А.Г. КОРОЛЕВ

МОДЕЛИРОВАНИЕ СИСТЕМ СРЕДСТВАМИ GPSS - FUTURE

ПРАКТИЧЕСКИЙ ПОДХОД В ПРИМЕРАХ И ЗАДАЧАХ

Северодонецк – 2009

УДК 62. 529

А.Г. Королёв Моделирование систем средствами GPSS - Future. Практический подход в примерах и задачах. Учебное пособие. Библиогр.14

ISBN

Рассматриваются методы и техника моделирования систем массового обслуживания, основанные на дискретно- событийном подходе. Изложение ведется на основе средств GPSS -Future. 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.

Оглавление

ОГЛАВЛЕНИЕ....2
1 ВВЕДЕНИЕ....3
2 ДОСТОИНСТВА GPSS - FUTURE В СРАВНЕНИИ С ДРУГИМИ ЯЗЫКАМИ МОДЕЛИРОВАНИЯ....6
3 РАБОТА С СИСТЕМОЙ GPSS - FUTURE....6
4 ОБЩИЕ СВЕДЕНИЯ О СИСТЕМЕ GPSS - FUTURE....9
5 ВВЕДЕНИЕ В ПРЕДМЕТНУЮ ОБЛАСТЬ ЯЗЫКА GPSS....11
6 СХЕМА ОБРАБОТКИ ОСНОВНЫХ СОБЫТИЙ....18
7 МОДЕЛЬНЫЙ ТАЙМЕР И ЗАВЕРШЕНИЕ МОДЕЛИРОВАНИЯ....18
8 ОДНОВРЕМЕННЫЕ СОБЫТИЯ....19
9 ВЫВОДЫ....20
10 ОСНОВНЫЕ КОНЦЕПЦИИ МОДЕЛИРОВАНИЯ НА GPSS....20
11 СПИСКИ В GPSS....21
12 ОСНОВНЫЕ ВИДЫ ОБЪЕКТОВ В GPSS....22
13 ПРОЦЕДУРЫ GPSS - FUTURE. БЛОКИ И СОБСТВЕННО ПРОЦЕДУРЫ....24
14 ОСНОВНЫЕ ПРОЦЕДУРЫ И БЛОКИ GPSS....25
15 РАБОТА С МОДЕЛЬЮ В GPSS - FUTURE....30
16 ТЕХНИКА ПОСТРОЕНИЯ МОДЕЛИ И ПРОВЕДЕНИЯ МОДЕЛИРОВАНИЯ35
17 РАБОТА С МНОГОКАНАЛЬНЫМИ УСТРОЙСТВАМИ....43
18 АРИФМЕТИЧЕСКИЕ, ЛОГИЧЕСКИЕ И СТРОКОВЫЕ ВЫРАЖЕНИЯ В GPSS - FUTURE....47
19 ТАБУЛИРОВАНИЕ ПЕРЕМЕННЫХ....49
20 ФУНКЦИИ В GPSS - FUTURE....52
21 ФУНКЦИИ И ПРОЦЕДУРЫ РАБОТЫ СО СТРОКАМИ....59
22 БЛОКИ ПРИСВАИВАНИЯ В GPSS....60
23 ПЕРЕНАПРАВЛЕНИЕ ПОТОКОВ ЗАЯВОК....76
24 БЛОК ПРОВЕРКИ УСЛОВИЙ....83
25 РАБОТА С ПРЕРЫВАНИЯМИ....117
26 ВВОД И ВЫВОД ДАННЫХ ПРИ МОДЕЛИРОВАНИИ....123
27 РАБОТА С СЕМЕЙСТВАМИ ЗАЯВОК....128
28 СПИСКИ ПОЛЬЗОВАТЕЛЯ....141
29 ОСОБЕННОСТИ СПИСКОВ БУДУЩИХ СОБЫТИЙ....149

30 БЛОКИ ЗАДАНИЯ ДОСТУПНОСТИ И НЕДОСТУПНОСТИ УСТРОЙСТВ 155
31 БЛОКИ РАБОТЫ С ГРУППАМИ. 164
32 РЕДКО ИСПОЛЬЗУЕМЫЕ БЛОКИ..... 174
33 ЭЛЕМЕНТАРНЫЕ СВЕДЕНИЯ О ЯЗЫКЕ PASCAL..... 175
34 ВЫБОРКА ТРЕБУЕМЫХ ОБЪЕКТОВ В GPSS - FUTURE..... 180
35 РАБОТА С ФАЙЛАМИ 185
36 ОСНОВЫ УПРАВЛЕНИЯ МОДЕЛЬЮ И РАЗВИТИЕ GPSS - FUTURE..... 193
ПРИЛОЖЕНИЕ 1..... 199
БЛОКИ, ПРОЦЕДУРЫ И ФУНКЦИИ ЯЗЫКА, "GPSS - FUTURE" 199
ПРОЦЕДУРЫ И ФУНКЦИИ МОДУЛЯ GPSS_SYS 206
РАБОТА_С_Р_ПАРАМЕТРАМИ..... 210
РАБОТА_С_ПЕРЕМЕННЫМИ_И_ВВОД_ВЫВОД..... 213
ТАБЛИЦЫ 214
ОДНОКАНАЛЬНЫЕ_УСТРОЙСТВА..... 215
ОЧЕРЕДИ..... 216
НАКОПИТЕЛИ..... 216
СПИСКИ_ПОЛЬЗОВАТЕЛЯ..... 217
СПИСКИ_БУДУЩЕГО 219
ГРУППЫ..... 220
ГРУППЫ_ЗАЯВОК..... 222
ФУНКЦИИ_СИСТЕМЫ..... 224
ДОПОЛНИТЕЛЬНЫЕ И ФУНКЦИИ, РЕАЛИЗОВАННЫЕ В МОДУЛЕ ADDMODELUNIT..... 228
ПРИЛОЖЕНИЕ 2. СТАНДАРТНАЯ ВЫХОДНАЯ СТАТИСТИКА..... 231
ПЕРЕЧЕНЬ ССЫЛОК..... 234

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 - Future, 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 с оболочкой языка, а также особенности применения различных блоков новой версии для решения конкретных задач событийного 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 по-прежнему 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 определяются 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 здесь 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 также 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 не более, чем одну заявку в каждый момент времени и «Памяти» (STORE) (переименованный в следующих версиях на STORAGE), которые могли 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.

Однако на этом история GPSS не заканчивается. В процессе эксплуатации GPSS World выяснилось, что эта система недостаточно полно использует возможности операционной системы Windows. Многие её особенности, сохранившиеся от прошлого, не способствуют освоению GPSS новым поколением специалистов, а скорее мешают им при разработке моделей. Короче говоря, возникла потребность совместить возможности языка GPSS – с возможностями одного из современных языков программирования класса 4GL. Только такое совмещение могло придать новый импульс развития языку GPSS.

В рамках этой идеи, было предложено средство, позволяющее описывать модели систем массового обслуживания прямо на Object Pascal, но в стиле языка GPSS. В этом случае модель системы пишется как набор процедур на Object Pascal, обеспечивающих моделирование системы. Сама система поддержки процесса моделирования также написана на Object Pascal, а точнее, на Delphi. Такой подход естественным образом обеспечивает модели все те возможности, которые есть в базовом языке.

На первый взгляд, модель системы, написанная как набор процедур на Object Pascal, должна быть громоздкой и сложной. Однако это не так. Большую часть текста модели можно получить автоматическими средствами, а собственно содержательная часть модели обычно оказывается небольшой, и по объему ненамного превосходит текст модели на GPSS – World или GPSS/h .

В целом, возможности системы GPSS - Future намного превосходят возможности других версий GPSS. Она является более стройной, содержит намного большее число блоков и процедур, и может быть легко развита за счет пользовательских процедур.

Пользование системой и разработка новых моделей не предполагает знания языка Object Pascal, а требует знания только основ программирования на любом языке высокого уровня.

Необходимые минимальные сведения об Object Pascal излагаются непосредственно при описании системы GPSS - Future и составляют весьма небольшую часть объема пособия.

Изложению особенностей системы дискретно-событийного моделирования - GPSS - Future и посвящена данная книга.

2 Достоинства GPSS - Future в сравнении с другими языками моделирования

Новый продукт, GPSS - Future содержит много новых функций и процедур, расширяющих возможности разработчика модели. Общее число блоков в системе доведено до 142. Система содержит 11 типов объектов и 173 процедуры для управления моделью. Общее число функций достигает 150.

Для работы с системой используется программа – конвертер, которая позволяет достаточно просто создавать, компилировать и запускать модели на выполнение. Скомпилированная модель и обеспечивает собственно моделирование.

Как программа – конвертер, так и скомпилированная модель имеют современный внешний вид, соответствующий возможностям ОС Windows и достаточно удобны как для построения моделей, так и для отображения результатов моделирования в текстовом или в графическом виде. В системе значительно облегчена отладка моделей, так как всегда можно наблюдать графики изменения нужных параметров модели, а также получать подробные «снимки» текущего состояния модели.

Функции и параметры модели – типизированы, и могут быть следующих базовых типов: целый, вещественный, строковый и логический.

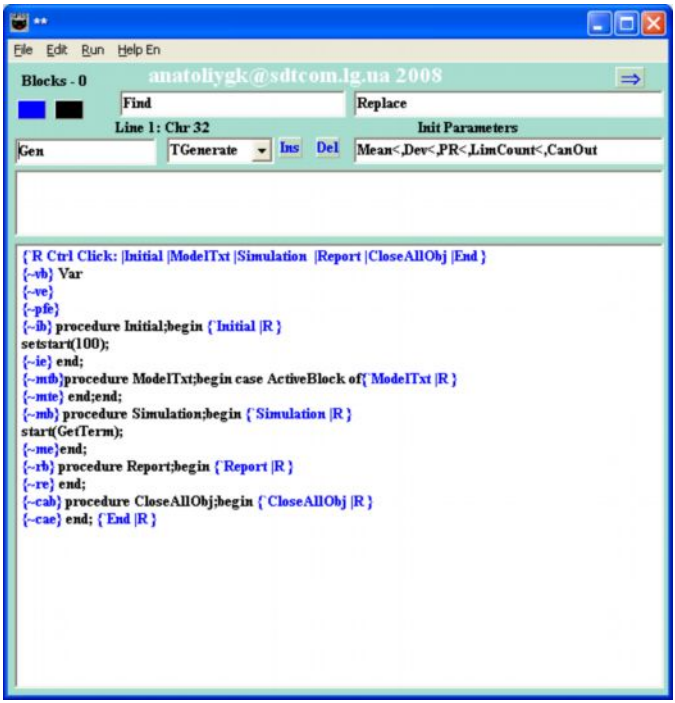
При моделировании можно использовать данные и процедуры на языке Object Pascal, которые подготавливаются разработчиком модели, если ему это нужно. Таким образом, можно упростить разработку сложных моделей, в том числе и таких, которые невозможно либо сложно реализовать на классическом GPSS. В системе можно программно управлять не только вычислениями, но и выводом результатов, но также управлять моделью как единым целым.

Система GPSS - Future, ориентирована на дискретно-событийное моделирование систем массового обслуживания. Представление жизни модели как движения во времени заявок, перемещающихся в модели и обслуживающихся в устройствах очень естественно для многих задач имитационного моделирования. Система GPSS - Future, достаточно легка 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 - Future, выполняется гораздо легче, чем другими методами. Компактность текста модели и возможность использования графического интерфейса позволяют ускорить создание прототипов моделей. При этом на каждую из них можно получить быстрый отклик после улучшений проведенных пользователем.

3 Работа с системой GPSS - Future

Подготовка модели ведется с помощью программы – конвертера, которая имеет оболочку, достаточно традиционную для приложений WINDOWS.

Внешний вид этой программы приводится ниже.



Основные возможности по управлению его работой предоставляет главное меню и дублирующие его возможности, горячие клавиши. Главное меню содержит следующие подменю.

File Edit Run Help.

Подменю File .

Это подменю содержит следующие пункты.

New

создать новую модель.

Open

открыть ранее созданную модель, или любой другой текстовый файл в формате TXT или RTF.

Save

сохранить новую версию модели в файле на диске.

Save As

сохранить модель в файле на диске, изменив имя и, возможно, папку в формате TXT или RTF.

Quit

закрытие программы – конвертера.

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

Подменю Edit

Это подменю содержит следующие пункты.

Insert Object

Позволяет вставить описание объекта, используемого в модели, а также вставить вызовы всех процедур, которые необходимы для корректной работы модели с этим объектом. Имя объекта задается в соответствующем поле, или формируется автоматически, а тип объекта выбирается из выпадающего списка.

Delete Object

Позволяет удалить описание объекта, используемого в модели, а также удалить вызовы всех процедур, которые были вставлены для корректной работы модели с этим объектом. Для удаления – следует вначале поместить курсор на строку описания объекта, или на строку со сгенерированным вызовом для объекта.

Find

Позволяет искать заданную подстроку. Регистр символов при поиске значения не имеет.

Replace

Позволяет заменить найденную подстроку на заданную подстроку и продолжить поиск заданной подстроки. Регистр символов при поиске значения не имеет.

Replace One Name

Заменяет имя, заданное в поле поиска, на имя заданное в поле замены. Имя для поиска не должно содержать разделителей, так как иначе этот пункт работать не будет.

Replace Names

Работает аналогично но пары имен должны задаваться в файле OldNewNames.txt в последовательно идущих парах строк.

FontSelected

вызов диалога изменения шрифта выделенной части модели.

Include

Union

Используются для моделей, содержащих имена файлов включения и соединения, смотри Приложение.ы

ReOpen

Читает заново файл текущей модели с диска.

Run.

выполняет конвертирование текста модели в набор процедур модели, то есть создает файл Model.pas, а также создает модель в виде EXE – файла и запускает ее на выполнение, если, конечно, не было ошибок компиляции. Иначе нужно исправить эти ошибки и повторить пункт Run. Кроме того, пункт сохраняет текущее состояние файла с текстом модели и помещает текст, сформированный компилятором, с возможными сообщениями об ошибках, в соответствующее поле конвертера. Этот текст позволяет найти и исправить формальные ошибки.

При перемещении по тексту модели, отслеживается номер строки, где находится курсор и для текста модели, и для файла Model.pas. Эта информация полезна при отладке синтаксических ошибок в модели. Сообщения об ошибках формирует компилятор с Object Pascal при выполнении пункта Run.

Подменю Help.

показывает файл справки. Справка доступна на Английском или Русском языке. По файлу справки можно перемещаться щелкая по синим гиперссылкам при нажатой клавише Ctrl, или выполнять поиск по кнопке Find.

Гиперссылка начинается с символа | и следует до пробела , а точка перехода – начинается с символа ` и тоже должна заканчиваться пробелом. По тексту модели также можно выполнять переходы по гиперссылкам. Но здесь гиперссылки и точки перехода должны находиться в комментариях, чтобы не мешать компиляции модели.

Синяя кнопка выделяет комментарии синим цветом, а текст самой модели – зеленым. Черная кнопка делает весть текст черным.

Дополнительное правое поле содержит имена всех переменных модели, а также полный перечень всех шаблонов процедур и функций системы моделирования.

Если имена переменных не соответствуют модели, то нужно закрыть и открыть это поле. Тогда они появятся все. В этом поле также можно перемещаться по гиперссылкам. Если кнопка над полем указывает влево, и вы держите нажатой клавишу Ctrl, то при щелчке по шаблону процедуры либо функции он переносится в соответствующую позицию модели. Если эта кнопка указывает вверх, и вы держите нажатой клавишу Ctrl, то шаблоны переносятся в верхнюю часть самого поля. То есть, можно вначале собрать все нужные для модели шаблоны вверху, а эатем их вставлять в нужных местах. Можно, правда и просто пользоваться буфером обмена.

4 Общие сведения о системе GPSS - Future

GPSS–Future – это высоко интегрированная компьютерная среда моделирования общего назначения, разработанная для профессионалов моделирования. Это - мощный инструмент моделирования, покрывающий, дискретно-событийное моделирование, с чрезвычайно высоким уровнем взаимодействия и визуализации. При использовании GPSS – Future, возможно предсказать поведение весьма 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 – Future.

Эта версия, GPSS – Future, - прямой потомок GPSS/PC и GPSS World для персональных компьютеров. Введение в 1984 году, GPSS /PC, а в 2000 году и GPSS World сохранило тысячам пользователей миллионы долларов. Теперь, система GPSS – Future расширяет их возможности.

Язык GPSS – Future разработан так, чтобы давать ответы быстро и надежно, с минимумом усилий.

Система GPSS – Future была разработана таким образом, чтобы обеспечить прозрачность моделирования. Она позволяет видеть внутренние механизмы моделей и зафиксировать результат. Взаимодействие модели с пользователем позволяет ему не только провести исследование, но и обеспечивает управление моделью. Так что теперь, можно проводить эксперименты и оптимизацию моделей автоматически, с относительно небольшими усилиями.

Моделирование большинства систем требует знания только малого подмножества блоков и процедур системы.

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

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 – Future, вы создаете и изменяете модель, с помощью текстового редактора. Вы затем создаете собственно компилированную модель, используя пункт Run. В модели, в ее процедуре Simulation, вы можете использовать мощный набор процедур для того, чтобы управлять ходом моделирования, а значит можно автоматизировать проведение экспериментов с моделью. Вы можете управлять моделью в интерактивном режиме, или включать процедуры управления прямо в первоначальную модель. В ходе тестирования и проверки, доступно слежение за значениями параметров модели как в виде графиков или значений, так и в текстовой форме.

Поскольку языком моделирования на самом деле является Object Pascal , то при построении модели доступны многочисленные функции и библиотеки этого языка. Написанные пользователем процедуры можно использовать наравне с библиотечными процедурами.

Типы данных в системе могут быть целые, вещественные, строковые и логические. Каждый тип строго контролируется, а преобразование типов, если необходимо, должно выполняться явным образом. В системе можно использовать многомерные массивы любых объектов, что намного повышает возможности разработчика моделей. В системе можно создавать и уничтожать наборы P - параметров в динамике.

Модели могут связываться с внешним миром, используя текстовые файлы или окна диалога. Вы можете использовать эти файлы, чтобы обратиться к данным, создавать файлы результата или заказные отчеты и делать многое другое.

Существенно важным отличием системы GPSS – Future от других систем, GPSS, в том числе и GPSS – World, Object GPSS, является наличие мощных средств работы со списком будущих событий, включая и возможность иметь несколько списков будущих событий. Сразу же отметим, что всем, кто активно использует GPSS, известно, что концепция будущего в этой системе весьма ущербна. То есть, если вы запланировали задержку заявок на некоторое время, то с этими заявками уже практически ничего сделать нельзя. Нужно ждать, пока они вновь появится в списке текущих событий. В определённом смысле, это концепция плановой экономики, когда всё, что произойдет в будущем – планируется заранее и не подлежит никакой ревизии. В новой системе – GPSS – Future список будущих событий рассматривается как наследник списка пользователя. А поэтому всегда можно извлечь из него заявки, удовлетворяющие нужным вам условиям, в том числе, и продвинув текущее время в системе к новому значению. В этом случае, вы

полностью контролируете заявки из списка будущих событий, пока они находятся в этом списке.

Например, вы просто и естественно сможете промоделировать ситуацию в банке, когда вы захотите предложить избранным кредиторам новые условия депозитного договора до истечения его срока.

В новой системе можно намного легче моделировать задачи с управляемым будущим, которое будет в полной вашей власти, до самого момента наступления этого будущего, то есть в этой системе будущее более соответствует тому, которое реально есть в нашем мире.

Когда больше не удалось продвинуть ни одну заявку из списка текущих событий, то в любой версии GPSS должна вызываться процедура продвижения к новому моменту модельного времени. В данной версии, как правило, вначале из списка будущих событий извлекаются заявки, чьё время ожидания уже истекло (так как, возможно, вы решили увеличить текущее время в процессе извлечения заявок из списка будущих событий), а, если их нет, то извлекаются заявки с минимальным временем завершения ожидания. В последнем случае, продвигается вперед текущее время моделирования. Эта процедура используется по умолчанию. Однако, если вы этого желаете, вы можете сами переписать процедуру продвижения к новому моменту модельного времени по своему вкусу. Набор практически всех возможных процедур продвижения к новому моменту модельного времени заранее подготовлен и имеется в системе.

В новой системе для генерации заявок используются процедура Future.NewWaitProc (инициализация) и блок Future.NewWait (генерация). Задержку выполняет блок Future.Wait. Здесь Future – это список будущих событий, который предлагается системой.

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 обслуживания является одинаково вероятным.

Мы будем считать, что нам нужно промоделировать работу системы при обслуживании 10000 клиентов.

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

Запустить программу Converter.exe.

Вставить такие объекты, как очередь (TQueue), одноканальное устройство (TFacility), а также генератор потока заявок (Generate). Это выполняется следующим образом.

В выпадающем списке выбирается Tqueue, а в поле имени вводится QKass. Выполняется пункт меню Insert Object.

Далее в выпадающем списке выбирается TFacility, а в поле имени вводится Kass. Выполняется пункт меню Insert Object.

И, наконец, в выпадающем списке выбирается Generate, в поле имени вводится Gen, а в поле аргументов вводится Exponential(120),0 . Выполняется пункт меню Insert Object.

После этого, в следующие строки процедуры ModelTxt вносится текст модели. В итоге, в этой процедуре вы должны получить.

```
{=Gen} ::Gen *:Future.NewTransaction(Exponential(120),0,Gen,Gen+1);
   *: Qkass.Queue ;
   *: Kass.Seize ;
   *: Qkass.Depart ;
   *: Future.Wait( MeanDev(105,25));
   *: Kass.release ;
   *: ToPoint (1,aclock,Qkass.A) ;
   *: Terminate (1);
```

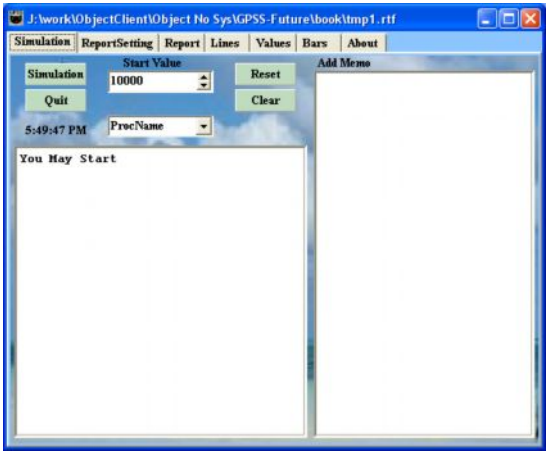
В процедуру Initial включите строку

```
  SetStart(10000);
```

Этим вы зададите число обслуживаемых клиентов.

Затем сохраните полученный файл на диске (пункт Save As), и выполните пункт Run.

Вы получите модель нужной вам системы. Она будет выглядеть следующим образом.



Нажмите на кнопку Simulation и дождитесь сообщения Stop Simulation.

Вы получите результат моделирования на вкладке Report и в файле Report.txt.

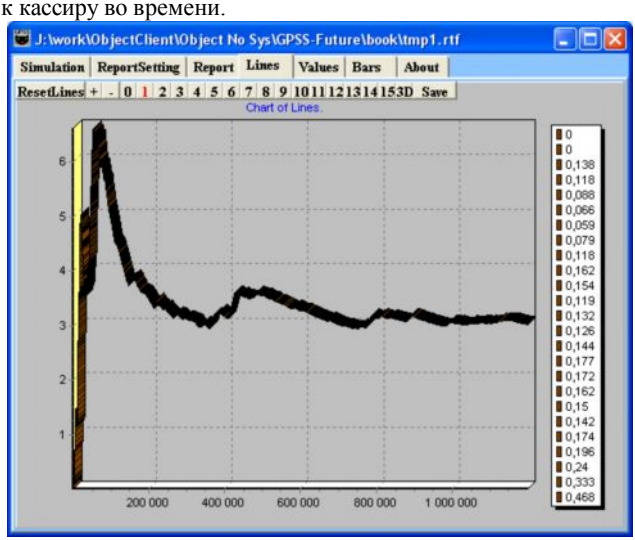
```
Date Time Stamp= 21.10.2008 17:50:54   Ctrl Click |Name - HyperLink `R
J:\work\ObjectClient\Object No Sys\GPSS-Future\book\tmp1.rtf
StartTime          0.00000 EndTime    1199885.83546
 |Model |Blocks |Values |CurrentList |FutureList
 |UserList |SynchronizeList |InterruptList |PList
   Num       NameObject
     2         |Future
     3           |Kass
     4          |Qkass

 |R `Model
 {=Gen} ::Gen 1:Future.NewTransaction(Exponential(120),0,Gen,Gen+1);
    2: Qkass.Queue ;
    3: Kass.Seize ;
    4: Qkass.Depart ;
    5: Future.Wait( MeanDev(105,25));
    6: Kass.release ;
    7: ToPoint (1,aclock,Qkass.A) ;
    8: Terminate (1);

 |R `Blocks     BLOCK Report
    Location      Total    Current
         1      10002          0
         2      10002          2
         3      10000          0
         4      10000          0
         5      10000          0
         6      10000          0
         7      10000          0
         8      10000          0
 |R `Values
 |R `CurrentList Report Current Transactions List  Count= 2
      Tran      Assem      Prior    Current       Next        TimeStamp        TimeEnter Inter
     10001      10001          0          2          3    1199627.56755    1199627.56755 FALSE
     10002      10002          0          2          3    1199783.99368    1199783.99368 FALSE
 |R    Facility    Entries    Current          Utility         AverTime       Tran    IntTran
Active Time     NextGo
```

```
          `Kass      10000          0          0.87515        105.00767          0          0
1199885.83546          0
 |R        Queue    Entries    Current        AverQueue         AverTime AverTime( - Ze)        Max
Min       Zero      Active Time     NextGo
          `Qkass      10002          2          3.01778        362.02679        476.19569         22
0       2398    1199885.83546          0
 |R         User    Entries    Current         AverUser         AverTime        Max        Min
Active Time     NextGo
          `Future      20003          1          1.29501         77.68180          2          0
1199885.83546          0
 |R `UserList Report  User List Count= 1
      Tran      Assem      Prior    Current       Next        TimeStamp        SortValue Inter
     10003      10003          0          0          1   1200030.28556   1200030.28556 FALSE
            Num/Name         Integer   |R `Future      Report IP LIST  Count= 2
I                  1           10003
I                  2           10003
```

А на вкладке Lines вы увидите следующий график изменения средней величины очереди к кассиру во времени.



Теперь мы можем обсудить как саму модель, так и полученные результаты. Как правило, вам нет никакой надобности, внимательно изучать содержимое всех процедур модели, а достаточно сосредоточиться на процедуре ModelTxt. Содержимое остальных процедур система пишет «для себя», и в большинстве случаев оно для вас не важно, если, конечно вы там ничего не испортили.

{=Gen} ::Gen *:Future.NewTransaction(Exponential(120),0,Gen,Gen+1);

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

*: Qkass.Queue ;

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

*: Kass.Seize ;

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.

*: Qkass.Depart ;

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.

*: Future.Wait(MeanDev(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.

*: Kass.release ;

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

*: ToPoint(1,aclock,Qkass.A) ;

Эта строка обеспечивает вывод точек графика средней очереди к кассиру по ходу изменения модельного времени.

*: 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 SetStart (то есть 10000).

Анализируя выходную статистику, можно определить следующие результаты моделирования. На обслуживание 10000 клиентов системе понадобилось 1199885.83546 секунд модельного времени. Всего поступило на обслуживание 10002 клиентов. Кассир был занят на 0.87515. Среднее время обслуживания клиента составило 105.00767 секунды. Максимальная величина очереди составила 22 клиента. 2398 клиентов вообще не стояли в очереди. Средняя величина очереди была 3.01778. А среднее время, которое клиент провёл в очереди, равно 362.02679 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 *: Future.Wait(MeanDev(105,25)) на

*: Future.Wait(MeanDev(80,25)), мы получим следующие результаты.

Средняя занятость кассира - `0.66525`. Среднее время обслуживания - `79.67684`, Средняя длина очереди - `0.69482`. Среднее время в очереди- `83.19341`, что уже гораздо лучше.

Если посмотреть, что именно мы сохранили в качестве текста модели, то обнаружится следующее ее содержимое.

```
{`R Ctrl Click: |Initial |ModelTxt |Simulation  |Report |CloseAllObj |End }
 {~vb} Var
  Future: TFuture;
{=Kass} Kass : TFacility;
{=Qkass} Qkass : TQueue;
```

```
 {~ve}
{~pfe}
 {~ib} procedure Initial;begin {`Initial |R }
  init(Future,'Future');
setstart(10000);
{=Gen} Future.NewTransactionProc(Gen,Exponential(120),0,Gen,Gen+1);
{=Kass} Init(Kass,'Kass');
{=Qkass} Init(Qkass,'Qkass');
 {~ie} end;
 {~mtb}procedure ModelTxt;begin case ActiveBlock of{`ModelTxt |R }
{=Gen} ::Gen *:Future.NewTransaction(Exponential(120),0,Gen,Gen+1);
   *: Qkass.Queue ;
   *: Kass.Seize ;
   *: Qkass.Depart ;
   *: Future.Wait( MeanDev(80,25));
//    *: Future.Wait( MeanDev(105,25));
   *: Kass.release ;
   *: ToPoint (1,aclock,Qkass.A) ;
   *: Terminate (1);
 {~mte} end;end;
 {~mb} procedure Simulation;begin {`Simulation |R }
 start(GetTerm);
 {~me}end;
 {~rb} procedure Report;begin {`Report |R }
 {~re} end;
 {~cab} procedure CloseAllObj;begin {`CloseAllObj |R }
 {~cae} end; {`End |R }
```

Здесь можно обнаружить раздел описания объектов модели, раздел ее инициализации (procedure initial), раздел собственно описания модели (procedure modeltxt) , раздел манипуляций с моделью (procedure Simulation) и раздел формирования отчета (procedure report). Раздел удаления элементов модели (procedure CloseAllObj). А также ряд дополнительных процедур и функций. Однако так как фактически мы написали только раздел описания модели, то обсуждение полного текста модели пока отложим. Здесь и в дальнейшем, текст, содержащий собственно модель будем выделять зеленым цветом.

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 время наступления некоторых из них можно запланировать. Когда наступит время для запланированного 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 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 времени 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 или ухода заявки из очереди должны быть расширены, 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) второй 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 перестроен 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 уже имеющихся заявок.

Для завершения моделирования, в GPSS предусмотрен так называемый счетчик завершений. Процедура Start из Simulation 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.

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 порядка 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 получения стандартных законов распределения (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 это 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 4-х видов:

динамических,

аппаратно-ориентированных,

статистических,

операционных.

Динамические объекты - это элементы потока обслуживания (заявки). Они создаются или уничтожаются в модели специальными операторами. Работа системы отображается в модели в виде перемещения заявок от блоков NewWait 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 (P - параметров), которые задают желаемые характеристики моделируемого подвижного объекта.

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 *:Future.Wait, то она переходит из списка текущих событий в список будущих событий и для неё планируется время начала движения (после задержки). Если заявку удалость продвинуть по модели, то просмотр списка текущих событий начинается сначала, иначе, очередная заявка, из числа потенциально активных заявок, получат возможность двигаться. Когда окажется, что ни одна заявка из списка текущих событий не может быть продвинута, 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, и просмотр списка текущих событий начинается сначала.

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 *: Future.Wait или запланированные к выходу из блока *:Future.NewWait.

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 ссылки на заявки, обслуживание которых прервано на устройстве. В этот список ссылки заносятся по мере поступления, а извлекаются по мере надобности. В этом 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 процедуры и функции, которые не принадлежат никакому классу.

Clock - текущее значение условного времени моделирования. Оно автоматически изменяется программой и устанавливается в 0 управляющими процедурами ClearModel или ResetModel;

AClock - текущее значение абсолютного времени моделирования. Оно автоматически изменяется программой. Эта величина не меняется под действием управляющей процедуры ResetModel и устанавливается в 0 ишь под действием процедуры ClearModel;

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

Наиболее важным объектом системы GPSS – Future являются заявки. Они появляются из блока NewWait и проходят через различные блоки модели, а затем уничтожаются блоками Terminate. С заявками связан ряд параметров, которые можно использовать для организации логики работы модели.

Параметры заявки (функции):

Tran Выдает номер активной заявки.

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

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

Assem - номер ансамбля, к которому принадлежит заявка.

P – параметры заявки могут иметь следующий вид: RP(номер) или IP(номер) или SP(номер) или BP(номер), обычно значение параметра «номер» выбирается для активной заявки. Эти параметры являются вещественными, целыми, строковыми или логическими соответственно. Если вы хотите узнать значение P- параметра, какой либо другой заявки (не активной), то следует указать ее номер вторым аргументом. Параметры заявки

перемещаются и гибнут вместе с заявкой. Параметр создается при первом присваивании ему значения.

IsMatch(номер) - флаг синхронизации: True , если заявка, находящаяся в блоке с данным номером, принадлежит тому же семейству, что и активная заявка; False - в противном случае.

С параметрами (функциями) других объектов GPSS, можно ознакомиться в Приложении 1.

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

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, Release. Для проверки различного рода условий используется блок Test. Прерывания моделируют блоки: Preempt, Return, и так далее.

Многоканальные устройства служат для моделирования объектов обладающих емкостью.

Вход и выход для 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 процедуры Init.

Для представления в модели системных данных, используются так называемые X – параметры, которые могут иметь следующий вид: ObjName.RX(номер) или ObjName.IX(номер) или ObjName.SX(номер) или ObjName.BX(номер), - значения любого X - параметра принадлежит системе. Эти параметры являются вещественными, целыми, строковыми или логическими соответственно. X - параметры обычно существуют до конца моделирования. Запись в эти параметры ведут блоки RXlet, IXlet, SXlet, BXlet. Данные, записанные в X – параметры, доступны любой заявке.

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, 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 параметра и представления ее в виде стандартной таблицы используется блок Tabulate, а для ее описания – процедура Init.

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 Процедуры GPSS - Future. Блоки и собственно процедуры

Каждый блок занимает в тексте модели определенное положение и имеет свой номер. Блоком называется особая процедура, которая работает только тогда, когда в нее входит заявка. Имя такой процедуры имеет право появиться только в процедуре ModelTxt.

Каждому блоку можно сопоставить символьную метку, которая располагается перед ключевым словом блока, отделяясь от него, по крайней мере, одним пробелом. Обыкновенный блок и блок с меткой должны выглядеть следующим образом.

*:Блок;

::метка *:Блок;

На символьную метку можно ссылаться непосредственно в выражениях и операндах. Метка может входить в выражения целого типа. Обычно метки имеют те блоки, на которые ссылаются другие блоки модели. В роли ссылки на метку можно использовать комбинацию символов !* , которая всегда замещается конвертером на номер следующего по порядку блока.

Операция блока - это глагол, определяющий основную роль блока.

Test - проверяет

Queue - ставит в очередь

*: Future.Wait - задерживает

Операнды задают уточняющую информацию для работы блока. Всего операндов может быть до 5. Типично когда в блоке 1 или 2 операнда.

Пример:

::Mm1 *: Future.Wait(MeanDev(300,20));

Блок *: Future.Wait задерживает заявку на 300+-20 единиц времени.

Операнды разделяются запятыми.

Комментарий – это текст, заключенный в символы { } или текст от пары символов // - до конца строки или текст, заключенный в пары символов (* *) . Вид комментария определяется по виду начала комментария. Рекомендуется первый или второй вид комментариев.

С блоками модели связаны следующие параметры (функции):

Total(номер) - общее число заявок, которое вошло в блок с данным номером. Подсчет ведется программой автоматически. Например, Total (MET1) - счетчик числа входов в блок с меткой ::MET1. Этот счетчик изменяется при каждом входе заявки в блок;

Current (номер) - текущее число заявок, которые находятся в блоке с данным номером. Значение этого счетчика также подсчитывается автоматически. Например, Current(MET2) - счетчик текущего числа заявок в блоке с меткой ::MET2.

Собственно процедуры ни в коем случае нельзя указывать в процедуре ModelTxt, и наоборот, блоки нельзя использовать вне процедуры ModelTxt. Обычно в подобных случаях возникает сообщение системы об ошибке или, реже - системная ошибка.

14 Основные процедуры и блоки GPSS

Процедура Start(число) - запускает моделирование. Она при работе с моделью может вызываться неоднократно. После прекращения моделирования, можно изменить модель с помощью процедур и вновь запустить ее с помощью процедуры Start. Здесь число - определяет содержимое счетчика завершений.

Блоки Terminate могут вычитать определенное значение из счетчика завершений. Когда в счетчике завершений будет 0 или отрицательная величина, то моделирование завершится.

Моделирование можно приостановить досрочно, нажав кнопку Start/Stop. Затем можно продолжить моделирование, вновь нажав эту же кнопку. Пример текста простой модели на GPSS – Future мы рассматривали выше.

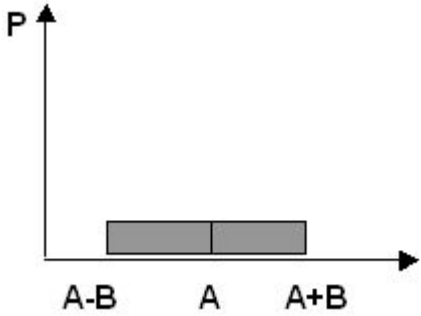
Для генерирования заявок используется блок *: Future.NewWait, имеющий вид:

::Gen *: Future.NewWait (Delay, PR, Gen, Gen+1<,Group<,CanOut<,LimCount<,NumBl);

Блок задает параметры потока заявок.

Здесь ::Gen - это обязательная метка блока.

Если нужно организовать равномерное распределение интервалов задержки в стиле GPSS, то нужно использовать функцию MeanDev(Mean,Dev) вместо Delay. Тогда операнды Mean и Dev, задаваемые как вещественные или целые положительные числа (возможно, выражения), определят равномерное распределение интервалов при генерации заявок. Интервалы между заявками могут быть от Mean-Dev до Mean +Dev включительно. Плотность вероятности в интервале равна 1/(2*Dev).



Здесь всегда Mean должно быть больше или равно Dev. Параметр PR – определяет приоритет заявки, рекомендуется 0. Это самый низкий приоритет.

Параметры Gen,Gen+1 – это номера IP - параметров, в которых сохраняется количество выпущенных блоком заявок, и номер последней запланированной заявки для блока. Узнать значения этих параметров можно с помощью функций IP(Gen,Future.N) и IP(Gen+1,Future.N). В принципе, значения этих параметров могут быть и не Gen,Gen+1, важно только, чтобы они были уникальны для каждой пары NewWaitProc и NewWait. Параметры Gen, Gen+1 менять не рекомендуется. Параметр Group, - номер группы заявок в списке, по умолчанию, и как правило, равный 0. Параметр LimCont – определяет граничное число заявок, которое может выпустить генератор, по умолчанию он равен 0 и

ограничений на число заявок - нет. Параметр CanOut определяет, может ли быть выпущена заявка из генератора, по умолчанию, он имеет значение истина (True). Параметр NumBl – определяет, куда пойдет запланированная заявка, по умолчанию или по 0 – в данный блок.

Приход новой заявки планируется только после выхода предыдущей.

Для каждого генератора в разделе инициализации помещается вызов процедуры

Future.NewWaitProc(Delay, PR, Gen, Gen+1<,Group);

Она создает генератор с именем Gen и меткой Gen, а также планирует приход в него первой заявки. Остальные параметры этой процедуры имеют тот же смысл, что и в блоке NewWait, но относятся к первой заявке.

Как правило, параметры Delay, PR – одинаковы в Future.NewWait и в Future.NewWaitProc.

Ввод заявок в систему ведет блок Future.NewWait, 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(Num);

Моделирование задержки заявки ведет блок *:Future.Wait(Delay<,Group), который имеет 1 или 2 параметра. Блок очень похож на *:Future.NewWait и его параметры Delay и Group имеют тот же смысл.

Войти в *:Future.NewWait и находиться в нем одновременно, может любое число заявок. Обычно блок *:Future.NewWait 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

*:Ustr.Seize;

а освобождение устройства отображается блоком

*: Ustr.Release;

Здесь Ustr – это имя устройства.

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

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 должна покидать систему.

Для моделирования доступности устройства и включения отсылки заявок в определенное место модели используется блок.

*: Ustr.SetGoto(NumBl);

Если NumBl=0, то заявкам будет разрешено входить в устройство. Если NumBl<0, то заявкам будет запрещено входить в устройство. Если NumBl>0, то заявки, претендующие на устройство, в безусловном порядке будут отсылаться на блок с данным номером. По умолчанию, NumBl=0.

С устройствами связаны следующие функции:

Ustr.L - 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.

Ustr.IsPree – логическое значение, определяющее, работает ли устройство в состоянии прерывания.

Ustr.A - определяет коэффициент занятости устройства.

Ustr.V - определяет временной интеграл устройства.

Ustr.T -определяет среднее время занятости устройства одной заявкой.

Ustr.C - определяет число заявок, вошедших в устройство.

Ustr.Tran - определяет номер обслуживаемой заявки для устройства.

Ustr. InterruptTran - определяет номер последней прерванной заявки для устройства.

Ustr.Go - определяет значение параметра NextGo, который установил блок *: Ustr.SetGoto.

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

*: QUst.Queue(Num) ;

и

*: QUst.Depart(Num) ;

Здесь QUst – это имя очереди, а Num- 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 *: QUst.Queue(Num) текущая длина очереди увеличивается на Num , при выходе через *: QUst.Depart(Num) , уменьшается на Num . По умолчанию, значение Num равно 1.

С очередями связаны следующие функции:

QUst.A- Определяет среднюю величину очереди.

QUst.T - Определяет среднее время в очереди для одной заявки.

QUst.X - Определяет среднее время в очереди для одной заявки.
Оно определяется без учета заявок, в очереди не задержавшихся.

QUst.C- Определяет число вхождений в очередь, с учетом кратности.

QUst.L - Определяет текущую длину очереди с учетом кратности.

QUst.M - Определяет текущую максимальную длину очереди с учетом кратности.

QUst.V - Определяет временной интеграл очереди с учетом кратности.

QUst.Z- Определяет текущее число вхождений в очередь с учетом кратности. Оно определяется для заявок, в очереди не задержавшихся.

Внимание!

При создании моделей не забывайте вставить в модель описания всех объектов, с которыми ваша модель работает.

Не забывайте, что каждый блок NewWait является новым генератором, а поэтому нельзя создавать новые блоки NewWait путем копирования.

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 (процедура ModelTxt) выглядит следующим образом.

```
{=Gen} ::Gen *:Future.NewTransaction(MeanDev(100,80),0,Gen,Gen+1);
   *: Qkass.Queue ;
   *: Kass.Seize ;
   *: Qkass.Depart ;
   *: Future.Wait( MeanDev(70,20));
   *: Kass.release ;
   *: terminate ;
{=Gen0} ::Gen0 *:Future.NewTransaction(100000,0,Gen0,Gen0+1);
   *: terminate(1) ;
```

В процедуре SetStart должно быть указан операнд , равный 1.

В данном примере, заявки приходят в систему с интервалом 100+- 80 единиц. Поступают в очередь Qkass, затем занимают устройство с именем Kass. Далее покидают очередь Qkass. Задерживаются в устройстве на 70+- 20 единиц времени. Затем покидают устройство Kass 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 этой модели должны быть описаны следующие объекты:

Gen, Gen0, Qkass, Kass.

То есть нужно описать два генератора заявок, очередь и одноканальное устройство.

В данном случае итоговая статистика модели будет выглядеть следующим образом.

```
Date Time Stamp= 21.10.2008 22:10:11   Ctrl Click |Name - HyperLink `R
J:\work\ObjectClient\Object No Sys\GPSS-Future\book\tmp2.rtf
StartTime         0.00000 EndTime     100000.00000
 |Model |Blocks |Values |CurrentList |FutureList
 |UserList |SynchronizeList |InterruptList |PList
   Num      NameObject
     2        |Future
     3          |Kass
     4         |Qkass

 |R `Model
 {=Gen} ::Gen 1:Future.NewTransaction(MeanDev(100,80),0,Gen,Gen+1);
     2: Qkass.Queue ;
     3: Kass.Seize ;
     4: Qkass.Depart ;
     5: Future.Wait( MeanDev(70,20));
     6: Kass.release ;
     7: terminate ;
{=Gen0} ::Gen0 8:Future.NewTransaction(100000,0,Gen0,Gen0+1);
     9: terminate(1) ;

 |R `Blocks    BLOCK Report
```

```
    Location     Total   Current
         1      1016         0
         2      1016         0
         3      1016         0
         4      1016         0
         5      1016         1
         6      1015         0
         7      1015         0
         8         1         0
         9         1         0
 |R `Values
 |R    Facility   Entries   Current         Utility        AverTime      Tran   IntTran
Active Time    NextGo
          `Kass      1016         1         0.71084        69.96440      1017         0
100000.00000         0
 |R       Queue   Entries   Current       AverQueue        AverTime AverTime( - Ze)       Max
Min      Zero     Active Time    NextGo
         `Qkass      1016         0         0.17538        17.26133       224.83992         4
0       938    100000.00000         0
 |R        User   Entries   Current        AverUser        AverTime       Max       Min
Active Time    NextGo
        `Future      2035         3         2.18159       107.20342         3         0
100000.00000         0
 |R `UserList Report  User List Count= 3
      Tran     Assem     Prior   Current      Next       TimeStamp       SortValue Inter
      1018      1018         0         0         1    100012.95221    100012.95221 FALSE
      1017      1017         0         5         6     99947.63370    100013.37028 FALSE
      1019      1019         0         0         8    200000.00000    200000.00000 FALSE
            Num/Name        Integer  |R `Future      Report IP LIST  Count= 4
I                  1           1017
I                  2           1018
I                  8              2
I                  9           1019
```

Пусть в первом примере, мы организуем многократное моделирование обслуживания по 500 групп заявок. Всего разрешим генератору выпустить 1200 заявок. В конце, когда моделирование завершится по ошибке, посмотрим содержимое отчета. Моделирование будем вести по количеству обслуженных заявок. Тогда вторая ветвьа модели, начинающаяся со второго генератора – отсутствует, а вызов процедуры SetStart имеет вид SetStart(500).

В такой модели строка генерации заявок будет выглядеть следующим образом.

```
{=Gen} ::Gen *:Future.NewTransaction(MeanDev(100,80),0,Gen,Gen+1,0,true,1200);
```

Нетрудно убедиться, что в этом случае моделирование действительно прервалось по ошибке.

Выходная статистика этого моделирования, имеют вид:

```
Date Time Stamp= 21.10.2008 22:13:37   Ctrl Click |Name - HyperLink `R
J:\work\ObjectClient\Object No Sys\GPSS-Future\book\tmp3.rtf
StartTime         0.00000 EndTime     123345.17014
 |Model |Blocks |Values |CurrentList |FutureList
 |UserList |SynchronizeList |InterruptList |PList
   Num      NameObject
     2        |Future
     3          |Kass
     4         |Qkass

 |R `Model
 {=Gen} ::Gen 1:Future.NewTransaction(MeanDev(100,80),0,Gen,Gen+1,0,true,1200);
     2: Qkass.Queue ;
     3: Kass.Seize ;
     4: Qkass.Depart ;
     5: Future.Wait( MeanDev(70,20));
     6: Kass.release ;
     7: terminate(1) ;

 |R `Blocks    BLOCK Report
    Location     Total   Current
```

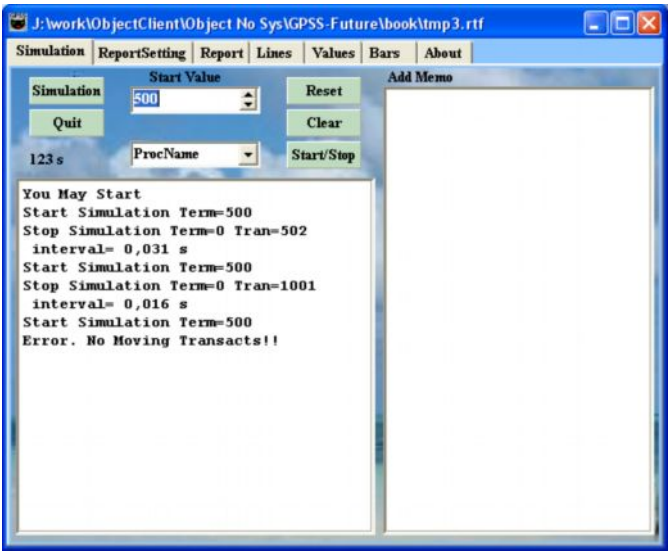
```
         1      1200         0
         2      1200         0
         3      1200         0
         4      1200         0
         5      1200         0
         6      1200         0
         7      1200         0
 |R `Values
 |R    Facility   Entries   Current         Utility        AverTime      Tran   IntTran
Active Time    NextGo
          `Kass      1200         0         0.68165        70.06563         0         0
123345.17014         0
 |R       Queue   Entries   Current       AverQueue        AverTime AverTime( - Ze)       Max
Min      Zero     Active Time    NextGo
         `Qkass      1200         0         0.15219        15.64372       182.25697         3
0      1097    123345.17014         0
 |R        User   Entries   Current        AverUser        AverTime       Max       Min
Active Time    NextGo
        `Future      2400         0         1.16012        59.62275         2         0
123345.17014         0
            Num/Name        Integer  |R `Future      Report IP LIST  Count= 2
I                 1           1200
I                 2           1200
```

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

Процедура Start в ходе моделирования может быть выполнена 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.

15 Работа с моделью в GPSS - Future

На вкладке Simulation приложения для работы с моделью расположены следующие кнопки.



Кнопка Simulation

Нажатие этой кнопки запускает выполнение процедуры Simulation, описанной в модели системы. В простейшем варианте, в этой процедуре находится только процедура Start(GetTerm); в которую передается значение поля, определяющего начальное содержимое счетчика завершений. В более сложных ситуациях здесь находятся процедуры управления моделью.

Кнопка Start/Stop

Нажатие этой кнопки позволяет временно приостановить выполнение моделирования. Однако, нажатие этой кнопки завершает выполнение текущей процедуры Start, и продолжение текущего процесса моделирования возможно только при повторном нажатии этой же кнопки.

Кнопка Reset

Нажатие этой кнопки вызывает сброс текущей статистики в текущее состояние. При этом все заявки остаются на прежних местах, и после выполнения процедуры Start – моделирование продолжится, однако статистика начнет собираться заново.

Кнопка Clear

Нажатие этой кнопки вызывает полный сброс модели, и она оказывается в том же состоянии, как и при запуске приложения.

Кнопка Quit

Нажатие этой кнопки завершает работу приложения.

На вкладке также находится поле для ввода значения счетчика завершений.

Кроме того, на вкладке есть выпадающий список, который позволяет вызывать пользовательские процедуры управления моделью.

Для текстовых полей по правой клавише мыши вызываются пункты

Open

Save

FontSelected

Они позволяют открыть в поле текстовый файл, сохранить содержимое поля как текстовый файл, изменить шрифт, а также обеспечивают работу с буфером обмена.

На вкладке Report Setting можно установить перечень частей отчета, которые попадут в отчет о моделировании и в файл Report.txt.

Эти части – следующие.

CurrentList – отображать или нет список текущих событий.

FutureList– отображать или нет список будущих событий.

SynchronizeList – отображать или нет список синхронизируемых заявок.

InterruptList– отображать или нет список ссылок на прерванные заявки.

UserList– отображать или нет списки пользователя.

PList – отображать или нет список P - параметров.

GroupList– отображать или нет списки групп.

ParamList – отображать или нет список P - параметров.

Blocks - формировать или нет отчет по блокам.

Queue - формировать или нет отчет по очередям.

Storage - формировать или нет отчет по многоканальным устройствам.

User- формировать или нет отчет по спискам пользователя.

Facility - формировать или нет отчет по одноканальным устройствам.

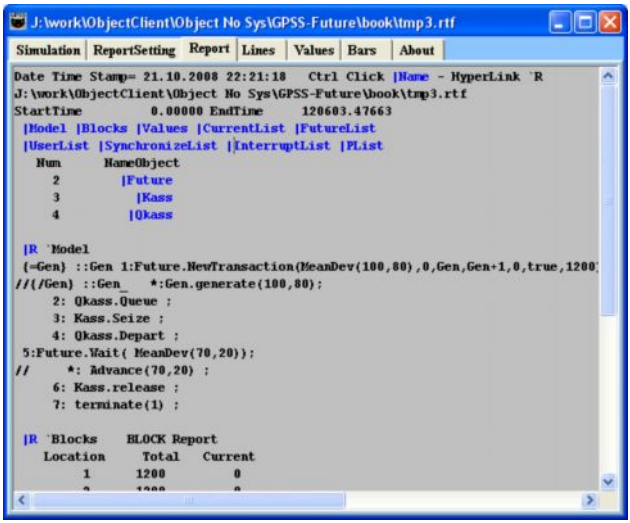
Table - формировать или нет отчет по таблицам.

Val - формировать или нет отчет по переменным модели.

Group - формировать или нет отчет по группам.

Model - формировать или нет отчет по тексту модели.

На вкладке Report можно увидеть отчет, соответствующий настройкам предыдущей вкладки.



По правой клавише мыши вызываются пункты

Open

Save

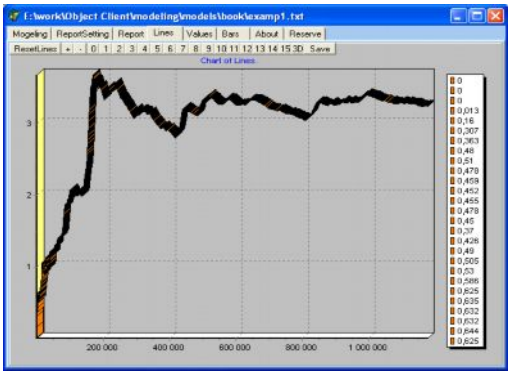
FontSelected

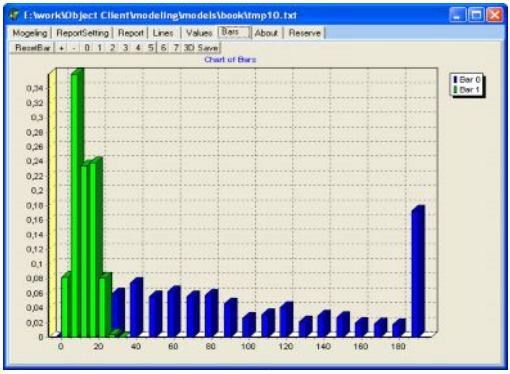
WordWrap

По первым трем ранее упомянутым пунктам делается то же самое, что и в остальных всплывающих меню, а по WordWrap – устанавливается и отменяется перенос строк для компонента с текстом.

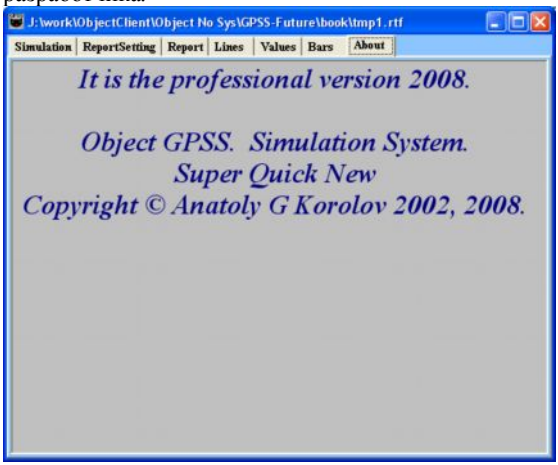
На вкладках Lines и Bars имеются кнопки для очистки графиков и гистограмм, а также для их сокрытия и отображения, и сохранения изображение в файле на диске. Если на этих вкладках левой кнопкой выделить прямоугольник, содержащий фрагмент изображения перемещением мыши левой клавишей вправо и вниз, то этот прямоугольник «растянется» на всю вкладку. Выделение прямоугольника другим способом – возвращает изображение к исходному виду.

Правой кнопкой можно перемещать графики или гистограммы как единое целое.

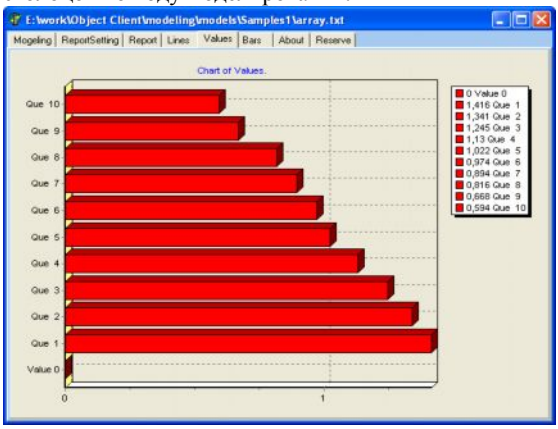




На вкладке About приводятся сведения об авторских правах и адрес разработчика.



Вкладка Values предназначена для вывода числовых значений в виде величин столбцов по ходу моделирования.



16 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 – Future, 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.

Перед набором текста, а если нужно то и по ходу набора, необходимо вставлять или удалять объекты, используя пункты Edit/Insert Object и Edit/Delete Object. Имейте в виду, что удаляются автоматически только те строки с объектом, у которых псевдокомментарий вида {/Kass} размещен с первой позиции строки.

Когда текст будет набран, нужно выбрать пункт Run. Будет создана компилированная модель. Если при компиляции были ошибки, то модель не будет создана, а ошибки можно посмотреть в поле сообщений компиляции. Ошибки следует исправить в тексте, используя сведения о строках с ошибками. В программе – конвертере строка с надписью вида Line 40 определяет номера строк в исходном файле, и в файле Model.pas. В окне компиляции номера строк с ошибками будут определяться по файлу Model.pas, так что обнаружение строк с ошибками не составит труда.

После исправления ошибок можно запустить модель на выполнение, повторив пункт Run.

Когда моделирование будет завершено, то сформируется итоговая статистика на вкладке Report. Имейте в виду, что статистика формируется каждый раз заново, при каждом щелчке по этой вкладке. Одновременно эта статистика попадает и в файл Report.txt

При необходимости можно будет продолжить процесс моделирования, щелкнув по кнопке Simulation. Перед этим можно выполнить полный или частичный сброс статистики кнопками Clear или Reset.

Подробности истолкования итоговой статистики можно узнать из приложения 2.

Если сохранить файл ModelName.exe, то вы сможете эксплуатировать модель и без программы – конвертера. 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 процедур GPSS – Future, или путем использования текстовых файлов, которые компилированная модель может прочитать. В процессе моделирования можно наблюдать за работой модели с помощью вкладок Lines, Values, Bar, а также прямо на вкладке Simulation в ее текстовых элементах. Подробнее эти возможности мы будем обсуждать позднее.

Рассмотрим способы управления компилированной моделью с помощью окон диалога и переменных.

Прежде всего, заменим все числа - переменными , и обеспечим задание их значения с помощью функций ввода.

Дополнительно нужно будет описать 5 вещественных переменных, а именно:

Inter_in, Inter_Work, Dev_in, Dev_Work, Time

Получим, например, следующий текст модели.

```
{`R Ctrl Click: |Initial |ModelTxt |Simulation  |Report |CloseAllObj |End }
 {~vb} Var
  Future: TFuture;
   Inter_in:Double;
   Inter_Work:Double;
   Dev_in:Double;
   Dev_Work:Double;
   Time:Double;
{/QKass} QKass:Tqueue;
{/Kass} Kass:Tfacility;
 {~ve}
  {~pfe}
 {~ib} procedure Initial;begin {`Initial |R }
  init(Future,'Future');
  Inter_in:=Rinput('Интервал поступления');
  Inter_Work:=Rinput('Интервал обслуживания');
  Dev_in:=Rinput('Отклонение поступления');
  Dev_Work:=Rinput('Отклонение обслуживания');
  Time:=Rinput('Время моделирования');
  SetStart(1);
{/QKass}  init(QKass,' QKass');
{/Kass}  init(Kass,' Kass');
{=Gen} Future.NewTransactionProc(Gen,MeanDev(Inter_in,Dev_in),0,Gen,Gen+1);
{=Gen0} Future.NewTransactionProc(Gen0,Time,0,Gen0,Gen0+1);
 {~ie} end;
 {~mtb}procedure ModelTxt;begin case ActiveBlock of{`ModelTxt |R }
{=Gen} ::Gen *:Future.NewTransaction(MeanDev(Inter_in,Dev_in),0,Gen,Gen+1);
    *: Qkass.Queue ;
    *: Kass.Seize ;
    *: Qkass.Depart ;
```

```
    *: Future.Wait(  MeanDev(Inter_Work,Dev_Work));
    *: Kass.release ;
    *: terminate() ;
{=Gen0} ::Gen0 *:Future.NewTransaction(Time,0,Gen0,Gen0+1);
    *: terminate(1) ;
 {~mte} end;end;
 {~mb} procedure Simulation;begin {`Simulation |R }
 start(GetTerm);
 {~me}end;
 {~rb} procedure Report;begin {`Report |R }
{/Inter_in}  reportval(Inter_in,'Inter_in = ');
{/Inter_Work}  reportval(Inter_Work,'Inter_Work = ');
{/Del_in}  reportval(Dev_in,'Dev_in = ');
{/Del_Work}  reportval(Dev_Work,'Dev_Work = ');
{/Time}  reportval(Time,'Time = ');
 {~re} end;
 {~cab} procedure CloseAllObj;begin {`CloseAllObj |R }
 {~cae} end; {`End |R }
```

Проведя моделирование, нетрудно убедиться, что результат этого моделирования будет точно таким же, как и у исходной модели, если, конечно, вы введете те же исходные данные.

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

Очевидно что, перед запуском модели заново, мы можем задать новые значения одной или нескольким переменным модели. Например, мы можем задать следующие значения переменных.

Inter_work = 90.35

Time = 200000

Получим другие результаты в файле итоговой статистики.

Аналогичным образом можно управлять другими параметрами модели. В этой модели каждая функция Rinput выведет окно для ввода вещественного значения, а текст, указанный в вызове попадет в заголовок окна.

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) и Normal(A,B). Здесь в Exponential(A)

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

A в Normal(A,B)

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

B – это среднеквадратичное отклонение интервала.

Например, если в предыдущей задаче интервал поступления заявок подчиняется экспоненциальному закону, интервал обслуживания – нормальному, то текст модели может приобрести следующий вид.

```
{`R Ctrl Click: |Initial |ModelTxt |Simulation  |Report |CloseAllObj |End }
 {~vb} Var
```

```
  Future: TFuture;
  Inter_in:Double;
   Inter_Work:Double;
   Dev_in:Double;
   Dev_Work:Double;
   Time:Double;
{/QKass} QKass:Tqueue;
{/Kass} Kass:Tfacility;
 {~ve}
{~pfe}
 {~ib} procedure Initial;begin {`Initial |R }
  init(Future,'Future');
  Inter_in:=Rinput('Интервал поступления');
  Inter_Work:=Rinput('Интервал обслуживания');
  Dev_in:=Rinput('Отклонение поступления');
  Dev_Work:=Rinput('Отклонение обслуживания');
  Time:=Rinput('Время моделирования');
  SetStart(1);
{/QKass}  init(QKass,' QKass');
{/Kass}  init(Kass,' Kass');
{=Gen} Future.NewTransactionProc(Gen,exponential(Inter_in),0,Gen,Gen+1);
{=Gen0} Future.NewTransactionProc(Gen0,Time,0,Gen0,Gen0+1);
 {~ie} end;
 {~mtb}procedure ModelTxt;begin case ActiveBlock of{`ModelTxt |R }
{=Gen} ::Gen *:Future.NewTransaction(exponential(Inter_in),0,Gen,Gen+1);
    *: Qkass.Queue ;
    *: Kass.Seize ;
    *: Qkass.Depart ;
    *: Future.Wait( Normal(Inter_Work,Dev_Work));
    *: Kass.release ;
    *: terminate() ;
{=Gen0} ::Gen0 *:Future.NewTransaction(Time,0,Gen0,Gen0+1);
    *: terminate(1) ;
 {~mte} end;end;
 {~mb} procedure Simulation;begin {`Simulation |R }
 start(GetTerm);
 {~me}end;
 {~rb} procedure Report;begin {`Report |R }
{/Inter_in}  reportval(Inter_in,'Inter_in = ');
{/Inter_Work}  reportval(Inter_Work,'Inter_Work = ');
{/Del_in}  reportval(Dev_in,'Dev_in = ');
{/Del_Work}  reportval(Dev_Work,'Dev_Work = ');
{/Time}  reportval(Time,'Time = ');
 {~re} end;
 {~cab} procedure CloseAllObj;begin {`CloseAllObj |R }
 {~cae} end; {`End |R }
```

Рассмотрим расширенный пример использования блоков работы с устройствами.

Пусть на одном компьютере работают два пользователя. Один из них использует его со средним интервалом 6 часов, который распределен по экспоненциальному закону. Работа на компьютере продолжается в течение 1.5 +-0.5 часа и включает в себя обращение к интернет через сервер в течении 20+-10 минут в начале работы.

Другой использует его со средним интервалом 4 часа, распределенным по экспоненциальному закону. Работа на компьютере продолжается в течении 1.3 +-0.75 часа и включает в себя обращение к интернет 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. При описании модели следует вставить 2 устройства: PC, SERV, 4 очереди: QPC, QSERV, Quser1, Quser2 и 4 генератора gen0, gen1, gen2, gen3. В дальнейшем, виды вставляемых устройств и их имена смотрите в разделе Var модели.

Текст модели может выглядеть, например, следующим образом.

```
{`R Ctrl Click: |Initial |ModelTxt |Simulation  |Report |CloseAllObj |End }
 {~vb} Var
  Future: TFuture;
{/PC} PC:Tfacility;
{/SERV} SERV:Tfacility;
{/QPC} QPC:Tqueue;
{/QSERV} QSERV:Tqueue;
{/Quser1} Quser1:Tqueue;
{/Quser2} Quser2:Tqueue;
 {~ve}
{~pfe}
 {~ib} procedure Initial;begin {`Initial |R }
  init(Future,'Future');
{/PC}  init(PC,' PC');
{/SERV}  init(SERV,' SERV');
{/QPC}  init(QPC,' QPC');
{/QSERV}  init(QSERV,' QSERV');
{/Quser1}  init(Quser1,' Quser1');
{/Quser2}  init(Quser2,' Quser2');
  SetStart(120);
{=Gen0} Future.NewTransactionProc(Gen0,exponential(6*60),0,Gen0,Gen0+1);
{=Gen1} Future.NewTransactionProc(Gen1,exponential(4*60),0,Gen1,Gen1+1);
{=Gen2} Future.NewTransactionProc(Gen2,exponential(30),0,Gen2,Gen2+1);
{=Gen3} Future.NewTransactionProc(Gen3,22*8*60,0,Gen3,Gen3+1);
```

```
{~ie} end;
{~mtb}procedure ModelTxt;begin case ActiveBlock of{`ModelTxt |R }
{=Gen0} ::Gen0 *:Future.NewTransaction(exponential(6*60),0,Gen0,Gen0+1);
   *: QPC.Queue;
    *: PC.Seize;
    *: QPC.Depart;
    *: Quser1.Queue;
    *: QSERV.Queue;
    *: SERV.SEIZE;
    *: QSERV.Depart;
    *:Future.Wait( MeanDev(20,10));
    *: SERV.Release;
    *:Future.Wait( MeanDev(1.5*60,0.5*60));
    *: Quser1.Depart;
    *: PC.Release;
    *: Terminate;
{=Gen1} ::Gen1 *:Future.NewTransaction(exponential(4*60),0,Gen1,Gen1+1);
    *: QPC.Queue;
    *: Quser2.Queue;
    *: PC.Seize;
    *: QPC.Depart;
    *: QSERV.Queue;
    *: SERV.Seize;
    *: QSERV.Depart;
    *:Future.Wait( MeanDev(15,7));
    *: SERV.Release;
    *:Future.Wait( MeanDev(1.3*60,0.75*60));
    *: Quser2.Depart;
    *: PC.Release;
    *: Terminate;
{=Gen2} ::Gen2 *:Future.NewTransaction(exponential(30),0,Gen2,Gen2+1);
   *: QSERV.Queue;
   *: SERV.Seize;
   *: QSERV.Depart;
   *:Future.Wait(exponential(15));
   *: SERV.Release;
   *: Terminate;
{=Gen3} ::Gen3 *:Future.NewTransaction(22*8*60,0,Gen3,Gen3+1);
    *: Topoint(0,aclock,Quser1.t);
    *: Topoint(1,aclock,Quser2.t);
    *: Terminate(1);
{~mte} end;end;
{~mb} procedure Simulation;begin {`Simulation |R }
start(GetTerm);
{~me}end;
{~rb} procedure Report;begin {`Report |R }
```

{~re} end;
{~cab} procedure CloseAllObj;begin {`CloseAllObj |R }
{~cae} end; {`End |R }

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 QPC, QSERV, Quser1, Quser2.

