В. ТОМАШЕВСКИЙ, Е. ЖДАНОВА

ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ В СРЕДЕ GPSS

Издательство Бестселлер
Москва
2003

УДК681.32
ББК 65в6

Рецензент:
доктор технических наук, профессор, заведующий кафедрой ИУ5 («Системы обработки информации и управления») МГТУ им. Н. Э. Баумана **В. М**. **Черненький**

Томашевский В., Жданова Е.
Имитационное моделирование в среде GPSS. – М.:Бестселлер, 2003. – 416 с. ISBN 5-98158-004-6

Книга о методах имитационного моделирования и его применении в науке, технологиях, бизнесе. Приведено описание языка GPSS и секретов его эффективного использования для моделирования сложных систем, **А** также примеры и задания.

Для специалистов по вычислительной технике, научных работников студентов, **А** также всех, кто интересуется современными компьютерными технологиями.

ПРЕДИСЛОВИЕ

В книге рассматриваются основы дискретно-событийного моделирования систем и технология имитационного моделирования. Авторами обобщен опыт преподавания дисциплин, связанных с имитационным моделированием, и предыдущих двух книг [1,2]. Приведенный материал имеет практическую направленность и позволяет самостоятельно освоить средства имитационного моделирования для решения практических задач. Большое количество приведенных примеров, пояснений и задач способствует этому. Основное назначение этой книги – научить принимать решение по результатам экспериментов с имитационными моделями.

Первая и вторая главы охватывают проблемы систем и сетей массового обслуживания, на которых основываются имитационные модели технических, технологических, информационных, экономических и других систем. Рассматриваются принципы построения моделирующих алгоритмов. В отличие от математических методов теории массового обслуживания используется операционный анализ, который позволяет анализировать эти системы, основываясь на логике их работы. Операционный анализ дает возможность во многих случаях провести предварительные расчеты модели без применения метода имитационного моделирования.

В третьей главе рассматриваются методы вероятностного моделирования дискретных и случайных непрерывных величин, которые позволяют учитывать при моделировании случайные воздействия на систему.

Четвертая глава знакомит с языком GPSS и конкретно с его реализацией GPSS World для построения имитационных моделей. Принцип изложения материала – от простого к сложному. Возможности языка иллюстрируются конкретными примерами с пояснениями. Приведены также основные средства языка для моделирования разных задач.

В пятой главе изложены основы моделирования вычислительных и операционных систем (ОС), **А** в шестом – основы моделирования разных процессов, которые надо знать, чтобы выполнить самостоятельные задания, приведенные в седьмой главе.

В восьмой главе рассматривается технология работы с программной системой lSS 2000, позволяющей автоматизировать процесс создания имитационных моделей на языке GPSS и проведения экспериментов с моделями без написания кода программы. Встроенные аналитические средства позволяют определить узкие места в моделируемой системе до начала моделирования.

Девятая глава охватывает проблемы построения имитационных проектов и технологии выполнения работ по моделированию и включает вопросы организации, планирования и проведения экспериментов. Особое внимание уделено вопросам точности получаемых оценок по результатам моделирования в переходном и установившемся режиме. Рассмотрен дисперсионный анализ; отсеивающий, оптимизационный и пользовательский эксперименты, приведены примеры их реализации в GPSS World. Рассмотрены вопросы структурной оптимизации моделируемой системы и выбора наилучшего из альтернативных вариантов.

Примеры принятия решений по результатам имитационного моделирования приведены в десятой главе.

Одиннадцатая глава включает задания для имитационных проектов из разных областей деятельности и охватывает технические, производственные, технологические, организационные и обслуживающие системы.

ВВЕДЕНИЕ

Компьютерное моделирование нашло практическое применение во всех сферах деятельности человека, начиная от моделей технических, технологических и организационных систем и заканчивая проблемами развития человечества и вселенной. Еще с детства человек через игрушки и игры узнает мир и таким образом моделирует действительность. Вместо того, чтобы учиться на своих ошибках или на ошибках других людей, целесообразно закреплять и проверять познание реальной действительности полученными результатами на компьютерной модели. В этом случае есть возможность «проигрывать» на модели любые ситуации, включая те, при которых реальная система вышла бы из строя. Это позволяет моделировать катастрофы, редкие события и т. п. Одно из преимуществ компьютерного моделирования – это также моделирование того, что не существует на самом деле, то есть

моделирование виртуальной реальности. Вспомним компьютерные игры: сидя в самолете или космическом корабле, мы осуществляем полет так, как будто мы действительно находимся там.

Когда же необходимо использовать компьютерное моделирование? Всегда, когда можно поставить вопрос, «что будет, если ...?». Следовательно, компьютерное моделирование используют, прежде всего, для принятия решений. Модель позволяет проигрывать любые ситуации и получать наиболее эффективные решения проблемы.

Из всех видов моделирования, **А** это в первую очередь математическое, графическое и прочее, рассмотрим имитационное моделирование. Обзор систем моделирования в работе [3] показывает, что имитационное моделирование является едва ли не самым популярным средством, используемым на практике для этих целей. Основная его ценность состоит в применении методологии системного анализа. Имитационное моделирование разрешает осуществить исследование анализируемой или проектируемой системы по схеме операционного исследования, которое содержит взаимосвязанные этапы:

- содержательная постановка задачи;
- разработка концептуальной модели;
- разработка и программная реализация имитационной модели; проверка правильности,
- достоверности модели и оценка точности результатов моделирование;
- планирование и проведение экспериментов;
- принятие решений.

Это позволяет использовать имитационное моделирование как универсальный подход для принятия решений в условиях неопределенности с учетом в моделях трудно формализуемых факторов, **А** также применять основные принципы системного подхода для решения практических задач.

Широкому внедрению этого метода на практике препятствует необходимость создания программных реализаций имитационных моделей, которые воссоздают в модельном времени динамику функционирования моделируемойой системы. В отличие от традиционных методов программирования разработка имитационной модели требует перестройки принципов мышления. Недаром принципы, положенные в основу имитационного моделирования, дали толчок к развитию объектного программирования. Поэтому усилия разработчиков программных средств имитации направлены на упрощение программных реализаций имитационных моделей: для этих целей создаются специализированные языки и системы. Программные средства имитации в своем развитии изменялись на протяжении нескольких поколений, начиная с языков моделирования и средств автоматизации конструирования моделей [4] до генераторов программ [5], интерактивных и интеллектуальных систем [6], распределенных систем моделирования. Основное назначение всех этих средств – уменьшение трудоемкости создания программных реализаций имитационных моделей и экспериментирования с моделями.

Одним из первых языков моделирования, облегчающих процесс написания имитационных программ, был язык GPSS, созданный в виде конечного продукта Джеффри Гордоном в фирме IBM в 1962 г. Этот язык в свое время входил в первую десятку лучших языков программирования, опережая транслятор с языка АЛГОЛ, и был реализован практически на всех типах ЭВМ. В настоящее время есть трансляторы для операционных систем DOS – GPSS/PC, для OS/2 и DOS – GPSS/H и для Windows – GPSS World. Изучение этого языка и создания моделей позволяет понять принципы разработки имитационных программ и научиться работать с имитационными моделями.

GPSS (General Purpose Simulation System – система моделирования общего назначения) – язык моделирования, который использует – ся для построения событийных дискретных имитационных моделей и проведения экспериментов на ЭВМ.

Модели систем на GPSS могут быть записаны в виде блок-схем или представлены в виде последовательности строк программы, эквивалентных блок-схеме. Блок-схема – это набор фигур с характерными контурами блоков языка GPSS, соединенных между собою линиями. Блоки – это подпрограммы, реализованные средствами макроассемблера. В разных версиях языка количество блоков для создания имитационных программ разное и составляет около 40. В язык моделирования GPSS входят специальные средства для описания динамического поведения систем через изменение состояний в дискретные моменты времени, то есть время моделирования изменяется случайно от события к событию.

Система GPSS представляет собой язык и транслятор. Как каждый язык он содержит словарь и грамматику, с помощью которых могут быть разработаны модели систем определенного типа.

Транслятор языка работает в две фазы. На первой фазе компиляции проверяется синтаксис и семантика написания строк GPSS-программы или всей программы в целом, **А** на второй (интерпретирующей) осуществляется продвижение транзактов по модели от блока к блоку.

ГЛАВА 1. МОДЕЛИ МАССОВОГО ОБСЛУЖИВАНИЯ

1.1. Системы массового обслуживания и их характеристики

С системами массового обслуживания (СМО) мы встречаем повседневно. Любому из нас приходилось когда-то ждать обслуживания в очереди (например, в магазине, на автозаправке, в библиотеке кафе и т. д.). Аналогичные ситуации возникают при потребности воспользоваться телефонной связью или выполнить свою программу на компьютере. Более того, любое производство можно представить как последовательность систем обслуживания. К типичным системам обслуживания относят также ремонтные и медицинские службы, транспортные системы, аэропорты, вокзалы и другие.

Особое значение приобрели такие системы при изучении процессов в информатике. Это, прежде всего, компьютерные системы сети передачи информации, ОС, базы и банки данных. Системы обслуживания играют значительную роль в повседневной жизни. Опыт моделирования разных типов дискретных событийных систем свидетельствует о том, что приблизительно 80% этих моделей основань на СМО.

Что же характеризует эти системы как СМО? Такие системы можно описать, если задать:

1) входящий поток требований или заявок, которые поступают на обслуживание;
2) дисциплину постановки в очередь и выбор из нее;
3) правило, по которому осуществляется обслуживание;
4) выходящий поток требований;
5) режимы работы.

Входящий поток. Для задания входящего потока требований необходимо описать моменты времени их поступления в систему (за кон поступления) и количество требований, которое поступило одно временно. Закон поступления может быть детерминированный (на пример, одно требование поступает каждые 5 мин) или вероятностный (требования могут появляться с равной вероятностью в интервале 5±2 мин). В общем случае входящий поток требований описывается распределением вероятностей интервалов времени между соседними требованиями. Часто предполагают, что эти интервалы времени независимые и имеют одинаковое распределение случайных величин, которые образуют стационарный входящий поток требований. Классическая теория массового обслуживания рассматривает так называемый ***пуассоновский*** (простейший) поток требований. Для этого потока число требований k для любого интервала времени распределено по закону Пуассона:

$$P_k(t) = \frac{(\lambda t)^k}{k!} e^{-\lambda t}, \; k \geq 0, t \geq 0, \qquad (1.1)$$

где λ – интенсивность потока требований (число требований за единицу времени).

На практике обоснованием того, что входящий поток требований имеет распределение Пуассона, является то, что требования поступают от большого числа независимых источников за определенный интервал времени. Примерами могут быть вызовы абонентов в телефонной сети, запросы к распределенной базе данных от абонентов сети за некоторое время и другие. Для того, чтобы при моделировании задать пуассоновскии поток требований в систему, достаточно задать экспоненциальное распределение интервалов времени поступление для соседних требований, графики функций плотности и распределения которых для $\lambda = 1$ показаны на рис. 1. 1.

Дисциплины постановки в очередь и выбора из нее определяют порядок постановки требований в очередь, если заняты устройства обслуживания, и порядок выбора из очереди, если освобождается обслуживающее устройство. Простейшая дисциплина допускает постановку в очередь в порядке поступления требований. Такую дисциплину называют ***«раньше поступил раньше обслужился»*** (РПРО, в англоязычной литературе FIFO – First In-First Out), например, очередь к телефону-автомату.

Рис. 1.1

Организация очереди по правилу ***«последний поступил – первый обслужился»*** (ППО, в англоязычной литературе LlFO – Last In-First Out) допускает, что на обслуживание выбираются последние требования из очереди. Это правило также называется «стеком» или «магазином».

Правило выбора из очереди может быть ***случайным*** (RANDOM).

Возможна также организация выбора из очереди ***по параметрам*** (например, мужчины в очереди пропускают женщин вперед).

На очередь могут накладываться ***ограничения по длине очереди*** или ***по времени пребывания в ней.*** Например, если в очереди находится больее трех требований, то новое требование, которое поступило, покидает систему; или, если время пребывания в очереди более двух минут, то требование покидает систему.

Очередь может быть с ограниченным количеством мест ожидания в ней – это так называемый ***буфер*** (например, бункер, в который поступают заготовки раньде, чем они будут обработаны станком). Для ускорения работы компьютеров используются буферы при обмене информацией между быстрыми и медленными устройствами (буферы ввода-вывода). Информация заранее размещается в буфере, **А** потом считывается из него. В сетях ЭВМ буферы используются для организации очередей сообщений или пакетов, если линия связи занята.

Правила обслуживания характеризуются длительностью обслуживания (распределением времени обслуживание), количеством требований, которые обслуживаются одновременно и дисциплиной обслуживания. Время обслуживания бывает детерминированным или заданным вероятностным законом распределения.

Обслуживание может организовываться с помощью одного устройства – это так называемые ***системы с одним устройством (каналом) обслуживания*** – или с несколькими идентичными устройствами обслуживания, например, если установлено несколько кабин с телефонами-автоматами. Системы с идентичными устройствами обслуживания называют ***многоканальными системами.*** Устройства обслуживания могут быть объединены в последовательную цепочку. Это ***многофазные системы обслуживания,*** в которых требования последовательно проходят несколько фаз обслуживания, перед тем как покинуть систему. В качестве примера многофазной системы обслуживания можно рассмотреть сборочный конвейер.

Дисциплины обслуживания определяют:

- при каких условиях прекращается обслуживание требований;
- как выбирается для обслуживания следующее требование;
- что делать с частично обслуженным требованием.

Различают дисциплины обслуживания ***бесприоритетные*** и ***приоритетные.*** При бесприоритетном обслуживании порядок обслуживания определяется дисциплиной выбора из очереди, например, РПРО. В компьютерных системах часто используются циклические дисциплины обслуживания, то есть требование (программа) многократно использует устройство (процессор) для обслуживания перед тем, как его оставит. После каждого этапа обслуживания требование снова поступает в очередь к устройству.

При приоритетном обслуживании требованию задается некоторый параметр, который определяет его ***приоритет.*** Этот параметр может задаваться в числовом виде ***(статический приоритет)*** или в виде функции, которая зависит от времени пребывания в системе ***(динамический приоритет).***

Дисциплины обслуживания могут быть с ***относительными*** или ***абсолютными приоритетами.*** Относительный приоритет предусматривает, что поступление требования с более высоким приоритетом не перерывает обслуживания менее приоритетного требования (обслуживание без прерывания). Из требований с одинаковыми приоритетами могут организовываться очереди.

При использовании абсолютного приоритета появление требования с более высоким приоритетом перерывает обслуживание менее приоритетного требования (обслуживание с прерыванием). В таких системах могут происходить вложенные прерывания, если требование, которое вытеснило из обслуживания менее приоритетное требование, само будет прервано более приоритетным требованием и т.д. Поэтому иногда в этих системах ограничивают глубину прерывания. Прерванные требования могут или оставлять систему обслуживания, или снова становиться в очередь для дообслуживания.

Понятно, что дисциплины с абсолютными приоритетами могут использоваться только для систем с одним устройством обслуживания.

Выходящий поток – это поток требований, которые покидают систему, причем требования в нем могут быть как обслуженные, так и не обслуженные. Структура выходящего потока может иметь большее значение для многофазных систем, где этот поток становится входящим для следующей фазы обслуживания. Распределение требований в выходящем потоке во времени зависит от плотности входящего потока и характеристик работы устройств обслуживания. Из теории массового обслуживания известно, что выходящий поток из СМО с m устройствами с ожиданием при простейшем входящем потоке с параметром Я и экспоненциальном распределении времени обслуживания с параметром μ есть простейший поток с параметром $\lambda = \min\{\lambda, m\mu\}$. Такое замечание дает возможность построить теорию сложных СМО, где выходящий поток из одних систем обслуживания есть входящий в другие системы. Это так называемые многофазные системы и сети СМО. Во всех других случаях распределение выходящих потоков из СМО имеет более сложную вероятностную природу и может изучаться только наблюдениями за функционированием этих СМО с помощью моделирования.

По практическим соображениям часто приходится изучать ***режимы работы*** СМО. Например, устройства обслуживания время оі времени могут выходить из строя (режим отказа), в особенности, если с помощью этих систем описывается некоторый производственный или информационный процесс. Есть еще один режим – блокирование обслуживания, – который связан с временным прерыванием процессг обслуживания или с замедлением его. Изменение режима работь: СМО может быть вызвано внешним влиянием (например, временньш отсутствием деталей в технологическом процессе, ремонтом оборудования и т.п.) или продолжительностью работы (например, выход и: строя элемента в компьютере).

Для СМО любого вида справедлив закон Литтла: для любогс распределения времени между двумя событиями поступления требований, любого распределения времени их обслуживания, любого количества устройств обслуживания и любой дисциплины обслуживания среднее количество требований $\overline{N}$ в СМО определяется через интенсивность поступления λ и среднее время пребывания требование в системе T, то есть:

$$\overline{N} = \lambda T . \qquad (1.2)$$

Интуитивное доказательство формулы Литтла основано на том, что требование, которое входит в систему, застанет в ней среднее количество требований $\overline{N}$, такое же как и в момент, когда оно покидает систему. Это свидетельствует о том, что СМО находится в состоянии равновесия или стационарном состоянии, то есть требования не остаются в системе бесконечно долго и всегда покидают систему. Таким образом, на вид СМО не накладываются никакие ограничения. Можно, например, представить, что СМО состоит только из одной очереди или только из одного устройства обслуживания.

1.2. Системы с одним устройством обслуживания

Рассмотрим одноканальную (с одним устройством обслуживания) СМО, показанную на рис. 1.2.

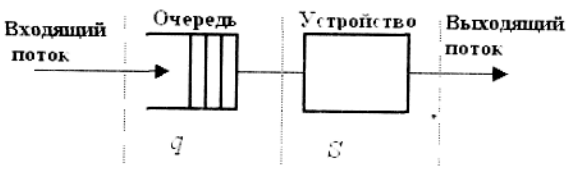


Рис. 1.2

Если обозначить среднее время пребывания требований в очереди *w* и рассматривать СМО как очередь *q,* то, используя формулу Литтла, можно найти среднее количество требований в очереди:

$$\overline{N}_q = \lambda w. \qquad (1.3)$$

Если обозначить среднее время обслуживания в устройстве $\overline{x}$ и рассматривать СМО как устройство *S*, то, используя формулу Литтла можно найти среднее количество требований в устройстве:

$$\overline{N}_S = \lambda \overline{x}. \qquad (1.4)$$

Всегда имеет место уравнение $T = w + \overline{x}$, где T – среднее время пребывания требований в СМО с одним устройством обслуживания.

Коэффициент загрузки р определяет, какую часть времени устройство было занято на протяжении всего времени наблюдения за СМО.

Для обозначения СМО используются три параметра для первых трех параметров: X/Y/Z, где **X** – распределение времени поступления; Y – распределение времени обслуживания; Z – число обслуживающих устройств.

В теории СМО некоторые аналитические решения были получены для систем вида D/D/1, M/M/1 и M/G/1. Для других значений параметров систем обслуживания аналитические решения не были получены, то есть эта проблема мотивирует использование моделирования.

Самая известная модель – это так называемая СМО типа M/M/1, где M – марковские процессы распределения времени поступления и обслуживания с одним устройством. Например, в системе M/M/1 время между двумя поступлениями в систему требований и время обслуживания имеют экспоненциальные распределения. Такая СМО иногда используется как модель для одного процессора компьютерной системы или как стандартное устройство ввода-вывода (например, магнитный диск). Система D/D/1 – детерминированная система, тогда как D/M/1 – смешанная. Если о системе мало известно, это обозначается как G/G/m, то есть система с произвольными распределениями и *m* устройствами.

Изучая любую систему, важно оценить характер ее рабочей нагрузки (например, при моделировании компьютерной системы важно знать: когда новые программы (задачи) поступают в систему; сколько времени нужно процессору для выполнения любой из них; как часто программа обращается к устройству ввода-вывода). Этот процесс можно отобразить графиком работы системы (графический метод моделирования), на котором показаны входы задач в систему, ресурсы к которым они обращаются, как долго задачи их используют и т.д.

Если описанный сценарий зафиксирован соответствующим графиком и часто возникает в моделируемой системе, то тогда он целиком отвечает выборке, которая получена методом измерений при наблюдении за работой компьютера. Тем не менее, моделирование при использовании такого описания рабочей нагрузки только воссоздает результаты работы этого специфического сценария. Этого недостаточно для выполнения системой других сценариев. Даже незначительное несоответствие заданному сценарию может привести к драматическим последствиям работы компьютера.

Часто рабочая нагрузка на систему определяется одним или несколькими распределениями вероятностей в отличие от заданных сценариев. Например, можно бросать монету каждые 15 мин на протяжении операции исследования системы, и если монета падает лицевой стороной, то новая задача поступает в систему в этот момент времени. Если монета падает обрятной стороной, то никакая задача не поступает в систему. Это пример метода розыгрыша случайной величины (метод Монте-Карло), который используется для моделирования вероятностных систем.

В компьютерном моделировании «бросание монеты» можно генерировать методом случайных чисел. Если выявлены статистические закономерности и используются соответствующие распределения вероятностей для определения рабочей нагрузки на систему, **А** также применяются соответствующие статистические методы анализа результатов моделирования, то полученные результаты относятся к более широкому диапазону рабочих нагрузок, чем подход с использованием определенного сценария.

Введем коэффициент вариации *C* как отношение стандартного отклонения к среднему:

$$C = \frac{\sigma_{\overline{x}}}{\overline{x}}, \qquad (1.5)$$

где $\sigma_{\overline{x}}$ – среднеквадратичное отклонение для $\overline{x}$.

Для экспоненциального закона распределения C= 1, поскольку $\bar{x}$ и $\sigma_{\bar{x}}$ для этого закона равняется λ. Для регулярного детерминированного закона распределения $C = 0$ ($\sigma_{\bar{x}}$ =0).

Для системы G/G/1 среднее количество требований определяется как

$$\bar{N} = \rho + \frac{\rho^2\left(1 + C^2\right)}{2(1-\rho)}. \qquad (1.6)$$

Используя результат Хинчина-Полячека, можно получить среднее время пребывания в одноканальной СМО по формуле

$$T = \bar{x}\left[1 + \frac{\rho\left(1 + C^2\right)}{2(1-\rho)}\right]. \qquad (1.7)$$

Основной результат (1.7) состоит в том, что среднее время пребывания требования в системе зависит только от математического ожидания и стандартного отклонения времени обслуживания. Таким образом, время ожидания определяется как

$$w = \frac{\bar{x}\rho\left(1 + c^2\right)}{2(1-\rho)}. \qquad (1.8)$$

Обычно интересуются нормированным временем ожидания:

$$\frac{w}{\bar{x}} = \frac{\rho\left(1 + c^2\right)}{2(1-\rho)}. \qquad (1.9)$$

Для системы M/M/1

$$\frac{w}{\bar{x}} = \frac{\rho}{(1-\rho)}, \qquad (1.10)$$

для системы M/D/1

$$\frac{w}{\bar{x}} = \frac{\rho}{2(1-\rho)}. \qquad (1.11)$$

Таким образом, система с регулярным обслуживанием характеризуется средним временем ожидания вдвое меньшим, чем система с показательным обслуживанием. Это закономерно, поскольку время пребывания в системе и количество требований в ней пропорциональны дисперсии времени обслуживания.

1.3. Основы дискретно-событийного моделирования СМО

Определим основные понятия и термины, используемые в моделировании.

Система – множество объектов (например, людей и машин), которые взаимодействуют одновременно для достижения одной или большего количества целей.

Модель – абстрактное представление системы, обычно содержит структурные, логические или математические отношения, которые описывают систему в терминах состояний, объектов и их свойств, множеств, процессов, событий, действий и задержек.

Состояние системы – множество переменных, которые содержат всю информацию, необходимую для описания свойств системы в любое время.

Объект – любой элемент или компонент в системе, который должен быть представлен в модели в явном виде (например, обслуживающее устройство, клиент, машина).

Свойство или ***атрибут*** – свойства данного объекта (например, приоритет ожидающего клиента, маршрут процесса выполнения работ в цеху).

Список множество (постоянное или временное) связанных объектов, упорядоченное некоторым логическим способом (например, все клиенты, находящиеся в настоящее время в очереди ожидания, упорядочены по принципу «раньше прибыл, раньше обслужился» или по приоритету).

Событие – мгновенно возникающее изменение состояние системы (например, прибытие нового требования).

Уведомление о событии – запись события, которое произойдет в потоке событий или в некотором будущем времени наряду с любыми связанными данными, необходимыми для обработки события (запись всегда включает тип события и время события).

Список событий – список намеченных будущих событий, упорядоченных по времени возникновения, известный также как список будущих событий (СБС).

Действие – продолжительность времени указанного промежутка (например, время обслуживания или время между поступлениями заявок), для которого известно, когда оно начинается и заканчивается (хотя оно может быть определено в терминах статистического распределения).

Задержка – продолжительность времени неопределенного промежутка, для которого неизвестно заранее, когда он заканчивается (например, задержка клиента в очереди по правилу «последний пришел – первый обслужился», так как начало обслуживания зависит от будущих поступлений).

Модельное время неотрицательная возрастающая величина, отражающая течение времени в имитационной модели.

Часы – переменная, отражающая протекание времени моделирования, называется в примерах ЧАСЫ (CLOCK).

Дискретно-событийное моделирование – моделирование системы в дискретные моменты времени, когда происходят события, отражающие последовательность изменения состояний системы во времени. В дальнейшем такое моделирование будем называть ***имитационным.*** Рассматриваемые здесь системы динамические, то есть изменяются во времени. Поэтому состояние системы, свойства объекта и число активных объектов, параметров, действий и задержек – все они функции времени и постоянно изменяются в процессе моделирования.

Для СМО с одним устройством обслуживания событиями будут поступление требования и конец его обслуживания устройством. Начало обслуживания – это условное событие, которое зависит от состояния прибора (занят или свободен) и числа требований, находящихся в очереди. Задержку иногда называют ***условным ожиданием,*** в то время как действие называют ***безусловным ожиданием.*** Действиями будут время между поступлениями требований и время обслуживания прибором. Завершение действия – событие, часто называемое ***первичным событием,*** для управления которым в СБС помещается уведомление о событии. Напротив, управление задержками связано с помещением объекта в другой список, возможно представляющий очередь ожидания до такого времени, когда системные условия разрешат обработку требования. Окончание задержки иногда называют ***условным*** или ***вторичным событием,*** но такие события не представлены в соответствующих уведомлениях о событиях и не появляются в СБС.

Пример. Представим себе, что есть магазин, в котором один продавец и он же кассир. Если пришедший покупатель застает продавца свободным, то продавец немедленно приступает к его обслуживанию. Продавец переходит в состояние «занятый». Если продавец занят обслуживанием покупателя и в это время приходит другой (другие) покупатели, то они становятся в конец очереди к продавцу. Когда продавец закончил обслуживание, то он переходит в состояние свободный.

Если в очереди есть покупатели, то продавец «выбирает» для обслуживания первого покупателя из очереди. Очередь уменьшается на единицу. Все остальные покупатели в очереди, если они есть, продвигаются на одну позицию вперед. Если в очереди нет покупателей, то продавец остается в состоянии «свободный» до прихода следующего покупателя.

В табл. 1.1 представлены интервалы времени между приходами покупателей в магазин н требуемыми временами их обслуживания продавцом, **А** на рис. 1.3 проведено графическим методом «ручное» моделирование СМО с одним устройством.

С точки зрения объектного подхода имеются динамические объекты – ***требования*** (ПОКУПАТЕЛИ) и некоторый ***ресурс*** – устройство обслуживания (статический объект ПРОДАВЕЦ), которое они используют. Если требование претендует на ресурс, **А** он занят, то оно становится в ОЧЕРЕДЬ к ресурсу. ОЧЕРЕДЬ может быть отдельным объектом или просто списком, связанным с ресурсом. Примем за правило обслуживания – FIFO.

Таблица 1.1

№ покупателя	1	2	3	4	5	6	7	8
Время поступления, $t_j^{вх}$	2	2	7	3	2	3	1	2

Время обслуживания, $t_j^{об}$	4	3	6	5	4	з	з	2

Для каждой пары «требование – ресурс» мы хотим определить, как долго требование *j* будет использовать ресурс *R,* т.е. необходимо определить интервал времени, когда требованию *j* ***назначен*** ресурс *R* и когда оно ***освободит*** этот ресурс. Однако, прежде чем ресурс будет назначен требованию *j,* он должен быть ***запрошен***. В общем случае требование может ожидать в очереди до назначения ресурса.

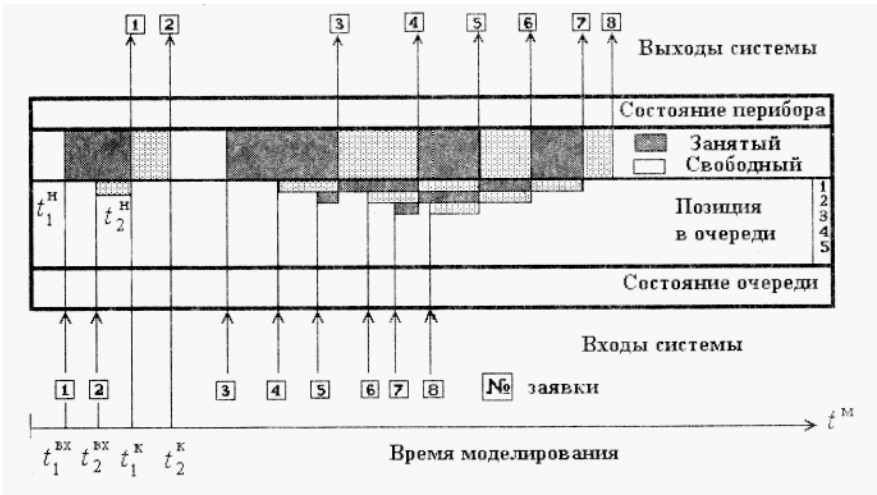


Рис. 1.3

Опишем алгоритм работы системы с точки зрения «жизненного цикла» ПОКУПАТЕЛЯ, т.е. от момента его прихода в магазин до момента выхода из магазина. Так как покупатели непрерывно приходят в магазин на протяжении некоторого периода времени наблюдения за системой (время, в течении которого моделируется система), то необходимо обеспечить поток ПОКУПАТЕЛЕЙ путем их создания в модели (генерации в некоторые моменты времени – моменты их прихода в магазин). Для генерации ПОКУПАТЕЛЕЙ используем специальную подпрограмму ГЕНЕРАТОР (в языке GPSS этой подпрограмме соответствует блок **GENERATE**). Алгоритм ее работы следующий:

1. Создать динамический объект ПОКУПАТЕЛЬ в виде структуры данных, включающей в себя поля: номер покупателя – *j,* момент его прихода – $t_j^{вх}$, **А** также при необходимости свойства покупателя или его атрибуты (например, его приоритет). Запланировать событие – приход покупателя *j* на момент времени $t_j^{вх}$ – т.е. создать уведомление о событии в СБС.

2. Запланировать следующее событие для покупателя *j* – ЗАПРОС-НАЗНАЧЕНИЕ ресурса *R* (ПРОДАВЦА) на момент времени $t_j^{вх}$. Запланировать приход следующего динамического объекта ПОКУПАТЕЛЬ *j*+ 1, т.е. определить событие прихода следующего покупателя $t_{j+1}^{вх} := t_j^{вх} + t_j^{об}$.

Описание процесса использования ресурса требованием целесообразно разбить на три подпрограммы.

Первая – это ЗАПРОС-НАЗНАЧЕНИЕ ресурса *R* требованию *j* (в языке GPSS этой подпрограмме соответствует блок **SEIZE).** Алгоритм ее работы следующий:

1. Если ресурс *R* (ПРОДАВЕЦ) может быть сразу назначен требованию *j*, то изменить состояние ресурса *R* на «занятый». Запомнить момент начала обслуживания требования $t_{i+1}^{н}$. Передать управление подпрограмме ОБСЛУЖИВАНИЕ требования *j*.

2. Если ресурс *R* занятый, то поставить требование *j* в ОЧЕРЕДЬ к ресурсу *R.*

Вторая подпрограмма – ОБСЛУЖИВАНИЕ требования *j* (в языке GPSS этой подпрограмме соответствует блок **ADVANCE**). Ее алгоритм работы очень простой – определить событие «конец обслуживания требования N_j как $t_j^{к} = t_j^{н} + t_j^{об}$ (где $t_j^{об}$ – время обслуживания в устройстве). Т.е. создается уведомление о событии в СБС для передачи управления подпрограмме освобождения ресурса *R* требованием *j*.

Третья подпрограмма – ОСВОБОЖДЕНИЕ ресурса *R* требованием j (в языке GPSS этой подпрограмме соответствует блок **RELEASE**). Алгоритм ее работы следующий.

1. Изменить состояние ресурса *R* (ПРОДАВЕЦ) на «свободный». Передать управление подпрограмме УНИЧТОЖЕНИЕ требования.

2. Проверить, есть ли требования в ОЧЕРЕДИ к ресурсу *R?* Если есть, то выбрать требование из ОЧЕРЕДИ и запланировать для него событие ЗАПРОС-НАЗНАЧЕНИЕ ресурса *R*.

Подпрограмма УНИЧТОЖЕНИЕ требования (в языке GPSS этой подпрограмме соответствует блок **TERMINATE**) необходима для уничтожения структуры данных каждого требования. Если требования не уничтожать, то со временем они переполнят память компьютера.

Кроме перечисленных подпрограмм необходима программа управления процессом моделирования (ПУМ), которая запускает процесс моделирования и отслеживает движение каждого требования по модели путем вызова названных подпрограмм обработки событий. Другое назначение этой программы – вести список упорядоченных во времени событий СБС и продвигать ЧАСЫ модельного времени от события к событию. В языке GPSS функции ПУМ выполняет программа-интерпретатор (транслятор).

Список будущих событий содержит все уведомления для событий, которые были намечены, чтобы произойти в будущем времени. Механизм продвижения времени моделирования, гарантирующий, что все события происходят в правильном хронологическом порядке, основан на СБС.

Планирование будущего события означает, что в момент начала действия его продолжительность вычисляется или определяется (например, из заданного статистического распределения), чтобы установить время конца действия и занести это событие вместе с его временем в СБС. В реальном мире большинство будущих событий не намечено, они просто происходят (как, например, случайные поломки оборудования или случайные поступления покупателей). В модели такие случайные события представлены концом некоторого действия, которое запланировано на будущее модельное время.

В любое данное время моделирования $t_1^{\text{м}}$ СБС содержит все предварительно намеченные будущие события и связанные с этими событиями времена $t_1^{\text{м}}$, $t_2^{\text{м}}$ .....В СБС события упорядочены в хронологическом порядке по времени, то есть времена событий удовлетворяют условиям

$$t^{\text{м}} < t_1^{\text{м}} < t_2^{\text{м}} < t_3^{\text{м}} < \cdots < t_n^{\text{м}}. \qquad (1.12)$$

Время $t^{\text{м}}$ значения ЧАСОВ – текущее значение времени моделирования. Событие, связанное со временем $t_1^{\text{м}}$, называется предстоящим событием, то есть это следующее событие, которое произойдет. После того, как отображающие состояния системы ЧАСЫ = $t^{\text{м}}$ во время моделирования были модифицированы, ЧАСЫ продвигаются ко времени моделирования ЧАСЫ = $t_1^{\text{м}}$, предстоящее намеченное событие удаляется из СБС и выполняется подпрограмма события. Выполнение подпрограммы предстоящего события означает, что отображено новое состояние системы в течение времени $t_1^{\text{м}}$, которое создано на основании старого состояния модели во время $t^{\text{м}}$ и характера предстоящего события. Во время $t^{\text{м}}$ новые будущие события могут произойти или не произойти, но если любые из них намечены, то создаются намеченные события и помещаются в соответствующие позиции в СБС. После того, как новое отображение состояния системы в течение времени $t_1^{\text{м}}$ было модифицировано, ЧАСЫ продвигаются ко времени нового предстоящего события, и выполняется подпрограмма этого события. Такой процесс повторяется до окончания имитации. На рис. 1.4 показана структурная схема имитационной модели.

В момент начала моделирования (время моделирования $t_0^{\text{м}}$=о) ПУМ передает управление подпрограмме ГЕНЕРАТОР, которая определяет момент прихода первого покупателя и намечает событие ЗАПРОС-НАЗНАЧЕНИЕ ресурса *R* в СБС на время $t_1^{\text{м}} = t_1^{\text{вх}}$. Так как больше событий в системе нет, то ЧАСЫ переводятся на значение времени $t_1^{\text{м}}$, вызывается подпрограмма ГЕНЕРАТОР и подпрограмма ЗАПРОС-НАЗНАЧЕНИЕ ресурса *R*.

Подпрограмма ГЕНЕРАТОР определяет будущее событие (момент прихода второго покупателя $t_2^{\text{вх}}$) и намечает это событие в СБС на время $t_2^{\text{вх}}$.

Подпрограмма ЗАПРОС-НАЗНАЧЕНИЕ проверяет состояние ресурса (ПРОДАВЦА). Так как ресурс *R* свободный, то он назначается первому покупателю и состояние ресурса *R* изменяется на «занятый». Запоминается момент начала обслуживания требования $t_1^{\text{н}}$. Передается управление подпрограмме ОБСЛУЖИВАНИЕ покупателя 1.

