0       0
                     7   ADVANCE           1001             0       0
                     8   TRANSFER          1001             0       0
BBC2                 9   SPLIT             6584             0       0
                    10   ADVANCE           6584             1       0
BBE2                11   LOOP              7584             0       0
                    12   RELEASE           1000             0       0
BG2                 13   QUEUE             7584             1       0
                    14   SEIZE             7583             0       0
                    15   DEPART            7583             0       0
                    16   ADVANCE           7583             1       0
                    17   RELEASE           7582             0       0
                    18   ASSEMBLE          7582             0       0
                    19   TERMINATE         1000             0       0

FACILITY          ENTRIES  UTIL.   AVE. TIME AVAIL. OWNER PEND INTER RETRY DELAY
 LINE1              1001    0.898     680.924  1     7547    0     0     0      4
 UST                7583    0.796      79.718  1     7589    0     0     0      1

QUEUE              MAX CONT. ENTRY ENTRY(0) AVE.CONT. AVE.TIME   AVE.(-0) RETRY
 OCH                17    4   1005     91     4.350   3287.203   3614.485   0
 UST                 4    1   7584   2875     0.302     30.226     48.681   0

FEC XN   PRI         BDT      ASSEM  CURRENT  NEXT  PARAMETER    VALUE
  7564    0      759493.927   7564      0      1
  7589    0      759497.839   7547     16     17       1          8.000
```

```
                                                    2          9.000
  7547    0      759516.134   7547     10     11      1          7.000
                                                    2          9.000
```

Блок GATHER имеет следующий формат.

GATHER <A>

Здесь А – число заявок семейства, которые необходимо собрать. После сборки заданного числа заявок, все они одновременно появляются на выходе блока. Блок во многом похож на блок ASSEMBLE.

Пример задачи на использование блока мог бы выглядеть следующим образом. Пусть в систему поступают электродвигатели для мойки и сушки в специальном аппарате с интервалом 60 минут, распределенном по экспоненциальному закону. Аппарат включают, только когда есть 5 электродвигателей.

Процесс мойки и сушки занимает 280+-20 минут.

Промоделировать процесс в течение 2400 минут.

Модель такой задачи может иметь вид.

```
APPAR STORAGE 5
GENERATE (EXPONENTIAL(1,0,60))
ADOPT 1
QUEUE OCH
GATHER 5
ENTER APPAR
DEPART OCH
ADVANCE  280,20
GATHER 5
LEAVE APPAR
TERMINATE

GENERATE 2400
TERMINATE 1
START 1
```

Здесь блок Adopt позволяет все заявки считать принадлежащими одному (первому) семейству. Без него эта задача имела бы намного более громоздкое решение.

Выходная статистика имеет вид:

```
                GPSS World Simulation Report - Untitled Model 1.1.1

              Thursday, February 12, 2004 07:25:13

 START TIME           END TIME  BLOCKS  FACILITIES  STORAGES
      0.000           2400.000    12        0          1


    NAME                      VALUE
APPAR                      10000.000
OCH                        10001.000

LABEL             LOC  BLOCK TYPE     ENTRY COUNT CURRENT COUNT RETRY
                  1    GENERATE            36             0       0
                  2    ADOPT               36             0       0
```

```
                    3    QUEUE              36             0       0
                    4    GATHER             36             1       0
                    5    ENTER              35             0       0
                    6    DEPART             35             0       0
                    7    ADVANCE            35             5       0
                    8    GATHER             30             0       0
                    9    LEAVE              30             0       0
                   10    TERMINATE          30             0       0
                   11    GENERATE            1             0       0
                   12    TERMINATE           1             0       0


QUEUE              MAX CONT. ENTRY ENTRY(0) AVE.CONT. AVE.TIME   AVE.(-0) RETRY
 OCH                 6    1     36       6     1.986    132.424    158.909   0

STORAGE            CAP. REM. MIN. MAX.  ENTRIES AVL.  AVE.C. UTIL. RETRY DELAY
 APPAR               5    0   0     5       35   1    4.180  0.836    0    0


FEC XN   PRI         BDT      ASSEM  CURRENT  NEXT  PARAMETER    VALUE
    35    0        2417.154      1      7      8
    34    0        2424.567      1      7      8
    38    0        2440.868     38      0      1
    32    0        2441.921      1      7      8
    36    0        2450.455      1      7      8
    33    0        2454.862      1      7      8
    39    0        4800.000     39      0     11
```

Блок MATCH имеет следующий формат:

MATCH <A>

Блок MATCH используется для синхронизации движения пар заявок, принадлежащих к одному семейству, без удаления этих заявок из модели.

Блоки MATCH используются парами, и, как правило, имеют метки. В поле A указывается метка другого блока пары. То есть блоки MATCH ссылаются друг на друга. Такие блоки называются сопряженными. Когда заявка семейства входит в один из сопряженных блоков, а в другом нет заявки из этого семейства, то она задерживается в блоке. Если же в другом блоке есть заявка этого семейства, то обе они одновременно появляются на выходах сопряженных блоков. Одновременно в сопряженных блоках может синхронизироваться произвольное число пар заявок из различных семейств.

Пример использования блока MATCH.

Пусть в 8 часов утра должен начаться финальный конкурс лучших по профессии. Конкурсанты могут опоздать на 2.5+-2.5 минут, после чего начинается соревнование из 5 туров, которые выполняются подряд без подведения промежуточных итогов. Каждый тур выполняется первым участником за 10+-3 минуты, а вторым – за 9.8+-4 минуты. Когда оба участника завершат все 5 туров, происходит подведение итогов. Протабулировать время начала конкурса, и время ожидания каждого из конкурсантов, до завершения задания другим конкурсантом. Моделирование провести для 1000 таких конкурсов. Текст модели для этой задачи может иметь вид:

```
beg table m1,0,0.5,30
wait1 table m1,0,1,30
wait2 table m1,0,1,30

generate 0,0,0,1
mm5 split 1,next
```

```
mark
advance 2.5,2.5
mm1 match mm2
tabulate beg
advance 10,3
advance 10,3
advance 10,3
advance 10,3
advance 10,3
mark
mm3 match mm4
tabulate wait1
terminate 1
next mark
advance 2.5,2.5
mm2 match mm1
tabulate beg
advance 9.8,4
advance 9.8,4
advance 9.8,4
advance 9.8,4
advance 9.8,4
mark
mm4 match mm3
tabulate wait2
transfer ,mm5
start 1000
```

Итоговая статистика модели может иметь вид:

```
    GPSS World Simulation Report - Untitled Model 1.3.1

         Wednesday, October 01, 2003 08:38:27

 START TIME           END TIME  BLOCKS  FACILITIES  STORAGES
      0.000          55379.393    28        0          0

    NAME                      VALUE
BEG                          10000.000
MM1                              5.000
MM2                             18.000
MM3                             13.000
MM4                             26.000
MM5                              2.000
NEXT                            16.000
WAIT1                        10001.000
WAIT2                        10002.000

 LABEL             LOC  BLOCK TYPE     ENTRY COUNT CURRENT COUNT RETRY
                   1    GENERATE             1             0       0
MM5                2    SPLIT             1000             0       0
                   3    MARK              1000             0       0
                   4    ADVANCE           1000             0       0
MM1                5    MATCH             1000             0       0
                   6    TABULATE          1000             0       0
                   7    ADVANCE           1000             0       0
                   8    ADVANCE           1000             0       0
```

	9	ADVANCE	1000	0	0
	10	ADVANCE	1000	0	0
	11	ADVANCE	1000	0	0
	12	MARK	1000	0	0
MM3	13	MATCH	1000	0	0
	14	TABULATE	1000	0	0
	15	TERMINATE	1000	0	0
NEXT	16	MARK	1000	0	0
	17	ADVANCE	1000	0	0
MM2	18	MATCH	1000	0	0
	19	TABULATE	1000	0	0
	20	ADVANCE	1000	0	0
	21	ADVANCE	1000	0	0
	22	ADVANCE	1000	0	0
	23	ADVANCE	1000	0	0
	24	ADVANCE	1000	0	0
	25	MARK	1000	0	0
MM4	26	MATCH	1000	1	0
	27	TABULATE	999	0	0
	28	TRANSFER	999	0	0

TABLE	MEAN	STD.DEV.	RANGE			RETRY	FREQUENCY	CUM.%
BEG	3.342	1.202				0		
			0.000	-	0.500		22	1.10
			0.500	-	1.000		52	3.70
			1.000	-	1.500		136	10.50
			1.500	-	2.000		132	17.10
			2.000	-	2.500		164	25.30
			2.500	-	3.000		190	34.80
			3.000	-	3.500		256	47.60
			3.500	-	4.000		320	63.60
			4.000	-	4.500		346	80.90
			4.500	-	5.000		382	100.00
WAIT1	1.932	3.299				0		
				-	0.000		591	59.10
			0.000	-	1.000		52	64.30
			1.000	-	2.000		57	70.00
			2.000	-	3.000		47	74.70
			3.000	-	4.000		53	80.00
			4.000	-	5.000		48	84.80
			5.000	-	6.000		32	88.00
			6.000	-	7.000		23	90.30
			7.000	-	8.000		22	92.50
			8.000	-	9.000		19	94.40
			9.000	-	10.000		17	96.10
			10.000	-	11.000		15	97.60
			11.000	-	12.000		6	98.20
			12.000	-	13.000		3	98.50
			13.000	-	14.000		6	99.10
			14.000	-	15.000		2	99.30
			15.000	-	16.000		3	99.60
			16.000	-	17.000		2	99.80
			17.000	-	18.000		1	99.90
			18.000	-	19.000		0	99.90
			19.000	-	20.000		1	100.00
WAIT2	3.311	4.159				0		
				-	0.000		409	40.94
			0.000	-	1.000		61	47.05
			1.000	-	2.000		70	54.05
			2.000	-	3.000		58	59.86
			3.000	-	4.000		61	65.97
			4.000	-	5.000		52	71.17
			5.000	-	6.000		45	75.68
			6.000	-	7.000		54	81.08
			7.000	-	8.000		30	84.08
			8.000	-	9.000		35	87.59
			9.000	-	10.000		36	91.19
			10.000	-	11.000		23	93.49
			11.000	-	12.000		26	96.10

```
                                12.000  -       13.000            7    96.80
                                13.000  -       14.000           12    98.00
                                14.000  -       15.000            5    98.50
                                15.000  -       16.000            5    99.00
                                16.000  -       17.000            4    99.40
                                17.000  -       18.000            1    99.50
                                18.000  -       19.000            2    99.70
                                19.000  -       20.000            2    99.90
                                20.000  -       21.000            0    99.90
                                21.000  -       22.000            0    99.90
                                22.000  -       23.000            0    99.90
                                23.000  -       24.000            1   100.00


CEC XN    PRI          M1      ASSEM  CURRENT  NEXT  PARAMETER    VALUE
  1001     0       55372.565      1     26     27
```

Из выходной статистики видно, что первый конкурсант оказался вторым 591 раз и отставал от второго в среднем почти на 2 минуты. А второй конкурсант оказался вторым – 409 раз и отставал от первого в среднем на 3.3 минуты.

Среднее время начала конкурса оказалось на 3.342 смещенным относительно официального начала.

27. Организация циклов.

Для организации циклов можно использовать блок LOOP, который имеет следующий формат:

LOOP A,B

Здесь A – это номер или имя P – параметра, используемого для организации цикла, а B – метка блока, куда будет послана заявка, если цикл еще не завершен. Обычно метка расположена в модели выше блока LOOP. В принципе, блок не относится к числу необходимых блоков. Он может быть заменен парой блоков ASSIGN и TEST.

Когда заявка входит в блок LOOP, то значение P – параметра уменьшается на 1, и если оно все еще больше 0, то заявка переходит по метке. Если оно равно 0, то заявка проходит сквозь блок и следует дальше по модели. Если значение P – параметра не целое, или отрицательное, то моделирование прерывается по ошибке.

Пример использования блока LOOP:

Заявки обрабатываются устройством случайное число раз, от 1до 10. Кратность обработки определяется для заявки заранее и вероятность каждого из этих чисел одинакова. Интервал поступления заявок – 100+-80, интервал обслуживания – 17+-7.

Промоделировать работу системы по обслуживанию 100 заявок.

Модель такой системы может выглядеть следующим образом.