Результат моделирования может выглядеть, например, следующим образом.

```
Date Time Stamp= 23.10.2008 21:10:28   Ctrl Click |Name - HyperLink `R
E:\work\ObjectClient\Object No Sys\GPSS-Future\book\tmp6.rtf
StartTime         0.00000 EndTime    1267200.00000
 |Model |Blocks |Values |CurrentList |FutureList
 |UserList |SynchronizeList |InterruptList |PList
   Num      NameObject
     2        |Future
     3           | PC
     4         | SERV
     5          | QPC
     6        | QSERV
     7       | Quser1
     8       | Quser2

 |R `Model
 {=Gen0} ::Gen0 1:Future.NewTransaction(exponential(6*60),0,Gen0,Gen0+1);
    2: QPC.Queue;
     3: PC.Seize;
     4: QPC.Depart;
     5: Quser1.Queue;
     6: QSERV.Queue;
     7: SERV.SEIZE;
     8: QSERV.Depart;
     9:Future.Wait( MeanDev(20,10));
     10: SERV.Release;
     11:Future.Wait( MeanDev(1.5*60,0.5*60));
     12: Quser1.Depart;
     13: PC.Release;
     14: Terminate;
{=Gen1} ::Gen1 15:Future.NewTransaction(exponential(4*60),0,Gen1,Gen1+1);
     16: QPC.Queue;
     17: Quser2.Queue;
     18: PC.Seize;
     19: QPC.Depart;
     20: QSERV.Queue;
     21: SERV.Seize;
     22: QSERV.Depart;
     23:Future.Wait( MeanDev(15,7));
     24: SERV.Release;
     25:Future.Wait( MeanDev(1.3*60,0.75*60));
     26: Quser2.Depart;
     27: PC.Release;
     28: Terminate;
{=Gen2} ::Gen2 29:Future.NewTransaction(exponential(30),0,Gen2,Gen2+1);
   30: QSERV.Queue;
   31: SERV.Seize;
   32: QSERV.Depart;
   33:Future.Wait(exponential(15));
   34: SERV.Release;
   35: Terminate;
{=Gen3} ::Gen3 36:Future.NewTransaction(22*8*60,0,Gen3,Gen3+1);
     37: Topoint(0,aclock,Quser1.t);
     38: Topoint(1,aclock,Quser2.t);
     39: Terminate(1);

 |R `Blocks    BLOCK Report
    Location     Total   Current
```

```
          1      3554         0
          2      3554         0
          3      3554         0
          4      3554         0
          5      3554         0
          6      3554         0
          7      3554         0
          8      3554         0
          9      3554         0
         10      3554         0
         11      3554         1
         12      3553         0
         13      3553         0
         14      3553         0
         15      5287         0
         16      5287         0
         17      5287         2
         18      5285         0
         19      5285         0
         20      5285         0
         21      5285         0
         22      5285         0
         23      5285         0
         24      5285         0
         25      5285         0
         26      5285         0
         27      5285         0
         28      5285         0
         29     42050         0
         30     42050         0
         31     42050         0
         32     42050         0
         33     42050         0
         34     42050         0
         35     42050         0
         36       120         0
         37       120         0
         38       120         0
         39       120         0
 |R `Values
 |R `CurrentList Report Current Transactions List  Count= 2
      Tran     Assem     Prior   Current      Next       TimeStamp       TimeEnter Inter
     51000     51000         0        17        18   1266993.82648   1266993.82648 FALSE
     51009     51009         0        17        18   1267062.79882   1267062.79882 FALSE
 |R    Facility   Entries   Current         Utility        AverTime      Tran   IntTran
Active Time    NextGo
          ` PC      8839         1         0.76294       109.37842     50994         0
1267200.00000         0
 |R    Facility   Entries   Current         Utility        AverTime      Tran   IntTran
Active Time    NextGo
          ` SERV     50889         0         0.61863        15.40456         0         0
1267200.00000         0
 |R       Queue   Entries   Current       AverQueue        AverTime AverTime( - Ze)       Max
Min      Zero     Active Time    NextGo
          ` QPC      8841         2         1.23463       176.96256       303.79146        12
0      3691   1267200.00000         0
 |R       Queue   Entries   Current       AverQueue        AverTime AverTime( - Ze)       Max
Min      Zero     Active Time    NextGo
         ` QSERV     50889         0         0.83254        20.73118        50.12062        14
0     29840   1267200.00000         0
 |R       Queue   Entries   Current       AverQueue        AverTime AverTime( - Ze)       Max
Min      Zero     Active Time    NextGo
        ` Quser1      3554         1         0.33498       119.44009    424490.07407         1
0      3553   1267200.00000         0
 |R       Queue   Entries   Current       AverQueue        AverTime AverTime( - Ze)       Max
Min      Zero     Active Time    NextGo
        ` Quser2      5287         2         1.16654       279.59801       359.58032        10
0      1176   1267200.00000         0
 |R        User   Entries   Current        AverUser        AverTime       Max       Min
Active Time    NextGo
```

```
        `Future    110743         5         4.70676        53.85806         6         0
1267200.00000         0
 |R `UserList Report  User List Count= 5
      Tran     Assem     Prior   Current      Next       TimeStamp       SortValue Inter
     50994     50994         0        11        12   1266961.92885   1267217.63341 FALSE
     51014     51014         0         0        29   1267226.47731   1267226.47731 FALSE
     51011     51011         0         0        15   1267338.97057   1267338.97057 FALSE
     51007     51007         0         0         1   1267568.98878   1267568.98878 FALSE
     51015     51015         0         0        36   1277760.00000   1277760.00000 FALSE
            Num/Name        Integer  |R `Future      Report IP LIST  Count= 8
I                 1           3555
I                 2          51007
I                15           5288
I                16          51011
I                29          42051
I                30          51014
I                36            121
I                37          51015
```

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 12 заданий, а средняя очередь – 1.23 задания. Сам персональный компьютер имел занятость 0.76 . Сервер имел максимальную очередь в 14 заданий, а среднюю – в 0.83 задания. Первый пользователь выполнил 3554 заданий, а второй - 5285 заданий. Среднее время выполнения задания пользователями составило 109.38 минуты. А среднее время выполнения задания сервером, составило 15.40 минуты. Пользователям удалось сразу сесть за персональный компьютер всего 3691 раз. В то время, как всего было выполнено 8839 заданий. И это далеко не вся информация, полученная в ходе моделирования такой простой модели!

Прокомментируем попутно блоки и процедуры вывода графических данных.

Всего в модели одновременно можно выводить до 16 графиков. Вывод выполняют блоки *:ToPoint(Num,X,Y);

Здесь Num – это номер графика, он может иметь значение от 0 до 15.
X,Y – координаты очередной точки графика.
Для того, чтобы графики выводились корректно при каждом запуске процесса моделирования, в процедуру Simulation, описывающую план моделирования, можно включить процедуру LineReset (Num); которая удаляет график с номером Num. Если параметр опущен, то удаляются все графики.

17 Работа с многоканальными устройствами

Для моделирования работы с многоканальными устройствами используют блоки
Stor.*:Enter (Units); и
Stor.*:Leave (Units);

Здесь Stor.- это имя устройства, а - Units - количество занимаемых или освобождаемых единиц его емкости.

Для описания таких устройств используется процедура
Init(Stor,’Stor’,Lim);

Здесь Stor.- это имя устройства, а - Lim - количество единиц его емкости.

Емкость должна быть целым числом. Она обычно определяет число заявок, которые могут одновременно обслуживаться в многоканальном устройстве.

Занятие многоканального устройства выполняется с помощью блока Stor.*:Enter (Units);

Освобождение выполняется с помощью блока

Stor.*:Leave (Units);

По умолчанию, Units 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.

Если в ходе моделирования нужно установить новую емкость многоканального устройства, то следует использовать блок

Stor.*:SetStorage (Units);

Здесь Units – новое количество каналов.

С целью управления доступностью многоканального устройства и, возможно, установления нового блока для принудительной отсылки заявки, претендующей на устройство, используется блок.

Stor.*:SetGoto (NumBl);

Смысл этого блока в точности такой же, как у одноименного блока для одноканальных устройств.

С многоканальными устройствами связаны следующие функции.

Stor.A - определяет среднее число занятых каналов.

Stor.AM - определяет среднее предельное число занятых каналов.

Stor.V - определяет временной интеграл устройства.

Stor.VM - определяет временной интеграл предельного числа занятых каналов.

Stor.T - определяет среднее время, проведенное заявкой в устройстве.

Stor.X - Определяет среднее время, проведенное заявкой в устройстве, для заявок, в устройстве задержавшихся.

Stor.C - определяет общее число занятий каналов.

Stor.L – определяет текущее число занятых каналов.

Stor.M - определяет максимальное число занятых каналов.

Stor.Z - определяет общее число занятий каналов, при которых заявки в устройстве не задерживались.

Stor.R - определяет текущее число свободных каналов.

Stor.U - определяет среднюю занятость устройства в расчете на 1 канал.

Stor.I - выдает минимальную занятость устройства.

Stor.Go - выдает значение поля NextGo, установленное блоком Stor.*:SetGoto.

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 400+- 100 сек. А затем они проходят через одноканальное устройство F, 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 350+- 200 сек, занимая по 2 канала. Затем они покидают модель. Промоделировать обслуживание 1000 заявок. Текст такой модели, мог бы иметь, например, следующий вид:

```
 {~vb} Var
 {~f}Future:TFuture;
{/F} F:Tfacility;
{/Nac} Nac:Tstorage;
{/QF} QF:Tqueue;
{/QNac} QNac:Tqueue;
 {~ve}
{~pfe}
 {~ib} procedure Initial;begin {`Initial |R }
 {~f}  init(Future,'Future');
SetStart(1000);
{/F}  init(F,'F');
{/Nac}  init(Nac,' Nac',5);
{/QF}  init(QF,' QF');
{/QNac}  init(QNac,' QNac');
{=G0} Future.NewWaitProc(G0,MeanDev(100,50),0,G0,G0+1);
{=G1} Future.NewWaitProc(G1,MeanDev(1000,100),0,G1,G1+1);
 {~ie}end;
 {~mtb}procedure ModelTxt;begin case ActiveBlock of
{=G0} ::G0 *:Future.NewWait(MeanDev(100,50),0,G0,G0+1);
    *: QNac.Queue ;
    *: Nac.Enter ;
    *: QNac.Depart ;
    *:Future.Wait(MeanDev(400,100));
    *: Nac.Leave ;
    *: QF.Queue ;
    *: F.Seize ;
    *: QF.Depart ;
    *:Future.Wait(MeanDev(90,20));
     *: F.Release ;
    *: Terminate(1) ;
{=G1} ::G1 *:Future.NewWait(MeanDev(1000,100),0,G1,G1+1);
    *: QNac.Queue ;
    *: Nac.Enter(2) ;
    *: QNac.Depart ;
   *:Future.Wait(MeanDev(350,200));
    *: Nac.Leave(2) ;
    *: Terminate ;
 {~mte} end;end;
 {~mb} procedure Simulation;  begin
  start(GetTerm);
```

{~me}end;
{~rb} procedure Report;begin
{~re} end;
{~cab} procedure CloseAllObj;begin
{~cae} end;

Результаты моделирования могут выглядеть, например, следующим образом:

```
Date Time Stamp= 15.11.2008 19:39:54   Ctrl Click |Name - HyperLink `R
J:\work\ObjectClient\Object No Sys\Future\GPSS-Future\book\tmp7.rtf
StartTime         0.00000 EndTime     101117.00593
 |Model |Blocks |Values |CurrentList |FutureList
 |UserList |SynchronizeList |InterruptList |PList
   Num      NameObject
     2        |Future
     3             |F
     4          | Nac
     5           | QF
     6         | QNac

 |R `Model
 {=G0} ::G0 1:Future.NewWait(MeanDev(100,50),0,G0,G0+1);
      2: QNac.Queue ;
      3: Nac.Enter ;
      4: QNac.Depart ;
      5:Future.Wait(MeanDev(400,100));
      6: Nac.Leave ;
      7: QF.Queue ;
      8: F.Seize ;
      9: QF.Depart ;
      10:Future.Wait(MeanDev(90,20));
      11: F.Release ;
      12: Terminate(1) ;
{=G1} ::G1 13:Future.NewWait(MeanDev(1000,100),0,G1,G1+1);
      14: QNac.Queue ;
      15: Nac.Enter(2) ;
      16: QNac.Depart ;
     17:Future.Wait(MeanDev(350,200));
      18: Nac.Leave(2) ;
      19: Terminate ;

 |R `Blocks    BLOCK Report
    Location     Total   Current
         1      1005         0
         2      1005         0
         3      1005         0
         4      1005         0
         5      1005         4
         6      1001         0
         7      1001         1
         8      1000         0
         9      1000         0
        10      1000         0
        11      1000         0
        12      1000         0
        13        99         0
        14        99         0
        15        99         0
        16        99         0
        17        99         0
        18        99         0
        19        99         0
 |R `Values
 |R `CurrentList Report Current Transactions List  Count= 1
      Tran     Assem  Priority   Current      Next       TimeStamp       TimeEnter Inter
      1101      1101         0         7         8    100671.75164    101090.72300 FALSE
 |R Facility = `F Entries = 1000 Current = 0 Utility =  0.88936 AverTime = 89.92934
    Tran = 0 IntTran = 0 Active Time = 101117.00593 NextGo = 0
```

```
 |R Storage = ` Nac Entries = 1203 Current = 4 Average =  4.64850 AverTime = 390.72556 Utility =
0.92970
    Max = 5 Min = 0 Tran = 1101 ACapac =  5.00000 Capacity = 5 Active Time = 101117.00593 NextGo
= 0
 |R Queue = ` QF Entries = 1001 Current = 1 AverQueue =  0.74810 AverTime = 75.56978
    AverTime(-Ze) = 152.81888 Max = 5 Min = 0 Zero = 506 Active Time = 101117.00593 NextGo = 0
 |R Queue = ` QNac Entries = 1104 Current = 0 AverQueue =  0.74637 AverTime = 68.36124
    AverTime(-Ze) = 219.39189 Max = 5 Min = 0 Zero = 760 Active Time = 101117.00593 NextGo = 0
 |R User = `Future Entries = 3210 Current = 6 AverUser =  6.53567 AverTime = 205.87756
    Max = 8 Min = 0 Active Time = 101117.00593 NextGo = 0
 |R `UserList Report  User List Count= 6
      Tran     Assem  Priority   Current      Next       TimeStamp       SortValue     Group
Inter
      1103      1103         0         5         6    100854.22199    101173.91511         0
FALSE
      1106      1106         0         0         1    101188.90629    101188.90629         0
FALSE
      1102      1102         0         5         6    100761.63560    101193.23894         0
FALSE
      1104      1104         0         5         6    100947.67815    101290.53632         0
FALSE
      1105      1105         0         5         6    101074.05370    101444.42068         0
FALSE
      1100      1100         0         0        13    101499.22239    101499.22239         0
FALSE
            Num/Name        Integer  |R `Future      Report IP LIST  Count= 4
I                  1           1006
I                  2           1106
I                 13            100
I                 14           1100
```

18 Арифметические, логические и строковые выражения в GPSS - Future

В GPSS - Future можно использовать различные выражения в качестве операндов.

Вид выражений фактически тот же, что и в Object Pascal. В выражении используют функции модели, 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. Числовые вычисления ведутся с двойной точностью.

Строковые константы представляют собой текст, заключенный в апострофы, например, ‘это строковая константа’ ‘It is string.’.

Логические константы могут быть только истина (True) и ложь (False).

Ниже приводятся операции и библиотечные функции, используемые в выражениях.

Арифметические выражения.

* - оператор арифметического умножения;

/ - оператор арифметического деления;

Div - оператор деления нацело, используется только для целых данных;

Mod - оператор деления по модулю, используется только для целых данных. Его результатом является остаток от деления нацело;

Shl - оператор сдвига целого числа влево. Он эквивалентен умножению числа на соответствующую степень двойки.

Shr - оператор сдвига целого числа вправо. Он эквивалентен делению числа на соответствующую степень двойки.

+ - оператор арифметического или строкового сложения;
- - оператор арифметического вычитания.
- знак минус;
Отношения.
Они могут соединять пару арифметических или строковых выражений.
Если это действительно выражения, то они должны быть в круглых скобках.
< - меньше;
<= - меньше или равно;
> - больше;
>= - больше или равно;
= - равно;
<> - не равно;
Логические выражения.
Они могут соединять логические данные, и, в частности, отношения.
NOT - логическое отрицание: True, если операнд False;
False, если операнд True;
AND - оператор логического умножения: True, если оба операнда True, False - в противном случае;
OR - оператор логического сложения: True, если хотя один из операндов True, False - в противном случае;
XOR - оператор сложения по модулю два: True, если операнды различны, False - в противном случае;
Элементарные функции.
Abs( ) - абсолютное значение операнда;
ArcTan( ) арктангенс операнда, результат выражен в радианах;
Cos( ) - косинус операнда в радианах;
Exp( ) - экспонента операнда;
Int( ) - целая часть числа, однако сам результат - вещественный;
Ln( ) - натуральный логарифм операнда;
Log10 ()- десятичный логарифм операнда;
Pi – предопределенная константа равная π.
Power (A,B) – возведение в вещественную степень B положительного вещественного числа A . Всегда может быть заменено выражением Exp(Ln(A)*B) .
Random – случайное число, равномерно распределенное в диапазоне от 0 до 1.
Random(Num) – случайное целое число, равномерно распределенное в диапазоне от 0 до Num-1.
RandSeed – целое число, представляет собой текущее значение встроенного генератора случайных чисел.
Round() – вещественное число округляется до целого.
Sin( ) - синус операнда в радианах;
Sqr( ) - квадрат операнда;
Sqrt() – корень квадратный из операнда.
Trunc() усеченное целое, дробная часть отбрасывается.
Tan( ) - тангенс операнда в радианах;
Функции преобразования.
StrToInt() – преобразует строку – в целое число.

IntToStr() – преобразует целое число – в строку.

StrToFloat() – преобразует строку – в вещественное число.

FloatToStr() – преобразует вещественное число – в строку.

Для строк можно выполнять операцию + , которая означает соединение строк, а также операции сравнения.

Порядок действий соответствует следующим приоритетам, расположенным от высшего - к низшему:

Логическое не. (Not)

Умножение, деление, остаток, логическое и. (* / Div Mod And Shl Shr)

Сложение и вычитание, логическое или. (+ - Or)

Отношения. (< <= > >= = <>)

Кроме выше приведенных функций, имеется большая группа встроенных функций.

19 Табулирование переменных

В Object GPSS можно табулировать значения различных параметров 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 ряд одинаковых интервалов и, кроме того, получим интервал от [-бесконечности до первой точки] и от [последней точки до +бесконечности] - эти последние интервалы называются интервалами переполнения.

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 имеет следующий вид. *:tab.tabulate(Value,Num);

Здесь tab – имя таблицы, Value табулируемое значение, а Num – добавляемая к интервалу величина. Она равна 1 по умолчанию.

Таблица описывается процедурой

init(Tab,’Tab’,Beg,Step,Num);

Здесь Beg - это первое граничное значение для интервалов;

Step- ширина интервала;

Num - это количество интервалов, включая два бесконечных.

Нижняя граница частотного класса всегда включается в предыдущий частотный класс.

С таблицами связаны следующие функции.

Tab.C - определяет число вхождений в таблицу с учетом кратности.

Tab.A - определяет среднее значение табулируемой величины с учетом кратности.

Tab.D - определяет среднеквадратичное отклонение табулируемой величины от средней.

Если вы хотите вывести гистограмму, соответствующей таблицы, то в процедуре Simulation нужно после Start поместить процедуру.

Show(Tab,Anum); или

GShow(Tab,Anum);

IShow(Tab,Anum);

Первая из них выводит таблицу как гистограмму, вторая выводит таблицу в виде графика, третья выводит таблицу в виде графика интегрального распределения. В этих процедурах Anum – номер ряда, используемого для вывода гистограммы или графика. Всего имеется 8 рядов (0.. 7) для гистограмм, и 16 рядов (0.. 15) для графиков.

Рассмотрим пример использования таблиц.

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

Пусть время обслуживания равно 100, и тоже распределено по экспоненциальному закону. Протабулируем полное время обслуживания заявок и текущую длину очереди.

Тогда модель работы такого устройства может иметь вид.

```
{~vb} Var
 {~f}Future:TFuture;
{/q2} q2:Tqueue;
{/fu} fu:Tfacility;
{/tbl} tbl:Ttable;
{/Tab0} Tab0:Ttable;
  {~ve}
{~pfe}
 {~ib} procedure Initial;begin {`Initial |R }
 {~f}  init(Future,'Future');
SetStart(1);
{/q2}  init(q2,'q2');
{/fu}  init(fu,'fu');
{/tbl}  init(tbl,' tbl',0,10,20);
{/Tab0} init(Tab0,' Tab0',0,5,100);
{=ng1} Future.NewWaitProc(ng1,exponential(110),0,ng1,ng1+1);
{=ng2} Future.NewWaitProc(ng2,100000,0,ng2,ng2+1);
  {~ie}end;
  {~mtb}procedure ModelTxt;begin case ActiveBlock of
{=ng1} ::ng1 *:Future.NewWait(exponential(110),0,ng1,ng1+1);
    *:q2.queue;
    *:fu.seize;
    *:q2.depart;
    *: mark ;
   *:Future.Wait(exponential(100));
    *:fu.release;
    *:tbl.tabulate(tranAge);
   *:tab0.tabulate(q2.L);
    *:terminate;
{=ng2} ::ng2 *:Future.NewWait(100000,0,ng2,ng2+1);
```

```
    *:terminate(1);
 {~mte} end;end;
 {~mb} procedure Simulation;  begin
{/sh}    start(GetTerm);
 {~me}end;
 {~rb} procedure Report;begin
 {~re} end;
 {~cab} procedure CloseAllObj;begin
 {~cae} end;
```

Результаты моделирования могут выглядеть, например, следующим образом:

```
Date Time Stamp= 15.11.2008 19:50:43   Ctrl Click |Name - HyperLink `R
J:\work\ObjectClient\Object No Sys\Future\GPSS-Future\book\tmp8.rtf
StartTime         0.00000 EndTime     100000.00000
 |Model |Blocks |Values |CurrentList |FutureList
 |UserList |SynchronizeList |InterruptList |PList
   Num      NameObject
     2        |Future
     3            |q2
     4            |fu
     5          | tbl
     6         | Tab0

 |R `Model
 {=ng1} ::ng1 1:Future.NewWait(exponential(110),0,ng1,ng1+1);
    2:q2.queue;
    3:fu.seize;
    4:q2.depart;
    5: mark ;
    6:Future.Wait(exponential(100));
    7:fu.release;
    8:tbl.tabulate(tranAge);
    9:tab0.tabulate(q2.L);
    10:terminate;
{=ng2} ::ng2 11:Future.NewWait(100000,0,ng2,ng2+1);
       12:terminate(1);

 |R `Blocks    BLOCK Report
    Location     Total   Current
         1       894         0
         2       894         3
         3       891         0
         4       891         0
         5       891         0
         6       891         1
         7       890         0
         8       890         0
         9       890         0
        10       890         0
        11         1         0
        12         1         0
 |R `Values
 |R `CurrentList Report Current Transactions List  Count= 3
      Tran     Assem  Priority   Current      Next       TimeStamp       TimeEnter Inter
       893       893         0         2         3     99898.42055     99898.42055 FALSE
       894       894         0         2         3     99947.84567     99947.84567 FALSE
       895       895         0         2         3     99963.69756     99963.69756 FALSE
 |R Facility = `fu Entries = 891 Current = 1 Utility =  0.87473 AverTime = 98.17429
    Tran = 892 IntTran = 0 Active Time = 100000.00000 NextGo = 0
 |R Queue = `q2 Entries = 894 Current = 3 AverQueue =  2.94301 AverTime = 329.19522
    AverTime(-Ze) = 426.52250 Max = 20 Min = 0 Zero = 204 Active Time = 100000.00000 NextGo = 0
 |R User = `Future Entries = 1788 Current = 3 AverUser =  2.41415 AverTime = 135.01969
    Max = 3 Min = 0 Active Time = 100000.00000 NextGo = 0
 |R `UserList Report  User List Count= 3
      Tran     Assem  Priority   Current      Next       TimeStamp       SortValue     Group
Inter
```

```
       896       896         0         0         1    100125.13783    100125.13783         0
FALSE
       892       892         0         6         7     99961.54624    100364.44183         0
FALSE
       897       897         0         0        11    200000.00000    200000.00000         0
FALSE
            Num/Name        Integer  |R `Future      Report IP LIST  Count= 4
I                  1            895
I                  2            896
I                 11              2
I                 12            897
 |R  Table = ` tbl  Entries = 890.00000 Mean = 98.24140 StdDev = 97.28701
          Range       Frequency        Integral
        0.00000         0.00000         0.00000
       10.00000        89.00000         0.10000
       20.00000        77.00000         0.18652
       30.00000        76.00000         0.27191
       40.00000        61.00000         0.34045
       50.00000        50.00000         0.39663
       60.00000        46.00000         0.44831
       70.00000        56.00000         0.51124
       80.00000        37.00000         0.55281
       90.00000        40.00000         0.59775
      100.00000        30.00000         0.63146
      110.00000        34.00000         0.66966
      120.00000        30.00000         0.70337
      130.00000        24.00000         0.73034
      140.00000        20.00000         0.75281
      150.00000        26.00000         0.78202
      160.00000        16.00000         0.80000
      170.00000        21.00000         0.82360
      180.00000        17.00000         0.84270
      190.00000       140.00000         1.00000
 |R  Table = ` Tab0  Entries = 890.00000 Mean =  3.60112 StdDev =  3.38717
          Range       Frequency        Integral
        0.00000       128.00000         0.14382
        5.00000       568.00000         0.78202
       10.00000       158.00000         0.95955
       15.00000        21.00000         0.98315
       20.00000        15.00000         1.00000
       25.00000         0.00000         1.00000
```

20 Функции в GPSS - Future

Для задания пользовательских функций с помощью точек или выражений, и некоторых других целей, используются числовые функции.

Вызов функции выполняется следующим образом.

Функция CFun(X, [,,],[,,]) -вычисляет функцию как непрерывную от заданного аргумента. В первых квадратных скобках следует перечислить через запятую значения аргумента X графика, а во вторых квадратных скобках следует перечислить через запятую значения функции Y графика. Можно без квадратных скобок указать имена динамических массивов с соответствующими значениями.

Функция DFun(X, [,,],[,,]) -вычисляет функцию как дискретную от заданного аргумента.

Функции ICFun(X, [,,],[,,]) и IDFun(X, [,,],[,,]) – аналогичны рассмотренным выше, но их значения округляются до целых.

Если вы хотите вывести числовую функцию как непрерывный или дискретный график, то в процедуре Simulation следует указать процедуру

CShow([,,],[,,],Anum);

DShow([,,],[,,],Anum);

Смысл ее параметров – такой же, как и у аналогичной процедуры, для таблицы, только список координат роднит ее с Cfun, DFun.

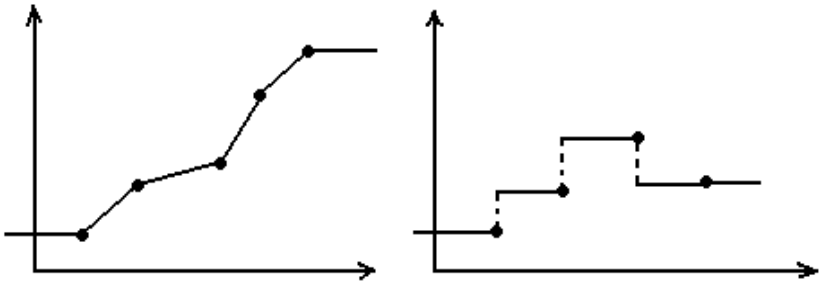
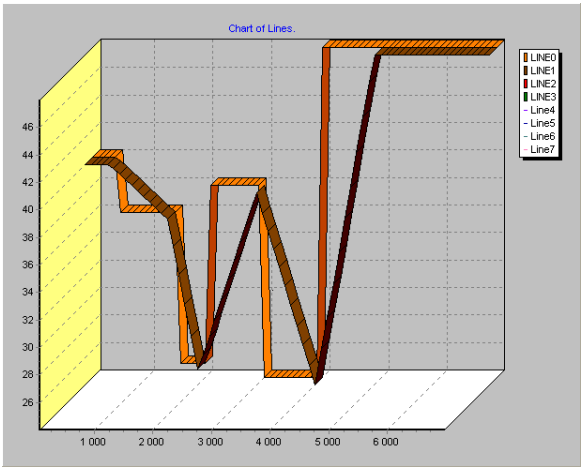


График непрерывной функции: График дискретной функции:

Пример описания и применения функции:

```
    {~vb} Var
 {~f}Future:TFuture;
  {~ve}
{~pfe}
 {~ib} procedure Initial;begin {`Initial |R }
 {~f}  init(Future,'Future');
  SetStart(1);
{=ng1} Future.NewWaitProc(ng1,MeanDev(100,50),0,ng1,ng1+1);
{=ng3} Future.NewWaitProc(ng3,7000,0,ng3,ng3+1);
  {~ie}end;
  {~mtb}procedure ModelTxt;begin case ActiveBlock of
{=ng1} ::ng1 *:Future.NewWait(MeanDev(100,50),0,ng1,ng1+1);
*:ToPoint(0,Clock,Dfun(Clock,[500,1500,2000,3000,4000,5000],[40,36,25,38,24,48]));
*:ToPoint(1,Clock,Cfun(Clock,[500,1500,2000,3000,4000,5000],[40,36,25,38,24,48]));
*:terminate;
{=ng3} ::ng3 *:Future.NewWait(7000,0,ng3,ng3+1);
    *:terminate(1);
  {~mte} end;end;
  {~mb} procedure Simulation;  begin
     start(GetTerm);
   {~me}end;
  {~rb} procedure Report;begin
  {~re} end;
  {~cab} procedure CloseAllObj;begin
  {~cae} end;
```

График дискретной и непрерывной функции, заданной одним и тем же набором точек можно видеть на вкладке Lines.



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: (Random).

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.

```
  {~vb} Var
 {~f} Future: TFuture;
{/q2} q2:Tqueue;
{/fu} fu:Tfacility;
{/tbl} tbl:Ttable;
  {~ve}
  {~pfe}
 {~ib} procedure Initial;begin {`Initial |R }
 {~f}  init(Future,'Future');
SetStart(1);
```

```
{/q2}  init(q2,' q2');
{/fu}  init(fu,' fu');
{/tbl}  init(tbl,' tbl',0,10,20);
{=ng1} Future.NewWaitProc(ng1,MeanDev(300,50),0,ng1,ng1+1);
{=ng3} Future.NewWaitProc(ng3,100000,0,ng3,ng3+1);
 {~ie}end;
 {~mtb}procedure ModelTxt;begin case ActiveBlock of
{=ng1} ::ng1 *:Future.NewWait(MeanDev(300,50),0,ng1,ng1+1);
   *:q2.queue;
   *:fu.seize;
   *:q2.depart;
   *: mark ;
  *:Future.Wait(DFun(random,[0.1,0.3,0.7,1],[100,80,120,60]));
   *:fu.release;
   *:tbl.tabulate(tranAge);
   *:terminate;
{=ng3} ::ng3 *:Future.NewWait(100000,0,ng3,ng3+1);
      *:terminate(1);
 {~mte} end;end;
 {~mb} procedure Simulation;  begin
   start(Getterm);
 show(tbl);
 {~me}end;
 {~rb} procedure Report;begin
 {~re} end;
 {~cab} procedure CloseAllObj;begin
 {~cae} end;
```

Итоговая статистика этой модели имеет вид.

```
Date Time Stamp= 13.12.2008 18:17:26   Ctrl Click |Name - HyperLink `R
E:\work\ObjectClient\Object No Sys\Future\GPSS-Future\book\tmp10.rtf
StartTime         0.00000 EndTime     100000.00000
 |Model |Blocks |Values |CurrentList |SynchronizeList |InterruptList
 |RPList |IPList |SPList |BPList
   Num      NameObject
     2        |Future
     3           | q2
     4           | fu
     5          | tbl
 |R `Model
 {=ng1} ::ng1 1:Future.NewWait(MeanDev(300,50),0,ng1,ng1+1);
    2:q2.queue;
    3:fu.seize;
    4:q2.depart;
    5: mark ;
   6:Future.Wait(DFun(random,[0.1,0.3,0.7,1],[100,80,120,60]));
    7:fu.release;
    8:tbl.tabulate(tranAge);
    9:terminate;
{=ng3} ::ng3 10:Future.NewWait(100000,0,ng3,ng3+1);
       11:terminate(1);

 |R `Blocks    BLOCK Report
    Location     Total   Current
        1        333        0
        2        333        0
```

```
         3       333         0
         4       333         0
         5       333         0
         6       333         0
         7       333         0
         8       333         0
         9       333         0
        10         1         0
        11         1         0
 |R `Values
 |R Facility = ` fu Entries = 333 Current = 0 Utility =  0.30400 AverTime = 91.29129
    Tran = 0 IntTran = 0 Active Time = 100000.00000 NextGo = 0
 |R Queue = ` q2 Entries = 333 Current = 0 AverQueue =  0.00000 AverTime =  0.00000
    AverTime(-Ze) =  0.00000 Max = 1 Min = 0 Zero = 333 Active Time = 100000.00000 NextGo = 0
 |R User = `Future Entries = 669 Current = 2 AverUser =  2.00000 AverTime = 298.95366
    Max = 3 Min = 0 Active Time = 100000.00000 NextGo = 0
 |R Report  User/Future List Count= 2
      Tran     Assem  Priority   Current      Next       TimeStamp       SortValue     Group
Inter
       335       335         0         0         1    100102.35792    100102.35792         0
FALSE
       336       336         0         0        10    200000.00000    200000.00000         0
FALSE
            Num/Name        Integer  |R `Future      Report IP LIST  Count= 4
I                 1            334
I                 2            335
I                10              2
I                11            336
 |R  Table = ` tbl  Entries = 333.00000 Mean = 91.29129 StdDev = 25.73877
           Range       Frequency        Integral
        50.00000         0.00000         0.00000
        60.00000       103.00000         0.30931
        70.00000         0.00000         0.30931
        80.00000        70.00000         0.51952
        90.00000         0.00000         0.51952
       100.00000        29.00000         0.60661
       110.00000         0.00000         0.60661
       120.00000       131.00000         1.00000
       130.00000         0.00000         1.00000
```

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

Для реализации возможностей других функций, которые применяются в GPSS, в GPSS - Future реализованы следующие встроенные функции.

Uniform(VMin,VMax) Выдает вещественную случайную величину распределенную по равномерному закону в интервале от VMin до VMax.

DUniform(VMin,VMax) Выдает целую случайную величину распределенную по равномерному закону в интервале от VMin до VMax.

Exponential(Mean) Выдает случайную величину распределенную по экспоненциальному закону со средним значением Mean.

Normal(Mean,Dev) Выдает случайную величину распределенную по нормальному закону. Ее среднее значение Mean и среднеквадратичное отклонение Dev.

RByBool([],[]) Содержит список логических выражений и список вещественных выражений. Оба списка указываются в [] через запятую. Для первого истинного выражения, вычисляется соответствующее выражение из второго списка.

IByBool([],[]) Содержит список логических выражений и список целых выражений. Оба списка указываются в [] через запятую. Для первого истинного выражения, вычисляется соответствующее выражение из второго списка.

SByBool([],[]) Содержит список логических выражений и список строковых выражений. Оба списка указываются в [] через запятую. Для первого истинного выражения, вычисляется соответствующее выражение из второго списка.

RByNum(Num,[]) Содержит целое выражение и список вещественных выражений. Список указываются в [] через запятую. Вычисляется выражение номер Num из списка.

IByNum(Num,[]) Содержит целое выражение и список целых выражений. Список указываются в [] через запятую. Вычисляется выражение номер Num из списка.

SByNum(Num,[]) Содержит целое выражение и список строковых выражений. Список указываются в [] через запятую. Вычисляется выражение номер Num из списка.

BByNum(Num,[]) Содержит целое выражение и список логических выражений. Список указываются в [] через запятую. Вычисляется выражение номер Num из списка.

Рассмотрим пример применения таких функций.

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. В остальных случаях время обслуживания равно 100 - <длина очереди>*15.

Модель работы такого устройства может иметь вид.

```
 {~vb} Var
{~f}Future:TFuture;
{/q2} q2:Tqueue;
{/fu} fu:Tfacility;
{/tbl} tbl:Ttable;
{/Tab0} Tab0:Ttable;
 {~ve}
{~pfe}
{~ib} procedure Initial;begin {`Initial |R }
{~f}  init(Future,'Future');
SetStart(1);
{/q2}  init(q2,' q2');
{/fu}  init(fu,' fu');
{/tbl}  init(tbl,' tbl',0,10,20);
{/Tab0}  init(Tab0,' Tab0',0,1,100);
{=ng3} Future.NewWaitProc(ng3,100000,0,ng3,ng3+1);
{=ng1} Future.NewWaitProc(ng1,exponential(100),0,ng1,ng1+1);
 {~ie}end;
 {~mtb}procedure ModelTxt;begin case ActiveBlock of
{=ng1} :ng1 *:Future.NewWait(exponential(100),0,ng1,ng1+1);
   *:q2.queue;
   *:fu.seize;
```

```
  *:q2.depart;
  *: mark ;
 *:Future.Wait(RByBool([q2.L<=2,q2.L<=7,q2.L<=11,True],
 [120-q2.L*5,110-q2.L*8,100-q2.L*6,100-q2.L*15]));
  *:fu.release;
  *:tbl.tabulate(TranAge);
  *:tab0.tabulate(q2.L);
  *:terminate;
{=ng3} ::ng3 *:Future.NewWait(100000,0,ng3,ng3+1);
    *:terminate(1);
 {~mte} end;end;
 {~mb} procedure Simulation;  begin
{/sh}     start(GetTerm);
 {~me}end;
 {~rb} procedure Report;begin
 {~re} end;
 {~cab} procedure CloseAllObj;begin
 {~cae} end;
```

Выходная статистика модели имеет вид.

```
Date Time Stamp= 13.12.2008 18:27:10   Ctrl Click |Name - HyperLink `R
E:\work\ObjectClient\Object No Sys\Future\GPSS-Future\book\tmp11.rtf
StartTime          0.00000 EndTime     100000.00000
 |Model |Blocks |Values |CurrentList |SynchronizeList |InterruptList
 |RPList |IPList |SPList |BPList
   Num       NameObject
     2         |Future
     3            | q2
     4            | fu
     5           | tbl
     6          | Tab0
 |R `Model
 {=ng1} ::ng1 1:Future.NewWait(exponential(100),0,ng1,ng1+1);
    2:q2.queue;
    3:fu.seize;
    4:q2.depart;
    5: mark ;
   6:Future.Wait(RByBool([q2.L<=2,q2.L<=7,q2.L<=11,True],
   [120-q2.L*5,110-q2.L*8,100-q2.L*6,100-q2.L*15]));
    7:fu.release;
    8:tbl.tabulate(TranAge);
    9:tab0.tabulate(q2.L);
    10:terminate;
{=ng3} ::ng3 11:Future.NewWait(100000,0,ng3,ng3+1);
       12:terminate(1);

 |R `Blocks     BLOCK Report
    Location      Total    Current
         1        970         0
         2        970         1
         3        969         0
         4        969         0
         5        969         0
         6        969         1
         7        968         0
         8        968         0
         9        968         0
        10        968         0
        11          1         0
        12          1         0
 |R `Values
```

```
 |R `CurrentList Report Current Transactions List  Count= 1
      Tran     Assem  Priority   Current      Next       TimeStamp       TimeEnter Inter
       971       971         0         2         3     99973.11112     99973.11112 FALSE
 |R Facility = ` fu Entries = 969 Current = 1 Utility =  0.94704 AverTime = 97.73390
    Tran = 970 IntTran = 0 Active Time = 100000.00000 NextGo = 0
 |R Queue = ` q2 Entries = 970 Current = 1 AverQueue =  2.40592 AverTime = 248.03331
    AverTime(-Ze) = 288.47999 Max = 10 Min = 0 Zero = 136 Active Time = 100000.00000 NextGo = 0
 |R User = `Future Entries = 1942 Current = 3 AverUser =  2.33354 AverTime = 120.16184
    Max = 3 Min = 0 Active Time = 100000.00000 NextGo = 0
 |R Report  User/Future List Count= 3
      Tran     Assem  Priority   Current      Next       TimeStamp       SortValue     Group
Inter
       970       970         0         6         7     99901.85225    100021.85225         0
FALSE
       972       972         0         0         1    100073.99595    100073.99595         0
FALSE
       973       973         0         0        11    200000.00000    200000.00000         0
FALSE
             Num/Name        Integer  |R `Future      Report IP LIST  Count= 4
I                  1            971
I                  2            972
I                 11              2
I                 12            973
 |R  Table = ` tbl  Entries = 968.00000 Mean = 97.73347 StdDev = 19.92948
          Range       Frequency        Integral
       40.00000         0.00000         0.00000
       50.00000         1.00000         0.00103
       60.00000        22.00000         0.02376
       70.00000       108.00000         0.13533
       80.00000       118.00000         0.25723
       90.00000       198.00000         0.46178
      100.00000         0.00000         0.46178
      110.00000       195.00000         0.66322
      120.00000       326.00000         1.00000
      130.00000         0.00000         1.00000
 |R  Table = ` Tab0  Entries = 968.00000 Mean =  3.35434 StdDev =  1.93682
          Range       Frequency        Integral
        0.00000        60.00000         0.06198
        1.00000       128.00000         0.19421
        2.00000       138.00000         0.33678
        3.00000       195.00000         0.53822
        4.00000       198.00000         0.74277
        5.00000       118.00000         0.86467
        6.00000        72.00000         0.93905
        7.00000        36.00000         0.97624
        8.00000        16.00000         0.99277
        9.00000         6.00000         0.99897
       10.00000         1.00000         1.00000
       11.00000         0.00000         1.00000
```

21 Функции и процедуры работы со строками

Эти функции могут использоваться для манипуляций со строками при вводе – выводе данных, а также для формирования значений X и P – параметров.

Кроме ранее рассмотренных, имеется следующий набор функций и процедур для работы со строками:

Функция

Pos(Substr, S)

Она ищет подстроку Substr в строке S. Когда найдена нужная подстрока, то позиция начала совпадения считается результатом, иначе результат равен 0.

Функция

Copy(S,Index,Count:)

Выделяет из строки S, начиная с позиции Index заданное число символов. Полученная строка считается результатом.

Процедура

Delete(S,Index,Count);

В строке S удаляется Count символов, начиная с символа с укзанным номером Index. Полученная строка заменяет S.

Процедура

Insert(Source,S, Index);

В строку S вставляется строка Source, начиная с позиции index. Полученная строка заменяет S.

Функция

Length(S) – определяет длину строки S.

Функция

StringReplace(S,OldPattern,NewPattern,Flags)

Возвращает новую строку, которая получена следующим образом. В строке S заменяется многократно подстрока OldPattern на подстроку NewPattern. Флаги, записанные в квадратных скобках управляют заменой.

Если есть флаг rfIgnoreCase, то при замене не учитывается регистр символов, то есть не различаются большие и малые буквы, иначе большие и малые буквы считаются различными.

Если есть флаг rfReplaceAll, то заменяются все подстроки, иначе только первая попавшаяся подстрока.

22 Блоки присваивания в GPSS

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

Для задания приоритета заявки используется следующий блок.

*:Priority (Num,ATran);

Здесь Num – это новый приоритет активной заявки. Если нужно установить новый приоритет, какой – либо другой заявке, то номер этой заявки указывается вторым параметром.

После присваивания 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.

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 – 170 мс., обслуживания – 100мс. Заявки второго типа обслуживаются с приоритетом.

Промоделировать работу системы в течение 1000000 мс.

Текст такой модели может иметь вид:

```
  {~vb} Var
 {~f}Future:TFuture;
{/ust0} ust0:Tqueue;
{/ust1} ust1:Tqueue;
{/ust2} ust2:Tqueue;
{/ust} ust:Tstorage;
  {~ve}
Const
Int1=145;
Wint1=25;
Int2=170;
Wint2=100;
{~pfe}
 {~ib} procedure Initial;begin {`Initial |R }
 {~f}  init(Future,'Future');
    SetStart(1);
{/ust0}  init(ust0,' ust0');
{/ust1}  init(ust1,' ust1');
{/ust2}  init(ust2,' ust2');
{/ust}  init(ust,' ust',2);
{=G} Future.NewWaitProc(G,Exponential(int1),0,G,G+1);
{=G0} Future.NewWaitProc(G0,Exponential(int2),0,G0,G0+1);
{=G1} Future.NewWaitProc(G1,1000000,0,G1,G1+1);
  {~ie}end;
  {~mtb}procedure ModelTxt;begin case ActiveBlock of
{=G} ::G *:Future.NewWait(Exponential(int1),0,G,G+1);
  *: ust0.queue ;
  *: ust.Enter ;
  *: ust0.depart ;
  *: ust1.queue ;
  *:Future.Wait(Exponential(Wint1));
  *: ust1.depart ;
  *: ust.Leave ;
  *: terminate ;
{=G0} ::G0 *:Future.NewWait(Exponential(int2),0,G0,G0+1);
  *: priority(1) ;
  *: ust0.queue ;
  *: ust.Enter(2) ;
  *: ust0.depart ;
  *: ust2.queue ;
  *:Future.Wait(Exponential(Wint2));
  *: ust2.depart ;
```

```
  *: ust.Leave(2) ;
  *: terminate ;
{=G1} ::G1 *:Future.NewWait(1000000,0,G1,G1+1);
    *: terminate(1) ;
  {~mte} end;end;
  {~mb} procedure Simulation;  begin
   start(GetTerm);
  {~me}end;
  {~rb} procedure Report;begin
  {~re} end;
  {~cab} procedure CloseAllObj;begin
  {~cae} end;
```

Здесь для сбора более подробной статистики, используются формальные очереди ust1 и ust2. Итоговая статистика модели имеет вид.

```
Date Time Stamp= 13.12.2008 20:48:06   Ctrl Click |Name - HyperLink `R
E:\work\ObjectClient\Object No Sys\Future\GPSS-Future\book\tmp12.rtf
StartTime          0.00000 EndTime    1000000.00000
 |Model |Blocks |Values |CurrentList |SynchronizeList |InterruptList
 |RPList |IPList |SPList |BPList
   Num      NameObject
     2        |Future
     3         | ust0
     4         | ust1
     5         | ust2
     6          | ust
 |R `Model
 {=G} ::G 1:Future.NewWait(Exponential(int1),0,G,G+1);
   2: ust0.queue ;
   3: ust.Enter ;
   4: ust0.depart ;
   5: ust1.queue ;
   6:Future.Wait(Exponential(Wint1));
   7: ust1.depart ;
   8: ust.Leave ;
   9: terminate ;
{=G0} ::G0 10:Future.NewWait(Exponential(int2),0,G0,G0+1);
   11: priority(1) ;
   12: ust0.queue ;
   13: ust.Enter(2) ;
   14: ust0.depart ;
   15: ust2.queue ;
   16:Future.Wait(Exponential(Wint2));
   17: ust2.depart ;
   18: ust.Leave(2) ;
   19: terminate ;
{=G1} ::G1 20:Future.NewWait(1000000,0,G1,G1+1);
      21: terminate(1) ;

 |R `Blocks    BLOCK Report
    Location      Total   Current
         1       6867         0
         2       6867         0
         3       6867         0
         4       6867         0
         5       6867         0
         6       6867         0
         7       6867         0
         8       6867         0
         9       6867         0
        10       5717         0
        11       5717         0
        12       5717         0
        13       5717         0
```

```
        14      5717         0
        15      5717         0
        16      5717         1
        17      5716         0
        18      5716         0
        19      5716         0
        20         1         0
        21         1         0
 |R `Values
 |R Storage = ` ust Entries = 18301 Current = 2 Average =  1.29278 AverTime = 70.63984 Utility =
0.64639
    Max = 2 Min = 0 Tran = 12582 ACapac =  2.00000 Capacity = 2 Active Time = 1000000.00000
NextGo = 0
 |R Queue = ` ust0 Entries = 12584 Current = 0 AverQueue =  2.99732 AverTime = 238.18463
    AverTime(-Ze) = 444.57363 Max = 51 Min = 0 Zero = 5842 Active Time = 1000000.00000 NextGo =
0
 |R Queue = ` ust1 Entries = 6867 Current = 0 AverQueue =  0.17222 AverTime = 25.07962
    AverTime(-Ze) = 60.79130 Max = 2 Min = 0 Zero = 4034 Active Time = 1000000.00000 NextGo = 0
 |R Queue = ` ust2 Entries = 5717 Current = 1 AverQueue =  0.56028 AverTime = 98.00228
    AverTime(-Ze) = 560279.01542 Max = 1 Min = 0 Zero = 5716 Active Time = 1000000.00000 NextGo
= 0
 |R User = `Future Entries = 25172 Current = 4 AverUser =  3.38048 AverTime = 134.29521
    Max = 5 Min = 0 Active Time = 1000000.00000 NextGo = 0
 |R Report  User/Future List Count= 4
      Tran     Assem  Priority   Current      Next       TimeStamp       SortValue     Group
Inter
     12586     12586         0         0         1   1000023.94872   1000023.94872         0
FALSE
     12582     12582         1        16        17    999930.39735   1000130.67221         0
FALSE
     12587     12587         0         0        10   1000171.03908   1000171.03908         0
FALSE
     12588     12588         0         0        20   2000000.00000   2000000.00000         0
FALSE
            Num/Name        Integer  |R `Future      Report IP LIST  Count= 6
I                 1           6868
I                 2          12586
I                10           5718
I                11          12587
I                20              2
I                21          12588
```

В GPSS имеются параметры заявки, которые, как правило, доступны только самой заявке (ее P – параметры). Для присваивания значений этим параметрам (RP(Num,ATran), IP(Num,ATran), SP(Num,ATran), BP(Num,ATran)) используются блоки

*:RPlet([Nums],[Values],ATran);

Это блок присваивания списка вещественных значений Values - RP -параметрам с номерами из списка Nums. По умолчанию, значения присваиваются активной заявке. Иначе, заявке с указанным номером ATran .

*:IPlet([Nums],[Values],ATran);

Это блок присваивания списка целых значений Values- IP –параметрам с номерами из списка Nums. По умолчанию, значения присваиваются активной заявке. Иначе, заявке с указанным номером ATran .

*:SPlet([Nums],[Values],ATran);

Это блок присваивания списка строковых значений Values - SP -параметрам с номерами из Nums. По умолчанию, значения присваивается активной заявке. Иначе, заявке с указанным номером ATran . Учтите, что строки сохраняются в SP – параметрах без хвостовых пробелов.

*:BPlet([Nums],[Values],ATran);

Это блок присваивания списка булевских значений Values BP -параметрам с номерами из Nums. По умолчанию, значения присваиваются активной заявке. Иначе, заявке с указанным номером ATran .

*:RPAdd(Nums,Values,ATran);

Это блок увеличивает на значения Values - RP -параметры с номерами Nums. По умолчанию, значения увеличиваются для активной заявки. Иначе, для заявки с указанным номером ATran .

*:IPAdd([Nums],[Values],ATran);

Это блок увеличивает на значения Values - IP -параметры с номерами Nums. По умолчанию, значения увеличиваются для активной заявки. Иначе, для заявки с указанным номером ATran .

Для работы с P – параметрами вместо номера можно использовать переменную, которой присвоено значение функции Init(st) , где St – строка для идентификации параметра. Вместо первого списка можно указывать начальный номер параметров.

Пример блока.

*:IPLET(5,[och.L+30])

Означает, что нужно вычислить величину och.L+30 и записать результат в IP(5).

Примечание: не существующий RP,IP - параметр в выражениях имеет нулевое значение, SP – параметр имеет значение пустой строки, BP - параметр имеет значение False.

Общая форма обращения к параметрам заявок имеет вид.

RP(Num,ATran), IP(Num,ATran), SP(Num,ATran), BP(Num,ATran) .

Здесь Num – это номер параметра, а ATran – номер заявки. То есть, имеется возможность получить значение не только активной заявки, но и любой другой. По умолчанию, вы получаете значение параметра активной заявки.

Примечание: следует избегать использования параметров с номером ' -1'.

Кроме параметров заявки, в GPSS имеются параметры системы, так называемые ячейки. На самом деле, эти параметры принадлежат не существующим заявкам с отрицательным номером.

Для работы с ними можно использовать объект типа Tparam, или любой другой объект. Объект позволяет иметь набор именованных P – параметров. Для работы с набором определена единственная функция N, которая численно равна –NumObj, то есть номеру объекта с минусом.

Пусть вы создали переменную SYS класса Tparam. Тогда для присваивания значений ячейкам (RP(Num, SYS.N), IP(Num, SYS.N), SP(Num, SYS.N), BP(Num, SYS.N)) используются блоки

*:RPlet(Nums,Values, SYS.N);

*:IPlet(Nums,Values, SYS.N);

*:SPlet(Nums,Values, SYS.N);

*:BPlet(Nums,Values, SYS.N);

Это блоки присваивания списка значений Values - P -параметрам с номерами из Nums.

Все блоки, процедуры, и функции, определенные для P- параметров заявок, доступны и для ячеек, если вместо номера заявки указать функцию, в данном случае, SYS.N, или её численное значение.

Например, пусть описана переменная ISYS :Integer; тогда после инициализации переменной SYS класса Tparam, можно написать

ISYS := SYS.N; и впоследствии вместо SYS.N писать везде ISYS.

Рассмотрим примеры использования блоков PLET .

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 1000000 мс. Определить суммарное время задержки заявок в очереди, к каждому устройству.

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

Тогда можно получить следующую модель.

```
  Const
num=5;
   {~vb} Var
 {~f}Future:TFuture;
{/fac0} fac0:array[1..num] of Tfacility;
{/que0} que0:array[1..num] of Tqueue;
{\Sys} Sys:Tparam;
  {~ve}
{~pfe}
  {~ib} procedure Initial; var i:Integer; begin
 {~f}  init(Future,'Future');
   setstart(1);
{/fac0}  for i:=1 to num do init(fac0[i] , 'fac0'+inttostr(i));
{/que0}  for i:=1 to num do init(que0[i] ,' que0'+inttostr(i));
{\Sys} Init(Sys,'Sys');
{=Gen0} Future.NewWaitProc(Gen0,Exponential(100),0,Gen0,Gen0+1);
{=Gen} Future.NewWaitProc(Gen,1000000,0,Gen,Gen+1);
  {~ie}end;
  {~mtb}procedure ModelTxt;begin case ActiveBlock of
{=Gen0} ::Gen0 *:Future.NewWait(Exponential(100),0,Gen0,Gen0+1);
   *: IPlet(1,[duniform(1,num)]);
   *: que0[IP(1)].Queue;
   *: fac0[IP(1)].Seize;
  *:Future.Wait(MeanDev(490,90));
   *: que0[IP(1)].Depart;
   *: fac0[IP(1)].Release;
   *: RPlet(IP(1),[RP(IP(1), sys.N)+TranAge] , sys.N) ;
   *: Terminate();
{=Gen} ::Gen *:Future.NewWait(1000000,0,Gen,Gen+1);
*: Terminate(1);
  {~mte} end;end;
  {~mb} procedure Simulation;  begin
   start(GetTerm);
  {~me}end;
  {~rb} procedure Report;  begin
  {~re} end;
```

{~cab} procedure CloseAllObj;begin
{~cae} end;

Обратите внимание, что очереди и устройства описаны как массивы, с номерами элементов от 1 до Num. В этой связи, в процедуре Initial, вызовы процедур для очередей и устройств выполнены в циклах. Обратите также внимание на то, что индексы объектов записываются в квадратных скобках.

Техника построения модели остается прежней. Нужно вначале вставить описания всех нужных объектов, а затем учесть, что часть из них является массивами. Делается это в соответствии с образцом.

Было
{/fac0} fac0: Tfacility;
Исправили и получили.
{/fac0} fac0:array[1..num] of Tfacility;

Аналогичным образом исправляются строки с вызовами процедур объекта

Было
{/fac0} init(fac0 , 'fac0');
Исправили и получили.
{/fac0} for i:=1 to num do init(fac0[i] , 'fac0'+inttostr(i));

Кроме этого, после заголовка процедуры Initial, добавлено описание переменной i, то есть, добавлен текст.

var i:integer;

В данном случае, удобнее задавать число элементов массива с помощью константы Num, а поэтому добавлен текст
num=5; в разделе Const.

Итоговая статистика этой модели будет иметь вид.

```
Date Time Stamp= 13.12.2008 20:58:28   Ctrl Click |Name - HyperLink `R
E:\work\ObjectClient\Object No Sys\Future\GPSS-Future\book\tmp13.rtf
StartTime         0.00000 EndTime    1000000.00000
 |Model |Blocks |Values |CurrentList |SynchronizeList |InterruptList
 |RPList |IPList |SPList |BPList
   Num      NameObject
     2        |Future
     3         |fac01
     4         |fac02
     5         |fac03
     6         |fac04
     7         |fac05
     8        | que01
     9        | que02
    10        | que03
    11        | que04
    12        | que05
    13           |Sys
 |R `Model
 {=Gen0} ::Gen0 1:Future.NewWait(Exponential(100),0,Gen0,Gen0+1);
   2: IPlet(1,[duniform(1,num)]);
   3: que0[IP(1)].Queue;
   4: fac0[IP(1)].Seize;
  5:Future.Wait(MeanDev(490,90));
   6: que0[IP(1)].Depart;
   7: fac0[IP(1)].Release;
   8: RPlet(IP(1),[RP(IP(1), sys.N)+TranAge] , sys.N) ;
   9: Terminate();
{=Gen} ::Gen 10:Future.NewWait(1000000,0,Gen,Gen+1);
11: Terminate(1);

 |R `Blocks    BLOCK Report
```

```
    Location      Total   Current
         1      10000         0
         2      10000         0
         3      10000        44
         4       9956         0
         5       9956         5
         6       9951         0
         7       9951         0
         8       9951         0
         9       9951         0
        10          1         0
        11          1         0
 |R `Values
 |R Facility = `fac01 Entries = 2036 Current = 1 Utility =  0.99894 AverTime = 490.63764
    Tran = 9953 IntTran = 0 Active Time = 1000000.00000 NextGo = 0
 |R Facility = `fac02 Entries = 1989 Current = 1 Utility =  0.97272 AverTime = 489.04827
    Tran = 9992 IntTran = 0 Active Time = 1000000.00000 NextGo = 0
 |R Facility = `fac03 Entries = 1981 Current = 1 Utility =  0.96999 AverTime = 489.64916
    Tran = 9974 IntTran = 0 Active Time = 1000000.00000 NextGo = 0
 |R Facility = `fac04 Entries = 2014 Current = 1 Utility =  0.98147 AverTime = 487.32134
    Tran = 9990 IntTran = 0 Active Time = 1000000.00000 NextGo = 0
 |R Facility = `fac05 Entries = 1936 Current = 1 Utility =  0.95022 AverTime = 490.81462
    Tran = 9871 IntTran = 0 Active Time = 1000000.00000 NextGo = 0
 |R Queue = ` que01 Entries = 2046 Current = 11 AverQueue = 24.17050 AverTime = 11813.53781
    AverTime(-Ze) = 11813.53781 Max = 59 Min = 0 Zero = 0 Active Time = 1000000.00000 NextGo = 0
 |R Queue = ` que02 Entries = 1992 Current = 4 AverQueue = 11.12727 AverTime = 5585.97713
    AverTime(-Ze) = 5639.77011 Max = 31 Min = 0 Zero = 19 Active Time = 1000000.00000 NextGo = 0
 |R Queue = ` que03 Entries = 1985 Current = 5 AverQueue =  9.17750 AverTime = 4623.42488
    AverTime(-Ze) = 4691.97259 Max = 24 Min = 0 Zero = 29 Active Time = 1000000.00000 NextGo = 0
 |R Queue = ` que04 Entries = 2016 Current = 3 AverQueue = 17.12462 AverTime = 8494.35736
    AverTime(-Ze) = 8558.03321 Max = 39 Min = 0 Zero = 15 Active Time = 1000000.00000 NextGo = 0
 |R Queue = ` que05 Entries = 1961 Current = 26 AverQueue =  8.93088 AverTime = 4554.24676
    AverTime(-Ze) = 4666.08040 Max = 29 Min = 0 Zero = 47 Active Time = 1000000.00000 NextGo = 0
 |R User = `Future Entries = 19959 Current = 7 AverUser =  6.34583 AverTime = 317.94309
    Max = 7 Min = 0 Active Time = 1000000.00000 NextGo = 0
 |R Report  User/Future List Count= 7
      Tran     Assem  Priority   Current      Next       TimeStamp       SortValue     Group
Inter
     10002     10002         0         0         1   1000006.55145   1000006.55145         0
FALSE
      9974      9974         0         5         6    997461.55557   1000067.55048         0
FALSE
      9871      9871         0         5         6    987119.33439   1000155.63974         0
FALSE
      9992      9992         0         5         6    998714.73302   1000250.71285         0
FALSE
      9953      9953         0         5         6    995284.28668   1000397.94382         0
FALSE
      9990      9990         0         5         6    998542.54693   1000548.01145         0
FALSE
     10003     10003         0         0        10   2000000.00000   2000000.00000         0
FALSE
            Num/Name        Integer  |R `Future      Report IP LIST  Count= 4
I                  1          10001
I                  2          10002
I                 10              2
I                 11          10003
            Num/Name         Double  |R `Sys       Report RP LIST  Count= 5
R                  1  24143971.32798
R                  2  11122763.07464
R                  3  9168619.41881
R                  4  17121307.11490
R                  5  8751474.47908
```

Здесь опущена статистика по списку текущих событий, и по спискам P – параметров.

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

Тогда модель приобретет следующий вид.

```
  Const
num=5;
   {~vb} Var
 {~f}Future:TFuture;
{/fac0} fac0:array[1..num] of Tfacility;
{/que0} que0:array[1..num] of Tqueue;
{/BSigm} BSigm:Tparam;
{/BSum} BSum:Tparam;
{/BSsqr} BSsqr:Tparam;
{\Sys} Sys:Tparam;
  {~ve}
  {~pfe}
  {~ib} procedure Initial; var i:integer; begin
 init(Future,'Future');
   setstart(1);
{/fac0}  for i:=1 to num do init(fac0[i], 'fac0'+inttostr(i));
{/que0}  for i:=1 to num do init(que0[i], 'que0'+inttostr(i));
{/BSigm}  init(BSigm,' BSigm');
{/BSum}  init(BSum,' BSum');
{/BSsqr}  init(BSsqr,' BSsqr');
{\Sys} Init(Sys,'Sys');
{=Gen0} Future.NewWaitProc(Gen0,Exponential(100),0,Gen0,Gen0+1);
{=Gen} Future.NewWaitProc(Gen,1000000,0,Gen,Gen+1);
  {~ie}end;
  {~mtb}procedure ModelTxt;begin case ActiveBlock of
{=Gen0} ::Gen0 *:Future.NewWait(Exponential(100),0,Gen0,Gen0+1);
   *: IPlet(1,[duniform(1,num)]);
   *: que0[IP(1)].Queue;
   *: fac0[IP(1)].Seize;
   *: RPlet(IP(1),[RP(IP(1) ,bssqr.N)+sqr(TranAge)],bssqr.N) ;
   *: RPlet(IP(1),[RP(IP(1) , bsum.N)+1], bsum.N) ;
   *: RPlet(IP(1),[sqrt(RP(IP(1) ,bssqr.N)/RP(IP(1),bsum.N))], bsigm.N) ;
   *:Future.Wait(MeanDev(490,90));
   *: que0[IP(1)].Depart;
   *: fac0[IP(1)].Release;
   *: RPlet(IP(1),[RP(IP(1), sys.N)+TranAge], sys.N) ;
   *: Terminate();
{=Gen} ::Gen *:Future.NewWait(1000000,0,Gen,Gen+1);
 *: Terminate(1);
  {~mte} end;end;
  {~mb} procedure Simulation;  begin
   start(GetTerm);
  {~me}end;
  {~rb} procedure Report;  begin
  {~re} end;
  {~cab} procedure CloseAllObj;begin
```

{~cae} end;

Итоговая статистика по модели будет иметь вид:

```
Date Time Stamp= 18.12.2008 10:31:02   Ctrl Click |Name - HyperLink `R
E:\work\ObjectClient\Object No Sys\Future\GPSS-Future\book\tmp14.rtf
StartTime          0.00000 EndTime    1000000.00000
 |Model |Blocks |Values |CurrentList |SynchronizeList |InterruptList
 |RPList |IPList |SPList |BPList
   Num      NameObject
     2        |Future
     3         |fac01
     4         |fac02
     5         |fac03
     6         |fac04
     7         |fac05
     8         |que01
     9         |que02
    10         |que03
    11         |que04
    12         |que05
    13        | BSigm
    14         | BSum
    15        | BSsqr
    16           |Sys
 |R `Model
 {=Gen0} ::Gen0 1:Future.NewWait(Exponential(100),0,Gen0,Gen0+1);
   2: IPlet(1,[duniform(1,num)]);
   3: que0[IP(1)].Queue;
   4: fac0[IP(1)].Seize;
   5: RPlet(IP(1),[RP(IP(1) ,bssqr.N)+sqr(TranAge)],bssqr.N) ;
   6: RPlet(IP(1),[RP(IP(1) , bsum.N)+1], bsum.N) ;
   7: RPlet(IP(1),[sqrt(RP(IP(1) ,bssqr.N)/RP(IP(1),bsum.N))], bsigm.N) ;
   8:Future.Wait(MeanDev(490,90));
   9: que0[IP(1)].Depart;
   10: fac0[IP(1)].Release;
   11: RPlet(IP(1),[RP(IP(1), sys.N)+TranAge], sys.N) ;
   12: Terminate();
{=Gen} ::Gen 13:Future.NewWait(1000000,0,Gen,Gen+1);
 14: Terminate(1);

 |R `Blocks    BLOCK Report
    Location     Total   Current
         1      9914         0
         2      9914         0
         3      9914        62
         4      9852         0
         5      9852         0
         6      9852         0
         7      9852         0
         8      9852         5
         9      9847         0
        10      9847         0
        11      9847         0
        12      9847         0
        13         1         0
        14         1         0
 |R `Values
 |R Facility = `fac01 Entries = 1963 Current = 1 Utility =  0.96211 AverTime = 490.12212
    Tran = 9869 IntTran = 0 Active Time = 1000000.00000 NextGo = 0
 |R Facility = `fac02 Entries = 1943 Current = 1 Utility =  0.94826 AverTime = 488.03949
    Tran = 9885 IntTran = 0 Active Time = 1000000.00000 NextGo = 0
 |R Facility = `fac03 Entries = 1975 Current = 1 Utility =  0.96625 AverTime = 489.23799
    Tran = 9842 IntTran = 0 Active Time = 1000000.00000 NextGo = 0
 |R Facility = `fac04 Entries = 1990 Current = 1 Utility =  0.97688 AverTime = 490.89438
    Tran = 9744 IntTran = 0 Active Time = 1000000.00000 NextGo = 0
 |R Facility = `fac05 Entries = 1981 Current = 1 Utility =  0.96631 AverTime = 487.79065
    Tran = 9906 IntTran = 0 Active Time = 1000000.00000 NextGo = 0
 |R Queue = `que01 Entries = 1972 Current = 10 AverQueue =  7.68588 AverTime = 3897.50628
    AverTime(-Ze) = 3937.43975 Max = 27 Min = 0 Zero = 20 Active Time = 1000000.00000 NextGo = 0
 |R Queue = `que02 Entries = 1950 Current = 8 AverQueue =  8.18424 AverTime = 4197.04391
```

```
   AverTime(-Ze) = 4298.44308 Max = 26 Min = 0 Zero = 46 Active Time = 1000000.00000 NextGo = 0
 |R Queue = `que03 Entries = 1991 Current = 17 AverQueue = 10.70572 AverTime = 5377.05607
   AverTime(-Ze) = 5470.47452 Max = 38 Min = 0 Zero = 34 Active Time = 1000000.00000 NextGo = 0
 |R Queue = `que04 Entries = 2018 Current = 29 AverQueue = 15.74714 AverTime = 7803.33997
   AverTime(-Ze) = 7897.26181 Max = 49 Min = 0 Zero = 24 Active Time = 1000000.00000 NextGo = 0
 |R Queue = `que05 Entries = 1983 Current = 3 AverQueue = 14.07842 AverTime = 7099.55387
   AverTime(-Ze) = 7186.53156 Max = 45 Min = 0 Zero = 24 Active Time = 1000000.00000 NextGo = 0
 |R User/Future = `Future Entries = 19769 Current = 7 AverUser =  6.30230 AverTime = 318.79689
   Max = 7 Min = 0 Active Time = 1000000.00000 NextGo = 0
 |R Report  User/Future List Count= 7
     Tran     Assem  Priority   Current      Next       TimeStamp       SortValue     Group
Inter
     9916      9916         0         0         1   1000023.59654   1000023.59654         0
FALSE
     9842      9842         0         8         9    991874.02299   1000192.58364         0
FALSE
     9885      9885         0         8         9    996621.54699   1000201.71405         0
FALSE
     9744      9744         0         8         9    982057.98355   1000206.67582         0
FALSE
     9869      9869         0         8         9    995045.40608   1000245.59869         0
FALSE
     9906      9906         0         8         9    998720.90572   1000355.76481         0
FALSE
     9917      9917         0         0        13   2000000.00000   2000000.00000         0
FALSE
            Num/Name        Integer  |R `Future      Report IP LIST  Count= 4
I                 1           9915
I                 2           9916
I                13              2
I                14           9917
            Num/Name         Double  |R ` BSigm       Report RP LIST  Count= 5
R                 1     4452.20579
R                 2     4798.88538
R                 3     6413.26258
R                 4     9060.50453
R                 5     8356.63720
            Num/Name         Double  |R ` BSum       Report RP LIST  Count= 5
R                 1     1963.00000
R                 2     1943.00000
R                 3     1975.00000
R                 4     1990.00000
R                 5     1981.00000
            Num/Name         Double  |R ` BSsqr       Report RP LIST  Count= 5
R                 1  38910853674.33553
R                 2  44745931712.19305
R                 3  81231625311.07827
R                 4  163364557126.39206
R                 5  138339936189.49823
            Num/Name         Double  |R `Sys       Report RP LIST  Count= 5
R                 1  7654549.65348
R                 2  8171820.43959
R                 3  10622302.43855
R                 4  15460984.94887
R                 5  14076299.46233
```

В этой модели объект Bsigm используется для накопления суммы квадратов времен в очереди для каждой из очередей. Объект Bsum – используется для определения числа заявок, покинувших каждую из очередей. Объект BSsqr - используется для определения среднеквадратичных значений времени в каждой из очередей. Очевидно, что такая модель выглядит относительно логичной и понятной за счет использования объектов класса Tparam.