Подпрограмма ОБСЛУЖИВАНИЕ определяет событие конца обслуживания покупателя 1, как $t_1^{к} = t_1^{н} + t_1^{об}$, т.е. создается уведомление о будущем событии в СБС для передачи управления подпрограмме ОСВОБОЖДЕНИЯ ресурса *R* требованием 1 на время $t_1^{к}$.

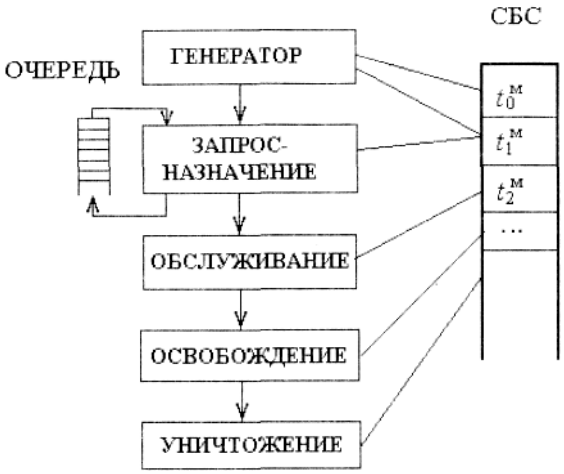


Рис. 1.4

Таким образом, в СБС имеется два элемента – один с намеченным событием появления покупателя 2 на время $t_2^{вх}$, ***A*** второй с намеченным событием окончания обслуживания покупателя 1 на время $t_1^{К}$. Если время $t_2^{вх} < t_1^{к}$, то ЧАСЫ будут переведены на время $t_2^{м} = t_2^{вх}$, т. е. снова будет вызвана подпрограмма ГЕНЕРАТОР и сгенерировано появление покупателя 2. В этот же момент времени $t_2^{м}$ произойдет вызов подпрограммы ГЕНЕРАТОР, которая наметит в СБС появление покупателя 3 на время $t_3^{вх}$ и вызов подпрограммы ЗАПРОС-НАЗНАЧЕНИЕ ресурса *R,* но так как ресурс занят обслуживанием покупателя 1, то покупатель 2 будет поставлен в очередь к ресурсу *R.*

В СБС снова окажутся два элемента – один для намеченного события появления покупателя 3 на время $t_3^{вх}$, **A** второй с намеченным событием окончания обслуживания покупателя 1 на время $t_1^{г}$. Если время $t_3^{вх} > t_1^{г}$, то ЧАСЫ будут переведены на время $t_3^{м} = t_1^{к}$, т.е. – будет вызвана подпрограмма ОСВОБОЖДЕНИЕ ресурса *R* покупателем 1. Она изменить состояние ресурса *R* (ПРОДАВЕЦ) на «свободный» и передаст управление подпрограмме УНИЧТОЖЕНИЕ требования. Затем проверит, есть ли требования в ОЧЕРЕДИ к ресурсу *R,* и выберет покупателя 2 из ОЧЕРЕДИ, наметив для него событие для подпрограммы ЗАПРОС-НАЗНАЧЕНИЕ ресурса *R.*

Вызванная в это же модельное время $t_3^{м}$ подпрограмма УНИЧТОЖЕНИЕ разрушит структуру данных (удалит ссылку на адрес) для покупателя 1. Эта же подпрограмма при необходимости могла бы вычислить время пребывания покупателя 1 в системе, как $t_3^{пр} = t_3^{м} - t_1^{вх}$, для дальнейшей статистической оценки времени пребывания покупателей в системе.

В дальнейшем жизненный цикл других покупателей в системе будет происходить по описанному выше алгоритму.

Остается открытым вопрос об окончании процесса моделирования. Возможны три варианта:

1. Через модель пройдут все покупатели, сгенерированные ГЕНЕРАТОРОМ, например, 100 покупателей. В этом случае в СБС после обслуживания последнего, сотого, покупателя не будет ни одного намеченного события.

2. Если поток покупателей от генератора не ограничен (например, генерируется неограниченный пуассоновский поток), то моделирование можно закончить после прохождения через модель определенного количества покупателей, например, 1000. Для этого в подпрограмме УНИЧТОЖЕНИЕ надо поставить счетчик покупателей и прекратить моделирование после 1000 покупателей. В языке GPSS такой счетчик организуется в команде START, которая начинает процесс моделирования.

3. Необходимо промоделировать работу системы в течение заданного периода времени, например, 480 мин. В этом случае можно при каждом продвижении ЧАСОВ модельного времени *t* проводить сравнение текущего времени со значением 480. Как только значение модельного времени будет больше или равно 480, необходимо прекратить моделирование. Однако такой способ неудачный, так как может сильно замедлить работу модели из-за проверки условия. Поэтому обычно поступают следующим образом. Генерируют специальное требование-таймер с помощью еще одной подпрограммы

ГЕНЕРАТОР с намеченным временем входа в модель t^m = 480. Требование-таймер после генерации сразу же направляется в еще одну подпрограмму УНИЧТОЖЕНИЯ, в которой ставят счетчик требований на единицу. По этому счетчику прекращают моделирование. В этом случае в СБС будет все время находиться элемент для требования-таймера со временем наступления события 480. Как только это событие станет предстоящим намеченным, ЧАСЫ будут переведены на время 480 и моделирование прекратится.

В процессе моделирования обычно собирается статистическая информация о работе модели при каждом продвижении ЧАСОВ модельного времени. Такой информацией может быть величина очереди, время пребывания в очереди и устройстве обслуживания, загрузка устройства, состояние прибора и другие параметры. Для сбора этой информации обычно создается подпрограмма ВЫБОРОЧНЫЙ ИЗМЕРИТЕЛЬ, которая накапливает ее и по окончании моделирования выдает стандартный статистический отчет. В языке GPSS такая статистическая информация накапливается в системных числовых атрибутах (СЧА) и доступна в процессе моделирования только на считывание. Доступ к СЧА дает возможность управлять процессом движения требований, например, ограничивать размер или время нахождения в очереди.

В этой главе даны основы организации моделирования на примере простой СМО. В языке GPSS обычно используются более сложные алгоритмы, описанные в параграфе 4.22.

1.4. Многоканальные системы массового обслуживания

Многоканальная СМО (с несколькими одинаковыми устройствами обслуживания) изображена на рис. 1.5. В отличие от одноканальных СМО многоканальные системы рассчитать сложнее. Теория массового обслуживания позволяет получать аналитические зависимости для расчетов характеристик работы многоканальных СМО в стационарном режиме работы, однако, эти зависимости можно получить только для системы M/M/m.

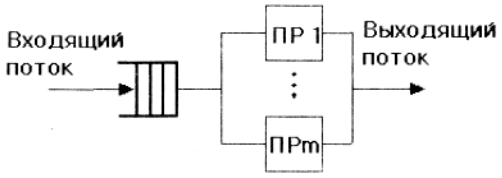


Рис. 1.5

Если система имеет *m* одинаковых устройств, то

$$\rho = \frac{\lambda \bar{x}}{m} . \qquad (1.13)$$

Для многоканальных СМО ρ можно трактовать, как математическое ожидание части занятых устройств.

Рассмотрим диаграмму работы многоканальной СМО (рис. 1.6) с двумя устройствами (ПР 1 и ПР 2) и двумя позициями для ожидания в очереди (Поз. 1 и Поз. *2)*. Время поступления и время, когда требование покинуло систему, показаны рядом с номером требования в нижней и верхней частях рис. 1.6, соответственно. Время наблюдения за СМО *($T_н$)* составляет 55 мин.

Рассчитаем по диаграмме некоторые оценки характеристик работы СМО.

1. Вероятность обслуживания требования

$$P_{об} = \frac{N_{об}}{N} = \frac{10}{12} = 0{,}83,$$

где $N_{об}$, N – количество обслуженных требований и общее количество требований, соответственно.

2. Пропускная способность СМО в требованиях в минуту

$$X = \frac{N_{об}}{T_н} = \frac{10}{55} = 0{,}18 ,$$

где $T_н$ – время наблюдения за системой.

3. Вероятность отказа в обслуживании

$$P_{\text{отк}} = \frac{N_{\text{отк}}}{N} = \frac{2}{12} = 0{,}166,$$

где $N_{отк}$ – количество требований, которым отказано в обслуживании.

4. Вероятность того, что требование застанет оба устройства свободными,

$$P_0 = \frac{T_{\text{св}}}{T_{\text{н}}} = \frac{3}{55} = 0{,}05,$$

где $T_{св}$ – время, на протяжении которого оба устройства были свободными.

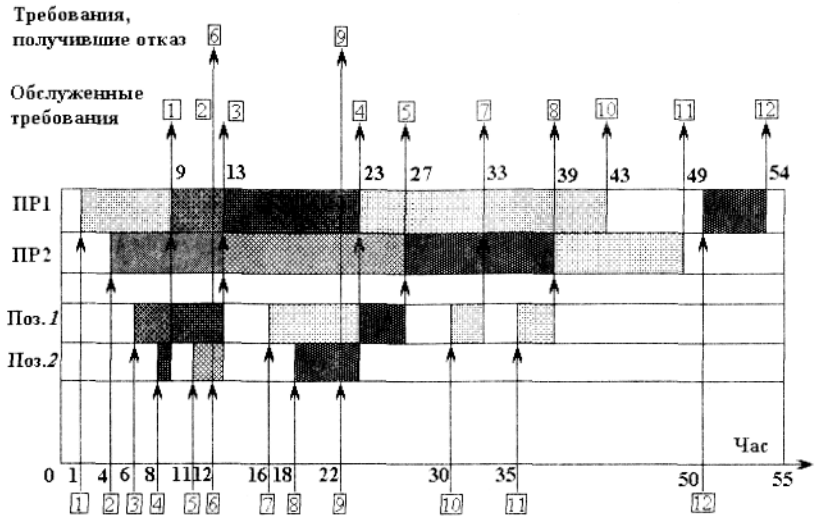


Рис. 1.6

5. Вероятность того, что обслуживанием занято только одно устройство из двух,

$$P_1 = \frac{T^1{}_{\text{з}} + T^2{}_{\text{з}}}{T_{\text{н}}} = \frac{7+6}{55} = 0{,}236,$$

где $T^1{}_{\text{з}}$, $T^2{}_{\text{з}}$, – время, когда было занято только первое и только второе устройство, соответственно.

6. Вероятность того, что обслуживанием заняты оба устройства,

$$P_2 = \frac{T^{1+2}{}_{\text{з}}}{T_{\text{н}}} = \frac{39}{55} = 0{,}709,$$

где $T^{1+2}{}_{\text{з}}$ – время, когда были занятые оба устройства.

7. Среднее количество занятых устройств

$$N_{\text{пр}} = 0 \cdot P_0 + 1 \cdot P_1 + 2 \cdot P_2 = 1 \cdot \frac{13}{55} + 2 \cdot \frac{39}{55} = 1{,}654.$$

8. Вероятность того, что в очереди нет требований,

$$P^0_{\text{чер}} = \frac{T^0_{\text{оч}}}{T_{\text{н}}} = \frac{(6-0)+(16-13)+(30-27)+(35-33)+(55-39)}{55} = \frac{30}{55} = 0{,}545,$$

где $T^0{}_{\text{оч}}$ – время, на протяжении которого в очереди не было требований.

9. Вероятность того, что в очереди есть только одно требование,

$$P^1_{\text{оч}} = \frac{T^1_{\text{оч}}}{T_{\text{н}}} = \frac{(8-6)+(11-9)+(18-16)+(27-23)+(33-30)+(39-35)}{55} =$$

$$= \frac{17}{55} = 0{,}309.$$

где $T^1{}_{оч}$ – время, когда в очереди было только одно требование.

10. Вероятность того, что в очереди два требования,

$$P^2_{\text{оч}} = \frac{T^2_{\text{оч}}}{T_{\text{н}}} = \frac{(9-8)+(13-11)+(23-18)}{55} = \frac{8}{55} = 0{,}145.$$

где $T^2_{оч}$ – время, на протяжении которого в очереди было два требо вания.

11. Среднее количество требований в очереди

$$N_{оч} = 0 \cdot P^0_{оч} + 1 \cdot P^1_{оч} + 2 \cdot P^2_{оч} = 0 + 1 \cdot \frac{17}{55} + 2 \cdot \frac{8}{55} = 0{,}6 .$$

12. Среднее время пребывания в очереди

$$t_{оч} = \frac{\sum_{i=1}^{10} t_i^{оч}}{N_{об}} = \frac{0+0+3+5+2+7+9+3+11+0}{10} = \frac{40}{10} = 4 \text{ мин},$$

где $t_i^{оч}$ – время пребывания i – го требования в очереди (i= 1,2,...).

13. Среднее время пребывания в очереди без учета требований, которые не ждали,

$$t_{оч} = \frac{\sum_{i=1}^{7} t_i^{оч}}{N_{об}(-0)} = \frac{3+5+2+7+9+3+11}{7} = \frac{40}{7} = 5{,}714 \text{ мин},$$

где $N_{об}(-0)$ – количество требований, которые не ждали в очереди.

14. Среднее время обслуживания требования в устройствах

$$t_{об} = \frac{\sum_{i=1}^{10} t_i^{об}}{N_{об}} = \frac{92}{10} = 9{,}2 \text{ мин},$$

где $t_i^{об}$ – время обслуживания i – го требования в СМО (i = 1,2,...).

15. Общее среднее время пребывания требования в СМО

$$T = t_{об} + t_{оч} = 4 + 9{,}2 = 13{,}2 \text{ мин}.$$

16. Среднее количество требований в системе обслуживания

$$\bar{N} = N_{пр} + N_{оч} = 1{,}654 + 0{,}6 = 2{,}254 .$$

На рис. 1.7 изображена гистограмма для времени поступления требований в СМО и аппроксимация ее экспоненциальным законом распределения. Из гистограммы видно, что количество требований, которое поступило в систему, недостаточно для статистической оценки. Поэтому гипотезу про экспоненциальный закон распределения поступления требований в СМО необходимо отклонить.

Рассчитанные числовые значения характеристик имеют иллюстративный характер и позволяют определиться, каким образом необходимо собирать статистические данные о работе СМО при ее моделировании.

Рис. 1.7

Приведем основные формулы для расчетов СМО вида M/M/m [7].

1. Вероятность того, что все устройства обслуживания свободны,

$$P_0 = \frac{1}{\sum_{k=0}^{m-1} \frac{(\lambda \bar{x})^k}{k!} + \frac{(\lambda \bar{x})^m}{(m-1)!\,(m - \lambda \bar{x})}}, \text{ при } \lambda \bar{x} < 1. \qquad (1.14)$$

2. Вероятность того, что занято обслуживанием k-е устройство или в системе находится k требований,

$$P_k = \frac{(\lambda \bar{x})^k}{k!} P_0,\ 1 \le k < m. \qquad (1.15)$$

3. Вероятность того, что все устройства заняты *(k≥m*). Обозначим эту вероятность через π :

$$\pi = \frac{(\lambda \bar{x})^m P_0}{(m-1)!(m-\lambda \bar{x})},\ \frac{\lambda \bar{x}}{m} < 1. \qquad (1.16)$$

4. Вероятность того, что все устройства заняты обслуживанием и *s* требований находятся в очереди,

$$P_{m+s} = \frac{(\lambda \bar{x})^{m+s}}{m!m^s} P_0\ ,\ s > 0 \qquad (1.17)$$

5. Вероятность того, что время пребывания требований в очереди превышает некоторую величину *t*,

$$P(w > t) = \pi e^{-\frac{1}{\bar{x}}(m-\lambda \bar{x})t}. \qquad (1.18)$$

6. Средняя длина очереди

$$\bar{N}_q = \frac{\lambda \bar{x} P_m}{m\left(1-\frac{\lambda \bar{x}}{m}\right)^2}. \qquad (1.19)$$

7. Среднее количество свободных от обслуживания устройств

$$N_{\text{св}} = \sum_{k=0}^{m-1} \frac{m-k}{k!} (\lambda \bar{x})^k P_0. \qquad (1.20)$$

8. Среднее количество занятых обслуживанием устройств

$$N_{\text{пр}} = m - N_{\text{св}}. \qquad (1.21)$$

9. Среднее время ожидания требованием начала обслуживания в системе

$$w = \frac{\pi \bar{x}}{m - \lambda \bar{x}},\ \frac{\lambda \bar{x}}{m} < 1. \qquad (1.22)$$

Приведенные формулы позволяют выполнять расчеты для СМО вида M/M/m и сравнивать их с полученными результатами имитационного моделирования.

ГЛАВА 2. ВЕРОЯТНОСТНЫЕ СЕТИ СИСТЕМ МАССОВОГО ОБСЛУЖИВАНИЯ

2.1. Общие сведения о сетях

В общем случае сеть СМО можно представить в виде графа, вершинами которого являются одноканальные и многоканальные СМО (дуги определяют потоки передачи требований).

Простейшая разомкнутая или открытая сеть получается при последовательном соединении СМО (рис.2.1). Она еще называется многофазной СМО.

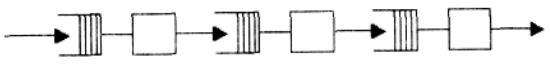


Рис. 2.1

Различают замкнутые и разомкнутые сети. Для замкнутой вероятностной сети не существует внешних источников требований, то есть в ней всегда находится одно и то же количество требований. Для разомкнутой сети имеются источники требований и стоки требований.

Простейшая замкнутая сеть показана на рис. 2.2. Эта система с отказами и восстановлениями хорошо известна из теории массового обслуживания. В системе постоянно находятся M1 требований, которые появляются при отказе устройств *M.* Если устройство отказало, то поступает требование на его ремонт к бригаде с *N* ремонтниками, которые ремонтируют устройство, а потом отремонтированное устройство восстанавливает свою работу. На рис. 2.2 это показано обратной связью от N_1 устройств. Сеть также используется при моделировании компьютерной системы, которая работает в режиме «запрос – ответ», то есть пользователь не посылает новый запрос к системе до тех пор, пока не получит ответ на предшествующий запрос. Запросы обрабатываются любым из N1 компьютеров. Примерами таких

систем могут быть автоматизированные системы продажи билетов на поезда или самолеты, системы передачи транзакций от кассиров в банке и т. п.

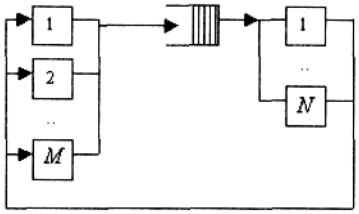


Рис. 2.2

Сеть (рис. 2.3) содержит K узлов и N требований, которые находятся в сети. Каждый узел может иметь одно или несколько одинаковых устройств обслуживания. С вероятностью (или частотой) q_{0j} – требования поступают к любому узлу сети, а с вероятностью q_{kj} (j = 1,..., *K)* требование, которое оставляет узел *k,* направляется к узлу *j*. Таким образом, любое требование до завершения своего обслуживания в сети обычно проходит несколько узлов.

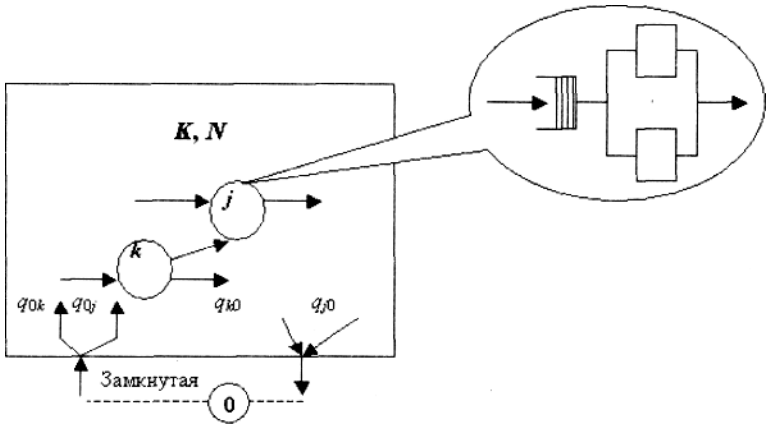


Рис. 2.3

Внешняя среда обозначается как узел 0 сети. Если сеть замкнутая, то требования с выхода направляются на вход (рис. 2.3, пунктирная линия) и количество требований *N* в сети не изменяется.

Для потоков требований в сети справедливы законы о суммарных потоках, которые показаны на рис. 2.4, 2.5, при условии, что сеть работает в установившемся режиме.

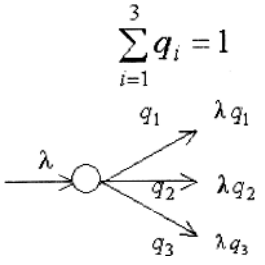


Рис. 2.4

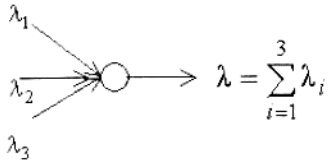


Рис. 2.5

Для расчетов сетей массового обслуживания используется теория вероятностных сетей, которая основывается на марковских и полумарковских процессах [8], но большинство результатов получено только для экспоненциальных законов распределения. При количестве узлов сети больше трех для расчетов используются численные приближенные методы. Операционный анализ [9] в отличие от теории массового обслуживания опирается на логику работы рассматриваемой или моделируемой системы. Это позволяет установить простые зависимости между параметрами и показателями работы системы, не абстрагируясь от процессов ее функционирования.

2.2. Операционный анализ вероятностных сетей

Операционный анализ вероятностных сетей базируется на следующих принципах:

– все предположения относительно операционных переменных можно проверить измерениями на реальной системе или на ее модели;

– в системе должен существовать баланс потоков: количество требований, которые покинули систему за некоторый период наблюдения, должно равняться количеству требований, которые поступили в систему за этот же период;

– переходы требований от одного узла к другому не должны зависеть от длин очередей в узлах.

Таким образом, рассматриваемая система должна работать в установившемся, **А** не в переходном режиме.

Основная задача операционного анализа вероятностных сетей состоит в определении таких показателей, как среднее время пребы-

вания требований в отдельных узлах сети, загрузка устройств в узлах, средние длины очередей к узлам и т.п.

Большинство результатов операционного анализа касается замкнутых сетей, когда требования, которые покидают сеть, снова возвращаются в нее. Замкнутые сети можно использовать, когда рассматриваемая система работает с перегрузкой. В этом случае можно считать, что вместо требования, которое покинуло систему, в систему поступает другое требование с такими же параметрами.

Введем операционные переменные, которые можно получить или измерениями, или в процессе имитационного моделирования системы:

$q_{0j}\ \left(j=\overline{1,K}\right)$ – вероятность (частота) поступления требований в сеть извне к любому узлу (K-общее количество узлов);

$q_{kj}\ \left(j=\overline{1,K}\right)$ – вероятность перехода требований из узла *k* к узлу j $\left(k=\overline{1,K}, j=\overline{1,K}\right)$

q_{k0} – вероятность того, что после окончания обслуживания в узле k требования покинут сеть;

$A_k\ \left(k=\overline{1,K}\right)$ – количество требований, которые поступили в узел *k;*

$C_{kj}\ \left(k=\overline{1,K},\ j=\overline{1,K}\right)$ – количество требований, которые покинули узел k и поступили в узел];

$B_k\ \left(k=\overline{1,K}\right)$ – общее время обслуживания требований узлом *k.*

T – общее время наблюдения за системой или время моделирования.

Внешнюю среду обозначим как вершину с номером 0. Тогда A_{0i}, C_{k0} будут приобретать значения количества требований, которые поступили в узел *j*, и требований, которые покинули узел *k,* соответственно.

Узел считается занятым, если в нем есть хотя бы одно требование. Введем дополнительные обозначения:

$$C_k=\sum_{j=1}^{K}C_{kj},\ A_0=\sum_{j=1}^{K}A_{0j},\ C_0=\sum_{j=1}^{K}C_{j0}. \qquad (2.1)$$

Для замкнутой сети $A_0 = C_0$.

Введенные переменные называются ***основными операционными переменными.*** Используя эти переменные и выполняя простейшие операции над ними, получают ***выводимые операционные переменные.*** Наиболее часто используют такие:

$$U_k=\frac{B_k}{T}, \qquad (2.2)$$

где U_k – коэффициент использования узла;

$$S_k = \frac{B_k}{C_k}, \qquad (2.3)$$

где S_k – среднее время обслуживания в узле *k;*

$$X_k = \frac{C_k}{T}, \qquad (2.4)$$

где X_k – интенсивность выходящего потока требований из узла *k;*

$$q_{kj} = \begin{cases} \frac{C_{kj}}{C_k}, & k = \overline{1,K}, \\ \frac{A_{0j}}{A_j}, & k = 0, \end{cases} \qquad \sum_{k=1}^{K} q_{kj} = 1, \qquad (2.5)$$

где q_{kj} – относительная частота перехода требований между узлами *k* и j.

Используя выражения (2.2 – 2.4), имеем:

$$U_k = X_k S_k \qquad (2.6)$$

2.3. Операционные зависимости

Основные результаты операционного анализа формулируются в виде соотношений между операционными переменными. Основой этих соотношений является гипотеза о балансе потоков в сети: ***количество требовании, которые поступили в некоторый узел на протяжении продолжительного периода T, равняется количеству требований, которые покинули этот узел.*** Эта гипотеза определяет работу сети СМО в установившемся режиме, то есть требования всегда покидают узлы сети.

Гипотеза о балансе позволяет установить зависимости между операционными переменными для каждого узла сети. Эта гипотеза позволяет записать уравнения баланса потоков:

$$X_j = \sum_{k=0}^{K} X_k q_{kj}, \; j = \overline{0,K}. \qquad (2.7)$$

Справедливость выражения (2.7) вытекает из предположения о балансе потоков в сети, то есть $A_j = Cj$, так как $\sum_{k=0}^{k} C_{kj} = C_j = A_j, j = \overline{0,K}$, но при условии, что $q_{kj} = \frac{C_{kj}}{C_k}$, находим $C_j = \sum_{k=0}^{k} C_k q_{kj}$. Поделив последнее соотношение (левую и праву его части) на общее время наблюдения *T,* получим выражение (2.7). Уравнения (2.7) будут иметь единственное решение для замкнутой сети при заданном x_0. Для разомкнутой сети уравнения (2.7) будут линейно зависимыми, однако, и в этом случае они имеют полезную информацию о динамике потоков сети. Найдем из выражения (2.6) производительность узла

$$X_k = \frac{U_k}{S_k}. \qquad (2.8)$$

Определим коэффициент посещаемости узла *k*

$$V_k = \frac{X_k}{X_0}. \qquad (2.9)$$

Уравнение баланса потока можно представить в эквивалентной системе, в которой вместо интенсивности потоков используются коэффициенты посещаемости каждого узла сети.

Поделим левую и правую части выражения (2.7) на X_0:

$$V_0 = 1, \quad V_j = q_{0j} + \sum_{k=1}^{K} V_k q_{kj}, \quad j = \overline{1,K}. \qquad (2.10)$$

Выражения(2.10) справедливы, если справедливы уравнения (2.7), поскольку (2.10) получены из (2.7).

Связь коэффициентов посещаемости и производительности узла определяем по формуле

$$\frac{X_k}{X_j} = \frac{V_k}{V_j}. \qquad (2.11)$$

Для определения среднего времени пребывания требования в вероятностной сети обозначим это время через *R, A* для отдельных узлов – через R_k. Введем еще одну операционную переменную – W_k, которая равняется суммарному времени ожидания и времени обслуживания требования узлом *k* на протяжении времени *T:*

$$R_k = \frac{W_k}{C_k}. \qquad (2.12)$$

Среднее время пребывания в системе можно найти через R_k и коэффициенты посещаемости отдельных узлов, то есть

$$R = \sum_{k=1}^{K} V_k R_k \ . \qquad (2.13)$$

Это общий ***закон времени пребывания,*** который справедлив и в том случае, если гипотеза о балансе потоков не выполняется.

Среднее количество требований в сети N, которое определяется через среднее количество требований в каждом узле n_k, равно

$$N = \sum_{k=1}^{K} n_k \ , \qquad (2.14)$$

где n_k – выводимая операционная переменная, которую можно получить из основных операционных переменных:

$$n_k = \frac{W_k}{T} \ . \qquad (2.15)$$

Для среднего времени пребывания требований в сети справедлив ***закон Литтла:*** среднее время пребывания в устройстве k определяется через среднее количество требований в устройстве и интенсивность потока

$$R_k = \frac{n_k}{X_k} \ . \qquad (2.16)$$

Обосновать формулу Литтла можно с помощью операционного анализа. Из выражения (2.15) находим:

$$W_k = n_k T \ . \qquad (2.17)$$

Подставляем полученную операционную переменную в уравнение(2.12):

$$R_k = \frac{n_k T}{C_k} = n_k / (C_k / T) = \frac{n_k}{X_k} \ . \qquad (2.18)$$

Закон Литтла справедлив также для всей сети в целом. Подставим выражение для V_k из уравнения (2.9) в (2.13) и выражение для R_k из (2.16), тогда

$$R = \sum_{k=1}^{K} \frac{n_k}{X_k} \frac{X_k}{X_0} = \frac{1}{X_0} \sum_{k=1}^{K} n_k = \frac{N}{X_0} \ . \qquad (2.19)$$

Покажем, как можно использовать операционный анализ для определения времени пребывания в замкнутой сети (рис. 2.6).

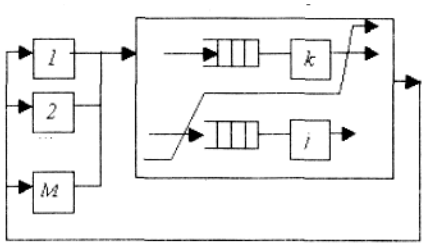


Рис. 2.6

Пусть есть M устройств, время обслуживания требования любым из них – Z. Среднее время пребывания требования в сети определяем по формуле

$$R = \frac{M}{X_0} - Z \ . \qquad (2.20)$$

Выражение (2.20) получено из таких соображений. Среднее время одного цикла взаимодействия, включая время обслуживания требования во внешней сети и пребывание в одном из M устройств, определяется суммой Z + R. Если предположить, что выполняется гипотеза о балансе потоков, то для рассматриваемого цикла справедлива формула Литтла. Поэтому величина $(Z + R)X^0$ должна определять

среднее количество занятых устройств или среднее количество работающих устройств для системы с отказами. Таким образом, общее количество устройств

$$M = (Z + R)X_0 \ . \qquad (2.21)$$

Продемонстрируем использование приведенных соотношений операционного анализа на примерах.

Пример 2.1. Пусть имеем M= 20 устройств. Среднее время обслуживания каждым Z = 25 с (рис. 2.7).

Для узлов l, *g, n* сети частоты перехода к узлу *t* равняются соответственно: q_{lt}=*0,5;* q_{mt}=*0,1;* q_{nt}= 0,85, **А** коэффициенты посещаемости этих узлов равняются V_l = 12; V_g = 17; V_n = 19. Узел *t* используется на 50%, среднее время обслуживания узлом *t* поступающих требований составляет 25 мс. Необходимо найти среднее время пребывания и среднее количество требований в сети.

Определим коэффициент посещаемости узла *t,* используя уравнения баланса потоков (2.10), записанные через коэффициенты посещаемости узлов:

$$V_t = V_l\, q_{lt} + V_g\, q_{gt} + V_n\, q_{nt};$$

$$V_t = 12 \cdot 0{,}5 + 17 \cdot 0{,}7 + 19 \cdot 0{,}85 = 34{,}05.$$

Находим интенсивность поступления требований в сеть

$$X_0 = \frac{X_t}{V_t} = \frac{U_t}{V_t S_t} \ . \qquad (2.22)$$

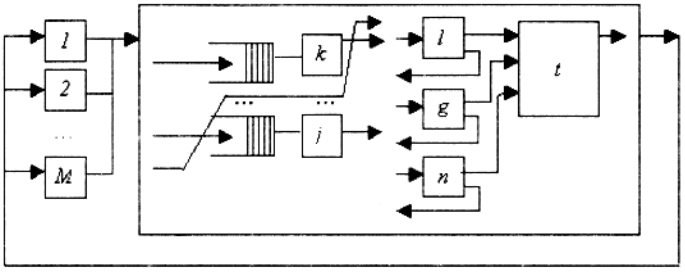


Рис. 2.7

В выражение (2.22) входят известные из условий операционные переменные: U_t = 50% и S_t = 0,025 с. Следовательно получим

$$X_0 = \frac{0{,}5}{34{,}05 \cdot 0{,}25} = 0{,}587 \ \text{требований/с}.$$

Из выражения(2.19) находим время пребывания требования в сети

$$R = \frac{20}{0{,}587} - 25 = 9{,}072 \ \text{с}.$$

Для определения среднего количества требований в сети воспользуемся формулой Литтла:

$$N = RX_0;$$

$$N = 9{,}072 \cdot 0{,}587 = 5{,}33 \ \text{требований}.$$

Пример 2.2. Рассмотрим сеть, в которую поступают требования как из обслуживающих устройств (замкнутая часть сети), так и извне (рис. 2.8).

Есть M= 40 обслуживающих устройств. Среднее время обслуживания каждым Z= 15 с. В результате проведенных исследований получены такие данные о сети:

– среднее время пребывания требований, которые поступают от 40 устройств обслуживания в сеть, равняется 5 с;

– среднее время обслуживания любого требования узлом *t* составляет 40 мс;

– каждое требование, которое поступает от M устройств обслуживания, порождает 10 требований к узлу *t;*

– каждое требование, которое поступает в систему извне, порождает 5 требований к узлу ?;

– узел *t* используется на 90%.

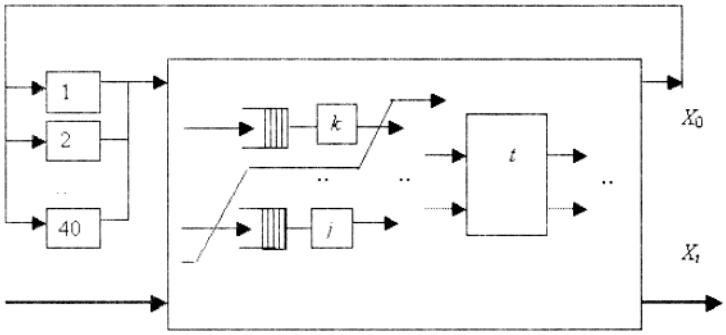


Рис. 2.8

Нужно определить нижнюю границу времени пребывания в сети требований, которые поступают от M устройств обслуживания с интенсивностью входящего потока X_0 и от внешнего источника требований в сеть с интенсивностью X_t, что выходят из узла t.

При решении поставленной задачи переменные, которые касаются поступающих от M устройств обслуживания требований, будем обозначать звездочкой.

Из выражения (2.20) для потока требований от М устройств находим

$$X_0^* = \frac{M}{Z + R^*} . \qquad (2.23)$$

где Z – среднее время обслуживания M устройствами; R – среднее время пребывания требований, которые поступили от 40 устройств обслуживания в сеть. Тогда

$$X_0^* = \frac{40}{15+5} = 2 \text{ требования/с} .$$

Интенсивность потока требований в узел t определяем как сумму

$$X_t^* + X_t = \frac{U_t}{S_t} \qquad (2.24)$$

интенсивности потоков требований от устройств обслуживания и интенсивности потока внешних требований, то есть $X_t^* + X_t$. Тогда в соответствии с выражением (2.3.1) можно записать:

$$X_t^* + X_t = \frac{0,9}{0,04} = 22,5 \text{ требований/с} .$$

Используя формулу для коэффициента посещаемости (2.8), находим X_t^* :

$$X_t^* = V_t^* X_0^*;$$

$$X_t^* = 10 \cdot 2 = 20 \text{ требований/с},$$

отсюда

$$X_t = 2,5 \text{ требований/с} .$$

Теперь можно найти интенсивность X_0 входящего потока внешних требований в сеть

$$X_0 = \frac{X_t}{V_t},$$

$$X_0 = \frac{2,5}{5} = 0,5 \text{ требований/с}.$$

Допустим, что исходные условия изменились и интенсивность входящего потока внешних требований увеличилась втрое, то есть $X_0 = 1,5$ требований/с. Тогда $X_t = V_t X_0 = 7,5$ требований/с. Считая, что среднее время обработки требований узлом t не изменилось, получаем, что максимально возможная интенсивность обслуживания требований узлом t, составляет – $\frac{1}{S_t} = 25$ требований/с при 100% использовании узла t. Таким образом, интенсивность обслуживания требований узлом t от устройств обслуживания не может превышать

$$25 - 7,5 = 17,5 \text{ требований/с}.$$

Исходя из этого,

$$X_0^* = \frac{X_t^*}{V_t^*} \le \frac{17,5}{10} = 1,75 \text{ требований/с}.$$

Итак, нижняя граница времени пребывания в сети требований, которые поступают от 40 устройств обслуживания в соответствии с выражением (2.19)

$$R^* = \frac{M}{X_0^*} - Z \geq \frac{40}{1{,}75} - 15 = 7{,}9 \text{ с.}$$

Таким образом, увеличение в три раза интенсивности потока внешних требований приведет к увеличению среднего времени пребывания требований в сети от 40 устройств обслуживания на 2,9 с.