```
GENERATE 100,80
 ASSIGN 5,(duniform(1,1,10))
mmm QUEUE 1
  SEIZE 1
DEPART 1
PRIORITY 1
 ADVANCE 17,7
```

```
RELEASE 1
LOOP  5,mmm
TERMINATE 1
 START 100
```

Выходная статистика будет следующей.

```
   GPSS World Simulation Report - Untitled Model 1.10.1

        Wednesday, October 01, 2003 09:19:53

 START TIME           END TIME  BLOCKS  FACILITIES  STORAGES
      0.000          10860.151    10        1          0

    NAME                       VALUE
MMM                             3.000


 LABEL              LOC  BLOCK TYPE     ENTRY COUNT CURRENT COUNT RETRY
                    1    GENERATE           108             0       0
                    2    ASSIGN             108             0       0
MMM                 3    QUEUE              612             7       0
                    4    SEIZE              605             1       0
                    5    DEPART             604             0       0
                    6    PRIORITY           604             0       0
                    7    ADVANCE            604             0       0
                    8    RELEASE            604             0       0
                    9    LOOP               604             0       0
                   10    TERMINATE          100             0       0


FACILITY         ENTRIES  UTIL.   AVE. TIME AVAIL. OWNER PEND INTER RETRY DELAY
 1                 605    0.935      16.779  1      101    0    0     0      7

QUEUE              MAX CONT. ENTRY ENTRY(0) AVE.CONT. AVE.TIME   AVE.(-0) RETRY
 1                  12    8    612     91     3.940     69.924     82.137   0

CEC XN   PRI          M1      ASSEM  CURRENT  NEXT  PARAMETER    VALUE
   101    0       10038.907    101      4      5       5          9.000

FEC XN   PRI          BDT      ASSEM  CURRENT  NEXT  PARAMETER    VALUE
   109    0       10907.521    109      0      1
```

28. Списки пользователя.

В GPSS имеется дополнительный тип списков заявок, названных списками пользователя, которые дают возможность удалять заявки из списков текущих событий и переводить их во временно неактивное состояние. Впоследствии эти заявки могут быть возвращены в список текущих событий.

Блок LINK имеет следующий формат.

```
LINK  A,B,C
```

Блок LINK удаляет заявки из списка текущих событий и помещает их в список пользователя.

Поле A задает номер или имя списка пользователя, в который, возможно, будет помещена вошедшая заявка.

Поле B задает алгоритм упорядочивания списка пользователя.

Операнд B может быть:

FIFO – вошедшая заявка помещается в конец списка пользователя;

LIFO - вошедшая заявка помещается в начало списка пользователя;

номером P – параметра, тогда входящие заявки располагаются в списке пользователя по возрастанию значения указанного параметра.

Здесь также можно указать PR или M1. Тогда заявки располагаются по возрастанию соответствующего параметра.

Поле C, указывает метку альтернативного выхода.

С каждым списком пользователя связан индикатор. Первоначально он установлен в 0. Если заявка пытается войти в список пользователя, то индикатор устанавливается в 1. Когда блок UNLINK пытается удалить заявку из пустого списка, то индикатор вновь устанавливается в 0. Использование индикатора упрощает работу со списками пользователя.

Если поле C пусто, индикатор, устанавливается в единицу. Это приводит к тому, что все входящие заявки, безусловно, заносятся в список пользователя.

Если поле C не пустое, проверяется индикатор списка пользователя. Если индикатор списка установлен в единицу, вошедшая заявка заносится в список пользователя. Если индикатор списка установлен в "0", он устанавливается в единицу, и вошедшая заявка переходит к блоку, заданному в поле C.

Пример: LINK TMP, FIFO

В этом примере вошедшая заявка помещается в конец списка пользователя с именем TMP.

Со списками пользователя связаны следующие СЧА.

CAномер или CA$имя - среднее число заявок в списке пользователя.

CC номер или CC$имя - общее число заявок, вошедших в список пользователя.

CH номер или CH$имя - текущее число заявок в списке пользователя.

CM номер или CM$имя - максимальное число заявок в списке пользователя.

CT номер или CT$имя - среднее время пребывания заявок в списке пользователя.

Блок UNLINK имеет следующий формат записи:

UNLINK X A,B,C,D,E,F

Блок UNLINK удаляет заявки из списка пользователя. Условия, записываемые во вспомогательном поле X , управляют отбором заявок, извлекаемых из списка пользователя. Сама заявка, вошедшая в блок, как правило, идет дальше по модели.

Если условие не указано, то предполагается отношение равенства (E).

Условия могут быть:

G - больше. Проверяется отношение между значением P - параметра, заданного в поле D, и значением в поле E. Сравнение ведется в числовой форме.

GE –аналогично проверяет отношение больше или равно.

L - меньше.

LE - меньше или равно.

E - равно.

NE - не равно.

A - номер или имя списка пользователя.

B - номер или метка блока, к которому будут отправлены заявки, удаленные из списка пользователя.

C – максимальное число удаляемых из списка заявок. По умолчанию – все. В этом случае можно писать ALL. Иначе нужно указать сколько именно.

D – проверяемое значение. Здесь может быть номер или имя P – параметра, ссылка на логическое выражение, или слово BACK. В последнем случае заявки удаляются из конца списка. Поле D касается заявок из списка пользователя.

E – второе проверяемое значение, оно сравнивается с D. Поле E касается активной заявки. Операнд E не используется, если D – это имя логического выражения.

F – номер или метка блока, куда уходит заявка, если список пользователя окажется пустым до окончания процесса удаления, или не будет найдено ни одной подходящей заявки для удаления.

Пример: из списка wait удалить одну заявку, и отправить ее по метке lab.

```
UNLINK    wait,lab,1
```

Если D – ссылка на логическое выражение, то, заявки будут удаляться, если для них выражение будет истинно. P – параметры, если они есть в выражении, относятся к заявкам из списка пользователя.

Индикатор списка устанавливается в 0, если блок UNLINK должен извлечь заявку, но список пользователя пуст.

Пример применения блоков LINK и UNLINK

Рассмотрим модель работы одноканального устройства, с обслуживанием очереди по алгоритму LIFO:

```
 tt table m1,0,5000,30
        GENERATE   (exponential(1,0,1000))
         QUEUE     och1
         LINK      list,LIFO,next
    next SEIZE     FAC1
         DEPART    och1
        tabulate tt
         ADVANCE   (exponential(1,0,800))
         RELEASE   FAC1
         UNLINK    list,next,1
        TERMINATE 1
         START 1000
```

Выходная статистика этой модели - следующая.

```
    GPSS World Simulation Report - Ethernet.9.1

         Wednesday, October 01, 2003 18:26:28

 START TIME           END TIME  BLOCKS  FACILITIES  STORAGES
      0.000         969031.802    10        1          0

    NAME                      VALUE
FAC1                        10003.000
LIST                        10002.000
NEXT                            4.000
OCH1                        10001.000
```

```
          TT                          10000.000


 LABEL              LOC  BLOCK TYPE     ENTRY COUNT CURRENT COUNT RETRY
                     1    GENERATE          1003             0       0
                     2    QUEUE             1003             0       0
                     3    LINK              1003             3       0
NEXT                 4    SEIZE             1000             0       0
                     5    DEPART            1000             0       0
                     6    TABULATE          1000             0       0
                     7    ADVANCE           1000             0       0
                     8    RELEASE           1000             0       0
                     9    UNLINK            1000             0       0
                    10    TERMINATE         1000             0       0


FACILITY         ENTRIES  UTIL.   AVE. TIME AVAIL. OWNER PEND INTER RETRY DELAY
 FAC1              1000    0.865     837.784  1        0    0    0     0      0


QUEUE              MAX CONT. ENTRY ENTRY(0) AVE.CONT. AVE.TIME   AVE.(-0) RETRY
 OCH1               16    3   1003    168     2.948   2848.346   3421.427   0


TABLE              MEAN    STD.DEV.       RANGE           RETRY FREQUENCY CUM.%
 TT             2853.659 7588.221                          0
                                          -        0.000          168    16.80
                                  0.000   -     5000.000          706    87.40
                               5000.000   -    10000.000           62    93.60
                              10000.000   -    15000.000           20    95.60
                              15000.000   -    20000.000           13    96.90
                              20000.000   -    25000.000            9    97.80
                              25000.000   -    30000.000            4    98.20
                              30000.000   -    35000.000            2    98.40
                              35000.000   -    40000.000            6    99.00
                              40000.000   -    45000.000            1    99.10
                              45000.000   -    50000.000            1    99.20
                              50000.000   -    55000.000            2    99.40
                              55000.000   -    60000.000            1    99.50
                              60000.000   -    65000.000            3    99.80
                              65000.000   -    70000.000            1    99.90
                              70000.000   -    75000.000            0    99.90
                              75000.000   -    80000.000            0    99.90
                              80000.000   -    85000.000            1   100.00


USER CHAIN         SIZE RETRY  AVE.CONT   ENTRIES  MAX     AVE.TIME
 LIST                 2    0      2.948      835    16     3421.427


CEC XN   PRI          M1      ASSEM  CURRENT  NEXT  PARAMETER    VALUE
  1003    0      968437.777   1003      3      4


FEC XN   PRI         BDT      ASSEM  CURRENT  NEXT  PARAMETER    VALUE
  1004    0      969846.935   1004      0      1
```

Анализируя выходную статистику, нетрудно видеть, что время ожидания в очереди некоторых заявок намного превосходит, ожидаемое время исходя из максимальной длины очереди. Это в свою очередь свидетельствует, что заявки действительно обслуживаются по дисциплине LIFO.

Хотя эта модель и не сложна, на ней можно проиллюстрировать несколько следующих важных моментов:

1) в этой системе активными, т.е. находящимися в списках текущих и будущих событий, списках прерываний или списках задержки, могут быть

только те заявки, которые выходят из блока GENERATE, и та заявка, которая в данный момент занимает устройство. Все остальные заявки находятся в списке пользователя LIST.

2) поскольку все задержанные заявки, т.е. заявки, находящиеся в очереди к устройству FAC1, будут помещены в список пользователя LIST, интерпретатор не будет тратить время на обработку этих заявок. Величина экономии времени зависит от длины очереди. Чем длиннее очередь, тем больше времени будет сэкономлено при помощи блоков LINK\UNLINK, применяемых для управления очередями к различным объектам.

3) пользователь имеет возможность формировать свои списки в динамике вне зависимости от списков задержки, которые управляются системой GPSS.

Списки пользователя моделируют специфические очереди, не подчиняющиеся принципу FIFO. Использование списков пользователя ускорит процесс моделирования при наличии очередей в системе.

Пример. Модель системы оперативной обработки данных в системе с разделением времени и кольцевым алгоритмом обслуживания и с учетом приоритетов. Объем памяти для заявки, ее приоритет и длительность обслуживания являются случайными величинами, рассчитываемыми по соответствующим формулам.