Для обнуления текущего времени жизни заявки, используется блок *:Mark(ATran);

Этот блок обнуляет время жизни заявки с данным номером, по умолчанию - активной.

Чаще всего блок используется для фиксации интервалов времени, прожитых заявками после прохождения определенного места в модели. Это позволяет фиксировать такие величины, как время обслуживания, время в очереди, и так далее.

Возможности данного блока очень легко перекрыть блоком RPlet.

Для этого, в нужном месте модели помещается блок, вида.

*:RPlet(Num,[AClock]);

Тогда в RP(Num) - параметр заявки заносится текущее время моделирования. Если впоследствии вычислить величину AClock - RP(Num) – то мы получим значение нужного нам интервала.

Чаще всего значение времени жизни заявки 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.

```
  {~vb} Var
{~f}Future:TFuture;
{/T} T:Ttable;
{/T0} T0:Ttable;
{/T1} T1:Ttable;
{/Q} Q:Tqueue;
{/Q0} Q0:Tqueue;
{/Q1} Q1:Tqueue;
{/F} F:Tfacility;
{/F0} F0:Tfacility;
{/F1} F1:Tfacility;
 {~ve}
 {~pfe}
 {~ib} procedure Initial;begin
 init(Future,'Future');
  SetStart(1000);
{/T}  init(T,'t',0,100,100);
{/T0}  init(T0,'t0',0,50,100);
{/T1}  init(T1,'t1',0,20,100);
{/Q}  init(Q,'q');
{/Q0}  init(Q0,'q0');
{/Q1}  init(Q1,'q1');
{/F}  init(F,'f');
{/F0}  init(F0,'f0');
{/F1}  init(F1,'f1');
{=G0} Future.NewWaitProc(G0,MeanDev(100,90),0,G0,G0+1);
 {~ie}end;
 {~mtb}procedure ModelTxt;begin case ActiveBlock of
{=G0} ::G0 *:Future.NewWait(MeanDev(100,90),0,G0,G0+1);
```

```
*: RPlet(1,[AClock]) ;
*: Q.Queue ;
*: F.Seize ;
*: Q.Depart ;
*: T.Tabulate(AClock-RP(1)) ;
*:Future.Wait(MeanDev(90,80));
*: F.Release ;
*: RPlet(2,[AClock]) ;
*: Q0.Queue ;
*: F0.Seize ;
*: Q0.Depart ;
*: T0.Tabulate(AClock-RP(2)) ;
*:Future.Wait(MeanDev(75,70));
*: F0.Release ;
*: RPlet(3,[AClock]) ;
*: Q1.Queue ;
*: F1.Seize ;
*: Q1.Depart ;
*: T1.Tabulate(AClock-RP(3)) ;
*:Future.Wait(MeanDev(60,55));
*: F1.Release ;
*: Terminate(1) ;
{~mte} end;end;
{~mb} procedure Simulation;  begin
start(GetTerm);
 Show(T,1);
 Show(T0,2);
 Show(T1,3);
{~me}end;
{~rb} procedure Report;begin
{~re} end;
{~cab} procedure CloseAllObj;begin
{~cae} end;
```

Тогда итоговая статистика будет иметь вид:

```
 Date Time Stamp= 18.12.2008 10:40:40   Ctrl Click |Name - HyperLink `R
E:\work\ObjectClient\Object No Sys\Future\GPSS-Future\book\tmp15.rtf
StartTime         0.00000 EndTime      97463.61951
 |Model |Blocks |Values |CurrentList |SynchronizeList |InterruptList
 |RPList |IPList |SPList |BPList
   Num      NameObject
     2        |Future
     3             |t
     4            |t0
     5            |t1
     6             |q
     7            |q0
     8            |q1
     9             |f
    10            |f0
    11            |f1
 |R `Model
 {=G0} ::G0 1:Future.NewWait(MeanDev(100,90),0,G0,G0+1);
        2: RPlet(1,[AClock]) ;
```

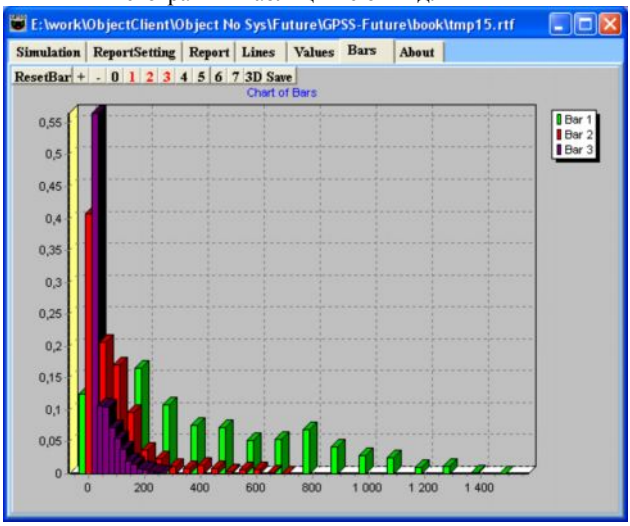
```
3: Q.Queue ;
4: F.Seize ;
5: Q.Depart ;
6: T.Tabulate(AClock-RP(1)) ;
7:Future.Wait(MeanDev(90,80));
8: F.Release ;
9: RPlet(2,[AClock]) ;
10: Q0.Queue ;
11: F0.Seize ;
12: Q0.Depart ;
13: T0.Tabulate(AClock-RP(2)) ;
14:Future.Wait(MeanDev(75,70));
15: F0.Release ;
16: RPlet(3,[AClock]) ;
17: Q1.Queue ;
18: F1.Seize ;
19: Q1.Depart ;
20: T1.Tabulate(AClock-RP(3)) ;
21:Future.Wait(MeanDev(60,55));
22: F1.Release ;
23: Terminate(1) ;

|R `Blocks    BLOCK Report
   Location     Total   Current
        1      1006         0
        2      1006         0
        3      1006         3
        4      1003         0
        5      1003         0
        6      1003         0
        7      1003         1
        8      1002         0
        9      1002         0
       10      1002         0
       11      1002         0
       12      1002         0
       13      1002         0
       14      1002         1
       15      1001         0
       16      1001         0
       17      1001         1
       18      1000         0
       19      1000         0
       20      1000         0
       21      1000         0
       22      1000         0
       23      1000         0
|R `Values
|R `CurrentList Report Current Transactions List  Count= 4
     Tran     Assem  Priority   Current      Next        TimeStamp       TimeEnter
Inter
     1004      1004         0         3         4     97081.27285     97081.27285
FALSE
     1005      1005         0         3         4     97250.08872     97250.08872
FALSE
     1006      1006         0         3         4     97376.90279     97376.90279
FALSE
     1001      1001         0        17        18     96943.05822     97402.25348
FALSE
|R `RPList  Report ( Double ) Count= 9
       Tran            Num/Name         Double
       1001                   1    96943.05822
       1001                   2    97380.86376
       1001                   3    97402.25348
       1002                   1    96955.45353
       1002                   2    97457.91985
       1003                   1    96972.91669
       1004                   1    97081.27285
       1005                   1    97250.08872
       1006                   1    97376.90279
```

```
          |R Facility = `f Entries = 1003 Current = 1 Utility =  0.92171 AverTime = 89.56419
             Tran = 1003 IntTran = 0 Active Time = 97463.61951 NextGo = 0
          |R Facility = `f0 Entries = 1002 Current = 1 Utility =  0.74703 AverTime = 72.66256
             Tran = 1002 IntTran = 0 Active Time = 97463.61951 NextGo = 0
          |R Facility = `f1 Entries = 1000 Current = 0 Utility =  0.62357 AverTime = 60.77582
             Tran = 0 IntTran = 0 Active Time = 97463.61951 NextGo = 0
          |R Queue = `q Entries = 1006 Current = 3 AverQueue =  3.64902 AverTime = 353.52560
             AverTime(-Ze) = 457.71783 Max = 15 Min = 0 Zero = 229 Active Time = 97463.61951
NextGo = 0
          |R Queue = `q0 Entries = 1002 Current = 0 AverQueue =  0.66548 AverTime = 64.73049
             AverTime(-Ze) = 202.68734 Max = 9 Min = 0 Zero = 682 Active Time = 97463.61951
NextGo = 0
          |R Queue = `q1 Entries = 1001 Current = 1 AverQueue =  0.23073 AverTime = 22.46508
             AverTime(-Ze) = 144.15092 Max = 4 Min = 0 Zero = 845 Active Time = 97463.61951
NextGo = 0
          |R User/Future = `Future Entries = 4012 Current = 3 AverUser =  2.52720 AverTime =
61.39337
             Max = 4 Min = 0 Active Time = 97463.61951 NextGo = 0
          |R Report  User/Future List Count= 3
               Tran     Assem  Priority   Current      Next       TimeStamp       SortValue
Group Inter
               1007      1007         0         0         1     97494.85544     97494.85544
0 FALSE
               1003      1003         0         7         8     96972.91669     97542.31057
0 FALSE
               1002      1002         0        14        15     96955.45353     97574.63404
0 FALSE
                     Num/Name        Integer  |R `Future      Report IP LIST  Count= 2
          I                 1           1007
          I                 2           1007
          |R  Table = `t  Entries = 1003.00000 Mean = 353.90246 StdDev = 327.42319
                  Range       Frequency        Integral
                 0.00000       125.00000         0.12463
               100.00000       155.00000         0.27916
               200.00000       166.00000         0.44467
               300.00000       109.00000         0.55334
               400.00000        76.00000         0.62911
               500.00000        73.00000         0.70189
               600.00000        53.00000         0.75474
               700.00000        55.00000         0.80957
               800.00000        70.00000         0.87936
               900.00000        43.00000         0.92223
              1000.00000        28.00000         0.95015
              1100.00000        26.00000         0.97607
              1200.00000        10.00000         0.98604
              1300.00000        12.00000         0.99801
              1400.00000         2.00000         1.00000
              1500.00000         0.00000         1.00000
          |R  Table = `t0  Entries = 1002.00000 Mean = 64.73049 StdDev = 105.62904
                  Range       Frequency        Integral
                 0.00000       408.00000         0.40719
                50.00000       206.00000         0.61277
               100.00000       172.00000         0.78443
               150.00000        96.00000         0.88024
               200.00000        37.00000         0.91717
               250.00000        23.00000         0.94012
               300.00000        12.00000         0.95210
               350.00000         8.00000         0.96008
               400.00000        13.00000         0.97305
               450.00000         8.00000         0.98104
               500.00000         4.00000         0.98503
               550.00000         6.00000         0.99102
               600.00000         7.00000         0.99800
               650.00000         2.00000         1.00000
               700.00000         0.00000         1.00000
          |R  Table = `t1  Entries = 1000.00000 Mean = 22.42618 StdDev = 37.33767
                  Range       Frequency        Integral
                 0.00000       565.00000         0.56500
                20.00000       107.00000         0.67200
                40.00000       105.00000         0.77700
```

60.00000	73.00000	0.85000
80.00000	56.00000	0.90600
100.00000	39.00000	0.94500
120.00000	21.00000	0.96600
140.00000	15.00000	0.98100
160.00000	8.00000	0.98900
180.00000	6.00000	0.99500
200.00000	4.00000	0.99900
220.00000	1.00000	1.00000
240.00000	0.00000	1.00000

Гистограммы таблиц имеют вид.



Для присваивания значений переменным используются блок.

*:Let(Varia,Value);

Этот блок присваивает значения Value - переменной Varia для вещественных, целых, строковых и логических значений соответственно. Если параметр Value отсутствует, то его значение будет 0, 0, ‘’, False.

*:Add(Varia,Value);

Этот блок увеличивает значение Value - переменной на значение Varia для вещественных и целых значений. Если параметр Value отсутствует, то его значение будет 1.

Если нужно задать значения переменным вне процедуры ModelTxt, то следует использовать оператор присваивания, вида Varia:=Value; .

В принципе, использование переменных или X – параметров – это дело вкуса автора модели, так как обычно все равно, что именно использовать.

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

*:Buffer;

Иначе говоря, после него активная 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(Prior); .

Блок *:Future.Wait(0); - позволяет продвинуться все заявкам списка текущих событий, до продвижения активной в данный момент заявки.

23 Перенаправление потоков заявок

Безусловное перенаправление заявок выполняет блок

*:Transfer(NumBl,NumIP);

Этот блок обеспечивает переход активной заявки на блок с меткой NumBl.

В параметр IP с номером NumIP - записывается номер следующего блока для возможного возврата. По умолчанию, номер следующего блока никуда не записывается. В полной форме этот блок напоминает переход на подпрограмму. А в сокращенной форме – он подобен оператору GoTO известных языков программирования.

При работе с фрагментами модели как с подпрограммами, числовые, строковые и логические данные могут передаваться как P – параметры заявки, или даже как временные наборы P – параметров.

Вероятностное перенаправление заявок выполняет блок.

*:Transfer(Ver,NumBl1,NumBl2); Здесь Numbl1 и Numbl2 - метки перехода, а Ver - вероятность. Вероятность соответствует второй метке.

Для создания временных наборов P – параметров можно использовать блок

*:IncTran, который увеличивает номер последней выпущенной заявки на 1, однако с сама заявка при этом не создается. Далее, с помощью функции MaxTran можно узнать этот номер и сохранить его в IP- параметре заявки, а затем, при работе с P – параметрами можно будет в качестве номера фиктивной заявки использовать этот номер. Когда временный набор P – параметров вам будет не нужен, то его можно удалить с помощью блоков

*: RPDelete(ATran); этот блок удаляет все RP -параметры данной заявки. По умолчанию – удаляются параметры активной заявки.

*: IPDelete(ATran); этот блок удаляет все IP -параметры данной заявки. По умолчанию - удаляются параметры активной заявки.

*: SPDelete(ATran); этот блок удаляет все SP -параметры данной заявки. По умолчанию - удаляются параметры активной заявки.

*: BPDelete(ATran); этот блок удаляет все BP -параметры данной заявки. По умолчанию - удаляются параметры активной заявки.

*:Loop(NumIP,NumBl);

Этот блок обеспечивает организацию цикла. Он уменьшает значение IP(NumIp) на 1.

Если оно не нулевое, то заявка уходит по метке Numbl, иначе она идет дальше по модели.

Примеры использования выше рассмотренных блоков.

10 сборщиков собирают изделия. Каждый собирает свое изделие за 30+-5 мин. Затем изделие подвергается напылению. На установке можно напылять сразу 2 изделия в течение 7+-3 мин. Затем проверяется качество каждого изделия по очереди в течение 3 мин. на установке контроля.

Промоделировать работу системы в течение 400 часов. Выполнить табулирование времени изготовления изделий.

```
  {~vb} Var
 {~f}Future:TFuture;
{/qq} qq:Tqueue;
{/dd} dd:Tqueue;
{/ss} ss:Tstorage;
{/un} un:Tfacility;
{/tab} tab:Ttable;
  {~ve}
  {~pfe}
  {~ib} procedure Initial;begin
   {~f}init(Future,'Future');
 setstart(1);
{/qq}  init(qq,' qq');
{/dd}  init(dd,'dd');
{/ss}  init(ss,'ss',2);
{/un}  init(un,'un');
{/tab}  init(tab,' tab',0,1,100);
{=Gen} Future.NewWaitProc(Gen,0,0,Gen,Gen+1);
{=Gen0} Future.NewWaitProc(Gen0,10000,0,Gen0,Gen0+1);
  {~ie}end;
  {~mtb}procedure ModelTxt;begin case ActiveBlock of
{=Gen} ::Gen *:Future.NewWait(0,0,Gen,Gen+1,0,true,10);
::ret   *:mark;
       *:Future.Wait(MeanDev(30,5));
       *:qq.queue;
       *:ss.enter(1);
       *:qq.depart;
   *:Future.Wait(MeanDev(7,3));
       *:ss.leave(1);
       *:dd.queue;
       *:un.seize;
       *:dd.depart;
   *:Future.Wait(3);
       *:un.release;
       *:tab.tabulate(TranAge);
       *:transfer(ret);
{=Gen0} ::Gen0 *:Future.NewWait(2400,0,Gen0,Gen0+1);
       *:terminate(1);
  {~mte} end;end;
  {~mb} procedure Simulation;  begin
 start(GetTerm);
  {~me}end;
  {~rb} procedure Report;begin
  {~re} end;
  {~cab} procedure CloseAllObj;begin
  {~cae} end;
```

Более сложный пример.

Заявки двух типов поступают в систему с интервалами 200+-150 и 300+-250. Они обслуживаются в устройстве за среднее время 90 и 110 соответственно. Протабулировать время, проведенное в устройстве заявками первого и второго типов. Промоделировать обслуживание 5000 заявок.

Текст такой модели может выглядеть следующим образом.

```
  {~vb} Var
 {~f}Future:TFuture;
{/table0} table0:Ttable;
{/table1} table1:Ttable;
{/facil0} facil0:Tfacility;
{/qu} qu:Tqueue;
  {~ve}
  {~pfe}
  {~ib} procedure Initial;begin
   init(Future,'Future');
  SetStart(5000);
{/table0}  init(table0,' table0',0,10,100);
{/table1}  init(table1,' table1',0,10,100);
{/facil0}  init(facil0,' facil0');
{/qu}  init(qu,' qu');
{=gener0} Future.NewWaitProc(gener0,MeanDev(200,150),0,gener0,gener0+1);
{=gener1} Future.NewWaitProc(gener1,MeanDev(300,250),0,gener1,gener1+1);
  {~ie}end;
  {~mtb}procedure ModelTxt;begin case ActiveBlock of
{=gener0} ::gener0 *:Future.NewWait(MeanDev(200,150),0,gener0,gener0+1);
      *: RPlet(2,[90])  ;
      *: transfer(lab,2) ;
      *: table0.tabulate(TranAge) ;
      *: terminate(1) ;
{=gener1} ::gener1 *:Future.NewWait(MeanDev(300,250),0,gener1,gener1+1);
      *: RPlet(2,[110])  ;
          *: transfer(lab,2) ;
          *: table1.tabulate(TranAge) ;
           *: terminate(1) ;
::lab *: qu.queue ;
       *: facil0.seize  ;
        *: qu.depart ;
       *: mark ;
       *:Future.Wait(exponential(Rp(2)));
      *: facil0.release  ;
      *: topoint(0,aclock,qu.a) ;
      *: transfer(Ip(2))  ;
  {~mte} end;end;
  {~mb} procedure Simulation;  begin
```

```
  start(GetTerm);
 show(table0,2);
 show(table1,1);
 {~me}end;
 {~rb} procedure Report;begin
 {~re} end;
 {~cab} procedure CloseAllObj;begin
 {~cae} end;
```

Здесь блоки Transfer используются в двух вариантах.

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 1000 заявок.

Модель такой системы может выглядеть следующим образом.

```
  {~vb} Var
 {~f}Future:TFuture;
{/QKass} QKass:Tqueue;
{/Kass} Kass:Tfacility;
 {~ve}
 {~pfe}
 {~ib} procedure Initial;begin
 init(Future,'Future');
 SetStart(1000);
{/QKass}  init(QKass,' QKass');
{/Kass}  init(Kass,' Kass');
{=Gen} Future.NewWaitProc(Gen,MeanDev(100,80),0,Gen,Gen+1);
 {~ie}end;
 {~mtb}procedure ModelTxt;begin case ActiveBlock of
{=Gen} ::Gen *:Future.NewWait(MeanDev(100,80),0,Gen,Gen+1);
     *: IPlet(5,[Duniform(1,10)]) ;
::mmm     *: Qkass.Queue ;
     *: Kass.Seize ;
    *: Qkass.Depart ;
    *:Future.Wait(MeanDev(17,7));
    *: Kass.release ;
    *: Loop(5,mmm) ;
    *: terminate(1) ;
 {~mte} end;end;
 {~mb} procedure Simulation;  begin
  start(GetTerm);
 {~me}end;
 {~rb} procedure Report;begin
 {~re} end;
 {~cab} procedure CloseAllObj;begin
 {~cae} end;
```

Результирующая статистика модели имеет вид.

```
Date Time Stamp= 18.12.2008 12:41:49   Ctrl Click |Name - HyperLink `R
E:\work\ObjectClient\Object No Sys\Future\GPSS-Future\book\tmp18.rtf
StartTime         0.00000 EndTime     100216.00106
 |Model |Blocks |Values |CurrentList |SynchronizeList |InterruptList
 |RPList |IPList |SPList |BPList
   Num      NameObject
     2        |Future
     3        | QKass
     4         | Kass
 |R `Model
 {=Gen} ::Gen 1:Future.NewWait(MeanDev(100,80),0,Gen,Gen+1);
      2: IPlet(5,[Duniform(1,10)]) ;
::mmm     3: Qkass.Queue ;
     4: Kass.Seize ;
    5: Qkass.Depart ;
    6:Future.Wait(MeanDev(17,7));
    7: Kass.release ;
    8: Loop(5,mmm) ;
    9: terminate(1) ;

 |R `Blocks    BLOCK Report
    Location     Total   Current
         1      1012         0
         2      1012         0
         3      5515        12
         4      5503         0
         5      5503         0
         6      5503         0
         7      5503         0
         8      5503         0
         9      1000         0
 |R `Values
 |R `CurrentList Report Current Transactions List  Count= 12
      Tran     Assem  Priority   Current      Next       TimeStamp       TimeEnter Inter
      1001      1001         0         3         4     99301.30917     99301.30917 FALSE
      1002      1002         0         3         4     99333.30315     99333.30315 FALSE
      1003      1003         0         3         4     99434.42440     99434.42440 FALSE
      1004      1004         0         3         4     99508.37287     99508.37287 FALSE
      1005      1005         0         3         4     99546.38466     99546.38466 FALSE
      1006      1006         0         3         4     99604.13852     99604.13852 FALSE
      1007      1007         0         3         4     99732.78069     99732.78069 FALSE
      1008      1008         0         3         4     99793.25410     99793.25410 FALSE
      1009      1009         0         3         4     99889.06444     99889.06444 FALSE
      1010      1010         0         3         4     99931.82216     99931.82216 FALSE
      1011      1011         0         3         4     99997.83009     99997.83009 FALSE
      1012      1012         0         3         4    100167.61320    100167.61320 FALSE
 |R `IPList Report ( Integer ) Count= 12
       Tran            Num/Name        Integer
       1001                   5              9
       1002                   5              9
       1003                   5              2
       1004                   5              8
       1005                   5              7
       1006                   5              1
       1007                   5             10
       1008                   5              2
       1009                   5              7
       1010                   5              6
       1011                   5              1
       1012                   5              4
 |R Facility = ` Kass Entries = 5503 Current = 0 Utility =  0.93059 AverTime = 16.94709
    Tran = 0 IntTran = 0 Active Time = 100216.00106 NextGo = 0
 |R Queue = ` QKass Entries = 5515 Current = 12 AverQueue =  2.00592 AverTime = 36.45061
    AverTime(-Ze) = 241.03732 Max = 13 Min = 0 Zero = 4681 Active Time = 100216.00106 NextGo = 0
 |R User/Future = `Future Entries = 6516 Current = 1 AverUser =  1.07801 AverTime = 16.57981
    Max = 2 Min = 0 Active Time = 100216.00106 NextGo = 0
 |R Report  User/Future List Count= 1
      Tran     Assem  Priority   Current      Next       TimeStamp       SortValue     Group
Inter
```

```
      1013      1013         0         0         1    100310.51563    100310.51563         0
FALSE
            Num/Name        Integer  |R `Future      Report IP LIST  Count= 2
I                  1           1013
I                  2           1013
```

Пусть в систему поступают заявки со средним интервалом 120, распределенным по экспоненциальному закону. В системе имеется два устройства. Первое устройство выбирается с вероятностью 0.4, а второе – с вероятностью 0.6 . Время обслуживания заявок в любом из устройств одинаково, и равно 150+-100. Промоделировать обслуживание 1000 заявок. Вывести графики занятости устройств, и средней очереди к ним.

Текст модели такой системы, с использованием обращения к объектам по индексу может выглядит следующим образом.

```
{~vb} Var
 {~f}Future:TFuture;
{/QKass} QKass:Tqueue;
{/Kass} Kass:Tfacility;
{/kass1} kass1:Tfacility;
K:array[1..2] of Tfacility;
  {~ve}
  {~pfe}
  {~ib} procedure Initial;begin
 init(Future,'Future');
  SetStart(1000);
{/QKass}  init(QKass,' QKass');
{/Kass}  init(Kass,' Kass');
{/kass1}  init(kass1,' kass1');
k[1]:=kass;
k[2]:=kass1;
{=Gen} Future.NewWaitProc(Gen,Exponential(120),0,Gen,Gen+1);
  {~ie}end;
  {~mtb}procedure ModelTxt;begin case ActiveBlock of
{=Gen} ::Gen *:Future.NewWait(Exponential(120),0,Gen,Gen+1);
     *: Qkass.Queue ;
     *: IPlet(1,[IByBool([random<0.4,true],[1,2])]) ;
     *: k[Ip(1)].Seize ;
     *: Qkass.Depart ;
    *:Future.Wait(MeanDev(150,100));
     *: k[Ip(1)].release ;
     *: ToPoint(2,aclock,Qkass.A) ;
     *: ToPoint(1,aclock,Kass1.A) ;
     *: ToPoint(0,aclock,Kass.A) ;
     *: terminate(1) ;
  {~mte} end;end;
  {~mb} procedure Simulation;  begin
   start(Getterm);
//    start(Getterm);
  {~me}end;
```

{~rb} procedure Report;begin
{~re} end;
{~cab} procedure CloseAllObj;begin
{~cae} end;

Выходная статистика такой модели выглядит следующим образом. По этой статистике видно, что устройство kass1 действительно занималось чаще, чем kass. А обращение к устройствам в по индексу действительно работает.

```
Date Time Stamp= 18.12.2008 12:50:02   Ctrl Click |Name - HyperLink `R
E:\work\ObjectClient\Object No Sys\Future\GPSS-Future\book\tmp19.rtf
StartTime         0.00000 EndTime     114177.83349
 |Model |Blocks |Values |CurrentList |SynchronizeList |InterruptList
 |RPList |IPList |SPList |BPList
   Num      NameObject
     2        |Future
     3        | QKass
     4         | Kass
     5        | kass1
 |R `Model
 {=Gen} ::Gen 1:Future.NewWait(Exponential(120),0,Gen,Gen+1);
     2: Qkass.Queue ;
     3: IPlet(1,[IByBool([random<0.4,true],[1,2])]) ;
     4: k[Ip(1)].Seize ;
     5: Qkass.Depart ;
    6:Future.Wait(MeanDev(150,100));
     7: k[Ip(1)].release ;
     8: ToPoint(2,aclock,Qkass.A) ;
     9: ToPoint(1,aclock,Kass1.A) ;
     10: ToPoint(0,aclock,Kass.A) ;
     11: terminate(1) ;

 |R `Blocks    BLOCK Report
    Location     Total   Current
         1      1007         0
         2      1007         0
         3      1007         6
         4      1001         0
         5      1001         0
         6      1001         1
         7      1000         0
         8      1000         0
         9      1000         0
        10      1000         0
        11      1000         0
 |R `Values
 |R `CurrentList Report Current Transactions List  Count= 6
      Tran     Assem  Priority   Current      Next       TimeStamp       TimeEnter Inter
      1000      1000         0         3         4    113575.87444    113575.87444 FALSE
      1001      1001         0         3         4    113896.65936    113896.65936 FALSE
      1003      1003         0         3         4    113939.94748    113939.94748 FALSE
      1005      1005         0         3         4    113953.09097    113953.09097 FALSE
      1006      1006         0         3         4    114015.74847    114015.74847 FALSE
      1007      1007         0         3         4    114154.59107    114154.59107 FALSE
 |R `IPList Report ( Integer ) Count= 7
        Tran            Num/Name        Integer
        1000                  1              2
        1001                  1              2
        1003                  1              2
        1004                  1              1
        1005                  1              1
        1006                  1              2
        1007                  1              2
 |R Facility = ` Kass Entries = 400 Current = 1 Utility =  0.52054 AverTime = 148.58458
    Tran = 1004 IntTran = 0 Active Time = 114177.83349 NextGo = 0
 |R Facility = ` kass1 Entries = 601 Current = 0 Utility =  0.78695 AverTime = 149.50404
    Tran = 0 IntTran = 0 Active Time = 114177.83349 NextGo = 0
 |R Queue = ` QKass Entries = 1007 Current = 6 AverQueue =  2.26641 AverTime = 256.97477
```

```
    AverTime(-Ze) = 443.86551 Max = 15 Min = 0 Zero = 424 Active Time = 114177.83349 NextGo = 0
 |R User/Future = `Future Entries = 2009 Current = 2 AverUser =  1.77571 AverTime = 100.91942
    Max = 3 Min = 0 Active Time = 114177.83349 NextGo = 0
 |R Report  User/Future List Count= 2
      Tran     Assem  Priority   Current      Next       TimeStamp       SortValue     Group
Inter
      1008      1008         0         0         1    114215.70160    114215.70160         0
FALSE
      1004      1004         0         6         7    113950.64962    114358.99738         0
FALSE
            Num/Name        Integer  |R `Future      Report IP LIST  Count= 2
I                  1           1008
I                  2           1008
```

24 Блок проверки условий

Для явной проверки условий в GPSS используется блок TEST, который имеет следующий формат:

*:Test(Logs,NumBls);

Здесь Logs,NumBls – списки логических выражений и меток. Оба списка записываются через запятую в квадратных скобках. Просматриваются логические выражения, и для первого выражения со значением True происходит переход на метку с таким же порядковым номером. Если в списке все значения False- то заявка не входит в блок Test, а остается на выходе блока, в котором она находилась.

Блоки Test 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 10 суток.

Текст модели может выглядеть следующим образом.

```
 {~vb} Var
 {~f}Future:TFuture;
{/PERE} PERE:Tstorage;
{/Tb} Tb:Ttable;
{/Tb0} Tb0:Ttable;
{/QPere} QPere:Tqueue;
{\Sys} Sys:Tparam;
  {~ve}
  {~pfe}
  {~ib} procedure Initial;begin
 init(Future,'Future');
   SetStart(10);
{/PERE}  init(PERE,' PERE',2);
{/Tb}  init(Tb,' Tb',0,10,100);
{/Tb0}  init(Tb0,' Tb0',0,3,100);
```

```
{/QPere}  init(QPere,' QPere');
{\Sys} Init(Sys,'Sys');
{=G} Future.NewWaitProc(G,Exponential(20),0,G,G+1);
{=G0} Future.NewWaitProc(G0,Exponential(20),0,G0,G0+1);
{=G1} Future.NewWaitProc(G1,3600,900,0,G1,G1+1);
{=G1} Future.NewWaitProc(G1,MeanDev(3600,900),0,G1,G1+1);
{=G2} Future.NewWaitProc(G2,3600*24,0,G2,G2+1);
  {~ie}end;
  {~mtb}procedure ModelTxt;begin case ActiveBlock of
{=G} ::G *:Future.NewWait(Exponential(20),0,G,G+1);
::mm1    *: QPere.Queue ;
   *: test([not BP(1, sys.N)],[!*]) ;
       *: Pere.Enter ;
       *: QPere.Depart ;
      *:Future.Wait(MeanDev(15,3));
       *: Pere.Leave ;
        *: Tb.Tabulate(TranAge);
        *: Terminate ;
{=G0} ::G0 *:Future.NewWait(Exponential(20),0,G0,G0+1);
         *: Transfer(mm1);
{=G1} ::G1 *:Future.NewWait(MeanDev(3600,900),0,G1,G1+1);
          *: BPlet(1,[True], sys.N) ;
      *:Future.Wait(MeanDev(300,60));
       *: Tb0.tabulate(Qpere.L) ;
          *: BPlet(1,[False], sys.N) ;
            *: Terminate ;
{=G2} ::G2 *:Future.NewWait(3600*24,0,G2,G2+1);
     *:toString(0,inttostr(term));
      *: Terminate(1)  ;
  {~mte} end;end;
  {~mb} procedure Simulation;  begin
   start(GetTerm);
  {~me}end;
  {~rb} procedure Report;begin
  {~re} end;
  {~cab} procedure CloseAllObj;begin
  {~cae} end;
```

Результирующая статистика может выглядеть следующим образом.

```
Date Time Stamp= 18.12.2008 13:19:19   Ctrl Click |Name - HyperLink `R
E:\work\ObjectClient\Object No Sys\Future\GPSS-Future\book\tmp20.rtf
StartTime          0.00000 EndTime     864000.00000
 |Model |Blocks |Values |CurrentList |SynchronizeList |InterruptList
 |RPList |IPList |SPList |BPList
   Num       NameObject
     2         |Future
     3          | PERE
     4            | Tb
     5           | Tb0
     6         | QPere
     7            |Sys
 |R `Model
```

```
 {=G} ::G 1:Future.NewWait(Exponential(20),0,G,G+1);
::mm1    2: QPere.Queue ;
   3: test([not BP(1, sys.N)],[4]) ;
        4: Pere.Enter ;
        5: QPere.Depart ;
       6:Future.Wait(MeanDev(15,3));
        7: Pere.Leave ;
         8: Tb.Tabulate(TranAge);
         9: Terminate ;
{=G0} ::G0 10:Future.NewWait(Exponential(20),0,G0,G0+1);
            11: Transfer(mm1);
{=G1} ::G1 12:Future.NewWait(MeanDev(3600,900),0,G1,G1+1);
           13: BPlet(1,[True], sys.N) ;
       14:Future.Wait(MeanDev(300,60));
        15: Tb0.tabulate(Qpere.L) ;
           16: BPlet(1,[False], sys.N) ;
              17: Terminate ;
{=G2} ::G2 18:Future.NewWait(3600*24,0,G2,G2+1);
      19:toString(0,inttostr(term));
       20: Terminate(1)  ;

 |R `Blocks    BLOCK Report
    Location     Total   Current
           1     43507         0
           2     86636         0
           3     86636         0
           4     86636         0
           5     86636         0
           6     86636         2
           7     86634         0
           8     86634         0
           9     86634         0
          10     43129         0
          11     43129         0
          12       476         0
          13       476         0
          14       476         0
          15       476         0
          16       476         0
          17       476         0
          18        10         0
          19        10         0
          20        10         0
 |R `Values
 |R Storage = ` PERE Entries = 86636 Current = 2 Average =  1.50406 AverTime =
14.99959 Utility =  0.75203
    Max = 2 Min = 0 Tran = 87125 ACapac =  2.00000 Capacity = 2 Active Time =
864000.00000 NextGo = 0
 |R Queue = ` QPere Entries = 86636 Current = 0 AverQueue = 10.02805 AverTime =
100.00733
    AverTime(-Ze) = 129.90636 Max = 58 Min = 0 Zero = 19940 Active Time = 864000.00000
NextGo = 0
 |R User/Future = `Future Entries = 174239 Current = 7 AverUser =  6.16100 AverTime =
30.55061
    Max = 9 Min = 0 Active Time = 864000.00000 NextGo = 0
 |R Report  User/Future List Count= 7
      Tran     Assem  Priority   Current      Next       TimeStamp       SortValue
Group Inter
     86749     86749         0         0        12    864007.06942    864007.06942
0 FALSE
     87123     87123         0         6         7    863986.30405    864009.45866
0 FALSE
     87125     87125         0         6         7    863988.60305    864010.86864
0 FALSE
     87124     87124         0         0         1    864013.59503    864013.59503
0 FALSE
     87126     87126         0         0        10    864020.23222    864020.23222
0 FALSE
     86870     86870         0         0        12    864527.89141    864527.89141
0 FALSE
```

```
                87127     87127         0         0        18    950400.00000    950400.00000
0 FALSE
                     Num/Name        Integer  |R `Future      Report IP LIST  Count= 9
          I                 0              1
          I                 1          43508
          I                 2          87124
          I                10          43130
          I                11          87126
          I                12            480
          I                13          86870
          I                18             11
          I                19          87127
           |R  Table = ` Tb  Entries = 86634.00000 Mean = 115.00929 StdDev = 104.06612
                    Range       Frequency        Integral
                 10.00000         0.00000         0.00000
                 20.00000     19555.00000         0.22572
                 30.00000     10315.00000         0.34478
                 40.00000      5897.00000         0.41285
                 50.00000      3344.00000         0.45145
                 60.00000      2220.00000         0.47708
                 70.00000      1967.00000         0.49978
                 80.00000      1648.00000         0.51880
                 90.00000      1497.00000         0.53608
                100.00000      1524.00000         0.55367
                110.00000      1546.00000         0.57152
                120.00000      1469.00000         0.58848
                130.00000      1500.00000         0.60579
                140.00000      1514.00000         0.62327
                150.00000      1701.00000         0.64290
                160.00000      1766.00000         0.66328
                170.00000      1750.00000         0.68348
                180.00000      1739.00000         0.70356
                190.00000      1872.00000         0.72517
                200.00000      1821.00000         0.74619
                210.00000      1766.00000         0.76657
                220.00000      1865.00000         0.78810
                230.00000      1860.00000         0.80957
                240.00000      1785.00000         0.83017
                250.00000      1852.00000         0.85155
                260.00000      1895.00000         0.87342
                270.00000      1632.00000         0.89226
                280.00000      1508.00000         0.90967
                290.00000      1371.00000         0.92549
                300.00000      1195.00000         0.93928
                310.00000       999.00000         0.95082
                320.00000       927.00000         0.96152
                330.00000       871.00000         0.97157
                340.00000       775.00000         0.98052
                350.00000       665.00000         0.98819
                360.00000       520.00000         0.99419
                370.00000       270.00000         0.99731
                380.00000       120.00000         0.99870
                390.00000        54.00000         0.99932
                400.00000        28.00000         0.99964
                410.00000        14.00000         0.99980
                420.00000         4.00000         0.99985
                430.00000         2.00000         0.99987
                440.00000         5.00000         0.99993
                450.00000         6.00000         1.00000
                460.00000         0.00000         1.00000
           |R  Table = ` Tb0  Entries = 476.00000 Mean = 29.59874 StdDev =  6.58329
                    Range       Frequency        Integral
                  6.00000         0.00000         0.00000
                  9.00000         1.00000         0.00210
                 12.00000         1.00000         0.00420
                 15.00000         5.00000         0.01471
                 18.00000         9.00000         0.03361
                 21.00000        32.00000         0.10084
                 24.00000        62.00000         0.23109
                 27.00000        71.00000         0.38025
```

```
30.00000        89.00000        0.56723
33.00000        88.00000        0.75210
36.00000        45.00000        0.84664
39.00000        40.00000        0.93067
42.00000        17.00000        0.96639
45.00000        11.00000        0.98950
48.00000         4.00000        0.99790
51.00000         0.00000        0.99790
54.00000         1.00000        1.00000
57.00000         0.00000        1.00000
    Num/Name        Boolean  |R `Sys      Report BP LIST Count= 1
B                1          FALSE
```

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:

```
    {~vb} Var
 {~f}Future:TFuture;
{/Fac1} Fac1:Tfacility;
{/Fac2} Fac2:Tfacility;
{/Fac3} Fac3:Tfacility;
{/Que} Que:Tqueue;
{/Que3} Que3:Tqueue;
  {~ve}
  {~pfe}
  {~ib} procedure Initial;begin
 init(Future,'Future');
  setstart(1000);
{/Fac1}  init(Fac1,' Fac1');
{/Fac2}  init(Fac2,' Fac2');
{/Fac3}  init(Fac3,' Fac3');
{/Que}  init(Que,' Que');
{/Que3}  init(Que3,' Que3');
{=Gen} Future.NewWaitProc(Gen,MeanDev(100,50),0,Gen,Gen+1);
  {~ie}end;
  {~mtb}procedure ModelTxt;begin case ActiveBlock of
{=Gen} ::Gen *:Future.NewWait(MeanDev(100,50),0,Gen,Gen+1);
    *: que.Queue ;
    *: test([fac1.L=0,fac2.L=0],[!*,tofac2]) ;
    *: fac1.seize ;
    *: que.depart ;
    *:Future.Wait(MeanDev(150,50));
    *: fac1.release ;
    *: transfer(tofac3) ;
::tofac2   *: fac2.seize ;
    *: que.depart ;
    *:Future.Wait(MeanDev(170,40));
```

```
    *: fac2.release ;
::tofac3     *: que3.Queue ;
    *: fac3.seize ;
    *: que3.depart ;
    *:Future.Wait(MeanDev(90,30));
    *: fac3.release ;
    *: terminate(1) ;
  {~mte} end;end;
  {~mb} procedure Simulation;  begin
   start(GetTerm);
  {~me}end;
  {~rb} procedure Report;begin
  {~re} end;
  {~cab} procedure CloseAllObj;begin
  {~cae} end;
```

Результат моделирования имеет следующий вид:

```
Date Time Stamp= 18.12.2008 13:26:35   Ctrl Click |Name - HyperLink `R
E:\work\ObjectClient\Object No Sys\Future\GPSS-Future\book\tmp21.rtf
StartTime         0.00000 EndTime      99595.44507
 |Model |Blocks |Values |CurrentList |SynchronizeList |InterruptList
 |RPList |IPList |SPList |BPList
   Num      NameObject
     2        |Future
     3         | Fac1
     4         | Fac2
     5         | Fac3
     6          | Que
     7         | Que3
 |R `Model
 {=Gen} ::Gen 1:Future.NewWait(MeanDev(100,50),0,Gen,Gen+1);
     2: que.Queue ;
     3: test([fac1.L=0,fac2.L=0],[4,tofac2]) ;
     4: fac1.seize ;
     5: que.depart ;
     6:Future.Wait(MeanDev(150,50));
     7: fac1.release ;
     8: transfer(tofac3) ;
::tofac2   9: fac2.seize ;
     10: que.depart ;
     11:Future.Wait(MeanDev(170,40));
     12: fac2.release ;
::tofac3     13: que3.Queue ;
     14: fac3.seize ;
     15: que3.depart ;
     16:Future.Wait(MeanDev(90,30));
     17: fac3.release ;
     18: terminate(1) ;

 |R `Blocks    BLOCK Report
    Location     Total   Current
         1      1002         0
         2      1002         0
         3      1002         0
         4       529         0
         5       529         0
         6       529         0
         7       529         0
         8       529         0
         9       473         0
        10       473         0
        11       473         1
        12       472         0
```

```
        13      1001         1
        14      1000         0
        15      1000         0
        16      1000         0
        17      1000         0
        18      1000         0
 |R `Values
 |R `CurrentList Report Current Transactions List  Count= 1
      Tran     Assem  Priority   Current      Next       TimeStamp       TimeEnter Inter
      1001      1001        0        13        14     99400.94602     99552.06081 FALSE
 |R Facility = ` Fac1 Entries = 529 Current = 0 Utility =  0.80181 AverTime = 150.95772
    Tran = 0 IntTran = 0 Active Time = 99595.44507 NextGo = 0
 |R Facility = ` Fac2 Entries = 473 Current = 1 Utility =  0.80971 AverTime = 170.49258
    Tran = 1002 IntTran = 0 Active Time = 99595.44507 NextGo = 0
 |R Facility = ` Fac3 Entries = 1000 Current = 0 Utility =  0.89922 AverTime = 89.55842
    Tran = 0 IntTran = 0 Active Time = 99595.44507 NextGo = 0
 |R Queue = ` Que Entries = 1002 Current = 0 AverQueue =  0.09451 AverTime =  9.39374
    AverTime(-Ze) = 522.91822 Max = 2 Min = 0 Zero = 984 Active Time = 99595.44507 NextGo = 0
 |R Queue = ` Que3 Entries = 1001 Current = 1 AverQueue =  0.39059 AverTime = 38.86240
    AverTime(-Ze) = 240.13125 Max = 3 Min = 0 Zero = 839 Active Time = 99595.44507 NextGo = 0
 |R User/Future = `Future Entries = 3005 Current = 2 AverUser =  2.85629 AverTime = 94.66684
    Max = 4 Min = 0 Active Time = 99595.44507 NextGo = 0
 |R Report  User/Future List Count= 2
      Tran     Assem  Priority   Current      Next       TimeStamp       SortValue     Group
Inter
      1003      1003         0         0         1     99642.55616     99642.55616         0
FALSE
      1002      1002         0        11        12     99498.86960     99670.31866         0
FALSE
            Num/Name        Integer  |R `Future      Report IP LIST  Count= 2
I                 1           1003
I                 2           1003
```

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

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 - 0.6 – а во второе – 0.4 . Промоделировать обслуживание 1000 заявок.

Тогда модель может выглядеть следующим образом.

```
{~vb} Var
 {~f}Future:TFuture;
{/Fac1} Fac1:Tfacility;
{/Fac2} Fac2:Tfacility;
{/Que2} Que2:Tqueue;
{/Que1} Que1:Tqueue;
  {~ve}
  {~pfe}
  {~ib} procedure Initial;begin
 init(Future,'Future');
  setstart(1000);
{/Fac1}  init(Fac1,' Fac1');
{/Fac2}  init(Fac2,' Fac2');
{/Que2}  init(Que2,' Que2');
{/Que1}  init(Que1,' Que1');
{=Gen} Future.NewWaitProc(Gen,MeanDev(100,70),0,Gen,Gen+1);
```

```
  {~ie}end;
  {~mtb}procedure ModelTxt;begin case ActiveBlock of
{=Gen} ::Gen *:Future.NewWait(MeanDev(100,70),0,Gen,Gen+1);
   *: test([Random<=0.6,true],[!*,tofac2]) ;
    *: que1.Queue ;
     *: fac1.seize ;
    *: que1.depart ;
    *:Future.Wait(MeanDev(150,90));
    *: fac1.release ;
    *: terminate(1) ;
::tofac2  *: que2.Queue ;
    *: fac2.seize ;
    *: que2.depart ;
    *:Future.Wait(MeanDev(200,80));
     *: fac2.release ;
    *: terminate(1) ;
  {~mte} end;end;
  {~mb} procedure Simulation;  begin
   start(GetTerm);
  {~me}end;
  {~rb} procedure Report;begin
  {~re} end;
  {~cab} procedure CloseAllObj;begin
  {~cae} end;
```

Выходная статистика этой модели имеет вид:

```
Date Time Stamp= 18.12.2008 14:18:20   Ctrl Click |Name - HyperLink `R
E:\work\ObjectClient\Object No Sys\Future\GPSS-Future\book\tmp22.rtf
StartTime         0.00000 EndTime      99136.18517
 |Model |Blocks |Values |CurrentList |SynchronizeList |InterruptList
 |RPList |IPList |SPList |BPList
   Num      NameObject
     2        |Future
     3         | Fac1
     4         | Fac2
     5         | Que2
     6         | Que1
 |R `Model
 {=Gen} ::Gen 1:Future.NewWait(MeanDev(100,70),0,Gen,Gen+1);
    2: test([Random<=0.6,true],[3,tofac2]) ;
     3: que1.Queue ;
      4: fac1.seize ;
     5: que1.depart ;
     6:Future.Wait(MeanDev(150,90));
     7: fac1.release ;
     8: terminate(1) ;
::tofac2  9: que2.Queue ;
     10: fac2.seize ;
     11: que2.depart ;
     12:Future.Wait(MeanDev(200,80));
      13: fac2.release ;
     14: terminate(1) ;

 |R `Blocks    BLOCK Report
    Location     Total   Current
        1       1005        0
        2       1005        0
        3        581        1
```

```
         4       580         0
         5       580         0
         6       580         0
         7       580         0
         8       580         0
         9       424         3
        10       421         0
        11       421         0
        12       421         1
        13       420         0
        14       420         0
 |R `Values
 |R `CurrentList Report Current Transactions List  Count= 4
      Tran     Assem  Priority   Current      Next       TimeStamp       TimeEnter Inter
      1000      1000         0         9        10     98670.82837     98670.82837 FALSE
      1002      1002         0         9        10     98903.64309     98903.64309 FALSE
      1004      1004         0         9        10     99007.60773     99007.60773 FALSE
      1005      1005         0         3         4     99114.38255     99114.38255 FALSE
 |R Facility = ` Fac1 Entries = 580 Current = 0 Utility =  0.86574 AverTime = 147.97585
    Tran = 0 IntTran = 0 Active Time = 99136.18517 NextGo = 0
 |R Facility = ` Fac2 Entries = 421 Current = 1 Utility =  0.82952 AverTime = 195.33385
    Tran = 998 IntTran = 0 Active Time = 99136.18517 NextGo = 0
 |R Queue = ` Que2 Entries = 424 Current = 3 AverQueue =  1.12727 AverTime = 263.56796
    AverTime(-Ze) = 483.77842 Max = 8 Min = 0 Zero = 193 Active Time = 99136.18517 NextGo = 0
 |R Queue = ` Que1 Entries = 581 Current = 1 AverQueue =  1.29952 AverTime = 221.73787
    AverTime(-Ze) = 400.09224 Max = 7 Min = 0 Zero = 259 Active Time = 99136.18517 NextGo = 0
 |R User/Future = `Future Entries = 2007 Current = 2 AverUser =  2.24982 AverTime = 111.13057
    Max = 3 Min = 0 Active Time = 99136.18517 NextGo = 0
 |R Report  User/Future List Count= 2
      Tran     Assem  Priority   Current      Next       TimeStamp       SortValue     Group
Inter
      1006      1006         0         0         1     99147.26192     99147.26192         0
FALSE
       998       998         0        12        13     98467.81140     99270.10283         0
FALSE
            Num/Name        Integer  |R `Future      Report IP LIST  Count= 2
I                 1           1006
I                 2           1006
```

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

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.

```
 {~vb} Var
 {~f}Future:TFuture;
{/Fac1} Fac1:Tfacility;
{/Que1} Que1:Tqueue;
{/Tb} Tb:Ttable;
 {~ve}
 {~pfe}
 {~ib} procedure Initial;begin
 init(Future,'Future');
 setstart(5000);
{/Fac1}  init(Fac1,' Fac1');
{/Que1}  init(Que1,' Que1');
{/Tb}  init(Tb,' Tb',0,100,50);
{=Gen} Future.NewWaitProc(Gen,exponential(100),0,Gen,Gen+1);
```

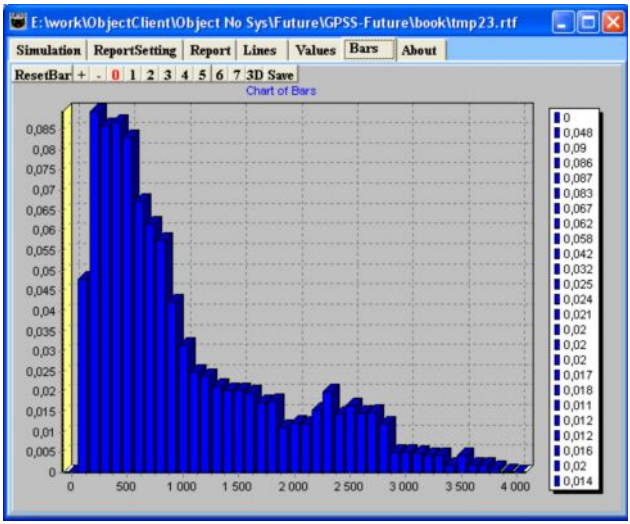
```
{~ie}end;
{~mtb}procedure ModelTxt;begin case ActiveBlock of
{=Gen} ::Gen *:Future.NewWait(exponential(100),0,Gen,Gen+1);
::ret     *: que1.Queue ;
     *: fac1.seize ;
    *: que1.depart ;
    *:Future.Wait(MeanDev(80,30));
    *: fac1.release ;
   *: test([Random<=0.15,true],[ret,!*]) ;
    *: tb.tabulate(TranAge) ;
    *: terminate(1) ;
{~mte} end;end;
{~mb} procedure Simulation;  begin
 start(GetTerm);
show(tb);
{~me}end;
{~rb} procedure Report;begin
{~re} end;
{~cab} procedure CloseAllObj;begin
{~cae} end;
```

Итоговая статистика имеет вид:

```
Date Time Stamp= 18.12.2008 14:23:24   Ctrl Click |Name - HyperLink `R
E:\work\ObjectClient\Object No Sys\Future\GPSS-Future\book\tmp23.rtf
StartTime          0.00000 EndTime     506810.28877
 |Model |Blocks |Values |CurrentList |SynchronizeList |InterruptList
 |RPList |IPList |SPList |BPList
   Num      NameObject
     2        |Future
     3         | Fac1
     4         | Que1
     5           | Tb
 |R `Model
 {=Gen} ::Gen 1:Future.NewWait(exponential(100),0,Gen,Gen+1);
::ret     2: que1.Queue ;
      3: fac1.seize ;
     4: que1.depart ;
     5:Future.Wait(MeanDev(80,30));
     6: fac1.release ;
    7: test([Random<=0.15,true],[ret,8]) ;
     8: tb.tabulate(TranAge) ;
     9: terminate(1) ;

 |R `Blocks    BLOCK Report
    Location      Total   Current
          1      5005         0
          2      5974         5
          3      5969         0
          4      5969         0
          5      5969         0
          6      5969         0
          7      5969         0
          8      5000         0
          9      5000         0
 |R `Values
 |R `CurrentList Report Current Transactions List  Count= 5
      Tran      Assem   Priority    Current       Next        TimeStamp        TimeEnter
Inter
      5001       5001          0           2          3     506247.63446     506247.63446
FALSE
```

```
                5002       5002          0          2          3     506437.31326     506437.31326
FALSE
                5003       5003          0          2          3     506565.72073     506565.72073
FALSE
                5004       5004          0          2          3     506711.29372     506711.29372
FALSE
                5005       5005          0          2          3     506728.65405     506728.65405
FALSE
            |R  Facility  =  `  Fac1  Entries  =  5969  Current  =  0  Utility  =   0.94234  AverTime  =
80.01094
               Tran = 0 IntTran = 0 Active Time = 506810.28877 NextGo = 0
            |R  Queue  =  `  Que1  Entries  =  5974  Current  =  5  AverQueue  =   8.58817  AverTime  =
728.58642
               AverTime(-Ze) = 961.47013 Max = 40 Min = 0 Zero = 1447 Active Time = 506810.28877
NextGo = 0
            |R User/Future = `Future Entries = 10975 Current = 1 AverUser =  1.29347 AverTime =
59.73075
               Max = 2 Min = 0 Active Time = 506810.28877 NextGo = 0
            |R Report  User/Future List Count= 1
                Tran      Assem   Priority    Current       Next         TimeStamp         SortValue
Group Inter
                5006       5006          0          0          1     507640.89605     507640.89605
0 FALSE
                      Num/Name        Integer  |R `Future      Report IP LIST  Count= 2
          I                  1           5006
          I                  2           5006
            |R  Table = ` Tb  Entries = 5000.00000 Mean = 965.75995 StdDev = 839.22030
                     Range       Frequency        Integral
                  0.00000         0.00000         0.00000
                100.00000       240.00000         0.04800
                200.00000       448.00000         0.13760
                300.00000       431.00000         0.22380
                400.00000       434.00000         0.31060
                500.00000       416.00000         0.39380
                600.00000       337.00000         0.46120
                700.00000       309.00000         0.52300
                800.00000       289.00000         0.58080
                900.00000       212.00000         0.62320
               1000.00000       158.00000         0.65480
               1100.00000       125.00000         0.67980
               1200.00000       120.00000         0.70380
               1300.00000       107.00000         0.72520
               1400.00000       102.00000         0.74560
               1500.00000       102.00000         0.76600
               1600.00000        99.00000         0.78580
               1700.00000        87.00000         0.80320
               1800.00000        88.00000         0.82080
               1900.00000        56.00000         0.83200
               2000.00000        62.00000         0.84440
               2100.00000        60.00000         0.85640
               2200.00000        78.00000         0.87200
               2300.00000       100.00000         0.89200
               2400.00000        72.00000         0.90640
               2500.00000        83.00000         0.92300
               2600.00000        74.00000         0.93780
               2700.00000        75.00000         0.95280
               2800.00000        60.00000         0.96480
               2900.00000        25.00000         0.96980
               3000.00000        25.00000         0.97480
               3100.00000        24.00000         0.97960
               3200.00000        22.00000         0.98400
               3300.00000        22.00000         0.98840
               3400.00000         9.00000         0.99020
               3500.00000        22.00000         0.99460
               3600.00000        10.00000         0.99660
               3700.00000        10.00000         0.99860
               3800.00000         6.00000         0.99980
               3900.00000         1.00000         1.00000
               4000.00000         0.00000         1.00000
```



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

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

Тогда модель может выглядеть следующим образом.

```
  {~vb} Var
 {~f}Future:TFuture;
{/Fac1} Fac1:Tfacility;
{/Fac2} Fac2:Tfacility;
{/Que2} Que2:Tqueue;
{/Que1} Que1:Tqueue;
  {~ve}
  {~pfe}
  {~ib} procedure Initial;begin
 {~f}init(Future,'Future');
 setstart(1000);
{/Fac1}  init(Fac1,' Fac1');
{/Fac2}  init(Fac2,' Fac2');
{/Que2}  init(Que2,' Que2');
{/Que1}  init(Que1,' Que1');
{=Gen} Future.NewWaitProc(Gen,MeanDev(5.6,5),0,Gen,Gen+1);
  {~ie}end;
  {~mtb}procedure ModelTxt;begin case ActiveBlock of
{=Gen} ::Gen *:Future.NewWait(MeanDev(5.6,5),0,Gen,Gen+1);
   *: test([que1.L<=que2.L,true],[!*,tofac2]) ;
    *: que1.Queue ;
```

```
     *: fac1.seize ;
    *: que1.depart ;
    *:Future.Wait(MeanDev(11,5));
    *: fac1.release ;
    *: terminate(1) ;
::tofac2  *: que2.Queue ;
    *: fac2.seize ;
    *: que2.depart ;
    *:Future.Wait(MeanDev(11,7));
    *: fac2.release ;
    *: terminate(1) ;
  {~mte} end;end;
  {~mb} procedure Simulation;  begin
   start(GetTerm);
  {~me}end;
  {~rb} procedure Report;begin
  {~re} end;
  {~cab} procedure CloseAllObj;begin
  {~cae} end;
```

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

```
  {~vb} Var
 {~f}Future:TFuture;
{/Fac1} Fac1:Tfacility;
{/Que1} Que1:Tqueue;
  {~ve}
  {~pfe}
  {~ib} procedure Initial;begin
 {~f}init(Future,'Future');
 setstart(5000);
{/Fac1}  init(Fac1,' Fac1');
{/Que1}  init(Que1,' Que1');
{=Gen} Future.NewWaitProc(Gen,exponential(100),0,Gen,Gen+1);
  {~ie}end;
  {~mtb}procedure ModelTxt;begin case ActiveBlock of
{=Gen} ::Gen *:Future.NewWait(exponential(100),0,Gen,Gen+1);
    *: test([que1.L<10,true],[!*,exit]) ;
    *: que1.Queue ;
    *: fac1.seize ;
    *: que1.depart ;
    *:Future.Wait(MeanDev(110,30));
    *: fac1.release ;
    *: terminate(1) ;
::exit     *: terminate ;
  {~mte} end;end;
  {~mb} procedure Simulation;  begin
```

```
  start(GetTerm);
 {~me}end;
 {~rb} procedure Report;begin
 {~re} end;
 {~cab} procedure CloseAllObj;begin
 {~cae} end;
```

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 из 15 имеющихся. Если таковое устройство не найдется, то использовать последнее. Определить общее и среднее время обслуживания 2000 заявок.

При решении задач, требующих массивов, разработчик модели непременно должен ответить себе на вопрос, как следует поступать, если в массиве не хватит элементов. Иначе модель может вести себя совершенно непредсказуемым образом. Впрочем, эта проблема есть в любых языках программирования.

Здесь понадобится использовать массивы устройств и очередей. Для создания массивов объектов следует вначале вставить простой объект. Затем в разделе констант (Const) следует определить размер массива, по образцу.

```
 num=15;
```

Здесь num – это имя константы, а 15 - количество элементов массива. Далее, в описании следует найти нужный объект и изменить его по образцу.

```
{/fac0} fac0 Tfacility; - заменить на
{/fac0} fac0:array[1..num] of Tfacility;
```

Аналогичным образом нужно изменить и описание очереди, получив.

```
{/que0} que0:array[1..num] of Tqueue;
```

Таким образом, мы получим описание 15 устройств и очередей.

Затем в процедуре initial строку

```
 {/sh} procedure initial; begin  - заменить на
 {/sh} procedure initial;  var  i:integer; begin
```

а строки

```
{/fac0}  .init(fac0);
{/que0} init(que0); - заменить на
{/fac0}  for i:=1 to num do init(fac0[i], 'fac0'+inttostr(i));
{/que0}  for i:=1 to num do init(que0[i], 'que0'+inttostr(i));
```

Таким образом, мы построим циклы создания 15 устройств и очередей.

Подобным образом нужно поступать всегда, когда вы создаете массивы объектов.

Доступ к элементам массива мы получаем, используя конструкцию NameObj[Num].Name

Здесь NameObj – имя объекта, Num – его номер, Name – имя процедуры или функции. Именно так мы работали с массивами в вышеуказанных процедурах.