2.4. Анализ узких мест в сети

Поиск узких мест в сети является важным аспектом анализа ее работы. Узкое место создается тем узлом сети, у которого коэффициент загрузки U приближается к единице. В этом узле образуется большая очередь, которая при $U \geq 1$ становится бесконечной, и сеть переходит в неустойчивый режим работы. Такой узел становится «насыщенным» требованиями. Узкие места в сети обусловливают ее пропускную способность, то есть полностью определяют время пребывания в сети. Поэтому при анализе работы сети необходимо особое внимание уделять поиску узких мест.

Покажем на простом примере, почему узкое место определяет пропускную способность сети. Рассмотрим трубопровод, в котором есть трубы разного диаметра, доставляющие воду потребителю. Если после трубы маленького диаметра поставить трубу с любым большим диаметром, то потребитель не получит большего количества воды за единицу времени, чем ее может пропустить узкая труба. Это – так называемый эффект «узкого горлышка». Поэтому при рассмотрении таких систем важно иметь сбалансированные потоки в сети, то есть такой баланс потоков в узлах, при котором среднее время пребывания в сети было бы минимально или ее пропускная способность максимальна.

Приведем соотношения, которые связывают коэффициенты использования узлов с коэффициентами посещаемости этих узлов:

$$\frac{U_k}{U_j} = \frac{V_k S_k}{V_j S_j}, \; j,k = \overline{1,K}. \qquad (2.25)$$

Устройство k будет «насыщено» требованиями, если его коэффициент использования близок к единице. В этом случае при выполнении гипотезы о балансе потоков интенсивности входящего потока и обслуживания будут практически совпадать, то есть

$$X_k = \frac{1}{S_k}, \text{ при } X_k < \frac{1}{S_k}, \; U_k < 1. \qquad (2.26)$$

При увеличении числа требований, одновременно обслуживающихся в сети, первым достигнет насыщения тот узел d, который будет иметь максимальную величину $V_i S_i$, $i = 1,...,K$, то есть

$$V_d S_d = max\{V_1 S_1, ..., V_k S_k\}. \qquad (2.27)$$

При увеличении количества требований коэффициент использования U_d приближается к 1 и $X_d = \frac{1}{S_d}$. Поскольку $\frac{X_0}{X_d} = \frac{1}{V_d}$, то $X_0 = \frac{1}{V_d S_d}$.

Таким образом, исходный поток из сети при большом числе N полностью определяется узлом d, который является узким местом.

Определим минимальное среднее время пребывания требования R_0, если в сети есть лишь одно требование, через коэффициенты посещаемости отдельных устройств и время обслуживания устройства

$$R_0 = \sum_{k=1}^{K} V_k S_k. \qquad (2.28)$$

На рис. 2.9 изображен график зависимости интенсивности потока в сети от количества требований в сети. При увеличении N интенсивность Хо монотонно возрастает до предельной асимптоты $V_d S_d$, то есть пока на эту интенсивность не начнет влиять потенциально узкое место – узел d. На рис. 2.9 через N^* обозначено число требований, при котором узкое место еще не влияет на пропускную способность сети.

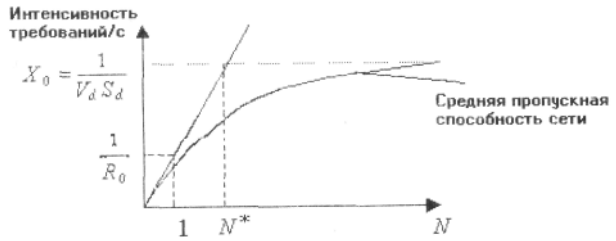


Рис. 2.9

Для простейшей замкнутой сети, если количество устройств M = 1, то $R' = R'_0$. При увеличении M поток из сети будет возрастать, но не больше, чем $X_0 = \frac{1}{V_d S_d}$. Таким образом,

$$R' \geq MV_d S_d - Z \geq MV_k S_k - Z, k = \overline{1,K} . \qquad (2.29)$$

Итак, при увеличении M среднее время пребывания имеет асимптоту MV_dS_d-Z. На рис.2.10 показана зависимость среднего времени пребывания в замкнутой сети от числа устройств M. Асимптота, которая создает узкое место в сети, пересекает ось абсцисс в точке $M_d = \frac{Z}{V_d S_d}$.

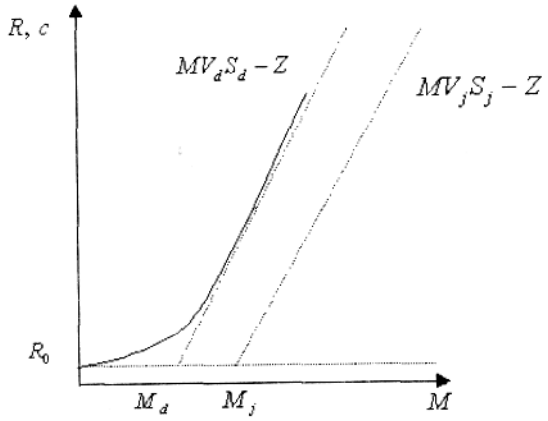


Рис. 2.10

Изложенный подход к поиску узких мест в сети просто использовать на практике. Покажем это на примерах.

Пример 2.3. Проведем расчет характеристик сети, которая изображена на рис. 2.11, там же приведены значения операционных переменных S_k, q_{kj} и Z.

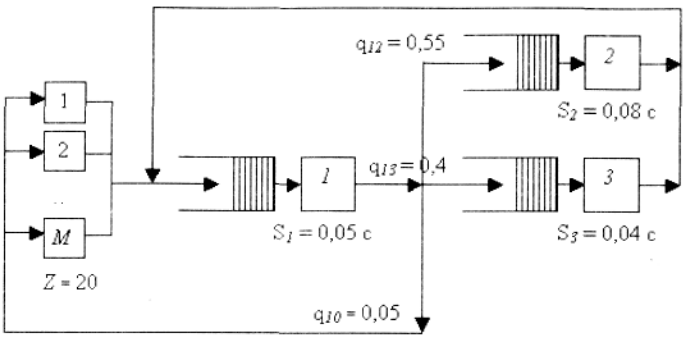


Рис. 2.11

Запишем уравнения баланса потоков для коэффициентов посещаемости этой сети:

$$V_0 = 1 = 0{,}05V_1;$$
$$V_1 = V_0 + V_2 + V_3;$$
$$V_2 = 0{,}55V_1;$$
$$V_3 = 0{,}4V_1.$$

Решая приведенную систему уравнений, получаем

$$V_1 = 20,\ V_2 = 11,\ V_3 = 8.$$

Определяем значения V_kS_k для каждого из узлов сети:

$$V_1S_1 = 20 \cdot 0{,}05 = 1\text{ с};$$
$$V_2S_2 = 11 \cdot 0{,}08 = 0{,}88\text{ с};$$
$$V_3S_3 = 8 \cdot 0{,}04 = 0{,}32\text{ с}.$$

Таким образом, минимальное среднее время пребывания одного требования составляет: R_0 = 1 + 0,088 + 0,32 = 2,2 с.

Поскольку $V_1S_1 > V_2S_2 > V_3S_3$, то потенциальным узким местом в сети является первый узел.

Основываясь на рассматриваемом методе операционного анализа, дадим ответ на некоторые вопросы.

1. Пусть измерениями определено, что X_0 = 0,715 требований/с, **А** среднее время пребывания требования в сети составляет 5,2 с. Какое среднее количество устройств обслуживания взаимодействует с сетью за все время наблюдения?

В соответствии с формулой (2.13) имеем:

$$M = (R + Z)X_0 = (5{,}2 + 20) \cdot 0{,}715 = 18\ \text{устройств}.$$

2. Можно ли обеспечить среднее время пребывания требований в сети равным 8 с при 30 устройствах обслуживания? Какое максимальное среднее время обслуживания требования должен иметь узел 7, чтобы это стало возможным?

В соответствии с формулой (2.13) имеем:

$$R \geq M / X_0 - Z = 30/1 - 20 = 10\text{ с}.$$

Таким образом, при взаимодействии с сетью 30 устройств, среднее время пребывания требования в ней превысит 10 с.

Обозначим через S_1* допустимое среднее время обслуживания требования. Тогда можно записать

$$MV_1S_1^* - Z \leq 8\text{ с};$$
$$S_1^* \leq 0{,}047\text{ с},$$

то есть максимально возможное среднее время обслуживания требования узлом 1 составляет 0,047 с. На рис. 2.12 изображены графики для асимптоты среднего времени обслуживания требования.

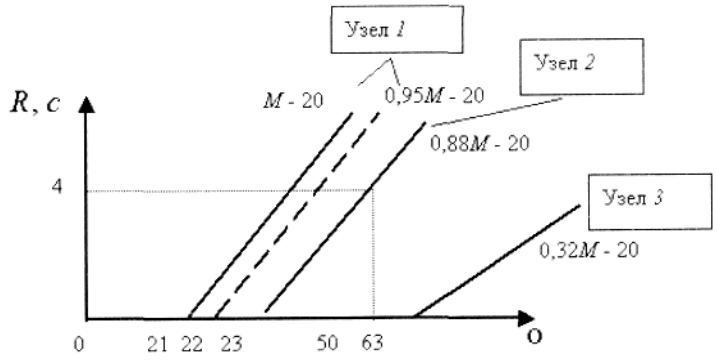


Рис.2.12

Пример 2.4. Допустим, что в сеть, кроме требований от устройств обслуживания, поступают еще и требования от узла *3,* как показано на рис. 2.13.

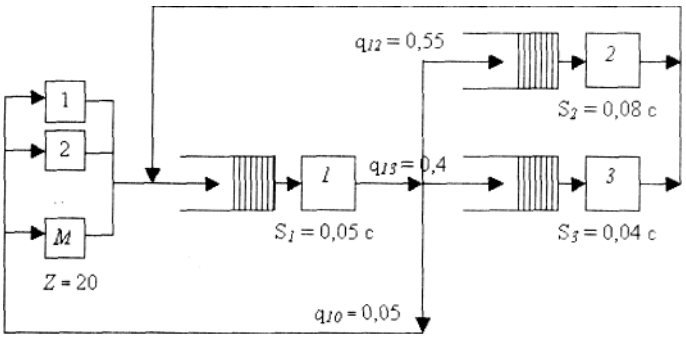


Рис. 2.13

На рис. 2.13 параметры со штрихом характеризуют требования узла *3*. Измерения в данной сети показали, что узел *3* загружен практически полностью, **А** время ответа системы равняется 7 с. Как в этих условиях загружен узел *1* и какое значение приобретает X_0?

Из рис. 2.13 видно, что $V'_1 = V'_3 = 1$. Тогда $F_1S_1 = 0{,}05$ с, $F_1S_1 = 0{,}1$ с.

Таким образом, потенциально узким местом для требований узла *3* есть сам узел *3*.

$$X_0 = \frac{M}{R+Z} = \frac{25}{7+20} = 0{,}924 \text{ требований/с.}$$

Поскольку $X_0 = \frac{X_3}{V_3} = \frac{X_1}{V_1}$, то $X_3 = 7{,}392$ требований/с, $X_1 = 18{,}48$ требований/с.

Загрузка узла *3,* создаваемая требованиями от устройств обслуживания, составляет

$$U_3 = X_3S_3 = 7{,}392 \cdot 0{,}04 = 0{,}296 .$$

Вторая загрузка узла *3* создается требованиями, которые поступают от этого узла ($U'_3 = 0{,}704$).

Поскольку требования от узла *3* циркулируют в замкнутом контуре, то при условии замкнутости сети имеем

$$X'_1 = X'_3 = X'_0 = \frac{U'_3}{S'_3} = 7{,}04 \text{ требований/с.}$$

Определим коэффициент использования узла *1*

$$U_1 + U'_1 = X_1S_1 + X'_1S'_1 = 18{,}48 \cdot 0{,}05 + 7{,}04 \cdot 0{,}01 = 0{,}994.$$

Таким образом, узел *1* также используется практически полностью.

Операционный анализ вероятностных сетей СМО и приведенные примеры расчетов таких сетей показывают, как, не прибегая к моделированию, можно получить некоторые расчетные характеристики на уровне средних значений. В технологии имитационного моделирования операционный анализ может быть использован для сравнения результатов моделирования с расчетными значениями при проверке правильности (валидации) имитационной модели и при поиске наилучших решений по результатам моделирования (см. главу 11).

ГЛАВА 3. ВЕРОЯТНОСТНОЕ МОДЕЛИРОВАНИЕ

3.1. Метод статистических испытаний

В тех случаях, когда при моделировании необходимо учитывать некоторый случайный фактор (элемент или явление), который невозможно описать аналитически, используют метод моделирования, называемый ***методом статистических испытаний*** **или** ***методом Монте-Карло.*** С помощью этого метода может быть решена любая вероятностная задача. Однако использовать его целесообразно в том случае, если решить задачу этим методом проще, чем любым другим.

Суть метода состоит в том, что вместо описания случайных явлений аналитическими зависимостями проводится розыгрыш случайного явления с помощью некоторой процедуры, которая дает случайный результат. С помощью розыгрыша получают одну реализацию случайного явления. Осуществляя многократно такой розыгрыш, накапливают статистический материал (то есть множество реализаций случайной величины), который можно обрабатывать статистическими методами. Рассмотрим этот метод на примерах.

Пример 3.1. Пусть четыре стрелка одновременно стреляют по движущейся цели. Вероятность попадания в цель каждым стрелком равняется 0,5 (попал или не попал). Цель считается пораженной, если в нее попало два или более стрелка. Найти вероятность поражения цели.

Эту задачу можно легко решить методами теории вероятности. Вероятность поражения цели $P_{пор} = 1 - P_{непор}$.

Вероятность непоражения $P_{непор}$ определяют как число сочетаний, когда в цель не попал ни один стрелок, плюс попал один из стрелков:

$$P_{\text{непор}} = 0{,}5^4 + C_4^1 \times 0{,}5^1 \times 0{,}5^3 = 0{,}5^4 + 4 \times 0{,}5^1 \times 0{,}5^3 = 0{,}3125,$$

$$P_{\text{пор}} = 1 - P_{\text{непор}} = 1 - 0{,}3125 = 0{,}6875.$$

Решим эту задачу методом статистических испытаний. Процедуру розыгрыша реализуем подбрасываниям одновременно четырех монет. Если монета падает лицевой стороной, то считаем, что стрелок попал в цель. Обозначим через *m* число успешных испытаний. Сделаем N испытании, тогда в соответствии с теоремой Бернулли: $P_{пор} \approx \frac{m}{N}$.

Пример 3.2. Пусть есть некоторая цель, на которую бомбардировщики сбрасывают *n* бомб. Каждая бомба поражает область в виде круга радиусом *r* (рис. 3.1). Цель считается пораженной, если одновременно бомбами накрыто *K* процентов площади *S.* Найти вероятность поражения цели.

Аналитически решить эту задачу очень трудно. Покажем, как ее можно решить методом статистических испытаний.

Наложим координатную сетку на всю возможную область попадания бомб. Разыграем *n* точек – координат попадания бомб. Опишем возле каждой точки круг радиусом *r* (рис. 3.2) и определим заштрихованную площадь поражения. Если заштрихованная площадь будет составлять *K* процентов и больше всей площади цели *S,* то цель считается пораженной, **А** испытание успешным. В противном случае цель не будет поражена и испытание не успешное.

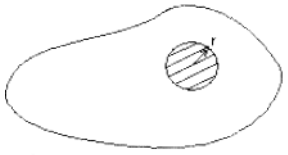


Рис.3.1

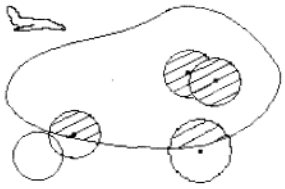


Рис.3.2

Выполним *N* испытаний. Тогда вероятность поражения цели $P_{пор} \approx \frac{m}{N}$, где *m* – количество испытании, при которых цель оыла поражена.

Методом статистических испытаний можно оценить математическое ожидание и другие вероятностные характеристики. Например, оценку математического ожидания площади поражения цели можно определить как $M(S) = \frac{1}{N}\sum_{i=1}^{N} S_i$. При $N \to \infty$ эта оценка будет приближаться к математическому ожиданию в соответствии с законом больших чисел. В этом выражении S_1, площадь поражения в *i*-м испытании.

Алгоритм метода статистических испытаний такой:

1. Определить, что собой будет представлять испытание или розыгрыш.

2. Определить, какое испытание является успешным, **А** какое – нет.

3. Провести большое количество испытаний.

4. Обработать полученные результаты, статистическими методами и рассчитать статистические оценки искомых величин.

К недостаткам метода можно отнести необходимость проведения большого количества испытаний, чтобы получить результат с заданной точностью.

Таким образом, метод статистических испытаний – это метод математического моделирования случайных величин, в котором сама случайность непосредственно включена в процесс моделирования и является его важным элементом. Каждый раз, когда в ход выполнения некоторой операции вмешивается случайный фактор, его влияние моделируется с помощью розыгрыша.

Для эффективного розыгрыша случайных величин используют генераторы случайных чисел. Такие генераторы строятся аппаратными и программными методами. Наиболее применимыми являются программные методы, которые дают возможность получить последовательности псевдослучайных чисел по рекуррентным формулам. Обычно используется мультипликативный конгруэнтный метод, рекуррентное соотношение для которого имеет вид:

$$X_{i+1} = a X_i (mod\ m), \qquad (3.1)$$

где a и m – некоторые константы. Необходимо взять последнее псевдослучайное число X_i, умножить его на постоянный коэффициент **А** и взять модуль полученного числа по m, то есть разделить на m и получить остаток. Этот остаток и будет следующим псевдослучайным числом X_{i+1}. Для двоичного компьютера $m = 2^g - 1$, где g – длина разрядной сетки. Например, для 32-разрядного компьютера $m = 2^{31} - 1 = 2147483647$, поскольку один разряд задает знак числа.

В языке GPSS World используется мультипликативный конгруэнтный алгоритм Лехмера с максимальным периодом, который генерирует 2147483647 уникальных случайных чисел без повторения. Эти числа генерируют специальные генераторы, которые обозначаются RN<№>, где № – номер генератора случайных чисел (может принимать значения от 1 до 7). При обращении к этим генераторам выдаются целые случайные числа в диапазоне от 0 до 999 включительно. При использовании генераторов в случайных функциях распределений случайные числа генерируются в диапазоне от 0 до 0,999999 включительно.

3.2. Моделирование дискретных случайных величин

Моделирование события. Пусть необходимо смоделировать появление некоторого события A, вероятность наступления которого равняется $P(A) = P$. Обозначим обращения к генератору, который разыгрывает псевдослучайные, равномерно распределенные на интервале (0, 1) числа r_i, через R. Событие A при розыгрыше будет наступать тогда, когда $r \le P$ (рис. 3.3), в противном случае происходит событие A с вероятностью $r > P$.

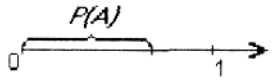


Рис. 3.3

Действительно:

$$P(r < P) = \int_0^P f(x)dx = \int_0^P f(r)dr = P = P(A). \qquad (3.2)$$

Данный метод используется в языке GPSS для блока **TRANSFER** в статистическом режиме работы, когда транзакты следуют по двум разным направлениям в зависимости от вероятности (см. параграф 4.9).

Моделирование группы несовместных событий. Пусть есть группа несовместимых событий $A_1, A_2, \ldots, A_k$. Известны вероятности наступления событий $P(A_1)\ P(A_2) \ldots, P(A_k)$. Тогда из-за несовместности событий $\sum_{i=0}^{k} P(A_i) = 1$. Пусть $P_i = P(A_i)$, $p_0 = 0$. На отрезке (0, l) отложим эти вероятности (рис. 3.4).

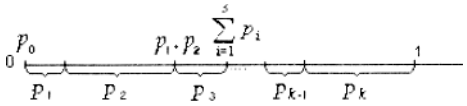


Рис. 3.4

Если полученное число попало в интервал от $\sum_{k=0}^{i-1} p_k$ до $\sum_{k=0}^{i} p_k$, то произошло событие A_i. Такую процедуру называют определением результата испытание по жребию, и она основывается на формуле

$$P\left\{\sum_{k=0}^{i-1} p_k < r \le \sum_{k=0}^{i} p_k\right\} = p_i = P(A_i), \qquad (3.3)$$

где $p_0 = 0$.

Моделирование случайной дискретной величины. Моделирование случайной дискретной величины выполняется аналогично моделированию группы несовместимых событий. Дискретная случайная величина **X** задается в соответствии с табл. 3.1.

Случайную величину **X** можно представить как полную группу событий:

$$A_1 = (X = x_1),\ A_2 = (X = x_2),\ \ldots,\ A_n = (X = x_n)$$

Таблица 3.1

Возможное значение	X_1	X_2	…	x_n
Вероятность	P_1	P_2	…	P_n

Данный метод используется в языке GPSS для моделирования дискретных случайных функций распределения (см. параграф 4.13).

Моделирование условного события. Моделирование условного события *A,* которое происходит при условии, что наступило событие *B* с вероятностью *P(A/B),* показано на рис. 3.2.3. Сначала моделируем событие *B.* Если событие *B* происходит, то моделируем наступление события *A,* если имеем $\overline{B}$, то не моделируем наступление события *A.*

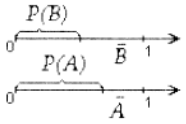


Рис. 3.5

3.3. Моделирование непрерывных случайных величин

В данном случае используется метод обратной функции. Пусть есть некоторая функция распределения случайной величины (рис.3.6). Разыграем на оси ординат точку r, используя функцию *F(x).* Тогда можем получить значение величины **X** такое, что *F(x)=r.*

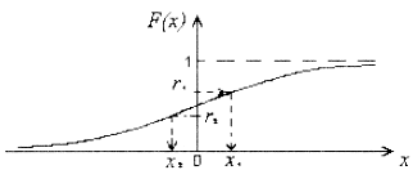


Рис. 3.6

Найдем функцию распределения *F(x)* случайной величины *X.* По определению она равна вероятности *P(X<x).* Из рис. 3.7 очевидно, что

$$P(X < x) = P(R < F(X)) = \int_0^{F(x)} f(r)\,dr = F(x). \qquad (3.4)$$

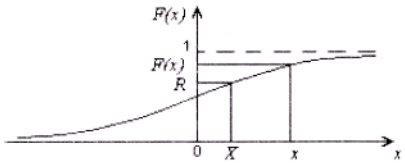


Рис. 3.7

Таким образом, последовательность r_1, r_2, r_3, ... , принадлежащая *R(0,*1), преобразуется в последовательность $x_1, x_2, x_3, \ldots$, которая имеет заданную функцию плотности распределения *f(x).*

Моделирование равномерного распределения в интервале (a, b) случайной величины. Для моделирования воспользуемся методом обратной функции. На рис. 3.8 показана функция плотности равномерного распределения.

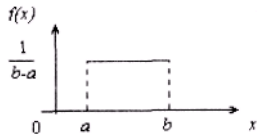


Рис. 3.8

Находим функцию распределения и приравниваем ее к случайному числу

$$R = \int_a^x \frac{dx}{b-a} = \frac{x-a}{b-a}. \qquad (3.5)$$

Отсюда **X** = *(b – a)R + a.*

Моделирование экспоненциального распределения случайной величины. Функция плотности экспоненциального распределения случайной величины $f(x) = \lambda e^{-\lambda x}$ и функция распределения показаны на рис. 1.1.

Воспользуемся методом обратной функции:

$$R = \int_0^x f(x)dx = \int_0^x \lambda e^{-\lambda x} dx = 1 - e^{-\lambda x}. \qquad (3.6)$$

Из выражения (3.6) находим *x*:

$$x = -\frac{1}{\lambda} ln(1 - R). \qquad (3.7)$$

Можно показать, что случайная величина (1-R) распределена так же, как и величина *R.* Тогда, сделав замену *(1-R)* на *R,* получаем

$$x = -\frac{1}{\lambda} ln\, R. \qquad (3.3.5)$$

Покажем, как, используя метод обратной функции, можно моделировать случайную величину, распределенную по экспоненциальному закону. Подобный подход принят в языке GPSS [10].

Пусть λ = 1. Выполним аппроксимацию функции экспоненциального распределения линейными участками, чтобы можно было использовать ее для моделирования методом обратной функции. Для аппроксимации достаточно 24 точек. В табл. 3.2 занесены соответствующие значения аргумента **X** и функции *F(x),* значения которой генерируют с помощью генератора случайных чисел.

Таблица 3.2

X	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,75	0,8	0,84	0,88
F(X)	0	0,1	0,222	0,355	0,509	0,69	0,915	1,2	1,38	1,6	1,83	2,12

X	2,3	2,52	2,81	2,99	3,2	3,5	3,9	4,6	5,3	6,2	7	8
F(X)	0,9	0,92	0,94	0,95	0,96	0,97	0,98	0,99	0,995	0,998	0,999	0,9998

На рис. 3.9 и 3.10 показаны графики двух функций. На рис. 3.9 изображена аппроксимация экспоненциальной функции с параметром λ = 1, **A** на рис. 3.10 – функция, обратная к аппроксимированной. Первая функция воспроизводит заданные в табл. 3.2 значения. Вторая функция используется для розыгрыша экспоненциального распределения, поскольку удобнее задавать значение *x, A* получать значение функции.

Если необходимо моделировать случайные величины *X,* распределенные по экспоненциальному закону с параметром $\ddot{e}_x \neq 1$, которые используется как задержка во времени с параметром $T = \frac{1}{\ddot{e}_X}$, например, для моделирования пуассоновского потока поступления требований, то поступают таким образом:

– генерируют значения случайной величины, распределенной по экспоненциальному закону с λ = 1 (рис. 3.10);

– находят произведение полученного значения и математического ожидания случайной величины $T = \frac{1}{\lambda_X}$.

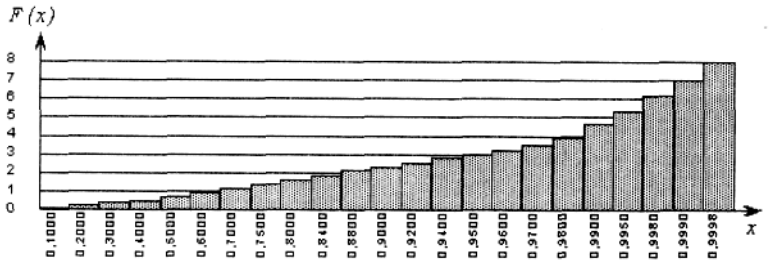


Рис. 3.10

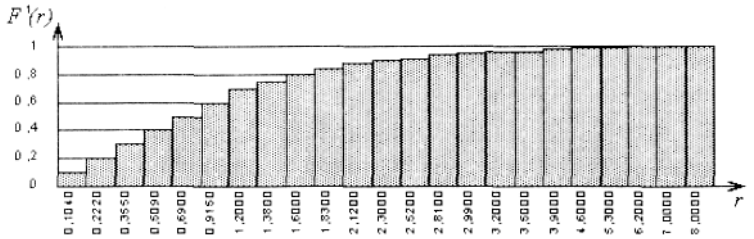


Рис. 3.9

В результате получают искомую последовательность значений реализации случайной величины *X.*

Моделирование нормального закона распределения случайной величины. Для моделирования нормального закона распределения случайной величины нельзя непосредственно воспользоваться методом обратной функции, поэтому используем центральную предельную теорему. Пусть случайная величина **X** имеет математическое ожидание m_x и среднеквадратичное отклонение σ_x, а случайная величина Z имеет математическое ожидание $m_z = 0$ и среднеквадратичное отклонение $\sigma_z = 1$. Легко показать, что

$$X = \sigma_x Z + m_x \ . \qquad (3.8)$$

Сформулируем центральную предельную теорему.

Если X_1 ..., X_n – независимые случайные величины со средним значением E[X_i] = a, $i = \overline{1,n}$ и дисперсией D[X_i] = σ^2, $i = \overline{1,n}$, то при неограниченном увеличении *n* функция распределения случайной величины $\overline{X}^*(n) = \frac{\frac{1}{n}(X_1 + ... + X_n) - a}{\sigma/\sqrt{n}} = \frac{(\overline{X}(n) - a)\sqrt{n}}{\sigma}$ приближается к функции распределения стандартного нормального закона Φ(z) при всех значениях аргумента, то есть

$$F_{\overline{X}^*(n)} \to \Phi(z), \qquad (3.9)$$

где $\Phi(z) = \frac{1}{\sqrt{2\pi}} \int_0^z e^{-\frac{z^2}{2}} dz$.

Для получения нормального закона распределения случайной величины достаточно суммировать шесть случайных величин, полученных с помощью генератора случайных чисел *R,* и, пронормировав полученные значения так, чтобы определить Z, по формуле (3.8) найти значение *X.*

Обычно суммируют 12 случайных величин R_i, $Z = \sum_{i=1}^{12} R_i$, тогда дисперсия D(Z) будет равняться единице.

Рассмотрим, как моделируются нормально распределенные случайные величины в системе моделирования GPSS.

Выполним аппроксимацию функции нормального распределения случайной величины Z с параметрами m_z =0 и σ_z=1. Для этого достаточно 25 точек. В табл. 3.3 занесенные соответствующие значения аргумента X и функции *F (x).*

Для того, чтобы получить функцию нормального распределения с математическим ожиданием $m_x \neq 0$ и среднеквадратичным отклонением $\sigma_x \neq 1$, необходимо сделать вычисления по формуле (3.8).

На рис. 3.11 изображен график функции, полученной в результате аппроксимации функции нормального распределения Φ*(z),* а на рис. 3.12 – более удобный для моделирования график функции (как аргумент используют генератор случайных чисел и получают значение функции).

Таблица 3.3

X	-5	-4	-3	-2,5	-2	-1,5
F(x)	0	0,00003	0,00135	0,00621	0,02275	6,06681

X	-1,2	-1	-0,8	-0,6	-0,4	-0,2
F(x)	0,11507	0,15866	0,21186	0,2742	0,34458	0,42074

X	0	0,2	0,4	0,6	0,8	1
F(x)	0,5	0,57964	0,65542	0,72575	0,78814	0,84134

X	1,2	1,5	2	2,5	3	4	5
F(x)	0,88493	0,93319	0,97725	0,99379	0,99865	0,99997	1

Если необходимо обеспечить положительные разыгрываемые значения, то нужно выполнить условие $m_x \geq 5\sigma_x$.

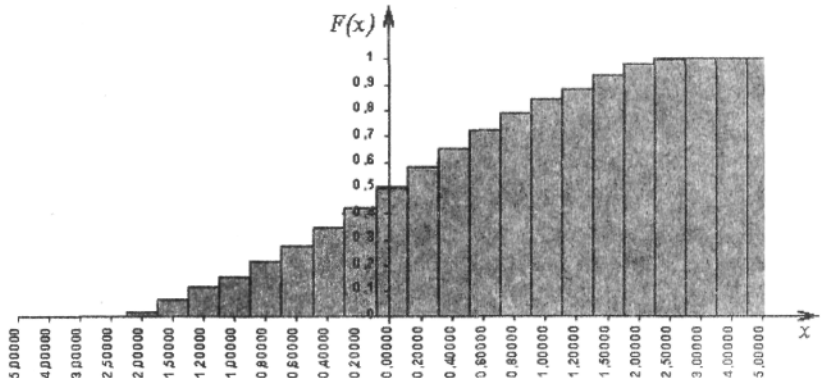


Рис. 3.11

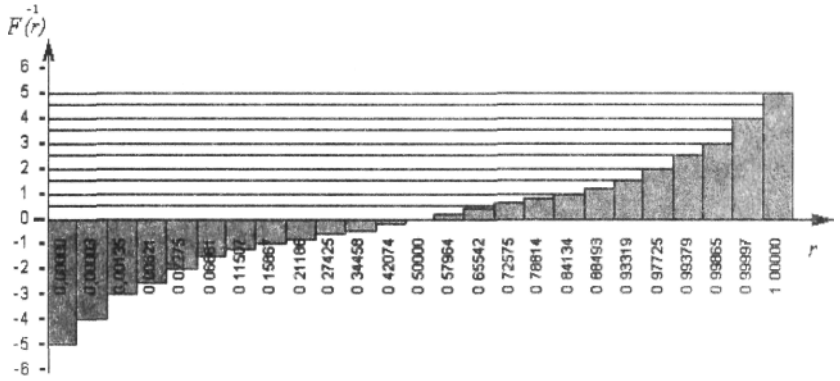


Рис. 3.12

В рассмотренных приближенных методах «хвосты» нормального распределения оказываются неточными. Существуют и более точные методы моделирования нормального распределения случайной величины [11].

3.4. Сбор статистических данных для получения оценок характеристик случайных величин

Основными элементами, из совокупности которых складывается вероятностная модель метода статистических испытаний, являются случайные реализации. Очевидно, что при решении некоторой задачи определения характеристик или параметров исходного случайного процесса должен быть определен этот случайный процесс.

Искомыми величинами при использовании метода статистических испытаний являются оценки:

– вероятности наступления некоторого события;

– математического ожидания случайной величины;

– дисперсии случайной величины;

– коэффициентов ковариации или корреляции случайной величины.

Для оценки вероятности p наступления некоторого события **A** используется частота наступления этого события

$$\hat{p} = \frac{m}{N}, \qquad (3.10)$$

где m – частота наступления события, а N – число опытов.

Для оценки математического ожидания случайной величины используется среднее значение

$$\hat{x} = \hat{E}[x] = \frac{1}{N}\sum_{i=1}^{N} x_i, \qquad (3.11)$$

где x_i – i-я реализация случайной величины.

Для оценки дисперсии случайной величины ξ, используют формулу

$$S^2 = \frac{1}{N-1}\sum_{k=1}^{N}\left(x_k - \bar{x}\right)^2, \qquad (3.12)$$

где S^2 – оценка дисперсии случайной величины ξ.

Непосредственно использовать эти формулы для вычисления дисперсии сложно, поскольку среднее значение изменяется по мере накопления x_i-, то есть нужно запоминать все N значений x_i. Поэтому для вычисления используют формулу:

$$S^2 = \hat{\sigma}^2 = \frac{1}{N-1}\left[\sum_{i=1}^{N} x_i^2 - \frac{1}{N}\left(\sum_{i=1}^{N} x_i\right)^2\right]. \qquad (3.13)$$

В этом случае достаточно накапливать две суммы значений $-x_i$ и x_i^2.

Непосредственное использование этой формулы для программирования может привести к переполнению разрядной сетки, если программировать ее в таком порядке, в котором она записана. Необходимо изменить последовательность действий, чтобы избавиться от очень больших чисел и переполнения разрядной сетки компьютера.

Все статистические оценки должны иметь определенные качественные показатели, к которым относятся ***несмещенность, эффективность*** и ***состоятельность оценки.***

Для случайных величин ξ, и η с возможными значениями x_k, y_k оценка корреляционного момента определяется так

$$\hat{K}_{\xi\eta} = \frac{1}{N-1}\left[\sum_{k=1}^{N}(x_k - x)(y_k - y)\right] \qquad (3.14)$$

или в удобной для вычислений форме

$$\hat{K}_{\xi\eta} = \frac{1}{N-1}\left[\sum_{k=1}^{N} x_k y_k - \frac{1}{N}\sum_{k=1}^{N} x_k \sum_{k=1}^{N} y_k\right]. \qquad (3.15)$$

3.5. Определение количества реализаций при моделировании случайных величин

Число испытаний N определяет точность получаемых результатов моделирования. Если необходимо оценить величину параметра **A** по результатам моделирования .x_i, то за оценку следует брать величину $\bar{x}$, которая выступает в функции от x_i.

Из-за случайности $\bar{x}$ будет отличаться от a, то есть

$$|a - x| < \varepsilon, \qquad (3.16)$$

где ε – точность оценки. Вероятность того, что данное неравенство выполняется, обозначим через αa :

$$P(|a - x| < \varepsilon) = \alpha. \qquad (3.17)$$

Для определения точности результатов статистических испытаний необходимо воспользоваться выражением (3.17).

Определение количества реализаций для оценки вероятности наступления события. Пусть целью моделирования будет определение вероятности наступления некоторого события A, определяющего состояние моделированной системы. В любой из N реализаций процесс наступления события **A** является случайной величиной, которая может приобретать значение $x_1 = 1$ с вероятностью p и $x_2 = 0$ с вероятностью $1 - p$. Тогда можно найти математическое ожидание

$$E[\xi] = x_1 p + x_2(1-p) = p \qquad (3.18)$$

и дисперсию

$$D[\xi] = (x_1 - E[\xi])^2 p + (x_2 - E[\xi])^2 (1-p) = p(1-p). \qquad (3.19)$$

В качестве оценки p используют частоту наступления события A. Эта оценка несмещенная, состоятельная и эффективная.

При условии, что N заведомо задано, достаточно накапливать m:

$$\frac{m}{N} = \frac{1}{N}\sum_{i=1}^{N} \xi_i, \qquad (3.20)$$

где ξ_i – наступление события **A** в реализации, ξ_i-={1 ,0}.