```
MM1 STORAGE 1024
   INITIAL X$kvant,50
TT1    TABLE MP$time,0,100,20
   GENERATE (exponential(1,0,150))
   ASSIGN num_st,(duniform(1,200,440)) ; память
   ASSIGN prio,(duniform(1,0,4)) ; приоритет
   ASSIGN num_ret,(duniform(1,1,5)) ; циклы
 M8   TEST LE  P$num_st,R$MM1,M4  ; R$MM1-свободная память
    ENTER mm1,P$num_st
  MARK time  ; в p$time- начало обслуживания
M2  LINK 1,P$prio,M3
M3  SEIZE cpu
   ADVANCE X$kvant
   RELEASE cpu
   UNLINK 1,M3,1
   LOOP num_ret,M2
   LEAVE mm1,P$num_st
   UNLINK 2,M8,1
   TABULATE TT1
   TERMINATE 1
M4 LINK 2,FIFO
            START 1000
```

Выходная статистика модели выглядит следующим образом.

```
GPSS World Simulation Report - Untitled Model 1.11.1
     Friday, October 03, 2003 09:15:38
```

START TIME	END TIME	BLOCKS	FACILITIES	STORAGES
0.000	152260.723	18	1	1

NAME	VALUE
CPU	10007.000
KVANT	10001.000
M2	8.000
M3	9.000
M4	18.000
M8	5.000
MM1	10000.000
NUM_RET	10005.000
NUM_ST	10003.000
PRIO	10004.000
TIME	10006.000
TT1	10002.000

LABEL	LOC	BLOCK TYPE	ENTRY COUNT	CURRENT COUNT	RETRY
	1	GENERATE	1003	0	0
	2	ASSIGN	1003	0	0
	3	ASSIGN	1003	0	0
	4	ASSIGN	1003	0	0
M8	5	TEST	1853	0	0
	6	ENTER	1001	0	0
	7	MARK	1001	0	0
M2	8	LINK	2905	1	0
M3	9	SEIZE	2904	0	0
	10	ADVANCE	2904	0	0
	11	RELEASE	2904	0	0
	12	UNLINK	2904	0	0
	13	LOOP	2904	0	0
	14	LEAVE	1000	0	0
	15	UNLINK	1000	0	0
	16	TABULATE	1000	0	0
	17	TERMINATE	1000	0	0
M4	18	LINK	852	2	0

FACILITY	ENTRIES	UTIL.	AVE. TIME	AVAIL.	OWNER	PEND	INTER	RETRY	DELAY
CPU	2904	0.954	50.000	1	0	0	0	0	0

STORAGE	CAP.	REM.	MIN.	MAX.	ENTRIES	AVL.	AVE.C.	UTIL.	RETRY	DELAY
MM1	1024	636	0	1024	319665	1	757.303	0.740	0	0

TABLE	MEAN	STD.DEV.	RANGE			RETRY	FREQUENCY	CUM.%
TT1	368.163	314.658				0		
			0.000	-	100.000		180	18.00
			100.000	-	200.000		201	38.10
			200.000	-	300.000		173	55.40
			300.000	-	400.000		157	71.10
			400.000	-	500.000		143	85.40
			500.000	-	600.000		35	88.90
			600.000	-	700.000		22	91.10
			700.000	-	800.000		24	93.50
			800.000	-	900.000		16	95.10
			900.000	-	1000.000		12	96.30
			1000.000	-	1100.000		4	96.70
			1100.000	-	1200.000		6	97.30
			1200.000	-	1300.000		5	97.80
			1300.000	-	1400.000		4	98.20
			1400.000	-	1500.000		3	98.50
			1500.000	-	1600.000		2	98.70
			1600.000	-	1700.000		3	99.00
			1700.000	-	1800.000		1	99.10
			1800.000	-	_		9	100.00

```
USER CHAIN          SIZE RETRY  AVE.CONT   ENTRIES  MAX     AVE.TIME
 1                     0    0     1.465      2733    3       81.637
 2                     1    0     9.935       852   28     1775.555


SAVEVALUE               RETRY       VALUE
 KVANT                    0         50.000


CEC XN   PRI          M1      ASSEM  CURRENT  NEXT  PARAMETER    VALUE
  1001    0      151958.560   1001      8       9
                                                   PRIO          0.000
                                                   NUM_RET       1.000
                                                   TIME      152110.723
                                                   NUM_ST      388.000
  1002    0      152022.438   1002     18       5
                                                   NUM_ST      422.000
                                                   NUM_RET       4.000
                                                   PRIO          2.000


FEC XN   PRI          BDT      ASSEM  CURRENT  NEXT  PARAMETER    VALUE
  1004    0      153122.483   1004      0       1
```

29. Блоки выборки требуемых объектов в GPSS.

Блок COUNT определяет количество объектов, удовлетворяющих заданному условию, и имеет следующий формат записи:

COUNT X A,B,C,D,E

Здесь X - условие отбора.

Если нужно подсчитать количество определенных устройств, то возможны следующие условия.

NU - устройство свободно (доступно);

U - устройство занято (в результате выполнения заявкой блока SEIZE или PREEMPT);

NI - устройство не прервано (т.е. либо оно свободно, либо занято заявкой, выполнившей блок SEIZE);

I - устройство прервано (устройство занято заявкой, выполнившей блок PREEMPT);

FV - устройство доступно;

FNV - устройство недоступно.

Если нужно подсчитать количество определенных многоканальных устройств, то возможны следующие условия.

SE - многоканальное устройство пусто (нулевое содержимое);

SNE - многоканальное устройство не пусто (ненулевое содержимое);

SF - многоканальное устройство заполнено;

SNF - многоканальное устройство не заполнено;

SV - многоканальное устройство доступно;

SNV - многоканальное устройство недоступно.

Если нужно подсчитать количество логические ключей, находящихся в определенном состоянии, то возможны следующие условия.

LR - ключ в состоянии "выключено";

LS - ключ в состоянии "включено";

При подсчете числа объектов, для которых выполняется отношение, используют следующие условия.

L - меньше. Условие выполняется, если значение стандартного числового атрибута, заданного в поле E, меньше значения стандартного числового атрибута, заданного в поле D;

LE – аналогично, меньше или равно.

E - аналогично, равно.

NE - аналогично, не равно.

G - аналогично, больше.

GE - аналогично, больше или равно.

MAX – определяет подсчет числа значений СЧА, которые имеют наибольшее значение.

MIN – определяет подсчет числа значений СЧА, которые имеют наименьшее значение.

Если используются отношения, то поля D и E блока COUNT должны быть заданы.

В поле A задается номер или имя P – параметра для вошедшей в блок заявки. В нем будет организован счетчик числа объектов.

В поле B определяется нижняя граница диапазона изменения номеров объектов, для которых проверяется заданное условие.

В поле C определяется верхняя граница диапазона изменения номеров объектов, для которых проверяется заданное условие.

Поле D задает сравниваемое значение (величина сравнения) для аргумента поля E. Это значение используется совместно с заданными условными операторами (E, NE, G, GE, L, LE).

Значение СЧА сравнивается со значением объекта, заданного аргументом поля E. Если отношение не используется, поле D можно не задавать.

Поле E используется совместно с полем D и условным оператором. В поле E задается какой-либо из стандартных числовых атрибутов просматриваемых объектов. Необходимо только записывать мнемоническое обозначение атрибута, то есть его начальную часть, без уточнения, поскольку диапазон изменения номеров объектов задан полями B и C.

Рассмотрим несколько примеров использования блока COUNT.

```
COUNT LE 1,2,6,X$CNT,FC
```

В этом примере подсчитывается число устройств (из устройств 2-6 включительно), у которых счетчик числа входов (FC) меньше или равен текущему значению ячейки CNT. результат подсчета будет записан в параметре P1 вошедшей в блок COUNT заявки.

В следующем примере подсчитывается число заполненных многоканальных устройств (SF) при изменении их номеров в интервале 10-20 (включительно). Результат подсчета записывается в параметре P5 вошедшей в блок заявки.

```
COUNT SF 5,10,20
```

Примечание. Просматриваемые объекты должны задаваться номерами, то есть, например, для просмотра многоканальных устройств, придется задавать эквивалентности между именами и номерами устройств.

Блок SELECT. Этот блок во многом аналогичен COUNT, и имеет следующий формат.

SELECT X A,B,C,D,E,F

Блок SELECT выбирает первый объект в заданном диапазоне, который удовлетворяет условию. Номер этого объекта записывается в заданный в поле A, P -параметр вошедшей в блок заявки. Назначения параметров X, B, C, D, E совершенно такие же, как у блока COUNT.

Поле F задает номер или метку альтернативного блока для входящей заявки, если заданным условиям не удовлетворяет ни один объект в диапазоне.

Рассмотрим примеры использования блока SELECT.

SELECT MAX NUM,2,6,,FR

В данном примере из устройств с номерами 5- 10 (включительно) выбирается устройство с максимальным коэффициентом использования (FR). Номер выбранного устройства записывается в параметре P$NUM вошедшей заявки.

SELECT LE NUM1, 11, 15, X$GRAN, Q

В этом примере из очередей с номерами 11-15 (включительно) выбирается первая очередь, длина которой (Q) меньше или равна значению ячейки (X$GRAN). Номер очереди, удовлетворяющей этому условию, записывается в параметре P$NUM1 вошедшей в блок заявки.

Поиск по SELECT выполняется до нахождения первого элемента оборудования, удовлетворяющего условию.

Пример. Пусть имеется 5 устройств. Нужно отправлять заявку в первое попавшееся устройство, у которого длина очереди меньше 2. Заявки приходят с интервалом 10+-8,а обслуживание занимает 49+-40 единиц времени.

Промоделировать обслуживание 1000 заявок. Если требуемого устройства нет, то заявки должны удаляться из системы.

```
 GENERATE 10,8
 SELECT L 1,1,5,2,Q
TEST NE P1,0,MM
 QUEUE P1
 SEIZE P1
 DEPART P1
 ADVANCE 49,40
 RELEASE P1
 TERMINATE 1
MM TERMINATE
 START 1000
```

Пример выбора минимальной и максимальной очереди.

Задача совпадает с предыдущей задачей, но заявка должна обслуживаться тем устройством, очередь к которому минимальна.

Модель отличается только блоком SELECT:

```
SELECT MIN 1,1,5,,Q
```

Рассмотрим задачу, когда блок SELECT не используется. Пусть номер устройства записывается в параметр p1,а параметр p2 служит для организации цикла просмотра устройств.