В итоге, текст модели может выглядеть следующим образом.

```
Const
num=15;
  {~vb} Var
 {~f}Future:TFuture;
{/fac0} fac0:array[1..num] of Tfacility;
{/que0} que0:array[1..num] of Tqueue;
  {~ve}
  {~pfe}
  {~ib} procedure Initial;  var  i:integer; begin
 {~f}init(Future,'Future');
   setstart(2000);
{/fac0}  for i:=1 to num do init(fac0[i], 'fac0'+inttostr(i));
{/que0}  for i:=1 to num do init(que0[i], 'que0'+inttostr(i));
{=Gen} Future.NewWaitProc(Gen,exponential(100),0,Gen,Gen+1);
  {~ie}end;
  {~mtb}procedure ModelTxt;begin case ActiveBlock of
{=Gen} ::Gen *:Future.NewWait(exponential(100),0,Gen,Gen+1);
   *: IPlet(1,[0]);
::rret  *: IPlet(1,[ip(1)+1]);
      *: test([(fac0[IP(1)].L=0)or(IP(1)=num),true],[!*,rret]) ;
   *: que0[IP(1)].Queue;
   *: fac0[IP(1)].Seize;
   *: que0[IP(1)].Depart;
   *:Future.Wait( MeanDev(500+40*ip(1),0.3*(500+40*ip(1))));
   *: fac0[IP(1)].Release;
   *: Terminate(1);
  {~mte} end;end;
  {~mb} procedure Simulation;  begin
   start(GetTerm);
  {~me}end;
  {~rb} procedure Report;  begin
  {~re} end;
  {~cab} procedure CloseAllObj;begin
  {~cae} end;
```

В данном случае, выходная статистика представляет интерес. Она, в частности, свидетельствует, что разные устройства имеют разное время задержки, и что в данном случае последнее устройство – единственное, которое имеет непустую очередь.

```
 Date Time Stamp= 18.12.2008 17:16:17   Ctrl Click |Name - HyperLink `R
E:\work\ObjectClient\Object No Sys\Future\GPSS-Future\book\tmp26.rtf
StartTime         0.00000 EndTime     198679.86368
 |Model |Blocks |Values |CurrentList |SynchronizeList |InterruptList
 |RPList |IPList |SPList |BPList
   Num      NameObject
     2        |Future
     3         |fac01
     4         |fac02
     5         |fac03
     6         |fac04
     7         |fac05
     8         |fac06
     9         |fac07
```

```
    10         |fac08
    11         |fac09
    12        |fac010
    13        |fac011
    14        |fac012
    15        |fac013
    16        |fac014
    17        |fac015
    18         |que01
    19         |que02
    20         |que03
    21         |que04
    22         |que05
    23         |que06
    24         |que07
    25         |que08
    26         |que09
    27        |que010
    28        |que011
    29        |que012
    30        |que013
    31        |que014
    32        |que015
 |R `Model
 {=Gen} ::Gen 1:Future.NewWait(exponential(100),0,Gen,Gen+1);
   2: IPlet(1,[0]);
::rret  3: IPlet(1,[ip(1)+1]);
      4: test([(fac0[IP(1)].L=0)or(IP(1)=num),true],[5,rret]) ;
   5: que0[IP(1)].Queue;
   6: fac0[IP(1)].Seize;
   7: que0[IP(1)].Depart;
   8:Future.Wait( MeanDev(500+40*ip(1),0.3*(500+40*ip(1))));
   9: fac0[IP(1)].Release;
   10: Terminate(1);

 |R `Blocks    BLOCK Report
    Location     Total   Current
         1      2003         0
         2      2003         0
         3      9794         0
         4      9794         0
         5      2003         0
         6      2003         0
         7      2003         0
         8      2003         3
         9      2000         0
        10      2000         0
 |R `Values
 |R `IPList Report ( Integer ) Count= 3
        Tran            Num/Name        Integer
        2001                  1              2
        2002                  1              3
        2003                  1              4
 |R Facility = `fac01 Entries = 313 Current = 0 Utility =  0.84817 AverTime = 538.38536
    Tran = 0 IntTran = 0 Active Time = 198679.86368 NextGo = 0
 |R Facility = `fac02 Entries = 283 Current = 1 Utility =  0.81051 AverTime = 569.02009
    Tran = 2001 IntTran = 0 Active Time = 198679.86368 NextGo = 0
 |R Facility = `fac03 Entries = 245 Current = 1 Utility =  0.77056 AverTime = 624.87424
    Tran = 2002 IntTran = 0 Active Time = 198679.86368 NextGo = 0
 |R Facility = `fac04 Entries = 220 Current = 1 Utility =  0.72506 AverTime = 654.79686
    Tran = 2003 IntTran = 0 Active Time = 198679.86368 NextGo = 0
 |R Facility = `fac05 Entries = 195 Current = 0 Utility =  0.69201 AverTime = 705.06528
    Tran = 0 IntTran = 0 Active Time = 198679.86368 NextGo = 0
 |R Facility = `fac06 Entries = 173 Current = 0 Utility =  0.63391 AverTime = 728.00966
    Tran = 0 IntTran = 0 Active Time = 198679.86368 NextGo = 0
 |R Facility = `fac07 Entries = 144 Current = 0 Utility =  0.57684 AverTime = 795.88082
    Tran = 0 IntTran = 0 Active Time = 198679.86368 NextGo = 0
 |R Facility = `fac08 Entries = 121 Current = 0 Utility =  0.50116 AverTime = 822.88880
    Tran = 0 IntTran = 0 Active Time = 198679.86368 NextGo = 0
 |R Facility = `fac09 Entries = 97 Current = 0 Utility =  0.41869 AverTime = 857.57727
```

```
   Tran = 0 IntTran = 0 Active Time = 198679.86368 NextGo = 0
|R Facility = `fac010 Entries = 74 Current = 0 Utility =  0.34402 AverTime = 923.65175
   Tran = 0 IntTran = 0 Active Time = 198679.86368 NextGo = 0
|R Facility = `fac011 Entries = 53 Current = 0 Utility =  0.24521 AverTime = 919.20457
   Tran = 0 IntTran = 0 Active Time = 198679.86368 NextGo = 0
|R Facility = `fac012 Entries = 32 Current = 0 Utility =  0.16445 AverTime = 1021.02238
   Tran = 0 IntTran = 0 Active Time = 198679.86368 NextGo = 0
|R Facility = `fac013 Entries = 25 Current = 0 Utility =  0.13059 AverTime = 1037.81078
   Tran = 0 IntTran = 0 Active Time = 198679.86368 NextGo = 0
|R Facility = `fac014 Entries = 14 Current = 0 Utility =  0.07290 AverTime = 1034.51452
   Tran = 0 IntTran = 0 Active Time = 198679.86368 NextGo = 0
|R Facility = `fac015 Entries = 14 Current = 0 Utility =  0.07796 AverTime = 1106.40893
   Tran = 0 IntTran = 0 Active Time = 198679.86368 NextGo = 0
|R Queue = `que01 Entries = 313 Current = 0 AverQueue =  0.00000 AverTime =  0.00000
   AverTime(-Ze) =  0.00000 Max = 1 Min = 0 Zero = 313 Active Time = 198679.86368 NextGo = 0
|R Queue = `que02 Entries = 283 Current = 0 AverQueue =  0.00000 AverTime =  0.00000
   AverTime(-Ze) =  0.00000 Max = 1 Min = 0 Zero = 283 Active Time = 198679.86368 NextGo = 0
|R Queue = `que03 Entries = 245 Current = 0 AverQueue =  0.00000 AverTime =  0.00000
   AverTime(-Ze) =  0.00000 Max = 1 Min = 0 Zero = 245 Active Time = 198679.86368 NextGo = 0
|R Queue = `que04 Entries = 220 Current = 0 AverQueue =  0.00000 AverTime =  0.00000
   AverTime(-Ze) =  0.00000 Max = 1 Min = 0 Zero = 220 Active Time = 198679.86368 NextGo = 0
|R Queue = `que05 Entries = 195 Current = 0 AverQueue =  0.00000 AverTime =  0.00000
   AverTime(-Ze) =  0.00000 Max = 1 Min = 0 Zero = 195 Active Time = 198679.86368 NextGo = 0
|R Queue = `que06 Entries = 173 Current = 0 AverQueue =  0.00000 AverTime =  0.00000
   AverTime(-Ze) =  0.00000 Max = 1 Min = 0 Zero = 173 Active Time = 198679.86368 NextGo = 0
|R Queue = `que07 Entries = 144 Current = 0 AverQueue =  0.00000 AverTime =  0.00000
   AverTime(-Ze) =  0.00000 Max = 1 Min = 0 Zero = 144 Active Time = 198679.86368 NextGo = 0
|R Queue = `que08 Entries = 121 Current = 0 AverQueue =  0.00000 AverTime =  0.00000
   AverTime(-Ze) =  0.00000 Max = 1 Min = 0 Zero = 121 Active Time = 198679.86368 NextGo = 0
|R Queue = `que09 Entries = 97 Current = 0 AverQueue =  0.00000 AverTime =  0.00000
   AverTime(-Ze) =  0.00000 Max = 1 Min = 0 Zero = 97 Active Time = 198679.86368 NextGo = 0
|R Queue = `que010 Entries = 74 Current = 0 AverQueue =  0.00000 AverTime =  0.00000
   AverTime(-Ze) =  0.00000 Max = 1 Min = 0 Zero = 74 Active Time = 198679.86368 NextGo = 0
|R Queue = `que011 Entries = 53 Current = 0 AverQueue =  0.00000 AverTime =  0.00000
   AverTime(-Ze) =  0.00000 Max = 1 Min = 0 Zero = 53 Active Time = 198679.86368 NextGo = 0
|R Queue = `que012 Entries = 32 Current = 0 AverQueue =  0.00000 AverTime =  0.00000
   AverTime(-Ze) =  0.00000 Max = 1 Min = 0 Zero = 32 Active Time = 198679.86368 NextGo = 0
|R Queue = `que013 Entries = 25 Current = 0 AverQueue =  0.00000 AverTime =  0.00000
   AverTime(-Ze) =  0.00000 Max = 1 Min = 0 Zero = 25 Active Time = 198679.86368 NextGo = 0
|R Queue = `que014 Entries = 14 Current = 0 AverQueue =  0.00000 AverTime =  0.00000
   AverTime(-Ze) =  0.00000 Max = 1 Min = 0 Zero = 14 Active Time = 198679.86368 NextGo = 0
|R Queue = `que015 Entries = 14 Current = 0 AverQueue =  0.06539 AverTime = 927.93755
   AverTime(-Ze) = 3247.78142 Max = 4 Min = 0 Zero = 10 Active Time = 198679.86368 NextGo = 0
|R User/Future = `Future Entries = 4007 Current = 4 AverUser =  7.51509 AverTime = 372.62196
   Max = 16 Min = 0 Active Time = 198679.86368 NextGo = 0
|R Report  User/Future List Count= 4
      Tran     Assem  Priority   Current      Next       TimeStamp       SortValue     Group
Inter
      2001      2001         0         8         9    198429.34092    198954.40736         0
FALSE
      2004      2004         0         0         1    199042.32340    199042.32340         0
FALSE
      2002      2002         0         8         9    198491.08758    199237.00028         0
FALSE
      2003      2003         0         8         9    198568.08500    199332.47729         0
FALSE
            Num/Name        Integer  |R `Future      Report IP LIST  Count= 2
I                  1           2004
I                  2           2004
```

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

```
Const
num=15;
 {~vb} Var
 {~f}Future:TFuture;
{/fac0} fac0:array[1..num] of Tfacility;
```

```
{/que0} que0:array[1..num] of Tqueue;
  {~ve}
  {~pfe}
  {~ib} procedure Initial;  var  i:integer; begin
 {~f}init(Future,'Future');
   setstart(2000);
{/fac0}  for i:=1 to num do init(fac0[i], 'fac0'+inttostr(i));
{/que0}  for i:=1 to num do init(que0[i], 'que0'+inttostr(i));
{=Gen0} Future.NewWaitProc(Gen0,exponential(100),0,Gen0,Gen0+1);
  {~ie}end;
  {~mtb}procedure ModelTxt;begin case ActiveBlock of
{=Gen0} ::Gen0 *:Future.NewWait(exponential(100),0,Gen0,Gen0+1);
   *: IPlet(1,[0]);
::rret  *: IPlet(1,[ip(1)+1]);
      *: test([(que0[IP(1)].L<4)or(IP(1)=num),true],[!*,rret]) ;
   *: que0[IP(1)].Queue;
   *: fac0[IP(1)].Seize;
   *: que0[IP(1)].Depart;
 *:Future.Wait(MeanDev(500+40*ip(1),0.3*(500+40*ip(1))));
   *: fac0[IP(1)].Release;
   *: Terminate(1);
  {~mte} end;end;
  {~mb} procedure Simulation;  begin
   start(GetTerm);
  {~me}end;
  {~rb} procedure Report;   begin
  {~re} end;
  {~cab} procedure CloseAllObj;begin
  {~cae} end;
```

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

```
Date Time Stamp= 18.12.2008 17:21:45   Ctrl Click |Name - HyperLink `R
E:\work\ObjectClient\Object No Sys\Future\GPSS-Future\book\tmp27.rtf
StartTime         0.00000 EndTime     201795.08179
 |Model |Blocks |Values |CurrentList |SynchronizeList |InterruptList
 |RPList |IPList |SPList |BPList
   Num      NameObject
     2        |Future
     3         |fac01
     4         |fac02
     5         |fac03
     6         |fac04
     7         |fac05
     8         |fac06
     9         |fac07
    10         |fac08
    11         |fac09
    12        |fac010
    13        |fac011
    14        |fac012
    15        |fac013
    16        |fac014
    17        |fac015
    18         |que01
```

```
    19         |que02
    20         |que03
    21         |que04
    22         |que05
    23         |que06
    24         |que07
    25         |que08
    26         |que09
    27        |que010
    28        |que011
    29        |que012
    30        |que013
    31        |que014
    32        |que015
 |R `Model
 {=Gen0} ::Gen0 1:Future.NewWait(exponential(100),0,Gen0,Gen0+1);
   2: IPlet(1,[0]);
::rret  3: IPlet(1,[ip(1)+1]);
       4: test([(que0[IP(1)].L<4)or(IP(1)=num),true],[5,rret]) ;
   5: que0[IP(1)].Queue;
   6: fac0[IP(1)].Seize;
   7: que0[IP(1)].Depart;
 8:Future.Wait(MeanDev(500+40*ip(1),0.3*(500+40*ip(1))));
   9: fac0[IP(1)].Release;
   10: Terminate(1);

 |R `Blocks    BLOCK Report
    Location     Total   Current
         1      2030         0
         2      2030         0
         3      7198         0
         4      7198         0
         5      2030        24
         6      2006         0
         7      2006         0
         8      2006         6
         9      2000         0
        10      2000         0
 |R `Values
 |R `CurrentList Report Current Transactions List  Count= 24
      Tran     Assem  Priority   Current      Next       TimeStamp       TimeEnter Inter
      2003      2003         0         5         6    199448.51289    199448.51289 FALSE
      2004      2004         0         5         6    199540.79947    199540.79947 FALSE
      2005      2005         0         5         6    199747.47528    199747.47528 FALSE
      2006      2006         0         5         6    199804.26289    199804.26289 FALSE
      2009      2009         0         5         6    199928.46186    199928.46186 FALSE
      2010      2010         0         5         6    200068.82563    200068.82563 FALSE
      2011      2011         0         5         6    200093.73221    200093.73221 FALSE
      2012      2012         0         5         6    200130.40975    200130.40975 FALSE
      2013      2013         0         5         6    200160.26937    200160.26937 FALSE
      2015      2015         0         5         6    200317.70399    200317.70399 FALSE
      2016      2016         0         5         6    200600.60696    200600.60696 FALSE
      2017      2017         0         5         6    200803.87438    200803.87438 FALSE
      2018      2018         0         5         6    200944.73636    200944.73636 FALSE
      2019      2019         0         5         6    200995.24456    200995.24456 FALSE
      2020      2020         0         5         6    201016.22156    201016.22156 FALSE
      2021      2021         0         5         6    201069.26087    201069.26087 FALSE
      2022      2022         0         5         6    201168.78379    201168.78379 FALSE
      2023      2023         0         5         6    201169.49324    201169.49324 FALSE
      2024      2024         0         5         6    201273.01806    201273.01806 FALSE
      2025      2025         0         5         6    201296.80494    201296.80494 FALSE
      2026      2026         0         5         6    201459.82132    201459.82132 FALSE
      2028      2028         0         5         6    201549.30179    201549.30179 FALSE
      2029      2029         0         5         6    201579.81881    201579.81881 FALSE
      2030      2030         0         5         6    201586.59675    201586.59675 FALSE
 |R `IPList Report ( Integer ) Count= 30
        Tran            Num/Name        Integer
        1999                   1              4
        2000                   1              5
        2001                   1              2
```

```
        2002                    1               3
        2003                    1               1
        2004                    1               4
        2005                    1               5
        2006                    1               2
        2008                    1               6
        2009                    1               1
        2010                    1               3
        2011                    1               4
        2012                    1               6
        2013                    1               6
        2015                    1               5
        2016                    1               1
        2017                    1               2
        2018                    1               3
        2019                    1               2
        2020                    1               4
        2021                    1               3
        2022                    1               1
        2023                    1               5
        2024                    1               6
        2025                    1               6
        2026                    1               4
        2027                    1               7
        2028                    1               7
        2029                    1               7
        2030                    1               7
|R Facility = `fac01 Entries = 372 Current = 0 Utility =  0.99934 AverTime = 542.10354
   Tran = 0 IntTran = 0 Active Time = 201795.08179 NextGo = 0
|R Facility = `fac02 Entries = 348 Current = 1 Utility =  0.99896 AverTime = 579.26681
   Tran = 2001 IntTran = 0 Active Time = 201795.08179 NextGo = 0
|R Facility = `fac03 Entries = 325 Current = 1 Utility =  0.99482 AverTime = 617.69358
   Tran = 2002 IntTran = 0 Active Time = 201795.08179 NextGo = 0
|R Facility = `fac04 Entries = 305 Current = 1 Utility =  0.99204 AverTime = 656.35530
   Tran = 1999 IntTran = 0 Active Time = 201795.08179 NextGo = 0
|R Facility = `fac05 Entries = 284 Current = 1 Utility =  0.96202 AverTime = 683.55858
   Tran = 2000 IntTran = 0 Active Time = 201795.08179 NextGo = 0
|R Facility = `fac06 Entries = 220 Current = 1 Utility =  0.79940 AverTime = 733.24811
   Tran = 2008 IntTran = 0 Active Time = 201795.08179 NextGo = 0
|R Facility = `fac07 Entries = 116 Current = 1 Utility =  0.45227 AverTime = 786.77075
   Tran = 2027 IntTran = 0 Active Time = 201795.08179 NextGo = 0
|R Facility = `fac08 Entries = 36 Current = 0 Utility =  0.14374 AverTime = 805.74071
   Tran = 0 IntTran = 0 Active Time = 201795.08179 NextGo = 0
|R Facility = `fac09 Entries = 0 Current = 0 Utility =  0.00000 AverTime =  0.00000
   Tran = 0 IntTran = 0 Active Time = 201795.08179 NextGo = 0
|R Facility = `fac010 Entries = 0 Current = 0 Utility =  0.00000 AverTime =  0.00000
   Tran = 0 IntTran = 0 Active Time = 201795.08179 NextGo = 0
|R Facility = `fac011 Entries = 0 Current = 0 Utility =  0.00000 AverTime =  0.00000
   Tran = 0 IntTran = 0 Active Time = 201795.08179 NextGo = 0
|R Facility = `fac012 Entries = 0 Current = 0 Utility =  0.00000 AverTime =  0.00000
   Tran = 0 IntTran = 0 Active Time = 201795.08179 NextGo = 0
|R Facility = `fac013 Entries = 0 Current = 0 Utility =  0.00000 AverTime =  0.00000
   Tran = 0 IntTran = 0 Active Time = 201795.08179 NextGo = 0
|R Facility = `fac014 Entries = 0 Current = 0 Utility =  0.00000 AverTime =  0.00000
   Tran = 0 IntTran = 0 Active Time = 201795.08179 NextGo = 0
|R Facility = `fac015 Entries = 0 Current = 0 Utility =  0.00000 AverTime =  0.00000
   Tran = 0 IntTran = 0 Active Time = 201795.08179 NextGo = 0
|R Queue = `que01 Entries = 376 Current = 4 AverQueue =  3.82335 AverTime = 2051.94845
   AverTime(-Ze) = 2057.42032 Max = 4 Min = 0 Zero = 1 Active Time = 201795.08179 NextGo = 0
|R Queue = `que02 Entries = 351 Current = 3 AverQueue =  3.74054 AverTime = 2150.49168
   AverTime(-Ze) = 2156.63595 Max = 4 Min = 0 Zero = 1 Active Time = 201795.08179 NextGo = 0
|R Queue = `que03 Entries = 328 Current = 3 AverQueue =  3.61824 AverTime = 2226.04394
   AverTime(-Ze) = 2232.85142 Max = 4 Min = 0 Zero = 1 Active Time = 201795.08179 NextGo = 0
|R Queue = `que04 Entries = 309 Current = 4 AverQueue =  3.41595 AverTime = 2230.81288
   AverTime(-Ze) = 2238.05578 Max = 4 Min = 0 Zero = 1 Active Time = 201795.08179 NextGo = 0
|R Queue = `que05 Entries = 287 Current = 3 AverQueue =  2.85285 AverTime = 2005.89444
   AverTime(-Ze) = 2116.51362 Max = 4 Min = 0 Zero = 15 Active Time = 201795.08179 NextGo = 0
|R Queue = `que06 Entries = 224 Current = 4 AverQueue =  1.85059 AverTime = 1667.14456
   AverTime(-Ze) = 1895.63645 Max = 4 Min = 0 Zero = 27 Active Time = 201795.08179 NextGo = 0
|R Queue = `que07 Entries = 119 Current = 3 AverQueue =  0.90149 AverTime = 1528.71605
```

```
   AverTime(-Ze) = 1977.36098 Max = 4 Min = 0 Zero = 27 Active Time = 201795.08179 NextGo = 0
|R Queue = `que08 Entries = 36 Current = 0 AverQueue =  0.15367 AverTime = 861.39773
   AverTime(-Ze) = 1409.55992 Max = 4 Min = 0 Zero = 14 Active Time = 201795.08179 NextGo = 0
|R Queue = `que09 Entries = 0 Current = 0 AverQueue =  0.00000 AverTime =  0.00000
   AverTime(-Ze) =  0.00000 Max = 0 Min = 0 Zero = 0 Active Time = 201795.08179 NextGo = 0
|R Queue = `que010 Entries = 0 Current = 0 AverQueue =  0.00000 AverTime =  0.00000
   AverTime(-Ze) =  0.00000 Max = 0 Min = 0 Zero = 0 Active Time = 201795.08179 NextGo = 0
|R Queue = `que011 Entries = 0 Current = 0 AverQueue =  0.00000 AverTime =  0.00000
   AverTime(-Ze) =  0.00000 Max = 0 Min = 0 Zero = 0 Active Time = 201795.08179 NextGo = 0
|R Queue = `que012 Entries = 0 Current = 0 AverQueue =  0.00000 AverTime =  0.00000
   AverTime(-Ze) =  0.00000 Max = 0 Min = 0 Zero = 0 Active Time = 201795.08179 NextGo = 0
|R Queue = `que013 Entries = 0 Current = 0 AverQueue =  0.00000 AverTime =  0.00000
   AverTime(-Ze) =  0.00000 Max = 0 Min = 0 Zero = 0 Active Time = 201795.08179 NextGo = 0
|R Queue = `que014 Entries = 0 Current = 0 AverQueue =  0.00000 AverTime =  0.00000
   AverTime(-Ze) =  0.00000 Max = 0 Min = 0 Zero = 0 Active Time = 201795.08179 NextGo = 0
|R Queue = `que015 Entries = 0 Current = 0 AverQueue =  0.00000 AverTime =  0.00000
   AverTime(-Ze) =  0.00000 Max = 0 Min = 0 Zero = 0 Active Time = 201795.08179 NextGo = 0
|R User/Future = `Future Entries = 4037 Current = 7 AverUser =  6.82376 AverTime = 341.09503
   Max = 9 Min = 0 Active Time = 201795.08179 NextGo = 0
|R Report  User/Future List Count= 7
     Tran     Assem  Priority   Current      Next       TimeStamp       SortValue     Group
Inter
     2008      2008         0         8         9    199807.56129    201841.60922         0
FALSE
     2031      2031         0         0         1    201956.17476    201956.17476         0
FALSE
     1999      1999         0         8         9    198817.73663    202018.47789         0
FALSE
     2001      2001         0         8         9    199203.03881    202143.41730         0
FALSE
     2002      2002         0         8         9    199246.97726    202207.68908         0
FALSE
     2027      2027         0         8         9    201537.85004    202277.19891         0
FALSE
     2000      2000         0         8         9    199047.80069    202429.04526         0
FALSE
            Num/Name        Integer  |R `Future      Report IP LIST  Count= 2
I                  1           2031
I                  2           2031
```

Пример применения блока BUFFER в сочетании с 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.

```
 {~vb} Var
{~f}Future:TFuture;
 {/stor} stor:Tstorage;
 num:Integer;
 key:Boolean;
 {~ve}
 {~pfe}
 {~ib} procedure Initial;begin
{~f}init(Future,'Future');
setstart( 1000);
```

```
  {/stor}  init(stor,' stor',11);
  Num:=0;
   Key:=false;
{=Gen1} Future.NewWaitProc(Gen1,exponential(100),0,Gen1,Gen1+1);
{=Gen2} Future.NewWaitProc(Gen2,10000,0,Gen2,Gen2+1);
  {~ie}end;
  {~mtb}procedure ModelTxt;begin  case ActiveBlock of
{=Gen1} ::Gen1 *:Future.NewWait(exponential(100),0,Gen1,Gen1+1);
    *:test([num=10,true],[!*,next]);
    *:let(key,true);
    *:buffer;
    *:let(key,false);
::next *:let( num,num+1);
    *:test([key],[!*]);
    *:test([stor.L=0,true],[!*,outl]);
    *:buffer;
    *:stor.enter;
    *:let( num,num-1);
   *:Future.Wait(MeanDev(900,300));
    *:stor.leave;
    *:terminate( 1);
::outl *:let( num,num-1);
     *:tovalue(1,Total(!*-1));
     *: terminate;
{=Gen2} ::Gen2 *:Future.NewWait(10000,0,Gen2,Gen2+1);
     *:currentblocks;
     *: terminate;
  {~mte} end;end;
  {~mb} procedure Simulation;  begin
 {/sh}     start(GetTerm);
  valueTitle(1,'stor');
  {~me}end;
  {~rb} procedure Report;begin
{/num}  reportval(num,'num = ');
{/key}  reportval(key,'key = ');
  {~re} end;
  {~cab} procedure CloseAllObj;begin
  {~cae} end;
```

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 *:test([stor.L=0,true],[!*,outl]); . 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 Test. 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 Test, 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:

```
        Date Time Stamp= 18.12.2008 17:28:22   Ctrl Click |Name - HyperLink `R
E:\work\ObjectClient\Object No Sys\Future\GPSS-Future\book\tmp28.rtf
StartTime         0.00000 EndTime     176787.01225
 |Model |Blocks |Values |CurrentList |SynchronizeList |InterruptList
 |RPList |IPList |SPList |BPList
   Num      NameObject
     2        |Future
     3         | stor
 |R `Model
 {=Gen1} ::Gen1 1:Future.NewWait(exponential(100),0,Gen1,Gen1+1);
    2:test([num=10,true],[3,next]);
    3:let(key,true);
    4:buffer;
    5:let(key,false);
::next 6:let( num,num+1);
    7:test([key],[8]);
    8:test([stor.L=0,true],[9,outl]);
    9:buffer;
    10:stor.enter;
    11:let( num,num-1);
   12:Future.Wait(MeanDev(900,300));
    13:stor.leave;
    14:terminate( 1);
::outl 15:let( num,num-1);
     16:tovalue(1,Total(17-1));
     17: terminate;
{=Gen2} ::Gen2 18:Future.NewWait(10000,0,Gen2,Gen2+1);
     19:currentblocks;
     20: terminate;

 |R `Blocks    BLOCK Report
    Location     Total   Current
         1      1688         0
         2      1688         0
         3       168         0
         4       168         0
         5       168         0
         6      1688         8
         7      1680         0
         8      1680         0
         9      1000         0
        10      1000         0
        11      1000         0
        12      1000         0
        13      1000         0
        14      1000         0
        15       680         0
        16       680         0
        17       680         0
        18        17         0
        19        17         0
        20        17         0
 |R `Values
num =          8
key =      FALSE
 |R `CurrentList Report Current Transactions List  Count= 8
      Tran     Assem  Priority   Current      Next       TimeStamp       TimeEnter Inter
      1699      1699         0         6         7    175589.02723    175589.02723 FALSE
      1700      1700         0         6         7    175746.21822    175746.21822 FALSE
      1701      1701         0         6         7    175941.86043    175941.86043 FALSE
      1702      1702         0         6         7    176014.78001    176014.78001 FALSE
```

```
      1703      1703         0         6         7    176119.88750    176119.88750 FALSE
      1704      1704         0         6         7    176322.43341    176322.43341 FALSE
      1705      1705         0         6         7    176359.64065    176359.64065 FALSE
      1706      1706         0         6         7    176709.81571    176709.81571 FALSE
 |R Storage = ` stor Entries = 1000 Current = 0 Average =  5.08773 AverTime = 899.44522 Utility
=  0.46252
    Max = 10 Min = 0 Tran = 1692 ACapac = 11.00000 Capacity = 11 Active Time = 176787.01225
NextGo = 0
 |R User/Future = `Future Entries = 2707 Current = 2 AverUser =  6.85066 AverTime = 447.39873
    Max = 12 Min = 0 Active Time = 176787.01225 NextGo = 0
 |R Report  User/Future List Count= 2
      Tran     Assem  Priority   Current      Next       TimeStamp       SortValue     Group
Inter
      1707      1707         0         0         1    176804.91681    176804.91681         0
FALSE
      1658      1658         0         0        18    180000.00000    180000.00000         0
FALSE
            Num/Name        Integer  |R `Future      Report IP LIST  Count= 4
I                 1           1689
I                 2           1707
I                18             18
I                19           1658
```

Более сложный пример использования блока Test, представляет задача о работе порта. Здесь используются функции, результат которых зависит от номера, или от условия.

Модель порта, постановка задачи.

В порту есть три причала: 1, 2 и 3 (Berth1, Berth2 и Berth3).

В любой заданный момент времени к причалу 1 могут пришвартовываться 2 маленьких корабля или один средний.

И причал 2, и причал 3 могут обслуживать один большой корабль, два средних или четыре маленьких.

Время между приходами кораблей равно 26 часам, оно распределено экспоненциально, маленькие, средние и большие корабли приходят в порт в пропорции 5:3:2 соответственно. Очереди для причалов основываются на том, что первый прибывший обслуживается первым, за исключением того, что ни маленький, ни средний корабль не может пришвартовываться к причалу, который уже ждет большой корабль, у средних кораблей больший приоритет, чем у маленьких кораблей.

Время разгрузки экспоненциально распределено со средним временем для маленьких кораблей - 15 часов, для средних - 30 часов, для больших - 45 часов. Время погрузки следующее:

Время маленьких кораблей равномерно распределено в течение 24±6 часов.

Время средних кораблей равномерно распределено в течение 36±10 часов. Время больших кораблей равномерно распределено в течение 56±12 часов.

Большие корабли могут причалить или отчалить от причалов 2 и 3 только во время прилива. 2 и 3. Отлив длится 3 часа, прилив - 10 часов. Необходимо:

1. Запустить процесс моделирования для 500 дней.
2. Определить распределение транзитного времени для каждого типа кораблей.

3. Определить коэффициенты использования трех причалов. Решение данной задачи достаточно сложно, и поэтому текст модели пересыщен комментариями.
Листинг модели достаточно близок к решению на классическом GPSS, и, тем не менее, он заметно проще него. Решение основано на модели Джерарда Каммингса (Gerard F. Cummings)

При решении этой задачи причалы фигурируют в двух качествах. Как требования на занятие или освобождение причала, и как реальные причалы.

Причалы как требования занимаются до реальной возможности причалить в связи с возможным отливом. А освобождаются – только после реального ухода корабля с учетом возможного отлива. Остальные подробности построения модели более или менее очевидны.

```
Const
// Константы нужны для того, чтобы использовать их имена в качестве
 // аргумента для P- параметров.
 Size=1;
 Capacity=2;
 Quenum=3;
 M_Unload=4;
 M_Load=5;
 Loadsp=6;
Berth_Num=7;
  {~vb} Var
 {~f}Future:TFuture;
{/Berth} Berth:ARRAY[1..3] OF Tstorage; // Реальный причал.
  Tide:Boolean;  // прилив есть.
{/tab} tab:ARRAY[1..3]  OF  Ttable;
{/que} que:ARRAY[1..3]  OF  Tqueue;
{/Berth1} Berth1:ARRAY[1..3] of Tstorage; // Причал, как заказ.
I:integer;
  {~ve}
  {~pfe}
  {~ib} procedure Initial;begin
 {~f}init(Future,'Future');
   SETSTART(5000);
{/Berth} for i:=1 to 3 do  init(Berth[i], 'Berth '+inttostr(i),ibynum(i,[2,4,4])); // Задание
емкости реального причала.
  Tide:=false;
{/tab}  for i:=1 to 3 do init(tab[i], 'tab '+inttostr(i),0,10,100);
{/que}  for i:=1 to 3 do  init(que[i], ' que '+inttostr(i));
{/Berth1}  for i:=1 to 3 do  init(Berth1[i] , 'Berth1 '+inttostr(i),ibynum(i,[2,4,4])); // Задание
емкости причала как заказа.
{=gener0} Future.NewWaitProc(gener0,24,0,gener0,gener0+1);
{=ge1} Future.NewWaitProc(ge1,0,0,ge1,ge1+1);
{=ge2} Future.NewWaitProc(ge2,Exponential(26),0,ge2,ge2+1);
  {~ie}end;
  {~mtb}procedure ModelTxt;begin case ActiveBlock of
```

```
// * Таймер дня
{=gener0} ::gener0 *:Future.NewWait(24,0,gener0,gener0+1);
        *:TERMINATE( 1) ; //Часы работают один раз в день.
{=ge1} ::ge1 *:Future.NewWait(0,0,ge1,ge1+1,0,true,1);
::Again  *: LET(Tide,FALSE) ; // ;Циклическая заявка моделирует прилив.
           *:Future.Wait(3); // Отлив длится 3 часа.
       *: LET(Tide,TRUE) ; // Прилив начинается.
        *:Future.Wait(10); // Полная вода длится 10 часов.
       *:TRANSFER(Again) ; // Заявка возвращается к 'Again'.

{=ge2} ::ge2 *:Future.NewWait(Exponential(26),0,ge2,ge2+1);  //Каждые 26 часов
появляется корабль.
       *:IPLET( Size,[IBYBOOL([RANDOM<=0.5,RANDOM<=0.6,true],[1,2,3])]); // тип
корабля в соответствии с вероятностями.
      *:IPLET( Capacity,[ibynum(Ip(size),[1,2,4] )]);  // емкость корабля
      *:PRIORITY(Ip(size)); // приоритет
      *:IPLET( Quenum,[Ip(size)]) ; // Очередь  для  корабля.
      *:RPLET( M_Unload,[ibynum(Ip(size),[15,30,45])]) ; //Среднее время разгрузки.
      *:RPLET( M_Load,[ibynum(Ip(size),[24,36,56])]) ; //Среднее время погрузки.
      *:RPLET( Loadsp,[ibynum(Ip(size),[6,10,12])]); // Разброс времени погрузки.
      *:que[ IP(Quenum)].QUEUE ;
      *:test([(berth1[1].r>=IP(Capacity))or(berth1[2].r>=
IP(Capacity))or(berth1[3].r>=IP(Capacity))],[!*]); // Проверка, есть ли место для корабля на
причале?
      *:IPLET( Berth_Num,[IBYBOOL([berth1[1].r>=IP(Capacity),
berth1[2].r>=IP(Capacity),berth1[3].r>=IP(Capacity)],[1,2,3])])  ; // Определяется свободный
причал.
      *:Berth1[IP(Berth_num)].ENTER (IP(Capacity)); // Занимаем причал с помощью
емкости корабля, как заказ.
        *:TEST([Ip(size)=3,true],[!*,Skipit1]) ; // Большой корабль?  Если переключатель
установлен, то прилив
       *:TEST([TIDE],[!*]) ; //Ожидается прилив.
::skipit1         *:Berth[IP(Berth_num)].ENTER (IP(Capacity)); // Занимаем реальный причал
с помощью емкости корабля.
       *:que[ IP(Quenum)].DEPART; // Выйти из очереди.
       *:Future.Wait(Exponential( RP(M_Unload))); // Время разгрузки.
        *:Future.Wait(MeanDev(RP(M_Load),RP(Loadsp))); // Время погрузки.
       *:Berth[IP(Berth_num)].LEAVE(IP(Capacity)) ;// Реальный причал освобождается на
емкость корабля.
       *:TEST([Ip(size)=3,true],[!*,Skipit]) ; // Большой корабль?  Если переключатель
установлен, то прилив
       *:TEST([TIDE],[!*]) ; //Ожидается прилив.
::Skipit  *:Berth1[IP(Berth_num)].LEAVE(IP(Capacity)) ;// Причал, как заказ, освобождается
на емкость корабля.
      *: TAB[IP(Quenum)].TABULATE(TranAge);//Транзитное время сводится в таблицу по
типу корабля.
```

```
      *:topoint(Ip(size)+3,aclock,que[Ip(size)].t); // график среднего времени в очереди.
     *:TERMINATE ;// Корабль отплывает.
  {~mte} end;end;
  {~mb} procedure Simulation;  begin
     start(GetTerm);
 Line3d(false);
  show(tab[1],2);
  show(tab[2],1);
  show(tab[3],0);
ReportSave ('RPort.txt');
  {~me}end;
  {~rb} procedure Report; begin
{/Tide}  reportval(Tide,'Tide = ');
  {~re} end;
  {~cab} procedure CloseAllObj;begin
  {~cae} end;
```

Результирующая статистика по модели выглядит следующим образом.

```
Date Time Stamp= 18.12.2008 17:53:45   Ctrl Click |Name - HyperLink `R
E:\work\ObjectClient\Object No Sys\Future\GPSS-Future\book\tmp29.rtf
StartTime         0.00000 EndTime     120000.00000
 |Model |Blocks |Values |CurrentList |SynchronizeList |InterruptList
 |RPList |IPList |SPList |BPList
   Num      NameObject
     2        |Future
     3       |Berth 1
     4       |Berth 2
     5       |Berth 3
     6         |tab 1
     7         |tab 2
     8         |tab 3
     9        | que 1
    10        | que 2
    11        | que 3
    12      |Berth1 1
    13      |Berth1 2
    14      |Berth1 3
 |R `Model
 // * Таймер дня
{=gener0} ::gener0 1:Future.NewWait(24,0,gener0,gener0+1);
          2:TERMINATE( 1) ; //Часы работают один раз в день.
{=ge1} ::ge1 3:Future.NewWait(0,0,ge1,ge1+1,0,true,1);
::Again  4: LET(Tide,FALSE) ; // ;Циклическая заявка моделирует прилив.
             5:Future.Wait(3); // Отлив длится 3 часа.
        6: LET(Tide,TRUE) ; // Прилив начинается.
         7:Future.Wait(10); // Полная вода длится 10 часов.
        8:TRANSFER(Again) ; // Заявка возвращается к 'Again'.

{=ge2} ::ge2 9:Future.NewWait(Exponential(26),0,ge2,ge2+1);  //Каждые 26 часов появляется
корабль.
        10:IPLET( Size,[IBYBOOL([RANDOM<=0.5,RANDOM<=0.6,true],[1,2,3])]); // тип корабля в
соответствии с вероятностями.
       11:IPLET( Capacity,[ibynum(Ip(size),[1,2,4] )]);  // емкость корабля
       12:PRIORITY(Ip(size)); // приоритет
       13:IPLET( Quenum,[Ip(size)]) ; // Очередь  для  корабля.
       14:RPLET( M_Unload,[ibynum(Ip(size),[15,30,45])]) ; //Среднее время разгрузки.
       15:RPLET( M_Load,[ibynum(Ip(size),[24,36,56])]) ; //Среднее время погрузки.
       16:RPLET( Loadsp,[ibynum(Ip(size),[6,10,12])]); // Разброс времени погрузки.
       17:que[ IP(Quenum)].QUEUE ;
18:test([(berth1[1].r>=IP(Capacity))or(berth1[2].r>=IP(Capacity))or(berth1[3].r>=IP(Capacity))],
[19]); // Проверка, есть ли место для корабля на причале?
```

```
       19:IPLET( Berth_Num,[IBYBOOL([berth1[1].r>=IP(Capacity),
berth1[2].r>=IP(Capacity),berth1[3].r>=IP(Capacity)],[1,2,3])])  ; // Определяется свободный
причал.
       20:Berth1[IP(Berth_num)].ENTER (IP(Capacity)); // Занимаем причал с помощью емкости
корабля, как заказ.
         21:TEST([Ip(size)=3,true],[22,Skipit1]) ; // Большой корабль?  Если переключатель
установлен, то прилив
        22:TEST([TIDE],[23]) ; //Ожидается прилив.
::skipit1          23:Berth[IP(Berth_num)].ENTER (IP(Capacity)); // Занимаем реальный причал с
помощью емкости корабля.
        24:que[ IP(Quenum)].DEPART; // Выйти из очереди.
        25:Future.Wait(Exponential( RP(M_Unload))); // Время разгрузки.
         26:Future.Wait(MeanDev(RP(M_Load),RP(Loadsp))); // Время погрузки.
        27:Berth[IP(Berth_num)].LEAVE(IP(Capacity)) ;// Реальный причал освобождается на емкость
корабля.
        28:TEST([Ip(size)=3,true],[29,Skipit]) ; // Большой корабль?  Если переключатель
установлен, то прилив
        29:TEST([TIDE],[30]) ; //Ожидается прилив.
::Skipit  30:Berth1[IP(Berth_num)].LEAVE(IP(Capacity)) ;// Причал, как заказ, освобождается на
емкость корабля.
       31: TAB[IP(Quenum)].TABULATE(TranAge);//Транзитное время сводится в таблицу по типу
корабля.
        32:topoint(Ip(size)+3,aclock,que[Ip(size)].t); // график среднего времени в очереди.
       33:TERMINATE ;// Корабль отплывает.

 |R `Blocks    BLOCK Report
    Location     Total   Current
         1      5000         0
         2      5000         0
         3         1         0
         4      9231         0
         5      9231         0
         6      9231         0
         7      9231         1
         8      9230         0
         9      4572         0
        10      4572         0
        11      4572         0
        12      4572         0
        13      4572         0
        14      4572         0
        15      4572         0
        16      4572         0
        17      4572         0
        18      4572         0
        19      4572         0
        20      4572         0
        21      4572         0
        22       902         0
        23      4572         0
        24      4572         0
        25      4572         2
        26      4570         1
        27      4569         0
        28      4569         0
        29       901         0
        30      4569         0
        31      4569         0
        32      4569         0
        33      4569         0
 |R `Values
Tide =       TRUE
 |R `RPList  Report ( Double ) Count= 9
        Tran            Num/Name         Double
        9569                   4       45.00000
        9569                   5       56.00000
        9569                   6       12.00000
        9571                   4       15.00000
        9571                   5       24.00000
        9571                   6        6.00000
```

```
        9573                    4       15.00000
        9573                    5       24.00000
        9573                    6        6.00000
 |R `IPList Report ( Integer ) Count= 12
        Tran            Num/Name        Integer
        9569                    1              3
        9569                    2              4
        9569                    3              3
        9569                    7              3
        9571                    1              1
        9571                    2              1
        9571                    3              1
        9571                    7              1
        9573                    1              1
        9573                    2              1
        9573                    3              1
        9573                    7              1
 |R Storage = `Berth 1 Entries = 2655 Current = 2 Average =  1.10904 AverTime = 50.12625 Utility
=  0.55452
    Max = 2 Min = 0 Tran = 9573 ACapac =  2.00000 Capacity = 2 Active Time = 120000.00000 NextGo
= 0
 |R Storage = `Berth 2 Entries = 3437 Current = 0 Average =  2.31240 AverTime = 80.73537 Utility
=  0.57810
    Max = 4 Min = 0 Tran = 9567 ACapac =  4.00000 Capacity = 4 Active Time = 120000.00000 NextGo
= 0
 |R Storage = `Berth 3 Entries = 2544 Current = 4 Average =  1.89707 AverTime = 89.48463 Utility
=  0.47427
    Max = 4 Min = 0 Tran = 9569 ACapac =  4.00000 Capacity = 4 Active Time = 120000.00000 NextGo
= 0
 |R Storage = `Berth1 1 Entries = 2655 Current = 2 Average =  1.10904 AverTime = 50.12625
Utility =  0.55452
    Max = 2 Min = 0 Tran = 9573 ACapac =  2.00000 Capacity = 2 Active Time = 120000.00000 NextGo
= 0
 |R Storage = `Berth1 2 Entries = 3437 Current = 0 Average =  2.32117 AverTime = 81.04167
Utility =  0.58029
    Max = 4 Min = 0 Tran = 9567 ACapac =  4.00000 Capacity = 4 Active Time = 120000.00000 NextGo
= 0
 |R Storage = `Berth1 3 Entries = 2544 Current = 4 Average =  1.90539 AverTime = 89.87675
Utility =  0.47635
    Max = 4 Min = 0 Tran = 9569 ACapac =  4.00000 Capacity = 4 Active Time = 120000.00000 NextGo
= 0
 |R Queue = ` que 1 Entries = 2312 Current = 0 AverQueue =  0.12035 AverTime =  6.24671
    AverTime(-Ze) = 80.23546 Max = 8 Min = 0 Zero = 2132 Active Time = 120000.00000 NextGo = 0
 |R Queue = ` que 2 Entries = 1358 Current = 0 AverQueue =  0.08582 AverTime =  7.58313
    AverTime(-Ze) = 71.51314 Max = 4 Min = 0 Zero = 1214 Active Time = 120000.00000 NextGo = 0
 |R Queue = ` que 3 Entries = 902 Current = 0 AverQueue =  0.21056 AverTime = 28.01291
    AverTime(-Ze) = 126.33821 Max = 7 Min = 0 Zero = 702 Active Time = 120000.00000 NextGo = 0
 |R User/Future = `Future Entries = 37179 Current = 6 AverUser =  4.45250 AverTime = 14.37102
    Max = 11 Min = 0 Active Time = 120000.00000 NextGo = 0
 |R Report  User/Future List Count= 6
      Tran     Assem  Priority   Current      Next       TimeStamp       SortValue     Group
Inter
      9571      9571         1        25        26    119980.13879    120001.78986         0
FALSE
         2         2         0         7         8         0.00000    120003.00000         0
FALSE
      9573      9573         1        25        26    119991.22007    120005.56350         0
FALSE
      9575      9575         0         0         1    120024.00000    120024.00000         0
FALSE
      9569      9569         3        26        27    119957.41999    120026.84824         0
FALSE
      9574      9574         0         0         9    120078.55143    120078.55143         0
FALSE
            Num/Name        Integer  |R `Future      Report IP LIST  Count= 6
I                  1           5001
I                  2           9575
I                  3              1
I                  4              2
I                  9           4573
```

```
I                 10           9574
 |R  Table = `tab 1  Entries = 2310.00000 Mean = 45.30412 StdDev = 32.52791
           Range       Frequency         Integral
        10.00000         0.00000         0.00000
        20.00000        24.00000         0.01039
        30.00000       595.00000         0.26797
        40.00000       742.00000         0.58918
        50.00000       361.00000         0.74545
        60.00000       226.00000         0.84329
        70.00000       135.00000         0.90173
        80.00000        69.00000         0.93160
        90.00000        45.00000         0.95108
       100.00000        21.00000         0.96017
       110.00000        18.00000         0.96797
       120.00000        16.00000         0.97489
       130.00000        13.00000         0.98052
       140.00000         4.00000         0.98225
       150.00000         4.00000         0.98398
       160.00000         4.00000         0.98571
       170.00000         5.00000         0.98788
       180.00000         5.00000         0.99004
       190.00000         4.00000         0.99177
       200.00000         3.00000         0.99307
       210.00000         3.00000         0.99437
       220.00000         0.00000         0.99437
       230.00000         1.00000         0.99481
       240.00000         1.00000         0.99524
       250.00000         0.00000         0.99524
       260.00000         2.00000         0.99610
       270.00000         0.00000         0.99610
       280.00000         1.00000         0.99654
       290.00000         0.00000         0.99654
       300.00000         1.00000         0.99697
       310.00000         1.00000         0.99740
       320.00000         0.00000         0.99740
       330.00000         0.00000         0.99740
       340.00000         0.00000         0.99740
       350.00000         0.00000         0.99740
       360.00000         1.00000         0.99784
       370.00000         0.00000         0.99784
       380.00000         0.00000         0.99784
       390.00000         1.00000         0.99827
       400.00000         0.00000         0.99827
       410.00000         2.00000         0.99913
       420.00000         0.00000         0.99913
       430.00000         0.00000         0.99913
       440.00000         1.00000         0.99957
       450.00000         0.00000         0.99957
       460.00000         1.00000         1.00000
       470.00000         0.00000         1.00000
 |R  Table = `tab 2  Entries = 1358.00000 Mean = 73.13224 StdDev = 37.51444
           Range       Frequency         Integral
        20.00000         0.00000         0.00000
        30.00000        19.00000         0.01399
        40.00000       141.00000         0.11782
        50.00000       254.00000         0.30486
        60.00000       237.00000         0.47938
        70.00000       149.00000         0.58910
        80.00000       124.00000         0.68041
        90.00000       114.00000         0.76436
       100.00000        79.00000         0.82253
       110.00000        50.00000         0.85935
       120.00000        48.00000         0.89470
       130.00000        30.00000         0.91679
       140.00000        29.00000         0.93814
       150.00000        20.00000         0.95287
       160.00000        17.00000         0.96539
       170.00000        13.00000         0.97496
       180.00000         9.00000         0.98159
       190.00000         5.00000         0.98527
```

```
      200.00000          2.00000          0.98675
      210.00000          4.00000          0.98969
      220.00000          3.00000          0.99190
      230.00000          2.00000          0.99337
      240.00000          3.00000          0.99558
      250.00000          1.00000          0.99632
      260.00000          2.00000          0.99779
      270.00000          0.00000          0.99779
      280.00000          1.00000          0.99853
      290.00000          0.00000          0.99853
      300.00000          1.00000          0.99926
      310.00000          1.00000          1.00000
      320.00000          0.00000          1.00000
|R  Table = `tab 3  Entries = 901.00000 Mean = 130.97555 StdDev = 69.83947
          Range        Frequency         Integral
       40.00000          0.00000          0.00000
       50.00000          8.00000          0.00888
       60.00000         32.00000          0.04440
       70.00000         76.00000          0.12875
       80.00000        107.00000          0.24750
       90.00000         90.00000          0.34739
      100.00000         68.00000          0.42286
      110.00000         57.00000          0.48613
      120.00000         64.00000          0.55716
      130.00000         52.00000          0.61487
      140.00000         54.00000          0.67481
      150.00000         34.00000          0.71254
      160.00000         31.00000          0.74695
      170.00000         27.00000          0.77691
      180.00000         18.00000          0.79689
      190.00000         20.00000          0.81909
      200.00000         28.00000          0.85017
      210.00000         26.00000          0.87902
      220.00000         15.00000          0.89567
      230.00000         10.00000          0.90677
      240.00000         14.00000          0.92231
      250.00000         12.00000          0.93563
      260.00000         11.00000          0.94784
      270.00000          6.00000          0.95450
      280.00000          8.00000          0.96337
      290.00000          5.00000          0.96892
      300.00000          7.00000          0.97669
      310.00000          2.00000          0.97891
      320.00000          1.00000          0.98002
      330.00000          2.00000          0.98224
      340.00000          2.00000          0.98446
      350.00000          1.00000          0.98557
      360.00000          3.00000          0.98890
      370.00000          0.00000          0.98890
      380.00000          0.00000          0.98890
      390.00000          1.00000          0.99001
      400.00000          1.00000          0.99112
      410.00000          0.00000          0.99112
      420.00000          1.00000          0.99223
      430.00000          3.00000          0.99556
      440.00000          0.00000          0.99556
      450.00000          0.00000          0.99556
      460.00000          1.00000          0.99667
      470.00000          0.00000          0.99667
      480.00000          1.00000          0.99778
      490.00000          0.00000          0.99778
      500.00000          0.00000          0.99778
      510.00000          1.00000          0.99889
      520.00000          0.00000          0.99889
      530.00000          0.00000          0.99889
      540.00000          0.00000          0.99889
      550.00000          0.00000          0.99889
      560.00000          1.00000          1.00000
      570.00000          0.00000          1.00000       4        17
```

Если количество элементов в массивах объектов невелико, известно заранее и не может меняться, то можно «погрузить» одиночные объекты в массивы соответствующих типов. Как это делается на практике – демонстрирует пример модели порта, в которой эта техника используется. Ваша выгода при таком построении модели в том, что не нужно использовать циклы в процедуре Initial.

Пример модели с погружением объектов в массивы.

```
Const
 Size=1;
 Capacity=2;
 Quenum=3;
 M_Unload=4;
 M_Load=5;
 Loadsp=6;
 Berth_Num=7;
  {~vb} Var
 {~f}Future:TFuture;
 Tide:Boolean;
{/BerthRe} BerthRe:Tstorage;
{/BerthRe0} BerthRe0:Tstorage;
{/BerthRe1} BerthRe1:Tstorage;
  BerthR:ARRAY[1..3] OF Tstorage;
{/BerthVir} BerthVir:Tstorage;
{/BerthVir0} BerthVir0:Tstorage;
{/BerthVir1} BerthVir1:Tstorage;
  BerthV:ARRAY[1..3] of Tstorage;
{/QBertch} QBertch:Tqueue;
{/QBertch0} QBertch0:Tqueue;
{/QBertch1} QBertch1:Tqueue;
 qBer:ARRAY[1..3]  OF  Tqueue;
{/Tabnum} Tabnum:Ttable;
{/Tabnum0} Tabnum0:Ttable;
{/Tabnum1} Tabnum1:Ttable;
  tab:ARRAY[1..3]  OF  Ttable;
  {~ve}
  {~pfe}
  {~ib} procedure Initial;begin
 {~f}init(Future,'Future');
   SETSTART(5000);
   Tide:=false;
{/BerthRe} init(BerthRe,' BerthRe',2);
{/BerthRe0} init(BerthRe0,' BerthRe0',4);
{/BerthRe1} init(BerthRe1,' BerthRe1',4);
{/BerthVir} init(BerthVir,' BerthVir',2);
{/BerthVir0} init(BerthVir0,' BerthVir0',4);
{/BerthVir1} init(BerthVir1,' BerthVir1',4);
{/QBertch} init(QBertch,' Berth');
```

```
{/QBertch0} init(QBertch0,' Berth0');
{/QBertch1} init(QBertch1,' Berth1');
{/Tabnum}  init(Tabnum,' Tabnum',0,10,100);
{/Tabnum0}  init(Tabnum0,' Tabnum0',0,10,100);
{/Tabnum1}  init(Tabnum1,' Tabnum1',0,10,100);
 BerthR[1]:=BerthRe;
 BerthR[2]:=BerthRe0;
 BerthR[3]:=BerthRe1;
 Berthv[1]:=Berthvir;
 Berthv[2]:=Berthvir0;
 Berthv[3]:=Berthvir1;
 qBer[1]:=QBertch;
 qBer[2]:=QBertch0;
 qBer[3]:=QBertch1;
 tab[1]:=Tabnum;
 tab[2]:=Tabnum0;
 tab[3]:=Tabnum1;
{=gener0} Future.NewWaitProc(gener0,24,0,gener0,gener0+1);
{=ge1} Future.NewWaitProc(ge1,0,0,ge1,ge1+1);
{=ge2} Future.NewWaitProc(ge2,Exponential(26),0,ge2,ge2+1);
  {~ie}end;
  {~mtb}procedure ModelTxt;begin  case ActiveBlock of
// * Таймер дня
{=gener0} ::gener0 *:Future.NewWait(24,0,gener0,gener0+1);
        *:TERMINATE( 1) ; //Часы работают один раз в день.
{=ge1} ::ge1 *:Future.NewWait(0,0,ge1,ge1+1,0,true,1);
::Again *: LET(Tide,FALSE) ; // ;Циклический транзакт моделирует прилив.
       *:Future.Wait(3); // Отлив длится 3 часа.
       *: LET(Tide,TRUE) ; // Прилив начинается.
       *:Future.Wait(10); // Полная вода длится 10 часов.
       *:TRANSFER(Again) ; // Транзакт возвращается к 'Again'.
{=ge2} ::ge2 *:Future.NewWait(Exponential(26),0,ge2,ge2+1);
       *:IPLET( Size,[IBYBOOL([RANDOM<=0.5,RANDOM<=0.6,true],[1,2,3])]); // тип
корабля
       *:IPLET( Capacity,[ibynum(Ip(size),[1,2,4] )]);  // емкость корабля
       *:PRIORITY(Ip(size)); // приоритет
       *:IPLET( Quenum,[Ip(size)]) ; // Очередь  для  кораблей.
       *:RPLET( M_Unload,[ibynum(Ip(size),[15,30,45])]) ; //Среднее время разгрузки.
       *:RPLET( M_Load,[ibynum(Ip(size),[24,36,56])]) ; //Среднее время погрузки.
       *:RPLET( Loadsp,[ibynum(Ip(size),[6,10,12])]); // Разброс времени погрузки.
       *:qber[ IP(Quenum)].QUEUE ;
   *:test([(berthv[1].r>=IP(Capacity)) or(berthv[2].r>=IP(Capacity))
or(berthv[3].r>=IP(Capacity))],[!*]);
       *:IPLET( Berth_Num,[IBYBOOL([berthv[1].r>=IP(Capacity),berthv[2].r>=
IP(Capacity),berthv[3].r>=IP(Capacity)],[1,2,3])])  ;
```

```
*:Berthv[IP(Berth_num)].ENTER (IP(Capacity)); // Занимаем причал с помощью
емкости корабля.
*:TEST([Ip(size)=3,true],[!*,Skipit1]) ; // Большой корабль?  Если переключатель
установлен, то прилив
*:TEST([TIDE],[!*]) ; //Ожидается прилив.
::skipit1 *:Berthr[IP(Berth_num)].ENTER (IP(Capacity)); // Занимаем причал с помощью
емкости корабля.
*:qber[ IP(Quenum)].DEPART; // Выйти из очереди.
*:Future.Wait(Exponential( RP(M_Unload))); // Время разгрузки.
*:Future.Wait(MeanDev(RP(M_Load),RP(Loadsp))); // Время погрузки.
*:Berthr[IP(Berth_num)].LEAVE(IP(Capacity)) ;// Причал освобождается на емкость
корабля.
*:TEST([Ip(size)=3,true],[!*,Skipit]) ; // Большой корабль?  Если переключатель
установлен, то прилив
*:TEST([TIDE],[!*]) ; //Ожидается прилив.
::Skipit  *:Berthv[IP(Berth_num)].LEAVE(IP(Capacity)) ;// Причал освобождается на
емкость корабля.
*: TAB[IP(Quenum)].TABULATE(tranage);//Транзитное время сводится в таблицу по
типу корабля.
*:topoint(Ip(size),aclock,qber[Ip(size)].a*100);
//    *:topoint(Ip(size)+3,aclock,qber[Ip(size)].t);
*:TERMINATE ;// Корабль отплывает.
 {~mte} end;end;
 {~mb} procedure Simulation;  begin
//  Line3d(false);
 start(GetTerm);
 show(tab[1],2);
 show(tab[2],1);
 show(tab[3],0);
 ReportSave ('RPort.txt');
 {~me}end;
 {~rb} procedure Report;begin
{/Tide}  reportval(Tide,'Tide = ');
 {~re} end;
 {~cab} procedure CloseAllObj;begin
 {~cae} end;
```

Обратите только внимание, что присваивания объектов элементам массивов – погружение должно вестись в конце процедуры Initial, например, как это сделано в данной модели, а описание массивов должно быть выполнено вручную и в дополнение к описаниям объектов.

```
BerthR[1]:=BerthRe;
BerthR[2]:=BerthRe0;
BerthR[3]:=BerthRe1;
Berthv[1]:=Berthvir;
Berthv[2]:=Berthvir0;
Berthv[3]:=Berthvir1;
```

qBer[1]:=QBertch;
qBer[2]:=QBertch0;
qBer[3]:=QBertch1;
tab[1]:=Tabnum;
tab[2]:=Tabnum0;
tab[3]:=Tabnum1;

25 Работа с прерываниями

Прерывания возможны только для одноканальных устройств. Если устройство в данный момент свободно, то занятие его по прерыванию ничем 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. Для моделирования прерываний используются следующие блоки.

*:Ustr.Extract(NumBl,NumRP);

Этот блок удаляет обслуживаемую заявку из устройства Ustr.
Здесь NumBl - номер или метка блока, куда удаляется заявка.
NumRP- номер RP - параметра для запоминания оставшегося времени задержки в блоке *:Future.Wait для удаляемой заявки. По умолчанию, время не запоминается. Устройство остается незанятым, и, даже если были прерванные заявки, то они его не получают.

Поэтому после выполнения этого блока возможно занятие устройства по Seize или Preempt. Возможно также выполнение Return0, для передачи устройства прерванной ранее на устройстве заявке.

*:Ustr.Preempt;

Этот блок прерывает обслуживание заявки на устройстве.
Активная заявка в любом случае занимает устройство по прерыванию.

*:Ustr.Return;

Этот блок освобождает устройство, занятое по прерыванию. Ранее прерванная заявка, если она есть, получает устройство для продолжения обслуживания.

*:Ustr.Preempt0;

Этот блок прерывает обслуживание заявки на устройстве. Устройство остается незанятым. Чаще всего используется для моделирования недоступности.

*:Ustr.Return0;

Этот блок возвращает для обслуживания в устройстве, прерванную заявку. Это происходит после того, как устройство было оставлено незанятым. Чаще всего он используется для моделирования возврата устройства из недоступности.

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:

```
{~vb} Var
{~f}Future:TFuture;
{/ust1} ust1:Tqueue;
{/ust} ust:Tfacility;
{/tt} tt:Ttable;
{~ve}
{~pfe}
{~ib} procedure Initial;begin
{~f}init(Future,'Future');
{/ust1}  init(ust1,' ust1');
{/ust}  init(ust,' ust');
setstart(1000);
{/tt}  init(tt,'tt',0,100,100);
{=Gen} Future.NewWaitProc(Gen,exponential(100),0,Gen,Gen+1);
{~ie}end;
{~mtb}procedure ModelTxt;begin case ActiveBlock of
{=Gen} ::Gen *:Future.NewWait(exponential(100),0,Gen,Gen+1);
*:priority (duniform(1,10));
*:ust1.queue;
*:test([pri(ust.Tran)<pri],[!*]);
*:ust.preempt;
*: Mark ;
*:ust1.depart;
*:Future.Wait(MeanDev(95,15));
*:tt.tabulate(TranAge);
*:ust.return;
*:terminate(1);
{~mte} end;end;
{~mb} procedure Simulation;  begin
   start(GetTerm);
   show(tt,2);
{~me}end;
{~rb} procedure Report;begin
{~re} end;
{~cab} procedure CloseAllObj;begin
{~cae} end;
```

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

```
Date Time Stamp= 19.12.2008 20:25:58   Ctrl Click |Name - HyperLink `R
E:\work\ObjectClient\Object No Sys\Future\GPSS-Future\book\tmp31.rtf
StartTime         0.00000 EndTime     103654.22528
 |Model |Blocks |Values |CurrentList |SynchronizeList |InterruptList
 |RPList |IPList |SPList |BPList
   Num      NameObject
     2        |Future
     3         | ust1
     4          | ust
     5            |tt
 |R `Model
 {=Gen} ::Gen 1:Future.NewWait(exponential(100),0,Gen,Gen+1);
  2:priority (duniform(1,10));
```

```
  3:ust1.queue;
  4:test([pri(ust.Tran)<pri],[5]);
  5:ust.preempt;
  6: Mark ;
  7:ust1.depart;
 8:Future.Wait(MeanDev(95,15));
  9:tt.tabulate(TranAge);
  10:ust.return;
  11:terminate(1);

 |R `Blocks    BLOCK Report
    Location     Total   Current
         1      1004         0
         2      1004         0
         3      1004         4
         4      1000         0
         5      1000         0
         6      1000         0
         7      1000         0
         8      1000         0
         9      1000         0
        10      1000         0
        11      1000         0
 |R `Values
 |R `CurrentList Report Current Transactions List  Count= 4
      Tran     Assem  Priority   Current      Next       TimeStamp       TimeEnter Inter
       951       951         1         3         4     98176.73340     98176.73340 FALSE
       953       953         1         3         4     98255.20262     98255.20262 FALSE
       955       955         1         3         4     98396.94562     98396.94562 FALSE
       962       962         1         3         4     99050.74058     99050.74058 FALSE
 |R Facility = ` ust Entries = 1000 Current = 0 Utility =  0.92198 AverTime = 95.56760
    Tran = 0 IntTran = 0 Active Time = 103654.22528 NextGo = 0
 |R Queue = ` ust1 Entries = 1004 Current = 4 AverQueue =  3.93794 AverTime = 406.55774
    AverTime(-Ze) = 1214.83325 Max = 16 Min = 0 Zero = 668 Active Time = 103654.22528 NextGo = 0
 |R User/Future = `Future Entries = 2005 Current = 1 AverUser =  2.56128 AverTime = 132.41281
    Max = 8 Min = 0 Active Time = 103654.22528 NextGo = 0
 |R Report  User/Future List Count= 1
      Tran     Assem  Priority   Current      Next       TimeStamp       SortValue     Group
Inter
      1005      1005         0         0         1    103796.81015    103796.81015         0
FALSE
            Num/Name        Integer  |R `Future      Report IP LIST  Count= 2
I                 1           1005
I                 2           1005
 |R  Table = `tt  Entries = 1000.00000 Mean = 223.12374 StdDev = 459.37994
          Range       Frequency        Integral
        0.00000         0.00000         0.00000
      100.00000       438.00000         0.43800
      200.00000       352.00000         0.79000
      300.00000        79.00000         0.86900
      400.00000        30.00000         0.89900
      500.00000        28.00000         0.92700
      600.00000        18.00000         0.94500
      700.00000        13.00000         0.95800
      800.00000         5.00000         0.96300
      900.00000         7.00000         0.97000
     1000.00000         2.00000         0.97200
     1100.00000         5.00000         0.97700
     1200.00000         4.00000         0.98100
     1300.00000         1.00000         0.98200
     1400.00000         1.00000         0.98300
     1500.00000         1.00000         0.98400
     1600.00000         1.00000         0.98500
     1700.00000         1.00000         0.98600
     1800.00000         1.00000         0.98700
     1900.00000         1.00000         0.98800
     2000.00000         1.00000         0.98900
```

Информация о таблице приводится не полностью.

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. Обслуживаемые заявки, как основные, так и пришедшие по прерыванию в 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:

```
{~vb} Var
 {~f}Future:TFuture;
{/ust} ust:Tfacility;
{/tt} tt:Ttable;
{/Fac} Fac:Tfacility;
{/Que} Que:Tqueue;
{/Que0} Que0:Tqueue;
{/Qust} Qust:Tqueue;
{/Tt1} Tt1:Ttable;
  {~ve}
  {~pfe}
  {~ib} procedure Initial;begin
 {~f}init(Future,'Future');
{/ust}  init(ust,' ust');
setstart(1000);
{/tt}  init(tt,'tt',0,80,100);
{/Fac}  init(Fac,' Fac');
{/Que}  init(Que,' Que');
{/Que0}  init(Que0,' Que0');
{/Qust}  init(Qust,' Qust');
{/Tt1}  init(Tt1,' Tt1',0,10,100);
{=Gen0} Future.NewWaitProc(Gen0,exponential(200),0,Gen0,Gen0+1);
{=Gen1} Future.NewWaitProc(Gen1,MeanDev(300,200),0,Gen1,Gen1+1);
  {~ie}end;
  {~mtb}procedure ModelTxt;begin case ActiveBlock of
{=Gen0} ::Gen0 *:Future.NewWait(exponential(200),0,Gen0,Gen0+1);
  *:QUst.queue;
  *:ust.seize;
  *:QUst.depart;
  *: Mark ;
 *:Future.Wait(MeanDev(120,20));
  *:ust.release;
  *:tt.tabulate(TranAge);
  *:terminate;
{=Gen1} ::Gen1 *:Future.NewWait(MeanDev(300,200),0,Gen1,Gen1+1);
  *:QUe.queue;
```

```
 *:ust.extract(next,1);
 *:ust.preempt;
 *: Mark ;
 *:QUe.depart;
*:Future.Wait(MeanDev( 150,100));
 *:tt.tabulate(TranAge);
 *:ust.return;
*:tt1.tabulate(TranAge );
 *:terminate(1);
::next *: Que0.queue ;
  *: fac.seize ;
  *: Que0.depart ;
  *:Future.Wait(3*RP(1));
  *: Fac.Release ;
 *:tt.tabulate(TranAge);
 *:terminate;
 {~mte} end;end;
 {~mb} procedure Simulation;  begin
{/sh}     start(GetTerm);
 show(tt,2);
 {~me}end;
 {~rb} procedure Report;begin
 {~re} end;
 {~cab} procedure CloseAllObj;begin
 {~cae} end;
```

Итоговая статистика этой модели имеет вид:

```
Date Time Stamp= 19.12.2008 20:35:29   Ctrl Click |Name - HyperLink `R
E:\work\ObjectClient\Object No Sys\Future\GPSS-Future\book\tmp32.rtf
StartTime         0.00000 EndTime     348728.74578
 |Model |Blocks |Values |CurrentList |SynchronizeList |InterruptList
 |RPList |IPList |SPList |BPList
   Num      NameObject
     2        |Future
     3          | ust
     4            |tt
     5          | Fac
     6          | Que
     7         | Que0
     8         | Qust
     9          | Tt1
 |R `Model
 {=Gen0} ::Gen0 1:Future.NewWait(exponential(200),0,Gen0,Gen0+1);
  2:QUst.queue;
  3:ust.seize;
  4:QUst.depart;
  5: Mark ;
 6:Future.Wait(MeanDev(120,20));
  7:ust.release;
  8:tt.tabulate(TranAge);
  9:terminate;
{=Gen1} ::Gen1 10:Future.NewWait(MeanDev(300,200),0,Gen1,Gen1+1);
  11:QUe.queue;
  12:ust.extract(next,1);
  13:ust.preempt;
  14: Mark ;
  15:QUe.depart;
 16:Future.Wait(MeanDev( 150,100));
```

```
 17:tt.tabulate(TranAge);
 18:ust.return;
19:tt1.tabulate(TranAge );
 20:terminate(1);
::next 21: Que0.queue ;
  22: fac.seize ;
  23: Que0.depart ;
  24:Future.Wait(3*RP(1));
  25: Fac.Release ;
 26:tt.tabulate(TranAge);
 27:terminate;

|R `Blocks    BLOCK Report
   Location     Total   Current
         1      1730         0
         2      1730         5
         3      1725         0
         4      1725         0
         5      1725         0
         6      1725         0
         7       881         0
         8       881         0
         9       881         0
        10      1153         0
        11      1153         0
        12      1153         0
        13      1153         0
        14      1153         0
        15      1153         0
        16      1153         0
        17      1000         0
        18      1000         0
        19      1000         0
        20      1000         0
        21       997         0
        22       997         0
        23       997         0
        24       997         1
        25       996         0
        26       996         0
        27       996         0
|R `Values
|R `CurrentList Report Current Transactions List  Count= 5
     Tran     Assem  Priority   Current      Next       TimeStamp       TimeEnter
Inter
     2878      2878         0         2         3    348234.11478    348234.11478
FALSE
     2879      2879         0         2         3    348506.42743    348506.42743
FALSE
     2882      2882         0         2         3    348546.83176    348546.83176
FALSE
     2883      2883         0         2         3    348594.75790    348594.75790
FALSE
     2884      2884         0         2         3    348679.29809    348679.29809
FALSE
|R `RPList  Report ( Double ) Count= 1
       Tran            Num/Name         Double
       2876                   1      112.69241
|R Facility = ` ust Entries = 2878 Current = 0 Utility =  0.91358 AverTime = 110.69880
   Tran = 0 IntTran = 0 Active Time = 348728.74578 NextGo = 0
|R Facility = ` Fac Entries = 997 Current = 1 Utility =  0.52871 AverTime = 184.93082
   Tran = 2876 IntTran = 0 Active Time = 348728.74578 NextGo = 0
|R Queue = ` Que Entries = 1153 Current = 0 AverQueue =  0.00000 AverTime =  0.00000
   AverTime(-Ze) =  0.00000 Max = 1 Min = 0 Zero = 1153 Active Time = 348728.74578
NextGo = 0
|R Queue = ` Que0 Entries = 997 Current = 0 AverQueue =  0.11172 AverTime = 39.07730
   AverTime(-Ze) = 779.20142 Max = 3 Min = 0 Zero = 947 Active Time = 348728.74578
NextGo = 0
|R Queue = ` Qust Entries = 1730 Current = 5 AverQueue =  2.98391 AverTime = 601.48934
```

```
            AverTime(-Ze) = 757.33374 Max = 20 Min = 0 Zero = 356 Active Time = 348728.74578
NextGo = 0
          |R User/Future = `Future Entries = 6760 Current = 3 AverUser =  2.86891 AverTime =
147.99872
            Max = 4 Min = 0 Active Time = 348728.74578 NextGo = 0
          |R Report  User/Future List Count= 3
              Tran     Assem  Priority   Current      Next       TimeStamp       SortValue
Group Inter
              2876      2876         0        24        25    348480.94259    348885.96050
0 FALSE
              2885      2885         0         0         1    348897.88168    348897.88168
0 FALSE
              2881      2881         0         0        10    348947.83251    348947.83251
0 FALSE
                    Num/Name        Integer  |R `Future      Report IP LIST  Count= 4
         I                 1           1731
         I                 2           2885
         I                10           1154
         I                11           2881
          |R  Table = `tt  Entries = 2877.00000 Mean = 188.27927 StdDev = 109.37589
                  Range       Frequency        Integral
             0.00000         0.00000         0.00000
            80.00000       173.00000         0.06013
           160.00000      1404.00000         0.54814
           240.00000       638.00000         0.76990
           320.00000       304.00000         0.87556
           400.00000       192.00000         0.94230
           480.00000        98.00000         0.97636
           560.00000        36.00000         0.98888
           640.00000        19.00000         0.99548
           720.00000         9.00000         0.99861
           800.00000         2.00000         0.99930
           880.00000         2.00000         1.00000
           960.00000         0.00000         1.00000
          |R  Table = ` Tt1  Entries = 1000.00000 Mean = 142.83907 StdDev = 56.04077
                  Range       Frequency        Integral
            50.00000         0.00000         0.00000
            60.00000        67.00000         0.06700
            70.00000        46.00000         0.11300
            80.00000        60.00000         0.17300
            90.00000        54.00000         0.22700
           100.00000        55.00000         0.28200
           110.00000        62.00000         0.34400
           120.00000        47.00000         0.39100
           130.00000        45.00000         0.43600
           140.00000        50.00000         0.48600
           150.00000        57.00000         0.54300
           160.00000        56.00000         0.59900
           170.00000        48.00000         0.64700
           180.00000        45.00000         0.69200
           190.00000        57.00000         0.74900
           200.00000        53.00000         0.80200
           210.00000        48.00000         0.85000
           220.00000        49.00000         0.89900
           230.00000        41.00000         0.94000
           240.00000        30.00000         0.97000
           250.00000        30.00000         1.00000
           260.00000         0.00000         1.00000
```

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

26 Ввод и вывод данных при моделировании

По ходу моделирования возможен вывод данных. Это позволяет наблюдать ход моделирования. Наиболее наглядный вывод данных о ходе моделирования дают всевозможные графики, которых мы вскользь касались и раньше.

Для этих целей используются, в основном, блоки и процедуры из модуля AddModelUnit.

Прямой вывод очередной точки графика выполняет блок *:ToPoint, который имеет вид:

*:ToPoint(Num,X,Y);

Здесь Num- номер графика, X,Y – координаты очередной точки. Всего имеется 16 графиков, от 0 до 15.

Процедуры, описанные ниже, следует вызывать или в процедуре Simulation, или в процедуре Initial.

Перед выводом графиков полезно очистить содержимое вкладки Line, или хотя бы конкретного графика. Это выполняет процедура

LineClearProc (Num). Здесь Num- номер графика. Ее следует помещать, до выполнения процедуры Start. Если процедура вызвана без параметра, то она удаляет все графики.

Для задания цвета графика и его имени можно вызвать процедуры.

LineColor(Num,Col) Задает цвет графика с номером Num .

LineTitle(Num,Line); Задает имя графика с номером Num .

Для задания цвета можно воспользоваться функцией RGB(R,G,B) . В этой функции R,G,B – это интенсивности красного, зеленого и голубого цветов соответственно. Интенсивность любого цвета должна лежать в диапазоне от 0 до 255. Так RGB(0,0,0) – определяет черный цвет, а RGB(255,255,255) – белый цвет.

Цвета также можно задавать с помощью предопределенных констант из следующего списка.

ClAqua – бирюзовый
clBlack - черный
clBlue - синий
clCream - кремовый
clDkGray Темно-серый
clFuchsia фуксиновый
clGray – темно серый
clGreen - зеленый
clLime – лимонно зеленый
clLtGray - светло-серый
clMaroon марон
clMedGray средне серый
clMoneyGreen цвет доллара
clNavy морской синий
clOlive оливковый
clPurple фиолетовый
clRed красный
clSilver серебряный
clSkyBlue голубой
clTeal темно бирюзовый
clWhite белый
clYellow желтый

Если вы хотите управлять видом графиков (плоские- объемные), то можно вызвать процедуру. Line3D(Yes3D);

Ее логический параметр Yes3D , переключает, плоские - объемные графики.

Если вы хотите сохранить графики в файле на диске, то вызовите процедуру. LineSave (FileName) .

PageColor(Page,Col); Задает цвет страницы. Список констант для SetPage, PageColor и ShowPage. spSimulation, ReportSetting, spReport, spLine, spValues, spBar, spAbout, spReserve ,spLicence

Процедура ToPointProc(Num,X,Y), подобна *:ToPoint(Num,X,Y), однако она не является блоком.

Аналогичным образом, блок *:LineClear (Num), очищающий график с данным номером, подобен процедуре LineClearProc (Num), но выполняется в ходе моделирования.

Для работы с гистограммами, в основном, используются процедуры, а не блоки.