По формулам (3.18-3.20) находим

$$E\left[\frac{m}{N}\right] = p, \; D\left[\frac{m}{N}\right] = \frac{p(1-p)}{N-1}.$$

В соответствии с центральной предельной теоремой (в данном случае можно взять теорему Лапласа) случайная величина $\frac{m}{N}$ будет иметь распределение, близкое к нормальному (рис.3.13). Поэтому для каждой достоверности α из таблиц нормального распределения можно найти такую величину t_a, что точность ε будет равняться величине

$$\varepsilon = t_\alpha \sqrt{D\left[\frac{m}{N}\right]}. \qquad (3.21)$$

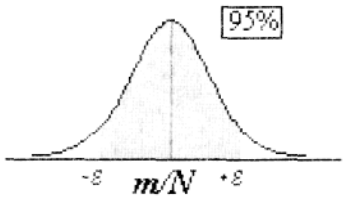


Рис. 3.13

При α = 0,95 t_α = 1,96.
При α = 0,997 t_α = 3.
Подставим в уравнение (3.21) выражение дисперсии

$$\varepsilon = t_\alpha \sqrt{\frac{p(1-p)}{N-1}}. \qquad (3.22)$$

Отсюда находим

$$N = t_\alpha{}^2 \frac{p(1-p)}{\varepsilon^2} + 1. \qquad (3.23)$$

Поскольку вероятность p заранее неизвестна, прибегают к пробным испытаниям *(*N = 50...100), получают частоту $\frac{m}{N}$ и подставляют ее значения в выражение (3.23) вместо *p,* после чего определяют конечное количество испытаний.

Определение количества реализаций для оценки среднего значения случайной величины. Пусть случайная величина имеет математическое ожидание **A** и дисперсию σ^2. В реализации с номером i она принимает значение x_i. Для оценки математического ожидания **A** используем среднее

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i. \qquad (3.24)$$

В соответствии с центральной предельной теоремой при больших значениях N среднее арифметическое $\bar{x}$ будет нормально распределено с математическим ожиданием **A** и дисперсией $\frac{\sigma^2}{N-1}$ тогда

$$\varepsilon = t_\alpha \frac{\sigma}{\sqrt{N-1}}. \qquad (3.25)$$

Отсюда

$$N = \frac{t_\alpha{}^2 \sigma^2}{\varepsilon^2} + 1. \qquad (3.26)$$

Поскольку дисперсия оцениваемой случайной величины неизвестна, необходимо провести 50-100 испытаний и оценить σ^2, **A** потом полученное значение оценки подставить в формулу (3.26), чтобы определить необходимое количество реализаций N.

ГЛАВА 4. СИСТЕМА МОДЕЛИРОВАНИЯ GPSS

4.1. Объекты

Язык GPSS – это язык декларативного типа, построенный по принципу объектно-ориентированного языка. Основными элементами этого языка являются транзакты и блоки, которые отображают соответственно динамические и статические объекты моделируемой системы.

Предназначение объектов системы различно. Выбор объектов в конкретной модели зависит от характеристик моделируемой системы. Каждый объект имеет некоторое число свойств, названых в GPSS стандартными числовыми атрибутами (СЧА)*. Часть СЧА доступна пользователю только для чтения, **А** на значение других он может влиять, используя соответствующие блоки.

* В GPSS World используется термин «системные числовые атрибуты». При изложении материала авторы придерживаются термина «стандартные числовые атрибуты» в соответствии с известной работой Т. Дж. Шрайбера [10], полагая, что понятие «системная» применимо к глобальной информации, которая не связана с какими-то отдельными элементами. Практически всегда к системным атрибутам относится системное время. См. Е. Киндлер «Языки моделирования», 1985, стр. 18.

Блоки и транзакты. Каждая GPSS-модель обязательно должна содержать такие объекты, как блоки и транзакты.

В GPSS концепция передачи управления от блока к блоку имеет специфические особенности. Последовательность блоков GPSS-модели показывает направления, в которых перемещаются элементы. Каждый такой элемент называется ***транзактом.*** Транзакты – это динамические элементы GPSS-модели.

Блоки языка GPSS представляют собой подпрограммы, написанные на макроассемблере или на языке Си, и содержат набор параметров (операндов) для обращения к ним. Как и во всех языках моделирования в GPSS существует внутренний механизм передачи управления, который реализуется в модельном времени, что дает возможность отобразить динамические процессы в реальных системах. Передача управления от блока к блоку в GPSS-программах реализуется с помощью движения транзактов в модельном времени. Обращение к подпрограммам блоков происходит через движение транзактов.

Содержательное значение транзактов определяет разработчик модели. Именно он устанавливает аналогию между транзактами и реальными динамическими элементами моделируемой системы. Такая аналогия никогда не указывается транслятору GPSS, она остается в воображении разработчика моделей. В табл. 4.1 приведены примеры аналогий между транзактами и элементами реальных систем.

Таблица 4.1

Система	***Элементы систем, которые моделируются транзактами***
Магазин	Покупатель
Автомобильное шоссе	Автомобиль
Склад	Заявка

С точки зрения программы ***транзакт это структура данных,*** которая содержит такие поля: имя или номер транзакта; время появления транзакта; текущее модельное время; номер блока, в котором находится транзакт; номер блока, куда он продвигается; момент времени начала продвижения; приоритет транзакта; параметры транзакта: P1, P2, ...

В языке GPSS все транзакты нумеруются по мере их появления в модели. Параметры транзактов отображают свойства моделируемого динамического объекта. Например, если моделируется движение автомобилей на участке дороги, то параметрами транзакта (автомобиля) в зависимости от целей моделирования могут быть скорость, тормозной путь, габариты и др.

Каждый транзакт занимает некоторые объем памяти ЭВМ. После того, как он закончит свое движение по блокам модели, его необходимо уничтожать для освобождения памяти, чтобы избежать ее переполнения. Поскольку транслятору не известно, сколько траизактов одновременно будет находиться в модели, то память под транзакты выделяется динамически.

Таким образом, при начале моделирования в GPSS-модели нс существует ни одного транзакта. В процессе моделирования транзакты входят в модель в определенные моменты времени, соответствующие логике функционирования моделируемой системы. Таким же образом транзакты покидают модель в зависимости от специфики моделирования. В общем случае в модели существует несколько транзактов, но в каждый момент времени движется только один из них.

Если транзакт начал свое движение, он передвигается от блока к блоку по пути, указанному блок-схемой (логикой работы модели). В тот момент, когда транзакт входит в блок, вызывается соответствующая этому блоку подпрограмма. Далее транзакт (в общем случае) пытается войти в следующий блок. Его перемещение продолжается до тех пор, пока не выполнится одно из таких возможных условий:

1. Транзакт входит в блок, функцией которого является задержка транзакта на определенное время.
2. Транзакт входит в блок, функцией которого является удаление транзакта из модели.
3. В соответствии с логикой модели транзакт пытается войти в следующий блок, но блок не принимает этот транзакт. В этом случае транзакт остается в том блоке, в котором в данное время находится, но позже будет повторять попытки войти в следующий блок. Когда условия в модели изменятся, одна из таких попыток может быть успешной. После этого транзакт продолжит свое перемещение по модели. Подробнее это рассматривается в параграфе 4.22.

Если выполняется одно из указанных условий, транзакт остается на месте и в модели делается попытка перемещения другого транзакта.

Объекты типа «ресурсы». Аналогами обслуживающих устройств реальных систем в GPSS являются объекты типа «ресурсы». К объектам этого типа относятся устройства, многоканальные устройства* и логические ключи.

* В языке GPSS используется термин STORAGE, дословный перевод которого – хранилище, память. На самом деле по функциональному начначению STORAGE определяет емкость (количество) однотипных устройств обслуживания, которые в одних случаях отображают обслуживание (кассиры, турникеты), **А** в других хранилища (емкосгь склада, память компьютера и т.п.). Поэтому, в тексте используется термин «многоканальное устройство», также как в [10].

Как и в каждом объектно-ориентированном языке в GPSS каждый объект имеет ***свойства*** **и** ***методы,*** которые изменяют эти свойства. В GPSS свойства объектов называют ***стандартными числовыми атрибутами*** *(СЧА).*

Устройство (одноканальное устройство, прибор) представляет собой ресурс, который в любой момент времени может быть занят только одним транзактом. Интерпретатор автоматически вычисляет такие его СЧА, как общее время занятости устройства, число транзактов, который занимали устройство, коэффициент использования устройства, среднее время занятости устройства одним транзактом и т. п.

Многоканальные устройства (МКУ) (несколько парачлельных одинаковых устройств) представляют собой объекты типа «ресурсы» для параллельной обработки. Они могут быть использованы несколькими транзактами одновременно. Пользователь определяет емкость каждого МКУ, который используется в модели, **А** интерпретатор ведет учет числа устройств, занятых в каждый момент времени. Интерпретатор также автоматически подсчитывает такие СЧА: число транзактов, которые вошли в МКУ; среднее число каналов, занятых одним транзактом; среднее время нахождения транзакта в устройстве и др.

Некоторые события в системе могут заблокировать или изменить движение транзактов. Например, кассир кинотеатра, идя на обед, ставит табличку «В другое окно», и все следующие клиенты на протяжении обеда обращаются в другую кассу. Для моделирования этих ситуаций введены логические ключи. Транзакт может устанавливать эти ключи в положение «Включено» или «Выключено». Через некоторое время состояние ключа может быть использовано другими транзактами для выбора одного из двух возможных путей движения или ожидания момента изменения состояния ключа. Состояние ключа может быть изменено любым транзактом.

Переменные. ***Арифметические переменные*** позволяют вычислять арифметические выражения, которые состоят из операций над СЧА объектов. В выражениях могут быть использованы функции (библиотечные или пользовательские). ***Булевы переменные*** позволяют пользователю одновременно проверять несколько условий, исходя из состояния объектов или значений СЧА.

Функции. Используя ***функции,*** пользователь может задавать непрерывную или дискретную функциональную зависимость между аргументом функции и ее значением. Функции в GPSS задаются табличным способом с помощью операторов описания функций.

Ячейки и матрицы сохраняемых величин. Ячейки сохраняемых величин и матрицы используются для хранения некоторой пользовательской числовой информации, запись в эти объекты выполняют транзакты. Записанную в этих объектах информацию может считывать любой транзакт. Таким образом, эти объекты являются глобальными и доступны из любой части модели.

Очереди. В любой системе движение потока транзактов может быть задержано из-за недоступности ресурсов (например, необходимые устройства или МКУ уже заняты). В этом случае задержанные транзакты становятся в ***очередь*** – еще один тип объектов GPSS. Учет этих очередей составляет одну из основных функций интерпретатора. Пользователь может специально определить точки модели, в которых необходимо собирать статистику об очередях, то есть установить ***регистраторы очереди.*** Тогда интерпретатор будет автоматически собирать статистику об очередях (длину очереди, среднее время нахождения в очереди и т.п.). Вся эта информация является СЧА и доступна пользователю в процессе моделирования.

Интерпретатором автоматически поддерживается дисциплина обслуживания очереди FIFO («первым пришел – первым обслужился»), и пользователь может получить стандартную статистическую информацию только об этих очередях. Если у пользователя возникает необходимость организовать очередь из транзактов с другой дисциплиной обслуживания (например, LIFO), то для этого используются ***списки пользователей.*** Эти списки также помогают осуществлять синхронизацию движения разных транзактов по модели.

Таблицы. Объект ***«таблица»*** предназначен для сбора статистики о случайных величинах, заданных пользователем. Таблица состоит из частотных классов, в которые заносится число попаданий конкретной величины (некоторого СЧА). Для каждой таблицы вычисляется математическое ожидание и среднеквадратическое отклонение.

4.2. ЧАСЫ модельного времени

Разные события реальных систем происходят в течение некоторого периода времени. Например, покупатели приходят в магазин, когда подходит их очередь, они попадают на обслуживание. Когда покупки сделаны, покупатели покидают магазин. Если все эти события представить в модели, то их возникновение должно происходить на фоне модельного времени. Интерпретатор автоматически обслуживает ЧАСЫ модельного времени.

В момент начала моделирования интерпретатор планирует появление первого транзакта. После этого ЧАСЫ модельного времени устанавливаются на значение времени, которое соответствует моменту появления первого транзакта в модели. Этот транзакт (и другие, если они приходят в этот же момент времени) входит в модель. Далее он передвигается через все возможные блоки модели, которые ему встречаются. События, которые возникают вследствие перемещения транзакта через блоки, планируются на дальнейшие моменты времени. Естественно, что в этот первый отмеченный момент времени ничего больше в системе не происходит. Интерпретатор GPSS продвигает дальше значения ЧАСОВ к тому значению времени, на которое запланировано следующее ближайшее событие. Если во второй, отмеченный ЧАСАМИ момент времени, нет транзактов, которые нужно перемещать. ЧАСЫ снова продвигаются вперед и т.д. Именно так, от события к событию, и происходит смена модельного времени.

Особенности ЧАСОВ GPSS:

1. ЧАСЫ в GPSS регистрируют целые значения (за исключением языка GPSS World, где время может иметь действительные значения), то есть события могут появляться только в целые моменты времени. Это сделано с целью ускорения процесса моделирования, поскольку целочисленная арифметика выполняется процессором ЭВМ быстрее и требует меньше памяти.
2. Единица модельного времени определяется разработчиком. Эту единицу времени интерпретатору не сообщают. Значение принятой единицы модельного времени выражают в неявном виде в форме временных данных модели. Так, если все данные выражены в минутах, то единицей времени будет

минута, то есть масштаб времени в модели будет такой: одна единица модельного времени равна одной минуте реального времени. Если все данные выражены в миллисекундах, то единицей модельного времени будет миллисекунда. Разработчик может задавать такую единицу времени, которая ему удобна для того, чтобы правильно отобразить события реальной системы в модели.

3. Система GPSS является интерпретатором «следующего события». Иначе говоря, после того, как модель полностью скорректирована в данный момент дискретно изменяющегося времени, ЧАСЫ перемещаются к следующему моменту времени, на который запланировано следующее событие. Таким образом, ЧАСЫ модельного времени продвигаются от одного события к другому.

4.3. Типы операторов

Операторы GPSS делятся на три типа:
1) блоки;
2) операторы описания данных;
3) команды GPSS.

Общие сведения о формате операторов GPSS. В GPSS для ссылки на числа, блоки и объекты используются ***имена*** (идентификаторы). Имя представляет собой алфавитно-цифровую последовательность длиной до 20 символов в GPSS/PC и до 250 символов в GPSS World, которая начинается с буквы. Допускается использование символов только латинского алфавита, цифр и знака подчеркивания.

Формат GPSS-блоков такой:

***[Номер строки*] [< *Метка* >] < *Операция* > < *Операнды* > <; *Комментарии* >**

Номер строки. Обязательное поле для GPSS/PC (в GPSS World – игнорируется). Начинается с первой позиции строки. Представляет собой десятичное число.

Метка (имя блока). Содержимым поля является имя – последовательность символов, начинающаяся с буквы. В некоторых операторах это поле является обязательным.

Операция. Операциями* блоков являются глаголы, которые описывают основные функциональные назначения блоков. Каждый из блоков характеризируется своим собственным предписанным ему глаголом.

* Термин используется согласно [10]. В GPSS World для чтого поля используется термин Verb – глагол.

Операнды. Блоки могут иметь операнды. Операнды блоков задают информацию, специфичную для действия данного блока. Число операндов блока зависит от типа блока. В блоках не может использоваться больше семи операндов. Операнды в общем случае обозначаются символами**: A, B, C**, **D, E, F, G**. Значения операндов определяются типом блока. Одни операнды некоторых блоков должны быть определены всегда, а другие могут задаваться или не задаваться (т.е. являются необязательными). Операнды следуют один за другим и отделяются запятыми или одним пробелом. Если операнд опущен, то вместо него ставится запятая. Между операндами не должно быть более одного пробела, так как это будет означать, что операнды закончились и интерпретатор прекращает чтение строки.

Комментарии. Необязательное поле. Комментарии отделяются от поля операндов символом «;». Допускается запись комментария с начала строки. В этом случае в первой позиции строки ставится символ «;» или «*». В GPSS/PC допускаются комментарии с использованием заглавных или строчных букв только латинского алфавита, в GPSS World также допускается использование символов кириллицы.

Строка описания блока может содержать до 79 символов в GPSS/PC и до 250 символов в GPSS World. При описании форматов квадратные скобки [ ] указывают на необязательность поля.

Именами и метками не могут быть названия или начальные символы названий блоков, операторов, команд и СЧА. Во избежание конфликтов с ключевыми словами рекомендуется в именах использовать символ подчеркивания.

4.4. Внесение транзактов в модель. Блок GENERATE

Блок GENERATE (ГЕНЕРИРОВАТЬ) – это блок, через который транзакты входят в модель. Не существует ограничений на количество разных блоков **GENERATE** в одной модели.

Интервал времени между последовательными появлениями транзактов из блока **GENERATE** называют ***интервалом поступления.*** Когда транзакт входит в модель через блок **GENERATE**. интерпретатор планирует время поступления следующего транзакта путем розыгрыша случайного числа с соответствующим распределением интервалов поступления на время, равное текущему значению ЧАСОВ плюс разыгранное значение. При достижении этого значения модельного времени следующий транзакт вводится в модель через блок **GENERATE** и т.д.

Разработчик должен задать функцию распределения интервалов поступления транзактов в блоке **GENERATE**.

Все возможные виды случайных распределений интервалов поступления транзактов в GPSS делятся на равномерное распределение и другие виды распределений. В нашем случае специально рассматривают самое простое из всех случайных нетривиальных распределений – равномерное распределение. Использование других видов распределений требует задания функций, которые описаны ниже (см. параграф 4.13).

Формат блока:

GENERATE [A],[B],[C],[D],[E]

Таблица 4.2

Операнд	***Значение***	***Значение по умолчанию****
A	Средний интервал времени (число, СЧА)	0
B	Половина поля допуска равномерно распределенного интервала (число, СЧА)	0
C	Смещение интервалов	Смещение отсутствует
D	Ограничитель транзактов	∞
E	Уровень приоритета транзакта. Возможные значения 0 – 127	0

* Если опущено поле операнда, транслятор проставляет значения по умолчанию или выдает ошибку.

Значение операндов:

A – среднее значение интервала поступления;

B – величина разброса возможных значений относительно среднего значения. (Если операнд **B** не задается, то интервал времени поступления – детерминированная величина);

C – момент времени, в который в блоке **GENERATE** должен появиться первый транзакт. (После этого первого прихода все остальные приходы транзактов возникают в соответствии с распределением, заданным операндами **A** и **B**);

D – ограничитель общего числа транзактов, которое может войти в модель через данный блок **GENERATE** на протяжении времени моделирования. (Если это число достигнуто, данный блок **GENERATE** перестает быть активным);

E – уровень или класс приоритета каждого из транзактов. которые вводятся в модель через данный блок **GENERATE.** (Всего существует 128 разных уровней, которые задаются с помощью чисел от 0 до 127. Чем больше число, тем выше приоритет).

1. Транзакты не могут входить в блок **GENERATE,** так как он сам их генерирует.
2. Если в модели GPSS/PC встречаются подряд два или больше блоков **GENERATE,** то последний блок переопределяет операнды предыдущих блоков. В GPSS World транслятор выдает ошибку.
3. Операнды не могут быть отрицательными числами.

Операнды **A, B, C** целочисленные (в GPSS World могут быть действительными числами).

Пример 4.1

1. Задание равномерного закона распределения:

GENERATE 6,4

Операнды: **A** = 6, **B** = 4. Интервал времени поступления является случайным числом со средним значением 6 и полем допуска 8, то есть он может приобретать только одно из девяти разных значений: 2, 3,4,5,6,7,8,9, 10.

2. Задание детерминированного значения интервалов поступления:

GENERATE 10

Операнды: **A** = 10, **B** = 0 (по умолчанию). Транзакты входят в модель каждые 10 единиц модельного времени.

3. Генерирование одного транзакта.

GENERATE ,,,1

Операнды: **A = B = C** = 0 (по умолчанию), **D** = 1. В нулевой момент в модель входит один транзакт.

Блоки **GENERATE** являются основными средствами создания фанзактов и ввода их в модель. Кроме блока **GENERATE,** для ввода гранзактов в модель используется также блок **SPLIT,** который создаст заданное число копий транзактов, вошедших в блок.

4.5. Удаление транзактов из модели. Блок TERMINATE

Транзакты удаляются из модели, попадая в блок **TERMINATE (ЗАВЕРШИТЬ).** В этот момент освобождается память, выделенная под транзакт. Эти блоки всегда позволяют выйти всем транзактам, которые пытаются это сделать. В модели может быть любое количество блоков **TERMINATE.** Формат блока:

TERMINATE [A]

Операнд **A** является величиной уменьшения специального счетчика, который называется ***счетчиком завершения.*** Этот операнд задает величину, которая вычитается из счетчика каждый раз, когда транзакт входит в блок **TERMINATE**. По умолчанию **A** = 0. Вход гранзакта в блок **TERMINATE** с нулевым значением операнда **A** не вызывает уменьшения счетчика завершения.

Счетчик завершения – это ячейка в памяти ЭВМ, которая хранит целое положительное число. Начальное значение этого счетчика устанавливается в начале моделирования. Оно равняется значению операнда **A** команды **START** (НАЧАТЬ). В процессе моделирования транзакты попадают в блок **TERMINATE** и, таким образом, уменьшают значение счетчика на величину операнда **A**. Моделирование заканчивается, когда значение счетчика становится равным нулю или отрицательному числу.

1. В модели может быть много блоков **TERMINATE,** но счетчик завершения – один, с начальным значением, указанным в команде **START.**

2. Не путать ограничитель транзактов в блоке **GENERATE** и счетчик завершения. Ограничитель задает число транзактов, которые войдут в модель, **A** счетчик – число транзактов, которые выйдут из модели. По окончании моделирования транзакты могут оставаться в модели.

Интерпретатор начинает моделирование по команде **START.** Ее формат:

START A,[B],[C],[D]

В операнде **A** задается начальное значение счетчика завершения. О назначении остальных операндов будет рассказано в параграфе 4.27.

Управление продолжительностью процесса моделирования.

В языке GPSS продолжительностью процесса моделирования можно управлять двумя способами:

1) завершать моделирование после того, как модель покинет заданное число транзактов определенного типа;

2) завершать моделирование по истечению заданного интервала времени.

Первый способ:

1. В команде **START** операнду **A** присваивается значение заданного числа транзактов.

2. Во всех блоках **TERMINATE,** через которые транзакты заданного типа покидают модель, операнду **A** присваивается значение «1» или другое, отличное от нуля (соответственно содержательному значению транзактов).

3. Во все других блоках **TERMINATE** используется значение операнда **A** по умолчанию (**A** = 0). Значение счетчика завершения не будет зависеть от этих блоков.

Первый способ позволяет закончить моделирование, когда через модель пройдет заданное количество транзактов, например 1000:

```
GENERATE    40,5
...
TERMINATE   1
START       1000
```

Второй способ:

Пусть разработчик выбрал за единицу модельного времени 1 мин и хочет смоделировать поведение системы на протяжении 8 часов. Это можно сделать таким образом:

1. Ввести в модель таймер-сегмент, состоящий из двух блоков:

```
GENERATE    480
TERMINATE   1
```

2. Во всех других блоках **TERMINATE** в модели использовать значение операнда **A** по умолчанию (**A** = 0). Это означает, что прекращение моделирования, определяемое счетчиком завершения, не будет зависеть от других блоков **TERMINATE.**
3. В команде **START** операнд **A** должен равняться единице.

Таким образом, в процессе моделирования завершение движения транзактов в других блоках **TERMINATE** не влияет на счетчик завершения. В момент времени 480 транзакт выйдет из блока **GENERATE** и сразу же перейдет в блок **TERMINATE.** Счетчик завершения уменьшится на единицу, и интерпретатор завершит моделирование.

4.6. Элементы, отображающие одноканальные обслуживающие устройства

Рассмотрим элементы, которые используются для представления обслуживания. Аналогами обслуживающих элементов могут быть люди, механизмы, линии связи и другие объекты реальных систем. В GPSS такие объекты моделируются с помощью устройств, МКУ, логических ключей.

Устройство характеризируется двумя основными свойствами:

1. Каждое устройство в любой момент времени может обслуживать только один транзакт. Если в процессе обслуживания появляется новый транзакт, то он должен:
   1) либо подождать своей очереди,
   2) либо направиться в другое место,
   3) либо, если вновь пришедший транзакт имеет больший приоритет, устройство прерывает текущее обслуживание и начинает обслуживать новый транзакт.
2. Когда транзакт поступает в устройство, он должен пробыть там необходимое для обслуживания время.

Всем устройствам необходимо задавать имена. Они могут быть или числовыми (числа должны быть положительными целыми), или символьными. Во время трансляции символьным именам сам транслятор присваивает числовые значения.

Для того, чтобы использовать одноканальное обслуживающее устройство (прибор), транзакту необходимо выполнить следующие шаги.

Первый шаг. Ждать своей очереди, если это необходимо. Ожидание длится в течение некоторого интервала времени.

Второй шаг. Когда подходит очередь, занять устройство. Событие «занятие устройства» происходит в некоторый момент модельного времени.

Третий шаг. Устройство находится в состоянии занятости до тех пор, пока не закончится обслуживание. Для обслуживания необходим некоторый интервал времени.

Четвертый шаг. Когда обслуживание закончится, освободить устройство. Событие «освобождение устройства» происходит в некоторый момент модельного времени.

Эта последовательность шагов выполняется GPSS при моделировании использования устройства. Второй и четвертый шаги реали-5уются блоками **SEIZE (ЗАНЯТЬ)** и **RELEASE (ОСВОБОДИТЬ).**

Формат блока:

SEIZE A

Таблица 4.3

Операнд	*Значение*	*Результат по умолчанию*
A	Имя (символьное или числовое) занимаемого устройства	Ошибка

Этот блок имеет следующие свойства:

1. Если в текущий момент времени устройство используется, то гранзакт не может войти в блок и должен ожидать своей очереди.

2. Если устройство свободно, транзакт может войти в блок. Вход транзакта в блок вызывает выполнение подпрограммы обработки этого блока. Состояние устройства изменяется со СВОБОДНОЕ на ЗАНЯТОЕ.

Предварительного объявления устройства в модели не требуется, так как тот факт, что блок **SEIZE** используется, свидетельствует о существовании данного устройства.

Предназначением блока **RELEASE** является изменение состояния ранее занятого устройства с ЗАНЯТОГО на СВОБОДНОЕ. Блок **RELEASE** никогда не запрещает вход транзакта.

Формат блока:

RELEASE A

Таблица 4.4

Операнд	*Значение*	*Результат по умолчанию*
A	Имя (символьное или числовое) освобождаемого устройства	Ошибка

В то время, как транзакты находятся в модели временно, устройства, используемые в модели, существуют в ней в течение всего периода моделирования.

Статистическая информация о работе устройства при моделировании собирается автоматически.

Если в модели используются объекты типа «устройство», то в файле стандартной статистики будет представлена информация об использованных устройствах.

Таблица 4.5

FACILITY Номер или имя устройства	**ENTRIES** Количество входов	**UTIL** Коэффициент использования	**AVE**, **TIME** Среднее время пребывания транзакта в устройстве	**AVAIL**. Состояние готовности
1	50	0,07	74,06	
2	51	0,10	100,29	

OWNER Номер последнего транзакта, занявшего устройство	**PEND** Количество прерванных в устройстве транзактов	**INTER** Количество прерывающих устройство транзактов	**RETRY** Количество транзактов, ожидающих специальных условий	**DELAY** Количество транзактов, ожидающих занятия устройства

Статистику работы устройств в процессе моделирования можно наблюдать в окне устройств для GPSS/PC (перейдя в это окно с помощью клавиш [ALT+F]) или в окне Facilities Window для GPSS World.

1. После блока **SEIZE** может сразу же следовать другой блок SEIZE, если транзакт должен одновременно занять два или более устройств (например, рабочего и инструмент).

2. Транзакт не может освободить устройство, которое он занимал.

4.7. Реализация задержки во времени. Блок ADVANCE

Перевод с английского языка блока **ADVANCE (ЗАДЕРЖАТЬ)** – продвигать, **А** не задерживать. Этот блок действительно продвигает ЧАСЫ модельного времени на некоторое значение, но фактически он осуществляет задержку продвижения транзакта в течение некоторого интервала времени. Обычно этот интервал задается случайной величиной.

В GPSS возможны следующие варианты распределения времени обслуживания:

1) детерминированное (постоянное);

2) равномерное распределение;

3) другие распределения.

Как и при использовании блока **GENERATE** особо рассматривается равномерное распределение случайных величин. Применение более сложных видов распределений требует использования дополнительных функций (см. параграф 4.13).

Формат блока:

ADVANCE A[,B]

Таблица 4.6

Операнд	***Значение***	***Значение по умолчанию***
A	Среднее время задержки на обслуживание (число, СЧА)	0
B	Половина поля допуска равномерно распределенного времени задержки (число, СЧА)	0

Блок никогда не препятствует входу транзакта. Любое число транзактов может находиться в этом блоке одновременно. Когда транзакт попадает в такой блок, выполняется соответствующая подпрограмма и вычисляется время пребывания в нем транзакта. Вновь прибывший транзакт никак не влияет на уже находящийся в блоке транзакт.

Если время пребывания в блоке равно нулю, то вместо задержки в блоке ADVANCE интерпретатор сразу же пытается переместить этот транзакт в следующий блок. Более подробно о взаимодействии блока **ADVANCE** с интерпретатором описано в параграфе 4.21.

1. В GPSS/PC не допускаются дробные значения времени задержки.
2. Отрицательное значение задержки всегда вызывает ошибку.

Пример 4.2

Использование блока **ADVANCE**:

```
ADVANCE     30,5
```

Время задержки транзакта в этом блоке – случайная величина, равномерно распределенная на интервале [25, 35], которая принимает одно из 11 целых значений.

Пример 4.3

Классический случай использования последовательности **SEIZE – ADVANCE – RELEASE:**

```
SEIZE       PRIB
ADVANCE     16,4
RELEASE     PRIB
```

Транзакт, двигаясь по этой цепочке блоков, займет устройство с именем PRIB, задержится там на 16±4 единицы времени и затем покинет его. После того как транзакт войдет в блок **RELEASE** и соответствующая этому блоку подпрограмма закончится, интерпретатор попытается переместить транзакт в следующий блок модели и следующий транзакт может уже использовать устройство PRIB.

Блоки **ADVANCE** можно располагать в любых местах программы, **А** не только между блоками **SEIZE** и **RELEASE.**

4.8. Сбор статистики об ожидании. Блоки QUEUE, DEPART

Эти блоки обеспечивают в GPSS возможность автоматического сбора статистических данных, описывающих вынужденное ожидание, которое может происходить время от времени в различных точках модели.

Система моделирования GPSS обеспечивает возможность сбора статистики с помощью такого средства, как ***регистратор очереди.***

При использовании регистратора очереди в тех точках модели, где число ресурсов ограничено, интерпретатор автоматически начинает собирать различную информацию об ожидании с помощью СЧА, **А** именно:

1) число входов транзактов в очередь;

2) количество транзактов, которые фактически присоединились к очереди и сразу ее покинули, т.е. имели время ожидания равное нулю;

3) максимальная длина очереди;

4) среднее число ожидавших транзактов;

5) среднее время ожидания тех транзактов, которым пришлось ждать.

В модели может быть несколько регистраторов очередей, различающихся именами. Правила присвоения имен те же, что и для устройств. Разработчик вносит регистратор очереди в модель с помощью пары взаимодополняющих блоков:

```
QUEUE     A[,B]
DEPART    A[,B]
```

Таблица 4.7

Операнд	***Значение***	***Результат по умолчанию***
А	Имя очереди, в которую необходимо стать транзакту или которую надо покинуть (числовое или символическое имя, СЧА)_	Ошибка
В	Число единиц, на которое увеличивается (уменьшается) длина очереди (число, СЧА)	1

При входе транзакта в блок **QUEUE (СТАТЬ В ОЧЕРЕДЬ)** выполняются четыре действия:

1) ***счетчик входов*** для данной очереди увеличивается на **В**;

2) ***длина очереди (счетчик текущего содержимого)*** для данной очереди увеличивается на **В**;

3) значение текущей ***длины очереди*** хранится в стандартном числовом атрибуте **Q$<имя очереди>;**

4) транзакт присоединяется к очереди с запоминаем ее имени и значения текущего модельного времени.

Транзакт перестает быть элементом очереди только после того, как он переходит в блок **DEPART (ПОКИНУТЬ ОЧЕРЕДЬ**) соответствующей очереди. Когда это происходит, интерпретатор выполняет такие операции:

1) ***длина очереди*** соответствующей очереди уменьшается на **В**;

2) используя привязку к значению времени, определяет: является ли время, проведенное транзактом в очереди, нулевым; если да, то такой транзакт по определению является транзактом с ***нулевым пребыванием*** в очереди и одновременно изменяется ***счетчик нулевых вхождении,***

3) ликвидируется «привязка» транзакта к очереди.

Если в модели используются объекты типа «очередь», то в файле стандартной статистики будет представлена информация об этих объектах. В конце моделирования интерпретатор автоматически выдает статистические данные: значение счетчика входов, максимальное значение длины очереди, среднее значение длины очереди, текущее значение длины очереди в конце периода моделирования, среднее значение времени нахождения в очереди и т.д.

Статистическая информация об ожидании выдается в следующем виде:

QUEUE Номер или имя очереди	MAX Максимальная длина очереди	CONT. Текущая длина очереди	ENTRY Общее кол-во входов	ENTRY(0) Количество «нулевых» входов	AVE. CONT. Средняя длина очереди	AVE.TIME Среднее время пребывания транзактов в очереди	AVE.(-0) Среднее время пребывания в очереди без учета «нулевых» входов	RETRY Количество транзактов, ожидающих специальных условий
ZZZ	4	0,04	894	792	88,59	1,37	15,57	

Пример 4.4

Пусть необходимо собрать статистику об ожидании в очереди при обслуживании устройством PRIB, тогда в сегмент модели будут введены блоки **QUEUE** и **DEPART:**

```
QUEUE     QPRIB
SEIZE     PRIB
DEPART    QPRIB
ADVANCE   16,4
RELEASE   PRIB
```

В этом примере все транзакты, попадающие в устройство, должны пройти через пару **QUEUE – DEPART** даже тогда, когда устройство свободно и его можно сразу же занять.

Пример 4.5

Увеличение на единицудлины **Q$QPR1** очереди **QPR1:**

QUEUE QPR1

Увеличение на две единицы длины **Q$QPR2** очереди **QPR2:**

QUEUE QPR2,2

Уменьшение на единицу длины **QSQWORKER** очереди **QWORKER:**

DEPART QWORKER

1. Когда транзакт входит в блок **QUEUE**, то ищется очередь с именем, определенным операндом **A**. При необходимости очередь создается.

2. Блок **QUEUE** не поддерживает список членов очереди, он только добавляет единицы к длине очереди.

3. Использование регистратора очереди необязательно. С его помощью интерпретатор собирает лишь статистику об ожидании. Если же регистратор не используется, то статистика не собирается, но везде, где должна возникать очередь, она возникает. Ожидание является следствием состояния устройства, **A** не следствием использования регистратора. Если в планы не входит обработка статистических данных об очередях, то лучше не собирать статистику – это сэкономит время, расходуемое на моделирование.

4. Один и тот же транзакт может одновременно увеличить длину нескольких очередей.

5. При выходе транзакта из очереди через блок **DEPART** транзакту не обязательно уменьшать длину очереди на ту же величину, на которую он увеличил ее при входе в блок **QUEUE**. Но в итоге число входов в очередь должно равняться числу выходов из нее.

4.9. Переход транзакта в блок, отличный от последующего. Блок TRANSFER

В GPSS блок **TRANSFER (ПЕРЕДАТЬ)** может быть использован в девяти разных режимах. Рассмотрим три основных.

Блок TRANSFER в режиме безусловной передачи. Его формат:

TRANSFER ,B

Таблица 4.8

Операнд	*Значение*	*Результат по умолчанию*
A	Не используется	—
B	позиция блока, в которую должен перейти транзакт	Ошибка

Позиция блока – это номер или метка блока. Так как операнд **A** не используется, то перед операндом B должна стоять запятая. В режиме безусловной передачи блок **TRANSFER** не может отказывать транзакту во входе. Кстати, если транзакт входит в блок, то он сразу же пытается войти в блок **B**.

Транслятор GPSS/PC не улавливает пропущенную запятую вместо операнда **A** (например, TRANSFER LAMD). На этапе трансляции метке LAMD присваивается числовое значение, и транзакт в этом случае направляется в блок с соответствующем номером.

Статистический режим. В этом режиме осуществляется передача транзакта в один из двух блоков случайным образом. Формат блока:

TRANSFER A,[B],C

Таблица 4.9

Операнд	***Значение***	***Результат по умолчанию***
A	Вероятность передачи транзакта в блок С, задаваемая в долях тысячи	Ошибка
B	Позиция блока, в которую должен перейти транзакт (с вероятностью 1 – A)	Следующий по порядку блок
C	Позиция блока, в которую должен перейти транзакт (с вероятностью A)	Ошибка

При задании вероятности (операнд A) используется не более трех цифр, первый символ записи частоты «.» (десятичная точка), если используется действительное число, которое должно быть в пределах от 0 до 1,0 (например, 0,235). Если операнд – положительное целое число, то вероятность интерпретируется в долях тысячи.