```
  GENERATE 10,8
  ASSIGN 1,5
  ASSIGN 2,4
MM1 TEST G  Q*1,Q*2,MM2
  ASSIGN 1,P2
MM2 LOOP 2,MM1
  QUEUE P1
  SEIZE P1
  DEPART P1
  ADVANCE 49,40
  RELEASE P1
  TERMINATE 1
  START 1000
```

Очевидно, что такая модель оказалось более громоздка и менее наглядна.

30. Выбор генератора случайных значений для моделирования.

Для сбора полноценной статистики, иногда полезно изменить генераторы случайных чисел, используемые в ходе моделирования. Это можно сделать с помощью вкладки Random Numbers, доступ к которой можно получить через пункты меню Edit, Settings.

31. Блоки задания доступности и недоступности устройств.

Блок FUNAVAIL предназначен для перевода одноканального устройства в недоступное для работы состояние. Он предусматривает возможность выполнения многих дополнительных операций и имеет вид:

```
FUNAVAIL   A,B,C,D,E,F,G,H
```

Здесь A – это имя или номер устройства.

B – флаг удаления (Remove) или продолжения (Continue) для обслуживаемой заявки. Он должен быть пустым или RE, CO. Флаг Co подразумевается. В этом случае заявка автоматически продолжит обслуживание, когда устройство станет свободным. Аналогичная реакция справедлива и для флагов E и G. Однако, если используется метка в поле C, то заявка уйдет по этой метке, но будет по-прежнему занимать устройство.

C – номер или метка блока. Определяет блок, куда должна отправиться обслуживаемая заявка.

D – номер P- параметра, для обслуживаемой заявки. В нем будет сохранено значение времени, оставшегося до конца задержки этой заявки в блоке ADVANCE, если она в нем находится.

E – флаг удаления (Remove) или продолжения (Continue) для прерванных заявок в устройстве. Он должен быть пустым или RE, CO.

F - номер или метка блока. Определяет блок, куда должны отправиться прерванные заявки. Прерванные заявки, продолжают занимать устройство, если флаг имеет значение CO.

G – флаг удаления (Remove) или продолжения (Continue) для заявок, претендующих на обслуживание в устройстве. Он должен быть пустым или RE, CO.

H - номер или метка блока. Определяет блок, куда должны отправиться заявки, претендующие на обслуживание в устройстве.

Пример

FUNAVAIL ust1

Устройство становится недоступным для заявок. Обслуживаемая заявка находится как бы в состоянии прерывания до конца периода недоступности. Остальные заявки, связанные с устройством, также не смогут его занять до конца периода недоступности.

Пример

FUNAVAIL ust1,RE,mm1,100,RE,mm2,CO

В этом примере, заявка в устройстве должна уйти по метке mm1. Если она находится в блоке ADVANCE, то в параметр P100 будет записано ее оставшееся время обслуживания в устройстве. Прерванные в устройстве заявки, если они есть, будут отправлены по метке mm2. А заявки, которые только претендуют на обслуживание, будут по-прежнему на него претендовать.

Блок FAVAIL предназначен для возврата одноканального устройства в доступное для работы состояние. Он имеет вид:

FAVAIL A

Здесь A – это имя или номер устройства.

Блок SUNAVAIL переводит многоканальное устройство в недоступное состояние. Он имеет вид.

SUNAVAIL A

Здесь A – это имя или номер многоканального устройства.

После выполнения этого блока, в многоканальное устройство запрещен вход новых заявок, пока оно вновь не станет доступным.

Блок SAVAIL переводит многоканальное устройство в доступное состояние. Он имеет вид.

SAVAIL A

Здесь A – это имя или номер многоканального устройства.

Пример использования блоков.

Пусть в систему поступают заявки со средним интервалом 100 единиц, распределенным по экспоненциальному закону. Время обслуживания заявок составляет 48+-20 единиц. Устройство имеет интервалы доступности и недоступности, распределенные по нормальному закону со средним значением

2000, и отклонением в 200 единиц. Промоделировать работу системы за 20 циклов доступности и недоступности. Текст такой модели может иметь вид:

```
generate (exponential(1,0,100))
queue ust
seize ust
depart ust
advance 48,20
release ust
terminate
generate 0,0,0,1
ret favail ust
advance (normal(1,2000,200))
funavail ust
advance (normal(1,2000,200))
split 1,ret
terminate 1
start 20
```

Выходная статистика модели подтверждает корректность расчетов среднего времени обслуживания и занятости устройства.

```
            GPSS World Simulation Report - Untitled Model 1.2.1

                 Monday, October 06, 2003 21:14:38

          START TIME           END TIME  BLOCKS  FACILITIES  STORAGES
               0.000          81356.413    14        1          0


             NAME                      VALUE
          RET                           9.000
          UST                       10000.000


 LABEL              LOC  BLOCK TYPE     ENTRY COUNT CURRENT COUNT RETRY
                    1    GENERATE           747             0       0
                    2    QUEUE              747            24       0
                    3    SEIZE              723             0       0
                    4    DEPART             723             0       0
                    5    ADVANCE            723             1       0
                    6    RELEASE            722             0       0
                    7    TERMINATE          722             0       0
                    8    GENERATE             1             0       0
RET                 9    FAVAIL              20             0       0
                   10    ADVANCE             20             0       0
                   11    FUNAVAIL            20             0       0
                   12    ADVANCE             20             0       0
                   13    SPLIT               20             0       0
                   14    TERMINATE           20             0       0


FACILITY          ENTRIES  UTIL.   AVE. TIME AVAIL. OWNER PEND INTER RETRY DELAY
 UST                 723    0.433      48.767  0        0    0    1     0     24


QUEUE              MAX CONT. ENTRY ENTRY(0) AVE.CONT. AVE.TIME   AVE.(-0) RETRY
 UST                26   24    747     43     9.489   1033.404   1096.524   0


CEC XN   PRI          M1      ASSEM  CURRENT  NEXT  PARAMETER    VALUE
   769    0           0.000       2      0      9
```

```
FEC XN    PRI         BDT      ASSEM  CURRENT  NEXT  PARAMETER    VALUE
   768    0       81687.968    768      0      1
```

Аналогичная задача для трехканального устройства при среднем интервале поступления заявок равном 33, решается следующим образом.

```
ust storage 3
generate (exponential(1,0,33.33))
queue ust
enter ust
depart ust
mark
advance 48,20
leave ust
lab savevalue time+,m1
terminate
generate 0,0,0,1
ret savail ust
advance (normal(1,2000,200))
sunavail ust
advance (normal(1,2000,200))
split 1,ret
savevalue rez,(X$time/n$lab)
savevalue rez1,st$ust
terminate 1
start 20
```

Здесь rez и rez1- средние времена обслуживания заявки в устройстве. Если обратиться к итоговой статистике, то можно убедиться, что величина ST$UST вычисляется в модели правильно, то есть в модели корректно учитывается период недоступности.

```
                    GPSS World Simulation Report - Untitled Model 1.6.1

                  Monday, October 06, 2003 21:29:51

           START TIME           END TIME  BLOCKS  FACILITIES  STORAGES
                0.000          81724.219    18        0          1


              NAME                       VALUE
          LAB                             8.000
          RET                            11.000
          REZ                         10002.000
          REZ1                        10003.000
          TIME                        10001.000
          UST                         10000.000

 LABEL              LOC  BLOCK TYPE     ENTRY COUNT CURRENT COUNT RETRY
                    1    GENERATE          2448             0       0
                    2    QUEUE             2448            85       0
                    3    ENTER             2363             0       0
                    4    DEPART            2363             0       0
                    5    MARK              2363             0       0
                    6    ADVANCE           2363             0       0
                    7    LEAVE             2363             0       0
```

```
LAB                  8    SAVEVALUE          2363             0       0
                     9    TERMINATE          2363             0       0
                    10    GENERATE              1             0       0
RET                 11    SAVAIL               20             0       0
                    12    ADVANCE              20             0       0
                    13    SUNAVAIL             20             0       0
                    14    ADVANCE              20             0       0
                    15    SPLIT                20             0       0
                    16    SAVEVALUE            20             0       0
                    17    SAVEVALUE            20             0       0
                    18    TERMINATE            20             0       0


QUEUE               MAX CONT. ENTRY ENTRY(0) AVE.CONT. AVE.TIME   AVE.(-0) RETRY
 UST                 85   85   2448    157    31.307   1045.144   1116.767   0


STORAGE             CAP. REM. MIN. MAX.  ENTRIES AVL.  AVE.C. UTIL. RETRY DELAY
 UST                  3    3   0     3     2363   0    1.384  0.461    0   85


SAVEVALUE               RETRY       VALUE
 TIME                     0     113125.882
 REZ                      0         47.874
 REZ1                     0         47.874