Процедура BarClearProc (Num) выполняет сброс заданной гистограммы, по умолчанию, сбрасываются все гистограммы.

Блок *:BarClear (Num); выполняет ту же работу, что и процедура BarClearProc (Num), но выполняется в ходе моделирования.

Процедура Bar3D(Yes3D) переключает, плоские - объемные гистограммы.

Процедура BarColor(Num,Col) задает цвет гистограммы с номером Num.

Процедура BarTitle(Num,Line) задает имя гистограммы с номером Num.

Процедура BarSave (FileName) сохраняет гистограммы в файле на диске.

Процедура Bar(Num,X,Y) создает гистограмму, если X и Y – динамические массивы координат а, Num – номер гистограммы.

Для работы с выводом значений величин на вкладке Value, используются следующие процедуры и блоки.

Блок *:ToValue(Num,Value) выполняет вывод значения Value для строки Num в виде величины соответствующего столбика.

Процедура ValueColor(Num,Col) задает цвет столбика с номером Num.

Процедура ValueTitle(Num,Line) задает имя столбика с номером Num.

Процедура ToValueProc(Num,Value) , подобна *:ToValue(Num,Value), однако она не является блоком.

Для ввода и вывода текстов и манипуляций с файлами, используются следующие процедуры и блоки.

Блок *:ToString(Num,Line) выполняет вывод на AddMemo строки. Num – номер строки, Line - значение вещественного, целого, строкового или логического типа.

Блок *:CurrentBlocks выполняет вывод на AddMemo статистики по блокам.

Процедура AddMemo(Line) добавляет к AddMemo значение вещественного, целого, строкового или логического типа.

Процедура AddMemoClearProc очищает AddMemo.

Процедура AddLoad (FileName) загружает файл в AddMemo.

Процедура AddSave (FileName) сохраняет в файле содержимое AddMemo.

Процедура ToStringProc(Num,Line) , подобна *:ToString(Num,Line) , однако она не является блоком.

Процедура RepMemoAdd (Line) добавляет строку к RepMEMO.

Процедура ReportSave (FileName) сохраняет отчет по модели в файле.

Для ввода параметров из окна ввода, следует использовать функции.

RInput(Line) она вызывает диалог для ввода вещественного значения. Line - строка для заголовка.

IInput(Line) она вызывает диалог для ввода целого значения. Line - строка для заголовка.

SInput(Line) она вызывает диалог для ввода строкового значения. Line - строка для заголовка.

BInput(Line) она вызывает диалог для ввода логического значения. Line - строка для заголовка.

InputItem(Line) она вызывает диалог для выбора строки из выпадающего списка, Line - заголовок.

Для заполнения выпадающего списка используется процедура

NewItems(lines) она заполняет строками выпадающий список для InputItem. Список строк записывается в квадратных скобках через запятую.

Для ввода параметров из компонента AddMemo, следует использовать функции.

RGet(NumStr) она выполняет ввод вещественного значения из строки Numstr в AddMemo.

IGet(NumStr) она выполняет ввод целого значения из строки Numstr в AddMemo.

SGet(NumStr) она выполняет ввод строкового значения из строки Numstr в AddMemo.

BGet(NumStr) она выполняет ввод логического значения из строки Numstr в AddMemo.

FileNameGet Вызывает диалог открытия файла и возвращает его имя как строку.

FileNamePut Вызывает диалог закрытия файла и возвращает его имя как строку.

Приведем пример работы с рассмотреннными процедурами и блоками.

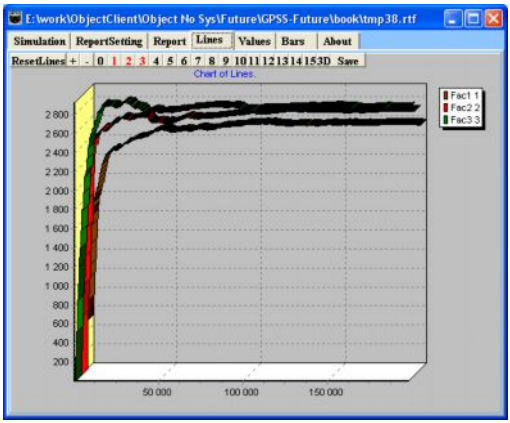
Пусть в систему поступают заявки с интервалом 100, распределенным по экспоненциальному закону. Пусть обслуживание заявок ведется набором устройств, каждое из которых имеет время обслуживания, равное в среднем 500 + 40*номер устройства. Отклонение времени обслуживания от среднего составляет 30%. Для обслуживания выбирается первое попавшееся устройство из 15 имеющихся устройств, у которого длина очереди меньше 5. Если таковое устройство не найдется, то использовать последнее. Определить общее и среднее время обслуживания 2000 заявок. Протабулировать время обслуживания в каждом из устройств. Вывести изменение значение среднего времени обслуживания для первых 3 устройств и гистограммы первых трех таблиц. В процессе моделирования отображать текущее состояние блоков.

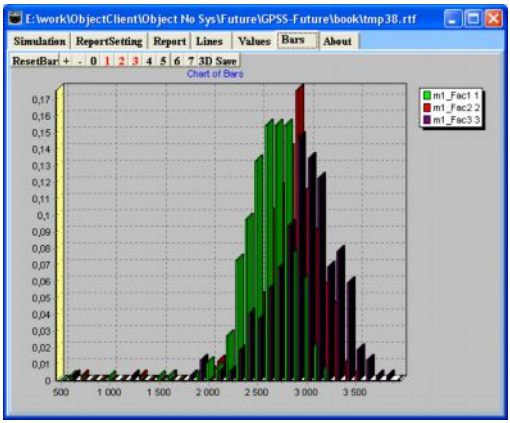
Модель в этом может иметь вид:

```
Const
num=15;
   {~vb} Var
 {~f}Future:TFuture;
{/fac0} fac0:array[1..num] of Tfacility;
{/que0} que0:array[1..num] of Tqueue;
{/Tab} Tab:array[1..num] of Ttable;
  {~ve}
  {~pfe}
  {~ib} procedure Initial;  var  i:integer; begin
 {~f}init(Future,'Future');
```

```
    setstart(2000);
{/fac0}  for i:=1 to num do init(fac0[i], 'fac0 '+inttostr(i));
{/que0}  for i:=1 to num do init(que0[i], ' que0 '+inttostr(i));
{/Tab}   for i:=1 to num do init(Tab[i], ' Tab '+inttostr(i),0,100,100);
{=Gen} Future.NewWaitProc(Gen,exponential(100),0,Gen,Gen+1);
  {~ie}end;
  {~mtb}procedure ModelTxt;begin case ActiveBlock of
{=Gen} ::Gen *:Future.NewWait(exponential(100),0,Gen,Gen+1);
   *: IPlet(1,[0]);
::rret  *: IPlet(1,[ip(1)+1]);
      *: test([(que0[IP(1)].L<4)or(IP(1)=num),true],[!*,rret]) ;
   *: que0[IP(1)].Queue;
   *: fac0[IP(1)].Seize;
   *: que0[IP(1)].Depart;
   *:Future.Wait(MeanDev(500+40*ip(1),0.3*(500+40*ip(1))));
   *: fac0[IP(1)].Release;
   *: tab[IP(1)].tabulate(TranAge) ;
   *: topoint(1,aclock, tab[1].a) ;
   *: topoint(2, aclock , tab[2].a) ;
   *: topoint(3, aclock, tab[3].a) ;
   *: currentblocks ;
   *: Terminate(1);
  {~mte} end;end;
  {~mb} procedure Simulation;  begin
  lineTitle(1,'Fac1');
  lineTitle(2,'Fac2');
  lineTitle(3,'Fac3');
 barTitle(1,'m1_Fac1');
 barTitle(2,'m1_Fac2');
 barTitle(3,'m1_Fac3');
    start(GetTerm);
   show(tab[1],1);
   show(tab[2],2);
   show(tab[3],3);
  {~me}end;
  {~rb} procedure Report;   begin
  {~re} end;
  {~cab} procedure CloseAllObj;begin
  {~cae} end;
```

Вместо отчета, который в данном случае особого интереса не представляет, приведем графики и гистограммы модели.





27 Работа с семействами заявок

В GPSS - Future имеются блоки, обеспечивающие работу с семействами.

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

*: Split(Num,NumBl,NumIP);

Блок создает Num копий заявок соответствующего семейства, и отправляет их по метке NumBl. NumIP – это номер IP - параметра для последовательной нумерации созданных заявок. Если параметр опущен, то заявки не нумеруются. При опущенном параметре Numbl – новые заявки переходят в следующий блок. После создания копий, заявка пытается перейти к следующему по номеру блоку.

Если, например, задан параметр NumIP как число 3, и в родительской заявке IP(3)=B, то параметру IP(3) копий будут присвоены значения B+1, B+2, B+3 и т.д.

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

Каждая копия становится членом семейства заявок, порожденного исходной заявкой, которая была создана блоком Future.NewWait. 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 Future.NewWait заявка создает новое семейство из одной заявки, а часть заявок удаляется из системы.

Узнать номер семейства активной заявки можно с помощью функции Assem.

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

*: Adopt(NumA1,ATran); Блок устанавливает новый номер семейства NumA1 активной заявке (по умолчанию), или заявке с заданным номером ATran.

В блоке SPLIT в качестве номера семейства используется поле Assem, которое первоначально содержит номер TRAN родительской заявки.

Понятие семейства используется блокам Assemble, Gather, MATCH.

Блок Assemble имеет следующий формат:

*: Assemble(Num);

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.

Параметр Num - 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:

```
  {~vb} Var
 {~f}Future:TFuture;
{\Sys} Sys:Tparam;
  {~ve}
  {~pfe}
  {~ib} procedure Initial;begin
 {~f}init(Future,'Future');
  setstart(1);
{\Sys} Init(Sys,'Sys');
{=Gen0} Future.NewWaitProc(Gen0,exponential(100),0,Gen0,Gen0+1);
{=Gen1} Future.NewWaitProc(Gen1,2000,0,Gen1,Gen1+1);
  {~ie}end;
  {~mtb}procedure ModelTxt;begin case ActiveBlock of
{=Gen0} ::Gen0 *:Future.NewWait(exponential(100),0,Gen0,Gen0+1);
```

```
    *:IPlet(1,[duniform(3,17)]) ;
    *: split(Ip(1)-1)  ;
   *:Future.Wait(MeanDev(300,290));
    *: assemble(Ip(1)) ;
   *: tostring(IP(1, sys.N),' ac1 '+Floattostr(aclock)+' M1 '+Floattostr(TranAge)
       +' P= '+inttostr(Ip(1))+' Assem '+inttostr(assem)) ;
   *: IPlet(1,[IP(1, sys.N)+1] , sys.N) ;
   *: terminate ;
{=Gen1} ::Gen1 *:Future.NewWait(2000,0,Gen1,Gen1+1);
    *: terminate(1) ;
  {~mte} end;end;
  {~mb} procedure Simulation;  begin
   start(GetTerm);
  Addsave('test.txt');
  {~me}end;
  {~rb} procedure Report;begin
  {~re} end;
  {~cab} procedure CloseAllObj;begin
  {~cae} end;
```

Выходная статистика модели имеет вид:

```
Date Time Stamp= 19.12.2008 20:50:28   Ctrl Click |Name - HyperLink `R
E:\work\ObjectClient\Object No Sys\Future\GPSS-Future\book\tmp34.rtf
StartTime         0.00000 EndTime       2000.00000
 |Model |Blocks |Values |CurrentList |SynchronizeList |InterruptList
 |RPList |IPList |SPList |BPList
   Num      NameObject
     2        |Future
     3           |Sys
 |R `Model
 {=Gen0} ::Gen0 1:Future.NewWait(exponential(100),0,Gen0,Gen0+1);
     2:IPlet(1,[duniform(3,17)]) ;
     3: split(Ip(1)-1)  ;
    4:Future.Wait(MeanDev(300,290));
     5: assemble(Ip(1)) ;
    6: tostring(IP(1, sys.N),' ac1 '+Floattostr(aclock)+' M1 '+Floattostr(TranAge)
         +' P= '+inttostr(Ip(1))+' Assem '+inttostr(assem)) ;
    7: IPlet(1,[IP(1, sys.N)+1] , sys.N) ;
    8: terminate ;
{=Gen1} ::Gen1 9:Future.NewWait(2000,0,Gen1,Gen1+1);
     10: terminate(1) ;

 |R `Blocks    BLOCK Report
    Location     Total   Current
         1        19         0
         2        19         0
         3        19         0
         4       207        26
         5       181         8
         6        16         0
         7        16         0
         8        16         0
         9         1         0
        10         1         0
 |R `Values
 |R `SynchronizeList Report Synchronize Transactions List Count= 8
      Tran     Assem  Priority   Current      Next       TimeStamp       TimeEnter Inter
       170       170         0         5         6      1701.02790      1864.87183 FALSE
       179       170         0         5         6      1701.02790      1941.03011 FALSE
       183       170         0         5         6      1701.02790      1857.48887 FALSE
       186       170         0         5         6      1701.02790      1874.32979 FALSE
```

```
      181       170         0         5         6      1701.02790      1824.09768 FALSE
      176       176         0         5         6      1891.09220      1997.21255 FALSE
      193       176         0         5         6      1891.09220      1934.17103 FALSE
      198       187         0         5         6      1895.88655      1931.08160 FALSE
|R `IPList Report ( Integer ) Count= 34
       Tran            Num/Name        Integer
        170                   1             11
        176                   1             10
        177                   1             11
        178                   1             11
        179                   1             11
        180                   1             11
        181                   1             11
        182                   1             11
        183                   1             11
        184                   1             11
        185                   1             11
        186                   1             11
        187                   1             13
        188                   1             10
        189                   1             10
        190                   1             10
        191                   1             10
        192                   1             10
        193                   1             10
        194                   1             10
        195                   1             10
        196                   1             10
        198                   1             13
        199                   1             13
        200                   1             13
        201                   1             13
        202                   1             13
        203                   1             13
        204                   1             13
        205                   1             13
        206                   1             13
        207                   1             13
        208                   1             13
        209                   1             13
|R User/Future = `Future Entries = 229 Current = 28 AverUser = 30.52690 AverTime = 266.61049
   Max = 77 Min = 0 Active Time = 2000.00000 NextGo = 0
|R Report  User/Future List Count= 28
 Tran     Assem  Priority   Current      Next       TimeStamp       SortValue     Group Inter
 209       187         0         4         5      1895.88655      2065.06675         0 FALSE
 195       176         0         4         5      1891.09220      2071.60694         0 FALSE
 191       176         0         4         5      1891.09220      2096.21671         0 FALSE
 177       170         0         4         5      1701.02790      2101.11669         0 FALSE
 182       170         0         4         5      1701.02790      2106.58407         0 FALSE
 180       170         0         4         5      1701.02790      2114.72349         0 FALSE
 187       187         0         4         5      1895.88655      2145.09334         0 FALSE
 190       176         0         4         5      1891.09220      2146.91985         0 FALSE
 188       176         0         4         5      1891.09220      2147.17934         0 FALSE
 194       176         0         4         5      1891.09220      2166.45039         0 FALSE
 178       170         0         4         5      1701.02790      2176.67909         0 FALSE
 206       187         0         4         5      1895.88655      2194.62880         0 FALSE
 184       170         0         4         5      1701.02790      2231.35468         0 FALSE
 189       176         0         4         5      1891.09220      2235.15333         0 FALSE
 185       170         0         4         5      1701.02790      2237.28029         0 FALSE
 196       176         0         4         5      1891.09220      2258.20508         0 FALSE
 192       176         0         4         5      1891.09220      2274.92941         0 FALSE
 205       187         0         4         5      1895.88655      2282.32320         0 FALSE
 208       187         0         4         5      1895.88655      2282.72858         0 FALSE
 197       197         0         0         1      2309.58294      2309.58294         0 FALSE
 202       187         0         4         5      1895.88655      2309.91938         0 FALSE
 204       187         0         4         5      1895.88655      2330.13973         0 FALSE
 201       187         0         4         5      1895.88655      2360.50846         0 FALSE
 200       187         0         4         5      1895.88655      2386.45924         0 FALSE
 207       187         0         4         5      1895.88655      2407.32167         0 FALSE
 203       187         0         4         5      1895.88655      2412.72478         0 FALSE
```

```
  199       187         0         4         5      1895.88655      2433.41658         0 FALSE
  210       210         0         0         9      4000.00000      4000.00000         0 FALSE
            Num/Name        Integer  |R `Future      Report IP LIST  Count= 4
I                 1             20
I                 2            197
I                 9              2
I                10            210
            Num/Name        Integer  |R `Sys      Report IP LIST  Count= 1
I                 1             16
```

Информация в выходном файле имеет вид:

ac1 740,648277792271 M1 495,269874450751 P= 12 Assem 3
ac1 780,093259646367 M1 573,426589868031 P= 16 Assem 1
ac1 849,804090326853 M1 376,40181502793 P= 3 Assem 19
ac1 1179,28675590521 M1 581,283438806422 P= 13 Assem 31
ac1 1188,3101507766 M1 436,651071128435 P= 8 Assem 62
ac1 1240,61277044264 M1 564,168923986144 P= 8 Assem 47
ac1 1250,67627228663 M1 580,724136293866 P= 15 Assem 34
ac1 1346,86241214049 M1 482,539103501476 P= 10 Assem 78
ac1 1363,22259244921 M1 524,310055482201 P= 13 Assem 70
ac1 1383,92998017413 M1 486,71413990669 P= 9 Assem 91
ac1 1454,01493883355 M1 514,353553419933 P= 5 Assem 110
ac1 1512,74460805151 M1 588,429100643843 P= 15 Assem 101
ac1 1737,08017481664 M1 450,256279963069 P= 8 Assem 147
ac1 1799,95922672132 M1 542,177032921463 P= 15 Assem 130
ac1 1803,43070826183 M1 587,439403794706 P= 17 Assem 125
ac1 1977,3524251216 M1 551,603623367846 P= 6 Assem 162
ac1 789,769981049556 M1 404,35008096043 P= 4 Assem 13
ac1 808,1871894559 M1 547,148886146024 P= 10 Assem 1
ac1 851,199071484527 M1 587,777505856939 P= 10 Assem 36

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.

```
  {~vb} Var
 {~f}Future:TFuture;
{/QLine} QLine:Tqueue;
{/QUst} QUst:Tqueue;
{/Line} Line:Tfacility;
{/Ust} Ust:Tfacility;
  {~ve}
  {~pfe}
```

```
 {~ib} procedure Initial;begin
{~f}init(Future,'Future');
 setstart(1000);
{/QLine}  init(QLine,' QLine');
{/QUst}  init(QUst,' QUst');
{/Line}  init(Line,' Line');
{/Ust}  init(Ust,' Ust');
{=Gen2} Future.NewWaitProc(Gen2,Exponential(750),0,Gen2,Gen2+1);
 {~ie}end;
 {~mtb}procedure ModelTxt;begin case ActiveBlock of
{=Gen2} ::Gen2 *:Future.NewWait(Exponential(750),0,Gen2,Gen2+1);
        *: Qline.queue ;
        *: Line.seize ;
        *: Qline.Depart ;
        *: IPlet(1,[duniform(3,12)]) ;
        *: IPlet(2,[IP(1)]) ;
        *:Future.Wait(MeanDev(90,20));
        *: transfer(bbe2) ;
::bbc2   *: split(1,bg2);
        *:Future.Wait(MeanDev(90,20));
::bbe2  *: loop(1,bbc2) ;
        *: Line.Release ;
::bg2   *: Qust.queue ;
       *: ust.seize ;
      *: Qust.Depart ;
        *:Future.Wait(MeanDev(80,50));
       *: Ust.Release ;
    *: Assemble(IP(2)) ;
    *: Terminate(1)  ;
 {~mte} end;end;
 {~mb} procedure Simulation;  begin
  start(GetTerm);
 {~me}end;
 {~rb} procedure Report;begin
 {~re} end;
 {~cab} procedure CloseAllObj;begin
 {~cae} end;
```

Итоговая статистика этой модели будет иметь вид:

```
  Date Time Stamp= 19.12.2008 21:12:04   Ctrl Click |Name - HyperLink `R
E:\work\ObjectClient\Object No Sys\Future\GPSS-Future\book\tmp35.rtf
StartTime         0.00000 EndTime     759092.67016
 |Model |Blocks |Values |CurrentList |SynchronizeList |InterruptList
 |RPList |IPList |SPList |BPList
   Num      NameObject
     2        |Future
     3        | QLine
     4         | QUst
     5         | Line
     6          | Ust
 |R `Model
```

```
{=Gen2} ::Gen2 1:Future.NewWait(Exponential(750),0,Gen2,Gen2+1);
         2: Qline.queue ;
         3: Line.seize ;
         4: Qline.Depart ;
         5: IPlet(1,[duniform(3,12)]) ;
         6: IPlet(2,[IP(1)]) ;
         7:Future.Wait(MeanDev(90,20));
         8: transfer(bbe2) ;
::bbc2   9: split(1,bg2);
         10:Future.Wait(MeanDev(90,20));
::bbe2  11: loop(1,bbc2) ;
         12: Line.Release ;
::bg2   13: Qust.queue ;
         14: ust.seize ;
        15: Qust.Depart ;
         16:Future.Wait(MeanDev(80,50));
         17: Ust.Release ;
      18: Assemble(IP(2)) ;
      19: Terminate(1)  ;

|R `Blocks    BLOCK Report
   Location     Total   Current
        1      1014         0
        2      1014        13
        3      1001         0
        4      1001         0
        5      1001         0
        6      1001         0
        7      1001         1
        8      1000         0
        9      6537         0
       10      6537         0
       11      7537         0
       12      1000         0
       13      7537         0
       14      7537         0
       15      7537         0
       16      7537         0
       17      7537         0
       18      7537         0
       19      1000         0
|R `Values
|R `CurrentList Report Current Transactions List  Count= 13
   Tran   Assem     Pri Current    Next       TimeStamp       TimeEnter Inter
   7442    7442       0       2       3    750980.80250    750980.80250 FALSE
   7463    7463       0       2       3    751914.95178    751914.95178 FALSE
   7474    7474       0       2       3    753040.94599    753040.94599 FALSE
   7484    7484       0       2       3    753338.72428    753338.72428 FALSE
   7488    7488       0       2       3    753449.83495    753449.83495 FALSE
   7490    7490       0       2       3    754179.24885    754179.24885 FALSE
   7498    7498       0       2       3    754742.93508    754742.93508 FALSE
   7505    7505       0       2       3    755466.55698    755466.55698 FALSE
   7513    7513       0       2       3    755806.42114    755806.42114 FALSE
   7517    7517       0       2       3    756410.82516    756410.82516 FALSE
   7524    7524       0       2       3    756863.45084    756863.45084 FALSE
   7529    7529       0       2       3    757212.80862    757212.80862 FALSE
   7533    7533       0       2       3    757945.62493    757945.62493 FALSE
|R `IPList Report ( Integer ) Count= 2
       Tran            Num/Name        Integer
       7437                   1              7
       7437                   2              7
|R Facility = ` Line Entries = 1001 Current = 1 Utility =  0.89468 AverTime = 678.46473
   Tran = 7437 IntTran = 0 Active Time = 759092.67016 NextGo = 0
|R Facility = ` Ust Entries = 7537 Current = 0 Utility =  0.79585 AverTime = 80.15414
   Tran = 0 IntTran = 0 Active Time = 759092.67016 NextGo = 0
|R Queue = ` QLine Entries = 1014 Current = 13 AverQueue =  3.09976 AverTime = 2320.52082
   AverTime(-Ze) = 2948.63172 Max = 15 Min = 0 Zero = 216 Active Time = 759092.67016 NextGo = 0
|R Queue = ` QUst Entries = 7537 Current = 0 AverQueue =  0.31496 AverTime = 31.72185
   AverTime(-Ze) = 233.02882 Max = 4 Min = 0 Zero = 6511 Active Time = 759092.67016 NextGo = 0
|R User/Future = `Future Entries = 16090 Current = 2 AverUser =  1.81638 AverTime = 85.69325
```

```
    Max = 3 Min = 0 Active Time = 759092.67016 NextGo = 0
 |R Report  User/Future List Count= 2
    Tran   Assem     Pri Current    Next       TimeStamp       SortValue   Group Inter
    7437    7437       0       7       8    749081.86876    759103.95586       0 FALSE
    7541    7541       0       0       1    759371.96544    759371.96544       0 FALSE
            Num/Name        Integer  |R `Future      Report IP LIST  Count= 2
I                 1           1015
I                 2           7541
```

Блок Gather имеет следующий формат.

*: Gather(Num);

Здесь Num – 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 70 минут, распределенном по экспоненциальному закону. Аппарат включают, только когда есть 5 электродвигателей. Процесс мойки и сушки занимает 280+-20 минут. Промоделировать процесс в течение 24000 минут.

Модель такой задачи может иметь вид.

```
  {~vb} Var
 {~f}Future:TFuture;
{/Qapp} Qapp:Tqueue;
{/app} app:Tstorage;
  {~ve}
  {~pfe}
  {~ib} procedure Initial;begin
 {~f}init(Future,'Future');
    setstart(1);
{/Qapp}  init(Qapp,' Qapp');
{/app}  init(app,' Qapp',5);
{=Gen} Future.NewWaitProc(Gen,Exponential(70),0,Gen,Gen+1);
{=Gen0} Future.NewWaitProc(Gen0,240000,0,Gen0,Gen0+1);
  {~ie}end;
  {~mtb}procedure ModelTxt;begin case ActiveBlock of
{=Gen} ::Gen *:Future.NewWait(Exponential(70),0,Gen,Gen+1);
         *: adopt(1) ;
        *: Qapp.Queue ;
         *: Gather(5) ;
        *: app.enter ;
        *: Qapp.depart ;
       *:Future.Wait(MeanDev(280,20));
         *: Gather(5) ;
        *: app.Leave ;
        *: terminate ;
{=Gen0} ::Gen0 *:Future.NewWait(240000,0,Gen0,Gen0+1);
        *: terminate(1)  ;
  {~mte} end;end;
  {~mb} procedure Simulation;  begin
   start(GetTerm);
```

{~me}end;
{~rb} procedure Report;begin
{~re} end;
{~cab} procedure CloseAllObj;begin
{~cae} end;

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:

```
 Date Time Stamp= 19.12.2008 21:17:42   Ctrl Click |Name - HyperLink `R
E:\work\ObjectClient\Object No Sys\Future\GPSS-Future\book\tmp36.rtf
StartTime         0.00000 EndTime     240000.00000
 |Model |Blocks |Values |CurrentList |SynchronizeList |InterruptList
 |RPList |IPList |SPList |BPList
   Num      NameObject
     2        |Future
     3         | Qapp
     4         | Qapp
 |R `Model
 {=Gen} ::Gen 1:Future.NewWait(Exponential(70),0,Gen,Gen+1);
          2: adopt(1) ;
         3: Qapp.Queue ;
          4: Gather(5) ;
         5: app.enter ;
         6: Qapp.depart ;
        7:Future.Wait(MeanDev(280,20));
          8: Gather(5) ;
         9: app.Leave ;
         10: terminate()  ;
{=Gen0} ::Gen0 11:Future.NewWait(240000,0,Gen0,Gen0+1);
         12: terminate(1)  ;

 |R `Blocks    BLOCK Report
    Location     Total   Current
        1      3455         0
        2      3455         0
        3      3455         0
        4      3455         5
        5      3450         0
        6      3450         0
        7      3450         2
        8      3448         3
        9      3445         0
       10      3445         0
       11         1         0
       12         1         0
 |R `Values
 |R `CurrentList Report Current Transactions List  Count= 5
    Tran   Assem     Pri Current    Next       TimeStamp       TimeEnter Inter
    3452       1       0       4       5    239714.14603    239714.14603 FALSE
    3453       1       0       4       5    239816.46545    239816.46545 FALSE
    3454       1       0       4       5    239838.48756    239838.48756 FALSE
    3455       1       0       4       5    239847.40372    239847.40372 FALSE
    3456       1       0       4       5    239927.05633    239927.05633 FALSE
 |R `SynchronizeList Report Synchronize Transactions List Count= 3
    Tran   Assem     Pri Current    Next       TimeStamp       TimeEnter Inter
    3447       1       0       8       9    239255.21156    239975.07255 FALSE
    3448       1       0       8       9    239266.55268    239989.56119 FALSE
    3451       1       0       8       9    239709.96900    239971.46560 FALSE
 |R Storage = ` Qapp Entries = 3450 Current = 5 Average =  4.22285 AverTime = 293.76363 Utility
=  0.84457
    Max = 5 Min = 0 Tran = 3451 ACapac =  5.00000 Capacity = 5 Active Time = 240000.00000 NextGo
= 0
 |R Queue = ` Qapp Entries = 3455 Current = 5 AverQueue =  3.59319 AverTime = 249.59910
    AverTime(-Ze) = 249.59910 Max = 23 Min = 0 Zero = 0 Active Time = 240000.00000 NextGo = 0
 |R User/Future = `Future Entries = 6908 Current = 4 AverUser =  5.74837 AverTime = 199.71175
    Max = 7 Min = 0 Active Time = 240000.00000 NextGo = 0
```

```
 |R Report  User/Future List Count= 4
    Tran   Assem     Pri Current    Next       TimeStamp       SortValue   Group Inter
    3450       1       0       7       8    239681.38270    240000.58284       0 FALSE
    3449       1       0       7       8    239611.34599    240007.06699       0 FALSE
    3457    3457       0       0       1    240048.12284    240048.12284       0 FALSE
    3458    3458       0       0      11    480000.00000    480000.00000       0 FALSE
            Num/Name        Integer  |R `Future      Report IP LIST  Count= 4
I                 1           3456
I                 2           3457
I                11              2
I                12           3458
```

Блок Match имеет следующий формат:

*: Match(NumBl);

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 NumBl 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:

```
 {~vb} Var
 {~f}Future:TFuture;
{/beg} beg:Ttable;
{/wait1} wait1:Ttable;
{/wait2} wait2:Ttable;
 {~ve}
 {~pfe}
 {~ib} procedure Initial;begin
 {~f}init(Future,'Future');
{/beg}  init(beg,' beg',0,0.8,30);
{/wait1}  init(wait1,' wait1',0,0.8,30);
{/wait2}  init(wait2,' wait2',0,0.8,30);
setstart( 1000);
{=Gen} Future.NewWaitProc(Gen,0,0,Gen,Gen+1);
 {~ie}end;
 {~mtb}procedure ModelTxt;begin case ActiveBlock of
{=Gen} ::Gen *:Future.NewWait(0,0,Gen,Gen+1,0,true,1);
::mm5 *:split( 1,next);
 *:mark;
```

```
 *:Future.Wait(MeanDev( 2.5,2.5));
::mm1 *:match( mm2);
  *:beg.tabulate( TranAge);
 *:Future.Wait(MeanDev(10,3));
 *:Future.Wait(MeanDev(10,3));
 *:Future.Wait(MeanDev(10,3));
 *:Future.Wait(MeanDev(10,3));
 *:Future.Wait(MeanDev(10,3));
  *:mark;
::mm3 *:match (mm4);
  *:wait1.tabulate (TranAge);
  *:terminate( 1);
::next *:mark;
 *:Future.Wait(MeanDev( 2.5,2.5));
::mm2 *:match( mm1);
  *:beg.tabulate( TranAge);
 *:Future.Wait(MeanDev(9.8,4));
 *:Future.Wait(MeanDev(9.8,4));
 *:Future.Wait(MeanDev(9.8,4));
 *:Future.Wait(MeanDev(9.8,4));
 *:Future.Wait(MeanDev(9.8,4));
 *:mark;
::mm4 *:match( mm3);
  *:wait2.tabulate( TranAge);
  *:transfer (mm5);
  {~mte} end;end;
  {~mb} procedure Simulation;  begin
    start(GetTerm);
show(wait1,0);
show(wait2,1)
   {~me}end;
  {~rb} procedure Report;begin
  {~re} end;
  {~cab} procedure CloseAllObj;begin
  {~cae} end;
```

Итоговая статистика модели может иметь вид:

```
Date Time Stamp= 19.12.2008 21:26:11   Ctrl Click |Name - HyperLink `R
E:\work\ObjectClient\Object No Sys\Future\GPSS-Future\book\tmp37.rtf
StartTime         0.00000 EndTime      55449.15581
 |Model |Blocks |Values |CurrentList |SynchronizeList |InterruptList
 |RPList |IPList |SPList |BPList
   Num      NameObject
     2        |Future
     3          | beg
     4        | wait1
     5        | wait2
 |R `Model
 {=Gen} ::Gen 1:Future.NewWait(0,0,Gen,Gen+1,0,true,1);
::mm5 2:split( 1,next);
  3:mark;
 4:Future.Wait(MeanDev( 2.5,2.5));
::mm1 5:match( mm2);
```

```
  6:beg.tabulate( TranAge);
 7:Future.Wait(MeanDev(10,3));
 8:Future.Wait(MeanDev(10,3));
 9:Future.Wait(MeanDev(10,3));
 10:Future.Wait(MeanDev(10,3));
 11:Future.Wait(MeanDev(10,3));
  12:mark;
::mm3 13:match (mm4);
  14:wait1.tabulate (TranAge);
  15:terminate( 1);
::next 16:mark;
 17:Future.Wait(MeanDev( 2.5,2.5));
::mm2 18:match( mm1);
  19:beg.tabulate( TranAge);
 20:Future.Wait(MeanDev(9.8,4));
 21:Future.Wait(MeanDev(9.8,4));
 22:Future.Wait(MeanDev(9.8,4));
 23:Future.Wait(MeanDev(9.8,4));
 24:Future.Wait(MeanDev(9.8,4));
 25:mark;
::mm4 26:match( mm3);
  27:wait2.tabulate( TranAge);
  28:transfer (mm5);

 |R `Blocks    BLOCK Report
    Location     Total   Current
         1         1         0
         2      1000         0
         3      1000         0
         4      1000         0
         5      1000         0
         6      1000         0
         7      1000         0
         8      1000         0
         9      1000         0
        10      1000         0
        11      1000         0
        12      1000         0
        13      1000         0
        14      1000         0
        15      1000         0
        16      1000         0
        17      1000         0
        18      1000         0
        19      1000         0
        20      1000         0
        21      1000         0
        22      1000         0
        23      1000         0
        24      1000         0
        25      1000         0
        26      1000         1
        27       999         0
        28       999         0
 |R `Values
 |R `CurrentList Report Current Transactions List  Count= 1
    Tran   Assem     Pri Current    Next       TimeStamp       TimeEnter Inter
    1001       1       0      26      27     55449.15581     55449.15581 FALSE
 |R User/Future = `Future Entries = 12001 Current = 0 AverUser =  0.87959 AverTime =  4.06404
    Max = 2 Min = 0 Active Time = 55449.15581 NextGo = 0
            Num/Name        Integer  |R `Future      Report IP LIST  Count= 2
I                 1              1
I                 2              1
 |R  Table = ` beg  Entries = 2000.00000 Mean =  3.31769 StdDev =  1.18530
          Range       Frequency        Integral
        0.00000         0.00000         0.00000
        0.80000        48.00000         0.02400
        1.60000       160.00000         0.10400
        2.40000       268.00000         0.23800
        3.20000       354.00000         0.41500
```

```
        4.00000       468.00000         0.64900
        4.80000       542.00000         0.92000
        5.60000       160.00000         1.00000
        6.40000         0.00000         1.00000
|R  Table = ` wait1  Entries = 1000.00000 Mean =  2.24965 StdDev =  3.34391
          Range       Frequency        Integral
        0.00000       527.00000         0.52700
        0.80000        45.00000         0.57200
        1.60000        46.00000         0.61800
        2.40000        48.00000         0.66600
        3.20000        49.00000         0.71500
        4.00000        40.00000         0.75500
        4.80000        40.00000         0.79500
        5.60000        40.00000         0.83500
        6.40000        34.00000         0.86900
        7.20000        22.00000         0.89100
        8.00000        24.00000         0.91500
        8.80000        21.00000         0.93600
        9.60000        22.00000         0.95800
       10.40000        13.00000         0.97100
       11.20000         5.00000         0.97600
       12.00000         8.00000         0.98400
       12.80000         6.00000         0.99000
       13.60000         3.00000         0.99300
       14.40000         2.00000         0.99500
       15.20000         2.00000         0.99700
       16.00000         0.00000         0.99700
       16.80000         0.00000         0.99700
       17.60000         1.00000         0.99800
       18.40000         1.00000         0.99900
       19.20000         0.00000         0.99900
       20.00000         1.00000         1.00000
       20.80000         0.00000         1.00000
|R  Table = ` wait2  Entries = 999.00000 Mean =  2.78835 StdDev =  3.84777
          Range       Frequency        Integral
        0.00000       472.00000         0.47247
        0.80000        42.00000         0.51451
        1.60000        51.00000         0.56557
        2.40000        44.00000         0.60961
        3.20000        62.00000         0.67167
        4.00000        43.00000         0.71471
        4.80000        40.00000         0.75475
        5.60000        34.00000         0.78879
        6.40000        36.00000         0.82482
        7.20000        34.00000         0.85886
        8.00000        29.00000         0.88789
        8.80000        17.00000         0.90490
        9.60000        16.00000         0.92092
       10.40000        18.00000         0.93894
       11.20000        13.00000         0.95195
       12.00000         9.00000         0.96096
       12.80000         8.00000         0.96897
       13.60000        11.00000         0.97998
       14.40000         6.00000         0.98599
       15.20000         6.00000         0.99199
       16.00000         1.00000         0.99299
       16.80000         4.00000         0.99700
       17.60000         1.00000         0.99800
       18.40000         2.00000         1.00000
       19.20000         0.00000         1.00000
```

Из выходной статистики видно, что первый конкурсант оказался вторым 527 раз и отставал от второго в среднем почти на 2.2 минуты. А второй конкурсант оказался вторым – 472 раз и отставал от первого в среднем на 2.79 минуты.

Среднее время начала конкурса оказалось на 3.32 смещенным относительно официального начала.

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.

Здесь, и далее ObjName – это имя списка.

Блок Link имеет вид.

*: ObjName.Link (TestGo,NumGo,Value<,Gr1);

*:ObjName.Link(Value<,Gr1); Если TestGo - истина, то текущая заявка уходит на блок с меткой NumGo. Иначе блок помещает активную заявку в список. Выражение Value используется для упорядочения списка. Не обязательный целый параметр Gr1 также используется для упорядочения, причем он учитывается до Value. Это дает дополнительные возможности для формирования списка пользователя.

Для удаления заявок из списка используется блок UnLink, он имеет вид.

*:ObjName. UnLink(NumBl, Num, Log, Cont<, IPCount<, NewAClock<, NumGr<, NumVal<, NumCurr<, NumNext); Блок удаляет заявки из списка. Здесь Numbl - метка, куда уйдут удаляемые заявки. Если Numbl=0 то заявка уйдет на тот блок, который следовал после блока Link, который поставил заявку в список пользователя. Num- максимальное число удаляемых заявок. При Num=0 удаляются все заявки. При Num>0 заявки удаляются из начала списка, при Num<0 - с конца. Удалению подлежат только те заявки, для которых значение функции Log - истинно. Здесь Log - это отдельно описанная функция без параметров с результатом логического типа. Удаление прекращается, когда функция Cont окажется ложной. Если нужно выбрать любые заявки, то в качестве функций Log и Cont указывается функция Any. Количество реально удаленных заявок, записывается в IP(IpCount) активной заявки. Если опущен параметр IPCount, то количество реально удаленных заявок никуда не записывается. NewAClock - как правило, должен быть равен 0, так как он влияет на текущее время моделирования. В удаленной из списка заявке в IP(NumGr) и RP(NumVal) может записываться значение функций Gr и Val для этой заявки, а в IP(NumCurr) и IP(NumNext) - может записываться прежний текущий и следующий блок для удаленной заявки.

*: ObjName. UnLink(NumBl, Num, Gr1, Log, Cont<, IPCount<, NewAClock<, NumGr<, NumVal<, NumCurr<, NumNext); Блок аналогичен предыдущему, но в дополнение учитывается требование, чтобы дополнительный параметр Gr был равен Gr1. Блок всегда работает быстрее, чем предыдущая версия.

*:ObjName.Select(AAXn1,Num, Log,Cont) ;

*:ObjName.Select(AAXn1,Num, Gr1, Log,Cont); Блок записывает в динамический массив AAXn1 номера заявок из списка пользователя. Смысл параметров Num, Gr1, Log, Cont - такой же, как в UnLink.

Для эффективной работы со списками можно использовать следующие функции

ObjName.A Определяет среднее число заявок в списке.

ObjName.V Определяет временной интеграл списка.

ObjName.T Определяет среднее время, проведенное заявкой в списке.

ObjName.C Определяет общее число занятий списка.

ObjName.M Определяет максимальное число заявок в списке.

ObjName.L Выдает текущее число заявок в списке.

ObjName.I Выдает минимальное число заявок в списке.

ObjName.Assem Выдает номер семейства для заявки из списка.

ObjName.TranAge Выдает время жизни для заявки из списка.

ObjName.Pri Выдает значение приоритета для заявки из списка.

ObjName.Tran Выдает значение номера заявки для заявки из списка.

ObjName.ScanNum Выдает порядковый номер заявки в списке в процессе просмотра с помощью UnLink. В начале просмотра значение функции равно 0.

ObjName.CurrentBlock

ObjName NextBlock Выдает значение текущего и следующего блоков для заявки из списка.

ObjName.DelCount определяет количество удаленных или выбранных заявок, как результат работы функций группы Sum, процедур SelectProc, UnlinkProc и блоков Select, Unlink.

ObjName.RP(Num) Выдает значение RP(Num) для заявки из списка.

ObjName.IP(Num) Выдает значение IP(Num) для заявки из списка.

ObjName.SP(Num) Выдает значение SP(Num) для заявки из списка.

ObjName.BP(Num) Выдает значение BP(Num) для заявки из списка.

ObjName.Gr Определяет значение Gr1 установленное для заявки из списка блоком Link.

ObjName.Val Выдает значение параметра упорядочения заявок для заявки из списка.

А вот более сложные функции. Их не рекомендуется вычислять внутри UnLink, а если нужно, то вычислять заранее и записывать в RP - параметры.

ObjName.Sum(Ssum, Num, Log, Cont)

ObjName.Sum(Ssum, Num, Gr1, Log, Cont)

Определяет значение суммы значений функции SSUM для заявок из списка. Смысл параметров Num, Gr1, Log, Cont - такой же, как в UnLink.

Функции типа Sum рекомендуется вычислять заранее и записывать в RP - параметры.

ObjName. Avg (Ssum, Num, Log, Cont)

ObjName. Avg (Ssum, Num, Gr1, Log, Cont) функция аналогична предыдущей функции, но вычисляет среднее значение.

ObjName. Min (Ssum, Num, Log, Cont)

ObjName. Min (Ssum, Num, Gr1, Log, Cont) функция аналогична предыдущей функции, но вычисляет минимальное значение.

ObjName. Max (Ssum, Num, Log, Cont)

ObjName. Max (Ssum, Num, Gr1, Log, Cont) функция аналогична предыдущей функции, но вычисляет максимальное значение.

ObjName. Count (Num, Log, Cont)

ObjName. Count (Num, Gr1, Log, Cont) функция аналогична предыдущей функции, но вычисляет количество заявок.

ObjName. MinVal

ObjName. MaxVal Функции вычисляют минимальное и максимальное значения Val в списке.

Пример применения блоков Link и Unlink.

Рассмотрим модель работы одноканального устройства, с обслуживанием очереди по алгоритму LIFO:

Заявки поступают в систему с интервалом 1000, распределенным по экспоненциальному закону. Заявки обслуживаются в одноканальном устройстве с интервалом 800, распределенным по экспоненциальному закону. Порядок обслуживания - последним пришел – первым обслужен.

```
  {~vb} Var
 {~f}Future:TFuture;
{/tt} tt:Ttable;
{/och1} och1:Tqueue;
{/fac1} fac1:Tfacility;
{/list} list:Tuser;
  {~ve}
  {~pfe}
  {~ib} procedure Initial;begin
 {~f}init(Future,'Future');
{/tt}  init(tt,'tt',0,2000,100);
{/och1}  init(och1,' och1');
{/fac1}  init(fac1,' fac1');
{/list}  init(list,' list');
        setSTART( 1000 );
{=Gen} Future.NewWaitProc(Gen,exponential(1000),0,Gen,Gen+1);
  {~ie}end;
  {~mtb}procedure ModelTxt;begin case ActiveBlock of
{=Gen} ::Gen *:Future.NewWait(exponential(1000),0,Gen,Gen+1);
          *:och1.QUEUE;
          *:TEST([FAC1.L=0,TRUE],[!*+1,!*]);
           *:LIST.LINK(-AClock);
       ::next     *:FAC1.SEIZE;
          *: OCH1.DEPART;
          *:tt.tabulate(TranAge);
          *:Future.Wait(exponential(800));
           *:FAC1.RELEASE;
          *:LIST. UNLINK(next,1,any,any);
         *: TERMINATE( 1) ;
  {~mte} end;end;
  {~mb} procedure Simulation;  begin
 {/sh}     start(GetTerm);
show(TT,0);
  {~me}end;
  {~rb} procedure Report;begin
  {~re} end;
  {~cab} procedure CloseAllObj;begin
  {~cae} end;
```

Выходная статистика этой модели - следующая.

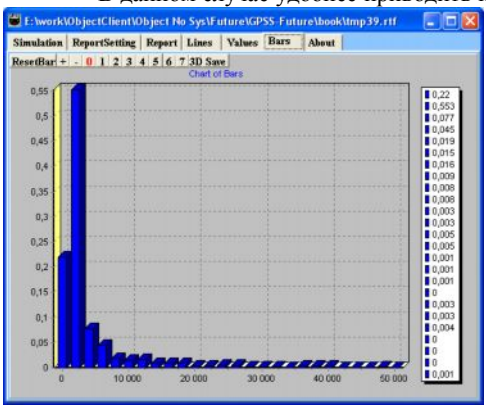
```
Date Time Stamp= 20.12.2008 13:32:20   Ctrl Click |Name - HyperLink `R
E:\work\ObjectClient\Object No Sys\Future\GPSS-Future\book\tmp39.rtf
StartTime         0.00000 EndTime    1051903.14169
 |Model |Blocks |Values |CurrentList |SynchronizeList |InterruptList
 |RPList |IPList |SPList |BPList
```

```
   Num      NameObject
     2        |Future
     3            |tt
     4         | och1
     5         | fac1
     6         | list
 |R `Model
 {=Gen} ::Gen 1:Future.NewWait(exponential(1000),0,Gen,Gen+1);
             2:och1.QUEUE;
             3:TEST([FAC1.L=0,TRUE],[4+1,4]);
              4:LIST.LINK(-AClock);
        ::next     5:FAC1.SEIZE;
             6: OCH1.DEPART;
             7:tt.tabulate(TranAge);
             8:Future.Wait(exponential(800));
              9:FAC1.RELEASE;
             10:LIST. UNLINK(next,1,any,any);
            11: TERMINATE( 1) ;

 |R `Blocks    BLOCK Report
    Location     Total   Current
         1      1008         0
         2      1008         0
         3      1008         0
         4       788         8
         5      1000         0
         6      1000         0
         7      1000         0
         8      1000         0
         9      1000         0
        10      1000         0
        11      1000         0
 |R `Values
 |R `CurrentList Report Current Transactions List  Count= 1
    Tran   Assem     Pri Current    Next       TimeStamp       TimeEnter Inter
     988     988       0       4       5   1035129.49002   1051903.14169 FALSE
 |R Facility = ` fac1 Entries = 1000 Current = 0 Utility =  0.74555 AverTime = 784.24711
    Tran = 0 IntTran = 0 Active Time = 1051903.14169 NextGo = 0
 |R Queue = ` och1 Entries = 1008 Current = 8 AverQueue =  2.48416 AverTime = 2592.35994
    AverTime(-Ze) = 7618.36391 Max = 16 Min = 0 Zero = 665 Active Time = 1051903.14169 NextGo =
0
 |R User/Future = ` list Entries = 788 Current = 7 AverUser =  2.48416 AverTime = 3316.11526
    Max = 16 Min = 0 Active Time = 1051903.14169 NextGo = 0
 |R Report  User/Future List Count= 7
    Tran   Assem     Pri Current    Next       TimeStamp       SortValue   Group Inter
     987     987       0       4       5   1034963.70165  -1034963.70165       0 FALSE
     986     986       0       4       5   1034864.75356  -1034864.75356       0 FALSE
     984     984       0       4       5   1033342.48530  -1033342.48530       0 FALSE
     983     983       0       4       5   1033212.19678  -1033212.19678       0 FALSE
     978     978       0       4       5   1029574.23940  -1029574.23940       0 FALSE
     974     974       0       4       5   1026374.02235  -1026374.02235       0 FALSE
     973     973       0       4       5   1026368.23843  -1026368.23843       0 FALSE
 |R User/Future = `Future Entries = 2009 Current = 1 AverUser =  1.32180 AverTime = 692.08959
    Max = 2 Min = 0 Active Time = 1051903.14169 NextGo = 0
 |R Report  User/Future List Count= 1
    Tran   Assem     Pri Current    Next       TimeStamp       SortValue   Group Inter
    1009    1009       0       0       1   1053683.13258   1053683.13258       0 FALSE
            Num/Name        Integer  |R `Future      Report IP LIST  Count= 2
I                 1           1009
I                 2           1009
 |R  Table = `tt  Entries = 1000.00000 Mean = 2451.70282 StdDev = 5749.53544
          Range       Frequency        Integral
        0.00000       220.00000         0.22000
     2000.00000       553.00000         0.77300
     4000.00000        77.00000         0.85000
     6000.00000        45.00000         0.89500
     8000.00000        19.00000         0.91400
    10000.00000        15.00000         0.92900
    12000.00000        16.00000         0.94500
    14000.00000         9.00000         0.95400
```

16000.00000	8.00000	0.96200
18000.00000	8.00000	0.97000
20000.00000	3.00000	0.97300
22000.00000	3.00000	0.97600
24000.00000	5.00000	0.98100
26000.00000	5.00000	0.98600
28000.00000	1.00000	0.98700
30000.00000	1.00000	0.98800
32000.00000	1.00000	0.98900
34000.00000	0.00000	0.98900
36000.00000	3.00000	0.99200
38000.00000	3.00000	0.99500
40000.00000	4.00000	0.99900
42000.00000	0.00000	0.99900
44000.00000	0.00000	0.99900
46000.00000	0.00000	0.99900
48000.00000	1.00000	1.00000
50000.00000	0.00000	1.00000

В данном случае удобнее приводить не таблицу из отчета, а гистограмму.



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 Future.NewWait, 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.

```
  Const
Num_st=1;
prio=2;
num_ret=3;
Kvant=1;
  {~vb} Var
 {~f}Future:TFuture;
{/tt} tt:Ttable;
{/fac1} fac1:Tfacility;
{/list} list:Tuser;
{/Stor} Stor:Tstorage;
{/List1} List1:Tuser;
{\Sys} Sys:Tparam;
  {~ve}
  {~pfe}
  {~ib} procedure Initial;begin
 {~f}init(Future,'Future');
{/tt}  init(tt,'tt',0,200,100);
{/fac1}  init(fac1,' fac1');
{/list}  init(list,' list');
            setSTART( 1000 );
{/Stor}  init(Stor,' Stor',1024);
{/List1}  init(List1,' List1');
{\Sys} Init(Sys,'Sys');
rPLetProc(Kvant,[50], sys.N);
{=Gen} Future.NewWaitProc(Gen,exponential(150),0,Gen,Gen+1);
  {~ie}end;
  {~mtb}procedure ModelTxt;begin case ActiveBlock of
{=Gen} ::Gen *:Future.NewWait(exponential(150),0,Gen,Gen+1);
          *: IPlet(num_st,[duniform(200,440)])  ;
          *: IPlet(prio,[duniform(0,4)])  ;
          *: IPlet(num_ret,[duniform(1,5)])  ;
::M8   *: TEST([ip(num_st)<=STOR.R,true],[!*,m4]) ;
        *: stor.Enter(ip(num_st)) ;
        *: mark ;
::m2    *:TEST([FAC1.L=0,TRUE],[!*+1,!*]);
    *: List.Link(IP(prio)) ;
::m3     *:FAC1.SEIZE;
          *:Future.Wait(RP(Kvant, sys.N));
```

```
*:FAC1.RELEASE;
*: List.UnLink(m3,1,any,any) ;
*: Loop(num_ret,m2) ;
*: stor.Leave(ip(num_st)) ;
*:LIST1. UNLINK(m8,1,any,any);
*:tt.tabulate(TranAge);
*: TERMINATE( 1) ;
::m4 *:LIST1. LINK(aclock);
{~mte} end;end;
{~mb} procedure Simulation;  begin
{/sh}     start(GetTerm);
show(TT,0);
{~me}end;
{~rb} procedure Report;begin
{~re} end;
{~cab} procedure CloseAllObj;begin
{~cae} end;
```

Выходная статистика модели выглядит следующим образом.

```
Date Time Stamp= 20.12.2008 13:48:07   Ctrl Click |Name - HyperLink `R
E:\work\ObjectClient\Object No Sys\Future\GPSS-Future\book\tmp40.rtf
StartTime         0.00000 EndTime     149920.10602
 |Model |Blocks |Values |CurrentList |SynchronizeList |InterruptList
 |RPList |IPList |SPList |BPList
   Num      NameObject
     2        |Future
     3            |tt
     4         | fac1
     5         | list
     6         | Stor
     7        | List1
     8           |Sys
 |R `Model
 {=Gen} ::Gen 1:Future.NewWait(exponential(150),0,Gen,Gen+1);
            2: IPlet(num_st,[duniform(200,440)])  ;
            3: IPlet(prio,[duniform(0,4)])  ;
            4: IPlet(num_ret,[duniform(1,5)])  ;
::M8   5: TEST([ip(num_st)<=STOR.R,true],[6,m4]) ;
          6: stor.Enter(ip(num_st)) ;
          7: mark ;
::m2    8:TEST([FAC1.L=0,TRUE],[9+1,9]);
     9: List.Link(IP(prio)) ;
::m3     10:FAC1.SEIZE;
            11:Future.Wait(RP(Kvant, sys.N));
               12:FAC1.RELEASE;
         13: List.UnLink(m3,1,any,any) ;
        14: Loop(num_ret,m2) ;
        15: stor.Leave(ip(num_st)) ;
              16:LIST1. UNLINK(m8,1,any,any);
              17:tt.tabulate(TranAge);
             18: TERMINATE( 1) ;
::m4 19:LIST1. LINK(aclock);

 |R `Blocks    BLOCK Report
    Location     Total   Current
         1      1019         0
         2      1019         0
         3      1019         0
         4      1019         0
         5      2010         0
         6      1001         0
         7      1001         0
```

```
        8      2992         0
        9       983         1
       10      2991         0
       11      2991         0
       12      2991         0
       13      2991         0
       14      2991         0
       15      1000         0
       16      1000         0
       17      1000         0
       18      1000         0
       19      1009        18
|R `Values
|R `CurrentList Report Current Transactions List  Count= 2
   Tran   Assem     Pri Current    Next       TimeStamp       TimeEnter Inter
   1001    1001       0       9      10    149770.10602    149820.10602 FALSE
   1003    1003       0      19       5    148016.84370    149920.10602 FALSE
|R `IPList Report ( Integer ) Count= 57
       Tran             Num/Name        Integer
        970                  1            402
        970                  2              1
        970                  3              5
        980                  1            397
        980                  2              4
        980                  3              4
        988                  1            397
        988                  2              4
        988                  3              5
        995                  1            433
        995                  2              3
        995                  3              2
       1001                  1            420
       1001                  2              4
       1001                  3              5
       1003                  1            222
       1003                  2              2
       1003                  3              3
       1004                  1            411
       1004                  2              4
       1004                  3              3
       1005                  1            374
       1005                  2              2
       1005                  3              2
       1006                  1            356
       1006                  2              4
       1006                  3              4
       1007                  1            380
       1007                  2              1
       1007                  3              3
       1009                  1            299
       1009                  2              1
       1009                  3              4
       1010                  1            309
       1010                  2              0
       1010                  3              3
       1011                  1            229
       1011                  2              1
       1011                  3              1
       1013                  1            421
       1013                  2              3
       1013                  3              1
       1014                  1            423
       1014                  2              1
       1014                  3              2
       1015                  1            318
       1015                  2              0
       1015                  3              3
       1017                  1            375
       1017                  2              0
       1017                  3              2
```

```
        1018                    1            412
        1018                    2              1
        1018                    3              3
        1019                    1            416
        1019                    2              1
        1019                    3              5
 |R Facility = ` fac1 Entries = 2991 Current = 0 Utility =  0.99753 AverTime = 50.00000
    Tran = 0 IntTran = 0 Active Time = 149920.10602 NextGo = 0
 |R Storage = ` Stor Entries = 320003 Current = 420 Average = 834.45241 AverTime = 390.93756
Utility =  0.81489
    Max = 1024 Min = 0 Tran = 1000 ACapac = 1024.00000 Capacity = 1024 Active Time =
149920.10602 NextGo = 0
 |R User/Future = ` list Entries = 983 Current = 0 AverUser =  0.31072 AverTime = 47.38870
    Max = 2 Min = 0 Active Time = 149920.10602 NextGo = 0
 |R User/Future = ` List1 Entries = 1009 Current = 17 AverUser = 17.39423 AverTime = 2584.48417
    Max = 35 Min = 0 Active Time = 149920.10602 NextGo = 0
 |R Report  User/Future List Count= 17
    Tran   Assem     Pri Current    Next       TimeStamp       SortValue   Group Inter
    1004    1004       0      19      20    148034.25493    148034.25493       0 FALSE
    1005    1005       0      19      20    148047.06253    148047.06253       0 FALSE
    1006    1006       0      19      20    148050.17380    148050.17380       0 FALSE
     970     970       0      19      20    142364.56170    148120.10602       0 FALSE
    1007    1007       0      19      20    148465.92109    148465.92109       0 FALSE
    1009    1009       0      19      20    148569.03467    148569.03467       0 FALSE
    1010    1010       0      19      20    148578.36492    148578.36492       0 FALSE
    1011    1011       0      19      20    148676.23065    148676.23065       0 FALSE
     995     995       0      19      20    146646.31824    148820.10602       0 FALSE
    1013    1013       0      19      20    149033.19118    149033.19118       0 FALSE
    1014    1014       0      19      20    149217.55455    149217.55455       0 FALSE
    1015    1015       0      19      20    149440.42464    149440.42464       0 FALSE
     980     980       0      19      20    144045.79477    149470.10602       0 FALSE
    1017    1017       0      19      20    149785.01969    149785.01969       0 FALSE
    1018    1018       0      19      20    149800.81494    149800.81494       0 FALSE
    1019    1019       0      19      20    149823.98763    149823.98763       0 FALSE
     988     988       0      19      20    145258.59623    149870.10602       0 FALSE
 |R User/Future = `Future Entries = 4011 Current = 1 AverUser =  1.15072 AverTime = 43.01091
    Max = 2 Min = 0 Active Time = 149920.10602 NextGo = 0
 |R Report  User/Future List Count= 1
    Tran   Assem     Pri Current    Next       TimeStamp       SortValue   Group Inter
    1020    1020       0       0       1    150090.02635    150090.02635       0 FALSE
            Num/Name        Integer  |R `Future      Report IP LIST  Count= 2
I                 1           1020
I                 2           1020
 |R  Table = `tt  Entries = 1000.00000 Mean = 395.03309 StdDev = 139.11746
          Range       Frequency        Integral
        0.00000         0.00000         0.00000
      200.00000       114.00000         0.11400
      400.00000       464.00000         0.57800
      600.00000       371.00000         0.94900
      800.00000        50.00000         0.99900
     1000.00000         1.00000         1.00000
     1200.00000         0.00000         1.00000
            Num/Name         Double  |R `Sys       Report RP LIST  Count= 1
                1       50.00000
```

29 Особенности списков будущих событий

Класс TFuture наследует все возможности класса Tuser, и имеет следующие дополнительные блоки и процедуры.

Уже упомянутые ранее процедуру

NameObj.NewWaitProc(Gen,Delay,PR,TranCount,LastTran<,Gr);

и блоки

*: NameObj.NewWait(Delay,PR,TranCount,LastTran<,Gr<, CanOut<, LimCount <, NumBl);

*: NameObj.Wait(Delay<,Gr););

А также блок

*: NameObj.Activate (NumBl, Num, Gr1, Log, Cont <,IPCount <,NewAClock <,NumGr <,NumVal <, NumCurr <, NumNext);

*: NameObj.Activate (NumBl, Num, Log, Cont <,IPCount <,NewAClock <,NumGr <,NumVal <, NumCurr <, NumNext);

Смысл этих параметров такой же, как у блока UnLink, а параметр NewAClock управляет системными часами и временем выхода заявок в список текущих событий. Если NewAClock>0 то это будет новое значение системных часов и временем выхода заявок. Если NewAClock<0 то новое значение системных часов и время выхода заявок будет равным параметру Val списка будущих событий. Если NewAClock=0 то время выхода заявок будет равным текущему времени системных часов. В удаленной из списка заявке, в отличие от блока UnLink, в RP(NumVal) может записываться значение выражения Val - Aclock, то есть, оставшееся время задержки заявки.

Применение блоков Activate, совместно с функциями из TUser, как раз и позволяет управлять заявками из списка будущих событий.

Процедура

UnLinkProc(NewAClock <, NumBl<, <,NumGr<,NumVal<, NumCurr<, NumNext);

Процедура извлекает все заявки, для которых Val<= NewAClock, в остальном она аналогична предыдущей процедуре.

Рассмотрим следующую задачу. Пусть к врачу, который ведет запись на прием на время в будущем, отстоящее не менее, чем на 1800 минут (30 часов рабочего времени) от текущего момента, записываются на прием больные, поступающие на запись со средним интервалом 20.5 минут. Они обслуживаются в свое время за 19+-7 минут.

Дополним задачу следующими событиями.

Со средним интервалом 100 минут у одного из больных возникают экстренные ситуации, и он отказывается от визита.

Кроме того, со средним интервалом 72000 минут (1200 часов), у врача возникают экстренные ситуации, и он сдвигает график приема всех больных на 1800 минут (30 часов).

Такую задачу трудно решить средствами обычного GPSS. А средствами GPSS-Future, в котором список будущих событий – наследник списка пользователя – задача решается легко.