Пример 4.6

```
         TRANSFER   .333,LPRIB1,LPRIB2
         ...
LPRIB1   SEIZE      PR1
         ...
LPRIB2   QUEUE      QPR2
```

С частотой 0,667 транзакт переходит в блок с меткой **LPRIB1** и с частотой 0,333 – в блок с меткой **LPRIB2.**

Пример 4.7

```
         TRANSFER   4,,LPRIB2
         SEIZE      PR1
         ...
LPRIB2   QUEUE      QPR2
```

С частотой 0,6 транзакт переходит в блок **SEIZE PR1** и с частотой 0,4 – в блок с меткой **LPRIB2.**

Режим BOTH. Если в операнде **A** стоит зарезервированное слово **BOTH,** то блок **TRANSFER** работает в режиме **BOTH.**

В этом режиме входящий транзакт сначала пытается перейти к блоку, указанному в операнде **B**. Если это сделать не удается, транзакт пытается перейти в блок, указанный в операнде **C**. Если транзакт не сможет перейти ни к тому, ни к другому блоку, то он остается в блоке **TRANSFER** и при каждом просмотре списка текущих событий, будет повторять в том же порядке попытки перехода до тех пор, пока не сможет выйти из блока **TRANSFER.**

Пример 4.8

```
         TRANSFER   BOTH,LL1,LL2
         ...
LL1      SEIZE      PR11
         ...
LL2      SEIZE      PR12
```

Транзакт сначала пытается перейти в блок с меткой **LL1.** Если устройство **PRI1** занято, транзакт пытается войти в блок с меткой **LL2.** Если транзакт не может войти и в этот блок (устройство **PRI2** также занято), он остается в списке текущих событий и повторяет эти попытки при каждом просмотре списка до тех пор, пока не выйдет из блока **TRANSFER.**

1. Не путайте метку блока **SEIZE** с именем соответствующего этому блоку устройства.

2. Если бы меткой **LL1** был помечен блок **QUEUE**, **A** не блок **SEIZE,** то все транзакты были бы направлены по метке **LL1,** так как в отличие от блока **SEIZE** блок **QUEUE** всегда готов принять транзакты.

4.10. Моделирование многоканальных устройств

Устройство в GPSS используют для моделирования одиночного устройства обслуживания. Два или более обслуживающих устройства, работающих параллельно, могут моделироваться в GPSS двумя или более одноканачьными устройствами. Обычно это необходимо, когда отдельные устройства являются разнородными, например, имеют различную интенсивность обслуживания.

Однако очень часто параллельно работающие устройства являются одинаковыми, и GPSS предоставляет для их моделирования объект, называемый многоканальным устройством (МКУ).

Количество устройств, которое моделируется каждым из МКУ, определяется пользователем. В этом смысле употребляют термин ***«емкость МКУ»***. Эта емкость заранее должна быть определена пользователем, чтобы интерпретатор знал, сколько устройств использует данное МКУ.

Блоки ENTER (ВОЙТИ) и **LEAVE** (ВЫЙТИ). Использование МКУ аналогично использованию одиночного устройства. Элементом, который занимает и использует МКУ, является транзакт. При моделировании МКУ события происходят в следующем порядке:

1) транзакт ожидает своей очереди, если это необходимо;
2) транзакт занимает устройство;
3) устройство осуществляет обслуживание на протяжении некоторого интервала времени;
4) транзакт освобождает устройство.

Блоки **ENTER** и **LEAVE** моделируют события 2 и 4. Формат блоков:

```
ENTER     A[,B]
LEAVE     A[,B]
```

Таблица 4.10

Операнд	***Значение***	***Результат по умолчанию***
A	Имя МКУ	Ошибка
B	Количество занимаемых одновременно устройств	1

Когда транзакт входит в блок **ENTER**, интерпретатор выполняет следующие действия:

1) увеличивает ***счетчик входов*** МКУ на значение операнда **B**;
2) увеличивает ***текущее содержимое*** МКУ на значение операнда **B**;
3) уменьшает ***доступную емкость*** МКУ на значение операнда **B**.

Когда транзакт входит в блок LEAVE, интерпретатор выполняет обратные действия:

1) уменьшает ***текущее содержимое*** МКУ на значение операнда **B**;
2) увеличивает ***доступную емкость*** МКУ на значение операнда **B**.

Операнду **B** можно присвоить значение, отличное от единицы.

Например, пусть транзакт моделирует корабль, **A** МКУ – причалы в порту. В зависимости от размера корабль может занимать нескольких причалов, т.е. **B**>1.

Если в модели используются объекты типа МКУ, то в файле стандартной статистики об этих объектах будет представлена такая информация:

STORAGE	**CAP.**	**REMAIN**	**MIN**	**MAX**	**ENTRIES**	**AVL.**	**AVE.C.**	**UTIL**	**RETRY**	**DELAY**
RPOOL	3	3	0	1	50	1	0,99	0,33	0	0

Поле **STORAGE** определяет имя или номер МКУ.

Поле **CAP.** определяет емкость МКУ, заданную оператором **STORAGE**

Поле **REMAIN** определяет количество единиц свободной емкости МКУ в конце периода моделирования.

Поле **MIN** определяет минимальное количество используемой емкости МКУ за период моделирования.

Поле **MAX** определяет максимальное количество используемой емкости МКУ за период моделирования.

Поле **ENTRIES** определяет количество входов в МКУ за период моделирования.

Поле **AVL**. определяет состояние готовности МКУ в конце периода моделирования: 1 – МКУ готов, 0 – не готов.

Поле **AVE**.C определяет среднее значение занятой емкости за период моделирования.

Поле **UTIL.** определяет средний коэффициент использования всех устройств МКУ.

Поле **RETRY** определяет количество транзактов, ожидающих специальных условий, зависящих от состояния МКУ.

Поле **DELAY** определяет количество транзактов. ожидающих возможности входа в блок **ENTER.**

Для GPSS/PC статистику о работе МКУ можно наблюдать в окне МКУ, перейдя в это окно с помощью клавиш [ALT+S]. **A** для GPSS World – в окне Storages Window.

Определение емкости МКУ. Все используемые в модели МКУ должны быть заранее описаны, т.е. должно быть определено количество однотипных устройств, входящих в МКУ. Для этого используется оператор **STORAGE** (ХРАНИЛИЩЕ или ПАМЯТЬ), определяющий емкость МКУ. Название **STORAGE** становится понятным, если представить себе, что МКУ это автоматизированный склад или многоэтажный гараж с определенным числом мест, которое и задает этот оператор. В таких случаях МКУ определяет не количество одинаковых устройств для обслуживания, **A** количество одинаковых мест для хранения.

Формат оператора задания емкости МКУ:

Таблица 4.11

Поле	***Информация в поле***
Метка	Символическое имя МКУ
Операция	**STORAGE**
Операнд **A**	Емкость МКУ

Пример 4.9

Пусть система состоит из восьми механиков и десяти подъемных кранов, тогда в GPSS-модель могут быть введены такие МКУ:

```
MECHANICIAN     STORAGE    8
LIFTING_CRANE   STORAGE    10
```

Существует возможность периодически переопределять емкость МКУ при необходимости выполнения нескольких прогонов за один этап моделирования. Это делается введением в программу между операторами **START** предыдущего прогона и оператором **START** последующего прогона нового определения емкостей.

Пример 4.10

Необходимо найти наилучшую комбинацию среди значений пар «количество механиков – количество кранов» за один этап моделирования. Для этого нужно использовать следующую последовательность операторов **STORAGE** и команд **START:**

```
MECHANICIAN     STORAGE     <значение 1.1>
LIFTING_CRANE   STORAGE     <значение 1.2>
        < текст GPSS-программы >
                START       1
                RESULT      REZ.TXT,1,1112
                CLEAR
MECHANICIAN     STORAGE     <значение 2.1>
LIFTING_CRANE   STORAGE     <значение 2.2>
                START       1
                RESULT      REZ.TXT,1,2122
                CLEAR
MECHANICIAN     STORAGE     <значение 3.1>
LIFTING_CRANE   STORAGE     <значение 3.2>
                START       1
                RESULT      REZ.TXT,1,3132
                …
                CLEAR

MECHANICIAN     STORAGE     <значение n.1>
LIFTING_CRANE   STORAGE     <значение n.2>
                START       1
                RESULT      REZ.TXT,1,n1n2
```

Действия, выполняемые оператором **CLEAR** описаны в параграфе 4.27. Команда **RESULT** используется только в GPSS/PC. Отметим, что второй параметр команды **RESULT** указывает на ячейку сохраняемых величин, в которой хранится значение критерия, по которому сравнивают комбинации пар значений.

4.11. Примеры построения GPSS-моделей

Пример4.11 [10]

Интервалы прихода клиентов в парикмахерскую с одним креслом распределены равномерно на интервале 18±6мин. Время стрижки также распределено равномерно на интервале 16±4мин. Клиенты приходят в парикмахерскую, стригутся в порядке очереди: «первым пришел – первым обслужился». Необходимо построить GPSS-модель парикмахерской, которая должна обеспечить сбор статистических данных об очереди. Промоделируйте работу парикмахерской в течение 8 часов.

Построение модели

Порядок блоков в модели соответствует порядку фаз, в которых клиент оказывается при движении в реальной системе:

1) клиент приходит;
2) если необходимо, ждет своей очереди;
3) садится в кресло парикмахера;
4) парикмахер обслуживает клиента;
5) клиент уходит из парикмахерской.

Таблица 4.12 (Таблица определений)

Элементы GPSS	***Интерпретация***
Транзакты	
В первом сегменте модели	Клиенты
Во втором сегменте модели	Таймер
Устройство	
BARBER	Парикмахер
Очередь	
BARBERQ	Очередь, используемая для сбора статистики об ожидании клиентов

Единица модельного времени – 1 минута.

Программа:

```
MODEL SEGMENT 1
GENERATE    18,6        ; Приход клиентов
QUEUE       BARBERQ     ; Присоединение к очереди
SEIZE       BARBER      ; Переход в кресло парикмахера
DEPART      BARBERQ     ; Выход из очереди
ADVANCE     16,4        ; Обслуживание у парикмахера
RELEASE     BARBER      ; Освобождение парикмахера
TERMINATE   0           ; Уход из парикмахерской
MODEL SEGMENT 2
GENERATE    480         ; Транзакт-таймер
TERMINATE   1           ; Завершение прогона
START       1
```

Пример 4.12[10]

В парикмахерскую с одним креслом приходят клиенты двух типов. Клиенты первого типа желают только стричься. Распределение интервалов их прихода – 35±10 мин. Клиенты второго типа желают постричься и побриться. Распределение интервалов их прихода – 60±20 мин. Парикмахер обслуживает клиентов в порядке «первым пришел – первым обслужился». Время, затраченное на стрижку, составляет 18±6мин, **А** на бритье – 10±2мин. Написать GPSS-модель парикмахерской, обеспечив сбор данных об очереди клиентов.

Построение модели

Необходимо реализовать отличие *в* обслуживании клиентов, которые только стригутся, и клиентов, которые стригутся и бреются.

Такую систему можно промоделировать с помощью двух сегментов. Один из них моделирует обслуживание только стригущихся клиентов, **А** второй – стригущихся и бреющихся. В каждом из сегментов пара **QUEUE-DEPART** должна описывать одну и ту же очередь. Таким же образом пара блоков **SEIZE-RELEASE** должна описывать в каждом из двух сегментов одно и то же устройство и моделировать работу парикмахера.

Таблица 4.13 (Таблица определений)

Элементы GPSS	***Интерпретация***
Транзакты	
В 1 сегменте модели	Клиенты, которые только стригутся
Во 2 сегменте модели	Клиенты, которые бреются и стригутся
В 3 сегменте модели	Таймер
Устройство	
BARBER	Парикмахер
Очередь	
BARBERQ	Очередь, используемая для сбора статистики об ожидании клиентов обоих типов

Единица модельного времени – 1 мин.

Программа:

```
HAIRCUT
GENERATE    35,10       ; Приход клиентов, которые только стригутся
QUEUE       BARBERQ     ; Присоединение к очереди
SEIZE       BARBER      ; Переход в кресло парикмахера
DEPART      BARBERQ     ; Выход из очереди
ADVANCE     18,6        ; Стрижка у парикмахера
RELEASE     BARBER      ; Освобождение парикмахера
TERMINATE   0           ; Уход из парикмахерской
HAIRCUT AND SHAVING
GENERATE    60,20       ; Приход клиентов, которые стригутся и бреются
QUEUE       BARBERQ     ; Присоединение к очереди
SEIZE       BARBER      ; Переход в кресло парикмахера
DEPART      BARBERQ     ; Выход из очереди
ADVANCE     10,2        ; Бритье у парикмахера
ADVANCE     18,6        ; Стрижка у парикмахера
RELEASE     BARBER      ; Освобождение парикмахера
TERMINATE   0           ; Уход из парикмахерской
TIMER – Сегмент таймера
GENERATE    480         ; Транзакт-таймер приходит в момент 480
TERMINATE   1           ; Завершение прогона
START       1
```

Пример 4.13 [10]

На фабрике в кладовой работает один кладовщик. Он выдает запасные части механикам, обслуживающим станки и устанавливающим эти части на испорченных станках. Запасные части довольно дорогие и, кроме того, их ассортимент слишком велик для того, чтобы каждый механик мог иметь все запасные части в своем ящике. Время, необходимое для удовлетворения запроса, зависит от типа запасной части. Запросы бывают двух категорий. Соответствующие данные приведены в табл. 4.14.

Таблица 4.14

Категория запроса	***Интервал времени прихода механиков, с***	***Время обслуживания, с***
1	420±360	300±90
2	360±240	100±30

Порядок обслуживания механиков кладовщиком такой: запросы первой категории обслуживаются только в том случае, когда в очереди нет ни одного запроса второй категории. Внутри одной категории дисциплина обслуживания – «первым пришел – первым обслужился». Необходимо создать модель работы кладовой, моделирование выполнять в течение восьмичасового рабочего дня.

Дисциплина обслуживания «первый пришел – первый обслужился» для двух категорий запросов с соответствующим приоритетом изображена на рис. 4.1.

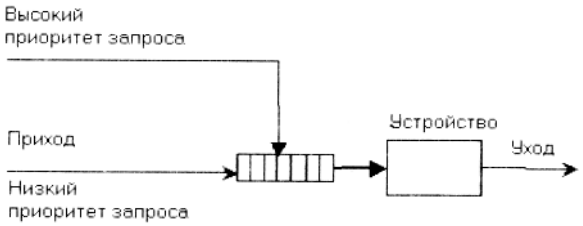


Рис. 4.1

Построение модели

Эта система очень похожа на систему из примера 4.12. Есть два различных типа заявок, поступающих на обслуживание к одному устройству. Различаются распределения интервалов приходов и времени обслуживания для этих типов заявок. Трудность заключается в том, что один из типов имеет преимущество в обслуживании. Построить модель можно, используя тот же подход, что и в примере 4.12. Но при этом необходимо использовать средство задания заявкам различных приоритетов. Итак, запросы первой категории будем моделировать одним сегментом, **A** запросы второй категории –

другим. Разные относительные приоритеты задаются путем использования для операнда **E** блока **GENERATE** запросов второй категории большего значения, чем для запросов первой категории.

Таблица 4.15 (Таблица определений)

Элементы GPSS	***Интерпретация***
Транзакты	
В первом сегменте модели	Механики, делающие запрос первой категории
Во втором сегменте модели	Механики, делающие запрос второй категории
В третьем сегменте модели	Таймер
Устройство	
STOCKMAN	Кладовщик
Очереди	
QSTCKM1 **QSTCKM2**	Очереди, используемые для сбора статистики об ожидании механиков различных категорий

Единица модельного времени – 1 с.

Программа:

```
MECHANICS OF TYPE 1
GENERATE    420,360,,, 1    ; Приход механиков 1-й категории
QUEUE       QSTCKM1         ; Присоединение к очереди 1-й кат.
SEIZE       STOCKMAN        ; Занятие кладовщика
DEPART      QSTCKM1         ; Выход из очереди
ADVANCE     300,90          ; Обслуживание кладовщиком
RELEASE     STOCKMAN        ; Освобождение кладовщика
TERMINATE   0               ; Уход из кладовой
MECHANICS OF TYPE 2
GENERATE    360,240,,,2     ; Приход механиков 2-й категории
QUEUE       QSTCKM2         ; Присоединение к очереди 2-й кат.
SEIZE       STOCKMAN        ; Занятие кладовщика
DEPART      QSTCKM2         ; Выход из очереди
ADVANCE     100,30          ; Обслуживание кладовщиком
RELEASE     STOCKMAN        ; Освобождение кладовщика
TERMINATE   0               ; Уход из кладовой
TIMER – Сегмент таймера
GENERATE    28800           ; Приход транзакта-таймера
TERMINATE   1               ; Завершение прогона
START       1
```

Пример 4.14 [10]

Изготовление деталей определенного вида включает длительный процесс сборки, который заканчивается коротким периодом обжига в печи. Поскольку эксплуатация печи обходится очень дорого, несколько сборщиков используют одну печь, в которой одновременно можно обжигать только одну деталь. Сборщик не может начать новую сборку, пока не вытащит из печи предыдущую деталь.

Таким образом, сборщик работает в таком режиме:

1) собирает следующую деталь;

2) ожидает возможности использования печи по принципу FIFO;

3) использует печь;

4) возвращается к п. 1.

Время, необходимое на выполнение различных операций, приведено в табл. 4.16.

Таблица 4.16

Операция	***Необходимое время, мин***
Сборка	30±5
Обжиг	8±2

Необходимо построить на GPSS модель описанного процесса. Определить оптимальное число сборщиков, использующих одну печь, т.е. такое количество, которое дает наибольшую прибыль при

моделировании в течение 40 часов модельного времени. Предполагается, что в течение рабочего дня нет перерывов, **А** рабочими днями являются все дни (без выходных).

Построение модели

В данном случае есть два ограничивающих условия, влияющих на пропускную способность системы: одна печь и некоторое фиксированное количество сборщиков, работающих в системе.

Естественно для моделирования печи использовать понятие «устройство», также естественно отождествлять сборщиков с транзактами. Тогда можно считать, что сборщики «циркулируют» в системе, периодически осуществляя сборку и обжиг. Аналогично и транзакты должны циркулировать в GPSS-модели системы. Как видно из постановки задачи, модель представляет собой замкнутую СМО с одним устройством обслуживания.

В реальной системе, после того, как сборщик вынимает из печи обожженную деталь, он возвращается и начинает новый этап сборки. В модели после того, как транзакт завершает использование устройства, он должен быть перемещен с помощью блока **TRANSFER** в блок следующей сборки. Для ограничения общего количества транзактов, циркулирующих в модели, необходимо использовать операнд **D** блока **GENERATE.**

Для вычисления прибыли при заданном числе сборщиков необходимо знать, сколько готовых деталей они сделают на протяжении моделируемого периода. Это значение дает количество выходов из печи, т.е. в процессе моделирования нас интересует именно эта величина.

Таблица 4.17 (Таблица определений)

Элементы GPSS	***Интерпретация***
Транзакты	
В 1 сегменте модели	Сборщики
Во 2 сегменте модели	Таймер
Устройство	
OVEN	Печь

Единица модельного времени – 1 мин.

Программа:

```
        GENERATE    ,,,4          ; Определение количества сборщиков
BACK1   ADVANCE                30,5   ; Сборка следующей детали
        SEIZE       OVEN          ; Занятие печи
        ADVANCE                8,2    ; Использование печи
        RELEASE                OVEN   ; Освобождение печи
        TRANSFER    ,BACK1        ; Переход к сборке следующей детали
        GENERATE    2400          ; Моделирование пяти дней роботы
        TERMINATE   1             ; Завершение работы
        START       1             ; Начало прогона модели
```

Пример 4.15 [10]

Морские судна двух типов прибывают в порт, где происходит их разгрузка. В порту есть два буксира, обеспечивающих ввод и вывод кораблей из порта. К первому типу судов относятся корабли малого тоннажа, которые требуют использования одного буксира. Корабли второго типа имеют большие размеры, и для их ввода и вывода из порта требуется два буксира. Из-за различия размеров двух типов кораблей необходимы и причалы различного размера. Кроме того, корабли имеют различное время погрузки-разгрузки. Исходные данные приведены в табл. 4.18.

Построить модель системы, в которой можно оценить время ожидания кораблями каждого типа входа в порт. (Время ожидания входа в порт включает время ожидания освобождения причала и буксира). Корабль, ожидающий освобождения причала, не обслуживается буксиром до тех пор, пока не будет предоставлен нужный причал. Корабль второго типа не займет буксир до тех пор, пока ему не будут доступны оба буксира.

Таблица 4.18

Значение	***Тип корабля***

	1	2
Интервал прибытия, мин	130 ± 30	390 ± 60
Время входа в порт, мин	30 ± 7	45 ± 12
Количество доступных причалов	6	3
Время погрузки-разгрузки, час	12 ± 2	18 ± 4
Время выхода из порта, мин	20 ± 5	35 ± 10

Программа:

```
PRCH1       STORAGE     6           ; 6 причалов для кораблей первого типа
PRCH2       STORAGE     3           ; 3причала для кораблей второго типа
BUKS        STORAGE     2           ; 2 буксира
;       SHIPS OF TYPE 1
            GENERATE    130,30      ; Подход к порту
            QUEUE       TYPE1
            ENTER       PRCH1       ; Получение причала
            ENTER       BUKS        ; Получение буксира
            DEPART      TYPE1
            ADVANCE     30,7        ; Буксирование до причала
            LEAVE       BUKS        ; Освобождение буксира
            ADVANCE     720,120     ; Погрузка-разгрузка
            ENTER       BUKS        ; Получение буксира
            LEAVE       PRCH1       ; Освобождение причал
            ADVANCE     20,5        ; Буксирование (отчаливание)
            LEAVE       BUKS        ; Освобождение буксира
            TERMINATE
        SHIPS OF TYPE 2
            GENERATE    390,60
            QUEUE       TYPE2
            ENTER       PRCH2
            ENTER       BUKS,2
            DEPART      TYPE2
            ADVANCE     45,12
            LEAVE       BUKS,2
    ADVANCE     1080,240
    ENTER       BUKS,2
    LEAVE       PRCH2
    ADVANCE     35,10
    LEAVE       BUKS,2
    TERMINATE   0

    GENERATE    48000
    TERMINATE   1
    START       1
```

Среднее время ожидания кораблями каждого типа входа в порт получаем в конце моделирования из стандартной статистики об очередях: оно равно показателю **AVERAGE TIME** соответствующей очереди. (Эти же значения дают СЧА **QT$TYPE1 и QT$TYPE2).**

4.12. Переменные

Общая характеристика переменных. При построении модели системы, иногда возникает необходимость задать сложные математические или логические соотношения между атрибутами системы. Для этой цели в программе используются ***переменные.***

В GPSS имеется три типа переменных:

1) арифметические переменные;
2) арифметические переменные с «плавающей точкой»;
3) булевы переменные.

Значение арифметических переменных может использоваться как:

1) операнд блока; в этом случае значение арифметической переменной может представлять собой:
– номер объекта (устройства, МКУ, очереди и т. п.);
– номер параметра транзакта;
– значение стандартного числового атрибута;
3) операнд **A** функции;
4) операнд **A** таблицы;
5) операнд выражения другой переменной.

В ***выражениях*** арифметические переменные используют такие арифметические операции:

+ алгебраическое сложение;
– алгебраическое вычитание;
алгебраическое умножение;
/ алгебраическое деление (результатом операции является целая часть частного);
@ деление по модулю;
^ возведение в степень;
\ деление без остатка (перед делением у обоих операндов отбрасываются дробные части, результатом операции есть целая часть частного).

Привычно используемый для умножения во многих языках знак «*», зарезервирован в GPSS для обозначения ***косвенной адресации,*** однако в GPSS World есть возможность в меню настройки параметров переопределить для умножения знак «*», **A** для косвенной адресации – «#». Косвенная адресация является мощным средством для построения компактных и гибких моделей. Ее идея заключается в том, что можно обратиться к любому объекту или СЧА через параметры транзактов. Доступ же к параметрам транзактов осуществляется через СЧА **P*j,*** где *j* – номер параметра транзакта (например, 10) или **P$имя,** где *имя* – имя (идентификатор) параметра транзакта. Так как обращение к объекту возможно только через параметр транзакта, то символ **P** может опускаться. Например, выражение Q*7 или Q*P7 определяет текущее значение длины очереди, номер которой задан в параметре 7 транзакта. Если в седьмом параметре хранится значение 3, то это будет текущее значение длины очереди с номером 3.

В выражениях может быть задано любое число приведенных операций в различных комбинациях. Знак результата вычисляется по обычным алгебраическим правилам. Допускаются отрицательные значения переменных. Выражения анализируются слева направо. Возведение в степень, умножение, деление и деление по модулю выполняются раньше, чем сложение и вычитание.

Вычисленное значение переменной является ее стандартным числовым атрибутом.

Арифметические переменные. Арифметические переменные аналогичны арифметическим выражениям в алгоритмических языках. Переменная задается оператором **VARIABLE,** называемым оператором описания переменной, который содержит арифметическое выражение. Формат оператора описания переменной:

Таблица 4.19

Поле	*Информация, задаваемая в поле*
Метка	\ Имя (числовое или символьное) переменной
Операция	] VARIABLE
Операнд **A**	Выражение, которое используется для вычисления значения переменной

При обращении к переменной используется обозначение **V<номер переменной>** или **V$<имя переменной>,** т.е. **V –** это СЧА переменной.

Пример 4.16

Оператор описания **VARIABLE** определяет арифметическую переменную **RSL**:

```
RSL    VARIABLE    QT$WAITL+3-FN$DSTRB#P7
```

При любом обращении к переменной **RSL** (употребляется обозначение **V$RSL)** ее значение вычисляется как текущая длина очереди **WAITL (QT$WAITL** – СЧА регистратора очереди) плюс константа 3 и минус произведение значения функции **DSTRB** на значение параметра 7 транзакта, обрабатываемого в данный момент. В приведенном выражении **FN** – СЧА для обращения к функции, а **P** – СЧА транзакта.

Перед выполнением любой арифметической операции определяется значение каждого элемента и выделяется его целая часть. Постоянные без знака считаются положительными числами.

В выражении арифметической переменной могут быть использованы любые СЧА, функции и другие арифметические переменные. Запрещается использование самой вычисляемой переменной, **A** также переменных со знаком, так как знаки в данном случае рассматриваются как арифметические операции.

Система моделирования GPSS допускает использование скобок в выражениях арифметических переменных (для группировки членов или для обозначения операции умножения).

В GPSS World выражения, записанные в круглых скобках, обрабатываются вычислительной процедурой встроенного алгоритмического языка PLUS. Поэтому их можно использовать в качестве операндов блоков и операторов языка GPSS. Например, выражение, описанное в примере 4.16, может быть использовано таким образом:

```
ADVANCE     (QT$WAITL+3-FN$DSTRB#P7)
```

1.В GPSS/PC выражение может содержать не больше пяти пар скобок (не считая скобок, используемых при описании элементов матриц).

2. Пробелы между символами в выражениях не допускаются. Левый пробел записи считается концом выражения. Для записи выражения, превышающего длину строки, можно ввести другой оператор **VARIABLE** с именем, отличным от имени первой переменной, и включить значение новой переменной в качестве одного из операндов в выражение первой арифметической переменной.

Пример 4.17

```
ADD     VARIABLE     P10+25
```

При обращении к арифметической переменной **ADD** ее значение вычисляется как сумма значений десятого параметра транзакта, обрабатываемого в данный момент, и константы 25.

```
F1          VARIABLE    Q9+3#V$F2-V$F3#FN$IO
F2          VARIABLE    9+R13-FN19#Q10
F3          VARIABLE    FN$TYPE1+S$PL#RSRC-Q$ENTRY
```

Выражение для **F1** содержит как операнды переменные **F2** и **F3**.

```
PROFIT      VARIABLE    V*P*FN*P2
```

Переменная **PROFIT** будет вычислена следующим образом. Вначале определяется значение второго параметра текущего транзакта. Пусть в параметре **P2** хранится значение 3. Затем вычисляется значение функции 3. Пусть оно равно 10. После этого определяется значение параметра с номером 10. Пусть содержимым этого параметра будет число 5. Тогда переменной **PROFIT** присваивается значение переменной 5.

```
COMP        VARIABLE    X*P4/100
```

При обращении к арифметической переменной **COMP** ее значение вычисляется как частное от деления значения сохраняемой величины (СЧА X), номер которой определяется четвертым параметром транзакта, обрабатываемого в данный момент, на константу 100. Сохраняемые величины позволяют хранить значения глобальных переменных, доступных из любой части модели. Такой прием используется для выделения старших разрядов чисел. Например, пусть в четвертом параметре текущего транзакта записано число 6. И пусть в сохраняемой величине 6 хранится число 12345. При обращении к переменной **COMP** происходит следующее:

1) определение значения параметра 4 (получаем число 6);

2)определение значения 12345, которое хранится в ячейке **X*P4,** т.е. в ячейке 6;

3)значение 12345 делится на 100 (с отбрасыванием остатка), в результате получается 123.

Арифметические переменные с плавающей точкой аналогичны рассмотренным арифметическим переменным, за исключением того, что все операции над операндами выражений переменных с плавающей точкой выполняются без преобразования операндов и промежуточных результатов в целые значения. Лишь окончательный результат вычисления преобразуется в целое число.

Формат операторов описания арифметических переменных с плавающей точкой идентичен рассмотренному выше формату операндов описания арифметических переменных за исключением того, что в поле операции записывается слово **FVARIABLE.** Правила написания операторов те же, что и для арифметических переменных. Арифметическая переменная и переменная с плавающей точкой не могут

иметь одинаковые номера. Если они имеют одинаковые номера, то при вычислении используется более позднее из двух описаний.

Различие результатов, полученных при вычислении с плавающей точкой и фиксированной, можно увидеть из такого примера:

```
FLOAT      FVARIABLE   10#(ll/3)
FIXED      VARIABLE    10#(ll/3)
```

Значение переменной **FLOAT** равно 36, так как константа 10 умножается на 3,67 и от результата 36,7 взята целая часть. Переменная **FIXED** равна 30, так как результат промежуточной операции деления будет округлен до 3.

1. Для переменных с плавающей точкой не допускается операция деления по модулю.
2. Использование дробных констант допускается только при описании переменных с плавающей точкой.

3.Стандартный числовой атрибут **V$<имя переменной>** используется для обращения к значениям как арифметических переменных, так и переменных с плавающей точкой. Способ вычисления переменной определяется оператором описания этой переменной.

Булевы переменные. Булевы переменные позволяют принимать решения в зависимости от значений СЧА и состояния объектов GPSS, используя для этого только одно выражение.

Булевы переменные – это логические выражения, состоящие из различных СЧА и (или) других булевых переменных. В булевой переменной проверяется одно или несколько логических условий. Результатом проверки есть единица ***(истина),*** если условия выполняются, и ноль ***(ложь)*** – в противном случае.

При описании булевых переменных используются три типа операторов: логические, булевы и операторы отношений.

Логические операторы связаны с такими ресурсами, как устройства, МКУ и логические ключи. Они используются для определения состояния данных объектов. Логические операторы, используемые в GPSS, представлены в табл. 4.20

Таблица 4.20

Логические операторы	***Значение оператора, отражающее состояние ресурса***
FVj или Fj	Равно 1, если устройство *j* занято или обслуживает прерывание, в противном случае – 0
FNVj	Равно 1, если устройство *j* не занято и не обслуживает прерывание, в противном случае – 0
Ij	Равно 1, если устройство *j* обслуживает прерывание, в противном случае – 0
NIj	Равно 1, если устройство *j* не обслуживает прерывание, иначе – 0
NUj	Равно 1, если устройство *j* не используется, в противном случае – 0
UJ	Равно 1, если устройство *j* используется, в противном случае – 0
SFJ	Равно 1, если многоканальное устройство у заполнено, иначе – 0
SNFj	Равно 1, если МКУ *j* не заполнено, иначе – 0
SEj	Равно 1, если МКУ *j* пусто, иначе – 0
SNEj	Равно 1, если МКУ *j* не пусто, иначе – 0
SVj	Равно 1, если МКУ *j* находится в состоянии использования, в противном случае – 0
SNVJ	Равно 1, если МКУ *j* не используется, в противном случае – 0
LRj	Равно 1, если логический ключ *j* выключен, иначе – 0

LSj	Равно 1, если логический ключ *j* включен, иначе – 0

Операторы отношения выполняют алгебраическое сравнение операндов. Операндами могут быть константы или стандартные числовые атрибуты. Все операторы отношений записываются в кавычках:

"G" (**G**reater) – больше;

"L" (**L**ess) – меньше;

"E" (**E**qual) – равно;

"NE" (**N**ot **E**qual) – не равно;

"LE" (**L**ess than or **E**qual) – меньше или равно;

"GE" (**G**reater than or **E**qual) – больше или равно;

Есть два ***булевых оператора:*** **"OR" –** оператор «или», и "**AND**" – оператор «и». Оператор «или» проверяет, выполняется ли хотя бы одно из проверяемых условий. Оператор «и» требует выполнения обоих условий.

4.13. Определение функции в GPSS

В GPSS рассматриваются пять типов функций:

1) дискретная числовая (D),

2) непрерывная числовая (C),

3) табличная числовая (L),

4) дискретная атрибутивная (E),

5) табличная атрибутивная (M). Рассмотрим два первых типа функций.

Дискретная функция представляет собой кусочно-постоянную функцию, которая состоит из горизонтальных ступеней (рис. 4.2). ***Непрерывная функция*** представляет собой кусочно-непрерывную функцию. Непрерывная функция в GPSS состоит из соединенных между собой прямых отрезков и представляет собой ломаную линию (рис. 4.3). Чтобы задать дискретную функцию, необходимо задать координаты крайних правых точек горизонтальных отрезков. Для непрерывной функции необходимо задать координаты всех точек, которые являются концами отрезков.

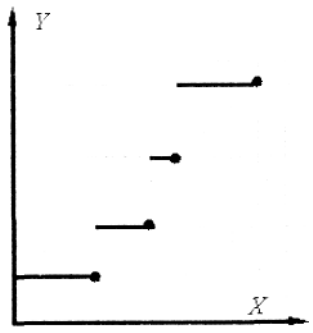


Рис. 4.2

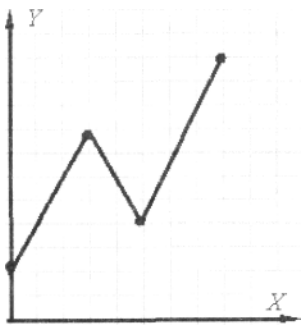


Рис. 4.3

Действия, необходимые для определения дискретной и непрерывной GPSS-функции:

1.Присвоить функции имя. Имя может быть числовым либо символьным.

2. Задать аргумент функции. Аргументом могут быть:

1) ссылка на генератор случайных чисел, используемый для розыгрыша в соответствии с распределением, заданным функцией;

2) стандартный числовой атрибут;

3) ссылка на любую другую функцию.

В первом случае аргумент задается в виде **RNj,** *j* **–** целое число (номер генератора). В GPSS/PC *j* = 1, ..., 7, т.е. возможно обращение к семи идентичным генераторам случайных чисел. При этом генераторы выдают случайные числа в диапазоне 0... 0,999. В GPSS World количество генераторов случайных чисел неограниченно, **A** выдаваемые ими значения 0... 0,999999.

3. Задать тип функции и число крайних точек функции.

4. Задать значения аргумента (переменной) и соответствующие значения функции (т.е. координаты крайних точек функции).

Три первых элемента информации указываются в ***операторе определения функции.*** Формат оператора представлен в таблице.

Таблица 4.21

Поле	***Информация, задаваемая в поле***
Метка	Имя функции (числовое или символьное)
Операция	**FUNCTION**
Операнды **A** **B**	RNJ O = 1, ..., 7) или СЧА. Dn либо Cn, где D определяет дискретную функцию, C определяет непрерывную функцию; n – для дискретной функции – это число различных значений, получаемых функцией (количество горизонтальных отрезков), для непрерывной функции – это число, на единицу больше числа отрезков, составляющих функцию (количество точек)

За каждым оператором описания **FUNCTION** следуют операторы задания координат точек функции (значений аргументов X_i и соответствующих им значений функции Y_i) – это ***операторы описания координат функции.*** Их формат:

1) если координаты всех точек расположены в одной строке оператора описания функции:

X_1, Y_1/X_2, $Y_2/$ ... $/X_n$, Y_n

2) если координаты точек расположены в нескольких операторах описания функции:

X_1, Y_1/X_2, $Y_2/$ ... $/X_i$, Y_i

X_{i+1}, Y_{i+1}/X_{i+2}, $Y_{i+2}/$ ... $/X_k$, Y_k

... ...

X_{m+1}, Y_{m+1}/X_{m+2}, $Y_{m+2}/$ ... $/X_n$, Y_n

где X_i и Y_i – координаты *i*-й точки функции (в случае моделирования случайной величины X_i является *i*-й суммарной (кумулятивной) частотой, Y_i – соответствующим значением случайной величины).

Особенности оператора описания координат функции:

1) основной единицей информации оператора описания координат функции является пара значений X_i, Y_i (координаты точки *i*);

2) значения координат X_i и Y_i – одной точки функции разделяются запятой;

3) последовательные наборы координат разделяются знаком «/»;

4) координаты X_i и Y_i – относящиеся к одной точке, задаются одним оператором, т.е. пара координат одной точки не должна разрываться;

5) все строки описания координат функции должны начинаться с первой позиции;

6) во всех случаях значения аргумента должны удовлетворять следующим неравенствам:

$X_1<X_2<...<X_i<...<X_n$.