CEC XN   PRI          M1      ASSEM  CURRENT  NEXT  PARAMETER    VALUE
  2470    0          0.000       2      0     11


FEC XN   PRI         BDT      ASSEM  CURRENT  NEXT  PARAMETER    VALUE
  2469    0       81748.653   2469      0      1
```

32 Блоки работы с группами.

Группы позволяют иметь числовые множества, и множества заявок. Используя группы, можно получать доступ к ее членам вне зависимости от того, в каких списках находится та или иная заявка. Для групп, можно выполнять проверку на принадлежность заявок или чисел к группе. В группу можно добавлять числа или заявки. Из нее можно удалять числа или заявки. Можно также менять значения P – параметров заявок группы или их приоритет. К блокам, обеспечивающим работу с группами, относятся следующие блоки.

Блок JOIN добавляет активную заявку в группу заявок, или добавляет числовое значение в числовую группу. Блок имеет вид:

JOIN A,B

Здесь A – это номер или имя группы.

B – числовое значение, которое должно быть добавлено к числовой группе.

Если второй параметр отсутствует, то формируется группа заявок. Иначе формируется числовая группа. Числовая группа может иметь как целые, так и вещественные значения. Однако лучше использовать целые числа. Если число или заявка уже имеются в группе, то оператор ничего не делает.

Числовые группы отличаются от групп заявок, даже если они имеют один и тот же номер. Членство в группе никак не ограничивает возможности заявки, и она продолжает свое движение по модели через блок JOIN. Для удаления заявок из группы, необходимо, чтобы они попали в блок TERMINATE или

ASSEMBLE, или были выбраны для удаления в блоке REMOVE. Кроме блока JOIN, для групп заявок можно использовать блоки ALTER, EXAMINE, REMOVE и SCAN. Для числовых групп можно использовать блоки EXAMINE и REMOVE.

С группами связаны только два СЧА.

GN$имя или GNномер текущее число членов числовой группы.

GT$имя или GTномер текущее число членов группы заявок.

Блок REMOVE удаляет заявки из группы заявок, или числа из числовой группы. Блок имеет вид:

REMOVE условие A,B,C,D,E,F

Здесь условие может иметь следующие значения: E, NE,G, GE, L, LE, MAX, MIN. Оно может быть также опущено.

A – номер или имя группы.

B – предельное число удаляемых элементов группы. Здесь может указываться ALL, тогда удаляются все элементы, иначе здесь должно быть целое значение. Параметр может отсутствовать, тогда подразумевается ALL.

C – числовое значение, которое должно быть удалено из числовой группы.

D – значение для проверки. Оно должно быть PR или номером P – параметра. Значение относится к членам группы.

E – сравниваемое значение. Именно с ним сравнивается значение из поля D. Оно относится к текущей заявке.

F – метка или номер блока, куда должна уйти активная заявка, если из группы не удалось удалить нужного числа заявок.

Пример

REMOVE List

Здесь подразумевается удаление из группы List активной заявки, если она там есть.

REMOVE G 5,11,,200,22.3,OUT

В этом случае из 5 группы заявок, удаляются до 11 заявок, у которых параметр P200 имеет значение большее, чем 22.3. Если в группе не найдется 11 заявок, удовлетворяющих условию, то активная заявка пытается уйти по метке OUT. Иначе заявка проходит через блок, и идет дальше по модели.

В числовом режиме блок REMOVE требует операнда C . Числовое значение из поля C удаляется из группы, если оно в нем есть. В этом режиме могут использоваться только операнды A, C, и F. Условия в этом режиме также не используются. В режиме заявок, в блоке REMOVE, напротив, параметр C должен отсутствовать. Если не указано условие, то подразумевается E. Если отсутствуют операнды B, D, или E, то удаляется только активная заявка. Если указано условие MIN или MAX, то удаляются из группы заявки с наименьшими или наибольшими значениями P – параметра или приоритета.

Блок EXAMINE – используется для проверок членов числовой группы, или группы заявок. Он имеет вид:

EXAMINE A,B,C

Здесь A – номер или имя группы.

B - используется только в числовом режиме. Это значение, которое проверяется на наличие в числовой группе.

C – номер или метка блока. Сюда уходит активная заявка, если число отсутствует в группе, или активная заявка не является членом группы. Иначе заявка проходит сквозь блок и идет дальше по модели.

Блок SCAN передает информацию из группы заявок – активной заявке. Этот блок имеет вид:

SCAN условие A,B,C,D,E,F

Здесь условие может быть таким же, как и в блоке REMOVE .

A - номер или имя группы.

B – значение для проверки. Оно должно быть PR или именем, или номером P – параметра. Значение относится к заявкам группы.

C – сравниваемое значение. Именно с ним сравнивается значение из поля B. Оно относится к текущей заявке, по умолчанию равно 0.

D – извлекаемое значение. Должно быть PR или именем, или номером P – параметра. Значение относится к заявкам группы. Это значение будет присвоено параметру активной заявки.

E – имя или номер принимающего P- параметра активной заявки. Получает значение параметра, указанного как D.

F – имя или номер альтернативного блока. Туда уйдет активная заявка, если не будет найдена нужная заявка в группе. Для первой попавшейся заявки из группы, которая удовлетворяет условию, происходит присваивание, и блок завершает свою работу.

Пример

SCAN MAX List,Price,,Number,10

В этом примере в группе List отыскивается заявка с максимальным значением параметра P$Price. Первая попавшаяся заявка с максимальным значением P$Price, используется для определения извлекаемого значения, которое равно значению P$Number. Это значение присваивается параметру P10 активной заявки.

Пример

SCAN E List1,Num,127,Price,Price,lab1

В этом примере в группе List1 отыскивается первая попавшаяся заявка со значением P$Num равным 127. Значение параметра P$ этой заявки заносится в такой же параметр активной заявки. Если нужной заявки в группе не находится, то активная заявка уходит в блок с меткой Lab1.

Блок ALTER предназначен для изменения значения P – параметра заявок группы, или их приоритета, для тех из них, которые удовлетворяет условию. Блок имеет следующий вид:

ALTER условие A,B,C,D,E,F,G

Здесь условие может быть таким же, как в блоках SCAN и REMOVE.

A – имя или номер группы.

B – предельное число заявок группы, параметры которых будут изменяться. Здесь может указываться ALL, тогда изменятся все элементы, иначе

здесь должно быть целое значение. Параметр может отсутствовать, тогда подразумевается ALL.

C –изменяемый атрибут. Он должен быть PR или номером или именем P – параметра. Значение относится к членам группы.

D – заменяемое значение. Оно присваивается атрибуту, определяемому полем C.

E – проверяемое значение. Задается как PR или имя или номер P – параметра. Оно относится к заявкам из группы. Именно с ним сравнивается значение из поля F.

F – сравниваемое значение. Именно с ним сравнивается значение из поля E. Оно относится к текущей заявке.

G – номер или метка альтернативного блока. Туда уйдет активная заявка, если не будут найдено нужное число подходящих заявок в группе.

Пример

ALTER List,ALL,Price,40.65

В этом примере все заявки группы List, получат значение 40.65 в P параметр с именем Price.

Пример

ALTER NE List1,10,Price,50,Num,100,Out

Здесь в группе с именем List1, 10 заявок, у которых параметр P$Num не равен 100, получат значение 50 в параметр P$Price. Если 10 таких заявок в группе нет, то активная заявка уйдет по метке Out. Иначе она пройдет сквозь блок и пойдет дальше по модели.

Пример задачи, использующей указанные блоки.

Пусть имеются 3 магазина, с примерно одинаковым набором товаров. В каждом из них часть товаров может отсутствовать. Покупатель отыскивает первый попавшийся магазин, в котором есть нужный товар. Товар может завозиться в магазин, или заканчиваться в нем со средним интервалом 36000+-18000. Коды товара могут иметь значения от 10 до 121. Поток покупателей распределен по экспоненциальному закону со средним интервалом 3 с. Определить число покупателей в каждом магазине за 100000 с. Поиск товара в каждом магазине занимает 300+-200 с.

Текст такой модели может выглядеть следующим образом:

```
; Начальная номенклатура товаров.
generate 0,0,0,100
join 1,(duniform(1,10,121))
join 2,(duniform(1,10,121))
join 3,(duniform(1,10,121))
terminate
generate 36000,18000
join 1,(duniform(1,10,121))
terminate
generate 36000,18000
join 2,(duniform(1,10,121))
terminate
generate 36000,18000
```

```
join 3,(duniform(1,10,121))
terminate
generate 36000,18000
remove 1,1(duniform(1,10,121))
terminate
generate 36000,18000
remove 2,1,(duniform(1,10,121))
terminate
generate 36000,18000
remove 3,1,(duniform(1,10,121))
terminate
generate (exponential(1,0,3))
assign tov,(duniform(1,10,121))
advance 300,200
examine 1,p$tov,next
savevalue mag1+,1
terminate
next  advance 0
advance 300,200
examine 2,p$tov,next1
savevalue mag2+,1
terminate
next1 advance 0
advance 300,200
examine 3,p$tov,next2
savevalue mag3+,1
terminate
next2 advance 0
advance 0
savevalue not_bye+,1
terminate
generate 100000
terminate 1
start 1
```

Выходная статистика модели имеет вид:

```
    GPSS World Simulation Report - Untitled Model 1.10.1

         Wednesday, October 08, 2003 11:16:21

 START TIME           END TIME  BLOCKS  FACILITIES  STORAGES
      0.000         100000.000    45        0          0

    NAME                      VALUE
MAG1                        10001.000
MAG2                        10002.000
MAG3                        10003.000
NEXT                           30.000
NEXT1                          35.000
NEXT2                          40.000
NOT_BYE                     10004.000
TOV                         10000.000
```

LABEL	LOC	BLOCK TYPE	ENTRY COUNT	CURRENT COUNT	RETRY
	1	GENERATE	100	0	0
	2	JOIN	100	0	0
	3	JOIN	100	0	0
	4	JOIN	100	0	0
	5	TERMINATE	100	0	0
	6	GENERATE	2	0	0
	7	JOIN	2	0	0
	8	TERMINATE	2	0	0
	9	GENERATE	2	0	0
	10	JOIN	2	0	0
	11	TERMINATE	2	0	0
	12	GENERATE	2	0	0
	13	JOIN	2	0	0
	14	TERMINATE	2	0	0
	15	GENERATE	3	0	0
	16	REMOVE	3	0	0
	17	TERMINATE	3	0	0
	18	GENERATE	2	0	0
	19	REMOVE	2	0	0
	20	TERMINATE	2	0	0
	21	GENERATE	3	0	0
	22	REMOVE	3	0	0
	23	TERMINATE	3	0	0
	24	GENERATE	33383	0	0
	25	ASSIGN	33383	0	0
	26	ADVANCE	33383	95	0
	27	EXAMINE	20051	0	0
	28	SAVEVALUE	20051	0	0
	29	TERMINATE	20051	0	0
NEXT	30	ADVANCE	26474	0	0
	31	ADVANCE	13237	46	0
	32	EXAMINE	8932	0	0
	33	SAVEVALUE	8932	0	0
	34	TERMINATE	8932	0	0
NEXT1	35	ADVANCE	8518	0	0
	36	ADVANCE	4259	16	0
	37	EXAMINE	2882	0	0
	38	SAVEVALUE	2882	0	0
	39	TERMINATE	2882	0	0
NEXT2	40	ADVANCE	2722	0	0
	41	ADVANCE	1361	0	0
	42	SAVEVALUE	1361	0	0
	43	TERMINATE	1361	0	0
	44	GENERATE	1	0	0
	45	TERMINATE	1	0	0

NUMERIC GROUP	GROUP SIZE	RETRY
1	67	0
2	67	0
3	64	0

SAVEVALUE	RETRY	VALUE
MAG1	0	20051.000
MAG2	0	8932.000
MAG3	0	2882.000
NOT_BYE	0	1361.000

FEC XN	PRI	BDT	ASSEM	CURRENT	NEXT	PARAMETER	VALUE
33380	0	100000.832	33380	31	32		
						TOV	87.000
33297	0	100003.427	33297	31	32		
						TOV	120.000
33505	0	100004.185	33505	0	24		
33363	0	100010.765	33363	26	27		
						TOV	117.000
33388	0	100012.827	33388	26	27		