Пример решения этой задачи.

```
{~vb} Var
  Future: TFuture;
  VizitTime: Integer;
  List:AInteger;
  NumTran:integer;
  Mean:Double=72000;
{=QDoctor} QDoctor : TQueue;
{=Doctor} Doctor : TFacility;
{=Tab} Tab : TTable;
{=QWait} QWait : TQueue;
 {~ve}
Function Ex1:Boolean;
 begin
```

```
   Result :=Future.Tran=NumTran;
 end;
 {~pfe}
 {~ib} procedure Initial;begin
initrand(10);
  init(Future,'Future');
 setstart(100000);
 VizitTime:=0;
{=Gen} Future.NewWaitproc(Gen,exponential(21.5),0,Gen,Gen+1);
{=QDoctor} Init(QDoctor,'QDoctor');
{=Doctor} Init(Doctor,'Doctor');
{=Gen0} Future.NewWaitproc(Gen0,exponential(100),0,Gen0,Gen0+1);
{=Tab} Init(Tab,'Tab',0,400);
{=Gen1} Future.NewWaitProc(Gen1,exponential(Mean),0,Gen1,Gen1+1);
{=QWait} Init(QWait,'QWait');
 {~ie} end;
 {~mtb}procedure ModelTxt;begin case ActiveBlock of
{=Gen} ::Gen *:Future.NewWait(exponential(21.5),0,Gen,Gen+1);
                 *: Let(VizitTime,IByBool([VizitTime<1800+Trunc(Aclock/20+1)*20,true],
                  [ 1800+Trunc(Aclock/20+1)*20,VizitTime+20]));
 // VizitTime - время визита больного.
                 *:QWait .Queue;
::AA             *:Future.Wait(VizitTime-Aclock,AA);
  // задержка до времени визита больного.
                 *:QWait .Depart;
::AA1            *:QDoctor.Queue;//  Приём больного
                 *:Doctor .Seize;
                 *:QDoctor.Depart;
                 *:Future.Wait( MeanDev(19,7)); // время обслуживания.
                 *:Doctor . Release;
                 *:Topoint(2,Aclock,Doctor.a);
                 *:Tab.Tabulate(TranAge);
                 *:Terminate(1);
{=Gen0} ::Gen0 *:Future.NewWait(exponential(100),0,Gen0,Gen0+1);
                 *:Future.Select(List,0,AA,any,any); // выборка всех ожидающих больных
                 *: test([length(List)>0,true],[!*,!*+2]);
                 *: let(NumTran,list[random(length(List))]); // выборка случайного больного
                 *:Future.UnLink(Exit, 1, Ex1, Any,-1,0,2,3); // удаление выбранного больного
                 *:Terminate;
::Exit            *:QWait .Depart;
                  *:Terminate;
{=Gen1} ::Gen1 *:Future.NewWait(exponential(Mean),0,Gen1,Gen1+1);
                 *:Future.Activate(Exit1, 0 ,AA,any,any,-1,0,2,3);
                 *:Add(VizitTime,1800);
                 *:Topoint(1,Aclock,Tab.a);
                 *:Future.Wait(100);
```

```
                *:Terminate;
::Exit1         *:QWait .Depart;
                *:QWait .Queue;
                *:Future.Wait(RP(3)+1800,AA);
                *:QWait .Depart;
                *:Transfer(AA1);
 {~mte} end;end;
 {~mb} procedure Simulation; begin {`Simulation |R }
  start(GetTerm);
  Show(tab,1);
 {~me}end;
 {~rb} procedure Report;begin {`Report |R }
reportval(VizitTime,'VizitTime=');
{~re} end;
 {~cab} procedure CloseAllObj;begin {`CloseAllObj |R }
 {~cae} end; {`End |R }
```

Результитрующая статистика подтверждает предполагаемые свойства модели.

```
    Date Time Stamp= 30.12.2008 22:05:00   Ctrl Click |Name - HyperLink `R
E:\work\ObjectClient\Object No Sys\Future\GPSS-Future\book\Doctor5.rtf
StartTime         0.00000 EndTime    2748375.77745
 |Model |Blocks |Values |CurrentList |SynchronizeList |InterruptList
 |RPList |IPList |SPList |BPList
   Num      NameObject
     2        |Future
     3       |QDoctor
     4        |Doctor
     5           |Tab
     6         |QWait
 |R `Model
                // собственно модель
{=Gen} ::Gen 1:Future.NewWait(exponential(21.5),0,Gen,Gen+1);
                     2: Let(VizitTime,IByBool([VizitTime<1800+Trunc(Aclock/20+1)*20,true],
                       [ 1800+Trunc(Aclock/20+1)*20,VizitTime+20]));  // VizitTime - время
визита больного.
                     3:QWait .Queue;
::AA               4:Future.Wait(VizitTime-Aclock,AA);  // задержка до времени визита больного.
                     5:QWait .Depart;
::AA1              6:QDoctor.Queue;//  Приём больного
                     7:Doctor .Seize;
                     8:QDoctor.Depart;
                     9:Future.Wait( MeanDev(19,7)); // время обслуживания.
                     10:Doctor . Release;
                     11:Topoint(2,Aclock,Doctor.a);
                     12:Tab.Tabulate(TranAge);
                     13:Terminate(1);
{=Gen0} ::Gen0 14:Future.NewWait(exponential(100),0,Gen0,Gen0+1);
                     15:Future.Select(List,0,AA,any,any); // выборка всех ожидающих больных
                     16: test([length(List)>0,true],[17,17+2]);
                     17: let(NumTran,list[random(length(List))]); // выборка случайного
больного
                     18:Future.UnLink(Exit, 1, Ex1, Any,-1,0,2,3); // удаление выбранного
больного
                     19:Terminate;
::Exit               20:QWait .Depart;
                        21:Terminate;
{=Gen1} ::Gen1 22:Future.NewWait(exponential(Mean),0,Gen1,Gen1+1);
                     23:Future.Activate(Exit1, 0 ,AA,any,any,-1,0,2,3);
                     24:Add(VizitTime,1800);
                     25:Topoint(1,Aclock,Tab.a);
                     26:Terminate;
::Exit1            27:QWait .Depart;
                     28:QWait .Queue;
```

```
29:Future.Wait(RP(3)+1800,AA);
30:QWait .Depart;
31:Transfer(AA1);

 |R `Blocks    BLOCK Report
    Location      Total   Current
         1    127836         0
         2    127836         0
         3    127836         0
         4    127836        82
         5     95156         0
         6    100000         0
         7    100000         0
         8    100000         0
         9    100000         0
        10    100000         0
        11    100000         0
        12    100000         0
        13    100000         0
        14     27754         0
        15     27754         0
        16     27754         0
        17     27754         0
        18     27754         0
        19     27754         0
        20     27754         0
        21     27754         0
        22        50         0
        23        50         0
        24        50         0
        25        50         0
        26        50         0
        27      6522         0
        28      6522         0
        29      6522         0
        30      4844         0
        31      4844         0
 |R `Values
VizitTime=   2750280
 |R Facility = `Doctor Entries = 100000 Current = 0 Utility =  0.69159 AverTime = 19.00739
    Tran = 0 IntTran = 0 Active Time = 2748375.77745 NextGo = 0
 |R Queue = `QDoctor Entries = 100000 Current = 0 AverQueue =  0.07610 AverTime =  2.09148
    AverTime(-Ze) = 674.67252 Max = 2 Min = 0 Zero = 99690 Active Time = 2748375.77745 NextGo =
0
 |R Queue = `QWait Entries = 134358 Current = 82 AverQueue = 124.78576 AverTime = 2552.56977
    AverTime(-Ze) = 2554.60420 Max = 361 Min = 0 Zero = 107 Active Time = 2748375.77745 NextGo =
0
 |R User/Future = `Future Entries = 390001 Current = 85 AverUser = 127.87870 AverTime =
901.17387
    Max = 364 Min = 0 Active Time = 2748375.77745 NextGo = 0
 |R Report  User/Future List Count= 85
    Tran   Assem     Pri Current    Next       TimeStamp       SortValue   Group Inter
  155642  155642       0       0       1   2748389.86720   2748389.86720       0 FALSE
  155643  155643       0       0      14   2748487.78795   2748487.78795       0 FALSE
  149010  149010       0       0      22   2800421.78914   2800421.78914       0 FALSE
  155531  155531       0       4       5   2746586.66638   2748420.00000       4 FALSE
  155533  155533       0       4       5   2746593.13457   2748460.00000       4 FALSE
  155534  155534       0       4       5   2746602.61737   2748480.00000       4 FALSE
  155535  155535       0       4       5   2746612.23208   2748500.00000       4 FALSE
  155537  155537       0       4       5   2746616.53162   2748520.00000       4 FALSE
  155538  155538       0       4       5   2746631.69396   2748540.00000       4 FALSE
  155540  155540       0       4       5   2746660.73980   2748580.00000       4 FALSE
  155542  155542       0       4       5   2746685.71211   2748620.00000       4 FALSE
  155544  155544       0       4       5   2746704.62469   2748640.00000       4 FALSE
  155545  155545       0       4       5   2746747.99526   2748660.00000       4 FALSE
  155546  155546       0       4       5   2746798.92781   2748680.00000       4 FALSE
  155547  155547       0       4       5   2746805.52065   2748700.00000       4 FALSE
  155548  155548       0       4       5   2746839.84173   2748720.00000       4 FALSE
  155549  155549       0       4       5   2746854.84335   2748740.00000       4 FALSE
  155550  155550       0       4       5   2746878.40193   2748760.00000       4 FALSE
```

155554	155554	0	4	5	2746910.82758	2748820.00000	4	FALSE
155555	155555	0	4	5	2746922.98118	2748840.00000	4	FALSE
155556	155556	0	4	5	2746957.89340	2748860.00000	4	FALSE
155557	155557	0	4	5	2746978.57774	2748880.00000	4	FALSE
155558	155558	0	4	5	2746996.40940	2748900.00000	4	FALSE
155559	155559	0	4	5	2747032.01217	2748920.00000	4	FALSE
155561	155561	0	4	5	2747044.83246	2748940.00000	4	FALSE
155562	155562	0	4	5	2747072.92199	2748960.00000	4	FALSE
155563	155563	0	4	5	2747075.17449	2748980.00000	4	FALSE
155566	155566	0	4	5	2747188.92526	2749020.00000	4	FALSE
155567	155567	0	4	5	2747206.38235	2749040.00000	4	FALSE
155569	155569	0	4	5	2747256.29624	2749080.00000	4	FALSE
155570	155570	0	4	5	2747271.84348	2749100.00000	4	FALSE
155572	155572	0	4	5	2747299.36095	2749120.00000	4	FALSE
155573	155573	0	4	5	2747336.64829	2749140.00000	4	FALSE
155574	155574	0	4	5	2747340.90375	2749160.00000	4	FALSE
155575	155575	0	4	5	2747364.14208	2749180.00000	4	FALSE
155576	155576	0	4	5	2747384.03030	2749200.00000	4	FALSE
155579	155579	0	4	5	2747445.64605	2749260.00000	4	FALSE
155580	155580	0	4	5	2747448.39726	2749280.00000	4	FALSE
155581	155581	0	4	5	2747448.44046	2749300.00000	4	FALSE
155582	155582	0	4	5	2747460.87613	2749320.00000	4	FALSE
155584	155584	0	4	5	2747488.18792	2749360.00000	4	FALSE
155585	155585	0	4	5	2747521.07375	2749380.00000	4	FALSE
155586	155586	0	4	5	2747531.21872	2749400.00000	4	FALSE
155587	155587	0	4	5	2747563.83853	2749420.00000	4	FALSE
155588	155588	0	4	5	2747570.58319	2749440.00000	4	FALSE
155589	155589	0	4	5	2747586.86446	2749460.00000	4	FALSE
155590	155590	0	4	5	2747649.43694	2749480.00000	4	FALSE
155592	155592	0	4	5	2747682.91658	2749520.00000	4	FALSE
155594	155594	0	4	5	2747713.09435	2749540.00000	4	FALSE
155596	155596	0	4	5	2747731.58623	2749560.00000	4	FALSE
155598	155598	0	4	5	2747745.30060	2749580.00000	4	FALSE
155599	155599	0	4	5	2747765.69707	2749600.00000	4	FALSE
155601	155601	0	4	5	2747766.54105	2749620.00000	4	FALSE
155602	155602	0	4	5	2747768.90539	2749640.00000	4	FALSE
155603	155603	0	4	5	2747774.45725	2749660.00000	4	FALSE
155604	155604	0	4	5	2747781.36367	2749680.00000	4	FALSE
155605	155605	0	4	5	2747805.83707	2749700.00000	4	FALSE
155606	155606	0	4	5	2747817.49984	2749720.00000	4	FALSE
155608	155608	0	4	5	2747821.25648	2749740.00000	4	FALSE
155609	155609	0	4	5	2747826.94628	2749760.00000	4	FALSE
155610	155610	0	4	5	2747838.72122	2749780.00000	4	FALSE
155611	155611	0	4	5	2747855.20865	2749800.00000	4	FALSE
155612	155612	0	4	5	2747857.53259	2749820.00000	4	FALSE
155613	155613	0	4	5	2747860.17958	2749840.00000	4	FALSE
155614	155614	0	4	5	2747936.76074	2749860.00000	4	FALSE
155616	155616	0	4	5	2747944.00722	2749880.00000	4	FALSE
155617	155617	0	4	5	2747991.34836	2749900.00000	4	FALSE
155619	155619	0	4	5	2748004.25657	2749920.00000	4	FALSE
155621	155621	0	4	5	2748029.58446	2749940.00000	4	FALSE
155622	155622	0	4	5	2748090.50389	2749960.00000	4	FALSE
155623	155623	0	4	5	2748162.99599	2749980.00000	4	FALSE
155624	155624	0	4	5	2748170.97658	2750000.00000	4	FALSE
155625	155625	0	4	5	2748178.06172	2750020.00000	4	FALSE
155628	155628	0	4	5	2748206.29345	2750060.00000	4	FALSE
155629	155629	0	4	5	2748245.05308	2750080.00000	4	FALSE
155631	155631	0	4	5	2748246.86350	2750100.00000	4	FALSE
155632	155632	0	4	5	2748252.07357	2750120.00000	4	FALSE
155633	155633	0	4	5	2748266.82495	2750140.00000	4	FALSE
155634	155634	0	4	5	2748273.33590	2750160.00000	4	FALSE
155635	155635	0	4	5	2748277.96858	2750180.00000	4	FALSE
155636	155636	0	4	5	2748306.12230	2750200.00000	4	FALSE
155637	155637	0	4	5	2748307.61688	2750220.00000	4	FALSE
155638	155638	0	4	5	2748312.96675	2750240.00000	4	FALSE
155640	155640	0	4	5	2748315.44366	2750260.00000	4	FALSE
155641	155641	0	4	5	2748342.16385	2750280.00000	4	FALSE

```
            Num/Name         Integer  |R `Future      Report IP LIST  Count= 6
I                  1          127837
I                  2          155642
```

```
I                 14          27755
I                 15         155643
I                 22             51
I                 23         149010
 |R  Table = `Tab  Entries = 100000.00000 Mean = 3045.57350 StdDev = 1440.68349
          Range        Frequency         Integral
      1600.00000         0.00000          0.00000
      2000.00000     36836.00000          0.36836
      2400.00000     15045.00000          0.51881
      2800.00000      4860.00000          0.56741
      3200.00000      6641.00000          0.63382
      3600.00000      7078.00000          0.70460
      4000.00000      8066.00000          0.78526
      4400.00000      3296.00000          0.81822
      4800.00000      4094.00000          0.85916
      5200.00000      3286.00000          0.89202
      5600.00000      3370.00000          0.92572
      6000.00000      2591.00000          0.95163
      6400.00000      1427.00000          0.96590
      6800.00000      1184.00000          0.97774
      7200.00000       758.00000          0.98532
      7600.00000       541.00000          0.99073
      8000.00000       416.00000          0.99489
      8400.00000       417.00000          0.99906
      8800.00000        90.00000          0.99996
      9200.00000         4.00000          1.00000
      9600.00000         0.00000          1.00000
```

30 Блоки задания доступности и недоступности устройств

Для моделирования доступности и недоступности одноканальных и многоканальных устройств в самом простом случае, могут использоваться блоки

*:ObjName.SetGoto (NumBl);

Смысла этих блоков мы касались вскользь ранее. Положительное значение NumBl –определяет номер блока, отрицательное – запрещает вход в блок. По умолчанию – восстанавливается возможность входа в блок.

Это позволяет, задав в *:ObjName.SetGoto (NumBl) параметр равным –1, заблокировать вход в устройство, а вызвав его без параметра, вновь разрешить вход в устройство.

Пример использования блока.

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:

```
 {~vb} Var
{~f}Future:TFuture;
 {/fac0} fac0:Tfacility;
{/que0} que0:Tqueue;
 {~ve}
 {~pfe}
 {~ib} procedure Initial;begin
{~f}init(Future,'Future');
```

```
 setstart(20);
{/fac0}  init(fac0,' fac0');
{/que0}  init(que0,' que0');
{=Gen0} Future.NewWaitProc(Gen0,exponential(100),0,Gen0,Gen0+1);
{=Gen1} Future.NewWaitProc(Gen1,0,0,Gen1,Gen1+1);
  {~ie}end;
  {~mtb}procedure ModelTxt;begin case ActiveBlock of
{=Gen0} ::Gen0 *:Future.NewWait(exponential(100),0,Gen0,Gen0+1);
   *: que0.Queue;
   *: fac0.Seize;
   *: que0.Depart;
   *:Future.Wait(MeanDev(48,20));
   *: fac0.Release;
   *: Terminate;
{=Gen1} ::Gen1 *:Future.NewWait(0,0,Gen1,Gen1+1,0,true,1);
::ret      *: fac0.Setgoto;
         *:Future.Wait(normal(2000,200));
         *: fac0.Setgoto(-1);
         *:Future.Wait(normal(2000,200));
     *: split(1,ret);
    *: Terminate(1);
  {~mte} end;end;
  {~mb} procedure Simulation;  begin
start(Getterm);
  {~me}end;
  {~rb} procedure Report;begin
  {~re} end;
  {~cab} procedure CloseAllObj;begin
  {~cae} end;
```

Выходная статистика модели подтверждает корректность расчетов среднего времени обслуживания и занятости устройства.

```
  Date Time Stamp= 20.12.2008 14:00:46   Ctrl Click |Name - HyperLink `R
E:\work\ObjectClient\Object No Sys\Future\GPSS-Future\book\tmp42.rtf
StartTime         0.00000 EndTime      79285.14709
 |Model |Blocks |Values |CurrentList |SynchronizeList |InterruptList
 |RPList |IPList |SPList |BPList
   Num       NameObject
     2         |Future
     3          | fac0
     4          | que0
 |R `Model
 {=Gen0} ::Gen0 1:Future.NewWait(exponential(100),0,Gen0,Gen0+1);
   2: que0.Queue;
   3: fac0.Seize;
   4: que0.Depart;
   5:Future.Wait(MeanDev(48,20));
   6: fac0.Release;
   7: Terminate;
{=Gen1} ::Gen1 8:Future.NewWait(0,0,Gen1,Gen1+1,0,true,1);
::ret      9: fac0.Setgoto;
           10:Future.Wait(normal(2000,200));
           11: fac0.Setgoto(-1);
           12:Future.Wait(normal(2000,200));
      13: split(1,ret);
```

```
    14: Terminate(1);

|R `Blocks    BLOCK Report
   Location     Total   Current
        1       856         0
        2       856        57
        3       799         0
        4       799         0
        5       799         0
        6       799         0
        7       799         0
        8         1         0
        9        20         0
       10        20         0
       11        20         0
       12        20         0
       13        20         1
       14        20         0
|R `Values
|R `CurrentList Report Current Transactions List  Count= 58
   Tran   Assem     Pri Current    Next       TimeStamp       TimeEnter Inter
    819     819       0       2       3     73540.37601     73540.37601 FALSE
    820     820       0       2       3     73659.72921     73659.72921 FALSE
    821     821       0       2       3     73662.41639     73662.41639 FALSE
    822     822       0       2       3     73810.02117     73810.02117 FALSE
    823     823       0       2       3     73845.60756     73845.60756 FALSE
    824     824       0       2       3     73882.10899     73882.10899 FALSE
    825     825       0       2       3     73993.60007     73993.60007 FALSE
    826     826       0       2       3     74050.42644     74050.42644 FALSE
    827     827       0       2       3     74323.03261     74323.03261 FALSE
    828     828       0       2       3     74529.04027     74529.04027 FALSE
    829     829       0       2       3     74564.02515     74564.02515 FALSE
    830     830       0       2       3     74672.33843     74672.33843 FALSE
    831     831       0       2       3     74734.09067     74734.09067 FALSE
    832     832       0       2       3     74816.44722     74816.44722 FALSE
    833     833       0       2       3     74923.14288     74923.14288 FALSE
    834     834       0       2       3     74945.93133     74945.93133 FALSE
    835     835       0       2       3     75040.72165     75040.72165 FALSE
    836     836       0       2       3     75070.26911     75070.26911 FALSE
    837     837       0       2       3     75136.90182     75136.90182 FALSE
    838     838       0       2       3     75211.35730     75211.35730 FALSE
    839     839       0       2       3     75616.10466     75616.10466 FALSE
    841     841       0       2       3     75685.71836     75685.71836 FALSE
    842     842       0       2       3     75745.99848     75745.99848 FALSE
    843     843       0       2       3     75755.63831     75755.63831 FALSE
    844     844       0       2       3     75826.19183     75826.19183 FALSE
    845     845       0       2       3     75952.44006     75952.44006 FALSE
    846     846       0       2       3     76135.46955     76135.46955 FALSE
    847     847       0       2       3     76241.11638     76241.11638 FALSE
    848     848       0       2       3     76291.29777     76291.29777 FALSE
    849     849       0       2       3     76336.26081     76336.26081 FALSE
    850     850       0       2       3     76664.93407     76664.93407 FALSE
    851     851       0       2       3     76883.23150     76883.23150 FALSE
    852     852       0       2       3     76953.40384     76953.40384 FALSE
    853     853       0       2       3     76987.04986     76987.04986 FALSE
    854     854       0       2       3     77223.42655     77223.42655 FALSE
    855     855       0       2       3     77249.19981     77249.19981 FALSE
    856     856       0       2       3     77295.73702     77295.73702 FALSE
    857     857       0       2       3     77330.69271     77330.69271 FALSE
    858     858       0       2       3     77340.12687     77340.12687 FALSE
    859     859       0       2       3     77431.11501     77431.11501 FALSE
    860     860       0       2       3     77549.50097     77549.50097 FALSE
    861     861       0       2       3     77573.66278     77573.66278 FALSE
    862     862       0       2       3     77908.75057     77908.75057 FALSE
    863     863       0       2       3     78272.43387     78272.43387 FALSE
    864     864       0       2       3     78384.56027     78384.56027 FALSE
    865     865       0       2       3     78391.29307     78391.29307 FALSE
    866     866       0       2       3     78418.00827     78418.00827 FALSE
    867     867       0       2       3     78560.50559     78560.50559 FALSE
    868     868       0       2       3     78642.06265     78642.06265 FALSE
```

```
     869     869       0       2       3     78649.09179     78649.09179 FALSE
     870     870       0       2       3     78676.86104     78676.86104 FALSE
     871     871       0       2       3     78688.23491     78688.23491 FALSE
     872     872       0       2       3     78761.99649     78761.99649 FALSE
     873     873       0       2       3     78818.46762     78818.46762 FALSE
     874     874       0       2       3     79038.61151     79038.61151 FALSE
     875     875       0       2       3     79044.55574     79044.55574 FALSE
     876     876       0       2       3     79167.67545     79167.67545 FALSE
     878       2       0      13       9         0.00000     79285.14709 FALSE
 |R Facility = ` fac0 Entries = 799 Current = 0 Utility =  0.48737 AverTime = 48.36200
    Tran = 0 IntTran = 0 Active Time = 79285.14709 NextGo = -1
 |R Queue = ` que0 Entries = 856 Current = 57 AverQueue = 25.63558 AverTime = 2374.44018
    AverTime(-Ze) = 2425.44247 Max = 60 Min = 0 Zero = 18 Active Time = 79285.14709 NextGo = 0
 |R User/Future = `Future Entries = 1697 Current = 1 AverUser =  2.08603 AverTime = 97.46101
    Max = 3 Min = 0 Active Time = 79285.14709 NextGo = 0
 |R Report  User/Future List Count= 1
    Tran   Assem     Pri Current    Next       TimeStamp       SortValue   Group Inter
     877     877       0       0       1     79340.14365     79340.14365       0 FALSE
            Num/Name         Integer  |R `Future      Report IP LIST  Count= 4
I                 1             857
I                 2             877
I                 8               1
I                 9               2
```

Аналогичная задача для трехканального устройства при среднем интервале поступления заявок равном 33, решается следующим образом.

```
   {~vb} Var
 {~f}Future:TFuture;
{/que0} que0:Tqueue;
{/St} St:Tstorage;
  {~ve}
  {~pfe}
  {~ib} procedure Initial;begin
 {~f}init(Future,'Future');
 setstart(20);
{/que0}  init(que0,' que0');
{/St}  init(St,' St',3);
{=Gen} Future.NewWaitProc(Gen,Exponential(33.33),0,Gen,Gen+1);
{=Gen0} Future.NewWaitProc(Gen0,0,0,Gen0,Gen0+1);
  {~ie}end;
  {~mtb}procedure ModelTxt;begin case ActiveBlock of
  {=Gen} ::Gen *:Future.NewWait(Exponential(33.33),0,Gen,Gen+1);
   *: que0.Queue;
   *: St.Enter;
   *: que0.Depart;
    *:Future.Wait(MeanDev(48,20));
   *: St.Leave;
   *: Terminate;
{=Gen0} ::Gen0 *:Future.NewWait(0,0,Gen0,Gen0+1,0,true,1);
::ret      *: St.Setgoto;
    *:Future.Wait(normal(2000,200));
      *: St.Setgoto(-1);
    *:Future.Wait(normal(2000,200));
      *: split(1,ret);
```

```
   *: Terminate(1);
  {~mte} end;end;
  {~mb} procedure Simulation;  begin
start(Getterm);
  {~me}end;
  {~rb} procedure Report;begin
  {~re} end;
  {~cab} procedure CloseAllObj;begin
  {~cae} end;
```

Если обратиться к итоговой статистике, то можно убедиться, что модель работает правильно, то есть в модели корректно учитывается период недоступности. В этой статистике не приводится список текущих событий, как малоинформативный в данном случае.

```
 Date Time Stamp= 20.12.2008 14:06:53   Ctrl Click |Name - HyperLink `R
E:\work\ObjectClient\Object No Sys\Future\GPSS-Future\book\tmp43.rtf
StartTime         0.00000 EndTime      79283.74224
 |Model |Blocks |Values |CurrentList |SynchronizeList |InterruptList
 |RPList |IPList |SPList |BPList
   Num      NameObject
     2        |Future
     3         | que0
     4           | St
 |R `Model
   {=Gen} ::Gen 1:Future.NewWait(Exponential(33.33),0,Gen,Gen+1);
   2: que0.Queue;
   3: St.Enter;
   4: que0.Depart;
    5:Future.Wait(MeanDev(48,20));
   6: St.Leave;
   7: Terminate;
{=Gen0} ::Gen0 8:Future.NewWait(0,0,Gen0,Gen0+1,0,true,1);
::ret      9: St.Setgoto;
    10:Future.Wait(normal(2000,200));
      11: St.Setgoto(-1);
    12:Future.Wait(normal(2000,200));
      13: split(1,ret);
     14: Terminate(1);

 |R `Blocks    BLOCK Report
    Location     Total   Current
         1      2334         0
         2      2334        46
         3      2288         0
         4      2288         0
         5      2288         0
         6      2288         0
         7      2288         0
         8         1         0
         9        20         0
        10        20         0
        11        20         0
        12        20         0
        13        20         1
        14        20         0
 |R `Values
  |R Storage = ` St Entries = 2288 Current = 0 Average =  1.38527 AverTime = 48.00252 Utility =
0.46176
    Max = 3 Min = 0 Tran = 2308 ACapac =  3.00000 Capacity = 3 Active Time = 79283.74224 NextGo
= -1
 |R Queue = ` que0 Entries = 2334 Current = 46 AverQueue = 29.98167 AverTime = 1018.44866
    AverTime(-Ze) = 1108.70297 Max = 87 Min = 0 Zero = 190 Active Time = 79283.74224 NextGo = 0
 |R User/Future = `Future Entries = 4664 Current = 1 AverUser =  3.02779 AverTime = 51.46971
```

```
    Max = 5 Min = 0 Active Time = 79283.74224 NextGo = 0
 |R Report  User/Future List Count= 1
    Tran   Assem     Pri Current    Next       TimeStamp       SortValue   Group Inter
    2355    2355       0       0       1     79295.08016     79295.08016       0 FALSE
            Num/Name        Integer  |R `Future      Report IP LIST  Count= 4
I                  1           2335
I                  2           2355
I                  8              1
I                  9              2
```

Для многоканальных устройств такого способа управления доступностью и недоступностью вполне достаточно. Однако, для одноканальных устройств, понятие недоступности более многогранное.

Поэтому вернемся к блокам

*:ObjName.Extract(NumBl,NumRP); Блок удаляет обслуживаемую заявку из устройства. NumBl - номер или метка блока, куда удаляется заявка. RP(NumRP) - параметра запоминает оставшееся время задержки заявки. Устройство остается незанятым, и, если были прерванные заявки, то они его не получают. Поэтому после выполнения этого блока возможно занятие устройства по Seize или Preempt. Возможно также выполнение Return0, для передачи его прерванной ранее заявке. Если последний параметр опущен, то оставшееся время задержки заявки не запоминается.

*:ObjName.ExtractInterrupt(NumBl,NumRP); Блок удаляет прерванные заявки из устройства. NumBl - номер или метка блока, куда удаляются заявки. RP(NumRP) - определяет оставшееся время для продолжения обслуживания заявки. Если последний параметр опущен, то оставшееся время задержки заявки не запоминается.

*:ObjName.Preempt0; Блок прерывает обслуживание заявки на устройстве. Устройство остается незанятым. Используется для моделирования недоступности.

*:ObjName.Return0; Блок возвращает для обслуживания в устройстве, прерванную заявку. Это происходит после того, как устройство было оставлено незанятым. Используется для моделирования возврата из недоступности.

Рассмотрим моделирование различных способов задания недоступности одноканального устройства с помощью этих блоков.

Вариант А

Пусть заявка проходит через блоки

*:ObjName.Preempt0;

*:ObjName.SetGoto(-1);

Тогда устройство станет недоступным, в том числе и для той заявки, которая только что обслуживалась.

Когда мы решим, что устройство снова следует сделать доступным, то следует выполнить блоки

*:ObjName.Return0;

*:ObjName.SetGoto;

Вариант В

Пусть заявка проходит через блоки

*:ObjName.Extract(NumBl,NumRP);

*:ObjName.SetGoto(-1);

Тогда устройство станет недоступным, а заявка, которая только что обслуживалась, будет удалена из устройства.

Когда мы решим, что устройство снова следует сделать доступным, то следует выполнить блок

*:ObjName.SetGoto;

Вариант C

Пусть заявка проходит через блоки

*:ObjName.Extract(NumBl,NumRP);

*:ObjName.ExtractInterrupt(NumBl,NumRP);

*:ObjName.SetGoto(-1);

Тогда устройство станет недоступным, а заявка, которая только что обслуживалась, будет удалена из устройства. Из него будут также удалены заявки, которые были ранее прерваны на этом устройстве.

Когда мы решим, что устройство снова следует сделать доступным, то следует выполнить блок

*:ObjName.SetGoto;

Отметим, что организацию доступности – недоступности должна выполнять не та заявка, которая сама претендует на устройство.

Вероятно, это далеко не все способы организации недоступности для устройства, которые обеспечивают эти блоки. Эти блоки обеспечивают и более гибкое управление устройством, не связанное с прерыванием. Например, если заявка пройдет через такую серию блоков

Вариант D

*:ObjName.Extract(NumBl,NumRP);

*:ObjName.ExtractInterrupt(NumBl,NumRP);

*:ObjName.Preempt;

То она получит в свое распоряжение устройство, избавив его от обслуживавшейся заявки, и всех ранее прерванных. Естественно, после обслуживания заявка должна выполнить блок

*:ObjName.Return;

Очевидно, что возможен также

Вариант E

*:ObjName.Extract(NumBl,NumRP);

*:ObjName.ExtractInterrupt(NumBl,NumRP);

*:ObjName.Seizet;

Тогда заявка получит в свое распоряжение устройство, избавив его от обслуживавшейся заявки, и всех ранее прерванных. Естественно, после обслуживания заявка должна выполнить блок

*:ObjName.Release;

Он очень похож на вариант D, но устройство при этом будет занято обычным образом.

Из этих вариантов, только вариант A позволяет заявке продолжить обслуживание после периода недоступности. Все остальные варианты, тем или иным способом удаляют заявки из устройства. Поэтому, рассмотрим пример моделирования доступности – недоступности в этом варианте.

Пусть заявки поступают в устройство со средним интервалом 100, распределенным по экспоненциальному закону.

Пусть обслуживание заявки занимает в среднем 80+-70 единиц времени.

Каждая следующая заявка прерывает обслуживание предыдущей заявки.

Пусть также со средним интервалом 2000+-1500 наступает период недоступности, длящийся в среднем 200 единиц времени, распределенный по экспоненциальному закону.

Промоделировать 100 периодов недоступности. Протабулировать время обслуживания заявок в устройстве.

Пример такой модели мог бы выглядеть следующим образом.

```
  {~vb} Var
 {~f}Future:TFuture;
{/Q} Q:Tqueue;
{/Fa} Fa:Tfacility;
{/Tb} Tb:Ttable;
  {~ve}
  {~pfe}
  {~ib} procedure Initial;begin
 {~f}init(Future,'Future');
 setStart(100);
{/Q}  init(Q,'Q');
{/Fa}  init(Fa,' Fa');
{/Tb}  init(Tb,' Tb',0,100,100);
{=Gen} Future.NewWaitProc(Gen,Exponential(100),0,Gen,Gen+1);
{=Gen0} Future.NewWaitProc(Gen0,MeanDev(2000,1500),0,Gen0,Gen0+1);
  {~ie}end;
  {~mtb}procedure ModelTxt;begin case ActiveBlock of
{=Gen} ::Gen *:Future.NewWait(Exponential(100),0,Gen,Gen+1);
       *: q.queue ;
       *: fa.preempt ;
       *: q.depart ;
       *: mark;
       *:Future.Wait(MeanDev(80,70));
       *: tb.tabulate(TranAge) ;
       *: fa.return ;
       *: Terminate ;
{=Gen0} ::Gen0 *:Future.NewWait(MeanDev(2000,1500),0,Gen0,Gen0+1);
            *:fa.Preempt0;
            *:fa.SetGoto(-1);
             *:Future.Wait(Exponential(200));
            *:fa.Return0;
            *:fa.SetGoto;
          *: Terminate(1) ;
  {~mte} end;end;
  {~mb} procedure Simulation;  begin
   start(GetTerm);
   show(tb,1);
  {~me}end;
  {~rb} procedure Report;begin
```

```
{~re} end;
{~cab} procedure CloseAllObj;begin
{~cae} end;
```

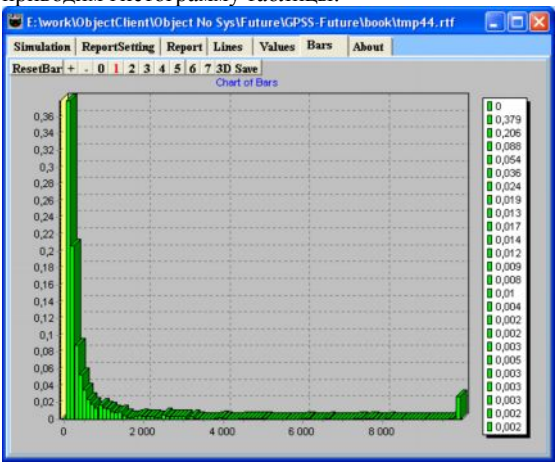
Результирующая статистика модели имеет вид.

```
Date Time Stamp= 20.12.2008 14:18:45   Ctrl Click |Name - HyperLink `R
E:\work\ObjectClient\Object No Sys\Future\GPSS-Future\book\tmp44.rtf
StartTime         0.00000 EndTime     207851.63924
 |Model |Blocks |Values |CurrentList |SynchronizeList |InterruptList
 |RPList |IPList |SPList |BPList
   Num      NameObject
     2        |Future
     3             |Q
     4           | Fa
     5           | Tb
 |R `Model
 {=Gen} ::Gen 1:Future.NewWait(Exponential(100),0,Gen,Gen+1);
         2: q.queue ;
         3: fa.preempt ;
         4: q.depart ;
         5: mark;
         6:Future.Wait(MeanDev(80,70));
         7: tb.tabulate(TranAge) ;
         8: fa.return ;
         9: Terminate ;
{=Gen0} ::Gen0 10:Future.NewWait(MeanDev(2000,1500),0,Gen0,Gen0+1);
11:fa.Preempt0;
12:fa.SetGoto(-1);
 13:Future.Wait(Exponential(200));
14:fa.Return0;
15:fa.SetGoto;
            16: Terminate(1) ;

 |R `Blocks    BLOCK Report
    Location     Total   Current
         1      2103         0
         2      2103         0
         3      2103         0
         4      2103         0
         5      2103         0
         6      2103         2
         7      2101         0
         8      2101         0
         9      2101         0
        10       100         0
        11       100         0
        12       100         0
        13       100         0
        14       100         0
        15       100         0
        16       100         0
 |R `Values
 |R `InterruptList Report Interrupt LIST  Count= 1
      Tran  NumFacil   IntTran
      2200         4         0
 |R Facility = ` Fa Entries = 2103 Current = 1 Utility =  0.80412 AverTime = 79.47596
    Tran = 2203 IntTran = 2200 Active Time = 207851.63924 NextGo = 0
 |R Queue = `Q Entries = 2103 Current = 0 AverQueue =  0.19384 AverTime = 19.15824
    AverTime(-Ze) = 265.06434 Max = 17 Min = 0 Zero = 1951 Active Time = 207851.63924 NextGo = 0
 |R User/Future = `Future Entries = 4408 Current = 4 AverUser = 11.54232 AverTime = 544.25820
    Max = 51 Min = 0 Active Time = 207851.63924 NextGo = 0
 |R Report  User/Future List Count= 4
    Tran   Assem     Pri Current    Next       TimeStamp       SortValue   Group Inter
    2203    2203       0       6       7    207628.64796    207885.93168       0 FALSE
    2204    2204       0       0       1    207934.12495    207934.12495       0 FALSE
    2205    2205       0       0      10    208595.48276    208595.48276       0 FALSE
    2200    2200       0       6       7    207442.33108      -101.51831       0  TRUE
            Num/Name        Integer  |R `Future      Report IP LIST  Count= 4
I                 1           2104
```

```
I                 2          2204
I                10           101
I                11          2205
```

В этой статистике не приведены результаты табулирования, вместо них мы приводим гистограмму таблицы.



31 Блоки работы с группами.

Группы позволяют иметь множества целых чисел, вещественных чисел, строк, а также множества заявок. Используя группы заявок, 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, а также обнулять время жизни заявок.

К блокам, обеспечивающим работу с группами вещественных чисел, относятся следующие блоки. Здесь ObjName- это имя группы.

*:ObjName.Join (ValueList); этот блок определяет вещественные числа, добавляемые к группе.

*:ObjName.Remove (Value,NumBl); этот блок определяет вещественное число, удаляемое из группы. Если нужного числа в группе нет, то заявка уходит по метке NumBl. Если метка опущена или равна 0, то заявка всегда переходит в следующий блок.

*:ObjName.RemoveAll ; этот блок удаляет все числа.

*:ObjName.Examine(Value,NumBl); этот блок проверяет наличие в группе заданного числа. Если его в группе нет, то заявка уходит по указанной метке.

*:ObjName.Copy(Values); Копирует данные из группы в динамический массив.

Следующие функции позволяют получить статистические данные по группе.

ObjName.A Определяет среднюю величину числовой группы.

ObjName.C Определяет число вхождений в группу.

ObjName.L Определяет текущую длину группы.

ObjName.M Определяет текущую максимальную длину группы.

Следующие процедуры управляют использованием группы.

ObjName.JoinProc (ValueList); определяет вещественные числа, добавляемые к группе.

ObjName.CopyProc(Values); Копирует данные из группы в динамический массив.

ObjName.RemoveAllProc; - удаляет все числа.

ObjName.ReportFile(FileName); Выводит отчет по группе. FileName – имя файла.

ObjName.GroupReportFile(FileName); Создает полный отчет по членам группы. FileName – имя файла.

Для работы с группами целых чисел, используются точно такие же блоки, процедуры и функции. Однако добавляемые, удаляемые, и проверяемые числа должны быть целыми, а тип группы TIGroup.

Для работы с группами строк, используются точно такие же блоки, процедуры и функции. Однако добавляемые, удаляемые, и проверяемые элементами должны быть строки, а тип группы TSGroup.

Для работы с группами заявок, используются большая номенклатура блоков, процедур и функций.

Блоки.

*:ObjName.Join (ATran) ; этот блок добавляет заданную заявку в группу, по умолчанию, добавляется активная заявка.

*:ObjName.Remove (Num,Del,NumIP); этот блок удаляет до Num заявок из группы. Если 0, то потенциально могут быть удалены все заявки. Del – отдельно описанная логическая функция без параметров, которая определяет заявки, подлежащие удалению. В IP(NumIP) запишется число реально удаленных заявок. Могут быть опущены, в том числе и все параметры. Тогда подразумевается удаление одной любой заявки. Причем тогда количество реально удаленных заявок никуда не записывается.

*:ObjName.Examine(NumBl,ATran); Блок проверяет наличие заданной заявки в группе. По умолчанию, проверяется наличие активной заявки. Если нужной заявки в группе нет, то активная заявка переходит на указанный блок.

*:ObjName.Scan (YesF,ValueF,NumRP,NumBl); Блок отыскивает заявки, удовлетворяющие отдельно описанной логической функции Yesf. Для первой попавшейся такой заявки, вычисляется значение вещественной функции ValueF. Оно записывается в RP(NumRP) - параметр активной заявки. Если нужной заявки не найдется, то заявка уходит по метке NumBl, если этот параметр есть.

*:ObjName.Scan (YesF,ValueF,NumIP,NumBl); Блок аналогичен предыдущему, но вычисляет значение целой функции. Запись происходит в IP - параметр.

*:ObjName.Scan (YesF,ValueF,NumSP,NumBl); Блок аналогичен предыдущему, но вычисляет значение строковой функции. Запись происходит в SP - параметр.

*:ObjName.Scan (YesF,ValueF,NumBP,NumBl); Блок аналогичен предыдущему, но вычисляет значение логической функции. Запись происходит в BP - параметр.

*:ObjName.Alter (YesF,ValueF,NumRP,Num,NumIPCount); Блок - вычисляет логическую функцию Yesf для определения подходящих заявок из группы. Для первых попавшихся подходящих заявок вычисляется значение вещественной функции ValueF. Оно записывается в RP(NumRP) для подходящей заявки из группы. При NumRP=-1 – выполняется изменение времени рождения на текущий момент времени. Всего модифицируется не более Num заявок из группы, или все. Реальное число модифицированных заявок записывается в IP(NumIPCount) активной заявки. Два

последних параметра могут быть опущены. Тогда модифицируются все заявки (значение Num=0).

*:ObjName.Alter (YesF,ValueF,NumIP,Num,NumIPCount); Блок аналогичен предыдущему, но вычисляется значение целой функции. Оно записывается в IP(NumIP) для подходящей заявки из группы. При NumIP=-1 Value рассматривается как новый приоритет заявки из группы.

*:ObjName.Alter (YesF,ValueF,NumSP,Num,NumIPCount); Блок аналогичен предыдущему, но вычисляется значение строковой функции. Оно записывается в SP(NumSP) для подходящей заявки из группы.

*:ObjName.Alter (YesF,ValueF,NumBP,Num,NumIPCount); Блок аналогичен предыдущему, но вычисляется значение логической функции. Оно записывается в BP(NumBP) для подходящей заявки из группы.

*:ObjName.Select(Log ,ANums); Блок записывает в динамический массив номера заявок из группы, для которых условие - функция Log - истинна.

Функции.

ObjName.C Определяет число вхождений в группу.

ObjName.L Определяет текущую длину группы.

ObjName.A Определяет среднее число заявок в группе.

ObjName.V Определяет временной интеграл.

ObjName.T Определяет среднее время, проведенное заявкой в группе.

ObjName.M Определяет максимальное число заявок в группе.

ObjName.I Выдает минимальное число заявок в группе.

Эти функции обычно используются внутри отдельно описанных функций YesF, Value, Ssum и Log.

ObjName.Assem Определяет семейство заявки из группы.

ObjName.TranAge Определяет время жизни заявки из группы.

ObjName.RP(Num) Определяет значение RP(Num) для заявки из группы.

ObjName.Pri Определяет приоритет для заявки из группы.

ObjName.IP(Num) Определяет значение IP(Num) для заявки из группы.

ObjName.BP(Num)Определяет значение BP(Num) для заявки из группы.

ObjName.SP(Num) Определяет значение SP(Num) для заявки из группы.

ObjName.Tran Определяет номер заявки для заявки из группы.

ObjName.ScanNum Выдает порядковый номер заявки в списке в процессе просмотра с помощью процедур группы Scan и Alter. В начале просмотра значение функции равно 0.

Эти функции хотя и возможно указывать в YesF и Value, но рекомендуется вычислять заранее и записывать в RP – параметр активной заявки.

ObjName.Sum(Ssum,Log) Определяет значение суммы значений функции Ssum для заявок из группы. Учитываются только заявки, удовлетворяющие условию - функции Log. Последний параметр может быть опущен. Тогда учитываются все заявки.

ObjName.Avg(Ssum,Log) Аналогична предыдущей функции, но вычисляет среднее значение.

ObjName.Min(Ssum,Log) Аналогична предыдущей функции, но вычисляет минимальное значение.

ObjName.Max(Ssum,Log) Аналогична предыдущей функции, но вычисляет максимальное значение.

Процедуры управления моделью.

Delete(Num, del) - удаляет заданное количество заявок, удовлетворяющих условию - функции Del - результат ее это количество реально удаленных заявок из группы.

ObjName.SelectProc(Log ,ATran); записывает в динамический массив номера заявок из группы, для которых условие - функция Log - истинна.

Delete; удаляет все заявки из группы.

ObjName.ReportFile(FileName); Выводит отчет по группе. FileName – имя файла.

TTGroup.(ObjName); Создает группу заявок.

ObjName.GroupReportFile(FileName); Создает полный отчет по заявкам группы. FileName – имя файла.

Пример задачи, использующей указанные блоки.

Пусть в каждом из трех магазинов, имеется набор товаров, с кодами от 10 до 121. Первоначальная номенклатура товаров каждого из магазинов создается путем наложения 150 случайно сгенерированных кодов товаров.

Далее, с интервалами 3600+-1800 в каждый магазин добавляется товар со случайным кодом, если его там не было. И с таким же интервалом, заканчивается товар со случайным кодом, если он там был.

Покупатели приходят в магазины за покупками со средним интервалом 3, распределенным по экспоненциальному закону. Код покупки генерируется равновероятно. Нас интересуют только те покупатели, для которых справедлив следующий порядок обхода магазинов: вначале проверяется, есть ли нужный товар в первом магазине, и если есть, то совершается покупка. Иначе посещается второй магазин, и так далее. Хотя, вероятно, есть и другие покупатели. Поиск товара в каждом из магазинов занимает 300+-200 единиц времени. Промоделировать работу магазинов в течение 100000 единиц времени. Определить число покупок в каждом из магазинов.

Текст такой модели может выглядеть следующим образом:

```
 {~vb} Var
{~f}Future:TFuture;
mag2:Integer;
mag3:Integer;
no_bye:Integer;
mag1:Integer;
{/gr1} gr1:TIgroup;
{/gr2} gr2:TIgroup;
{/gr3} gr3:TIgroup;
 {~ve}
 {~pfe}
 {~ib} procedure Initial;begin
{~f}init(Future,'Future');
   setstart (1 );
 mag2:=0;
 mag3:=0;
 no_bye:=0;
```

```
 mag1:=0;
{/gr1}  init(gr1,' gr1');
{/gr2}  init(gr2,' gr2');
{/gr3}  init(gr3,' gr3');
{=Gen} Future.NewWaitProc(Gen,0,0,Gen,Gen+1);
{=Gen2} Future.NewWaitProc(Gen2,MeanDev(3600,1800),0,Gen2,Gen2+1);
{=Gen3} Future.NewWaitProc(Gen3,MeanDev(3600,1800),0,Gen3,Gen3+1);
{=Gen4} Future.NewWaitProc(Gen4,MeanDev(3600,1800),0,Gen4,Gen4+1);
{=Gen5} Future.NewWaitProc(Gen5,MeanDev(3600,1800),0,Gen5,Gen5+1);
{=Gen6} Future.NewWaitProc(Gen6,MeanDev(3600,1800),0,Gen6,Gen6+1);
{=Gen7} Future.NewWaitProc(Gen7,MeanDev(3600,1800),0,Gen7,Gen7+1);
{=Gen8} Future.NewWaitProc(Gen8,exponential(3),0,Gen8,Gen8+1);
{=Gen9} Future.NewWaitProc(Gen9,100000,0,Gen9,Gen9+1);
 {~ie}end;
 {~mtb}procedure ModelTxt;begin case ActiveBlock of
{=Gen} ::Gen *:Future.NewWait(0,0,Gen,Gen+1,0,true,150);
*:gr1.join([duniform(10,121)]);
*:gr2.join([duniform(10,121)]);
*:gr3.join([duniform(10,121)]);
*:terminate;
{=Gen2} ::Gen2 *:Future.NewWait(MeanDev(3600,1800),0,Gen2,Gen2+1);
*:gr1.join ([duniform(10,121)]);
*:terminate;
{=Gen3} ::Gen3 *:Future.NewWait(MeanDev(3600,1800),0,Gen3,Gen3+1);
*:gr2.join([duniform(10,121)]);
*:terminate;
{=Gen4} ::Gen4 *:Future.NewWait(MeanDev(3600,1800),0,Gen4,Gen4+1);
*:gr3.join([duniform(10,121)]);
*:terminate;
{=Gen5} ::Gen5 *:Future.NewWait(MeanDev(3600,1800),0,Gen5,Gen5+1);
*:gr1.remove(duniform(10,121));
*:terminate;
{=Gen6} ::Gen6 *:Future.NewWait(MeanDev(3600,1800),0,Gen6,Gen6+1);
*:gr2.remove(duniform(10,121));
*:terminate;
{=Gen7} ::Gen7 *:Future.NewWait(MeanDev(3600,1800),0,Gen7,Gen7+1);
*:gr3.remove(duniform(10,121));
*:terminate;
{=Gen8} ::Gen8 *:Future.NewWait(exponential(3),0,Gen8,Gen8+1);
*:IPlet(1,[duniform(10,121)]);
 *:Future.Wait(MeanDev( 300,200));
*:gr1.examine(Ip(1),next);
*:Let(mag1,mag1+1);
*:terminate;
::next   *:Future.Wait(MeanDev( 300,200));
*:gr2.examine(Ip(1) ,next1);
```

```
*:Let(mag2,mag2+1);
*:terminate;
::next1 *:Future.Wait(MeanDev( 300,200));
*:gr3.examine(Ip(1),next2);
*:Let(mag3,mag3+1);
*:terminate;
::next2 *:Let(no_bye,no_bye+1);
*:terminate;

{=Gen9} ::Gen9 *:Future.NewWait(100000,0,Gen9,Gen9+1);
*:terminate(1);
  {~mte} end;end;
  {~mb} procedure Simulation;  begin
 {/sh}     start(GetTerm);
  {~me}end;
  {~rb} procedure Report;begin
 reportval(mag1,'mag1=');
 reportval(mag2,'mag2=');
 reportval(mag3,'mag3=');
  reportval(no_bye,'no_bye=');
  {~re} end;
  {~cab} procedure CloseAllObj;begin
  {~cae} end;
```

Выходная статистика модели имеет вид:

```
Date Time Stamp= 20.12.2008 15:07:41   Ctrl Click |Name - HyperLink `R
E:\work\ObjectClient\Object No Sys\Future\GPSS-Future\book\tmp46.rtf
StartTime         0.00000 EndTime     100000.00000
 |Model |Blocks |Values |CurrentList |SynchronizeList |InterruptList
 |RPList |IPList |SPList |BPList
   Num      NameObject
     2        |Future
     3          | gr1
     4          | gr2
     5          | gr3
 |R `Model
 {=Gen} ::Gen 1:Future.NewWait(0,0,Gen,Gen+1,0,true,150);
2:gr1.join([duniform(10,121)]);
3:gr2.join([duniform(10,121)]);
4:gr3.join([duniform(10,121)]);
5:terminate;
{=Gen2} ::Gen2 6:Future.NewWait(MeanDev(3600,1800),0,Gen2,Gen2+1);
7:gr1.join ([duniform(10,121)]);
8:terminate;
{=Gen3} ::Gen3 9:Future.NewWait(MeanDev(3600,1800),0,Gen3,Gen3+1);
10:gr2.join([duniform(10,121)]);
11:terminate;
{=Gen4} ::Gen4 12:Future.NewWait(MeanDev(3600,1800),0,Gen4,Gen4+1);
13:gr3.join([duniform(10,121)]);
14:terminate;
{=Gen5} ::Gen5 15:Future.NewWait(MeanDev(3600,1800),0,Gen5,Gen5+1);
16:gr1.remove(duniform(10,121));
17:terminate;
{=Gen6} ::Gen6 18:Future.NewWait(MeanDev(3600,1800),0,Gen6,Gen6+1);
19:gr2.remove(duniform(10,121));
20:terminate;
{=Gen7} ::Gen7 21:Future.NewWait(MeanDev(3600,1800),0,Gen7,Gen7+1);
22:gr3.remove(duniform(10,121));
```

```
23:terminate;
{=Gen8} ::Gen8 24:Future.NewWait(exponential(3),0,Gen8,Gen8+1);
25:IPlet(1,[duniform(10,121)]);
 26:Future.Wait(MeanDev( 300,200));
27:gr1.examine(Ip(1),next);
28:Let(mag1,mag1+1);
29:terminate;
::next   30:Future.Wait(MeanDev( 300,200));
31:gr2.examine(Ip(1) ,next1);
32:Let(mag2,mag2+1);
33:terminate;
::next1 34:Future.Wait(MeanDev( 300,200));
35:gr3.examine(Ip(1),next2);
36:Let(mag3,mag3+1);
37:terminate;
::next2 38:Let(no_bye,no_bye+1);
39:terminate;

{=Gen9} ::Gen9 40:Future.NewWait(100000,0,Gen9,Gen9+1);
41:terminate(1);

 |R `Blocks    BLOCK Report
    Location      Total   Current
         1        150         0
         2        150         0
         3        150         0
         4        150         0
         5        150         0
         6         29         0
         7         29         0
         8         29         0
         9         27         0
        10         27         0
        11         27         0
        12         25         0
        13         25         0
        14         25         0
        15         27         0
        16         27         0
        17         27         0
        18         25         0
        19         25         0
        20         25         0
        21         27         0
        22         27         0
        23         27         0
        24      33550         0
        25      33550         0
        26      33550       119
        27      33431         0
        28      23029         0
        29      23029         0
        30      10402        38
        31      10364         0
        32       6978         0
        33       6978         0
        34       3386        21
        35       3365         0
        36       2022         0
        37       2022         0
        38       1343         0
        39       1343         0
        40          1         0
        41          1         0
 |R `Values
mag1=     23029
mag2=      6978
mag3=      2022
no_bye=      1343
 |R `IPList Report ( Integer ) Count= 178
```

```
        Tran            Num/Name        Integer
       33460                   1            105
       33513                   1             84
       33550                   1             21
       33578                   1             48
       33582                   1             97
|R User/Future = `Future Entries = 81207 Current = 186 AverUser = 148.78999 AverTime = 183.22311
    Max = 197 Min = 0 Active Time = 100000.00000 NextGo = 0
 |R Report  User/Future List Count= 186
    Tran   Assem     Pri Current    Next       TimeStamp       SortValue   Group Inter
   33868   33868       0       0      24    100001.17775    100001.17775       0 FALSE
   33738   33738       0      26      27     99646.73800    100002.36399       0 FALSE
   33648   33648       0      30      31     99394.24929    100004.02044       0 FALSE
   33681   33681       0      30      31     99481.68147    100004.83576       0 FALSE
   33686   33686       0      30      31     99488.43939    100009.02279       0 FALSE
   33786   33786       0      26      27     99796.07546    100012.04627       0 FALSE
   33784   33784       0      26      27     99790.26785    100013.33300       0 FALSE
   33804   33804       0      26      27     99846.86877    100013.55171       0 FALSE
   33772   33772       0      26      27     99747.37564    100013.56538       0 FALSE
   33811   33811       0      26      27     99868.38683    100013.74775       0 FALSE
   33765   33765       0      26      27     99730.84337    100019.02302       0 FALSE
   33824   33824       0      26      27     99900.46576    100019.65044       0 FALSE
   33767   33767       0      30      31     99735.31368    100019.87692       0 FALSE
   33661   33661       0      30      31     99427.05472    100023.65259       0 FALSE
   33631   33631       0      30      31     99317.97315    100025.38919       0 FALSE
   33729   33729       0      26      27     99621.34468    100029.77646       0 FALSE
            Num/Name        Integer  |R `Future      Report IP LIST  Count= 18
I                  1            150
I                  2            158
I                  6             30
I                  7          33693
I                  9             28
I                 10          32793
I                 12             26
I                 13          32332
I                 15             28
I                 16          33383
I                 18             26
I                 19          33511
I                 21             28
I                 22          33407
I                 24          33551
I                 25          33868
I                 40              2
I                 41          33869
 |R GroupInt = ` gr1 Entries = 179 Current = 72 Average = 77.13946 AverTime = 43094.66844
    Max = 84 Min = 0 Active Time = 100000.00000 NextGo = 0
 gr1       Report GROUP LIST ( Integer ) num=72
             10
             11
             12
             13
|R GroupInt = ` gr2 Entries = 177 Current = 78 Average = 78.90349 AverTime = 44578.24489
    Max = 81 Min = 0 Active Time = 100000.00000 NextGo = 0
 gr2       Report GROUP LIST ( Integer ) num=78
             10
             11
             14
             15
             16
             17
             18
             19
             22
             23
             25
|R GroupInt = ` gr3 Entries = 175 Current = 72 Average = 78.10751 AverTime = 44632.86391
    Max = 81 Min = 0 Active Time = 100000.00000 NextGo = 0
 gr3       Report GROUP LIST ( Integer ) num=72
             10
```

```
11
12
16
17
19
20
21
22
```

Часть списков приводится не полностью.

Сугубо экспериментальная модель, просто подтверждающая работоспособность блоков, работающих с группами заявок.

Пусть заявки поступают на вход системы с интервалом 100+-90 и задерживаются на 300+-200 единиц времени. Заявки получают случайное значение RP(4), равномерно распределенное в интервале от 1 до 10. И случайный приоритет, равномерно распределенный в интервале от 3 до 7.

С вероятностью 0.4 заявки попадают в группу заявок, где они пребывают 1500+-200 единиц времени, а затем покидают систему. С вероятностью 0.6 заявки задерживаются на 400+-200 единиц времени, а затем покидают систему.

Заявки, манипулирующие с группой, появляются с интервалом 2000+-900 единиц времени, а затем ищут в группе заявки, у которых параметр RP(4)> 7 и в параметр RP(2) заносят значение RP(4)*PR/2, взятое из параметров заявок группы.

При успехе поиска заявка задерживается на 10+-5 единиц времени, а при неуспехе – сразу удаляется из системы. Выполнить моделирования до ухода 100 заявок манипулирующих с группой.

Текст этой модели имеет вид.

```
 {~vb} Var
 {~f}Future:TFuture;
{/grou} grou:TTgroup;
 {~ve}
{/logf}   function logf : boolean;   begin result:=grou.rp(4)> 7 ;       end;
{/rezf}   function rezf : double;   begin result:=grou.rp(4)*grou.pri/2  ;   end;
 {~pfe}
 {~ib} procedure Initial;begin
 {~f}init(Future,'Future');
setstart(100) ;
{/grou}  init(grou,' grou');
{=Gen} Future.NewWaitProc(Gen,MeanDev(100,90),0,Gen,Gen+1);
{=Gen0} Future.NewWaitProc(Gen0,MeanDev(2000,900),0,Gen0,Gen0+1);
 {~ie}end;
 {~mtb}procedure ModelTxt;begin case ActiveBlock of
{=Gen} ::Gen *:Future.NewWait(MeanDev(100,90),0,Gen,Gen+1);
    *:Future.Wait(MeanDev(300,200));
 *:rPlet(4,[duniform(1,10)]);
 *:priority(duniform(3,7));
  *:test([random<0.4,true],[!*,lb3]);
   *:grou.Join;
    *:Future.Wait(MeanDev(1500,200));
   *:terminate;
::lb3   *:Future.Wait(MeanDev(400,200));
```

```
      *:terminate;
{=Gen0} ::Gen0 *:Future.NewWait(MeanDev(2000,900),0,Gen0,Gen0+1);
*:grou.scan(logf,rezf,2,md1);
 *:split(1,!*+1);
    *:Future.Wait(MeanDev(10,5));
::md1        *:tostring(0,'TG1= '+INTTOSTR(Term));
*:terminate(1);
  {~mte} end;end;
  {~mb} procedure Simulation;  begin
 {/sh}     start(GetTerm);
  {~me}end;
  {~rb} procedure Report;begin
  {~re} end;
  {~cab} procedure CloseAllObj;begin
  {~cae} end;
```

Итоговая статистика подтверждает, что модель работает в соответствии с описанием.

```
Date Time Stamp= 20.12.2008 15:18:54   Ctrl Click |Name - HyperLink `R
E:\work\ObjectClient\Object No Sys\Future\GPSS-Future\book\tmp48.rtf
StartTime          0.00000 EndTime       98912.02892
 |Model |Blocks |Values |CurrentList |SynchronizeList |InterruptList
 |RPList |IPList |SPList |BPList
   Num       NameObject
     2         |Future
     3          | grou
 |R `Model
 {=Gen} ::Gen 1:Future.NewWait(MeanDev(100,90),0,Gen,Gen+1);
     2:Future.Wait(MeanDev(300,200));
 3:rPlet(4,[duniform(1,10)]);
 4:priority(duniform(3,7));
  5:test([random<0.4,true],[6,lb3]);
   6:grou.Join;
     7:Future.Wait(MeanDev(1500,200));
   8:terminate;
::lb3   9:Future.Wait(MeanDev(400,200));
         10:terminate;
{=Gen0} ::Gen0 11:Future.NewWait(MeanDev(2000,900),0,Gen0,Gen0+1);
12:grou.scan(logf,rezf,2,md1);
 13:split(1,14+1);
     14:Future.Wait(MeanDev(10,5));
::md1         15:tostring(0,'TG1= '+INTTOSTR(Term));
16:terminate(1);

 |R `Blocks    BLOCK Report
    Location     Total   Current
         1      1003         0
         2      1003         3
         3      1000         0
         4      1000         0
         5      1000         0
         6       397         0
         7       397         5
         8       392         0
         9       603         2
        10       601         0
        11        53         0
        12        53         0
        13        47         0
        14        47         0
        15       100         0
        16       100         0
```

```
 |R `Values
 |R `RPList  Report ( Double ) Count= 7
        Tran            Num/Name         Double
        1086                   4        2.00000
        1087                   4        8.00000
        1088                   4        5.00000
        1094                   4        5.00000
        1097                   4        1.00000
        1098                   4        5.00000
        1099                   4        7.00000
 |R User/Future = `Future Entries = 3108 Current = 12 AverUser = 12.92594 AverTime = 411.36766
    Max = 22 Min = 0 Active Time = 98912.02892 NextGo = 0
 |R Report  User/Future List Count= 12
    Tran   Assem     Pri Current    Next       TimeStamp       SortValue   Group Inter
    1103    1103       0       0       1     99019.79198     99019.79198       0 FALSE
    1100    1100       0       2       3     98641.01963     99080.46002       0 FALSE
    1086    1086       3       7       8     97289.80680     99156.05031       0 FALSE
    1101    1101       0       2       3     98730.60078     99174.06407       0 FALSE
    1088    1088       3       7       8     97483.97715     99228.09516       0 FALSE
    1098    1098       4       9      10     98442.61345     99294.80843       0 FALSE
    1102    1102       0       2       3     98836.00345     99326.08887       0 FALSE
    1099    1099       7       9      10     98629.64127     99368.49838       0 FALSE
    1087    1087       3       7       8     97377.19532     99414.47214       0 FALSE
    1094    1094       6       7       8     97933.43153     99500.25017       0 FALSE
    1097    1097       3       7       8     98345.99645    100181.21748       0 FALSE
    1104    1104       0       0      11    100997.52914    100997.52914       0 FALSE
            Num/Name        Integer  |R `Future      Report IP LIST  Count= 4
I                 1           1004
I                 2           1103
I                11             54
I                12           1104
 |R GroupTran = ` grou Entries = 397 Current = 5 Average =  6.03396 AverTime = 1503.35369
    Max = 13 Min = 0 Active Time = 98912.02892 NextGo = 0
 grou     Report TGROUP LIST ( TRANSACTIONS ) num=5
           1086
           1087
           1088
           1094
           1097
```

32 Редко используемые блоки.

*: MoveNext; блок без операции, иногда этот блок бывает полезным, чтобы точнее разобраться в статистике по результатам моделироввания.

*: SetTerm(Num); этот блок устанавливает новое значение счетчика завершений.

*: Displace(ATran,NumBl,NumRP,NumSP); этот блок отправляет заявку с номером ATran- в блок с меткой Numbl. NumRP - определяет номер RP - параметра, где сохраняется оставшееся время задержки. NumSP- номер SP -параметра, где записывается обозначение списка, где заявка обнаружена.

Здесь "N" - это отсутствие заявки. "P" – заявка обнаружена в списке прерываний.

"C"- заявка обнаружена в списке текущих событий. "F" – заявка обнаружена в списке будущих событий. "U" - заявка обнаружена в списке пользователя. "S" – заявка обнаружена в списке синхронизации. По умолчанию, соответствующая информация не записывается в P – параметры.

*:FindTran(ATran,NumSP,NxtIP,CurIP); этот блок разыскивает заявку с данным номером. Параметр NumSp – аналогичен такому же параметру блока Displace. Номер следующего блока и текущего блока, записываются в параметры IP(NxtIP), IP(CurIP). По умолчанию, эти номера не записываются.

*: RPDelete([NumList],ATran); Блок удаляет параметр RP с номерами из списка для данной заявки. По умолчанию - для активной заявки.

*: IPDelete([NumList],ATran); Блок удаляет параметр IP с номерами из списка для данной заявки. По умолчанию - для активной заявки.

*: SPDelete([NumList],ATran); Блок удаляет параметр SP с номерами из списка для данной заявки. По умолчанию - для активной заявки.

*: BPDelete([NumList],ATran); Блок удаляет параметр BP с номерами из списка для данной заявки. По умолчанию - для активной заявки.

*: RPDelete(ATran); Блок удаляет все RP -параметры данной заявки. По умолчанию - для активной заявки.

*: IPDelete(ATran); Блок удаляет все IP -параметры данной заявки. По умолчанию - для активной заявки.

*: SPDelete(ATran); Блок удаляет все SP -параметры данной заявки. По умолчанию - для активной заявки.

*: BPDelete(ATran); Блок удаляет все BP -параметры данной заявки. По умолчанию - для активной заявки.

*: IncTran; Блок увеличивает на 1 номер последней запланированной или выпущенной заявки. Он предназначен для создания фиктивных наборов P –параметров, для которых нет соответствующей заявки. Таким образом, можно в любой момент создать новый набор P –параметров.

33 Элементарные сведения о языке Pascal

Данный раздел не является руководством по языку Pascal, он скорее содержит некоторые сведения об этом языке, которые с большей вероятностью могут понадобиться разработчикам моделей.

Этот язык состоит из следующих операторов и разделов:

- ✓ Procedure - определяет заголовок процедуры.
- ✓ Function - определяет заголовок функции.
- ✓ Var - определяет переменные в модели в целом, процедуре или функции.
- ✓ Const - определяет константы в модели в целом.
- ✓ Все операторы, кроме заголовков процедур и функций могут иметь метку, отделяющуюся от оператора пробелами.
- ✓ Ключевые слова BEGIN и END- определяют начало и конец составного оператора в процедуре, то есть позволяют создавать составные операторы.
- ✓ Оператор присваивания.
- ✓ Вызов процедур или функций.
- ✓ Оператор If Then или
- ✓ If Then Else -используется для ветвлений в программе.
- ✓ Оператор Case -используется для ветвлений в программе.
- ✓ Оператор While Do- позволяет организовывать циклы с проверкой в начале .
- ✓ Оператор For Do- позволяет организовывать циклы с известным числом повторений.
- ✓ Оператор Repeat Until - позволяет организовывать циклы с проверкой в конце.

- ✓ Оператор GOTO – позволяет выполнить безусловный переход.

Оператор присваивания имеет вид:

A:=B;

Здесь A – глобальная или локальная переменная в модели, или такой же элемент массива. B - выражение.

Например

Home:=(RX(time)+RP(num))*3.6+home;

Примечание. В процедуре нельзя изменить значения X или P – параметров путем присваивания, но для этого есть процедуры типа RXSet, RPSet и так далее.

Текст заголовка процедуры имеет вид

Pprocedure имя(список аргументов с их типом через точку с запятой); Var описание временных переменных функции; составной оператор;

Составной оператор представляет собой последовательность операторов, разделенных точкой с запятой, и обрамленных ключевыми словами BEGIN END;

Процедуры и функции могут использоваться рекурсивно, то есть вызывать сами себя, как прямо, так и косвенно.

Заголовок функции имеет вид

Function имя(список аргументов с их типом через точку с запятой):тип функции; Var описание временных переменных функции; Составной оператор;

Как правило, используются переменные и функции следующих типов.

Integer, Double, String, Boolean, что означает целый, вещественный двойной точности строковый и логический тип данных. При описании переменных в заголовке процедуры или функции, каждая переменная описывается следующим образом.

Имя:Тип;

Если перед именем переменной стоит Var , то изменение соответствующей переменной в процедуре или функции изменяет значение переменной, указанной в вызове функции.

При разработке моделей, не рекомендуется использовать процедуры и функции с параметрами, а рекомендуется напрямую использовать параметры модели, которые все равно «видны» в процедуре или функции.

Переменная Result используется для присваивания значения результату функции.

Раздел Var описывает в процедуре временные величины, которые существуют только во время выполнения процедуры. Если Var относится к модели в целом, то есть, описан вне процедур и функций, то его переменные существуют все время работы программы. Раздел имеет вид:

Var - список описаний переменных через точку с запятой.

Описание массивов выглядит несколько иначе, имя: Array[список пар границ индексов через запятую] of Тип элемента массива. Каждая пара границ индексов определяет минимальное и максимальное значение соответствующего индекса. Они разделяются двумя точками, по образцу 2..15, однако, минимальное значение традиционно бывает только 0 или 1.

Примеры

Var Value1:Integer; tmp:Double;

Var MATRIX DataArray: array [5,7,15] of String;

Fac:array [1..20] of Tfacility;

Раздел Const описывает в модели имена, сопоставленные константам. Он состоит из пар

Имя=значение;

Тогда в модели можно использовать описанное имя вместо соответствующей константы. Чаще всего используются константы для номеров X и P – параметров, для повышения наглядности модели.

Оператор If имеет вид:

If логическое выражение Then оператор;

Если выражение равно True (истина), то встроенный оператор выполняется, иначе он не выполняется.

Форма оператора.

If логическое выражение Then оператор1
  Else оператор2;

В этом случае выполнение оператора происходит несколько по-другому. Если выражение равно True, то выполняется встроенный оператор1, иначе оператор2.

Оператор Case имеет вид:

```
case Par of
Const1:Oper1;
Const2: Oper2;
.
.
else ElseOper;
 end;
```

Здесь Par -это управляющее выражение оператора. Оно должно быть целым или символьным. Const1, Const2 – целые или символьные константы. Oper1, Oper2, ElseOper – простые или составные операторы. Если значение Par совпадает с одной из констант, то выполняется соответствующий оператор, а затем управление передается на оператор, расположенный после Case. Если значение Par не совпадает ни с одной из констант, то выполняется оператор ElseOper, а затем управление передается на оператор, расположенный после Case.

Часть else ElseOper – может отсутствовать. Количество пар Const:Oper – не ограничено.

Оператор GoTo имеет вид:

GoTo A;

Здесь A – это метка в процедуре.

Следующим будет выполняться оператор с этой меткой. Этот оператор категорически не рекомендуется.

Оператор While предназначен для организации циклов и имеет вид:

While логическое выражение DO оператор;

Оператор здесь, как правило, является составным. Логическое выражение вычисляется и если оно имеет значение True, то выполняется оператор, вставленный в цикл. Иначе 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 и функции.

Вызов процедуры и функции выглядит одинаково, однако вызовы процедур нельзя помещать в выражения. Они образуют самостоятельный оператор. Вызов имеет следующий вид.

Имя(список параметров через запятую)

Операторы For Do и Repeat Until – не являются необходимыми. Однако их тоже часто используют для построения циклов.

Оператор For Do имеет следующий вид.

For I:=начало to конец Do оператор;

Здесь I – это переменная цикла. Внутренний оператор цикла выполняется при всех целых значениях переменной цикла, от начала – до конца. Если оператор имеет вид

For I:=конец DownTo начало оператор;

То внутренний оператор цикла выполняется при всех целых значениях переменной цикла, от конца – до начала.

Оператор Repeat Until имеет следующий вид.

Repeat

список операторов через точку с запятой

Until - логическое выражение.

Операторы, указанные между Repeat и Until выполняются многократно, пока логическое выражение не окажется истинным. Тогда цикл завершается, и выполнение процедуры идет дальше.

Очень важно учитывать, что операторы могут вкладываться друг в друга.

В качестве примера рассмотрим процедуру вычисления факториалов, а также модель, для вычисления первых 15 значений факториала.

```
  {~vb} Var
 {~f}Future:TFuture;
{\Sys} Sys:Tparam;
  {~ve}
{/factorial}   function factorial(i:integer) : double;
begin
  Result:=1;
  while i>1 do
  begin
    result:=result*i;
    i:=i-1;
  end;
end;
  {~pfe}
  {~ib} procedure Initial;begin
  {~f}init(Future,'Future');
setstart(15);
{\Sys} Init(Sys,'Sys');
{=Gen} Future.NewWaitProc(Gen,1,0,Gen,Gen+1);
  {~ie}end;
  {~mtb}procedure ModelTxt;begin case ActiveBlock of
{=Gen} ::Gen *:Future.NewWait(1,0,Gen,Gen+1);
```

```
                *: IPlet(1,[iP(1, sys.N)+1] , sys.N) ;
                *: RPlet(iP(1, sys.N),[Factorial(iP(1, sys.N))] , sys.N) ;
                *: Terminate(1)  ;
        {~mte} end;end;
        {~mb} procedure Simulation;  begin
         start(GetTerm);
        {~me}end;
        {~rb} procedure Report;begin
        {~re} end;
        {~cab} procedure CloseAllObj;begin
        {~cae} end;
```

Здесь в ячейки с номерами 1..15 заносятся значения соответствующих факториалов. Важно, что формальные параметры процедуры, и переменные, описанные в Var, локализованы в ней. То есть такая процедура может изменять только значения глобальных переменных или X и P - параметров, а также возвращать результат вычислений через Result.

Выходная статистика этой модели имеет вид:

```
Date Time Stamp= 20.12.2008 21:27:36   Ctrl Click |Name - HyperLink `R
E:\work\ObjectClient\Object No Sys\Future\GPSS-Future\book\tmp55.rtf
StartTime         0.00000 EndTime         15.00000
 |Model |Blocks |Values |CurrentList |SynchronizeList |InterruptList
 |RPList |IPList |SPList |BPList
   Num      NameObject
     2        |Future
     3           |Sys
 |R `Model
 {=Gen} ::Gen 1:Future.NewWait(1,0,Gen,Gen+1);
                2: IPlet(1,[iP(1, sys.N)+1] , sys.N) ;
                3: RPlet(iP(1, sys.N),[Factorial(iP(1, sys.N))] , sys.N) ;
                4: Terminate(1)  ;

 |R `Blocks    BLOCK Report
    Location     Total   Current
        1         15        0
        2         15        0
        3         15        0
        4         15        0
 |R `Values
 |R User/Future = `Future Entries = 16 Current = 1 AverUser =  1.00000 AverTime =  0.93750
    Max = 0 Min = 0 Active Time = 15.00000 NextGo = 0
 |R Report  User/Future List Count= 1
    Tran   Assem     Pri Current    Next       TimeStamp       SortValue   Group Inter
      16      16       0       0       1        16.00000        16.00000       0 FALSE
            Num/Name         Integer  |R `Future      Report IP LIST  Count= 2
I                 1             16
I                 2             16
            Num/Name          Double  |R `Sys       Report RP LIST  Count= 15
R                 1         1.00000
R                 2         2.00000
R                 3         6.00000
R                 4        24.00000
R                 5       120.00000
R                 6       720.00000
R                 7      5040.00000
R                 8     40320.00000
R                 9    362880.00000
R                10   3628800.00000
R                11  39916800.00000
R                12  479001600.00000
R                13  6227020800.00000
R                14  87178291200.00000
R                15  1307674368000.00000
```

```
            Num/Name        Integer  |R `Sys      Report IP LIST  Count= 1
I                 1             15
```

Количество и содержание процедур определяется потребностями разработчика модели.