Значение функции является ее стандартным числовым атрибутом. Способ ссылки на этот атрибут зависит от того, как задано имя функции: в символьном или числовом виде. Если имя числовое, то к значению функции обращаемся через FNj (где j – номер функции), если имя символьное – через FN$<имя функции>.

1. Аргументом функции может быть и значение какой-либо другой функции.
2. Каждая функция должна иметь, по крайней мере, две описанные точки.

Пример 4.18

Пусть необходимо смоделировать дискретную случайную переменную, заданную в табл. 4.22.

Таблица 4.22

Значение случайной переменной	***Относительная частота***	***Суммарная частота***	***Диапазон***	***Интервал***
2	0,15	0,15	[0,0-0,15]	1
5	0,20	0,35	(0,15-0,35]	2
8	0,25	0,60	(0,35 – 0,60]	3
9	0,22	0,82	(0,60 – 0,82]	4
12	0,18	1,00	(0,82 – 1,0]	5

GPSS-функцию можно определить таким образом:

```
SERV    FUNCTION    RN4,D5
.15,2/.35,5/.6,8/.82,9/1,12
```

Графическая интерпретация функции показана на рис. 4.4.

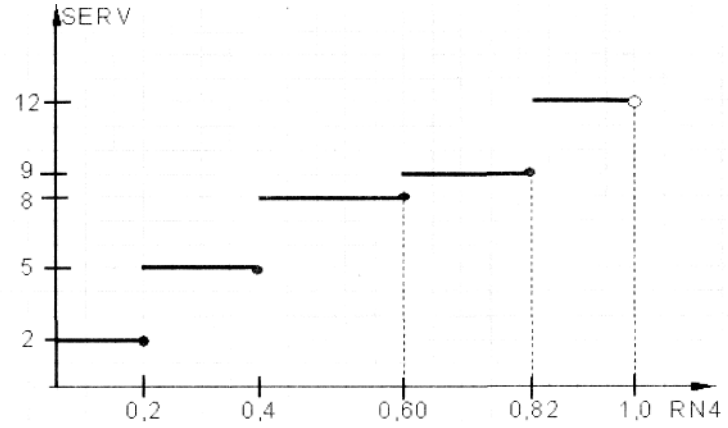


Рис. 4.4

Особенности вычисления дискретных и непрерывных GPSS-функций:

1. В начальной фазе выполняемые действия при вычислении дискретной и непрерывной функции одинаковы. При обращении к функции определяется значение ее аргумента. Потом просматривается упорядоченный ряд значений $X_1<X_2<...<X_i<...<X_n$ для определения интервала, в который попало значение аргумента (пусть это будет интервал между точками $i-1$ и i).

2. Если функция дискретная, то второй элемент соответствующей пары X_i, Y_j является значением функции. Если функция непрерывная, выполняется линейная интерполяция для пары точек $i-1$ и i, находящихся на краях интервала значений функции, на который указало значение аргумента. ***Целая часть*** результата интерполяции и является значением функции.

3. Если значение аргумента функции больше значения координаты X_n последней точки, то в обоих случаях (дискретной и непрерывной функции) значениями функции являются значения Y_n.

Пример 4.19

Моделирование случайной переменной, равномерно распределенной на интервале [2,5].

Эта случайная переменная может быть смоделирована функцией:

INN FUNCTION RN2,C2
0,2/1,6

Графическая интерпретация непрерывной функции показана на рис. 4.5.

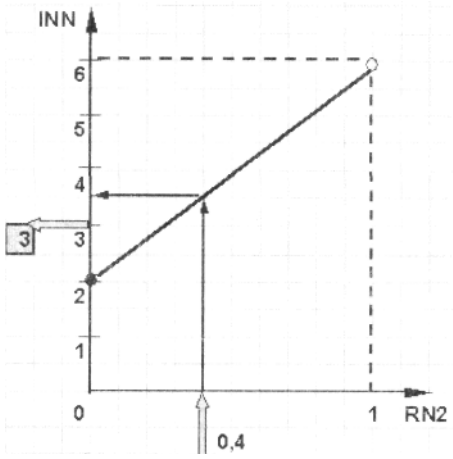


Рис. 4.5

Так как максимальное значение, которое может выдать генератор случайных чисел, равно 0,999, то фактические значения интервалов времени распределены равномерно на интервале [2, 5] и равны, соответственно, одному из значений: 2, 3, 4, 5. Если генератор выдаст число 0,999, функция, которая показана на рис. 4.5, примет значение 5,996, целая часть которого равна 5 (это и будет значением GPSS-функции **INN**). В случае, если генератор случайных чисел RN2 выдаст значение 0,4, GPSS-функция **INN** примет значение 3 (см, рис. 4.5).

GPSS-функция **INN** не может принять значение, равное 6 (несмотря на то, что второй элемент второй пары оператора описания координат функции 0,2/1,6 равен 6).

Точные граничные значения **RN2,** соответствующие возможным значениям функции **INN**, представлены в табл. 4.23.

Таблица 4.23

Целая часть значения функции	***Диапазон значений RN2***
2	[0,0 – 0,25]
3	(0,25 – 0,50]
4	(0,50 – 0,75]
5	(0,75 – 0,999]

Равномерное распределение [2, 3, 4, 5] не может быть задано непосредственно с помощью операндов **A** и **B** блока **GENERATE.** Здесь имеем четыре возможных значения, тогда как интервал **A ± B** (**A** и **B** целые) всегда имеет нечетное число элементов.

Пример 4.20

Часто возникают ситуации, когда в процессе моделирования необходимо переходить в различные блоки программы в зависимости от логики работы модели. Стандартные блоки GPSS WORLD такие, как **TEST** (см. параграф 4.16) и **TRANSFER,** не всегда могут решить эту проблему, так как они позволяют распределять транзакты максимум по двум направлениям. В случае, когда осуществляется условный переход на одну из нескольких меток (если более двух, то в обычных языках программирования используется оператор **CASE OF**), необходимо построить переключающую функцию. Для вызова переключающей функции используется блок **TRANSFER** в режиме безусловного перехода.

Пример переключающей функции:

```
PEREKL      FUNCTION    RN4,D5
0.2,LB1/0.4,LB2/0.6,LB3/0.8,LB4/1,LB5

            GENERATE    ,,,100
            TRANSFER    ,FN$PEREKL
LB1         QUEUE       STOR1
            TERMINATE   0
LB2         QUEUE       STOR2
            TERMINATE   0
LB3         QUEUE       STOR3
            TERMINATE   0
LB4         QUEUE       STOR4
            TERMINATE   0
LB5         QUEUE       STOR5
TERMINATE   0

GENERATE    1
TERMINATE   1
START       1
```

Переключающая функция построена на основе дискретной функции с тем отличием, что результатом функции являются метки. В этом примере переход осуществляется к одной из меток в зависимости от числа, которое получаем от генератора случайных чисел.

Моделирование неравномерных случайных величин. Использование функций в блоках GENERATE и ADVANCE. Пусть распределение интервалов поступления через определенный блок **GENERATE** или время задержки в некотором блоке **ADVANCE** ***не является равномерным*** (либо является равномерным с «плавающими во времени», т.е. нефиксированными значениями среднего и половины поля допуска). Для входов транзактов в модель через этот блок **GENERATE** и для задания закона времени задержки в соответствующем блоке **ADVANCE** необходимо использовать функции и (или) СЧА. Использование функций, заданных в операндах блоков, зависит от контекста. От значения функции берется целая часть, за исключением тех случаев, когда это значение используется в качестве операнда В блоков **GENERATE и ADVANCE** или операнда **С** блока **ASSIGN.** В табл. 4.24 показаны различные варианты использования функций и СЧА в качестве операндов **А** и **В** блоков **GENERATE** и **ADVANCE**. Под результатом понимается значение интервала поступления или задержки.

Таблица 4.24

Операнд А	***Операнд В***	***Результат***
α (число или СЧА)	β (число или СЧА)	Генерируется случайное число, равномерно распределенное на интервале α ± β. Результат равен полученному числу
FN$DIS	Отсутствует	Результат равен значению функции DIS
Отсутствует	FN$B	Данная комбинация недопустима
FN$DIS	β (число или СЧА)	Вначале вычисляется значение функции DIS. Берется целая часть этого значения (пусть это будет число α), после чего генерируется случайное число, равномерно распределенное на интервале α ± β. Результат равен полученному числу
α (число или СЧА)	FN$DIS	Вначале вычисляется значение функции DIS (пусть это будет число (3), после чего находится произведение α × β. Результат равен целой части этого произведения
FN$DIS1	FN$DIS2	Вычисляются значение функций DISl и DIS2 (пусть это будет числа α и β), после чего находится произведение α × β. Результат равен целой части этого произведения

Пример 4.21

```
TIME  FUNCTION    RN3,D5
.05,5/.15,6/.75,8/.93,9/1,10
```

К этой функции можно обратиться таким образом:

```
GENERATE    FNSTIME
...
ADVANCE     FNSTIME
```

Пример 4.22

Пусть в моделируемой системе время обслуживания некоторым устройством распределено равномерно на интервале A ± 2, где среднее время обслуживания A с вероятностью 0,4 принимает значение 5, а с вероятностью 0,6 – значение 7. Эту ситуацию можно смоделировать следующим образом.

Определим функцию **AVERAGET:**

```
AVERAGE_T  FUNCTION    RN1,D2
.4,5/1,7
```

Используемее в блоке **ADVANCE:**

```
ADVANCE      FN$AVERAGE_T,2
```

Выполнение подпрограммы блока **ADVANCE** включает расчет функции **AVERAGE_T.** Это, в свою очередь, требует обращения к генератору случайных чисел **RN1.** Пусть генератор выдал значение меньшее, чем 0,4. Тогда соответствующее значение функции **AVERAGE_T** равно 5. Таким образом, время задержки текущего транзакта в устройстве будет равномерно распределено на интервале 5±2.

Непрерывные случайные переменные, рассматриваемые как дискретные. Как известно, дискретные случайные переменные могут принимать только фиксированное число значений. В противоположность этому, непрерывные (в классическом смысле этого термина) случайные переменные могут иметь неограниченное число различных значений.

На практике обычно достаточно, чтобы все случайные переменные имели конечное число конкретных значений. Нет необходимости в тщательном определении значений этих случайных переменных, за исключением случаев, когда необходимо делать расчеты с высокой степенью точности. Таким образом, вполне возможна дискретизация непрерывных распределений. После этого они могут быть определены в GPSS с помощью дискретных и непрерывных GPSS-функций (непрерывные GPSS-функции по сути также являются дискретными, поскольку множество их значений дискретно и конечно).

Функции распределения случайных величин. В языке GPSS возможность задания функций распределения случайных величин ограничена заданием их в табличном виде путем аппроксимации непрерывными функциями. Поэтому можно задать только те функции, которые легко преобразовать для новых значений параметров. К таким функциям, например, относится функция экспоненциального распределения с параметром $\lambda = 1$, **A** также функция стандартного нормального распределения с математическим ожиданием $m = 0$ и стандартным отклонением $\sigma = 1$.

Эти ограничения не касаются языка GPSS World, в котором для задания различных вероятностных функций распределения можно использовать библиотечные процедуры, написанные на языке PLUS. Однако использование вероятностных распределений в табличном виде значительно ускоряет процесс моделирования.

Моделирование пуассоновского потока. Рассмотрим табличный способ задания пуассоновского потока заявок. Пуассоновский входящий поток описывается таким образом: вероятность поступления k заявок пуассоновского потока в течение интервала t составляет

$$p_k(t) = \frac{e^{-\lambda t}(\lambda t)^k}{k!}, \quad k = 1,2,\ldots, \qquad (4.1)$$

где λ – интенсивность потока.

Интервалы времени между соседними заявками пуассоновского потока распределены по ***экспоненциальному закону.*** Согласно ***методу обратной функции,*** можно получить ряд чисел, которые

имеют экспоненциальное распределение, если ряд случайных чисел *R,* равномерно распределенных на интервале [0,1], преобразовать в соответствии с функцией, обратной к экспоненциальной функции распределения:

$$t_j = F^{-1}(x) = -\overline{T}\ln(r_j), \qquad (4.2)$$

где t_j – *j*-й разыгранный интервал времени поступления; $\overline{T} = 1/\lambda$ – средний интервал времени поступления; r_j – *j-e* число в последовательности случайных чисел *R* с равномерным распределением на интервале [0, 1].

Разработчиками GPSS была осуществлена аппроксимация функции F^{-1} (*x*), обратной к экспоненциальной функции распределения с параметром λ = 1. Таким образом, функция F^{-1} *(x)* была заменена 23 отрезками, которые использовались для преобразования значений **RNj** в значение – ln(RNj).

Функция **XPDIS** определяет экспоненциальное распределение с интенсивностью λ = 1 :

```
XPDIS FUNCTION     RN1,C24        ; exponential distribution function
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915/.7,1.2/
.75,1.38/.8,1.6/.84,1.83/.88,2.12/.9,2.3/.92,2.52/.94,2.81/
.95,2.99/.96,3.2/.97,3.5/.98,3.9/.99,4.6/.995,5.3/.998,6.2/.999,7/.9998,8
```

Пуассоновский входящий поток с интенсивностью λ, отличной от единицы, моделируется с помощью блока **GENERATE** таким образом:

1) в качестве операнда **A** используют среднее значение интервалов времени $T = 1/\lambda$, где λ – интенсивность пуассоновского потока;

2) в качестве операнда **B** используют СЧА – значение функции **XPDIS,** операторы определения и описания которой приведены выше.

Пример 4.23

Пусть среднее значение интервалов поступления *T* в пуассоновском потоке требований равно 2 ч, **A** единица времени в модели равна 1 мин, тогда поступление заявок моделируется блоком:

```
GENERATE     120,FN$XPDIS
```

Если необходимо моделировать задержку, распределенную no экспоненциальному закону со средним значением времени 345, то для этого используется блок:

```
ADVANCE      345,FN$XPDIS
```

Свойство ***ординарности*** пуассоновского потока гласит: вероятность поступления двух или более заявок в течение малого временного интервала равна нулю.

Пусть пуассоновский поток моделируется блоком

```
GENERATE     5,FN$XPDIS
```

Если в результате обращения к функции XPDIS полученное значение меньше, чем 1/5, то целая часть произведения числа 5 и значения функции XPDIS равна нулю. Отсюда следует нарушение свойства ординарности. Во избежание этого рекомендуется, чтобы операнд **A** в блоке **GENERATE** был ***больше 50.*** Это легко достигается путем варьирования значения единицы модельного времени.

Моделирование гипер– и гипоэкспоненциального распределений. Экспоненциальную функцию распределения можно использовать также для моделирования гипер – и гипоэкспоненциального распределений.

Неэкспоненциальное распределение с ***коэффициентом вариации**** *C* > 1 можно получить с помощью взвешенной суммы экспонент – ***гиперэкспоненциального распределения:***

$$F_t(x) = P(t < x) = \sum_{i=1}^{k} \omega_i \left(1 - e^{-\mu_i x}\right),$$

$$\mu_i > 0, \quad \omega_i > 0, \quad \sum_{i=1}^{k} \omega_i = 1; \qquad (4.3)$$

* Коэффициент вариации C – это отношение стандартного отклонения к математическому ожиданию случайной величины.

$$\bar{t} = \sum_{i=1}^{k} \frac{\omega_i}{\mu_i}; \qquad (4.4)$$

$$D(t) = \sum_{i=1}^{k} \frac{2\omega_i}{\mu_i^2} - \left(\sum_{i=1}^{k} \frac{\omega_i}{\mu_i} \right)^2; \qquad (4.5)$$

$$C = \frac{\sqrt{D(t)}}{\bar{t}} \geq 1. \qquad (4.6)$$

Если $\mu_i = \mu$ для всех i, то $C = 1$ – имеем экспоненциальное распределение.

Гиперэкспоненциальное распределение можно получить при параллельном соединении k (рис. 4.6) экспоненциальных обслуживающих устройств с интенсивностью обслуживания μ_i и вероятностью ω_i использования для обслуживания $(i = \overline{1,k})$. Причем в произвольный момент времени может быть занято не более одного устройства из k. Такое распределение хорошо описывает распределение времени работы центрального процессора компьютера.

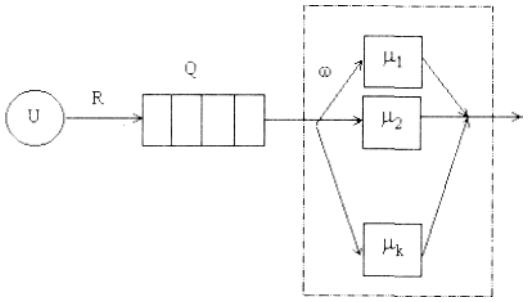


Рис. 4.6

Для моделирования гиперэкспоненциального распределения со средним значением 6,28 и стандартным отклонением 8,4 необходимо определить переменную

HYP FVARIABLE (410+(RN2'L'234)(#(1334-410)))#FN$XPDIS

Эту переменную можно использовать в блоке задержки так:

ADVANCE V$HYP

Гипоэкспоненциальное распределение с коэффициентом вариации $C<1$ описывается таким образом:

$$F_t(x) = P(t \leq x) = 1 - \frac{\mu_2}{\mu_2 - \mu_1} e^{-\mu_1 x} - \frac{\mu_1}{\mu_1 - \mu_2} e^{-\mu 2 x}, \quad (\mu_1 \neq \mu_2); \qquad (4.7)$$

$$\bar{t} = \frac{1}{\mu_1} + \frac{1}{\mu_2}; \qquad (4.8)$$

$$C = \sqrt{1 - \frac{2\mu_1\mu_2}{(\mu_1 + \mu_2)^2}} < 1; \qquad (4.9)$$

$$D(t) = \frac{1}{\mu_1^2 + \mu_2^2}. \qquad (4.10)$$

При равенстве всех коэффициентов μ распределение времени пребывания в обслуживающем центре (на рис. 4.7 обведен пунктирной линией) будет ***k-распределением Эрланга:***

$$F_t(x) = 1 - e^{-k\mu x} \sum_{i=0}^{k-1} \frac{(k\mu x)^2}{i!} \qquad (4.11)$$

Гипоэкспоненциальное распределение характерно, например, для времени обслуживания устройств ввода-вывода. Его можно получить последовательным соединением обслуживающих

экспоненциальных устройств, причем в любой момент времени должно быть занято не более одного устройства (рис. 4.7).

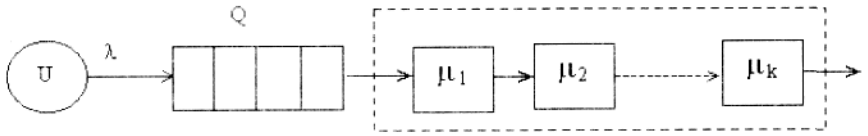


Рис. 4.7

Моделирование эрланговского потока. Экспоненциальное распределение не всегда адекватно описывает время обслуживания и поступления требований в систему. Более реалистичным является распределение Эрланга. В то же время, это распределение является частным случаем гамма-распределения, которое описано ниже. Для потока Эрланга k-го порядка с интенсивностью λ математическое ожидание и дисперсия определяются так: $E[x]=\frac{1}{k\lambda}, D[x]=\frac{1}{k\lambda^2}$. Для моделирования ***распределения Эрланга*** может также использоваться экспоненциальная функция распределения. Как было показано в главе 1, для этого достаточно просуммировать *k* случайных экспоненциально распределенных величин. С ростом *k* распределение Эрланга будет приближаться к нормальному распределению. Например, поток Эрланга второго порядка со средним значением времени поступления 180 можно задать таким образом:

```
        GENERATE    ,,,1
SDFG    ADVANCE     90,FN$EXPDIS
        ADVANCE     90,FN$EXPDIS
        SPLIT       1,SDFG
        ...
        TERMINATE   1
```

В нулевой момент времени в модель вводится транзакт. Этот транзакт в каждом их двух последующих блоков **ADVANCE** задерживается на экспоненциально распределенный промежуток времени. Блок **SPLIT** (подробнее см. в параграфе 4.19) создает копию транзакта и направляет ее на блок с меткой **SDFG,** исходный транзакт поступает в модель и т.д.

Пример 4.24

Для того, чтобы исследовать свойства распределения Эрланга можно воспользоваться следующей моделью:

```
EXPDIS      FUNCTION    RN1,C24
0,0/.100,.104/.200,.222/.300,.355/.400,.509
.500,.690/.600,.915/.700,1.200/.750,1.380
.800,1.600/.840,1.830/.880,2.120/.900,2.300
.920,2.520/.940,2.810/.950,2.990/.960,3.200
.970,3.500/.980,3.900/.990,4.600/.995,5.300
.998,6.200/.999,7/1,8
TP          TABLE       X2,1,20,50
            GENERATE    ,,,1
SDFG        ADVANCE     100,FN$EXPDIS
            ADVANCE     100,FN$EXPDIS
            ADVANCE     100,FN$EXPDIS
            SPLIT       1,SDFG
            SAVEVALUE   2,C1
            SAVEVALUE   2-,X1
            SAVEVALUE   1,C1
            TABULATE    TP
            TERMINATE   1
```

Оператор **TABLE,** блоки **SPLIT, SAVEVALUE** и **TABULATE** использованы для сбора статистики об интервалах прихода транзактов в модель (об их назначении см. в параграфах 4.17, 4.19 и 4.21).

Построенная в результате моделирования гистограмма (при использовании оператора **START 100000000)** приведена на рис. 4.8. Читателю предлагается исследовать распределение Эрланга при различных значениях *k,* путем изменения количества блоков **ADVANCE** в приведенной программе.

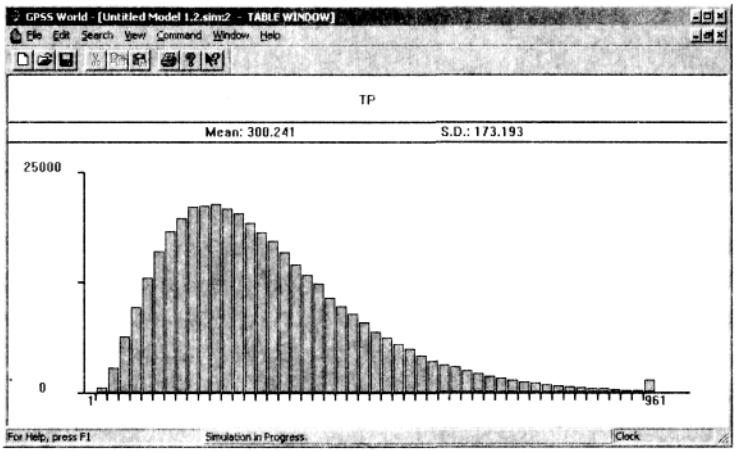


Рис. 4.8

Моделирование нормального закона распределения. Функция стандартного нормального закона распределения с параметрами $m = 0$, $\sigma = 1$ задается в GPSS 24 отрезками следующим образом:

```
NOR    FUNCTION    RN1,C25
0,-5/.00003,-4/.00135,-3/.00621,-2.5/.02275,-2
.06681,-1.5/.11507,-1.2/.15866,-1/.21186,-.8/.27425,-.6
.34458,-.4/.42074,-.2/.5,0/.57926,.2/.65542,.4
.72575,.6/.78814,.8/.84134,1/.88493,1.2/.93319,1.5
.97725,2/.99379,2.5/.99865,3/.99997,4/1,5
```

Для того, чтобы получить функцию нормального распределения случайной величины **X** с математическим ожиданием $m_x \neq 0$ и среднеквадратичным отклонением $\sigma_x \neq 1$, необходимо произвести вычисления по формуле

$$X = m_x + \sigma_x Z, \qquad (4.12)$$

где Z – случайная величина со стандартной нормальной функцией распределения. Например, если случайная величина **X** имеет параметры $m_x = 60$ и $\sigma_x = 10$, то в GPSS эта случайная величина моделируется так:

```
NOR1   FVARIABLE   60+10#FN$NOR
```

Если необходимо осуществить задержку по этому закону распределения, то используется блок

```
ADVANCE      V$NOR1
```

При использовании функции нормального распределения для блоков **GENERATE** и **ADVANCE** необходимо обеспечить неотрицательность значений интервалов поступления и задержки. Это можно сделать, если $m_x \geq 5\sigma_x$.

Моделирование других законов распределения. Все другие виды функций распределения случайных величин в GPSS/PC необходимо задавать табличным способом для конкретных значений параметров этих функций. Для этого можно использовать специальные программы, которые позволяют числовым способом вычислять необходимое значение числа отрезков аппроксимации этих функций, как это сделано, например, в системе ИСИМ [5]. Пример меню такой программы представлен на рис. 4.9.

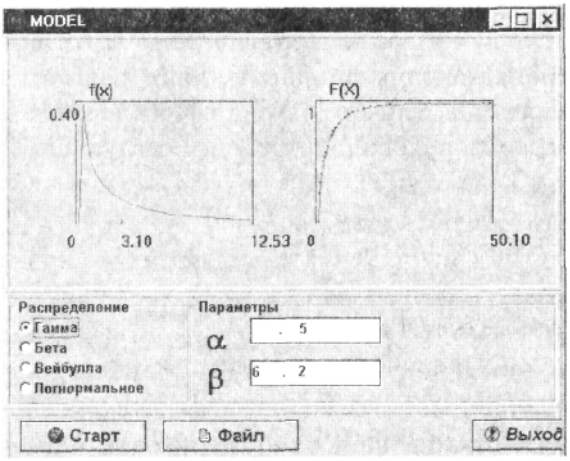


Рис. 4.9

Описание функции гамма-распределения для параметров (рис. 4.9):

```
GAMMA        FUNCTION RN1,C26
0,0/0.24217,0.29471/0.33719,0. 89451/0.40669,0.88412
0.46254,1.17882/0.50946,1.47353/0.5499,1.76824/0.58536,2.06294
0.64503,2.65235/0.69351,3.24176/0.73373,3.83118/0.76758,4.42059
0.79637,5.01/0.83207,5.89412/0.86078,6.77824/0.88409,7.66235
0.90874,8.84118/0.9278,10.02/0.94583,11.49353/0.96139,13.26176
0.97381,15.32471/0.98307,17.68235/0.99062,20.92412/0.99575,25.34471
0.99878,32.41765/0.99992,48.33176/1,50.1
```

Моделирование вероятностных функций распределения в GPSS World. В GPSS World в библиотеку процедур включено 24 вероятностных распределений. При вызове вероятностного распределения требуется определить аргумент ***Stream*** (может быть выражением), который определяет номер генератора случайных чисел. При моделировании генераторы случайных чисел создаются по мере необходимости и их явное определение не обязательно. Большинство вероятностных распределений имеют некоторые параметры. Аргументы процедур, называемые обычно ***Locate, Scale*** и ***Shape,*** часто используются для этих целей. Аргумент ***Locate*** используется после построения применяемого распределения и прибавляется к нему. Это позволяет горизонтально перемещать функцию распределения по оси X. Аргумент ***Scale*** обычно меняет масштаб функции распределения, **А *Shape* –** ее форму.

Встроенная библиотека процедур содержит следующие вероятностные распределения:

1) бета (Beta);
2) биномиальное (Binomial);
3) Вейбулла(Weibula);
4) дискретно-равномерное (Discrete Uniform);
5) гамма (Gamma);
6) геометрическое (Geometric);
7) Лапласа (Laplace);
8) логистическое (Logistic);
9) логлапласово (LogLaplace);
10) логлогистическое (LogLogistic);
11) логнормальное (LogNormal);
12) нормальное (Normal);
13) обратное Вейбулла (Inverse Weibull);
14) обратное Гаусса (Inverse Gaussian);
15) отрицательное биномиальное (Negative Binomial);
16) Парето (Pareto);
17) Пирсона типа V (Pearson Type V);
18) Пирсона типа VI (Pearson Type VI);
19) Пуассона (Poisson);

20) равномерное (Uniform);
21) треугольное (Triangular);
22) экспоненциальное (Exponential);
23) экстремального значения A (Extreme Value A);
24) экстремального значения B (Extreme Value B).

В качестве примера покажем, как для генерации потока транзактов можно использовать библиотечную процедуру экспоненциального распределения с параметром λ = 0,25 и использованием генератора случайных чисел RN1 :

GENERATE (Exponential(l ,0,(l/0.25)))

Из всех приведенных распределений опишем те, которые наиболее часто используются на практике.

Логарифмически нормальное распределение. Логарифмически нормальное распределение (логнормальное) – это распределение случайной величины, натуральный логарифм которой нормально распределен. Это распределение пригодно для моделирования мультипликативных процессов так же, как нормальное – для аддитивных.

С помощью центральной предельной теоремы можно показать, что произведение независимых положительных случайных величин стремится к логарифмически нормальной случайной величине.

Логнормальная случайная величина формируется под влиянием большого числа независимых факторов, причем каждый отдельный фактор оказывает равномерно незначительное и равновероятное по знаку влияние. Прирост каждого следующего фактора пропорционален уже достигнутому к этому времени значению исследуемой величины. То есть рассмотренный характер воздействия является мультипликативным.

Функция плотности логнормального распределения:

$$f_\eta(x) = \frac{1}{\sqrt{2\pi}\sigma(x-\lambda)} e^{-\frac{(\ln(x-\lambda)-\mu)^2}{2\sigma^2}}, \qquad (4.13)$$

если **X** > λ, в противном случае – $f_\eta(x) = 0$.

Если после логарифмирования каждого элемента некоторого набора данных этот трансформированный набор данных нормально распределен, то исходные данные логарифмически нормально распределены.

Это распределение используется при моделировании экономических, информационных, физических и биологических систем. Оно хорошо моделирует процессы в случае, когда значение наблюдаемой переменной является случайной долей от значения предыдущего наблюдения.

Примерами использования этого распределения могут быть:
1) размеры и вес частиц, образуемых при дроблении;
2) доход семьи;
3) зарплата работников;
4) долговечность изделия, работающего в режиме износа и старения;
5) размер банковского вклада;
6) длины слов в языке;
7) длины передаваемых сообщений.

Например, когда неизвестно распределение длины передаваемых сообщений, размера файлов или длины запроса к базе данных, то с большой вероятностью можно предположить логнормальное распределение для этих величин.

Математическое ожидание и дисперсия логнормально распределенной случайной величины таковы:

$$E_\eta = e^{\mu+\frac{\sigma^2}{2}} + \lambda, \qquad (4.14)$$

$$D_\eta = e^{2\mu+\sigma^2}\left(e^{\sigma^2}+1\right), \qquad (4.15)$$

где параметр σ задает среднеквадратическое отклонение, μ – математическое ожидание из нормального распределения, λ – величину сдвига для определения местоположения распределения.

Для вызова логнормального распределения используется библиотечная процедура

LOGNORMAL(Stream, Locate, Scale, Shape),

где ***Stream*** – номер генератора случайных чисел, автоматически преобразуется в целое число, которое должно быть больше или равно 1 ; ***Locate***=*λ,* ***Scale***=*σ;* ***Shape***=*μ.* Все параметры обязательные.

Гамма-распределение является обобщенным распределением Эрланга для случая, когда число **А** суммируемых величин является нецелым. Гамма-распределенная величина имеет значения от 0 до ∞, то есть неотрицательна. Если α – целое, то это будет распределение Эрланга.

Функция распределения значительно изменяет свою форму при различных параметрах, что позволяет использовать это распределение для моделирования различных физических явлений.

Гамма-распределение можно интерпретировать как сумму квадратов нормально распределенных случайных величин, то есть как χ-распределение.

Таким образом, χ^2-распределение, распределение Эрланга и экспоненциальное распределение являются частными случаями гамма-распределения.

Функция плотности гамма-распределения имеет вид:

$$f_\gamma(x)=\frac{\beta^{-\alpha}(x-\lambda)^{\alpha-1}e^{-\frac{(x-\lambda)}{\beta}}}{\Gamma(\alpha)},\quad x>\lambda, \qquad (4.16)$$

где $0 \le x < \infty$; $f_\gamma(x)=0$ если $x<0$; $\Gamma(\alpha)=\int_0^\infty x^{\alpha-1}e^{-x}dx$ – гамма-функция Эйлера.

Математическое ожидание и дисперсия гамма-распределенной случайной величины таковы:

$$E_\gamma=\alpha\beta+\lambda, \qquad (4.17)$$

$$D_\gamma=\alpha\beta^2, \qquad (4.19)$$

где параметр α задает форму распределения, β - масштаб для сжатия или растяжения распределения, λ – величину сдвига для определения местоположения распределения.

Для вызова гамма-распределения используется библиотечная процедура

GAMMA (Stream, Locate, Scale, Shape),

где ***Stream*** **–** номер генератора случайных чисел, автоматически преобразуется в целое число, которое должно быть больше или равно 1; ***Locate***=*λ,* ***Scale***=*β,* ***Shape***=*α.* Все параметры обязательные.

Когда аргумент ***Shape*** равен 1, гамма-распределение вырождается в экспоненциальное. Это означает, что **GAMMA (Stream, Locate, Scale, 1)** имеет то же распределение, что и **EXPONENTIAL (Stream, Locate, Scale).**

Распределение Вейбулла. Это распределение используется яри моделировании жизненного цикла сложного изделия или индивидуума.

Функция плотности распределения Вейбулла имеет вид:

$$f(x)=\frac{\alpha(x-\alpha)^{\alpha-1}}{\beta^\alpha}e^{-\left(\frac{x-\lambda}{\beta}\right)^\alpha} \qquad (4.20)$$

при **X** >α, в противном случае – $f(x) = 0$.

Математическое ожидание и дисперсия:

$$E=\frac{\beta}{\alpha}\Gamma\left(\frac{1}{\alpha}\right)+\lambda, \qquad (4.21)$$

$$D=\frac{\beta^2}{\alpha}\left(2\Gamma\left(\frac{2}{\alpha}\right)-\frac{1}{\alpha}\left(\Gamma\left(\frac{1}{\alpha}\right)\right)^2\right), \qquad (4.22)$$

где Γ(α) – гамма-функция Эйлера, параметр α задает форму распределения, β – интенсивность отказов, λ – величину сдвига для определения местоположения распределения.

Для вызова распределения Вейбулла используется библиотечная процедура

WEIBULL (Stream, Locate, Scale, Shape).

где ***Strcam*** номер генератора случайных чисел, автоматически преобразуется в целое число, которое должно быть больше или равно 1 ; ***Locate***=*λ;* ***Scale***=*β;* ***Shape***=*a.* Все параметры обязательные.

На рис. 4.10 изображен «жизненный цикл» сложного изделия, в котором можно выделить три подцикла (им соответствуют три указанных на графике участка). Каждому периоду соответствует своя

функция $\beta(x)$ и, следовательно, свой закон распределения времени жизни изделия. Для участка приработки изделия $\alpha < 1$, для участка нормальной эксплуатации $\alpha = 1$, для участка старения $\alpha > 1$.

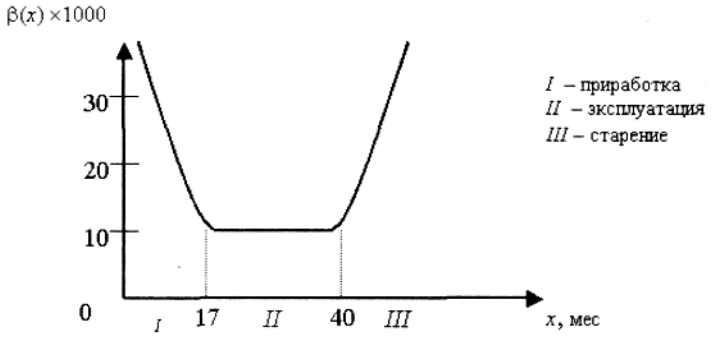


Рис. 4.10

Когда аргумент ***Shape*** равен 1, распределение Вейбулла вырождается в экспоненциальное. Это означает, что **WEIBULL (Stream, Locate, Scale, 1)** имеет то же распределение, что и **EXPONENTIAL (Stream, Locate, Scale).**

4.14. Стандартные числовые атрибуты, параметры транзактов. Блоки ASSIGN, MARK, LOOP

В процессе моделирования интерпретатор автоматически регистрирует и корректирует информацию, касающуюся различных элементов, используемых в модели. Большая часть информации доступна только интерпретатору и используется для сбора статистической информации о работе модели. Однако к некоторым атрибутам (свойствам) объектов может обращаться и программист, управляя процессом моделирования в зависимости от их значений.

Рассмотрим несколько примеров зависимости функционирования элементов модели от системных атрибутов, т.е. СЧА.

1. Интенсивность работы некоторого устройства зависит от длины очереди. Для определения времени обслуживания при каждом поступлении транзакта на обслуживание необходимо знать значение такого системного атрибута, как длина очереди.

2 . Интенсивность обслуживания некоторого устройства зависит от общей продолжительности его функционирования (проявление усталости – интенсивность со временем уменьшается, разогрев устройства – интенсивность со временем увеличивается). Время обслуживания – функция, которая зависит от времени, прошедшего с начала работы.

3. Имеются два устройства и диспетчер, который распределяет работы между ними таким образом, чтобы загрузка устройств была равномерной. Для этого в пункте диспетчеризации необходимо иметь информацию о коэффициентах загрузки устройств и возможность выбора пути перемещения транзакта в зависимости от этих двух величин.