```
                                                    TOV          88.000
33300    0      100012.991   33300    31     32
                                                    TOV          22.000
33191    0      100013.706   33191    31     32
                                                    TOV          48.000
33352    0      100016.428   33352    26     27
                                                    TOV          97.000
33474    0      100019.809   33474    26     27
                                                    TOV          11.000
33370    0      100021.881   33370    26     27
```

и так далее.

В той версии GPSS - World, которой располагает автор, к сожалению, некорректно отслеживается число вошедших заявок, для блока, куда переходит заявка после неудачной проверки в EXEMINE. Правда на остальные результаты моделирования это, к счастью, не влияет.

33. Редко используемые блоки.

Среди этих блоков – DISPLACE. Он предназначен для перемещения указанной заявки в указанный блок. Его т вид:

DISPLACE A,B,C,D

Здесь A – номер заявки.

B – номер или метка блока, куда должна уйти указанная заявка.

C – номер или имя P- параметра перемещаемой заявки. Сюда заносится оставшееся время задержки, если заявку перемещают из блока ADVANCE.

D – номер или метка блока, куда уйдет активная заявка, если нужной заявки, указанной в поле A- нет.

Пример

DISPLACE X$Num,lab1,time,NoTtran

Здесь заявка, номер которой находится в X$Num, уйдет по метке lab1. Если она находится в списке будущих событий, то время, которое она там еще должна пробыть, записывается в параметр P$time. Если же в системе сейчас уже или еще нет этой заявки, то активная заявка уходит в блок с меткой NoTtran. Иначе активная заявка проходит сквозь блок, и идет дальше по модели.

Заявка не извлекается из списков прерываний. Она не освобождает занятые ею устройства.

Для правильной работы с блоком имеются следующие функции типа запросов.

Главная из них -QueryXNExist(A).

Здесь A - это номер заявки. Функция возвращает 1, если заявка есть в модели, иначе – возвращает 0.

Функция QueryXNParameter(A.B).

Здесь A - это номер заявки.

B – это имя или номер P –параметра. Функция возвращает значение P – параметра.

Функция QueryXNAssemblySet(A)

Здесь A - это номер заявки.

Функция возвращает значение номера ансамбля.

Функция QueryXNPriority(A)

Здесь A - это номер заявки.
Функция возвращает значение приоритета.
Функция QueryXNM1(A)
Здесь A - это номер заявки.
Функция возвращает значение времени жизни заявки.

Блок EXECUTE предназначен для выполнения любого блока из модели, с возвратом после этого на следующий после него блок. Он имеет вид:

EXECUTE A

Здесь A – номер или метка блока.

Если блок может направлять заявки по новому пути, то, возможно, заявка и не вернется на следующий блок.

Блок INDEX – в принципе, является устаревшим. Имеет следующий вид:

INDEX A,B

Здесь A – номер или имя P- параметра.
B – числовое значение.
Вычисляется сумма значения P- параметра и числового значения.
Результат записывается в параметр P1.

Блоки TRACE и UNTRACE включают и выключают трассировку движения активной заявки. Могут быть полезными для отладки.

34. Процедурный язык PLUS.

Этот язык состоит из следующих операторов:

- ✓ PROCEDURE - определяет заголовок процедуры, точнее функции.
- ✓ EXPERIMENT - определяет заголовок специальной процедуры-эксперимента.
- ✓ TEMPORARY - определяет временные переменные и матрицы в процедуре (функции).
- ✓ Большинство операторов могут иметь метку, отделяющуюся от оператор пробелами.
- ✓ Ключевые слова BEGIN и END- определяют блок в процедуре, как в языке Pascal, то есть позволяют создавать составные операторы.
- ✓ Оператор присваивания.
- ✓ Вызов процедур, точнее функций.
- ✓ Оператор IF THEN или
- ✓ IF THEN ELSE . Используется для ветвлений в программе.
- ✓ Оператор WHILE DO- позволяет организовывать циклы.
- ✓ Оператор GOTO – позволяет выполнить безусловный переход.
- ✓ Оператор RETURN – завершает процедуру и формирует ее результат.

Оператор присваивания имеет вид:

A=B

Здесь A – глобальная или локальная переменная (метка) в модели, или такой же элемент матрицы. B - выражение.

Например

Home=(X$time+P$num)#3.6+home

Примечание. В процедуре нельзя изменить значения X или P – параметров.

Текст процедуры имеет вид

PROCEDURE имя(список аргументов через запятую) оператор;

Оператор обычно является составным, а поэтому представляет собой последовательность операторов, разделенных точкой с запятой, и обрамленных ключевыми словами BEGIN END; Чисто формально, описание процедуры отличается от описания эксперимента только ключевым словом EXPERIMENT.

Процедуры могут использоваться рекурсивно, подобно процедурам во многих других языках программирования.

Оператор IF имеет вид:

IF (числовое выражение) THEN оператор;

Если выражение не равно 0, то встроенный оператор выполняется, иначе он не выполняется.

Форма оператора.

IF (числовое выражение) THEN оператор1;
ELSE оператор2;

В этом случае выполнение оператора происходит несколько по-другому. Если выражение не равно 0, то выполняется встроенный оператор1, иначе оператор2.

Оператор GOTO имеет вид:

GOTO A;

Здесь A – это метка в процедуре.

Следующим будет выполняться оператор с этой меткой.

Оператор RETURN имеет вид:

RETURN выражение ;

При выполнении оператора RETURN, процедура завершается, а выражение считается ее результатом. Результат может быть числовым или строковым.

Оператор TEMPORARY описывает временные величины, которые существуют только во время выполнения процедуры. Он имеет вид:

TEMPORARY список переменных ;

Переменные в списке разделяются запятыми.

При описании матриц оператор имеет вид:

TEMPORARY MATRIX имя [список целых];

Целые величины определяют размерности матрицы по каждому индексу. Всего индексов может быть до 6. Номера ячеек матрицы начинаются с 1. Количество операторов TEMPORARY в процедуре не регламентируется.

Примеры

```
TEMPORARY Value1,tmp;
TEMPORARY MATRIX DataArray[5,7,15];
```

Оператор WHILE предназначен для организации циклов и имеет вид:

WHILE (выражение) DO оператор;

Оператор здесь, как правило, является составным. Выражение вычисляется как целое, и если оно не равно 0, то выполняется оператор, вставленный в цикл, а если выражение равно 0, то вставленный оператор не выполняется, цикл заканчивается, и выполнение процедуры идет дальше. Как в процедурах, так и в выражениях модели, можно вызывать как встроенные, так и написанные вами процедуры, точнее функции, так как они возвращают значения.

Как можно заметить, этот язык во многом напоминает гибрид языков C и Pascal.

В качестве примера рассмотрим процедуру вычисления факториалов, а также модель, для вычисления первых 15 значений факториала.

```
            GENERATE      1
            SAVEVALUE     Answer+,1
            SAVEVALUE     X$Answer,(Factorial(X$Answer))
            TERMINATE     1
start 15

PROCEDURE Factorial(Arg1) BEGIN
  TEMPORARY Result;
  Result=1;
  WHILE(Arg1>1) DO BEGIN
    Result=Result#Arg1;
    Arg1=(Arg1-1);
  END;
Fac=X$Answer^2+1;
  RETURN Result;
END;
```

Здесь в ячейки с номерами 1..15 заносятся значения соответствующих факториалов. Попутно, в метку Fac многократно заносится квадрат значения ячейки X$Answer плюс 1. По выходной статистике видно, что метка действительно получает соответствующее значение. Обратите внимание, что метка не встречается в модели, а только в процедуре. Важно также, что формальные параметры процедуры, и переменные, описанные в TEMPORARY, локализованы в ней. То есть процедура может изменять только значения глобальных меток (или матриц), а также возвращать результат вычислений через RETURN. Изменять значения аргументов – невозможно.

Выходная статистика этой модели имеет вид:

```
              GPSS World Simulation Report - Factorl.11.1

                   Saturday, October 11, 2003 17:11:37

            START TIME           END TIME  BLOCKS  FACILITIES  STORAGES
                 0.000             15.000     4        0          0

               NAME                      VALUE
            ANSWER                     10001.000
            ARG1                       UNSPECIFIED
            FAC                          226.000
            FACTORIAL                  10000.000
            RESULT                     UNSPECIFIED


 LABEL              LOC  BLOCK TYPE     ENTRY COUNT CURRENT COUNT RETRY
                    1    GENERATE            15             0       0
                    2    SAVEVALUE           15             0       0
                    3    SAVEVALUE           15             0       0
                    4    TERMINATE           15             0       0


SAVEVALUE               RETRY       VALUE
 1                        0           1.000
 2                        0           2.000
 3                        0           6.000
 4                        0          24.000
 5                        0         120.000
 6                        0         720.000
 7                        0        5040.000
 8                        0       40320.000
 9                        0      362880.000
 10                       0     3628800.000
 11                       0    39916800.000
 12                       0   479001600.000
 13                       0  6227020800.000
 14                       0 87178291200.000
 15                       0 1307674368000.000
 ANSWER                   0          15.000


FEC XN   PRI         BDT      ASSEM  CURRENT  NEXT  PARAMETER    VALUE
    16    0         16.000     16      0      1
```

Полезно привести и файл компиляции, он имеет вид:

```
10/11/03 17:11:37  Model Translation Begun.
10/11/03 17:11:37  Ready.
10/11/03 17:11:37  FACTORIAL Procedure registered.
10/11/03 17:11:37  Simulation in Progress.
10/11/03 17:11:37  The Simulation has ended.  Clock is 15.000000.
10/11/03 17:11:37  Reporting in Factorl.11.1  -  REPORT Window.
```

Здесь видно, что система зарегистрировала процедуру FACTORIAL. Версия примерно такой же процедуры в рекурсивной форме, имеет вид:

```
GENERATE        1
 SAVEVALUE      Answer+,1
SAVEVALUE       X$Answer,(Factorial(X$Answer))
TERMINATE       1
start 15