В принципе, потребность в разработке личных процедур и функций возникает только при профессиональном использовании системы.

34 Выборка требуемых объектов в GPSS - Future.

В GPSS - Future не предусмотрены блоки выборки требуемых объектов, однако их несложно создать самому. В принципе, в этой среде гораздо удобнее создавать не блоки выборки, а функции. Как это может выглядеть на практике, демонстрирует следующая задача.

Пусть в систему поступают заявки с интервалом 100+-80 минут. Для обслуживания заявок следует выбрать первое попавшееся устройство из 10 имеющихся, у которого активная очередь - минимальна. Примечание: занятие пустого устройства - предпочтительнее. Обслуживание в любом устройстве
занимает в среднем 800 минут и распределено по экспоненциальному закону.

Промоделировать обслуживание 1000 заявок.

Текст модели, решающей эту задачу с помощью пользовательской функции имеет вид.

```
Const
num=10;
  {~vb} Var
 {~f}Future:TFuture;
{/fac0} fac0:array[1..num] of Tfacility;
{/que0} que0:array[1..num] of Tqueue;
  {~ve}
procedure  facShow;
var i:integer;
begin
  for i:=1 to num do
  ToValueProc(i,fac0[i].A);
end;
procedure  QueShow;
var i:integer;
begin
  for i:=1 to num do
 ToValueProc(i,que0[i].a);
end;
{/fun0}   function fmin : integer;
var i,j:integer;
begin
  j:=1;
  for i:=1 to num do
  if (que0[i].L<que0[j].L) or (que0[i].L=que0[j].L)
  and (fac0[i].L=0)and (fac0[j].L=1)   then
```

```
 j:=i;
 result:=j;
end;
  {~pfe}
  {~ib} procedure Initial;  var  i:integer; begin
 {~f}init(Future,'Future');
   setstart(1000);
{/fac0}  for i:=1 to num do init(fac0[i] , 'fac0'+inttostr(i));
{/que0}  for i:=1 to num do init(que0[i] , ' que0'+inttostr(i));
customProc(['facShow','QueShow'],[facShow,QueShow]);
{=Gen} Future.NewWaitProc(Gen,MeanDev(100,80),0,Gen,Gen+1);
  {~ie}end;
  {~mtb}procedure ModelTxt;begin case ActiveBlock of
{=Gen} ::Gen *:Future.NewWait(MeanDev(100,80),0,Gen,Gen+1);
  *: IPlet(1,[fmin]);
  *: que0[IP(1)].Queue;
  *: fac0[IP(1)].Seize;
  *:Future.Wait(exponential(800));
  *: que0[IP(1)].Depart;
  *: fac0[IP(1)].Release;
  *: Terminate(1);
  {~mte} end;end;
  {~mb} procedure Simulation;  begin
   start(GetTerm);
  {~me}end;
  {~rb} procedure Report;  var  i:integer;  begin
  {~re} end;
  {~cab} procedure CloseAllObj;begin
  {~cae} end;
```

Итоговая статистика по задаче выглядит следующим образом.

```
Date Time Stamp= 20.12.2008 15:29:36   Ctrl Click |Name - HyperLink `R
E:\work\ObjectClient\Object No Sys\Future\GPSS-Future\book\tmp49.rtf
StartTime         0.00000 EndTime     101268.60510
 |Model |Blocks |Values |CurrentList |SynchronizeList |InterruptList
 |RPList |IPList |SPList |BPList
   Num      NameObject
     2        |Future
     3         |fac01
     4         |fac02
     5         |fac03
     6         |fac04
     7         |fac05
     8         |fac06
     9         |fac07
    10         |fac08
    11         |fac09
    12        |fac010
    13        | que01
    14        | que02
    15        | que03
    16        | que04
    17        | que05
    18        | que06
    19        | que07
```

```
   20        | que08
   21        | que09
   22       | que010
|R `Model
{=Gen} ::Gen 1:Future.NewWait(MeanDev(100,80),0,Gen,Gen+1);
  2: IPlet(1,[fmin]);
  3: que0[IP(1)].Queue;
  4: fac0[IP(1)].Seize;
  5:Future.Wait(exponential(800));
  6: que0[IP(1)].Depart;
  7: fac0[IP(1)].Release;
  8: Terminate(1);

|R `Blocks    BLOCK Report
   Location     Total   Current
        1      1020         0
        2      1020         0
        3      1020        11
        4      1009         0
        5      1009         9
        6      1000         0
        7      1000         0
        8      1000         0
|R `Values
|R `CurrentList Report Current Transactions List  Count= 11
   Tran   Assem     Pri Current    Next       TimeStamp       TimeEnter Inter
    986     986       0       3       4     98385.40788     98385.40788 FALSE
   1006    1006       0       3       4    100113.07495    100113.07495 FALSE
   1007    1007       0       3       4    100209.40023    100209.40023 FALSE
   1012    1012       0       3       4    100602.95843    100602.95843 FALSE
   1013    1013       0       3       4    100681.68557    100681.68557 FALSE
   1014    1014       0       3       4    100736.90468    100736.90468 FALSE
   1016    1016       0       3       4    100925.22652    100925.22652 FALSE
   1017    1017       0       3       4    100955.73481    100955.73481 FALSE
   1018    1018       0       3       4    101076.63489    101076.63489 FALSE
   1019    1019       0       3       4    101182.16789    101182.16789 FALSE
   1020    1020       0       3       4    101234.38272    101234.38272 FALSE
|R `IPList Report ( Integer ) Count= 20
       Tran            Num/Name        Integer
        981                 1              2
        986                 1              2
        991                 1              6
        997                 1              3
       1000                 1              8
       1001                 1              1
       1005                 1              7
       1006                 1              1
       1007                 1              3
       1010                 1              9
       1011                 1              4
       1012                 1              5
       1013                 1              6
       1014                 1              7
       1015                 1             10
       1016                 1              8
       1017                 1              9
       1018                 1             10
       1019                 1              4
       1020                 1              1
|R Facility = `fac01 Entries = 111 Current = 1 Utility =  0.96090 AverTime = 876.65920
   Tran = 1001 IntTran = 0 Active Time = 101268.60510 NextGo = 0
|R Facility = `fac02 Entries = 134 Current = 1 Utility =  0.93450 AverTime = 706.23754
   Tran = 981 IntTran = 0 Active Time = 101268.60510 NextGo = 0
|R Facility = `fac03 Entries = 111 Current = 1 Utility =  0.90194 AverTime = 822.86401
   Tran = 997 IntTran = 0 Active Time = 101268.60510 NextGo = 0
|R Facility = `fac04 Entries = 121 Current = 1 Utility =  0.87380 AverTime = 731.30679
   Tran = 1011 IntTran = 0 Active Time = 101268.60510 NextGo = 0
|R Facility = `fac05 Entries = 119 Current = 0 Utility =  0.87534 AverTime = 744.91246
   Tran = 0 IntTran = 0 Active Time = 101268.60510 NextGo = 0
|R Facility = `fac06 Entries = 97 Current = 1 Utility =  0.79281 AverTime = 827.69922
```

```
   Tran = 991 IntTran = 0 Active Time = 101268.60510 NextGo = 0
|R Facility = `fac07 Entries = 101 Current = 1 Utility =  0.73781 AverTime = 739.77487
   Tran = 1005 IntTran = 0 Active Time = 101268.60510 NextGo = 0
|R Facility = `fac08 Entries = 73 Current = 1 Utility =  0.67310 AverTime = 933.75568
   Tran = 1000 IntTran = 0 Active Time = 101268.60510 NextGo = 0
|R Facility = `fac09 Entries = 72 Current = 1 Utility =  0.63613 AverTime = 894.72774
   Tran = 1010 IntTran = 0 Active Time = 101268.60510 NextGo = 0
|R Facility = `fac010 Entries = 70 Current = 1 Utility =  0.45182 AverTime = 653.64888
   Tran = 1015 IntTran = 0 Active Time = 101268.60510 NextGo = 0
|R Queue = ` que01 Entries = 113 Current = 3 AverQueue =  1.35016 AverTime = 1209.98923
   AverTime(-Ze) = 2485.97787 Max = 3 Min = 0 Zero = 58 Active Time = 101268.60510 NextGo = 0
|R Queue = ` que02 Entries = 135 Current = 2 AverQueue =  1.25117 AverTime = 938.55289
   AverTime(-Ze) = 2262.58285 Max = 3 Min = 0 Zero = 79 Active Time = 101268.60510 NextGo = 0
|R Queue = ` que03 Entries = 112 Current = 2 AverQueue =  1.14718 AverTime = 1037.26350
   AverTime(-Ze) = 2420.28150 Max = 3 Min = 0 Zero = 64 Active Time = 101268.60510 NextGo = 0
|R Queue = ` que04 Entries = 122 Current = 2 AverQueue =  1.14539 AverTime = 950.75241
   AverTime(-Ze) = 2974.14857 Max = 3 Min = 0 Zero = 83 Active Time = 101268.60510 NextGo = 0
|R Queue = ` que05 Entries = 120 Current = 1 AverQueue =  1.01497 AverTime = 856.53942
   AverTime(-Ze) = 3806.84185 Max = 2 Min = 0 Zero = 93 Active Time = 101268.60510 NextGo = 0
|R Queue = ` que06 Entries = 98 Current = 2 AverQueue =  0.90566 AverTime = 935.87063
   AverTime(-Ze) = 3987.62269 Max = 2 Min = 0 Zero = 75 Active Time = 101268.60510 NextGo = 0
|R Queue = ` que07 Entries = 102 Current = 2 AverQueue =  0.80358 AverTime = 797.81360
   AverTime(-Ze) = 5425.13249 Max = 2 Min = 0 Zero = 87 Active Time = 101268.60510 NextGo = 0
|R Queue = ` que08 Entries = 74 Current = 2 AverQueue =  0.72247 AverTime = 988.70190
   AverTime(-Ze) = 5627.99543 Max = 2 Min = 0 Zero = 61 Active Time = 101268.60510 NextGo = 0
|R Queue = ` que09 Entries = 73 Current = 2 AverQueue =  0.67007 AverTime = 929.55195
   AverTime(-Ze) = 5654.77434 Max = 2 Min = 0 Zero = 61 Active Time = 101268.60510 NextGo = 0
|R Queue = ` que010 Entries = 71 Current = 2 AverQueue =  0.47423 AverTime = 676.40369
   AverTime(-Ze) = 6860.66599 Max = 2 Min = 0 Zero = 64 Active Time = 101268.60510 NextGo = 0
|R User/Future = `Future Entries = 2030 Current = 10 AverUser =  8.43534 AverTime = 420.80525
   Max = 11 Min = 0 Active Time = 101268.60510 NextGo = 0
|R Report  User/Future List Count= 10
   Tran   Assem     Pri Current    Next       TimeStamp       SortValue   Group Inter
   1021    1021       0       0       1    101350.56707    101350.56707       0 FALSE
    981     981       0       5       6     98064.74306    101425.85891       0 FALSE
   1005    1005       0       5       6    100079.88156    101511.82539       0 FALSE
   1001    1001       0       5       6     99884.75929    101524.92687       0 FALSE
   1015    1015       0       5       6    100891.05501    101728.89265       0 FALSE
   1000    1000       0       5       6     99852.54758    101984.90896       0 FALSE
   1011    1011       0       5       6    100569.93488    102043.92935       0 FALSE
   1010    1010       0       5       6    100537.56614    102080.00832       0 FALSE
    991     991       0       5       6     98805.44256    102241.44419       0 FALSE
    997     997       0       5       6     99662.58344    103928.50232       0 FALSE
           Num/Name        Integer  |R `Future      Report IP LIST  Count= 2
I                 1           1021
I                 2           1021
```

Пусть необходимо решить такую же задачу, но длина допустимой очереди ограничена 2. И, если подходящего устройства не найдется, то заявка должна удаляться из системы.

Тогда текст модели остается почти тем же.

```
 Const
num=10;
  {~vb} Var
 {~f}Future:TFuture;
{/fac0} fac0:array[1..num] of Tfacility;
{/que0} que0:array[1..num] of Tqueue;
  {~ve}
procedure  facShow;
var i:integer;
begin
  for i:=1 to num do
  ToValueProc(i,fac0[i].A);
```

```
end;
procedure  QueShow;
var i:integer;
begin
  for i:=1 to num do
 ToValueProc(i,que0[i].a);
end;
{/fun0}   function fmin : integer;
var i,j:integer;  begin
j:=1;
for i:=1 to num do if (que0[i].L<que0[j].L) then j:=i;
  if que0[j].L>1 then j:=0;
 result:=j;
  end;
  {~pfe}
  {~ib} procedure Initial;  var  i:integer; begin
 {~f}init(Future,'Future');
   setstart(1000);
{/fac0}  for i:=1 to num do init(fac0[i] , 'fac0'+inttostr(i));
{/que0}  for i:=1 to num do init(que0[i] , ' que0'+inttostr(i));
customProc(['facShow','QueShow'],[facShow,QueShow]);
{=Gen} Future.NewWaitProc(Gen,MeanDev(100,80),0,Gen,Gen+1);
  {~ie}end;
  {~mtb}procedure ModelTxt;begin case ActiveBlock of
{=Gen} ::Gen *:Future.NewWait(MeanDev(100,80),0,Gen,Gen+1);
   *: IPlet(1,[fmin]);
   *: test([Ip(1)<>0,true ],[!*,out]) ;
   *: que0[IP(1)].Queue;
   *: fac0[IP(1)].Seize;
   *:Future.Wait(exponential(800));
   *: que0[IP(1)].Depart;
   *: fac0[IP(1)].Release;
   *: Terminate(1);
::out  *: Terminate ;
  {~mte} end;end;
  {~mb} procedure Simulation;  begin
   start(GetTerm);
  {~me}end;
  {~rb} procedure Report; begin
  {~re} end;
  {~cab} procedure CloseAllObj;begin
  {~cae} end;
```

В нем только изменена пользовательская функция выбора. В ней после нахождения очереди с минимальной длиной выполняется проверка. Не является ли она более 1. И, если это так, то результат функции окажется равным 0.

Поэтому в новой модели добавлен блок

: test([Ip(1)<>0,true],[!,out]) ;

Он пропускает заявки с ненулевым значением Ip(1), и отправляет на метку Out все другие заявки.

Даже на этих примерах, становится ясно, что модели с пользовательскими функциями выбора более логичны и понятны, чем модели, решающие эту задачу «в лоб». Подобным же образом можно выбирать любые другие требуемые объекты.

35 Работа с файлами

В принципе, для работы с текстовыми файлами следует использовать процедуры самого языка Pascal. Для этого следует описать текстовый файл как переменную в самой модели. Эта переменная должна иметь тип TextFile.

Далее следует связать эту переменную с файлом на диске и открыть файл для чтения или записи данных.

В первом случае нужно вызвать процедуры

AssignFile(FileVar,FileName);

Reset(FileVar);

А во втором случае нужно вызвать процедуры

AssignFile(FileVar,FileName);

ReWrite(FileVar);

Здесь FileVar – это файловая переменная, а FileName – это строка, содержащая имя файла на диске.

Далее для чтения данных из файла следует использовать процедуры

Read(FileVar, список переменных через запятую);

Readln(FileVar, список переменных через запятую);

Процедура Read читает данные, не обращая внимания на концы строк, а процедура Readln обязательно дочитывает начатую строку до конца, даже если в ней есть данные, которым не сопоставлены переменные.

Тип переменных может быть вещественным, целым, или строковым.

Переменные в указанном порядке будут читаться из файла, если в файле еще есть непрочитанные данные. Вещественные и целые данные должны разделяться пробелами или концами строки, а строковые - должны занимать строку целиком. В процедуре Read не следует указывать строковые переменные.

А в процедуре Readln можно указывать данные любого из трех указанных типов.

Обращаю ваше внимание, что чтение данных возможно только в переменные соответствующих типов, и невозможно чтение в P или X – параметры.

Для проверки закончился ли файл в процессе чтения, можно использовать функцию EOF(FileVar) – которая имеет значение True (Истина), если данные в файле уже закончились.

Для записи данных в файл следует использовать процедуры

Write(FileVar, список выражений с указание ширины полей вывода через запятую);

Writeln(FileVar, список выражений с указание ширины полей вывода через запятую);

Процедура Write записывает данные в одну строку подряд, а процедура Writeln завершает запись концом строки.

Тип выражений может быть вещественным, целым, или строковым.

Выражения будут вычислены и их значения в указанном порядке будут записаны в файл. После каждого выражения может стоять двоеточие и минимальная ширина поля вывода для значения выражения. При выводе, данные могут «слипаться», если ширина поля вывода будет недостаточной для вывода значения. Поэтому рекомендуется включать в список выражений строки из одного или нескольких пробелов в апострофах, или выбирать ширину поля вывода достаточно большой.

После того, как работа с файлом завершена, файл необходимо закрыть с помощью процедуры CloseFile(FileVar).

Если же вы хотите выполнять те же действия в ходе моделирования, то вы можете воспользоваться следующими блоками.

*: OpenGetBlock(Filevar,FileName); Блок открывает файл Filevar для чтения. Имя файла - FileName.

*: OpenPutBlock(Filevar,FileName); Блок открывает файл Filevar для записи. Имя файла - FileName.

*: CloseBlock(Filevar); Блок закрывает файл Filevar.

*: WriteBlock(Filevar,Value,Leng); Блок записывает вещественное, целое, или строковое Value в файл Filevar. Leng - длина поля вывода, по умолчанию - 0.

*: WritelnBlock(Filevar,Value,Leng); Блок записывает вещественное, целое, или строковое Value с концом строки в файл Filevar. Leng - длина поля вывода, по умолчанию - 0.

*: WritelnBlock(Filevar); Блок записывает конец строки в файл Filevar.

*: ReadBlock(Filevar,Value); Блок читает вещественное или целое число Value из файла Filevar.

*: ReadlnBlock(Filevar,Value); Блок читает вещественное или целое число Value с концом строки из файла Filevar.

*: ReadBlock(Filevar,Value); Блок читает строку Value с концом строки из файла Filevar.

*: ReadlnBlock(Filevar); Блок читает конец строки из файла Filevar.

Категорически не рекомендуется открывать и закрывать файлы прямо в ходе моделирования. Гораздо лучше, если вы откроете файлы в процедуре Initial, а закроете в CloseAllObj.

В принципе, для вывода в текстовые файлы информации типа отчетов «россыпью», можно использовать следующие процедуры, которые, впрочем, вряд ли найдут широкое применение.

ObjName.ReportRXList(FileName); Создает отчет по RX - параметрам объекта. Здесь FileName – имя файла для вывода.

ObjName.ReportIXList(FileName); Создает отчет по IX - параметрам объекта. Здесь FileName – имя файла для вывода.

ObjName.ReportSXList(FileName); Создает отчет по SX - параметрам объекта. Здесь FileName – имя файла для вывода.

ObjName.ReportBXList(FileName); Создает отчет по BX - параметрам объекта. Здесь FileName – имя файла для вывода.

ObjName.ReportFile(FileName); Создает отчет по X - параметрам объекта. Здесь FileName – имя файла для вывода.

ObjName.GroupReportFile(FileName); Создает полный отчет по заявкам группы. FileName – имя файла.

ReportBlocks(FileName); Выводит отчет о блоках в указанный файл.

ReportCurrentList(FileName); Выводит отчет о списке текущих событий в указанный файл.

ReportFutureList(FileName); Выводит отчет о списке будущих событий в указанный файл.

ReportSynchronizeList(FileName); Выводит отчет о списке синхронизации в указанный файл.

ReportInterruptList(FileName); Выводит отчет о списке прерываний в указанный файл.

ReportUserList(FileName); Выводит отчет о списках пользователя в указанный файл.

ReportRPList(FileName); Выводит отчет о RP -параметрах в указанный файл.

ReportIPList(FileName); Выводит отчет о IP -параметрах в указанный файл.

ReportSPList(FileName); Выводит отчет о SP -параметрах в указанный файл.

ReportBPList(FileName); Выводит отчет о BP -параметрах в указанный файл.

В качестве примеров работы с вводом и выводом данных, рассмотрим следующие задачи.

Пусть заявки первого типа поступают на обслуживание в систему с интервалом int1±int2 секунд. Они проходят вначале через num канальное устройство Nac, где обслуживаются в течение, time1±time2 сек. А затем они проходят через одноканальное устройство F, где обслуживаются в течение, time3±time4 сек, и покидают модель.

Пусть также заявки второго типа поступают на обслуживание в систему с интервалом int3±int4 секунд. Они проходят через num канальное устройство Nac, где обслуживаются в течение, time5±time6 сек, занимая по 2 канала. Затем они покидают модель. Промоделировать обслуживание count заявок. Текст такой модели, мог бы иметь, например, следующий вид:

```
 {~vb} Var
 {~f}Future:TFuture;
{/F} F:Tfacility;
{/Nac} Nac:Tstorage;
{/QF} QF:Tqueue;
{/QNac} QNac:Tqueue;
 int1:Double;
 int2:Double;
 int3:Double;
 int4:Double;
 time1:Double;
 time2:Double;
 time3:Double;
 time4:Double;
```

```
 time5:Double;
 time6:Double;
 num:Integer;
 count:Integer;
input:textfile;
  {~ve}
  {~pfe}
  {~ib} procedure Initial;begin
 {~f}init(Future,'Future');
 Assignfile(input,'test1.txt');
 reset(input);
readln( input,int1,int2, int3, int4);
readln( input,time1, time2, time3, time4, time5, time6 );
readln( input, num, count);
closefile(input);
 SetStart(count);
{/F}  init(F,'F');
{/Nac}  init(Nac,' Nac',num);
{/QF}  init(QF,' QF');
{/QNac}  init( QNac,' QNac');
{=Gen} Future.NewWaitProc(Gen,MeanDev(int1,int2),0,Gen,Gen+1);
{=Gen0} Future.NewWaitProc(Gen0,MeanDev(int3,int4),0,Gen0,Gen0+1);
  {~ie}end;
  {~mtb}procedure ModelTxt;begin case ActiveBlock of
{=Gen} ::Gen *:Future.NewWait(MeanDev(int1,int2),0,Gen,Gen+1);
     *: QNac.Queue ;
     *: Nac.Enter ;
     *: QNac.Depart ;
     *:Future.Wait(MeanDev(time1,time2));
     *: Nac.Leave ;
     *: QF.Queue ;
     *: F.Seize ;
     *: QF.Depart ;
     *:Future.Wait(MeanDev(time3,time4));
      *: F.Release ;
     *: Terminate(1) ;
{=Gen0} ::Gen0 *:Future.NewWait(MeanDev(int3,int4),0,Gen0,Gen0+1);
     *: QNac.Queue ;
     *: Nac.Enter(2) ;
     *: QNac.Depart ;
     *:Future.Wait(MeanDev(time5,time6));
     *: Nac.Leave(2) ;
     *: Terminate ;
  {~mte} end;end;
  {~mb} procedure Simulation;  begin
   start(Getterm);
```

```
{~me}end;
{~rb} procedure Report;begin
reportval(int1,'int1 = ');
reportval(int2,'int2 = ');
reportval(int3,'int3 = ');
reportval(int4,'int4 = ');
reportval(time1,'time1 = ');
reportval(time2,'time2 = ');
reportval(time3,'time3 = ');
reportval(time4,'time4 = ');
reportval(time5,'time5 = ' );
reportval(time6,'time6 = ');
reportval(num,'num = ');
reportval(count,'count = ');
{~re} end;
{~cab} procedure CloseAllObj;begin
{~cae} end;
```

По этому тексту следует дать ряд комментариев. Переменные не следует вставлять с помощью пункта Insert Object, а нужно просто описать в разделе Var. Это обусловлено тем, что чтение и запись этих переменных при инициализации и закрытии модели могут создать проблемы взаимодействия с файлом ‘Test1.txt’. По этой же причине, в процедуру report следует самостоятельно вставить строки вида reportval(int1,'int1 = '); по каждой переменной.

Они обеспечат вывод значений этих переменных в отчет. Обратите также внимание, что открытие файла происходит в процедуре initial а закрытие – в процедуре closeallobj модели.

Содержимое файла ‘Test1.txt’, могло быть следующим.

100 50 1000 100 int1 int2 int3 int4

400 100 90 20 350 200 time1 time2 time3 time4 time5 time6

5 1000 num count

Здесь данные расположены в соответствии с операторами чтения. А тексты вида int1 int2 int3 int4 на самом деле читаться не будут, так как чтение выполняется операторами readln. Сами такие тексты просто поясняют смысл чисел, введенных в строке. В принципе, подобный подход рекомендуется во всех случаях, когда данные вводятся из файла. Это поможет вам избежать ошибок при вводе данных.

Результирующая статистика по модели имеет вид.

```
Date Time Stamp= 20.12.2008 20:59:15   Ctrl Click |Name - HyperLink `R
E:\work\ObjectClient\Object No Sys\Future\GPSS-Future\book\tmp51.rtf
StartTime         0.00000 EndTime     100506.81747
 |Model |Blocks |Values |CurrentList |SynchronizeList |InterruptList
 |RPList |IPList |SPList |BPList
   Num      NameObject
     2        |Future
     3             |F
     4           | Nac
     5            | QF
     6          | QNac
 |R `Model
 {=Gen} ::Gen 1:Future.NewWait(MeanDev(int1,int2),0,Gen,Gen+1);
      2: QNac.Queue ;
```

```
      3: Nac.Enter ;
      4: QNac.Depart ;
      5:Future.Wait(MeanDev(time1,time2));
      6: Nac.Leave ;
      7: QF.Queue ;
      8: F.Seize ;
      9: QF.Depart ;
      10:Future.Wait(MeanDev(time3,time4));
       11: F.Release ;
      12: Terminate(1) ;
{=Gen0} ::Gen0 13:Future.NewWait(MeanDev(int3,int4),0,Gen0,Gen0+1);
      14: QNac.Queue ;
      15: Nac.Enter(2) ;
      16: QNac.Depart ;
      17:Future.Wait(MeanDev(time5,time6));
      18: Nac.Leave(2) ;
      19: Terminate ;

 |R `Blocks    BLOCK Report
    Location     Total   Current
         1      1005         0
         2      1005         1
         3      1004         0
         4      1004         0
         5      1004         3
         6      1001         0
         7      1001         1
         8      1000         0
         9      1000         0
        10      1000         0
        11      1000         0
        12      1000         0
        13       100         0
        14       100         0
        15       100         0
        16       100         0
        17       100         1
        18        99         0
        19        99         0
 |R `Values
int1 =     100.0000000
int2 =      50.0000000
int3 =    1000.0000000
int4 =     100.0000000
time1 =     400.0000000
time2 =     100.0000000
time3 =      90.0000000
time4 =      20.0000000
time5 =     350.0000000
time6 =     200.0000000
num =           5
count =       1000
 |R `CurrentList Report Current Transactions List  Count= 2
    Tran   Assem     Pri Current    Next       TimeStamp       TimeEnter Inter
    1106    1106       0       2       3    100459.32647    100459.32647 FALSE
    1101    1101       0       7       8    100014.86668    100476.95082 FALSE
 |R Facility = `F Entries = 1000 Current = 0 Utility =  0.89386 AverTime = 89.83902
    Tran = 0 IntTran = 0 Active Time = 100506.81747 NextGo = 0
 |R Storage = ` Nac Entries = 1204 Current = 5 Average =  4.66489 AverTime = 389.41263 Utility =
0.93298
    Max = 5 Min = 0 Tran = 1105 ACapac =  5.00000 Capacity = 5 Active Time = 100506.81747 NextGo
= 0
 |R Queue = ` QF Entries = 1001 Current = 1 AverQueue =  0.81317 AverTime = 81.64709
    AverTime(-Ze) = 149.96099 Max = 5 Min = 0 Zero = 456 Active Time = 100506.81747 NextGo = 0
 |R Queue = ` QNac Entries = 1105 Current = 1 AverQueue =  0.91227 AverTime = 82.97695
    AverTime(-Ze) = 216.76011 Max = 5 Min = 0 Zero = 682 Active Time = 100506.81747 NextGo = 0
 |R User/Future = `Future Entries = 3211 Current = 6 AverUser =  6.57602 AverTime = 205.83447
    Max = 8 Min = 0 Active Time = 100506.81747 NextGo = 0
 |R Report  User/Future List Count= 6
    Tran   Assem     Pri Current    Next       TimeStamp       SortValue   Group Inter
```

```
    1107    1107       0       0       1    100528.68960    100528.68960       0 FALSE
    1093    1093       0      17      18    100042.12333    100533.98934       0 FALSE
    1102    1102       0       5       6    100137.71135    100632.23217       0 FALSE
    1104    1104       0       5       6    100280.57252    100674.45878       0 FALSE
    1105    1105       0       5       6    100402.96658    100867.64439       0 FALSE
    1103    1103       0       0      13    101137.12857    101137.12857       0 FALSE
            Num/Name        Integer  |R `Future      Report IP LIST  Count= 4
I                 1           1006
I                 2           1107
I                13            101
I                14           1103
```

В тех случаях, когда данные из модели выводятся в файл, проблем с файлами заметно меньше. Рассмотрим следующую задачу

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. Эту задачу мы уже рассматривали ранее, но сейчас выведем в файл ‘Test2.txt’ информацию по обслуживанию 50 заявок в виде. Номер заявки, время обслуживания.

Тогда модель работы такого устройства может иметь вид.

```
  {~vb} Var
 {~f}Future:TFuture;
{/q2} q2:Tqueue;
{/fu} fu:Tfacility;
{/tbl} tbl:Ttable;
out:textfile;
  {~ve}
  {~pfe}
  {~ib} procedure Initial;begin
 {~f}init(Future,'Future');
 SetStart(50);
{/q2}  init(q2,'q2');
{/fu}  init(fu,' fu');
{/tbl}  init(tbl,' tbl',0,10,20);
{=Gen1} Future.NewWaitProc(Gen1,MeanDev(300,50),0,Gen1,Gen1+1);
  {~ie}end;
  {~mtb}procedure ModelTxt;begin case ActiveBlock of
{=Gen1} ::Gen1 *:Future.NewWait(MeanDev(300,50),0,Gen1,Gen1+1);
   *:q2.queue;
   *:fu.seize;
   *:q2.depart;
   *: mark ;
  *:Future.Wait(DFun(random,[0.1,0.3,0.7,1],[100,80,120,60]));
   *:fu.release;
   *:tbl.tabulate(TranAge);
   *: writeblock(out,'Tran= ') ;
   *: writeblock(out,Tran) ;
   *: writeblock(out,'m1= ') ;
   *: writelnblock(out,TranAge) ;
   *:terminate(1);
```

```
  {~mte} end;end;
  {~mb} procedure Simulation;  begin
  Assignfile(out,'test2.txt');
  Rewrite(out);
 {/sh}     start(Getterm);
 show(tbl);
show(tbl,2);
closefile(out);
  {~me}end;
  {~rb} procedure Report;begin
  {~re} end;
  {~cab} procedure CloseAllObj;begin
  {~cae} end;
```

Результирующая статистика в данном случае интереса не представляет, а содержимое выходного файла имеет вид. Отметим только, что в данной модели содержимое этого файла следует смотреть только после закрытия модели. Если вы хотите посмотреть ее раньше, то процедуру закрытия файла следует переместить в процедуру Simulation и расположить после процедуры start.

```
Tran= 1m1=  6.0E+0001
Tran= 2m1=  1.2E+0002
Tran= 3m1=  1.2E+0002
Tran= 4m1=  1.2E+0002
Tran= 5m1=  1.2E+0002
Tran= 6m1=  1.2E+0002
Tran= 7m1=  1.2E+0002
Tran= 8m1=  1.2E+0002
Tran= 9m1=  8.0E+0001
Tran= 10m1=  6.0E+0001
Tran= 11m1=  1.2E+0002
Tran= 12m1=  1.2E+0002
Tran= 13m1=  1.2E+0002
Tran= 14m1=  8.0E+0001
Tran= 15m1=  1.0E+0002
Tran= 16m1=  8.0E+0001
Tran= 17m1=  1.2E+0002
Tran= 18m1=  1.2E+0002
Tran= 19m1=  1.2E+0002
Tran= 20m1=  6.0E+0001
Tran= 21m1=  6.0E+0001
Tran= 22m1=  1.0E+0002
Tran= 23m1=  1.2E+0002
Tran= 24m1=  6.0E+0001
Tran= 25m1=  1.0E+0002
Tran= 26m1=  6.0E+0001
Tran= 27m1=  8.0E+0001
```

Tran= 28m1= 1.2E+0002
Tran= 29m1= 6.0E+0001
Tran= 30m1= 1.0E+0002
Tran= 31m1= 6.0E+0001
Tran= 32m1= 1.2E+0002
Tran= 33m1= 8.0E+0001
Tran= 34m1= 8.0E+0001
Tran= 35m1= 8.0E+0001
Tran= 36m1= 1.2E+0002
Tran= 37m1= 8.0E+0001
Tran= 38m1= 1.2E+0002
Tran= 39m1= 8.0E+0001
Tran= 40m1= 1.2E+0002
Tran= 41m1= 1.2E+0002
Tran= 42m1= 1.2E+0002
Tran= 43m1= 8.0E+0001
Tran= 44m1= 1.2E+0002
Tran= 45m1= 6.0E+0001
Tran= 46m1= 1.2E+0002
Tran= 47m1= 6.0E+0001
Tran= 48m1= 1.2E+0002
Tran= 49m1= 8.0E+0001
Tran= 50m1= 6.0E+0001

Если вы хотите выводить данные в отчет не с помощью Reporval, а с помощью операторов Write(FileVar, Value_List); Writeln(FileVar, Value_List); то вместо FileVar следует указывать имя файла вывода для отчета, а именно, FileRep.

36 Основы управления моделью и развитие GPSS - Future

Для управления моделью используются процедуры переопределения объектов, вывода результатов в той или иной форме, рассмотренные ранее, а также следующие процедуры.

Start(Tg); Задает начальное значение параметру Term и запускает модель.

ResetModel; Выполняет сброс модели. Точнее она сбрасывает в ноль статистику, накопленную по устройствам, очередям, спискам пользователя, таблицам и блокам. Однако она не удаляет заявок из модели.

Подробнее действие процедуры ResetModel можно описать следующим образом:

значение условного относительного времени (Clock) устанавливается в ноль;
значение условного абсолютного времени (AClock) остается неизменным;
датчик псевдослучайных чисел остается неизменным;
значения P - параметров и переменных не изменяются.

Счетчики блоков (Total(j)) полагаются равными (Current(j)). Временные интегралы устройств устанавливаются в ноль.

Временные интегралы содержимого многоканальных устройств устанавливаются в ноль. Счетчики числа входов в многоканальное устройство и максимальное содержимое многоканального устройства устанавливаются в

соответствии с текущим числом занятых каналов в данный момент времени. Временные интегралы всех очередей сбрасываются в ноль.

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

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

ClearModel(Clear); Очищает модель от всех P параметров при Clear =True . При Clear =False - в модели значения всех P - параметров остаются неизменными.

Следующие процедуры могут использоваться в процедуре Repot.

Reportval(Varia,Name); Выводит значение вещественного, целого, строкового, или логического типа в отчет, Name - заголовок.

Эти процедуры могут использоваться в процедуре Initial.

InitRand(Value); Устанавливает начальное целое значение для генератора случайных чисел. Эта процедура может применяться для гарантированного повторения одного и того же процесса моделирования. Для этого нужно вызвать ее в самом начале процедуры Initial, по крайней мере, до всех процедур инициализации генераторов заявок.

Для полноценного управления моделью может быть недостаточно возможностей имеющихся в модели процедур. Самое простое – это, возможно, вам понадобится несколько различных процедур в стиле Simulation. Или, допустим, вы захотите выводить гистограммы таблиц по одной. Тогда, возможно, вам понадобится иметь несколько пользовательских процедур. Чтобы иметь возможность вызывать такие процедуры, на главной странице модели имеется выпадающий список подписанный Custom Procedure. Из этого списка можно вызывать любую пользовательскую процедуру типа Simulation.
Эти процедуры должны быть описаны в модели, и не иметь параметров.

Далее, в процедуре Initial вам следует вызвать процедуру

CustomProc(st,apr); она создает перечень заголовков для вызова процедур пользователя и перечень имен самих процедур. Перечни записываются в квадратных скобках через запятую. После этого, на главной странице вы можете вызывать эти процедуры.

Процедура

SetPage(Page); - переключает приложение на нужную. Именованные константы для страниц.
spSimulation, ReportSetting, spReport, spLine, spValues, spBar, spAbout, spReserve ,spLicence

Если вам недостаточно тех блоков, процедур и функций, которые есть в модели, то нужные вам процедуры и функции просто описываются в начальной части модели, сразу после описания ее переменных.
Для построения блоков следует учитывать, что блок – это особая процедура, которая корректно работает со списками заявок. Поэтому, новые блоки должны строиться как особые процедуры, которые при любых значениях их параметров выполняют ровно один из ранее определенных блоков. Чаще всего новые блоки строятся на основе блока MoveNext, который, не делая ничего полезного, продвигает заявку в следующий блок. После того, как вы построили новый блок, вы можете пользоваться им наравне с блоками системы.

Приведем примеры применения процедуры CustomProc(st,apr), которая позволяет иметь несколько планов работы с моделью, а также примеры создания новых блоков.

В парикмахерскую приходят клиенты для стрижки с интервалом 16+-6 минут. Обслуживание клиента занимает 15+-3 минуты.

Промоделировать 1000 дней работы парикмахерской по 8 часов в день, учитывая, что по окончании рабочего дня новые клиенты на обслуживание не принимаются, а обслуживаются только те, которые уже есть в парикмахерской. В данной модели имеется 3 пользовательских процедуры, которые позволяют создать и просмотреть гистограммы таблиц, а также график, созданный в динамических массивах типа ADouble.

```
  {~vb} Var
 {~f}Future:TFuture;
SHUTDOWN:Boolean;
{/JOEQ} JOEQ:Tqueue;
{/JOE} JOE:Tfacility;
{/tb} tb:Ttable;
{/tab0} tab0:Ttable;
{/B0} BX,BY:ADouble;
  {~ve}
procedure p1;
begin
   show(tb,0);
   setPage(spbar);
end;
procedure p2;
begin
   show(tab0,1);
   setPage(spbar);
end;
procedure p3;
begin
  CShow(bX,BY,1);
  bX := Nil;
  bY := Nil;
  setPage(spline);
end;
  {~pfe}
  {~ib} procedure Initial;begin
 {~f}init(Future,'Future');
SETSTART(1000);
  SHUTDOWN:=false;
{/JOEQ}  init(JOEQ,' JOEQ');
{/JOE}  init(JOE,' JOE');
{/tb}  init(tb,' tb',0,1,100);
{/tab0}  init(tab0,' tab0',0,1,100);
   CustomProc(['Tb','Tab0','Line'],[p1,p2,p3]);
```

```
{=Gen} Future.NewWaitProc(Gen,MeanDev(16,6),0,Gen,Gen+1);
{=Gen0} Future.NewWaitProc(Gen0,480,0,Gen0,Gen0+1);
  {~ie}end;
  {~mtb}procedure ModelTxt;begin case ActiveBlock of
{=Gen} ::Gen *:Future.NewWait(MeanDev(16,6),0,Gen,Gen+1);
 // ARRIVALS EVERY 18 +- 6 MINUTES
   *: TEST([NOT SHUTDOWN,true],[!*,go]);  // NOT AFTER SHUTDOWN
   *: JOEQ.QUEUE; // GET IN LINE
   *: JOE.SEIZE;  // GRAB THE BARBER
   *: JOEQ.DEPART;  //  EXIT LINE
   *:Future.Wait(MeanDev(15,3));
  // HAIRCUT TAKES 15 +- 3 MINUTES ;
   *: JOE.RELEASE;   // FREE THE BARBER ;
   *: tb.tabulate(TranAge);
   *: TERMINATE(0);  // EXIT THE SHOP ;
::go  *: TERMINATE(0);  // EXIT THE SHOP ;

{=Gen0} ::Gen0 *:Future.NewWait(480,0,Gen0,Gen0+1);
 //  RUN FOR 800 HRS (IN MINUTES)
   *:LET(SHUTDOWN,TRUE);  //LOGIC S SHUTDOWN ENTER SHUTDOWN
MODE
   *:TEST([(JOEQ.L=0) AND (JOE.L=0)],[!*]);   // DRAIN QUEUE,WAIT UNTIL
JOE IS IDLE
   *:LET(SHUTDOWN,false);
//   *:topoint(1,aclock,tb.b);
   *:  RAadd(BX,[aclock])  ;
   *:  RAadd(BY,[tb.A])  ;
   *: tab0.tabulate(TranAge);
   *:tostring(0,inttostr(term));
   *:currentblocks;
   *:TERMINATE( 1); // SHUT DOWN
  {~mte} end;end;
  {~mb} procedure Simulation;  begin
    Linetitle(1,'JOEQ.a');
    start(GetTerm);
  {~me}end;
  {~rb} procedure Report;begin
  reportval(SHUTDOWN,'SHUTDOWN = ');
  {~re} end;
  {~cab} procedure CloseAllObj;begin
  {~cae} end;
```

В данном случае итоговая статистика не представляет интереса, просто по окончании моделирования можно просмотреть соответствующие результаты с помощью пользовательских функций.

Рассмотрим пример создания пользовательских функций.

Пусть в систему поступают заявки со средним интервалом 120, распределенным по экспоненциальному закону. Пусть с вероятностью 0.4 выбирается для обслуживания устройство Kass, а с вероятностью 0.6 - устройство Kass1. В устройстве Kass заявка обслуживается за 150+-100 единиц времени, а в устройстве Kass1 – за среднее время 200, распределенное по экспоненциальному закону. Промоделировать обслуживание 1000 заявок. Модель такой системы с применением пользовательского блока выглядит, например, следующим образом.

```
  {~vb} Var
 {~f}Future:TFuture;
{/QKass} QKass:Tqueue;
{/Kass} Kass:Tfacility;
{/kass1} kass1:Tfacility;
K:array[1..2] of Tfacility;
  {~ve}
{/Proc0}   procedure  AddAdvance(i:integer)   ;
   begin
      if i=1 then  Future.Wait(MeanDev(150,100))
      else  Future.Wait(exponential(200));
    end;
  {~pfe}
  {~ib} procedure Initial;begin
 {~f}init(Future,'Future');
  SetStart(1000);
{/QKass}  init(QKass,' QKass');
{/Kass}  init(Kass,' Kass');
{/kass1}  init(kass1,' kass1');
k[1]:=kass;
k[2]:=kass1;
{=Gen} Future.NewWaitProc(Gen,Exponential(120),0,Gen,Gen+1);
  {~ie}end;
  {~mtb}procedure ModelTxt;begin case ActiveBlock of
{=Gen} ::Gen *:Future.NewWait(Exponential(120),0,Gen,Gen+1);
    *: Qkass.Queue ;
    *: IPlet(1,[IByBool([random<0.4,true],[1,2])]) ;
    *:K[Ip(1)].Seize ;
   *: Qkass.Depart ;
   *: AddAdvance(ip(1)) ;
   *: K[Ip(1)].release ;
   *: ToPoint(2,aclock,Qkass.A) ;
  *: ToPoint(1, aclock,Kass1.A) ;
  *: ToPoint(0, aclock,Kass.A) ;
   *: terminate(1) ;
  {~mte} end;end;
  {~mb} procedure Simulation;  begin
   start(Getterm);
  {~me}end;
```

{~rb} procedure Report;begin
{~re} end;
{~cab} procedure CloseAllObj;begin
{~cae} end;

Выходная статистика такой модели имеет следующий вид, и ее содержание подтверждает ее предполагаемые свойства.

```
Date Time Stamp= 20.12.2008 21:22:16   Ctrl Click |Name - HyperLink `R
E:\work\ObjectClient\Object No Sys\Future\GPSS-Future\book\tmp54.rtf
StartTime          0.00000 EndTime     120662.75034
 |Model |Blocks |Values |CurrentList |SynchronizeList |InterruptList
 |RPList |IPList |SPList |BPList
   Num      NameObject
     2        |Future
     3        | QKass
     4         | Kass
     5        | kass1
 |R `Model
 {=Gen} ::Gen 1:Future.NewWait(Exponential(120),0,Gen,Gen+1);
     2: Qkass.Queue ;
     3: IPlet(1,[IByBool([random<0.4,true],[1,2])]) ;
     4:K[Ip(1)].Seize ;
    5: Qkass.Depart ;
    6: AddAdvance(ip(1)) ;
    7: K[Ip(1)].release ;
    8: ToPoint(2,aclock,Qkass.A) ;
   9: ToPoint(1, aclock,Kass1.A) ;
   10: ToPoint(0, aclock,Kass.A) ;
    11: terminate(1) ;

 |R `Blocks    BLOCK Report
    Location     Total   Current
         1      1013         0
         2      1013         0
         3      1013        12
         4      1001         0
         5      1001         0
         6      1001         1
         7      1000         0
         8      1000         0
         9      1000         0
        10      1000         0
        11      1000         0
 |R `Values
 |R `CurrentList Report Current Transactions List  Count= 12
    Tran   Assem     Pri Current    Next       TimeStamp       TimeEnter Inter
     997     997       0       3       4    119198.10723    119198.10723 FALSE
     998     998       0       3       4    119201.39572    119201.39572 FALSE
     999     999       0       3       4    119351.40164    119351.40164 FALSE
    1002    1002       0       3       4    119436.83717    119436.83717 FALSE
    1004    1004       0       3       4    119986.49012    119986.49012 FALSE
    1005    1005       0       3       4    120053.71240    120053.71240 FALSE
    1006    1006       0       3       4    120323.97982    120323.97982 FALSE
    1009    1009       0       3       4    120433.52204    120433.52204 FALSE
    1010    1010       0       3       4    120508.99187    120508.99187 FALSE
    1011    1011       0       3       4    120570.90924    120570.90924 FALSE
    1012    1012       0       3       4    120580.94785    120580.94785 FALSE
    1013    1013       0       3       4    120647.30428    120647.30428 FALSE
 |R `IPList Report ( Integer ) Count= 13
        Tran            Num/Name        Integer
         996                   1              2
         997                   1              2
         998                   1              2
         999                   1              2
        1002                   1              2
        1004                   1              2
        1005                   1              2
        1006                   1              2
```

```
        1009                  1              2
        1010                  1              2
        1011                  1              2
        1012                  1              2
        1013                  1              1
 |R Facility = ` Kass Entries = 434 Current = 0 Utility =  0.54367 AverTime = 151.15280
    Tran = 0 IntTran = 0 Active Time = 120662.75034 NextGo = 0
 |R Facility = ` kass1 Entries = 567 Current = 1 Utility =  0.91753 AverTime = 195.25794
    Tran = 996 IntTran = 0 Active Time = 120662.75034 NextGo = 0
 |R Queue = ` QKass Entries = 1013 Current = 12 AverQueue =  8.46165 AverTime = 1007.90322
    AverTime(-Ze) = 1429.98034 Max = 27 Min = 0 Zero = 299 Active Time = 120662.75034 NextGo = 0
 |R User/Future = `Future Entries = 2015 Current = 2 AverUser =  1.95827 AverTime = 117.26587
    Max = 3 Min = 0 Active Time = 120662.75034 NextGo = 0
 |R Report  User/Future List Count= 2
    Tran   Assem     Pri Current    Next       TimeStamp       SortValue   Group Inter
    1014    1014       0       0       1    120685.14862    120685.14862       0 FALSE
     996     996       0       6       7    119188.00549    121085.12499       0 FALSE
            Num/Name         Integer  |R `Future      Report IP LIST  Count= 2
I                 1           1014
I                 2           1014
```

Такая модель выглядит более наглядной за счет использования пользовательских функций. Если бы ее построить «в лоб», то она оказалась бы менее наглядной.

Приложение 1.
Блоки, процедуры и функции языка, "GPSS - Future"

Работа с конвертером.

Внимание, категорически запрещается удаление или редактирование строк, содержащих комментарий следующего вида {~текст}. Любая модификация этих строк может разрушить работоспособность модели!! Эти строки изначально должны быть в модели. Если модель испорчена, то нужно выполнить пункт "New", и строить ее заново.

Построение модели начинается с пункта "New" или сразу после запуска. Вначале надо вставить в модель все нужные объекты, (пункт "Insert Object") а затем в процедуре ModelTxt дописать текст модели. Далее модель следует сохранить в файле на диске, и с ней можно начинать работу. Если позднее понадобится вставить новые объекты или удалить имеющиеся, то это всегда можно сделать.

Подробности этого процесса описаны ниже. Для работы с системой следует знать любую версию языка GPSS, основы программирования на любом языке высокого уровня, например, Basic, Pascal, C++ и так далее.

Кроме того, нужно проанализировать несколько примеров из папки Models. При разработке первых простых моделей, не следует вручную изменять содержимое каких - либо процедур или разделов модели, кроме ModelTxt.

Далеко не все примеры из папки Models в работоспособном состоянии. Они являются отладочными примерами, соответствующими разным стадиям развития системы. Однако неработающих примеров немного, и в них просто не так названы некоторые процедуры и блоки, или должен быть другой набор параметров.

Любая модель состоит из следующих разделов.

1. Раздел описаний личных переменных и объектов. Он заканчивается строкой
   {~ve}

2.Раздел описаний личных процедур и функций, которые используются разработчиком в модели. Он заканчивается строкой
   {~pfe}

3. Раздел инициализации (процедура Initial). Здесь создаются нужные объекты, и задаются начальные значения переменным.
Здесь же может быть выполнена любая другая подготовительная работа для начала моделирования. Он заканчивается строкой.
{~ie}end;
4. Раздел собственно модели (процедура ModelTxt). Здесь последовательно располагаются блоки модели.
(Фактически это специальные процедуры GPSS - Future). Они размещаются после строки
{~mtb}procedure ModelTxt;begin case NextBlock of
Блоки модели должны иметь вид:
::метка *:блок; или
*:блок;

В тексте модели метки указываются непосредственно, и могут входить в корректные выражения целого типа. После конвертирования, метки представляются как константы целого типа, а конструкции *: - как номера блоков. Вместо метки следующего блока можно указывать конструкцию !* . Она также может участвовать в выражениях.

Если блок модели является составным, то есть, составлен из нескольких операторов языка Pascal, то он обрамляется ключевыми словами Begin End, и должен работать так, чтобы непременно пройти ровно через один блок GPSS - Future, лучше всего - блок Nop. При создании новых блоков их желательно оформлять их по образцу:

```
Begin
  If ErrBl('BlockName') then exit; // Эта строка не обязательна!!
  DoSomething;
  Nop;
End;
```

Здесь ErrBl('BlockName') - функция проверки на правильность положения блока.
В описании, перед именем каждого блока помещен текст вида *: .
При развитии системы, рекомендуется придерживаться тех же правил для вновь созданных блоков.
Раздел заканчивается строкой
{~mte} else modelerror;end;end;
5. Раздел манипуляций с моделью (процедура Simulation). Здесь указаны конкретные команды манипуляции с моделью. В нём можно настраивать модель, и готовить ее к новому моделированию. В частности, можно управлять P - параметрами, многоканальными устройствами, таблицами, и переменными, выводить графики и гистограммы и делать многое другое. Раздел завершается строкой
{~me}end;
6. Раздел формирования дополнительных элементов отчета, например, отчет по выражениям или переменным. (процедура Report). Раздел завершается строкой
{~re} end;
7. Раздел удаления динамически созданных пользователем переменных (процедура CloseAllObj). Раздел завершается строкой
{~cae} end; В нём следует каждой динамически созданной переменной (Чаще всего, массиву), присвоить значение Nil.

При определении объектов, следует использовать пункт меню "Insert Obj".

Тогда тип объекта выбирается из списка, а имя объекта должно вводиться в соответствующем поле приложения. В принципе, можно согласиться с автоматически сгенерированным именем переменной или объекта.

Все нужные вызовы процедур для работы с объектом автоматически размещаются в соответствующих разделах. Иногда эти вызовы имеют различные наборы аргументов, тогда следует их ввести в поле аргументов.

Для удаления всех вызовов и описания объекта, вставленного с помощью пункта "Insert Obj", удобно воспользоваться пунктом "Delete Obj". Тогда нужно поместить курсор в строку, содержащую сгенерированный для объекта текст, и щелкнуть по "Delete Obj". За счет комментария в начале строки, будут удалены все строки текста, сгенерированные для объекта пунктом " Insert Obj".

Пример полного текста модели приводится ниже. Фактически, при использовании пунктов "Insert Obj" и "Delete Obj" ваши затраты времени на построение модели сравнимы, со временем на ввод собственно текста модели в разделе 4 (зелёный текст).

```
 {`RR Ctrl Click: |Initial |ModelTxt |Simulation  |Report |CloseAllObj |End }
 {~vb} Var
 {~f}  Future: TFuture;
{=Kass} Kass : TFacility;
{=Qkass} Qkass : TQueue;
 {~ve}
{~pfe}
 {~ib} procedure Initial;begin {`Initial |RR }
 {~f}  init(Future,'Future');
setstart(10000);
{=Gen} Future.NewWaitProc(Gen,Exponential(120),0,Gen,Gen+1);
{=Kass} Init(Kass,'Kass');
{=Qkass} Init(Qkass,'Qkass');
 {~ie} end;
 {~mtb}procedure ModelTxt;begin case ActiveBlock of{`ModelTxt |RR }
{=Gen} ::Gen *:Future.NewWait(Exponential(120),0,Gen,Gen+1);
   *: Qkass.Queue ;
   *: Kass.Seize ;
   *: Qkass.Depart ;
   *: Future.Wait( MeanDev(105,25));
   *: Kass.release ;
   *: ToPoint (1,aclock,Qkass.A) ;
   *: Terminate (1);
 {~mte} end;end;
 {~mb} procedure Simulation;begin {`Simulation |RR }
 start(GetTerm);
 {~me}end;
 {~rb} procedure Report;begin {`Report |RR }
 {~re} end;
 {~cab} procedure CloseAllObj;begin {`CloseAllObj |RR }
 {~cae} end; {`End |RR }
```

При отладке модели наиболее опасной ошибкой является появление блоков вне процедуры ModelTxt или появления вызовов процедур, не являющихся блоками в процедуре
ModelTxt. Практически все эти ошибки обнаруживает система, а остальные - обычно вызывают системные прерывания.

Отметим, что конвертер корректно обрабатывает все виды комментариев, допустимых в языка Object Pascal.

Большая часть элементов оборудования в GPSS являются объектами.
Почти все процедуры и функции для любого объекта, кроме процедур Init, должны вызываться в форме ObjName.ProcedureName(Argument List) или ObjName.FunctionName(Argument List) .
Здесь ObjName - это то имя, под которым описан объект.

Для работы с системой, следует знать одну из версий GPSS, например, GPSS - World, и ознакомиться с примерами, из папки Models.

Работа по созданию конкретной модели требует выполнения следующих шагов.
1. Запустить приложение Converter.exe
2. Открыть в нем имеющуюся модель, или создать новую. Если создана новая модель, то ее следует сохранить на диске.
3. Выполнить пункт "Run" При выполнении этого пункта текст модели автоматически сохраняется на диске.
Если в тексте модели имеются ошибки, то нужно их просмотреть, исправить, и вновь выполнить шаг 3.

Как правило, строки с ошибками подсвечиваются в модели, когда вы просматриваете сообщения об ошибках, однако часть ошибок может и не подсвечиваться. Тогда их нужно искать самостоятельно. Если ошибок нет, то запускается приложение с конкретной моделью и с ним можно проводить опыты.

Конвертер может выполнять вставку файла. Файл может быть в текстовом формате, или в формате .RTF.
Вставка выполняется по пунктам подменю Edit. Текст со вставками не перезаписывается при компиляции. Возврат к первоначальному тексту происходит по пункту ReOpen. Вставки могут быть вложенными. Они могут располагаться и в ModelTxt, то есть собственно в модели. Не следует в модели использовать стандартные вставки компилятора, а только специальные вставки системы. Вставка имеет вид
- {#I.. FileName } и описывает файл относительно модели (папка с моделью).
Здесь допускается и имя файла относительно диска, то есть можно указывать полный путь к файлу.
Соединение моделей позволяет выполнять конструкция вида:
{#U.. FileNameList }. Здесь FileNameList - список файлов моделей через запятую.
В этом случае соединяются одноименные разделы файлов с моделями.
Если нужно, чтобы часть текста процедуры ModelTxt не попала в отчет по моделированию, и по нему не была сформирована статистика, то нужно обрамить эту часть текста символами !- и !+ , которые выключают и включают отображение сведений о блоках и выдачу самого текста блоков. Символы не могут появляться вне процедуры ModelTxt.
То есть они должны располагаться ниже строки с {~mtb}, но выше строки с {~mte}.
Выражения и функции языка Object Pascal в GPSS - Future.

В GPSS - Future можно использовать выражения в качестве операндов,
Вид выражений фактически тот же, что и в Object Pascal. В выражении используют функции модели, 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.
Для вещественных данных используется 8- байтовое представление (тип Double), а для целых - 4 - байтовое (тип Integer).
Строковые константы представляют собой текст, заключенный в апострофы, например, 'это строковая константа' 'It is string.' (тип String).
Логические константы могут быть только истина (True) и ложь (False) (тип Boolean).
Массивы переменных и объектов могут использоваться для косвенного обращения к их элементам.
Ниже приводятся операции и библиотечные функции, используемые в выражениях.
Арифметические операции.
* - операция арифметического умножения;
/ - операция арифметического деления;
Div - операция деления нацело, используется только для целых данных;
Mod - операция определения модуля, используется только для целых данных. Ее результатом является остаток от деления нацело;
+ - операция арифметического или строкового сложения;
- - операция арифметического вычитания.
знак минус;
Отношения.
Могут соединять пару арифметических или строковых выражений.
Если это действительно выражения, то они должны быть в круглых скобках.
< - меньше;
<= - меньше или равно;
> - больше;
>= - больше или равно;
= - равно;
<> - не равно;
Логические выражения.
Могут соединять логические данные, и, в частности, отношения.
NOT - логическое отрицание: True, если операнд False;
False, если операнд True;
AND - операция логического умножения: True, если оба операнда True, False - в противном случае;
OR - операция логического сложения: True, если хотя один из операндов True, False - в противном случае;
XOR - операция сложения по модулю два: True, если операнды различны, False - в противном случае;
Элементарные функции.
Abs( ) - абсолютное значение операнда;
ArcTan( ) - арктангенс аргумента;
Cos( ) - косинус операнда в радианах;
Exp( ) - экспонента операнда;

Int( ) - целая часть числа, результат - вещественный;
Ln( ) - натуральный логарифм операнда;
Log10()- десятичный логарифм операнда;
Pi - предопределенная константа равная пи.
Power(base, exponent) - возведение в степень exponent основания base.
Random - случайное число, равномерно распределенное в диапазоне от 0 до 1.
Random(Num) - случайное целое число, равномерно распределенное в диапазоне от 0 до Num-1 (Num>0).
RandSeed - целое число, которое представляет собой текущее значение встроенного генератора случайных чисел.
Round() - вещественное число округляется до целого.
Sin( ) - синус операнда в радианах;
Sqr( ) - квадрат операнда;
Sqrt() - корень квадратный из операнда.
Trunc() усеченное целое, дробная часть отбрасывается.
Tan( ) - тангенс операнда в радианах;
Функции преобразования.
StrToInt() - преобразует строку - в целое число.
IntToStr() - преобразует целое число - в строку.
StrToFloat() - преобразует строку - в вещественное число.
FloatToStr() - преобразует вещественное число - в строку.
Для строк можно выполнять операцию + , которая означает соединение строк, а также операции сравнения.
SetLength(DynArray,Length) - создает динамический массив длиной Length.
Порядок действий соответствует следующим приоритетам, расположенным от высшего - к низшему:
Логическое не. (Not)
Умножение, деление, остаток, логическое и. (* / Div Mod Or)
Сложение и вычитание, логическое или. (+ - Or)
Отношения. (< <= > >= = <>)
Кроме выше приведенных функций, имеется большая группа других функций.
Функции и процедуры работы со строками

Эти функции могут использоваться для манипуляций со строками при вводе - выводе данных, а также для формирования значений P - параметров.
Функция Pos(Substr, S) Она ищет подстроку Substr в строке S. Когда найдена нужная подстрока, то позиция начала совпадения считается результатом, иначе результат равен 0.
Функция Copy(S,Index,Count) Выделяет из строки S, начиная с позиции Index заданное число символов. Полученная строка считается результатом.
Процедура Delete(S,Index,Count); В строке S удаляется Count символов, начиная с символа с указанным номером Index. Полученная строка заменяет S.
Процедура Insert(Source,S, Index); В строку S вставляется строка Source, начиная с позиции index. Полученная строка заменяет S.
Функция Length(S) - определяет длину строки S или динамического массива.
Функция StringReplace(S,OldPattern,NewPattern,Flags) Возвращает новую строку, которая получена следующим образом. В строке S заменяется многократно подстрока OldPattern на подстроку NewPattern. Флаги, записанные в квадратных скобках, управляют заменой.

Если есть флаг rfIgnoreCase, то при замене не учитывается регистр символов, то есть не различаются большие и малые буквы. Если есть флаг rfReplaceAll, то заменяются все подстроки, иначе только первая попавшаяся подстрока.
Работа с файлами.
Эти процедуры могут рассматриваться как процедуры управления моделью.
В принципе, для работы с текстовыми файлами используются процедуры самого языка Pascal. Для этого следует описать текстовый файл как переменную в самой модели. Эта переменная должна иметь тип TextFile.
Далее следует связать эту переменную с файлом на диске и открыть файл для чтения или записи данных.
В первом случае нужно вызвать процедуры
AssignFile(FileVar,FileName);
Reset(FileVar);
А во втором случае нужно вызвать процедуры
AssignFile(FileVar,FileName);
ReWrite(FileVar);
Здесь FileVar - это файловая переменная, а FileName - это строка, содержащая имя файла на диске.
Далее для чтения данных из файла следует использовать процедуры
Read(FileVar, список переменных через запятую);
Readln(FileVar, список переменных через запятую);
Процедура Read читает данные, не обращая внимания на концы строк, а процедура Readln обязательно дочитывает начатую строку до конца, даже если в ней есть данные, которым не сопоставлены переменные.
Тип переменных может быть вещественным, целым, или строковым.
Переменные в указанном порядке будут читаться из файла, если в файле еще есть непрочитанные данные. Вещественные и целые данные должны разделяться пробелами или концами строки, а строковые - должны занимать строку целиком. В процедуре Read не следует указывать строковые переменные.
В процедуре Readln можно указывать данные любого из трех указанных типов.
Обращаю ваше внимание, что чтение данных возможно только в переменные соответствующих типов, и невозможно чтение в P - параметры.
Для проверки закончился ли файл в процессе чтения, можно использовать функцию EOF(FileVar) - которая имеет значение True (Истина), если данные в файле уже закончились.
Для записи данных в файл следует использовать процедуры
Write(FileVar, список выражений с указание ширины полей вывода через запятую);
Writeln(FileVar, список выражений с указание ширины полей вывода через запятую);
Процедура Write записывает данные в одну строку подряд, а процедура Writeln завершает запись концом строки.
Тип выражений может быть вещественным, целым, или строковым.
Выражения будут вычислены и их значения в указанном порядке будут записаны в файл. После каждого выражения может стоять двоеточие и минимальная ширина поля вывода для значения выражения. При выводе, данные могут «слипаться», если ширина поля вывода будет недостаточной для вывода значения. Поэтому рекомендуется включать в список выражений строки из одного или нескольких пробелов в апострофах, или

выбирать ширину поля вывода достаточно большой.
После того, как работа с файлом завершена, файл необходимо закрыть с помощью процедуры CloseFile(FileVar). Работа с файлами в процессе моделирования рассматривается дальше.
Если вы хотите выводить данные в отчет не с помощью ReporVal, см. ниже,
а с помощью операторов Write(FileVar, Value_List); Writeln(FileVar, Value_List);
то вместо FileVar следует указывать имя файла вывода для отчета, а именно, FileRep.

Процедуры и функции модуля GPSS_SYS

Во всех объектах, кроме таблиц, и P - параметров, имеется блок
*:ObjName.Active (True/False); при True - объект считается работающим и для него продолжается подсчет времени работы, а при False - объект считается не работающим и для него прекращается подсчет времени работы.
Функция ObjName.VA - определяет суммарное время работы объекта.
Функция ObjName.LA - определяет текущую активность объекта (1 - он активен , 0 - не активен).
|Для вызова процедур или функций любого класса следует вначале указать имя объекта класса, затем поместить точку, а затем - имя процедуры или функции. Естественно, сам объект вначале должен быть вставлен в модель. Здесь ObjName -это имя объекта.
Вместо упоминаемых в описании списков в квадратных скобках, можно использовать динамические массивы соответствующего типа.

Общие процедуры и блоки системы.

В данной системе имеется предопределенный объект Future, который служит в качестве списка будущих событий, и является наследником списка пользователя. В принципе, в модели может быть несколько списков будущих событий, но хотя бы один должен быть обязательно. Главное, чисто внешнее отличие GPSS - Future от Object GPSS состоит в несколько иной организации входного потока заявок, и в ином блоке задержки заявок.
В Object GPSS поток заявок создает процедура
{=Gen} Init(Gen,'Gen',Gen_,Mean<,Dev<,Pri<,LimCount<,CanOut);
и блок
{=Gen} ::Gen_ *:Gen.Generate(Mean<,Dev<,Pri<,LimCount<,CanOut);

Здесь и далее конструкция <,Name - означает необязательный параметр, который можно опускать.

В процедурах Gen_ - метка блока генератора, Mean и Dev - среднее значение интервала между заявками и отклонение от среднего, по умолчанию - оно равно нулю. Считается, что интервал между заявками задан с помощью равномерного закона. Pri - приоритет создаваемых заявок, по умолчанию - он равен нулю. Заявки, у которых приоритет больше - обслуживаются раньше заявок с более низким приоритетом . LimCount - предельное количество заявок, которое может выпустить генератор. По умолчанию, или по 0 - количество заявок не ограничено. CanOut - условие, которое определяет, может ли заявка выйти из генератора, когда настало её время выхода. Новая заявка планируется к выходу из генератора, только в момент выхода предыдущей заявки.
В GPSS - Future поток заявок создает процедура
{=Gen} Future.NewWaitProc(Gen,Delay,Pri,Gen,Gen+1<,Gr);
и блок

{=Gen} ::Gen *:Future.NewWait(Delay,Pri,Gen,Gen+1<,Gr<,CanOut<,LimCount<,NumBl);

Здесь Gen - метка блока генератора, Delay значение интервала между заявками. Если вам нужно задать интервал в стиле Object GPSS, используйте функцию MeanDev(Mean,Dev) в качестве Delay. Pri - приоритет создаваемых заявок, по умолчанию - он равен нулю. Заявки, у которых приоритет больше - обслуживаются раньше заявок с более низким приоритетом . Gen,Gen+1 - номера IP - параметров, в которых сохраняется количество выпущенных блоком заявок, и номер последней запланированной заявки для блока. Узнать значения этих параметров можно с помощью функций IP(Gen,Future.N) и IP(Gen+1,Future.N). В принципе, значения этих параметров могут быть и не Gen,Gen+1, важно только, чтобы они были уникальны для каждой пары NewWaitProc и NewWait.

У блока Future.NewWait могут быть дополнительные параметры. Gr, - номер группы заявок в списке, по умолчанию, и как правило, равный 0. CanOut - условие, которое определяет, может ли заявка выйти из генератора, когда настало её время выхода. По умолчанию, True. Новая заявка планируется к выходу из генератора, только в момент выхода предыдущей заявки. LimCount - предельное количество заявок, которое может выпустить генератор. По умолчанию, или по 0 - количество заявок не ограничено. NumBl - номер блока, куда уйдет запланированная заявка, по умолчанию или по 0 - это данный блок.

Параметр Gr может быть и у процедуры NewWaitProc.

Блоки

Вместо блока

*:Advance(Mean,Dev);

в GPSS - Future используется блок

*:Future.Wait(Delay<,Gr);

Блок задерживает заявку на заданное время Delay. Gr- - номер группы заявок в списке, по умолчанию, и как правило, равный 0. Если вам нужно, то используйте функцию MeanDev(Mean, Dev) в качестве Delay.

*:Terminate(<Num); Блок уничтожает активную заявку и уменьшает значение Term на значение Num. По умолчанию Num равно 0.

*:Priority(Num<,Tran); Блок задает новый приоритет Num заявке Tran, по умолчанию - активной заявке.

*:Mark(<Tran); Блок обнуляет время жизни заявки, по умолчанию - активной заявки.

*:Test([LogList],[NumBlList]); Блок перехода активной заявки на блок с меткой из NumblList, для первого выражения с истинным значением из LogList. И LogList, и NumBlList - это списки. Они записываются в квадратных скобках через запятую. Если истинных значений нет, то заявка не входит в блок.

*:Transfer(NumBl<,NumIP); Блок перехода активной заявки на блок с меткой Numbl. В параметр IP с номером NumIP - записывается номер следующего блока для возможного возврата. По умолчанию, запись номера не ведется.

*:Transfer(Ver,NumBl1,NumBl2); Блок перехода активной заявки на блок с меткой Numbl1 или Numbl2 с вероятностью Ver. Вероятность соответствует второй метке.

*:Transfer([LabelList]); Обеспечивает переход на одну из меток списка, если один из блоков с меткой примет заявку. Иначе заявка не входит в блок. Ни один из блоков с меткой не должен быть таким же Transfer. Впрочем, в этом нет смысла.