Условно атрибуты можно поделить на две категории:

1) атрибуты системы;

2) атрибуты транзактов.

Атрибуты системы **–** это параметры, которые описывают состояние объектов модели. Такие количественные показатели, как «текущая длина очереди» или «коэффициент загрузки устройства» являются типичными системными атрибутами. Стандартный набор атрибутов, подобных указанным, автоматически поддерживается интерпретатором GPSS.

Транзакты также могут иметь некоторые числовые характеристики (например, ***уровень приоритета).*** Кроме того, транзакт снабжается некоторым числом ***параметров.***

В языке GPSS атрибуты (свойства) объектов – это СЧА. Каждый объект GPSS имеет свой набор СЧА. Доступ к СЧА осуществляется при использовании специальных обозначений этих атрибутов. Имя СЧА состоит из двух частей:

1) ***групповое имя*** **–** состоит из одной или двух букв, идентифицирует тип объекта и тип информации о нем;

2) имя конкретного члена группы.

Объекты могут идентифицироваться с помощью числовых и символьных имен. Если объект идентифицируется с помощью номера (числовое имя), то ссылка на его стандартный числовой атрибут записывается как СЧАj, где *j* – номер объекта (целое число). При символьной идентификации объекта ссылка на его стандартный числовой атрибут записывается, как СЧА$<имя объекта> (в приведенных ссылках под «СЧА» понимается групповое имя).

Стандартные числовые атрибуты. В табл. 4.26 показаны СЧА устройств, в табл. 4.27 – СЧА МКУ, в табл. 4.28 – СЧА очередей. Полный список всех СЧА приведен в Приложении А.

Таблица 4.26

Обозначение	***Значение***
Fj или **F$имя**	Показатель занятости устройства (0 – если не занято, 1 – если занято)
FCj или **FC$имя**	Число занятий устройства
FRj или FR$имя	Нагрузка устройства, выраженная в долях тысячи
FTj или FT$имя	Целая часть значения среднего времени задержки транзакта в устройстве
FVj или **FV$имя**	Флаг готовности устройства к использованию (1 – готово, 0-в противном случае)

Таблица 4.27

Обозначение	***Значение***
Rj или **R$имя**	Емкость незаполненной части МКУ
Sj или **S$имя**	Емкость заполненной части МКУ
SAj или **SA$имя**	Целая часть среднего заполнения МКУ
SCj или **SC$имя**	Счетчик числа входов в МКУ. (При каждом выполнении блока **ENTER** значение счетчика увеличивается на значение операнда **B** этого блока)
SMj или **SM$имя**	Максимально занятая емкость МКУ. Запоминает максимальное значение **Sj** (**S$имя**)
SRj или **SRSимя**	Нагрузка МКУ, выраженная в долях тысячи
STj или **ST$имя**	Целая часть среднего времени пребывания транзакта в МКУ
SVj или **SV$имя**	Флаг готовности МКУj к использованию (1 – готово, 0 – не готово)

Таблица 4.28

Обозначение	*Значение*
Qj или **Q$имя**	Текущее значение длины очереди (текущее содержимое)
QAj или **QA$имя**	Целая часть среднего значения длины очереди
QCj или **QC$имя**	Число входов в очередь. При каждом входе в блок **QUEUE** очереди значение **QCj** (**QC$имя**) увеличивается на значение операнда **B**, при каждом входе в блок **DEPART** очереди значение **QCj** (**QC$имя**) уменьшается на значение операнда **B**
QMj или **QM$имя**	Максимальное значение длины очереди (максимальное значение **Qj** (**C)$имя**))
QTj или **QT$имя**	Целая часть среднего времени пребывания в очереди всех транзактов, которые входили *в* очередь (включая и те, которые не ждали – нулевые входы)

QXj или **QX$имя**	Целая часть среднего времени пребывания в очереди для транзактов, которые ждали в очереди (ненулевые входы)
QZj или **QZ$имя**	Число нулевых входов в очередь

СЧА блоков и системные СЧА. Блоки имеют два стандартных числовых атрибута (их подсчет ведется автоматически):

Wj (W$<метка блока>) – ***счетчик текущего содержимого блока*** с номером *j* (с меткой блока);

Nj (N$<метка блока>) – ***счетчик входов,*** т.е. общее число транзактов, вошедших в блок *j* с момента последнего действия операторов **RESET и CLEAR** (от начала работы модели, если не было операторов **RESET и CLEAR).**

Например, **W$BL1 –** это число транзактов, которые находятся в блоке с меткой **BL1; N$QP** – число транзактов, вошедших в блок с меткой **QP; W210 –** значение текущего содержимого счетчика блока, который имеет 210-ю позицию в модели.

Важные системные СЧА:

C1 – текущее значение ***относительного модельного времени;*** автоматически изменяется интерпретатором и устанавливается в 0 при выполнении операторов **CLEAR и RESET;**

AC1 – текущее значение ***абсолютного*** модельного времени; автоматически изменяется интерпретатором и устанавливается в 0 при выполнении оператора **CLEAR;**

TG1 – текущее значение ***счетчика завершения;***

PR – ***приоритет транзакта,*** обрабатываемого в данный момент;

Ml – ***время пребывания в модели*** транзакта, обрабатываемого интерпретатором в данный момент.

Пример 4.25

Использование СЧА в блоках **ENTER и ADVANCE.**

```
ENTER       3,R3
```

При входе транзакта в блок ENTER, он занимает R3 каналов устройства с именем 3. Поскольку R3 – число доступных каналов МКУ 3, то транзакт занимает все каналы, которые остались свободными до его входа.

```
ENTER       HS,R$SH
```

Войдя в блок **ENTER,** транзакт занимает R$SH каналов МКУ с именем **HS.**

```
ADVANCE     FC$PRIB
```

Задержка в этом блоке равна числу занятий устройства PRIB.

Параметры транзактов. Параметры транзактов – это свойства транзакта, определяемые пользователем. Множество параметров транзакта – набор стандартных числовых атрибутов, которые принадлежат транзакту. Параметры транзакта являются локальными переменными, которые доступны только данному транзакту.

В процессе перемещения транзакта по модели, его параметры могут задаваться и модифицироваться в соответствии с логикой работы модели.

Особенности параметров транзактов:

1. Доступ к параметрам транзактов осуществляется таким образом:

P<номер> или **P$<имя>,**

где **P** – СЧА транзакта, определяющий его групповое имя, т.е. имя всех параметров транзакта.

2. Номера (имена) конкретных членов множества параметров задаются с помощью целых чисел 1, 2, ... или символьных имен. Например, **P22** – это 22-параметр транзакта, **P$COLOR –** параметр с именем **COLOR.**

3. При входе транзакта в модель ***начальное значение*** всех его параметров устанавливается в ***ноль.***

4. Значения параметров транзактов и их изменение определяет пользователь.

5. Значениями параметров транзактов могут быть любые числа (в системе GPSS/PC – только целые числа). Параметры могут приобретать отрицательные значения.

6. Транзакт может обращаться только к своим параметрам. Если необходимо получить доступ к параметрам других транзактов, то это можно сделать только через ячейки сохраняемых величин или используя группы транзактов.

7. Параметры можно использовать в качестве операндов блоков и в качестве аргументов функций.

8. Параметры позволяют организовать косвенную адресацию блоков. Это дает возможность агрегированного представления объектов моделирования в программе.

Пример 4.26

```
AAA    FUNCTION     P3, D3
-3,4/3,7/10,8
       ...
       ADVANCE      FN$AAA,3
```

Выполнение подпрограммы блока **ADVANCE** предусматривает расчет функции **AAA**. Это в свою очередь требует определения значения аргумента функции, т.е. параметра **P3.** Пусть значение третьего параметра гранзакта, вошедшего в блок **ADVANCE,** равно 4. Соответствующее значение функции **AAA** равно 8. Таким образом, время задержки будет равномерно распределено на интервале 8±3.

Пример 4.27

```
SEIZE        PSPRIB
ADVANCE      PSTSERV
RELEASE      PSPRIB
```

Транзакт занимает устройство, номер которого задан параметром транзакта **PRIB.** Время обслуживания задается параметром **TSERV**

Изменение значений параметров. Блок ASSIGN (НАЗНАЧИТЬ). При входе транзакта в этот блок значения параметров могут задаваться или изменяться.

Формат блока:

```
ASSIGN       A[+,-], B[,C]
```

Таблица 4.29

Операнд	***Значение***	***Результат по умолчанию***
A	Номер или имя модифицируемого или задаваемого параметра	Ошибка
B	Величина, используемая для модификации (число или СЧА)	Ошибка
C	Имя функции	Не используется

Блок **ASSIGN** может быть использован как в режиме замещения значения параметра (начальное значение всех параметров транзактов равно 0), так и в режиме увеличения и уменьшения. В режиме увеличения предшествующее значение параметра увеличивается на значение, стоящее в операнде **B**. В режиме уменьшения оно уменьшается на величину, стоящую в операнде B. Режимы увеличения и уменьшения определяются введением соответственно знаков «плюс» и «минус» перед запятой, которая разделяет операнды **A** и **B**.

При использовании операнда **C** значение операнда **B** умножается на значение функции, указанной в операнде C. Параметр, заданный в операнде **A**, изменяется на величину полученного произведения (в режиме увеличения и уменьшения) или приобретает значение результата (в режиме замещения).

Пример 4.28

Блок **ASSIGN** в режиме присваивания:

```
ASSIGN       MEST,36
```

Параметру транзакта с именем **MEST** присваивается значение 36.

```
ASSIGN       3,25
```

Параметру **P3** присваивается значение 25.

ASSIGN P4,FR$BB

Параметру транзакта с номером, записанным в параметре **P4**, присваивается значение величины загрузки устройства **BB** (оба операнда заданы косвенным образом).

Блок **ASSIGN** в режимах накопления и уменьшения:

ASSIGN 4+,Q5

Параметр 4 увеличивается на значение, равное текущей длин очереди 5.

ASSIGN P2-,7

От значения параметра, номер которого задан параметром **P2** вычитается 7.

Отметка времени. При каждом входе транзакта в модель ин терпретатор фиксирует для него текущее значение времени. Это значение времени называется ***отметкой времени.*** Она может быть интерпретирована как время «рождения» транзакта или время входа транзакта в модель. В явном виде отметка времени недоступна. Однако существует СЧА, который тесно связан со значением времет входатранзакта в модель. Его имя **M1**, а значение определяется так:

$$\mathbf{M1} = \begin{pmatrix} \text{Текущее значение} \\ \text{таймера абсолютного} \\ \text{времени} \end{pmatrix} - \begin{pmatrix} \text{Значение времени} \\ \text{входа транзакта} \\ \text{в модель} \end{pmatrix}$$

Значение **M1** для каждого транзакта изменяется в процессе моделирования. Сразу после входа транзакта в модель **M1**=0, через 10 единиц модельного времени **M1**=10 и т.д.

Пример 4.29

Пусть в устройстве, номер которого хранится в параметре **CLR_1** транзакта, продолжительность обслуживания зависит от времени, которое прошло с момента входа транзакта в модель. Предполагается, что это время не может превышать 1000 ед. времени. Для этого в модели можно использовать такие блоки:

```
ZATR  FUNCTION    M1,C2
1,4/1000,80
      …
      SEIZE       P$CLR_1
      ADVANCE     FN$ZATR
      RELEASE     P$CLR_1
      …
```

Транзитное время. Блок MARK (ОТМЕТИТЬ). Стандартный числовой атрибут **M1** измеряет время, которое прошло с момента входа транзакта в модель. Однако очень часто требуется знать время, затраченное на перемещение транзакта между двумя произвольными точками модели. Для этого используется блок **MARK.**

При входе транзакта в блок **MARK** значение таймера абсолютного времени записывается в качестве одного из его параметров. Такую запись называют ***отметкой транзакта.*** Формат блока **MARK:**

MARK A

Таблица 4.30

Операнд	***Значение***	***Результат по умолчанию***
A	Номер параметра, в который записывается значение абсолютного времени (целое число, СЧА)	При отсутствии операнда **A** отметка времени заменяется текущим значением абсолютного времени

Пусть необходимо определить интервал времени, на протяжении которого транзакт проходит от точки T_1 к точке T_2. Для этого нужно выполнить два действия:

1) в точку T_1 поместить блок **MARK j,** где **j –** номер параметра, в который записывается значение абсолютного времени в момент записи;

2) в точке T_2 обратиться к СЧА с именем **MPj,** где **j** – номер параметра, в котором сделана отметка времени транзакта; СЧА **MPj** будет иметь такое значение:

$$\mathbf{MPj} = \begin{pmatrix} \text{Текущее значение} \\ \text{таймера абсолютного} \\ \text{времени} \end{pmatrix} - \begin{pmatrix} \text{Значение} \\ \text{j – го} \\ \text{параметра} \end{pmatrix}$$

Организация циклов. Блок LOOP (ЦИКЛ). С помощью параметров транзактов в программе можно организовать циклы. Для этого используется блок **LOOP.** Он управляет количеством повторных прохождений транзактом определенной последовательности блоков модели.

Формат блока:

LOOP A[,B]

Таблица 4.31

Операнд	***Значение***	***Результат по умолчанию***
A	Параметр транзакта, используемый для организации цикла (переменная цикла). Он может быть именем, положительным целым числом, СЧА, С4А*СЧА (косвенная адресация).	Ошибка
B	Метка (имя блока) начального блока цикла	Ошибка

Когда транзакт входит в блок **LOOP,** параметр, указанный в операнде **A**, уменьшается на единицу, **A** затем проверяется его значение на равенство нулю. Если значение не равно нулю, то транзакт переходит в блок, указанный в операнде **B**. Если значение параметра равно нулю, транзакт переходит в следующий блок.

Переменная блока **LOOP** может только уменьшаться.

Пример 4.30

```
        ASSIGN      1,3
SIS     SEIZE       PC
        ...
        RELEASE     PC
        LOOP        1,SIS
```

Цикл организован по первому параметру транзакта. Его начальное значение равно 3. После освобождения устройства проверяется значение первого параметра. Если оно не равно нулю, го транзакт возвращается к блоку, помеченному меткой **SIS,** т.е. занимает устройство с именем **PC.** Всего каждый транзакт будет занимать это устройство три раза.

Примеры фрагментов GPSS-моделей с использованием СЧА и параметров гранзактов

Пример 4.31

Определение функции, значения которой зависят от текущего содержимого блока с именем **PPP**. Вид зависимости задан в табл. 4.32.

Таблица 4.32

Текущее содержимое блока с именем **PPP**	0	1,2 или 3	4 или *5*	6	7 и больше
Значение функции	1	4	*2*	4	5

```
FFF     FUNCTION    W$PPP, D5
0,1/3,4/5,2/6,4/7,5
```

Пример 4.32

Определение функции, значения которой были бы вдвое больше текущей длины очереди ALPHA для значений 0, 1, 2, 3, 4. Для остальных значений содержимого очереди значение функции должно быть равно 10.

Это можно сделать двумя способами:

1) с помощью дискретной функции, определяемой 6 значениями;

2) с помощью непрерывной функции, определяемой 2 значениями.

Первый способ:

```
LONG        FUNCTION    Q$ALPHA,D6
0,0/1,2/2,4/3,6/4,8/5,10
```

Второй способ:

```
SHORT       FUNCTION    Q$ALPHA, C2
0,0/5,10
```

Пример 4.33 [10]

В СМО с одним устройством и очередью поступает пуассоновский поток заявок с интенсивностью 12 приходов за 1 ч. Обслуживание имеет экспоненциальное распределение, но среднее время обслуживания зависит от числа заявок, которые находятся в очереди к устройству. Эта зависимость приведена в табл. 4.33. Промоделировать обслуживание 500 заявок.

Таблица 4.33

Длина очереди	0	1 или 2	3,4 или 5	6 и больше
Среднее время обслуживания, мин	5.5	5.0	4.5	4.0

Таблица 4.34 (Таблица определений)

Элементы GPSS	***Интерпретация***
Транзакты	Заявки
Устройство SURVR	Обслуживающее устройство
Функции: MEAN XPDIS	Функция, определяющая среднее время обслуживания в зависимости от длины очереди Функция розыгрыша случайных чисел в соответствии с экспоненциальным законом со средним значением 1
Очередь WAIT	Регистратор очереди для сбора статистики о состоянии очереди перед устройством

Для учета длины очереди при определении интенсивности обслуживания в модель необходимо включить дискретную функцию, в которой текущая длина очереди является аргументом. Эта функция используется для определения среднего значения интенсивности обслуживания.

Единица модельного времени – 1 с.

Программа:

```
MEAN    FUNCTION     Q$WAIT,D4
0,330/2,300/5,270/6,240

XPDIS   FUNCTION     RN1,C24
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915/.7,1.2/
.75,1.38/.8,1.6/.84,1.83/.88,2.12/.9,2.3/.92,2.52/.94,2.81/
.95,2.99/.96,3.2/.97,3.5/.98,3.9/.99,4.6/.995,5.3/.998,6.2/
.999,7/.9998,8

GENERATE    300,FN$XPDIS
QUEUE       WAIT
SEIZE       SURVR
DEPART      WAIT
ADVANCE     FN$MEAN,FN$XPDIS
RELEASE     SURVR
TERMINATE   1

START       500
```

Пример 4.34[10]

Устройство с экспоненциальным временем обслуживания имеет свойство уменьшать интенсивность своей работы на протяжении восьмичасового рабочего дня. В течение первых двух часов ему требуется в среднем 12 мин для выполнения обслуживания. В течение последующих двух часов среднее время обслуживания составляет 15 мин. В течение пятого, шестого и седьмого часа – 17 мин, в течение восьмого часа – 20 мин. Предполагая, что единица времени в модели равна 0,1 мин, определить функцию, значения которой давали бы среднее время, необходимое устройству для выполнения обслуживания. Также показать, как эту функцию нужно использовать в блоке **ADVANCE.** (СЧА **C1** – текущее значение относительного времени работы модели).

Фрагмент программы:

```
LAMDA FUNCTION     C1,D4
1200,120/2400,150/4200,170/4800,200

XPDIS              FUNCTION     RN1, C24
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915/.7,1.2/
.75,1.38/.8,1.6/.84,1.83/.88,2.12/.9,2.3/.92,2.52/.94,2.81/
.95,2.99/.96,3.2/.97,3.5/.98,3.9/.99,4.6/.995,5.3/.998,6.2/
.999,7/.9998,8
 ...
ADVANCE        FN$LAMBDA,FN$XPDIS
 ...
```

Пример 4.35 [10]

Ситуация, описанная в примере 4.34, предполагает, что среднее время обслуживания увеличивается скачкообразно. Более реально полагать, что среднее время возрастает постепенно в течение дня. Определить непрерывную функцию, которая описывала бы увеличение времени обслуживания в соответствии с правилом: в нулевой момент времени среднее время равно 12 мин, к концу второго часа оно увеличивается до 15 мин, к концу четвертого часа – до 17 мин, к концу седьмого – до 20 мин, **А** к концу восьмого – до 25 мин. Считать, что среднее время обслуживания на указанных интервалах времени увеличивается непрерывно и равномерно.

Фрагмент программы:

```
MEAN               FUNCTION     C1, C5
0,120/1200,150/2400,170/4200,200/4800,250

XPDIS              FUNCTION     RN1, C24
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915/.7.1.2/
.75,1.38/.8,1.6/.84,1.83/.88,2.12/.9,2.3/.92,2.52/.94,2.81/
.95,2.99/.96,3.2/.97,3.5/.98,3.9/.99,4.6/.995,5.3/.998,6.2/
.999,7/.9998,8
 ...
          ADVANCE      FN$MEAN, FN$XPDIS
 ...
```

4.15. Изменение приоритета транзактов. Блок PRIORITY

Блок **PRIORITY (НАЗНАЧИТЬ ПРИОРИТЕТ)** присваивает или изменяет приоритет транзакта, если он был задан блоком **GENERATE** (по умолчанию приоритет транзакта равен нулю). Его формат:

PRIORITY A[,B]

Таблица 4.35

Операнд	***Значение***	***Результат по умолчанию***
A	Новое значение приоритета (целое число, СЧА, СЧА*СЧА)	Ошибка
B	Этот операнд определяет режим BUFFER (подробнее о нем см. документацию [ ])	

Новое значение приоритета может быть меньше, больше или равно текущему значению приоритета транзакта. Приоритет влияет на порядок выбора транзакта для обслуживания устройствами и на порядок просмотра транзактов в списке текущих событий [7].

Стандартный числовой атрибут этого блока – **PR**. Поскольку уровень приоритета транзакта может изменяться от 0 до 127, то **PR** будет выдавать значение в диапазоне 0-127.

Пример 4.36

```
PRIORITY      100
```

Вошедшему в этот блок транзакту присваивается приоритет 100.

```
DELAY          FUNCTION     PR, D3
1,4/2,7/3,10
 …
        ADVANCE      FN$DELAY
 …
```

Задержка в блоке **ADVANCE** зависит от приоритета транзакта. Гранзакт с наиболее низким приоритетом (1) задерживается на 4 единицы модельного времени, транзакт с наиболее высоким приоритетом (3) задерживается на 10 единиц модельного времени.

4.16. Организация обслуживания с прерыванием. Блоки PREEMPT и RETURN

Во многих случаях возникает необходимость организации обслуживания в устройстве с прерываниями (например, при выполнении некоторой операции на станке произошла его поломка). Такую ситуацию можно смоделировать, считая, что отказ оборудования представляет собой транзакт, приоритет которого выше, чем у транзакта, обрабатываемого станком. В этом случае более приоритетный транзакт должен прервать обслуживание менее приоритетного транзакта, т.е. выгрузить его из устройства. Отсюда понятен дословный перевод с английского слова ***preempt*** – выгрузить, но с точки зрения работы одноканальной СМО принято использовать термин ЗАХВАТИТЬ устройство [10]. Для организации обслуживания в устройстве с прерываниями используют пару блоков **PREEMPT (ЗАХВАТИТЬ) – RETURN (ВЕРНУТЬ)** так же, как для обычного устройства без прерываний использовались блоки **SEIZE – RELEASE.**

Блок **PREEMPT** имеет следующий формат:

PREEMPT A,[B],[C],[D],[E]

Таблица 4.36

Операнд	***Значение***	***Результат по умолчанию***
A	Имя устройства (числовое или символьное)	Ошибка
B	Возможность захвата по приоритету	Режим прерывания

C	Имя блока (числовое или символьное), в который переходит прерванный транзакт	См. объяснение ниже
D	Номер параметра (числовое или символьное имя) у прерванного транзакта	См. объяснение ниже
E	Возможность снятия с обслуживания	См. объяснение ниже

Блок **PREEMPT** позволяет транзакту в зависимости от условий, заданных в операндах блока, занять устройство. Блок **PREEMPT** может также задержать транзакт на входе.

Операнд **A** определяет номер или имя устройства, на котором генерируется прерывание. Операнд может быть именем, положительным целым, СЧА или СЧА*СЧА.

Операнд **B** задает приоритетный режим (если **B**=PR) или режим прерывания (если этот операнд опущен). При работе в приоритетном режиме транзакт, уже занимающий устройство или генерирующий на нем прерывание, может быть прерван только транзактом, приоритет которого выше приоритета данного транзакта. Прерванные транзакты претендуют на дополнительное использование устройства, когда прервавший их транзакт войдет в соответствующий блок **RETURN.** Прерванные транзакты помещаются в список задержки в порядке приоритета.

Операнд **C** задает номер или имя блока, в который в этот же момент времени должен попытаться войти прерванный транзакт. Прерванный транзакт теряет управление устройством, но претендует на право его использования, если только не задан аргумент операнда E. В приоритетном режиме работы желательно задавать операнд **C**, если прерывающий транзакт имеет более высокий приоритет, чем прерываемый. Операнд может быть именем, положительным целым, СЧА или СЧА*СЧА.

Операнд **D** задает номер параметра, связанного с прерванным транзактом. Если прерываемый транзакт в момент прерывания направляется в список будущих событий (см. параграф 4.22), тогда остаток времени записывается в заданный параметр. Если такой параметр не существует, то он создается. В приоритетном режиме работы операнд **D** задают только в том случае, если прерывающий транзакт имеет более высокий приоритет, чем прерываемый транзакт. Операнд может быть именем, положительным целым, СЧА или СЧА*СЧА.

Операнд **E** задает либо не задает режим удаления (**RE**). В режиме удаления **RE** прерванный транзакт более не претендует на использование устройства и пытается войти в блок, заданный операндом **C** (если в операнде **E** стоит **RE**, то должен быть определен и операнд **C**). В приоритетном режиме работы режим RE используется только в том случае, если приоритет прерывающего транзакта больше приоритета прерываемого транзакта. При использовании **RE** прерванный транзакт не должен входить в блоки **RELEASE** или **RETURN,** связанные устройством, в котором обслуживался прерванный транзакт. Если режим **RE** не задан (операнд **E** опущен), то прерванный транзакт по возвращении в список текущих событий будет вновь пытаться занять устройство.

Прерываемый транзакт может находиться в списке будущих событий. Если надо сделать это, то используют операнд **D**.

Прерванный транзакт борется за устройство, даже если он перемещен операндом **C** (если **RE** не используется в операнде **E**). Если прерванный транзакт все еще борется за устройство, то попытка транзакта войти в блок **TERMINATE** приводит к ошибке. Такой транзакт перед входом в блок **TERMINATE** должен войти в блоки **RELEASE** или **RETURN**

Транзакт может быть прерван на любом количестве устройств.

Устройство может быть захвачено любое количество раз, но не два раза подряд одним транзактом.

Транзакт не может войти в блок, если в приоритетном режиме устройство уже захвачено транзактом с приоритетом равным или большим, чем приоритет активного транзакта. Активный транзакт помещается в соответствии с приоритетом в список задержки устройства.

Транзакт не может войти в блок, если устройство находится в недоступном состоянии. Такие транзакты помещаются в список задержки устройства в соответствии с приоритетом, **A** внутри приоритета – по правилу FIFO.

Стандартные числовые атрибуты, связанные с описываемым блоком, те же, что и в табл. 4.26, с добавлением СЧА **FIj** – флаг прерывания устройства (1, если устройство находится в состоянии прерывания, 0 – в противном случае).

Следует обратить внимание, что при задании операндов **D** и (или) **E**, операнд **C** также должен быть задан.

Если приоритетный режим не задан (нет **PR** в операнде **B**), то операнды **C**, **D** и (или) **E** игнорируются. Однако возможен вариант, когда для прерванного транзакта выбирается альтернативный выход, причем приоритет транзакта не учитывается. Этот случай возникает тогда, когда задан операнд **C** (а иногда и операнды **D** и (или) **E**), но в операнде **B** не задан приоритетный режим. Такое использование операндов приводит к тому, что занимающий устройство транзакт прерывается и направляется по альтернативному пути. В данном случае многоуровневые прерывания не происходят.

Пары блоков **SEIZE – RELEASE** и **PREEMPT – RETURN** могут использовать одни и те же имена занимаемых устройств. В зависимости от логики работы модели пользователь должен сам определить, в каком случае разрешать прерывания, **A** в каком – нет.

Блок **RETURN** является парным к блоку **PREEMPT,** также как блок **RELEASE** к блоку **SEIZE,** и предназначен для освобождения ранее захваченного устройства. Он имеет следующий формат:

RETURN A

Таблица 4.37

Операнд		***Значение***		***Результат по умолчанию***
A	Имя устройства (числовое или символьное)			Ошибка

В операнде **A** задается номер устройства, с которого снимается прерывание. Прерывание может быть снято в блоке **RETURN** только тем транзактом, которым оно было сгенерировано.

Операнд **A** может быть именем, положительным целым, СЧА или СЧА*СЧА.

Пример 4.37

Рассмотрим пример работы компьютера, задействованного в управлении технологическим оборудованием. Для контроля состояния оборудования каждые 20 мин запускается одна из трех типов задач. Через каждые 5 мин работы процессора каждая задача выводит результаты работы в базу данных. При обращении двух и более задач к базе данных (БД) образуется очередь, которая обслуживается по правилу FIFO.

Общий объем памяти компьютера 1024Кбайт. В первоначальный момент запуска компьютера загружается ОС, ядро которой постоянно находится в памяти и занимает 200 Кбайт. Компьютер рабогает в мультипрограммном режиме и во время выполнения операций вывода в БД процессор может выполнять другую задачу, если она загружена в память. После последнего вывода в БД задача выгружается из памяти и завершает свою работу.

Периодически с интенсивностью λ=0,005 мин$^{-1}$ и экспоненциальным распределением возникает аварийный режим оборудования, при котором немедленно запускается на выполнение задача четвертого типа, выводящая оборудование из аварийного режима. Она прерывает работу всех других задач. Прерванная задача выгружается из памяти без вывода результатов в БД. По окончании выполнения задачи четвертого типа, она имеет преимущество для вывода в БД перед другими задачами. Вытесненные задачи с магнитного диска загружаются в память и продолжают работу. Необходимые данные для моделирования приведены в табл. 4.38.

Таблица 4.38

Тип задачи	1	2	3	4
Вероятность возникновения	0,5	0,3 5	0,1 5	–
Объем памяти, Кбайт	200	300	400	500
Время обработки ЦП, мин	15	20	25	5
Время вывода в БД, мин	3	5	7	2

Необходимо промоделировать работу компьютера в течение пяти суток и оценить размер очереди к памяти, ее загрузку и загрузку процессора.

Учитывая, что программа полностью прокомментирована, дадим только некоторые пояснения к ней. Задачи 1-го, 2-го, 3-го типов имеют приоритет равный 0. Задача четвертого типа, обрабатывающая аварийную ситуацию, имеет приоритет 3. При ее появлении немедленно занимается процессор и, если в это время выполняется задача другого типа, то она прерывается и выгружается из памяти (операнд **B** блока **PREEMPT** направляет прерванный транзакт в блок с меткой **SVOP).**

Для запуска работы компьютера используется один транзакт с приоритетом 5, который занимает 2 единицы памяти (200 Кбайт) и имитирует загрузку ядра ОС в память.

Программа:

```
EXPDIS FUNCTION     RN1,C24
0,0/.100,.104/.200,.222/.300,.355/.400,.509
.500,.690/.600,.915/.700,1.200/.750,1.380
.800,1.600/.840,1.830/.880,2.120/.900,2.300
.920,2.520/.940,2.810/.950,2.990/.960,3.200
.970,3.500/.980,3.900/.990,4.600/.995,5.300
.998,6.200/.999,7/1,8
;
TYPE            FUNCTION    RN1,D3  ; Тип задачи
.5,1/.85,2/1,3
;
RAM             FUNCTION    P1,D3   ; Объем памяти в 100 Кбайт для задач 1, 2,
                                    ; 3 типов
1,2/2,3/3,4
TIME_CP         FUNCTION    P1,D3   ; Время работы процессора для задач
                                    ; 1, 2, 3 типов
1,15/2,20/3,25
TIME_BD         FUNCTION    P1,D3   ; Время вывода в БД  для задач
                                    ; 1, 2, 3 типов
1,3/2,5/3,7
RAM             STORAGE     10      ; Объем памяти  компьютера 100 Кбайт
*
        GENERATE    20                  ; Время появления
                                        ; задач 1, 2, 3 типов
        ASSIGN      1,FN$TYPE           ; Определение типа задачи
        ASSIGN      2,FN$RAM            ; Требуемый объем памяти
        ASSIGN      3,FN$TIME_BD        ; Время вывода в БД
        ASSIGN      4,FN$TIME_CP        ; Требуемое время процессора
        QUEUE       Q_RAM               ; Ждать освобождения памяти
        ENTER       RAM,P2              ; Занять память для задачи
        DEPART      Q_RAM               ; Освободить очередь к памяти
CALC    SEIZE       CP                  ; Занять процессор
        ADVANCE     5                   ; Считать 5 мин
        RELEASE     CP                  ; Освободить процессор
        SEIZE       BD                  ; Начать вывод в БД
        ADVANCE     P3                  ; Время вывода в БД
        RELEASE     BD                  ; Закончить вывод в БД
        ASSIGN      4-,5                ; Сколько времени процессора
                                        ; еще надо?
        TEST LE     P4,0,CALC           ; Задача закончилась?
SVOP    LEAVE       RAM,P2              ; Да, освободить память RAM
        STORAGE     10                  ; Объем памяти компьютера 100 Кб
*
        GENERATE    200,FN$EXPDIS,,,3   ; Появление аварийной
                                        ; ситуации – задача 4
        QUEUE       Q_RAM
        PREEMPT     CP,PR,SVOP          ; Занять немедленно процессор,
                                        ; выгрузить прерванную
                                        ; задачу из памяти
        DEPART      Q_RAM
        CALC        SEIZE       CP      ; Занять процессор
        ADVANCE     5                   ; Считать 5 мин
        RETURN      CP                  ; Освободить процессор
        SEIZE       BD                  ; Начать вывод в БД
        ADVANCE     2                   ; Время вывода в БД
        RELEASE     BD                  ; Закончить вывод в БД
        LEAVE       RAM,2               ; Освободить память
        TERMINATE                       ; Завершить задачу типа 4
*
        GENERATE    ,,,1,5              ; Запустить компьютер
        ENTER       RAM,2               ; Загрузить ядро ОС в память
        ADVANCE     7200                ; Работать 5 суток (60*24*5 мин)
        LEAVE       RAM,2               ; Выгрузить ядро ОС
        TERMINATE   1                   ; Завершить работу компьютера
```

Результаты моделирования:

```
FACILITY      ENTRIES  UTIL.   AVE. TIME AVAIL. OWNER PEND INTER RETRY DELAY
 CP             1293   0.887     4.940   1        0    0    0     0      0
 BD             1261   0.783     4.473   1      397    0    0     0      0

QUEUE          MAX CONT. ENTRY ENTRY(0) AVE.CONT. AVE.TIME   AVE.(-0) RETRY
 Q_RAM           5    0    395    194      0.711     12.954     25.457   0

STORAGE        CAP. REM. MIN. MAX.  ENTRIES AVL.  AVE.C. UTIL. RETRY DELAY
 RAM            10    8   0    10     1023   1    8.101  0.810    0     0
```

4.17. Сохраняемые величины

В GPSS пользователю предоставляется возможность определить «свои» глобальные переменные, начальные значения которых могут быть заданы перед моделированием и к которым можно обратиться из любого места модели в любой момент времени. Эти переменные называют ***сохраняемыми величинами (ячейками).*** Совокупность логически связанных между собой ячеек образует ***матрицу*** (аналог массива).

В отличие от параметров транзакта, приоритета и отметки времени, которые теряются в момент выхода транзакта из модели, ячейки доступны на протяжении всего процесса моделирования. Значения сохраняемых величин не подсчитываются интерпретатором автоматически (как СЧА устройств, очередей, МКУ и т.п.), **А** задаются и изменяются программистом.

Сохраняемые величины могут принимать положительные и отрицательные значения. Стандартный числовой атрибут **Xj (X$<имя ячейки>)** дает значение соответствующей сохраняемой величины. Например, **X2** – значение ячейки 2; **X$DAY –** значение ячейки **DAY**.

С матрицами связан стандартный числовой атрибут **MXj**(*m, n*) – значение, записанное в строке *m* и в столбце *n* матрицы **j** или **MX$<имя матрицы>** *(m,n),* если матрица имеет символьное имя.

СЧА сохраняемой величины может быть использован для косвенного задания данных, **А** также как аргумент функций и таблиц.

Перед использованием матрица определяется оператором описания **MATRIX.** Начальные значения ячеек и матриц можно задать с помощью оператора описания **INITIAL.**

1. Допускается косвенная адресация ячеек, матриц, **А** также их строк и столбцов. Например: **X*P2** – значение ячейки с номером, который сохраняется в параметре 2; **MX*P5(5,2)** – элемент (5,2) матрицы, номер которой записан в параметре 5; **MX3(P4,X2)** – значение, записанное в строке Р4 и в столбце Х2 матрицы 3.

2. Недопустимой является запись такого вида: **MXl(MX2(l,2),MX3(3,4)),** поскольку нельзя задавать номера строк и столбцов матрицы с помощью СЧА матриц.

Оператор INITIAL (ИНИЦИАЛИЗИРОВАТЬ). Если в процессе моделирования происходит обращение к сохраняемой величине, которая не была задана, то интерпретатор выдает ошибку в процессе выполнения программы. Поэтому перед началом моделирования все сохраняемые величины должны быть инициализированы с помощью оператора **INITIAL.**

Формат оператора представлен в таблице.

Таблица 4.39

Поле	***Информация в поле***
Метка	Не используется
Операция	**INITIAL**
Операнд **А**	Имя сохраняемой величины
Операнд **В**	Начальное значение

Пример 4.38

INITIAL X$TIMER,1000000

Ячейке **TIMER** присваивается начальное значение 1000000;

INITIAL X3,2S

Ячейке с номером 3 присваивается начальное значение 25.

INITIAL MX8(2,4),-53

Величина – 53 записывается в строку 2, столбец 4 матрицы 8.

INITIAL MX$COST(l,3),22

Величина 22 записывается в строку 1, столбец 3 матрицы **COST.**

Блок SAVEVALUE (СОХРАНИТЬ ВЕЛИЧИНУ). Значение сохраняемой величины изменяется при входе транзакта в блок, **SAVEVALUE** (СОХРАНИТЬ ВЕЛИЧИНУ).