PROCEDURE Factorial(Arg1) BEGIN
 IF (Arg1<=1) THEN RETURN 1;
```

```
  ELSE RETURN (Arg1#(Factorial(Arg1-1)));
END;
```

Количество и содержание процедур определяется потребностями разработчика модели. Мы оставили в стороне некоторые возможности системы моделирования, связанные с экспериментами, командами статистического анализа, и интегрирования, так как потребность в них возникает только при профессиональном использовании системы.

Приложение 1. Стандартные числовые атрибуты.

К системным числовым атрибутам относятся следующие величины:

RNj - число, вычисляемое j датчиком случайных чисел . Все датчики генерируют последовательность равномерно распределенных случайных чисел из интервала от 0 до 999 включительно. В случае использования его в качестве аргумента функции они генерируют случайные числа из интервала от 0 до 0.999999;

C1 - текущее значение условного времени. Автоматически изменяется программой и устанавливается в 0 управляющими операторами CLEAR или RESET;

AC1 - текущее значение абсолютного времени. Автоматически изменяется программой. Эта величина не меняется под действием управляющего оператора RESET и устанавливается в 0 лишь под действием оператора CLEAR;

TG1 - число, равное текущему значению счетчика завершений. Заявки, вошедшие в блоки TERMINATE с ненулевым операндом A, уменьшают значение этого счетчика на число, равное значению операнда A;

Z1 - возвращает размер свободной оперативной памяти в байтах;

Заявки имеют следующие СЧА:

XN1 - номер активной заявки;

M1 - время пребывания заявки в модели. Эта величина может изменяться блоком MARK. Время пребывания вычисляется следующим образом: M1 равно разнице текущего значения абсолютного времени и отметки времени активной заявки;

PR - приоритет активной заявки. Эта величина может изменяться блоком PRIORITY. По умолчанию приоритет равен 0.

A1 - номер ансамбля, к которому принадлежит заявка.

Pномер или *номер, или *имя, или P$имя - значение P -параметра с указанным номером или значение P - параметра с именем имя для активной заявки.

Косвенное обращение к стандартному числовому атрибуту формируется как начальная часть СЧА, затем * , а затем имя или номер P параметра, в котором находится номер или имя уточнения основного СЧА. Например, FC*1 – это количество заявок, вошедших в устройство, номер или имя которого занесено в P1, SC*NUM – это количество заявок, вошедших в многоканальное устройство, номер или имя которого занесено в P$NUM. Для косвенного обращения можно использовать только P - параметры.

MPномер или MP$имя - значение времени, равное разности абсолютного модельного времени и содержимого соответствующего параметра текущей заявки;

MBномер или MB$метка - флаг синхронизации : 1 , если заявка в блоке с данным номером или меткой принадлежит тому же семейству, что и активная заявка ; 0 - в противном случае. Проверка имеет смысл, если блок с номером или меткой является одним из следующих блоков: ASSEMBLE, GATHER, или MATCH.

Блоки имеют следующие СЧА:

Nномер или N$метка - общее число заявок, которое вошло в блок с данным номером или меткой. Подсчет ведется программой автоматически. Например, N$MET1 - счетчик числа входов в блок MET1. Этот счетчик изменяется при каждом входе заявки в блок MET1;

Wномер или W$метка - текущее число заявок, которые находятся в блоке с данным номером или меткой. Значение этого счетчика также подсчитывается автоматически. Например, W$MET2 - счетчик текущего числа заявок в блоке MET2.

Многоканальные устройства имеют следующие СЧА:

Sномер или S$метка - текущее число занятых единиц емкости в многоканальном устройстве . Величина изменяется блоками ENTER и LEAVE. Например, S$OPER – текущий занятый объем многоканального устройства OPER;

Rномер или R$метка - число свободных единиц емкости многоканального устройства. Эта величина также изменяется блоками ENTER и LEAVE. Например, R$MACH – текущий свободный объем многоканального устройства MACH;

SRномер или SR$метка - коэффициент использования многоканального устройства в тысячных долях, т.е., если коэффициент равен 0.65, то SR равно 650;

SAномер или SA$метка - среднее содержимое многоканального устройства ;

SMномер или SM$метка - максимальное содержимое многоканального устройства ;

SCномер или SC$метка - общее число входов в многоканальное устройство;

STномер или ST$имя - среднее время пребывания заявок в многоканальном устройстве.

SEномер или SE$имяj - флаг не занятости многоканального устройства : 1 - свободно, 0 - занято;

SFномер или SF$имя - флаг заполненности многоканального устройства : 1 - заполнено, 0 - не заполнено;

SVномер или SV$имя - флаг доступности многоканального устройства : 1 - доступно , 0 - не доступно;

Устройства имеют следующие СЧА:

Fномер или F$имя - текущее состояние устройства. Эта величина равна 0, если устройство свободно, и 1 - во всех остальных случаях. Этот атрибут

изменяется блоками SEIZE, RELEASE, PREEMPT и RETURN. Например, F$ACPU - состояние устройства ACPU;

FIномер или FI$имя - флаг прерывания устройства: 1, если устройство находится в состоянии прерывания, 0 - в противном случае;

FVномер или FV$имя - флаг доступности устройства к использованию: 1 , если доступно, 0 - в противном случае;

FRномер или FR$имя - коэффициент использования устройства в тысячных долях, т.е., если коэффициент равен 0.88, то FR равен 880;

FCномер или FC$имя - общее число входов в устройство;

FTномер или FT$имя - среднее время использования устройства одной заявкой.

Очереди имеют следующие СЧА:

Qномер или Q$имя - длина соответствующей очереди. Эта величина может изменяться блоками QUEUE и DEPART. Например, Q2 соответствует очереди 2, а Q$FAC соответствует очереди FAC;

QAномер или QA$имя - средняя длина очереди.

QMномер или QM$имя - максимальная длина очереди. Это значение автоматически определяется и сохраняется программой;

QCномер или QC$имя - общее число вхождений в очередь. Это значение определяется автоматически;

QZномер или QZ$имя - число нулевых вхождений в очередь. То есть число заявок, которые вошли в очередь, но в ней не задержались. Число нулевых вхождений подсчитывается как число заявок, которые, войдя в Queue, тут же прошли в другой блок, вне зависимости от того, что это за блок. Это значение также подсчитывается автоматически;

QTномер или QT$имя - среднее время пребывания заявки в очереди (включая нулевые вхождения);

QXномер или QX$имя - среднее время пребывания заявок в очереди (без нулевых вхождений).

Таблицы имеют следующие СЧА:

TBномер или TB$имя - вычисленное среднее для таблицы. Для занесения в таблицу используется блок TABULATE;

TCномер или TC$имя - общее число включений в таблицу;

TDномер или TD$имя - вычисленное среднеквадратичное отклонение для таблицы.

Ячейки, ключи и матрицы ячеек сохраняемых величин имеют следующие СЧА:

Xномер или X$имя - содержимое ячейки;

LSномер или LS$имя - возвращает состояние логического ключа : 1 - установлен, 0 - не установлен.

MXномер(A,B) или MX$имя(A,B)- содержимое элемента матрицы ячеек, расположенного в строке A и в, столбце B;

Вычислительные объекты имеют следующие СЧА:

FNномер или FN$имя - вычисленное значение функции.

Vномер или V$имя - вычисленное значение переменной.

BVномер или BV$имя - вычисленное значение булевской переменной.

Фактически, при использовании описателей выражений VARIABLE, FVARIABLE, BVARIABLE результат всегда вещественный.

Группы имеют следующие СЧА:

GNномер или GN$имя - текущее число членов в числовой группе;

GTномер или GT$имя -текущее число членов в группе заявок.

Списки пользователя имеют следующие СЧА:

CHномер или CH$имя - текущее число заявок в списке пользователя;

CAномер или CA$имя -среднее число заявок в списке пользователя;

CMномер или CM$имя -максимальное число заявок списке пользователя;

CCномер или CC$имя -общее число заявок, попавших в список пользователя;

CT номер или CT$имя - среднее время пребывания заявки в списке пользователя;

Приложение 2. Стандартная выходная статистика.

Окно статистики состоит из подразделов, содержащих стандартную статистику об объектах GPSS World, используемых в данной модели (FACILITY, QUEUE, STORAGE и т.д.). Начинается окно статистики с заголовка, который содержит имя файла с текстом модели, номер версии, и номер последней выполненной команды START. Он выглядит, например следующим образом.

```
GPSS World Simulation Report - ex2.10.5
```

Затем указывается дата и время моделирования. Например, в следующем виде.

```
Monday, December 02, 2002 11:44:01
```

Затем идет информация о времени моделирования и о числе блоков, устройств, и многоканальных устройств. Например, в следующем виде.

```
START TIME          END TIME  BLOCKS  FACILITIES  STORAGES
119722.946        244383.594    50        1          3
```

Элементы статистики, представленные в этой строке имеют следующее содержание:

- START TIME - абсолютное системное время в момент начала моделирования. Оно эквивалентно абсолютному системному времени, после последнего применения операторов RESET или CLEAR;
- END TIME - абсолютное время, когда счетчик завершений принимает значение 0;
- BLOCKS - количество блоков, использованных в текущей модели, к моменту завершения моделирования;

- FACILITIES - количество устройств, использованных в модели, к моменту завершения моделирования;

- STORAGES - количество многоканальных устройств, использованных в текущей модели к моменту завершения моделирования.

Затем идет информация о метках, использованных в модели и об их значениях. Например, в следующем виде.

```
    NAME                      VALUE
FFC                         10010.000
UST                         10004.000
FF                          10006.000
```

Здесь NAME – это имя метки, а VALUE - это её числовое значение.

Далее описываются блоки текущей модели, например в следующем виде:

```
LABEL              LOC  BLOCK TYPE     ENTRY COUNT CURRENT COUNT RETRY
                     1    GENERATE          1025             0       0
                     2    QUEUE             1025             6       0
                     3    SEIZE             1019             0       0
                     4    DEPART            1019             0       0
                     5    ADVANCE           1019             1       0
                     6    RELEASE           1018             0       0
```

Здесь поле LABEL определяет метку блока, поле LOC определяет имя или номер этого блока. Поле BLOCK TYPE определяет тип блока GPSS World.

Поле ENTRY COUNT определяет количество заявок, вошедших в данный блок, после последнего выполнения блоков RESET или CLEAR, или с начала работы программы.

Поле CURRENT COUNT определяет количество заявок, находящихся в данном блоке в конце моделирования.

Поле RETRY определяет количество заявок, ожидающих специальных условий, зависящих от состояния данного блока.

Если в модели используются объекты типа "устройство", то далее в файле статистики идет информация об этих объектах. Например,

```
FACILITY         ENTRIES  UTIL.   AVE. TIME AVAIL. OWNER PEND INTER RETRY DELAY
 FAC1              1000    0.865     837.784  1        0    0    0     0      0
```

Поле FACILITY определяет номер или имя объекта типа "устройство".

Поле ENTRIES определяет количество раз, когда устройство было занято или прервано после последнего выполнения блоков RESET или CLEAR или с начала работы программы.

Поле UTIL. определяет часть периода моделирования, в течение которого устройство было занято.

Поле AVE.TIME определяет среднее время занятости устройства одной заявкой в течение периода моделирования после последнего выполнения операторов RESET или CLEAR.

Поле AVAILABLE определяет состояние готовности устройства в конце периода моделирования. Оно равно 1 , если устройство готово и 0 - если не готово.

Поле OWNER определяет номер последней заявки, занявшей устройство. 0 означает, что устройство не занято.

Поле PEND определяет количество заявок, ожидающих устройство, находящееся в "режиме прерывания".

Поле INTER определяет количество заявок, прерывающих устройство в данный момент (счетчик заявок в списке прерывания).

Поле RETRY определяет количество заявок, ожидающих специальных условий, зависящих от состояния объекта типа "устройство".

Поле DELAY определяет количество заявок, ожидающих занятия устройства. Сюда входят также заявки, ожидающие освобождения устройства в "режиме прерывания" (блок PREEMPT).

В случае использования в модели объектов типа "очередь", далее следует информация об этих объектах. Например,

```
QUEUE              MAX CONT. ENTRY ENTRY(0) AVE.CONT. AVE.TIME   AVE.(-0) RETRY
 UST                26   24    747     43     9.489   1033.404   1096.524   0
```

Поле QUEUE определяет имя или номер объекта типа "очередь".

Поле MAX определяет максимальное содержимое объекта типа "очередь" в течение периода моделирования, который начинается с начала работы программы или с последнего оператора RESET или CLEAR.

Поле CONT определяет текущее содержимое объекта типа "очередь" в конце периода моделирования.

Поле ENTRIES определяет общее количество входов в очередь в течение периода моделирования (счетчик входов).

Поле ENTRIES(0) определяет общее количество входов в очередь с нулевым временем ожидания (счетчик "нулевых" входов).

Поле AVE.CONT определяет среднее значение содержимого очереди.

Поле AVE.TIME определяет среднее время, проведенное в очереди с учетом всех входов в очередь.

Поле AVE.(-0) определяет среднее время, проведенное в очереди без учета "нулевых" входов в очередь.

Поле RETRY определяет количество заявок, ожидающих специальных условий, зависящих от состояния объекта типа "очередь".

Если в модели использовались объекты типа "многоканальное устройство", то далее в файле статистики идет информация об этих объектах. Например,