*:Transfer(NumBl<,NumIP,[NumList],[ValueList]); Блок перехода активной заявки на блок

с меткой Numbl. В параметр IP с номером NumIP - записывается номер следующего блока для возможного возврата. По умолчанию, запись номера не ведется. Списки NumList и ValueList аналогичны спискам блока PLet.
*:Loop(NumIP,NumBl); Блок организации цикла, уменьшает значение IP(NumIP) на 1. Если оно не нулевое, то заявка уходит по метке NumBl, иначе она идет дальше.
*:Buffer; Блок перемещает активную заявку в конец ее приоритетного класса.
*:Split(Num,NumBl,NumIP); Блок создает Num заявок соответствующего семейства, и отправляет их по метке NumBl. NumIP - это номер IP - параметра для последовательной нумерации созданных заявок. Если параметр опущен, то заявки не нумеруются. При опущенном параметре NumBl - заявки переходят на следующий блок. Все копии заявки имеют такой же набор P- параметров и время рождения.
*:Assemble(Num); Блок объединения заданного числа заявок одного семейства в одну заявку - инициатор.
*:Gather(Num); Блок группировки заданного числа заявок одного семейства.
*:Match(NumBl); Блок синхронизации заявок в текущем блоке, и в блоке с данной меткой.
*:Match([ListOfLabel]); Блок синхронизации заявок в текущем блоке, и в остальных блоках с данными метками.
*:Adopt(NumAssem<,Tran); Блок устанавливает новый номер семейства активной заявке, или заявке с заданным номером.
*:NewTransaction(NumBl, Pri); Блок создает заявку и отправляет их по метке NumBl. Pri - приоритет новой заявки.
*:SetTerm(Num); Блок устанавливает новое значение Term - счетчика.
*:Execute(Num); Исполняет блок с указанным номером. В косвенно указанном блоке нужно аккуратно пользоваться метками.
*:Displace(Tran,NumBl,NumRP,NumSP); Блок отправляет заявку с номером Tran - в блок с меткой NumBl. NumRP - определяет номер RP параметра, где сохраняется оставшееся время задержки. NumSP- номер SP параметра, где записывается обозначение списка, где заявка обнаружена.
Здесь "N" - это отсутствие заявки. "P" - заявка обнаружена в списке прерываний. "C"- заявка обнаружена в списке текущих событий. "F" - заявка обнаружена в списке будущих событий. "U" - заявка обнаружена в списке пользователя. "S" - заявка обнаружена в списке синхронизации. По умолчанию, соответствующая информация не записывается в P - параметры.
*:IncTran; Блок увеличивает на 1 номер последней запланированной или выпущенной заявки. Он предназначен для создания фиктивных наборов P -параметров, для которых нет соответствующей заявки. Таким образом, можно в любой момент создать новый набор P -параметров.
*:Select(Log ,Atran); Блок записывает в динамический массив (Atran) номера заявок из списка текущих событий, для которых условие - функция Log - истинна.
*:SelectSyn(Log , Atran);
*:SelectInt(Log , Atran);
Аналогичные блоки для списков синхронизации и прерываний.
*:Pause(Tran,Flag); Блок, переводит заявку Tran в паузу по умолчанию или при Flag=True, а при Flag=False - заявка вновь потенциально активна.
Функции.

TranAge(<,Tran) Выдает время жизни заявки с данным номером, по умолчанию-активной заявки.
Tran Выдает номер активной заявки.
TranSyn
TranInt
Аналогичные функции для списков синхронизации и прерываний.
Фактически, они могут быть полезны только для использования внутри функций выбора Log для функции Select.(смотри выше).
PreemptTran
PreemptTranOld
PreemptNumFac Эти функции при просмотре списка синхронизации блоком SelectInt позволяют узнать номер прерванной заявки, номер предыдущей прерванной заявки и номер устройства, где произошло это прерывание.
CurrentBlock
NextBlock Эти функции, при просмотре списка текущих событий блоком Select позволяют узнать номер текущего и следующего блоков для заявок из списка.
CurrentBlockSyn
NextBlockSyn
Аналогичные функции для списка синхронизации.
Pri(<,Tran) Выдает приоритет заявки с данным номером, по умолчанию - активной заявки.
Assem(<,Tran) Выдает номер семейства заявки с данным номером, по умолчанию - активной заявки.
PriSyn
AssemSyn
Аналогичные функции для списка синхронизации.
InterruptTran(Tran) сообщает, находится ли заявка в состоянии прерывания.
TranCurrent(Tran)
TranNext(Tran)
TranList(Tran)
Функции для данной заявки позволяют узнать номер текущего и следующего блоков, а также вид списка (Первую букву C F S U).
Clock Выдает текущее время моделирования.
AClock Выдает абсолютное текущее время моделирования.
IntervalToTime(Interval) Выдает интервал во временном формате DD:HH:MM:SS .
Term Выдает значение счетчика завершений.
Total(NumBl) Выдает число вхождений заявок в блок с данным номером.
Current(NumBl) Выдает число заявок в блоке с данным номером.
IsMatch(NumBl) Для текущего блока, определяет наличие заявки того же семейства в блоке с данным номером.
MaxTran Определяет, максимум среди номеров запланированных или выпущенных заявок.
MaxNumObj -максимальный номер объекта.
ActiveBlock Выдает номер блока, в который пытается войти заявка.
GetParam(Num) - эта функция определяет объект типа TParam по его Num= ObjName.N.
GetFacility(Num) - эта функция определяет объект типа TFacility по его Num= ObjName.N.

GetQueue(Num) - эта функция определяет объект типа TQueue по его Num= ObjName.N.
GetStorage(Num) - эта функция определяет объект типа TStorage по его Num= ObjName.N.
GetTable(Num) - эта функция определяет объект типа TTable по его Num= ObjName.N.
GetUser(Num) - эта функция определяет объект типа TUser по его Num= ObjName.N.
GetFuture(Num) - эта функция определяет объект типа TFuture по его Num= ObjName.N.
GetRGroup(Num) - эта функция определяет объект типа TRGroup по его Num= ObjName.N.
GetIGroup(Num) - эта функция определяет объект типа TIGroup по его Num= ObjName.N.
GetSGroup(Num) - эта функция определяет объект типа TSGroup по его Num= ObjName.N.
GetTGroup(Num) - эта функция определяет объект типа TTGroup по его Num= ObjName.N.
NameObj(Num)- эта функция определяет имя объекта по его Num= ObjName.N
Процедуры управления моделью.
ParamInList(Yes); определяет способ вывода в отчет значений параметров заявок. По умолчанию, они выводятся как списки параметров для всех заявок, а при Yes=True, по каждой
заявке в отдельности.
SetDistance(Dis); устанавливает предельное число блоков, для продвижения одной заявки за один раз. По умолчанию установлена величина 10000. Этот параметр защищает модель от зацикливания.
IncTranProc; Увеличивает на 1 номер последней запланированной или выпущенной заявки.
DisplaceProc(Tran,NumBl,NumRP,NumSP);
SelectProc(YesF,ATran);
SelectSynProc(YesF, ATran);
SelectIntProc(YesF, ATran);
Процедуры, аналогичные блокам
*:Displace(Tran,NumBl,NumRP,NumSP);
*:Select(YesF, ATran);
*:SelectSyn(YesF, ATran);
*:SelectInt (YesF, ATran);

Работа_с_P_параметрами

P - параметры - это, в первую очередь, параметры заявок. Они могут быть вещественного - RP(Num<,Tran) , целого -IP(Num<,Tran), строкового - SP(Num<,Tran) и логического типов BP(Num<,Tran). Здесь Num - это номер параметра. Если последний элемент данных (Tran) опущен, то параметр относится к активной заявке. Если же вместо Tran указано ObjName.N - то это P - параметр объекта.

Для работы с P - параметрами вместо номера можно использовать переменные, типа Integer или TPname, которым присвоено значение функции Init(St), где St - строка для идентификации параметра.

*:RPLet([NumList],[Valuelist]<,Tran); Блок присваивания списку RP- параметров значений из списка. Списки указываются в квадратных скобках. Присваивание ведется

для указанной заявки, по умолчанию - для активной заявки.
*:IPLet([NumList],[Valuelist]<,Tran); Блок присваивания списку IP- параметров значений из списка. Списки указываются в квадратных скобках. Присваивание ведется для указанной заявки, по умолчанию - для активной заявки.
*:SPLet([NumList],[Valuelist]<,Tran); Блок присваивания списку SP- параметров значений из списка. Списки указываются в квадратных скобках. Присваивание ведется для указанной заявки, по умолчанию - для активной заявки.
*:BPLet([NumList],[Valuelist]<,Tran); Блок присваивания списку BP- параметров значений из списка. Списки указываются в квадратных скобках. Присваивание ведется для указанной заявки, по умолчанию - для активной заявки.
*:RPAdd([NumList],[Valuelist]<,Tran); Блок добавления списку RP- параметров значений из списка. Списки указываются в квадратных скобках. Присваивание ведется для указанной заявки, по умолчанию - для активной заявки.
*:IPAdd([NumList],[Valuelist]<,Tran); Блок добавления списку IP- параметров значений из списка. Списки указываются в квадратных скобках. Присваивание ведется для указанной заявки, по умолчанию - для активной заявки.
*:PLet([Numlist],[Valuelist]<,Tran); Блок задает набор значений для P параметров списков. Списки указываются в квадратных скобках. Тип параметра определяется типом выражения.
Вместо первого списка (NumList) можно указать нижнее значение для номеров, запоминаемых значений.
*:RAssign(Num,Value<,Tran);
*:IAssign(Num,Value<,Tran);
*:SAssign(Num,Value<,Tran);
*:BAssign(Num,Value<,Tran);
Блоки присваивания RP(Num) IP(Num) SP(Num) BP(Num) - значения Value.
*:PSortNum(ANum, AVal<,NumB<,NumE<,Tran); Блок копирует значения P - параметров и их номеров в динамические массивы AVal и ANum, упорядоченные по номерам параметров. Здесь не обязательные параметры NumB, NumE, Tran - определяют диапазон номеров и номер самой заявки. Тип параметров определяется массивом AVal. Если вместо Tran указать Obj.N - то блок работает для P- параметров объекта.
*:RPDelete([NumList]<,Tran); Блок удаляет RP - параметры из списка для данной заявки. По умолчанию - для активной заявки. Список указывается в квадратных скобках. Вместо списка можно указать верхнее и нижнее значение для номеров, удаляемых значений.
*:RPDelete([NumList],Obj.N); Блок удаляет RP - параметры из списка для данного объекта. Список указывается в квадратных скобках. Вместо списка можно указать верхнее и нижнее значение для номеров, удаляемых значений.
*:IPDelete([NumList]<,Tran); Блок удаляет IP - параметры из списка для данной заявки. По умолчанию - для активной заявки. Список указывается в квадратных скобках. Вместо списка можно указать верхнее и нижнее значение для номеров, удаляемых значений.
*:IPDelete([NumList],Obj.N); Блок удаляет IP - параметры из списка для данного объекта. Список указывается в квадратных скобках. Вместо списка можно указать верхнее и нижнее значение для номеров, удаляемых значений.
*:SPDelete([NumList]<,Tran); Блок удаляет SP - параметры из списка для данной заявки. По умолчанию - для активной заявки. Список указывается в квадратных скобках. Вместо списка можно указать верхнее и нижнее значение для номеров, удаляемых значений.

*:SPDelete([NumList],Obj.N); Блок удаляет SP - параметры из списка для данного объекта. Список указывается в квадратных скобках. Вместо списка можно указать верхнее и нижнее значение для номеров, удаляемых значений.
*:BPDelete([NumList]<,Tran); Блок удаляет BP - параметры из списка для данной заявки. По умолчанию - для активной заявки. Список указывается в квадратных скобках. Вместо списка можно указать верхнее и нижнее значение для номеров, удаляемых значений.
*:BPDelete([NumList],Obj.N); Блок удаляет BP - параметры из списка для данного объекта. Список указывается в квадратных скобках. Вместо списка можно указать верхнее и нижнее значение для номеров, удаляемых значений.
*:RPDelete(<Tran); Блок удаляет все RP -параметры данной заявки. По умолчанию - для активной заявки.
*:RPDelete(Obj.N); Блок удаляет все RP -параметры данного объекта.
*:IPDelete(<Tran); Блок удаляет все IP -параметры данной заявки. По умолчанию - для активной заявки.
*:IPDelete(Obj.N); Блок удаляет все IP -параметры данного объекта.
*:SPDelete(<Tran); Блок удаляет все SP -параметры данной заявки. По умолчанию - для активной заявки.
*:SPDelete(Obj.N); Блок удаляет все SP -параметры данного объекта.
*:BPDelete(<Tran); Блок удаляет все BP -параметры данной заявки. По умолчанию - для активной заявки.
*:BPDelete(Obj.N); Блок удаляет все BP -параметры данного объекта.
*:PDelete(<Tran); Блок удаляет все P -параметры данной заявки. По умолчанию - для активной заявки.
*:PDelete(Obj.N); Блок удаляет все P -параметры данного объекта.
Функции
RPExist(Num<,Tran) Определяет, имеется ли RP(Num) для данной заявки Tran, по умолчанию, используется активная заявка. Если здесь и далее вместо Tran указано ObjName.N, то функция относится к P - параметрам объекта.
IPExist(Num<,Tran) Определяет, имеется ли IP(Num) для данной заявки Tran, по умолчанию, используется активная заявка.
SPExist(Num<,Tran) Определяет, имеется ли SP(Num) для данной заявки Tran, по умолчанию, используется активная заявка.
BPExist(Num<,Tran) Определяет, имеется ли BP(Num) для данной заявки Tran, по умолчанию, используется активная заявка.
RPCount(<Tran) Определяет количество RP - параметров у заявки с данным номером. По умолчанию - у активной заявки.
IPCount(<Tran) Определяет количество IP - параметров у заявки с данным номером. По умолчанию - у активной заявки.
SPCount(<Tran) Определяет количество SP - параметров у заявки с данным номером. По умолчанию - у активной заявки.
BPCount(<Tran) Определяет количество BP - параметров у заявки с данным номером. По умолчанию - у активной заявки.
Процедуры
RPLetProc ([Num List],[Value list],Tran);
IPLetProc([Num List],[Value list],Tran);
RPAddProc ([Num List],[Value list],Tran);

IPLetAdd([Num List],[Value list],Tran);
IPAddProc([Num List],[Value list],Tran);
SPLetProc([Num List],[Value list],Tran);
BPLetProc([Num List],[Value list],Tran);
PLetProc([Numlist],[Valuelist],Tran);
RAssignProc(Num,Value<,Tran);
IAssignProc(Num,Value<,Tran);
SAssignProc(Num,Value<,Tran);
BAssignProc(Num,Value<,Tran);
PSortNumProc (ANum, AVal,NumB,NumE,Tran);
RPDeleteProc ([Num List],Tran);
IPDeleteProc ([Num List],Tran);
SPDeleteProc ([Num List],Tran);
BPDeleteProc ([Num List],Tran);
RPDeleteProc(Tran);
IPDeleteProc(Tran);
SPDeleteProc(Tran);
BPDeleteProc(Tran);
PDeleteProc(Tran);
Эти процедуры, аналогичные блокам
*:RPLet ([Num List],[Value list],Tran);
*:IPLet([Num List],[Value list],Tran);
*:RPAdd ([Num List],[Value list],Tran);
*:IPAdd([Num List],[Value list],Tran);
*:SPLet([Num List],[Value list],Tran);
*:BPLet([Num List],[Value list],Tran);
*:PLet([Numlist],[Valuelist],Tran);
*:RAssign(Num,Value<,Tran);
*:IAssign(Num,Value<,Tran);
*:SAssign(Num,Value<,Tran);
*:BAssign(Num,Value<,Tran);
*:PSortNum (ANum, AVal,NumB,NumE,Tran);
*:RPDelete ([Num List],Tran);
*:IPDelete ([Num List],Tran);
*:SPDelete ([Num List],Tran);
*:BPDelete ([Num List],Tran);
*:RPDelete(Tran);
*:IPDelete(Tran);
*:SPDelete(Tran);
*:BPDelete(Tran);
*:PDelete(Tran);

Работа_с_переменными_и_ввод_вывод

*:Let(Varia<,Value); Блок присваивает вещественное, целое, строковое, логическое

значение Value - переменной Varia. Если Value опущено, то оно равно пустому значению.
*:Add(Varia<,Value); Блок увеличивает вещественное или целое значение Varia на величину Value. Если Value опущено, то оно равно 1.
*:OpenGetBlock(Filevar,FileName); Блок открывает файл Filevar для чтения. Имя файла - FileName.
*:OpenPutBlock(Filevar,FileName); Блок открывает файл Filevar для записи. Имя файла - FileName.
*:CloseBlock(Filevar); Блок закрывает файл Filevar.
*:WriteBlock(Filevar,Value,Leng); Блок записывает вещественное, целое, или строковое Value в файл Filevar. Leng - длина поля вывода, по умолчанию - 0.
*:WritelnBlock(Filevar,Value,Leng); Блок записывает вещественное, целое, или строковое значение Value с концом строки в файл Filevar. Leng - длина поля вывода, по умолчанию - 0.
*:WritelnBlock(Filevar); Блок записывает конец строки в файл Filevar.
*:ReadBlock(Filevar,Value); Блок читает вещественное или целое значение Value из файла Filevar.
*:ReadlnBlock(Filevar,Value); Блок читает вещественное или целое значение Value с концом строки из файла Filevar.
*:ReadlnBlock(Filevar,Value); Блок читает строку Value с концом строки из файла Filevar.
*:ReadlnBlock(Filevar); Блок читает конец строки из файла Filevar.

Таблицы

Класс - TTable. Блок.
*:ObjName.Tabulate(Value,ANum); Блок выполняет табулирование величины, Value с кратностью ANum. По умолчанию параметр Anum равен 1.

Функции.
ObjName.C Определяет число вхождений в таблицу с учетом кратности.
ObjName.A Определяет среднее значение табулируемой величины с учетом кратности.
ObjName.D Определяет среднеквадратичное отклонение табулируемой величины от средней.

Процедуры управления моделью.
ObjName. GetXYI(AX,AY,AI); записывает таблицу в три динамических массивов.
ObjName. TabulateProc(Value,ANum); Выполняет табулирование величины, Value с кратностью ANum.
Show(ObjName,ANum); Выводит таблицу в виде гистограммы. В ней указываются номер графика.
GShow(ObjName,ANum); Выводит таблицу в виде графика. В остальном подобен Show.
IShow(Tab, ANum); Выводит таблицу в виде интегрального графика. В остальном подобен Show.
ObjName.ReportFile(FileName); Выводит отчет по таблице. FileName - имя файла.
ObjName.Change(Abeg,Astep,Anum); процедура аналогична init, но используется для переопределения таблицы.

Пользование этой процедурой нужно предоставить системе.
Init(ObjName,' ObjName', Abeg, Astep, ANum); Создает таблицу и определяет ее -Начало, шаг и количество интервалов. Для этого объекта отчет не создается, если текст в апострофах - пуст.

Одноканальные_Устройства

Класс - TFacility. Блоки.
*:ObjName.Seize; Блок занятия устройства.
*:ObjName.Release; Блок освобождения устройства. Даже если у устройства есть прерванная заявка, она устройство не получит.
*:ObjName.Extract(NumBl<,NumRP<,NumSP); Блок удаляет обслуживаемую заявку из устройства. NumBl - номер или метка блока, куда удаляется заявка. RP(NumRP) - параметра запоминает оставшееся время задержки заявки. Устройство остается незанятым, и, если были прерванные заявки, то они его не получают. Поэтому после выполнения этого блока возможно занятие устройства по Seize или Preempt. Возможно также выполнение Return0, для передачи его прерванной ранее заявке. Если предпоследний параметр опущен, то оставшееся время задержки заявки не запоминается. Если указан последний параметр, то в нем записывается признак списка, откуда удаляется заявка. ('C' 'F ' 'S' 'U').
*:ObjName.ExtractInterrupt(NumBl,NumRP); Блок удаляет прерванные заявки из устройства. NumBl - номер или метка блока, куда удаляются заявки. RP(NumRP) - определяет время для завершения обслуживания заявки. Если последний параметр опущен, то оставшееся время задержки заявки не запоминается.
*:ObjName.Preempt; Блок прерывает обслуживание заявки на устройстве. Активная заявка занимает устройство по прерыванию.
*:ObjName.Preempt0; Блок прерывает обслуживание заявки на устройстве. Устройство остается незанятым. Используется для моделирования недоступности.
*:ObjName.Return; Блок освобождает устройство, занятое по прерыванию . Если есть прерванная на устройстве заявка, то она получат устройство.
*:ObjName.Return0; Блок возвращает для обслуживания в устройстве, прерванную заявку. Это происходит после того, как устройство было оставлено незанятым. Используется для моделирования возврата из недоступности.
*:ObjName.SetGoto(NumBl); Блок устанавливает номер блока, куда, возможно, будут отсылаться активные заявки. Положительное значение NumBl -определяет номер блока, отрицательное - запрещает вход в устройство. По умолчанию - восстанавливается возможность входа в устройство.
|R
Функции.
ObjName.A Определяет коэффициент занятости устройства.
ObjName.V Определяет коэффициент временной интеграл устройства.
ObjName.T Определяет среднее время занятости устройства одной заявкой.
ObjName.L Определяет занятость устройства. 1 - устройство занято, 0 - свободно.
ObjName.C Определяет число заявок, вошедших в устройство.
ObjName.Tran Определяет номер обслуживаемой заявки для устройства.

ObjName. InterruptTran Определяет номер прерванной заявки для устройства.
ObjName.IsPree Определяет состояние прерывания устройства. True - если прервано, иначе False.
ObjName.Go Определяет значение параметра, установленного SetGoto.

Процедура управления моделью.
ObjName.ReportFile(FileName); Выводит отчет по устройству. FileName - имя файла.
Пользование этой процедурой нужно предоставить системе.
Init(ObjName,' ObjName'); Создает устройство. Для этого объекта отчет не создается, если текст в апострофах - пуст.

Очереди

Класс - TQueue. Блоки.
*:ObjName.Queue(Units); Блок занятия очереди. В ней указывается число занимаемых единиц. По умолчанию 1.
*:ObjName.Depart(Units); Блок освобождения очереди. В ней указывается число освобождаемых единиц. По умолчанию 1.

Функции.
ObjName.A Определяет среднюю величину очереди.
ObjName.T Определяет среднее время в очереди для одной заявки.
ObjName.X Определяет среднее время в очереди для одной заявки. Оно определяется без учета заявок, в очереди не задержавшихся.
ObjName.C Определяет число вхождений в очередь, с учетом кратности.
ObjName.L Определяет текущую длину очереди с учетом кратности.
ObjName.M Определяет текущую максимальную длину очереди с учетом кратности.
ObjName.V Определяет временной интеграл очереди с учетом кратности.
ObjName.Z Определяет текущее число вхождений в очередь с учетом кратности. Оно определяется для заявок, в очереди не задержавшихся.

Процедура управления моделью.
ObjName.ReportFile(FileName); Создает отчет по очереди. FileName - имя файла.
Пользование этой процедурой нужно предоставить системе.
Init(ObjName,' ObjName'); Создает очередь. Для этого объекта отчет не создается, если текст в апострофах - пуст.

Накопители

Класс - TStorage. Блоки.
*:ObjName.Enter(Units); Блок определяет возможность входа в устройство. Units - требуемое число занимаемых единиц емкости. По умолчанию Units равно 1.
*:ObjName.Leave(Units); Блок обеспечивает выход из устройства. Units - количество освобождаемых единиц емкости. По умолчанию Units равно 1.
*:ObjName.SetStorage(Units); Блок устанавливает новую предельную емкость устройства.
*:ObjName.SetGoto(NumBl); Блок устанавливает номер блока, куда, возможно, будут

отсылаться активные заявки. Положительное значение NumBl -определяет номер блока, отрицательное - запрещает вход в устройство. По умолчанию - восстанавливается возможность входа в устройство.

Функции.
ObjName.A Определяет среднее число занятых каналов.
ObjName.AM Определяет среднее предельное число занятых каналов.
ObjName.V Определяет временной интеграл.
ObjName.VM Определяет временной интеграл предельного числа занятых каналов.
ObjName.T Определяет среднее время, проведенное заявкой в устройстве.
ObjName.X Определяет среднее время, проведенное заявкой в устройстве, для заявок, в устройстве задержавшихся.
ObjName.C Определяет общее число занятий каналов.
ObjName.L Определяет текущее число занятых каналов.
ObjName.M Определяет максимальное число занятых каналов.
ObjName.Z Определяет общее число занятий каналов, при которых заявки в устройстве не задерживались.
ObjName.R Определяет текущее число свободных каналов.
|R
ObjName.U Определяет среднюю занятость в расчете на 1 канал.
ObjName.I Выдает минимальную занятость устройства.
ObjName.Go Выдает значение, установленное по SetGoto.

Процедуры управления моделью.
ObjName.Change(Units); Процедура аналогична Init, но используется для переопределения многоканального устройства.
ObjName.ReportFile(FileName); Выводит отчет по устройству. FileName - имя файла. Пользование этой процедурой нужно предоставить системе.
Init(ObjName,' ObjName',Lim); Создает устройство с заданным числом каналов. Для этого объекта отчет не создается, если текст в апострофах - пуст.

Списки_пользователя

Класс - Tuser. Блоки.
*: ObjName.Link (TestGo,NumGo,Value<,Gr1);
*:ObjName.Link(Value<,Gr1); Если TestGo - истина, то текущая заявка уходит на блок с меткой NumGo. Иначе блок помещает текущую заявку в список. Выражение Value используется для упорядочения списка. Не обязательный целый параметр Gr1 также используется для упорядочения, причем он учитывается до Value. Это дает дополнительные возможности для формирования списка пользователя.
*:ObjName. UnLink(NumBl, Num, Log, Cont<, IPCount<, NewAClock<, NumGr<, NumVal <, NumCurr<, NumNext); Блок удаляет заявки из списка. Здесь Numbl - метка, куда уйдут удаляемые заявки. Если Numbl=0 то заявка уйдет на тот блок, который следовал после блока Link, который поставил заявку в список пользователя. Num- максимальное число удаляемых заявок. При Num=0 удаляются все заявки. При Num>0 заявки удаляются из начала списка, при Num<0 - с конца. Удалению подлежат только те заявки, для

которых значение функции Log - истинно. Здесь Log - это отдельно описанная функция без параметров с результатом логического типа. Удаление прекращается, когда функция Cont окажется ложной. Количество реально удаленных заявок, записывается в IP(IpCount) активной заявки. Если опущен параметр IPCount, то количество реально удаленных заявок никуда не записывается. NewAClock - как правило, должен быть равен 0, так как он влияет на текущее время моделирования. В удаленной из списка заявке в IP(NumGr) и RP(NumVal) может записываться значение функций Gr и Val для этой заявки, а в IP(NumCurr) и IP(NumNext) - может записываться прежний текущий и следующий блок для удаленной заявки.
*: ObjName. UnLink(NumBl, Num, Gr1, Log, Cont<, IPCount<, NewAClock<, NumGr<, NumVal
<, NumCurr<, NumNext); Блок аналогичен предыдущему, но в дополнение учитывается требование, чтобы дополнительный параметр Gr был равен Gr1. Блок всегда работает быстрее, чем предыдущая версия.
*:ObjName.Select(AAXn1,Num, Log,Cont) ;
*:ObjName.Select(AAXn1,Num, Gr1, Log,Cont) ; Блок записывает в динамический массив AAXn1 номера заявок из списка пользователя. Смысл параметров Num, Gr1, Log, Cont - такой же, как в UnLink.

Функции.
ObjName.A Определяет среднее число заявок в списке.
ObjName.V Определяет временной интеграл списка.
ObjName.T Определяет среднее время, проведенное заявкой в списке.
ObjName.C Определяет общее число занятий списка.
ObjName.M Определяет максимальное число заявок в списке.
ObjName.L Выдает текущее число заявок в списке.
ObjName.I Выдает минимальное число заявок в списке.
ObjName.Assem Выдает номер семейства для заявки из списка.
ObjName.TranAge Выдает время жизни для заявки из списка.
ObjName.Pri Выдает значение приоритета для заявки из списка.
ObjName.Tran Выдает значение номера заявки для заявки из списка.
ObjName.ScanNum Выдает порядковый номер заявки в списке в процессе просмотра с помощью UnLink. В начале просмотра значение функции равно 0.
ObjName.CurrentBlock
ObjName NextBlock Выдает значение текущего и следующего блоков для заявки из списка.
ObjName.DelCount определяет количество удаленных или выбранных заявок, как результат функций группы Sum, процедур SelectProc, UnlinkProc и блоков Select, Unlink.
ObjName.RP(Num) Выдает значение RP(Num) для заявки из списка.
ObjName.IP(Num) Выдает значение IP(Num) для заявки из списка.
ObjName.SP(Num) Выдает значение SP(Num) для заявки из списка.
ObjName.BP(Num) Выдает значение BP(Num) для заявки из списка.
ObjName.Gr Определяет значение AddValue установленное для заявки из списка блоком Link.
ObjName.Val Выдает значение параметра упорядочения заявок для заявки из списка.
ObjName.Sum(Ssum, Num, Log, Cont)

ObjName.Sum(Ssum, Num, Gr1, Log, Cont)
Определяет значение суммы значений функции SSUM для заявок из списка. Смысл параметров Num, Gr1, Log, Cont - такой же, как в UnLink.
Функции типа Sum рекомендуется вычислять заранее и записывать в RP - параметры.
ObjName. Avg (Ssum, Num, Log, Cont)
ObjName. Avg (Ssum, Num, Gr1, Log, Cont) функция аналогична предыдущей функции, но вычисляет среднее значение.
ObjName. Min (Ssum, Num, Log, Cont)
ObjName. Min (Ssum, Num, Gr1, Log, Cont) функция аналогична предыдущей функции, но вычисляет минимальное значение.
ObjName. Max (Ssum, Num, Log, Cont)
ObjName. Max (Ssum, Num, Gr1, Log, Cont) функция аналогична предыдущей функции, но вычисляет максимальное значение.
ObjName. Count (Num, Log, Cont)
ObjName. Count (Num, Gr1, Log, Cont) функция аналогична предыдущей функции, но вычисляет количество заявок.
ObjName. MinVal
ObjName. MaxVal Функции вычисляют минимальное и максимальное значения Val в списке.
Процедуры управления моделью.
ObjName. UnLinkProc (NumBl, Num, Gr1, Log, Cont<, NewAClock<, NumGr<, NumVal
<, NumCurr<, NumNext);
ObjName. UnLinkProc (NumBl, Num, Log, Cont<, NewAClock<, NumGr<, NumVal
<, NumCurr<, NumNext);
Процедуры аналогичны блоку UnLink, но параметр IPCount - отсустствует. Вместо этого можно использовать функцию ObjName.DelCount. Будьте внимательны!!!
ObjName. SelectProc (AAXn1,Num, Log,Cont) ;
ObjName. SelectProc (AAXn1,Num, Gr1, Log,Cont) ;
Процедуры аналогичны блоку Select.
ObjName.ReportFile(FileName); Выводит отчет по списку. FileName - имя файла.
Пользование этой процедурой нужно предоставить системе.
Init(ObjName,' ObjName'); Создает список. Для этого объекта отчет не создается, если текст в апострофах - пуст.

Списки_будущего

Класс - TFuture. Блоки.
Этот класс наследует все возможности класса Tuser, и имеет следующие дополнительные блоки и процедуры.
Уже упомянутые ранее процедуру
NameObj.NewWaitProc(Gen,Delay,PR,TranCount,LastTran<,Gr);
и блоки
*: NameObj.NewWait(Delay,PR,TranCount,LastTran<,Gr<, CanOut<, LimCount <, NumBl);
*: NameObj.Wait(Delay<,Gr););
А также блок
*: NameObj.Activate (NumBl, Num, Gr1, Log, Cont <,IPCount <,NewAClock <,NumGr

<,NumVal <, NumCurr <, NumNext);
*: NameObj.Activate (NumBl, Num, Log, Cont <,IPCount <,NewAClock <,NumGr <,NumVal <, NumCurr <, NumNext);
Смысл этих параметров такой же, как у блока UnLink, а параметр NewAClock управляет системными часами и временем выхода заявок в список текущих событий. Если NewAClock>0 то это будет новое значение системных часов и временем выхода заявок. Если NewAClock<0 то новое значение системных часов и время выхода заявок будет равным параметру Val списка будущих событий. Если NewAClock=0 то время выхода заявок будет равным текущему времени системных часов. В удаленной из списка заявке, в отличие от блока UnLink, в RP(NumVal) может записываться значение выражения Val - Aclock, то есть, оставшееся время задержки заявки.
Процедура
UnLinkProc(NewAClock <, NumBl<, <,NumGr<,NumVal<, NumCurr<, NumNext);
Процедура извлекает все заявки, для которых Val<= NewAClock, в остальном она аналогична предыдущей процедуре.

Группы

Класс - TRGroup. Блоки.
*:ObjName.Join([ValueList]); Блок определяет вещественные числа, добавляемые к группе.
*:ObjName.Remove(Value<,NumBl); Блок определяет вещественное число, удаляемое из группы. Если нужного числа в группе нет, то заявка уходит по метке NumBl. Если метка опущена или равна 0, то заявка всегда переходит в следующий блок.
*:ObjName.RemoveAll; - удаляет все числа.
*:ObjName.Examine(Value,NumBl); Блок проверяет наличие в группе заданного числа. Если его в группе нет, то заявка уходит по указанной метке.
*:ObjName.Copy(Values); Копирует данные из группы в динамический массив.
Функции.
ObjName.A Определяет среднюю величину числовой группы.
ObjName.C Определяет число вхождений в группу.
ObjName.L Определяет текущую длину группы.
ObjName.V Определяет временной интеграл длину группы.
ObjName.M Определяет текущую максимальную длину группы.
ObjName.I Определяет текущую минимальную длину группы.
ObjName.T Определяет среднее время, проведенное элементом в группе.
ObjName.Exist(Value); Проверяет наличие в группе заданного числа.

Процедуры управления моделью.
ObjName.JoinProc([ValueList]); Определяет вещественные числа, добавляемые к группе.
ObjName.CopyProc(Values); Копирует данные из группы в динамический массив.
ObjName.RemoveAllProc; - удаляет все числа.
ObjName.ReportFile(FileName); Выводит отчет по группе. FileName - имя файла.
ObjName.GroupReportFile(FileName); Создает полный отчет по членам группы. FileName - имя файла.
Пользование этой процедурой нужно предоставить системе.

Init(ObjName,' ObjName'); Создает числовую группу из вещественных чисел. Для этого объекта отчет не создается, если текст в апострофах - пуст.

Класс - TIGroup. Блоки.

Это класс группы целых чисел. В нем все параметры, процедуры и функции аналогичны таким же в группе вещественных чисел.

*:ObjName.Join([ValueList]);
*:ObjName.Remove(Value,NumBl);
*:ObjName.RemoveAll;
*:ObjName.Examine(Value,NumBl);
*:ObjName.Copy (Values)

Функции.
ObjName.A
ObjName.C
ObjName.L
ObjName.M
ObjName.V
ObjName.I
ObjName.T
ObjName.Exist(Value);

Процедуры управления моделью.
ObjName.JoinProc([ValueList]);
ObjName.CopyProc (Values);
ObjName.RemoveAllProc;
ObjName.ReportFile(FileName);
ObjName.GroupReportFile(FileName);
Пользование этой процедурой нужно предоставить системе.
Init(ObjName,' ObjName');

Класс - TSGroup. Блоки.

Это класс группы строк. В нем все параметры, процедуры и функции аналогичны таким же в группе вещественных чисел.

*:ObjName.Join([ValueList]);
*:ObjName.Remove(Value,NumBl);
*:ObjName.RemoveAll;
*:ObjName.Examine(Value,NumBl);
*:ObjName.Copy(Values)

Функции.
ObjName.A
ObjName.C
ObjName.L
ObjName.M
ObjName.V
ObjName.I
ObjName.T

ObjName.Exist(Value);

Процедуры управления моделью.
ObjName.JoinProc([ValueList]);
ObjName.CopyProc(Values);
ObjName.RemoveAllProc;
ObjName.ReportFile(FileName);
ObjName.GroupReportFile(FileName);
Пользование этой процедурой нужно предоставить системе.
Init(ObjName,' ObjName');

Группы_заявок

Класс - TTGroup. Блоки.
*:ObjName.Join(<Tran) ; Блок - добавляет заданную заявку в группу, по умолчанию, он добавляет активную заявку.
*:ObjName.Remove(Num,Del<,NumIP); Блок удаляет до Num заявок из группы. Если 0, то потенциально могут быть удалены все заявки. Del - отдельно описанная логическая функция без параметров, которая определяет заявки, подлежащие удалению. В IP(NumIP), возможно, запишется число реально удаленных заявок.
*:ObjName.Remove(<Tran); Блок удаляет заявку с номером Tran. По умолчанию - удаляется из группы активная заявка.
*:ObjName.RemoveAll - удаляет все заявки.
*:ObjName.Examine(NumBl<,Tran); Блок проверяет наличие заданной заявки в группе. По умолчанию, проверяется наличие активной заявки. Если ее там нет, то активная заявка переходит на указанный блок.
*:ObjName.Scan(YesF,ValueF,NumRP<,NumBl); Блок отыскивает заявки, удовлетворяющие отдельно описанной логической функции YesF. Для первой попавшейся такой заявки, вычисляется значение вещественной функции ValueF. Оно записывается в RP(NumRP) параметр активной заявки. Если нужной заявки не найдется, то заявка уходит по метке NumBl, если этот параметр есть.
*:ObjName.Scan(YesF,ValueF,NumIP<,NumBl); Блок аналогичен предыдущему, но вычисляет значение целой функции. Запись происходит в IP параметр.
*:ObjName.Scan(YesF,ValueF,NumSP<,NumBl); Блок аналогичен предыдущему, но вычисляет значение строковой функции. Запись происходит в SP параметр.
*:ObjName.Scan(YesF,ValueF,NumBP<,NumBl); Блок аналогичен предыдущему, но вычисляет значение логической функции. Запись происходит в BP параметр.
*:ObjName.Alter(YesF,ValueF,NumRP<,Num<,NumIPCount); Блок - вычисляется логическая функция YesF для определения подходящих заявок. Для первых попавшихся подходящих заявок вычисляется значение вещественной функции ValueF. Оно записывается в RP(NumRP). При NumRP=-1 - происходит изменение времени рождения на текущий момент времени. Всего модифицируется не более Num заявок из группы, или все. Реальное число модифицированных заявок записывается в IP(NumIPCount). Два последних параметра могут быть опущены. Тогда модифицируются все заявки (значение Num=0).
*:ObjName.Alter(YesF,ValueF,NumIP<,Num<,NumIPCount); Блок аналогичен

предыдущему, но вычисляется значение целой функции. Оно записывается в IP(NumIP) При NumIP=-1 - изменение приоритета.
*:ObjName.Alter(YesF,ValueF,NumSP<,Num<,NumIPCount); Блок аналогичен предыдущему, но вычисляется значение строковой функции. Оно записывается в SP(NumSP).
*:ObjName.Alter(YesF,ValueF,NumBP<,Num<,NumIPCount); Блок аналогичен предыдущему, но вычисляется значение логической функции.Оно записывается в BP(NumBP).
*:ObjName.Select(Log ,ANum); Блок записывает в динамический массив номера заявок из группы, для которых условие - функция Log - истинна.
Функции.
ObjName.C Определяет число вхождений в группу.
ObjName.L Определяет текущую длину группы.
ObjName.A Определяет среднее число заявок в группе.
ObjName.V Определяет временной интеграл.
ObjName.T Определяет среднее время, проведенное заявкой в группе.
ObjName.M Определяет максимальное число заявок в группе.
ObjName.I Выдает минимальное число заявок в группе.
ObjName.Assem Определяет семейство заявки из группы.
ObjName.TranAge Определяет время жизни заявки из группы.
ObjName.Pri Определяет значение приоритета для заявки из группы.
ObjName.RP(Num) Определяет значение RP(Num) для заявки из группы.
ObjName.IP(Num) Определяет значение IP(Num) для заявки из группы.
ObjName.BP(Num)Определяет значение BP(Num) для заявки из группы.
ObjName.SP(Num) Определяет значение SP(Num) для заявки из группы.
ObjName.Tran Определяет номер заявки для заявки из группы.
ObjName.ScanNum Выдает порядковый номер заявки в списке в процессе просмотра с помощью процедур группы Scan и Alter. В начале просмотра значение функции равно 0.
ObjName.Sum(Ssum,Log) Определяет значение суммы значений функции Ssum для заявок из группы. Учитываются только заявки, удовлетворяющие условию - функции Log. Последний параметр может быть опущен. Тогда учитываются все заявки. Функции группы Sum,
рекомендуется вычислять заранее и записывать в RP - параметр.
ObjName.Avg(Ssum,Log) функция аналогична предыдущей функции, но вычисляет среднее значение.
ObjName.Min(Ssum,Log) функция аналогична предыдущей функции, но вычисляет минимальное значение.
ObjName.Max(Ssum,Log) функция аналогична предыдущей функции, но вычисляет максимальное значение.
ObjName.Count(Log) функция аналогична предыдущей функции, но вычисляет количество заявок.
ObjName.Exist(Value); Проверяет наличие в группе заданного числа, по умолчанию - номера активной заявки .

Процедуры управления моделью.
ObjName.RemoveProc(Num,Del,NumIP); Удаляет до Num заявок из группы. Если 0, то

потенциально могут быть удалены все заявки. Del - отдельно описанная логическая функция без параметров, которая определяет заявки, подлежащие удалению. В IP(NumIP) запишется число реально удаленных заявок.
ObjName.RemoveProc(Axn1); Удаляет заявку с номером Axn1. По умолчанию -удаляется из группы активная заявка.
ObjName.RemoveAllProc - удаляет все заявки.
ObjName.SelectProc(Log ,ATran); записывает в динамический массив номера заявок из группы, для которых условие - функция Log - истинна.
Delete удаляет все заявки из группы.
ObjName.ReportFile(FileName); Выводит отчет по группе. FileName - имя файла.
ObjName.GroupReportFile(FileName); Создает полный отчет по заявкам группы. FileName - имя файла.
Пользование этой процедурой нужно предоставить системе.
Init(ObjName,' ObjName'); Создает группу заявок. Для этого объекта отчет не создается, если текст в апострофах - пуст.

Функции_системы

Общие функции и процедуры. Функции.
MeanDev(Mean, Dev) Выдает вещественную случайную величину, распределенную по равномерному закону в интервале от Mean - Dev до Mean + Dev.
Uniform(VMin,VMax) Выдает вещественную случайную величину, распределенную по равномерному закону в интервале от VMin до VMax.
DUniform(VMin,VMax) Выдает целую случайную величину, распределенную по равномерному закону в интервале от VMin до Vmax включительно.
Exponential(Mean) Выдает случайную величину, распределенную по экспоненциальному закону со средним значением mean.
Normal(Mean,Dev) Выдает случайную величину, распределенную по нормальному закону. Ее среднее значение mean и среднеквадратичное отклонение dev.
Binomial(TrialCount,Probabilityt) Выдает случайную величину целого типа, распределенную по биномиальному закону. Ее степень - TrialCount, а Probabilityt - вероятность положительного исхода при испытании.
Geometric(Probabilityt) Выдает случайную величину целого типа, распределенную по геометрическому закону. Probabilityt - вероятность положительного исхода при испытании.
Negbinom(SuccessCount,Probabilityt) Выдает случайную величину целого типа, распределенную по негативному биномиальному закону. Ее счетчик положительных исходов - SuccessCount, а Probabilityt - вероятность положительного исхода при испытании.
Triangular(Min,Max,Mode) Выдает случайную величину, распределенную по треугольному закону. Min,Max,Mode - это минимальное, максимальное, и максимально-вероятное значение в треугольном распределении.
RByBool(Log,Values) Выдает значение вещественного выражения из списка Values, для первого истинного выражения из списка Log. Элементы каждого списка записываются в квадратных скобках через запятую.
IByBool(Log,Values) Выдает значение целого выражения из списка Values, для первого

истинного выражения из списка Log. Элементы каждого списка записываются в квадратных скобках через запятую.
SByBool(Log, Values) Выдает значение строкового выражения из списка Values, для первого истинного выражения из списка Log. Элементы каждого списка записываются в квадратных скобках через запятую.
RByNum(Num,Values) Выдает значение вещественного выражения из списка Values, по номеру Num>=1.
IByNum(Num,Values) Выдает значение целого выражения из списка Values, по номеру Num>=1.
SByNum(Num,Values) Выдает значение строкового выражения из списка Values, по номеру Num>=1.
BByNum(Num,Values) Выдает значение логического выражения из списка Values, по номеру Num>=1.
CFun(X, Args,Values) Вычисляет числовую функцию как непрерывную. Функция задана списками координат.
DFun(X, Args,Values) Вычисляет функцию как дискретную. Функция задана списками координат.
ICFun(X, Args,Values); Вычисляет числовую функцию как непрерывную, но округленную до целого. Функция задана списками координат.
IDFun(X, Args,Values); Вычисляет числовую функцию как дискретную, но округленную до целого. Функция задана списками координат.
Any Всегда истинная логическая функция.
GetFile(FileName) Выдает содержимое файла в виде строки.
RMaxNum(Values) Выдает номер вещественного выражения из списка в квадратных скобках, у которого значение - максимально. Значение функции всегда >=0.
IMaxNum(Values) Выдает номер целого выражения из списка в квадратных скобках, у которого значение - максимально. Значение функции всегда >=0.
SMaxNum(Values) Выдает номер строкового выражения из списка в квадратных скобках, у которого значение - максимально. Значение функции всегда >=0.
BMaxNum(Values) Выдает номер логического выражения из списка в квадратных скобках, у которого значение - максимально. Значение функции всегда >=0.
RMinNum(Values) Выдает номер вещественного выражения из списка в квадратных скобках, у которого значение - минимально. Значение функции всегда >=0.
IMinNum(Values) Выдает номер целого выражения из списка в квадратных скобках, у которого значение - минимально. Значение функции всегда >=0.
SMinNum(Values) Выдает номер строкового выражения из списка в квадратных скобках, у которого значение - минимально. Значение функции всегда >=0.
BMinNum(Values) Выдает номер логического выражения из списка в квадратных скобках, у которого значение - минимально. Значение функции всегда >=0.
RMaxVal(Values) Выдает значение вещественного выражения из списка в квадратных скобках, у которого значение - максимально.
IMaxVal(Values) Выдает значение целого выражения из списка в квадратных скобках, у которого значение - максимально.
SMaxVal(Values) Выдает значение строкового выражения из списка в квадратных скобках, у которого значение - максимально.
BMaxVal(Values) Выдает значение логического выражения из списка в квадратных

скобках, у которого значение - максимально.
RMinVal(Values) Выдает значение вещественного выражения из списка в квадратных скобках, у которого значение - минимально.
IMinVal(Values) Выдает значение целого выражения из списка в квадратных скобках, у которого значение - минимально.
SMinVal(Values) Выдает значение строкового выражения из списка в квадратных скобках, у которого значение - минимально.
BMinVal(Values) Выдает значение логического выражения из списка в квадратных скобках, у которого значение - минимально.
BoolToInt(Log) - представляет логическое значение - как целое.
Специальные процедуры управления моделью.
InitRand(Value); Устанавливает начальное целое значение для генератора случайных чисел.
ReportVal(Varia,Name); Выводит выражение вещественного, целого, строкового, или логического типа в отчет, Name - заголовок, по умолчанию он пустой.
ReportProc; Выводит отчет в целом.
ReportBlocks(FileName); Выводит отчет о блоках в указанный файл.
ReportCurrentList(FileName); Выводит отчет о списке текущих событий в указанный файл.
ReportSynchronizeList(FileName); Выводит отчет о списке синхронизации в указанный файл.
ReportInterruptList(FileName); Выводит отчет о списке прерываний в указанный файл.
ReportRPList(FileName); Выводит отчет о RP -параметрах в указанный файл.
ReportIPList(FileName); Выводит отчет о IP -параметрах в указанный файл.
ReportSPList(FileName); Выводит отчет о SP -параметрах в указанный файл.
ReportBPList(FileName); Выводит отчет о BP -параметрах в указанный файл.
Start(Tg); Задает начальное значение счетчика завершений и запускает модель.
AReset; Выполняет сброс модели.
ClearModel(Clear); Очищает модель от всех данных при Clear =True . При Clear =False - в модели сохраняются без изменений значения всех переменных и всех X - параметров.
AClear - то же самое, что и ClearModel(True).
Динамические массивы.
Для облегчения работы с массивами предусмотрены следующие типы динамических массивов.
ADouble=array of Double;
AIntger=array of Integer;
AString=array of String;
ABoolean=array of Boolean;
Переменные для массивов должны быть описаны по образцу
Name: ADouble;
Для дозаписи в них списка значений можно использовать блоки.
*:RALet(Name,[список],Num);
*:IALet(Name,[список],Num);
*:SALet(Name,[список],Num);
*:BALet(Name,[список],Num);
Здесь, Num - позиция, начиная с которой динамический массив Name пополняется

списком. По умолчанию, содержимое динамического массива замещается списком.
Блоки
*:RAAdd(Name,[список]);
*:IAAdd(Name,[список]);
*:SAAdd(Name,[список]);
*:BAAdd(Name,[список]);
Дописывают список в конец динамического массива.
Динамические массивы можно указывать вместо списков, но тогда они указываются без квадратных скобок.
Для записи в массивы списка значений можно использовать процедуры.
RALetProc(Name,[список],Num);
IALetProc(Name,[список],Num);
SALetProc(Name,[список],Num);
BALetProc(Name,[список],Num);
Они аналогичны блокам группы ALet. А также
RAAddProcNext(Name,[список]);
IAAddProc(Name,[список]);
SAAddProc(Name,[список]);
BAAddProc(Name,[список]);
Они аналогичны блокам группы AAdd.
Номера элементов динамических массивов начинаются с 0.
Для вывода отчетов о массивах - можно использовать процедуры. Здесь вместо списка можно указывать динамические массивы.
RAReportVal([список], заголовок)
IAReportVal([список], заголовок)
SAReportVal([список], заголовок)
BAReportVal([список], заголовок)
Функция FindSort(AVal,Val) -в динамическом массиве AVal, отыскивается номер, в котором расположено первое попавшееся подходящее значение, большее или равное Val. Массив AVal непременно должен быть упорядоченным и быть одного из следующих типов: ADouble, AInteger, AString, ABoolean.
Функция ASub(AVal, NumB,Count) -в динамическом массиве AVal, выделяется под массив начиная с NumB, длиной в Count элементов. Массив должен быть одного из следующих типов: ADouble, AInteger, AString, ABoolean.
Для описания динамических массивов основных объектов используются следующие типы.
ATParam=Array of TParam;
ATFacility=Array of TFacility;
ATQueue=Array of TQueue;
ATStorage=Array of TStorage;
ATTable=Array of TTable;
ATUser=Array of TUser;
ATFuture=Array of TFuture;
ATRGroup=Array of TRGroup;
ATIGroup=Array of TIGroup;
ATSGroup=Array of TSGroup;

ATTGroup=Array of TTGroup);
Набор процедур для этих массивов очень ограничен и состоит из следующих процедур для присваивания массивам набора значений, то есть объектов. Тогда к объектам можно будет обращаться как по именам, так и по их номерам в массиве. В процедурах - Name - это имя массива TParam, TFacility ..
ParamSet(Name, [список]);
FacilitySet(Name, [список]);
QueueSet(Name, [список]);
StorageSet(Name, [список]);
TableSet(Name, [список]);
UserSet(Name, [список]);
FutureSet(Name, [список]);
RGroupSet(Name, [список]);
IGroupSet(Name, [список]);
SGroupSet(Name, [список]);
BGroupSet(Name, [список]);
TGroupSet(Name, [список]);
В процедуре CloseAllObj все динамические массивы должны быть освобождены, то есть для каждого массива следует выполнить оператор
Name:=Nil;
Для описания динамических массивов процедур и функций без параметров используются следующие типы.
ATProcedure=Array of TProcedure;
ATRFunction=Array of TRFunction;
ATIFunction=Array of TIFunction;
ATSFunction=Array of TSFunction;
ATBFunction=Array of TBFunction;
Набор процедур для этих массивов аналогичен набору процедур для основных объектов и состоит из следующих процедур для присваивания массивам
набора значений, то есть процедур и функций. Тогда к процедурам или функциям можно будет обращаться как по именам, так и по их номерам в массиве.
В процедурах - Name - это имя массива.
ProcedureSet(Name, [список]);
RFunctionSet(Name, [список]);
IFunctionSet(Name, [список]);
SFunctionSet(Name, [список]);
BFunctionSet(Name, [список]);
В процедуре CloseAllObj все динамические массивы должны быть освобождены, то есть для каждого массива следует выполнить оператор
Name:=Nil;

Дополнительные и функции, реализованные в модуле AddModelUnit.

При работе с графическими изображениями можно передвигать их при нажатой правой кнопке. При выделении прямоугольника левой кнопкой, прямоугольник, выделенный

вправо и вниз, увеличивает масштаб выделенной части, прямоугольник, выделенный любым другим способом - возвращает изображение в исходное состояние. Эти правила действуют на вкладках Lines и Bars.

Блоки.

*:ToPoint(Num,X,Y); Блок выполняет вывод на график следующей точки. Num>=0 - номер серии, X,Y - координаты.

*:ToString(Num,Value); Блок выполняет вывод на AddMemo строкового, вещественного целого или логического параметра . Num>=0 - номер строки, Value - параметр.

*:ToValue(Num,Value); Блок выполняет вывод значения Value для строки Num>=0 в виде величины соответствующего столбика.

*:LineClear(Num); Блок сброса графика с данным номером Num>=0, по умолчанию сбрасываются все графики.

*:BarClear(Num); Блок сброса гистограмм с данным номером Num>=0, по умолчанию сбрасываются все гистограммы.

*:CurrentBlocks; Блок выполняет вывод на AddMemo статистики по блокам.

*:Stop; блок, обеспечивающий почти немедленное прекращение моделирования. Полезен при отладке и в некоторых других случаях. Содержимое счетчика завершений не становится равным нулю.

*:Message('Сообщение'); выводит сообщение в окне сообщений. Если текста нет, то сообщение закрывается.

GetTerm Выдает значение для Term из формы.

RInput(Line,Def) Вызывает диалог для ввода вещественного значения. Line - строка для заголовка. Def- начальное значение, по умолчанию -0.

IInput(Line,Def) Вызывает диалог для ввода целого значения. Line - строка для заголовка. Def- начальное значение, по умолчанию -0.

SInput(Line,Def) Вызывает диалог для ввода строкового значения. Line - строка для заголовка. Def- начальное значение, по умолчанию - пустая строка.

BInput(Line,Def) Вызывает диалог для ввода логического значения. Line - строка для заголовка. Def- начальное значение, по умолчанию -False.

InputItem(Line,Def) Вызывает диалог для выбора строки из выпадающего списка, Line - заголовок. Def- начальное значение номера строки, по умолчанию -0.

RGet(NumStr) Ввод вещественного значения из строки Numstr>=0 в AddMemo.

IGet(NumStr) Ввод целого значения из строки Numstr>=0 в AddMemo.

SGet(NumStr) Ввод строкового значения из строки Numstr>=0 в AddMemo.

BGet(NumStr) Ввод логического значения из строки Numstr>=0 в AddMemo.

RGB(Red,Green,Blue) Формирует цвет по интенсивностям трех основных цветов. Каждый цвет может иметь значение от 0 до 255.

FileNameGet. Вызывает диалог открытия файла и возвращает его имя как строку.

FileNamePut. Вызывает диалог закрытия файла и возвращает его имя как строку.

GetColor. Позволяет выбрать цвет из диалога.

Процедуры управления моделью.

CShow(X,Y, Num); Выводит график непрерывной функции. В ней указываются координаты точек, номер графика. По умолчанию, номер графика - 0.

DShow(X,Y, Num); Выводит график дискретной функции. В ней указываются координаты точек, номер графика. По умолчанию, номер графика - 0.

NewItems(lines); Заполняет строками выпадающий список для InputItem. Список строк

записывается в квадратных скобках через запятую.
AddMemo(Line); Добавляет вещественное, целое, строковое или логическое значение к AddMemo.
AddMemoReset; Очистка AddMemo.
RepMemoReset; Очистка RepMemo.
AddLoad(FileName); Загружает файл в AddMemo.
AddSave(FileName); Сохраняет файл из AddMemo.
Line3D(Yes3D); Переключает, плоские - объемные графики.
Bar3D(Yes3D); Переключает, плоские - объемные гистограммы.
LineSave(FileName); Сохраняет графики в файле на диске.
BarSave(FileName); Сохраняет гистограммы в файле на диске.
ReportSave(FileName); Сохраняет отчет по модели в файле.
LineColor(Num,Col); Задает цвет графика с номером Num>=0 .
LineTitle(Num,Line); Задает имя графика.
ValueColor(Num,Col); Задает цвет столбика с номером Num>=0.
ValueTitle(Num,Line); Задает имя столбика с номером Num>=0.
BarColor(Num,Col); Задает цвет гистограммы с номером Num>=0.
BarTitle(Num,Line); Задает имя гистограммы с номером Num>=0.
GetLine (Num,AX,AY); линия с номером Num копируется в динамические массивы.
GetBar (Num,AX,AY); гистограмма с номером Num копируется в динамические массивы.
SetStart(Tg1); Устанавливает значения счетчика завершений на главной форме.
ToStringProc(Num,Value); На AddMemo выполняет вывод вещественного, целого, строкового или логического значения Value. Номер строки - Num>=0.
ToValueProc(Num,Value); Выполняет вывод значения Value для строки Num>=0 в виде величины соответствующего столбика.
LineShow(HideShow,Num); Показать- скрыть (HideShow) график Num>=0. По умолчанию, все графики.
BarShow(HideShow,Num); Показать- скрыть (HideShow) гистограмму Num>=0. По умолчанию, все гистограммы.
RepMemoAdd(Line); Добавляет значение вещественного, целого строкового или логического выражения к RepMEMO.
Bar(X,Y, Num); создает из динамических массивов X,Y - гистограмму, Num>=0 - серия.
ToPointProc(Num,X,Y); Добавляет следующую точку X,Y к графику, Num>=0 - серия.
LineClearProc(Num); Сброс заданной серии для графиков, по умолчанию, сбрасываются все серии.
BarClearProc(Num); Сброс заданной серии для гистограмм, по умолчанию, сбрасываются все серии.
ValueClearProc; Сброс серии значений.
CustomProc(st,apr); Создает перечень заголовков процедур пользователя и перечень самих процедур. Перечни записываются в квадратных скобках через запятую.
SetPage(Page); Переключает приложение на страницу.
ShowPage(Page,flag); Показывает или скрывает страницу.
PageColor(Page,Col); Задает цвет страницы.
Список констант для SetPage, PageColor и ShowPage.
spSimulation, spReportSetting, spReport, spLine, spValues, spBar, spAbout, spReserve

,spLicence
Для работы в пакетном режиме, имеются следующие процедуры.
FormShow(log); Показывает или скрывает форму.
Quit; Закрывает приложение.
RunBat; Выполняет моделирование в пакетном режиме. На самом деле при этом выполняются FormShow(False); Simulation; ReportProc; CloseApp;
SetReportFlag(Flag, val); Соответствующий флаг отчета устанавливается в True или False, по умолчанию - в истину.
MessageProc('Сообщение'); выводит сообщение в окне сообщений. Если текста нет, то сообщение закрывается.
Константы для управления настройкой отчета из модели.
rfModel, rfInterruptList, rfUserList, rfPList, rfXList, rfCurrentList,
rfFutureList, rfSynchronizeList, rfGroupList, rfBlocks, rfQueue,rfStorage,
rfUser, rfFacility, ;rfTable, rfVal, rfGroup
Список предопределенных констант для задания цвета.
ClAqua - бирюзовый clBlack - черный clBlue - синий clCream - кремовый clDkGray Темно-серый clFuchsia фуксиновый clGray - темно серый clGreen - зеленый clLime - лимонно зеленый
clLtGray - светло-серый clMaroon марон clMedGray средне серый clMoneyGreen цвет доллара
clNavy морской синий clOlive оливковый clPurple фиолетовый clRed красный
clSilver серебряный clSkyBlue голубой clTeal темно бирюзовый clWhite белый clYellow желтый

Приложение 2. Стандартная выходная статистика.

На вкладке Report содержится стандартная статистику об объектах GPSS - Future, используемых в данной модели (FACILITY, QUEUE, STORAGE и т.д.). Начинается отчет со строки, сообщающей о начальном и конечном времени моделирования, которая выглядит, например, следующим образом.

```
StartTime        0.00000 EndTime    1211359.83515
```

Далее идет конвертированный текст самой модели, который мы уже неоднократно наблюдали, например:

```
{=Gen} ::Gen 1:Future.NewTransaction(Exponential(120),0,Gen,Gen+1);
    2: Qkass.Queue ;
    3: Kass.Seize ;
    4: Qkass.Depart ;
    5: Future.Wait( MeanDev(105,25));
    6: Kass.release ;
    7: ToPoint (1,aclock,Qkass.A) ;
    8: Terminate (1);
```

Затем идет информация о блоках, в которой первая колонка определяет номер блока, вторая – число вхождений в блок, а третья – текущее число заявок в блоке. Это выглядит, например, следующим образом.

При истолковании статистики по блокам следует учитывать, что заявка может находиться в блоке, или оказаться задержанной на его выходе, из-за невозможности войти в следующий блок. Колонка `Current` содержит суммарное число заявок, как внутри блока, так и на его выходе. Для различения этих ситуаций можно включать в текст

модели блоки *:Nop. Этот блок не выполняет никаких действий, но и не препятствует входу в него заявки.

Также рекомендуется помещать блок *:Priority(-1) непосредственно перед *:Terminate, тогда статистика будет соответствовать ситуации в модели, когда уже завершена обработка событий текущего момента. Впрочем, для большинства задач такие подробности не имеют значения.

```
|R `Blocks    BLOCK Report
   Location     Total   Current
        1     10002         0
        2     10002         2
        3     10000         0
        4     10000         0
        5     10000         0
        6     10000         0
        7     10000         0
        8     10000         0
```

Далее в модели идет информация о списке текущих событий, списке будущих событий, списке синхронизации, списке прерываний, списке пользователя, списках P – параметров. А затем, в произвольном порядке, точнее в порядке описания, идет информация об объектах следующих типов Tparam, TFacility, TQueue, TStorage, TTable, TUser, TRGroup, TIGroup, TSGroup, TTGroup.

Если какой либо объект отсутствует, или список пуст, то соответствующий раздел отчета – отсутствует. Можно отменить вывод любого из перечисленных разделов отчета.

Список текущих событий выводится в следующей форме.

```
        |R `CurrentList Report Current Transactions List  Count= 2
    Tran   Assem     Pri Current    Next       TimeStamp       TimeEnter Inter
   10001   10001       0       2       3   1211101.73957   1211101.73957 FALSE
   10002   10002       0       2       3   1211139.47898   1211139.47898 FALSE
```

Здесь `TRAN` - номер заявки, `Assem` – номер семейства, `Pri` – приоритет, `Current` – блок, где заявка находится в данный момент, `Next` – блок, куда заявка планирует перейти, `TimeStamp` – время рождения заявки, `TimeEnter` – время поступления заявки в список, `Inter` – состояние прерывания, 0(False)- заявка не прервана, 1(True)- прервана.

Список будущих или список пользователя событий выводится в следующей форме.

```
        |R User/Future = `Future Entries = 76609 Current = 4 AverUser =  3.40610 AverTime =
133.38250
    Max = 5 Min = 0 Active Time = 3000000.00000 NextGo = 0
```

Здесь `User` - имя списка, `Entries` - количество заявок, вошедших в список, `Current` - текущее число заявок в списке, `Max` - максимальное число заявок в списке, `Min` - минимальное число заявок в списке, `AverUser` - среднее число заявок в списке, `AverTime` - среднее время, проведенное заявкой в списке, `Active Time` - время активности объекта.

```
|R Report  User/Future List Count= 4
    Tran   Assem     Pri Current    Next       TimeStamp       SortValue   Group Inter
   38305   38305       0       0       1   3000094.41658   3000094.41658       0 FALSE
   38304   38304       1      16      17   2999976.30636   3000151.87426       0 FALSE
   38307   38307       0       0      10   3000898.77531   3000898.77531       0 FALSE
   38308   38308       0       0      20   4000000.00000   4000000.00000       0 FALSE
```

В этом списке назначение колонок почти такое же, как и для списка текущих событий, а колонка `SortValue` – определяет момент начала движения для заявки или параметр упорядочения.

Список синхронизации выводится в следующей форме.

```
|R `SynchronizeList Report Synchronize Transactions List Count= 1
 Tran   Assem     Pri Current    Next       TimeStamp       TimeEnter Inter
46303   46299       0      13      14   9999078.29184   9999993.98877 FALSE
```

В этом списке назначение колонок почти такое же, как и для списка текущих событий.

Список прерываний выводится в следующей форме.

```
Report Interrupt LIST
Tran  NumFacil      IntTran
991        5        991
992        5        991
```

В этом списке `TRAN` - номер заявки, `NumFacil` - номер объекта устройство, `TRANP` - номер прерванной заявки.

Списки P – параметров выводится в следующей форме.

```
RP LIST ( Double )
Tran     Num/Name          Double
1001         1     99305.13187
1001         2     99622.20574
```

В этом списке вначале выводится заголовок, определяющий вид списка, а затем сам список в три колонки, первая – номер заявки, вторая – номер параметра, а третья – его значение.

Информация об устройствах имеет вид.

```
|R Facility = `Kass Entries = 10000 Current = 0 Utility =  0.66295 AverTime = 80.30700
 Tran = 0 IntTran = 0 Active Time = 1211359.83515 NextGo = 0
```

Здесь `Facility` - имя устройства, `Entries` - количество заявок, вошедших в устройство, `Current` - активная занятость устройства, `Utility` - коэффициент занятости, `AverTime` - среднее время обслуживания заявки, `NextGo` - номер блока для возможного ухода заявки, `TRAN` - номер обслуживаемой заявки, `TRANP` - номер прерванной заявки, `NumObj` - номер объекта.

Информация об очередях имеет вид.

```
|R Queue = `Qkass Entries = 10002 Current = 2 AverQueue =  0.69956 AverTime = 84.72497
AverTime(-Ze) = 186.57400 Max = 10 Min = 0 Zero = 5460 Active Time = 1211359.83515 NextGo = 0
```

Здесь `Queue` - имя очереди, `Entries` - количество заявок, вошедших в очередь с учетом кратности, `Current` - активная длина очереди, `Max` - максимальная длина очереди, `Zero` - количество заявок в очереди не задержавшихся, `AverQueue` - средняя длина очереди, `AverTime` - среднее время, проведенное заявкой в очереди, `AverTime(-Ze)` - среднее время, проведенное заявкой в очереди, без учета в очереди не задержавшихся, `NumObj` - номер объекта.

Информация о многоканальных устройствах имеет вид.

```
|R Storage = `evm Entries = 48260 Current = 3 Average =  6.70275 AverTime =  9.99995 Utility =  0.55856
 Max = 12 Min = 0 Tran = 7753 ACapac = 12.00000 Capacity = 12 Active Time = 72000.00000 NextGo = 0
```

Здесь `Storage` - имя устройства, `Entries` - количество заявок, вошедших в устройство с учетом кратности, `Current` - активная занятость, `Max` - максимальная занятость, `Min` - минимальная занятость, `AverStor` - среднее число занятых каналов, `AverTime` - среднее время, проведенное заявкой в устройстве, `Utility` - коэффициент занятости, `NextGo` - номер блока для возможного ухода заявки, `TRAN` - - номер последней поступившей заявки, `AC1` - время поступления последней заявки, `ACapac` - средняя емкость, `Capacity` - активная емкость, `NumObj` - номер объекта.

Информация о таблице имеет вид.

```
|R  Table = `tt[i]  Entries = 1887.00000 Mean = 13.09724 StdDev =  5.98429
          Range        Frequency        Integral
        0.00000          0.00000         0.00000
```

```
10.00000      1161.00000         0.61526
20.00000       556.00000         0.90991
30.00000       123.00000         0.97509
40.00000        34.00000         0.99311
50.00000         8.00000         0.99735
60.00000         4.00000         0.99947
70.00000         1.00000         1.00000
80.00000         0.00000         1.00000
```

Здесь в заголовке последовательно указаны имя таблицы, количество вхождений в нее с учетом кратности, среднее значение по таблице, среднеквадратичное отклонение, номер объекта. А далее, в два столбца идет правая граница интервала, и количество вхождений в интервал.

Информация о группе целых, вещественных, и строк имеет вид.

```
   |R GroupInt = `Igr Entries = 10006 Current = 7 Average =  7.29331 AverTime = 729.90327
    Max = 12 Min = 0 Active Time = 1001385.50178 NextGo = 0
Igr       Report GROUP LIST ( Integer ) num=7
              2
              3
              4
              8
              9
             13
             14
```

Здесь `GroupInt` - имя группы, `Entries` - количество заявок, вошедших в группу, `Current` - текущее количество элементов группы, `Max` - максимальное количество элементов группы, `Average` - среднее количество элементов группы, и так далее. А ниже следует колонка значений элементов группы.

Информация о группе заявок имеет вид.

```
|R GroupTran = ` grou Entries = 434 Current = 6 Average =  5.68657 AverTime = 1492.31492
    Max = 12 Min = 0 Active Time = 113893.75522 NextGo = 0
 grou     Report TGROUP LIST ( TRANSACTIONS ) num=6
           1225
           1226
           1227
           1228
           1230
           1238
```

Здесь `GroupTran` - имя группы, `Entries` - количество заявок, вошедших в группу, `Current` - текущее количество заявок в группе, и так далее. А ниже следует колонка номеров заявок из группы.

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.- М.: Машиностроение, 1980.-592с.