```
STORAGE            CAP. REM. MIN. MAX.  ENTRIES AVL.  AVE.C. UTIL. RETRY DELAY
 UST                 3    3   0     3     2363   0    1.384  0.461    0   85
```

Поле STORAGE определяет имя или номер объекта типа "многоканальное устройство".

Поле CAP. определяет емкость многоканального устройства, заданную оператором STORAGE.

Поле REMAIN определяет число единиц свободной емкости многоканального устройства в конце периода моделирования.

Поле MIN определяет минимальное количество используемой емкости многоканального устройства за период моделирования.

Поле MAX определяет максимальное количество используемой емкости многоканального устройства за период моделирования.

Поле ENTRIES определяет количество входов в многоканальное устройство за период моделирования.

Поле AVL. определяет состояние готовности многоканального устройства в конце периода моделирования. 1 - означает, что многоканальное устройство готово, 0 - не готово.

Поле AVE.C определяет среднее значение занятой емкости за период моделирования.

Поле UTIL. определяет часть периода моделирования в течение, которого многоканальное устройство использовалось.

Поле RETRY определяет количество заявок, ожидающих специальных условий, зависящих от состояния многоканального устройства.

Поле DELAY определяет количество заявок, ожидающих возможности входа в блок ENTER.

Если в модели используются блоки TABLE или QTABLE в файле стандартной статистики будет представлена информация о таблицах. Например,

TABLE	MEAN	STD.DEV.	RANGE			RETRY	FREQUENCY	CUM.%
TT	1.035	1.960				0		
				-	0.000		117	58.50
			0.000	-	1.000		47	82.00
			1.000	-	2.000		14	89.00
			2.000	-	3.000		2	90.00
			3.000	-	4.000		3	91.50
			4.000	-	5.000		5	94.00
			5.000	-	6.000		5	96.50
			6.000	-	7.000		2	97.50
			7.000	-	8.000		1	98.00
			8.000	-	9.000		3	99.50
			9.000	-	10.000		1	100.00

Поле TABLE определяет имя или номер объекта типа "таблица" или "Q-таблица".

Поле MEAN определяет среднее взвешенное значение табулируемого аргумента. Значение преобразовывается в формат двойной точности при выводе в файл статистики.

Поле STD.DEV определяет взвешенное среднеквадратичное отклонение.

S.D. = SQR((SOS/(COUNT-1)) - (SUM/(COUNT)(COUNT-1)))

где SOS - сумма квадратов значений аргументов,
COUNT - число входов в таблицу,
SUM - квадрат суммы значений аргументов таблицы.

S.D. преобразуется в формат двойной точности при выводе в файл статистики.

Поле RETRY определяет количество заявок, ожидающих выполнения специальных условий, зависящих от состояния объекта типа "таблица".

Поле RANGE определяет верхний и нижний пределы частотных классов. При попадании табулируемого аргумента в интервал, который имеет значение большее нижней границы частотного класса и меньшее или равное верхней границе, изменяется значение частоты (FREQUENCY). Операнд B блока TABULATE может быть использован для определения величины, которая добавляется в частотный класс при попадании табулируемого значения в этот частотный класс. Частотные классы, суммарное значение которых равно нулю, в файл статистики не выводятся. Значения частотных классов не уменьшаются при их изменении.

Поле FREQUENCY определяет суммарную величину, которая формируется при попадании табулируемого аргумента в указанные границы. Суммируются значения операнда B блоков TABULATE.

Поле CUM.% определяет величину частоты в процентах к общему количеству значений табулируемого аргумента.

Далее в файле выходной статистики следует информация о логических переключателях, если они использовались в модели.

LOGICSWITCH	VALUE	RETRY
SWITCH1	1	0

Поле LOGICSWITCH определяет имя или номер объекта типа "логический переключатель".

Поле VALUE определяет значение логического переключателя в конце моделирования. Здесь 1 - означает "установлен" или "истина", а 0 - означает "сброшен" или "ложь".

Поле RETRY определяет количество заявок, ожидающих наступления специальных условий, зависящих от состояния логического переключателя.

Далее в файле статистики следует информация о сохраняемых величинах (ячейках), если они использовались в модели. Например,

SAVEVALUE	VALUE	RETRY

CLOCKSAVE +100571 0

Поле SAVEVALUE определяет имя или номер объекта, типа "сохраняемая величина".

Поле VALUE определяет значение сохраняемой величины в конце моделирования.

Поле RETRY определяет количество заявок, ожидающих наступления специальных условий, зависящих от состояния сохраняемой величины.

Приложение 3. Работа с окнами наблюдения процесса моделирования в GPSS- World.

Доступ к этим окнам возможен после компиляции модели. Желательно открывать их до начала моделирования, то есть до выполнения команды Start.

Если в главном меню системы войти в пункты Window, Simulation Window, то можно выбрать возможность наблюдения за следующими окнами:

Blocks Window
Expression Window
Facilities Window
LogSwitches Window
Matrix Window
Plot Window
Queues Window
SaveValues Window
Storages Window
Table Window

В принципе, можно выбрать наблюдение за любым количеством окон.

В окне Blocks Window можно наблюдать за прохождением заявок по блокам, количеством вошедших в блок заявок и количеством заявок, находящихся в блоке.

В окне Expression Window можно наблюдать за изменениями значений выражений в ходе моделирования. Для этого в окне редактирования выражений нужно ввести метку нового выражения, и само выражение. Затем, щелкнув по View или по Memorize внести его в окно наблюдения, либо запомнить, с тем, чтобы можно было его наблюдать позже. Те выражения, которые находятся в панели Window Content – вычисляются и отображаются в ходе моделирования. Их можно удалить, выделив, и щелкнув по Remove.

Те выражения, которые находятся в панели Memorized Expressions, можно внести в Window Content, если их выделить и щелкнуть по View. Их можно удалить, щелкнув по Delete. Списки выражений сохраняются, пока не удалено окно компиляции.

В окне Facilities Window можно наблюдать за СЧА устройств в ходе моделирования.

В окне LogSwitches Window можно наблюдать за логическими ключами в ходе моделирования.

В окне Matrix Window можно наблюдать за значениями элементов матриц в ходе моделирования. Каждую матрицу можно наблюдать в отдельном окне. Для просмотра тех значений матрицы, которые не видны сразу, используйте вертикальную и горизонтальную прокрутки.

В окне Plot Window можно наблюдать графики изменения значений выражений в ходе моделирования. Окно редактирования для графиков во многом похоже на подобное окно для выражений, но требует указания шага вывода данных по времени, минимального значения и максимального значения выводимой величины. При просмотре графика можно использовать прокрутки.

В окне Queues Window можно наблюдать за изменениями значений СЧА очередей в ходе моделирования.

В окне SaveValues Window можно наблюдать за изменениями значений ячеек в ходе моделирования.

В окне Storages Window можно наблюдать за изменениями значений СЧА многоканальных устройств в ходе моделирования.

В окне Table Window можно наблюдать за изменениями в гистограмме таблицы. Каждую таблицу нужно наблюдать в отдельном окне.

Приложение 4 Дополнительные встроенные процедуры и функции языка PLUS.

Процедура DoCommand(строка); – компилирует и выполняет оператор языка. Сам оператор обычно задается как строковая переменная, так как иначе проще указать нужный оператор сразу. Эту процедуру разрешается вызывать только внутри эксперимента, или в процедуре, вызванной в эксперименте. В строке, представляющей оператор, не должны встречаться временные переменные или элементы временных матриц. Оператор, содержащийся в строке должен быть исполнительным, а не описательным.

Процедура Exit – завершает работу с системой GPSS.

Она имеет вид:

Exit(код завершения);

Код может быть равен -1, 0, 1 , что определяет режим завершения.

Он может быть "без сохранения", "запрос на сохранение", или "сохранить все" , то есть все модели и результаты компиляции.

Процедура используется достаточно редко. Разве что в экспериментах.

Функции работы с файлами могут использоваться в выражениях и процедурах языка PLUS.

Функция Open(A, B) - открывает файл.

Здесь – A номер файла, а B – имя файла, оно должно быть строкой.

Результатом функции является код ошибки.

Коды ошибок – следующие:

0 – нет ошибок.

10 – слишком длинное имя файла.

11 - ошибка считывания внешнего файла.
12 – не хватило памяти для отражения файла в памяти.

Функция Close(A) закрывает файл с сохранением его на диске.
Здесь A – номер файла. Возвращает код ошибки. Ошибка могла произойти и раньше в процессе работы с файлом.
Коды ошибок:
0 – нет ошибок.
10 - ошибка открытия файла, имя слишком длинное.
11 - ошибка считывания внешнего файла.
12 – не хватило памяти для отражения файла в памяти.
21 – ошибка чтения.
22 - ошибка чтения, файл не был открыт.
31 – ошибка записи.
32 - ошибка записи, файл не был открыт.
41 - ошибка закрытия файла.
43 - ошибка закрытия файла, файл не был открыт.
51 – ошибка поиска, файл не был открыт.

Функция Read(A) – читает данные из файла с номером A. Возвращаемое значение – прочитанная строка. Подробности ее работы, такие же, как у соответствующего блока.

Функция Write(A,B) – она записывает строку B в файл с номером A. Возвращает код ошибки.
Он может быть:
0 – нет ошибки.
31 – ошибка записи.
32 - ошибка записи, файл не был открыт.

Функция Seek (A,B) – перемещает указатель файла с номером A на строку номер B. Возвращает номер строки, на которую ранее был установлен указатель файла.
В библиотеке имеется большое число теоретических функций распределения, с которыми при необходимости, можно ознакомиться в разделе «справка».

Перечень ссылок

1. Основы теории вычислительных систем, М. Высшая школа,1978.
2. Бусленко Н.П. Моделирование сложных систем, М. Наука,1978.
3. Советов Б.Я. Моделирование систем, М. Высшая школа,1985.
4. Советов Б.Я. Моделирование систем (курсовое проектирование), М. Высшая школа,1986.
5. Советов Б.Я. Моделирование систем (лабораторный практикум), М. Высшая школа,1986.

6. Молчанов А.А. Моделирование и проектирование сложных систем, К. Выща школа,1988.
7. Альянах И.Н. Моделирование вычислительных систем, Л. Машиностроение, 1988
8. Серия разработка САПР Имитационное моделирование В.М. Черненький, N9, М. Высшая школа,1990.
9. Дал У., Мюрхауг Б., Нюгорд К. Универсальный язык моделирования. - М.: Мир, 1969.- 316 с.
10. Емельянов А.А., Власова Е.А. Имитационное моделирование в экономических информационных системах. - М.: МЭСИ, 1996.- 108 с.
11. Клейнен Дж. Статистические методы в имитационном моделировании. - М.: Статистика, 1978.-221с.
12. Машинные имитационные эксперименты с моделями экономических систем./ Под ред. Нейлора Т.М. - М.: Мир, 1975.-501с.
13. Шеннон Р. Дж. Имитационное моделирование систем - искусство и наука.-М.:Мир,1978.-418с.
14. Шрайбер Т. Дж. Моделирование на GPSS World.- М.: Машиностроение, 1980.-592с.

Оглавление