Его формат:

SAVEVALUE A[+,-],B

Таблица 4.40

Операнд	***Значение***	***Результат по умолчанию***
A	Номер или символьное имя сохраняемой величины	Ошибка
B	Величина, используемая для модификации (число или СЧА)	Ошибка

Подобно блоку **ASSIGN** блок **SAVEVALUE** может быть использован как в режиме замещения величины, так и в режиме увеличения или уменьшения. В режиме увеличения предыдущее значение сохраняемой величины увеличивается на значение, заданное операндом **B**, **A** в режиме уменьшения – уменьшается на это значение. Режимы увеличения и уменьшения определяются введением соответственно знака «плюс» или «минус» перед запятой, разделяющей операнды **A** и **B**.

Пример 4.39

SAVEVALUE P5,VSALPHA

При входе транзакта в блок **SAVEVALUE** прежде всего вычисляется величина переменной **ALPHA.** Полученный результат присваивается сохраняемой величине, номер которой записан в параметре **P5**.

SAVEVALUE 5+,X2

При входе транзакта в блок величина **X5** увеличивается на значение величины **X2**.

SAVEVALUE PROFIT-,FN$COSTS

При входе транзакта в блок величина **X$PROFIT** уменьшается на значение функции **FN$COSTS**.

Оператор описания матрицы MATRIX. Каждая матрица должна быть объявлена до ее использования, т.е. должна иметь оператор описания. Формат оператора представлен в таблице.

Таблица 4.41

Поле	***Информация в поле***
Метка	Имя матрицы
Операция	MATRIX
Операнд **A**	Не используется (оставлен для совместимости с более старыми версиями GPSS)
Операнд **B**	Число строк матрицы (целое положительное)
Операнд **C**	Число столбцов матрицы (целое положительное)

Матрица в GPSS/PC может содержать не более чем 8191 элемент. Она может быть переопределена или инициализирована повторно другим оператором **MATRIX** с тем же именем. Переопределение, при

котором размер матрицы изменяется, повлечет за собой выделение памяти под новую матрицу. Выделенная до этого оперативная память остается занятой.

Блок MSAVEVALUE. Блок MSAVEVALUE используется для записи значений в матрицы, **A** также для увеличения или уменьшения значений элементов матриц. Его формат:

MSAVEVALUE A[+,-|3,C,D

Таблица 4.42

Операнд	***Значение***	***Результат по умолчанию***
A	Имя матрицы	Ошибка
B	Номер строки матрицы	Ошибка
C	Номер столбца матрицы	Ошибка
D	Величина, используемая для модификации	Ошибка

Операнды **A**, **B** и **C** могут быть именем, положительным целым, СЧА или СЧА*СЧА. Операнд **D** может быть именем, СЧА или СЧА*СЧА.

Подобно блокам **ASSIGN** и **SAVEVALUE** этот блок может быть использован как в режиме замещения величины, так и в режиме увеличения или уменьшения.

Когда транзакт входит в блок **MSAVEVALUE,** то анализируется операнд **A** и ищется матрица с указанным именем. Если матрица ненайдена, то возникает ошибка. Соответствующий элемент матрицы определяется содержимым операндов **B** и **C**. Если такого элемента не существует, то также возникает ошибка.

4.18. Проверка числовых выражений. Блок TEST

Сравнение СЧА может быть выполнено с помощью блока **TEST (ПРОВЕРИТЬ)**
Его формат:

TEST X A,B|,C|

Таблица 4.43

Операнд	***Значение***	***Результат по умолчанию***
A	СЧА	Ошибка
B	СЧА	Ошибка
C	Имя блока, в который переходит транзакт при условии, что ответ на вопрос, подразумеваемый оператором отношения, отрицательный	При отсутствии операнда **C** проверку выполняют в режиме отказа
X	Вспомогательный оператор, который представляет собой оператор отношения, использующийся при проверке	
	Значение оператора отношений:	***Вопрос оператора отношения:***
	G	**A** больше **B**?
	GE	**A** больше или равно **B**?
	E	**A** равно **B**?
	NE	**A** не равно **B**?
	LE	**A** меньше или равно **B**?
	L	**A** меньше **B**?

Операнды **A** и **B** – имена СЧА, которые сравниваются. Вспомогательный оператор **X** указывает способ сравнения этих двух СЧА друг с другом.

Пример 4.40

Режим отказа

```
TEST LE      Ql,Q2
```

Проверяющий транзакт будет задержан в предыдущем блоке до тех пор, пока длина первой очереди не станет меньше или равна длине второй очереди.

Пример 4.41

Режим условного перехода

```
TEST LE      Ql,Q2,ZHVS
```

Проверяющий транзакт перейдет в следующий по порядку блок, если содержимое первой очереди меньше или равно содержимому второй очереди. Если это условие не выполняется, транзакт перейдет в блок с меткой **ZHVS.**

Пример 4.42

```
      TEST L       KSSCANNER,XSMAX_lJTIL,LLL
      SEIZE        SCANNER
LLL   QL'EL'E      QSCANNER2
```

Если устройство **SCANNER** имеет коэффициент загрузки меньше, чем значение сохраняемой величины **MAX_UTIL,** то транзакт идет на обслуживание этим устройством, в противном случае – переходит к блоку с меткой **LLL.**

Пример 4.43

Пусть в точке модели **DISPATCHER** необходимо удалить те транзакты, которые находились в модели больше, чем 100 ед. модельного времени. Для этого в модели можно использовать такие блоки:

```
            GENERATE    18,6
            ...
DISPATCHER  TEST L      M1,100,KILL
            ...
KILL        TERMINATE
```

4.19. Определение и использование таблиц

Для накопления выборочных значений случайных величин и статистической обработки этих выборок используются GPSS-таблицы. Графическим аналогом GPSS-таблицы является гистограмма выборочных значений случайной величины, которую можно просмотреть в окне таблицы. Прежде чем использовать таблицу, ее нужно определить, **А** потом задать собираемые выборочные значения.

Оператор TABLE (ТАБЛИЦА). В модели может быть несколько таблиц. Каждую таблицу нужно сначала определить и только потом использовать в модели. Для определения таблицы необходимо указать:

1) имя таблицы (числовое или символьное);

2) имя случайной переменной, значение которой будет табулироваться;

3) число, являющееся первым граничным значением. (Значения выборки, меньшие или равные этому числу, попадают в самый левый (нижний) интервал (частотный класс) таблицы);

4) ширину интервала, общую для всех интервалов таблицы за исключением левого (нижнего) и правого (верхнего);

5) общее число интервалов таблицы, включая нижний и верхний. Формат оператора представлен в таблице.

Таблица 4.44

Поле	***Информация поля***
Метка	Имя таблицы
Операция	**TABLE**
Операнд **А**	СЧА, значение которого учитывается в таблице

Операнд **B**	Первое граничное значение (целое число)
Операнд **C**	Ширина всех промежуточных интервалов (целое положительное число)
Операнд **D**	Общее число интервалов таблицы, включая левый и правый (целое положительное число)

На рис. 4.11 показана ось действительных значений и ее разделение на ряд интервалов таблицы.

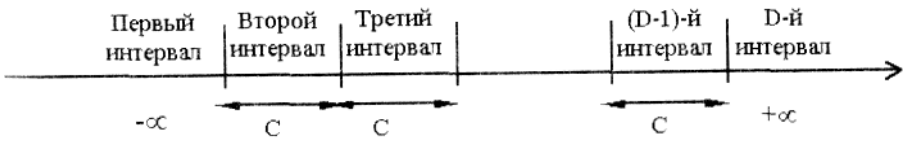


Рис. 4.11

Для сбора статистических данных об очередях используется оператор **QTABLE.** Его формат совпадает с форматом оператора **TABLE,** за исключением того, что операнд **A** задает имя очереди.

Блок TABULATE (ТАБУЛИРОВАТЬ). Выборочные значения попадают в таблицу в моменты вхождения транзактов в блок **TABULATE**. Его формат:

TABULATE A

Таблица 4.45

Операнд	***Значение***	***Результат по умолчанию***
A	Имя (символьное или числовое) таблицы, в которой табулируется соответствующий СЧА	Ошибка

Операнд **A** задает имя таблицы, в которую попадают выборочные значения. Одну таблицу можно использовать в нескольких блоках **TABULATE** модели. Отметим, что СЧА, по которому собирается статистика, в блоке **TABULATE** не указывается, так как он уже записан в операторе **TABLE.**

Если в модели используется блок **TABLE** или **QTABLE,** в файле стандартной статистики информация будет представлена в таком виде:

TABLE Имя таблицы	MEAN Среднее значение	STD. DEV. Среднеквадратическое отклонение	RANGE Границы интервалов	RETRY Ждут условий	FREQUENCY Количество попаданий	CUM,% Накопленная частость
5	1432.75	475.42	700 - 800	0	1	12,50
			800 - 900		1	25,00
			1100 - 1200		1	37,50
			1400 - 1500		1	50,00
			1600 - 1700		1	62,50
			1800 - 1900		2	87,50
			1900		1	100,00

Для получения этих данных была определена GPSS-таблица MVP, в которой фиксировалось время нахождения транзакта в модели:

MVP TABLK Mi,100,100,20

Стандартные числовые атрибуты таблицы:

TB<номер таблицы>, TB$<имя таблицы> вычисленное среднее значение соответствующего СЧА;

TC<номер таблицы>, TC$<имя таблицы> общее число входов в таблицу:

TD<номер таблицы>, TO$<имя таблицы> вычисленное средне-квадратическое отклонение соответствующего СЧА.

Пример 4.44

```
TIMSERV     TABLE       MP3,10,5,6      ; Оператор определения таблицы
            GENERATE    100,15
            ...
            MARK        3
            ...
            TABULATE    TIMSERV         ; Оператор табулирования
```

В таблице **TIMSERV** будет табулироваться частотное распределение случайной величины – времени пребывания транзакта между блоками **MARK** и **TABULATE.** В табл. 4.46 приведены интервалы регистрации времени пребывания между блоками **MARK** и **TABULATE.**

Таблица 4.46

Левый (нижний) интервал	Первый интервал	-∞, < 10
Промежуточные интервалы	Второй интервал	≥10, <15
	Третий интервал	≥15, <20
	Четвертый интервал	≥20, <25
	Пятый интервал	≥20, < 30
Правый (верхний) интервал	Шестой интервал	≥30, + ∞

Для того, чтобы данные, собираемые в таблицу, не попадали в один-два интервала, рекомендуется:

1) сначала сделать пробный прогон; по пробному прогону определить диапазон значений, в которых может колебаться анализируемая случайная величина (в GPSS/PC перейти к окну таблиц с помощью клавиш [ALT+T]);

3) с учетом полученного диапазона скорректировать значения операндов **B, C** и **D** соответствующей таблицы.

Пример 4.45

В таблице **SYSTIME** собирается время пребывания транзакта в модели, которое предположительно принимает значения в интервале [100,2000].

```
SYSTIME     TABLE       M1,100,12,10
            ...
SAVEVALUE   STD,TB$SYSTIME
```

При входе транзакта в блок **SAVEVALUE** среднее значение СЧА (времени пребывания), соответствующего таблице **SYSTIME** заносится в сохраняемую величину с именем **STD.**

4.20. Косвенная адресация

Ранее были рассмотрены прямые способы адресации, когда:

1) номер объекта задается константой

```
QUEUE       2       ; Стать в очередь 2
...
SEIZE       1       ; Занять устройство 1
```

2)номер объекта задается СЧА

```
LEAVE       P1      ; Освободить МКУ с номером, задаваемым параметром 1
```

Адресация может быть относительной и по отношению к самому блоку. Это осуществляется при помощи записи **+/-n.* Например,

```
TRANSFER    ,*+2
```

В этом случае сам блок **TRANSFER** является ориентировочным блоком. Транзакт пытается войти во второй, относительно блока **TRANSFER,** блок.

Подобные ссылки не зависят от свойств транзакта, обрабатываемого в данный момент времени. Использование прямой адресации может привести к введению большого числа дополнительных блоков только для того, чтобы записать номера объектов в СЧА, т.е. к увеличению объема модели.

Существенным образом сократить объем модели и использовать зависимость номеров объектов от свойств транзактов позволяет косвенная адресация.

Общий формат косвенной адресации:

СЧА*СЧАj или **СЧА*СЧА<имя>**

Там, где допустимо использование СЧА, обычно могут использоваться и СЧА*СЧА.

При использовании косвенной адресации параметр **P** может опускаться.

Пример 4.46

SEIZE P*X1

Занять устройство, номер которого содержится в параметре транзакта, **A** номер параметра определяется значением ячейки **X1.**

SAVEVALUE l,X*P2

Поместить в ячейку с номером 1 значение, содержащееся в ячейке, номер которой определяется параметром 2.

Пример 4.47

```
4       FUNCTION    RN1,D3          ; Функция распределения типов сообщени
0.2,1/0.5,2/0.999999,3

1       FUNCTION    RN2,C20         ; Функция распределения числа символов
                                    ; сообщения типа 1
        ...
2       FUNCTION    RN3,C20         ; Функция распределения числа символов
                                    ; сообщения типа 2
        ...
3       FUNCTION    RN4,C20         ; Функция распределения числа символов
                                    ; сообщения типа 3
...
GENERATE    100,10
ASSIGN      1,FN4           ; Определение типа сообщения
ASSIGN      2,FN*P1         ; Определение числа символов сообщения
```

В данной модели предполагается, что через некоторый узел сети проходят информационные сообщения грех типов, причем каждый из них характеризуется своим распределением числа символов. В модели первый параметр транзакта содержит тип сообщения, **A** второй параметр – число символов сообщения.

При использовании косвенной адресации при обращении к устройствам, МКУ или функциям через параметры приходится применять числовые значения имен. Транслятор GPSS/PC на этапе компиляции сам присваивает именам числовые значения, однако, при этом пользователь не контролирует такое присвоение. Для того, чтобы не возникали конфликты между числовыми значениями имен, присвоенных транслятором, и именами, заданных пользователем, необходимо использовать простое правило: в модели использовать имена либо только числовые, либо только символьные.

Назначение именам числовых значений осуществляется с помощью оператора **EQU** (аналог **#define** в языке программирования Си).

Пример 4.48

Необходимо занять МКУ, номер которого определяется в первом параметре транзакта. По логике функционирования модели занимаетсяодин из двух МКУ: **SECTION_A** или **SECTION_B.**

```
SECTION_A   EQU         1
SECTION_B   EQU         2
SECTION_A   STORAGE     2
SECTION_B   STORAGE     3
            ...
            ASSIGN      1,1
            ...
            ASSSIGN     1,2
            ...
            ENTER       P1
            ...
```

Пример 4.49

Рассмотрим приход клиентов в банк и их обслуживание кассирами. Время прихода клиентов задается функцией распределения **ARRIVE.** Для обслуживания открыто семь окошек кассиров, к каждому из них может образовываться очередь. Во время прихода клиента в банк, если хотя бы один кассир свободен, клиент сразу же подходит к этому кассиру. В противном случае он присоединяется к любой кратчайшей очереди на текущий момент. Порядок обслуживания клиентов из очереди – FIFO.

Обслуживание в кассе разделяется на пять видов операций, частота появления которых и среднее время обслуживания задаются функцией **MEAN**.

В модели с несколькими очередями могут возникнуть одновременные события прихода клиента и конца обслуживания кассиром. Если событие «приход» будет рассмотрено ранее события «конец обслуживания», то устройство в момент проверки будет считаться занятым и вновь прибывший клиент не сможет занять кассира. Поэтому необходимо обеспечить приоритетность в обработке события «освобождения кассира». Это можно сделать с помощью задания уровня приоритета (блок **PRIORITY 1)** между блоками **GENERATE** и **ADVANCE**.

Для реализации поиска свободного кассира используется блок **TEST E**, помеченный меткой **NEXT**, который проверяет, свободен ли кассир. Максимальный номер кассира запоминается в параметре гранзакта **P5.** По этому же параметру образуется цикл для проверки всех устройств (блок **LOOP).** Номер первого свободного кассира запоминается в параметре транзакта **P3**, и транзакт передается на обслуживание. Если все устройства заняты, то транзакт переходит к следующему блоку после **LOOP.** т.е. к блоку **TEST**, помеченному меткой **SIT**. Этот блок сравнивает длины очередей для всех устройств. Первоначально для сравнения в параметр **P2** транзакта помещается число 1000. Аналогично организуется цикл по всем очередям с помощью параметра **P1** транзакта. Первоначально туда помещается максимальный номер очереди. Если длина следующей просматриваемой очереди меньше предыдущей, то запоминается номер очереди в параметре **P3** и ее длина в параметре **P2** транзакта. После просмотра всех очередей (перехода к следующему блоку после **LOOP)** в этих параметрах окажется номер минимальной очереди и ее длина, соответственно.

Приведем текст соответствующей программы.

```
*       Функция времени обслуживания
020     MEAN        FUNCTION    RN1,D5
1,450/.29,750/.61,1000/.85,1500/1.0,3000
*       Функция времени прихода
030     ARRIVEL     FUNCTION    RN1,D6
0.0,50/.25,100/.60,150/.80,200/.90,250/1.0,300
*
040     T_1         TABLE       M1,200,600,10   ; Сбор данных о времени
                                                ; пребывания
050     ********************************************************
055                 GENERATE    FN$ARRIVEL      ; Приход клиентов
060                 ASSIGN      1,7             ; Задать MAX номер очереди
065                 ASSIGN      2,1000          ; Запомнить большое число
070                 ASSIGN      4,FN$MEAN       ; Запомнить в P4
                                                ; время обслуживания
075                 ASSIGN      5,7             ; Задать MAX номер кассира
080     NEXT        TEST E      F*5,0,FAC       ; Есть свободный кассир?
085                 ASSIGN      3,P5            ; «Да» – запомнить его номер
090                 TRANSFER    ,QUI            ; Идти на обслуживание
95      FAC         LOOP        5,NEXT          ; Цикл по всем кассирам
100     ********************************************************
105     SIT         TEST L      Q*1,P2,SIS      ; Поиск MIN очереди
110                 ASSIGN      3,P1            ; Запомнить номер очереди
115                 ASSIGN      2,Q*1           ; Запомнить длину очереди
120     SIS         LOOP        1,SIT           ; Цикл по всем очередям
125     QUI         QUEUE       P3              ; Встать в очередь
130                 SEIZE       P3              ; Занять кассира
135                 DEPART      P3              ; Покинуть очередь
140                 PRIORITY    1
145                 ADVANCE     P4              ; Обслужиться
150                 RELEASE     P3              ; Освободить кассира
155                 TABULATE    T_1             ; Время пребывания
165                 TERMINATE
170     ********************************************************
175                 GENERATE    14400           ; Таймер времени работы
180                 TERMINATE   1
```

Отчет с результатами моделирования:

FACILITY	ENTRIES	UTIL.	AVE.TIME	AVAIL.	OWNER	PEND	INTER	RETRY	DELAY
1	10	0.854	1230.000	1	75	0	0	0	2
2	12	0.851	1020.833	1	78	0	0	0	1
3	7	0.931	1914.286	1	64	0	0	0	2
4	6	0.958	2300.000	1	59	0	0	0	2
5	11	0.938	1227.273	1	65	0	0	0	2
6	12	0.965	1158.333	1	62	0	0	0	2
7	13	0.983	1088.462	1	73	0	0	0	2

QUEUE	MAX	CONT.	ENTRY	ENTRY(0)	AVE.CONT.	AVE.TIME	AVE.(-0)	RETRY
1	2	2	12	5	0.417	500.000	857.143	0
2	2	1	13	6	0.524	580.769	1078.571	0
3	2	2	9	3	0.639	1022.222	1533.333	0
4	2	2	8	2	0.767	1381.250	1841.667	0
5	2	2	13	6	0.823	911.538	1692.857	0
6	2	2	14	4	0.865	889.286	1245.000	0
7	3	2	15	3	1.035	993.333	1241.667	0

Гистограмма времени пребывания в системе приведена на рис.4.12

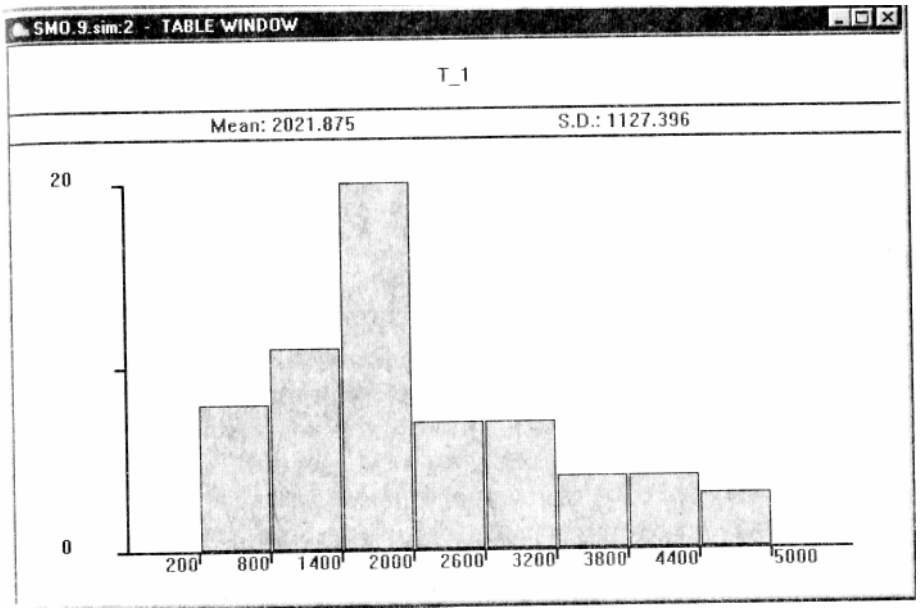


Рис. 4.12

Косвенная адресация является мощным инструментом языка GPSS, позволяющим существенно сократить размер модели и во многих случаях уложиться в ограничения для бесплатно распространяемых студенческих версий языка GPSS (в модели должно быть до 150 блоков).

4.21. Обработка транзактов, принадлежащих одному семейству

Кроме блока **GENERATE,** для создания транзактов может использоваться блок **SPLIT** (РАЗДЕЛИТЬ), который выполняет функцию копирования транзакта, входящего в него. Этот транзакт называется ***начальным*** или ***порождающим.*** Все копии формируются в момент входа начального транзакта в блок **SPLIT.** Каждая новая копия становится членом ***семейства (ансамбля)*** транзактов, порожденных одним начальным транзактом, который был создан блоком **GENERATE**

Блок имеет такой формат:

SPLIT A,|B|,[C]

Таблица 4.47

Операнд	*Значение*	*Результат по умолчанию*
A	Число создаваемых копий транзакта	Ошибка
B	Метка блока, куда направляются копии	
C	Параметр, в котором запоминаются номера копий транзактов	

Операнд **A** может быть положительным целым, СЧА, СЧА*СЧА. Если вычисленное значение операнда **A** равно нулю, то блок **SPLIT** не выполняет никаких операций. После создания копий начальный транзакт пытается перейти к очередному блоку.

Операнд **B** задает блок, в который переходят копии начального транзакта. Операнд может быть именем (меткой), положительным целым, СЧА, СЧА*СЧА (в трех последних случаях операнд **B** задает номер блока). Значение операнда **B** вычисляется для каждой копии отдельно.

Операнд **C** задает параметр транзакта. который используется для присвоения копиям последовательных номеров. Операнд **C** может быть именем, положительным целым, СЧА, СЧА*СЧА.

Тран.закты, принадлежащие одному семейству, объединяются интерпретатором в список. По связям внутри семейства транзактов невозможно установить, какой из транзактов семейства является начальным. Если копия транзакта входит в блок **SPLIT**, то повторная копия становится членом того же семейства, что и первичная копия. Таким образом, каждый транзакт является членом одного и только одного семейства. Семейство может состоять из произвольного числа транзактов. Когда транзакт уничтожается, интерпретатор автоматически исключает его из членов соответствующего семейства. Таким образом, семейство существует до тех пор, пока из модели не удалится последний из ее членов.

В модели одновременно может присутствовать произвольное число семейств, оно все время меняется, поскольку каждый транзакт, генерируемый блоком **GENERATE**, может создать свое семейство.

Пример 4.50

В цех каждые 14±3 мин поступают партии деталей. Каждая партия состоит из 3±2 деталей. Все детали поступают на обработку станком. Время обработки составляет 3±1 мин.

```
COPY  FUNCTION    RN1,C2
0,1/1,6
      GENERATE    14,3
      SPLIT       FNSCOPY
      SEIZE       MACHINE
      ADVANCE     3,1
      RELEASE     MACHINE
      TERMINATE   1
```

Для синхронизации движения транзактов, принадлежащих одному семейству, используются блоки **MATCH** (СОГЛАСОВАТЬ), **ASSEMBLE** (СОБРАТЬ), **GATHER** (СОЕДИНИТЬ).

Блок **MATCH** синхронизирует движение транзактов с другим блоком **MATCH.**

Формат блока:

MATCH A

Операнд **A** указывает имя сопряженного блока. Сопряженным блоком является также блок **MATCH**.

Пример 4.51

В локальной сети рабочая станция опрашивается каждые 30 мс. Если на рабочей станции есть сообщение для передачи, то оно занимает канал.

```
LABEL1    MATCH     LABEL2     ; Сообщение ждет опроса рабочей станции
          SEIZE     CHANNEL    ; Занять канал
          ...
LABEL2    MATCH     LABEL1     ; Опрос рабочей станции
          ADVANCE   30         ; Время до следующего опроса
```

При входе транзакта-сообщения в блок **MATCH** с меткой **LABEL1** он будет ждать (в списке синхронизации) момента, когда другой опросный транзакт, принадлежащий тому же семейству, не войдет в сопряженный блок **MATCH** с меткой **LABEL2**. Только после этого сообщение займет канал **CHANNEL**, а опросное сообщение перейдет в блок **ADVANCE**.

Блок **ASSEMBLE** собирает начальный транзакт и все транзакты-копии из одного семейства, удаляет копии и выдает один начальный транзакт. После сборки из блока **ASSEMBLE** выходит только один транзакт, который переходит в следующий по номеру блок. Формат блока:

ASSEMBLE A

Операнд **A** задает счетчик сборки, указывающий сколько членов одного семейства должны быть объединены. Операнд **A** может быть именем, положительным целым, СЧА, СЧА*СЧА. Первоначальное значение операнда **A** не должно быть меньше или равно единице.

Блок **GATHER** скапливает заданное количество транзактов, принадлежащих одному семейству. Он задерживает их до тех пор, пока не соберется необходимое число, указанное операндом **A**. Затем накопленные транзакты одновременно попытаются войти в следующий по номеру блок.

Формат блока:

GATHER A

Операнд **A** задает число транзактов, принадлежащих к одному семейству, которое нужно накопить. Операнд **A** может быть именем, положительным целым, СЧА, СЧА*СЧА.

Для управления транзактами, принадлежащими к одному семейству, используется блок **GATE**.

Пример 4.52 [20]

Некоторая фирма производит центробежные насосы, сборка которых осуществляется по заказу покупателей. Заказы прибывают в случайные моменты времени. Интервалы времени между поступлениями двух последовательных заказов распределены по нормальному закону с математическим ожиданием 19 мин и стандартным отклонением 3 мин.

Когда прибывает заказ, делается две его копии. Оригинал заказа используется для получения двигателя со склада и подготовки его для сборки. Время выполнения этой операции является экспоненциально распределенной случайной величиной со средним значением 8 мин. Первый экземпляр копии используется для заказа и адаптации насоса (время 10±2 мин), **A** второй экземпляр используется для начала изготовления плиты основания (время 15 мин).

Когда насос и плита основания готовы, производится пробная подгонка (время 5+1 мин). Все три компонента собираются вместе (время распределено по нормальному закону с математическим ожиданием 6 мин и стандартным отклонением 1 мин), когда они имеются налицо. Затем установка разбирается, насос и двигатель подвергаются окраске. Время покраски двигателя 2±0,5 мин, **A** время покраски насоса распределено по экспоненциальному закону со средним значением 1,5 мин. Плита основания гальванизируется 4 мин. После этого производится окончательная сборка. Время сборки – нормально распределенная случайная величина с математическим ожиданием 8 мин и стандартным отклонением 1 мин.

Промоделировать сборку 100 центробежных насосов и оценить среднее время их сборки, используя для этого таблицу.

Учитывая подробное описание самой модели и комментарии, приведенные в листинге прототипа программы, опишем кратко логику работы модели.

Транзакты имитируют заказы покупателей. Когда транзакт входит в блок **SPLIT**, создается еще два транзакта копии. Это позволяет одновременно продолжить выполнение индивидуальных заказов на мотор, насос и плиту основания.

Транзакты, имитирующие насос и плиту, ожидают друг друга в блоках **MATCH** с метками **PUMP** (насос) и **PLATE** (плита). Если и насос, и плита прибыли, то имитируется задержка на их начальную сборку. После того, как прибудут все три заказа в блок **GATHER**, блок **ADVANCE** имитирует пробную подгонку трех компонентов изделия друг к другу. Затем три заказа снова разделяются для окончательной отделки. Блок **ASSEMBLE** (сборка) с меткой **BUILD** вызывает отсрочку окончательной сборки, пока не поступят все компоненты.

В таблице **TRANSIT** собирается распределение времени выполнения заказов. Единица модельного времени 1 с.

Программа:

```
;                МОДЕЛИРОВАНИЕ ПРОЦЕССА СБОРКИ НАСОСОВ
10      XPDIS       FUNCTION    RN1,C24          ; Экспоненц. функция
                                                 ; распределения
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915/.7,1.2/.75,1.38
.8,1.6/.84,1.83/.88,2.12/.9,2.3/.92,2.52/.94,2.81/.95,2.99/.96,3.2
.97,3.5/.98,3.9/.99,4.6/.995,5.3/.998,6.2/.999,7/.9998,8
41      ***************************************************************
50      SNOR        FUNCTION    RN1,C25          ; Нормальная функция
                                                 ; распределения
0.0,-5/0.00003,-4/.00135,-3/.00621,-2.5/.02275,-2/.06681,-1.5
.11507,-1.2/.15866,-1/.21186,-.8/.27425,-.6/.34458,-.4/.42074,-.2
.5,0.0/.57926,.2/.65542,.4/.72575,.6/.78814,.8/.84134,1/.88493,1.2
.93319,1.5/.97125,2/.99379,2.5/.99865,3/.99997,4/1.0,5
91      ***************************************************************
100     TPL1        VARIABLE    360+60#FN$SNOR   ; Задание нормальных
105     TPL2        VARIABLE    480+60#FN$SNOR   ; законов распределений
110     T_ARRIVAL   VARIABLE    1140+180#FN$SNOR
111     ***************************************************************
115     TRANSIT     TABLE       M1,1200,600,20
116     ***************************************************************
120                 GENERATE    VST_ARRIVAL      ; Поступление заказа
125                 SPLIT       2,COPY,1         ; Создание двух копий
                                                 ; оригинала
126     ***************************************************************
127     ; Получение двигателя. В этом фрагменте проходит оригинал заказа
130                 SEIZE       MOTOR            ; Получение двигателя
135                 ADVANCE     480,FN$XPDIS     ; Подготовка двигателя
                                                 ; для сборки
140                 RELEASE     MOTOR            ; Освобождение устройства
145                 TRANSFER    ,TRY             ; Переход к пробной
                                                 ; сборке
146     ***************************************************************
147     ; Заказ и адаптация насоса. Этот фрагмент проходит копия 1 (p1=2)
150     COPY        TEST E      P1,2,BAS         ; p1=2?
155                 SEIZE       PUMPS            ; Получение насоса
160                 ADVANCE     600,120          ; Заказ и адаптация насоса
165     PUMP        MATCH       PLATE            ; Ожидание готовности
                                                 ; плиты основания
170                 ADVANCE     300,60           ; Пробная подгонка насоса
                                                 ; к плите основания
180                 RELEASE     PUMPS            ; Освободить устройство
190                 TRANSFER    ,TRY             ; Передать на пробную
                                                 ; сборку
191     ***************************************************************
195     BAS         SEIZE       BASE             ; Получить устройство
                                                 ; для изготовления плиты
200                 ADVANCE     900              ; Изготовление плиты
                                                 ; основания
205     PLATE       MATCH       PUMP             ; Ожидание готовности
                                                 ; насоса
210                 ADVANCE     300,60           ; Пробная подгонка насоса
                                                 ; к плите основания
220                 RELEASE     BASE             ; Освобождение устройства
221     ********** Пробная подгонка **********
225     TRY         GATHER      3                ; 3 части налицо
230                 ADVANCE     VSTPL1           ; Пробная подгонка
240                 TEST E      P1,1,FIN1        ; Это двигатель? (p1=1?)
241     ********** Покраска двигателя **********
245                 SEIZE       PAINT1           ; Занять 1-е устройство
                                                 ; покраски
250                 ADVANCE     120,30           ; Покраска двигателя
260                 RELEASE     PAINT1           ; Освободить 1-е
                                                 ; устройство покраски
265                 TRANSFER    ,BUILD           ; Отправить на
                                                 ;окончательную сборку
266     **********Покраска плиты основания **********
270     FIN1        TEST E      P1,2,FIN2        ; Это насос? (p1=2?)
280                 SEIZE       PAINT2           ; Получить 2-е устройство
                                                 ; покраски
```

```
285                 ADVANCE     90,FNSXPDIS     ; Покраска насоса
290                 RELEASE     PAINT2          ; Освободить 2-е
                                                ; устройство покраски
295                 TRANSFER    ,BUILD          ; Идти на окончательную
                                                ; сборку
296 ********** Гальванизация плиты основания **********
300 FIN2            SEIZE       GALVA           ; Занять устройство
                                                ; гальванизации
305                 ADVANCE     240             ; Гальванизация плиты
                                                ; основания
310                 RELEASE     GALVA           ; Освободить устройство
                                                ; гальванизации
311 ************************************************************
315 BUILD           ASSEMBLE    3               ; Ожидание готовности
                                                ; 3-х частей
320                 ADVANCE     VSTPL2          ; Сборка
325                 TABULATE    TRANSIT         ; Табулирование времени
                                                ; сборки
330 KON             TERMINATE   1               ; Выход готового изделия
335                 START       100             ; Сборка 100 насосов
```

В результате моделирования получена следующая статистика по устройствам.

FACILITY	ENTRIES	UTIL.	AVE.TIME	AVAIL.	OWNER	PEND	INTER	RETRY	DELAY
MOTOR	106	0.385	446.794	1	0	0	0	0	0
PUMPS	101	0.983	1198.410	1	303	0	0	0	5
BASE	101	0.984	1198.843	1	304	0	0	0	5
PAINT1	100	0.097	119.469	1	0	0	0	0	0
PAINT2	100	0.082	101.107	1	0	0	0	0	0
GALVA	100	0.195	240.000	1	0	0	0	0	0

Гистограмма времени сборки насосов показана на рис. 4.13.

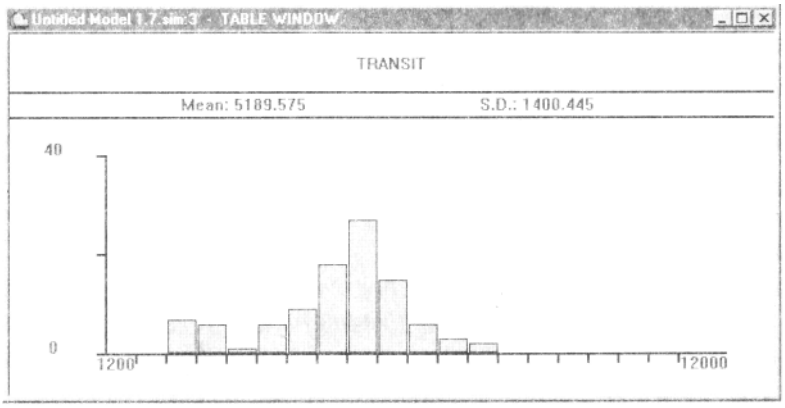


Рис. 4.13

4.22. Управление процессом моделирования в системе GPSS

В данной главе рассматриваются общие принципы управления моделированием в GPSS. Разные трансляторы по-разному могут реализовывать эти принципы.

В системе GPSS ***интерпретатор (программа управления моделированием*** (ПУМ)) поддерживает сложные структуры организации списков (рис. 4.14). С целью уменьшения затрат компьютерного времени на просмотр списков система GPSS ведет два основных списка событий. Первым является ***список текущих событии*** (СТС), куда входят все события, запланированные на текущий момент модельного времени независимо от того, условные они или безусловные. Программа управления моделированием просматривает в первую очередь этот список и пытается переместить по модели те транзакты, для которых выполнены условия. Если в этом списке таких транзактов нет, то ПУМ обращается к другому списку – ***списку будущих событий*** (СБС). Она переносит все события, которые запланированы на ближайший момент модельного времени, из этого списка в СТС и повторяет его просмотр. Такой перенос осуществляется также в случае совпадения текущего времени моделирования со временем первого события в списке будущих событий.

В СТС транзакты размещены в порядке уменьшения приоритета (то есть транзакты с более высоким приоритетом размещены ближе к началу списка). Транзакты с одинаковыми приоритетами размещаются в соответствии с последовательностью поступления в список. Каждый транзакт в СТС может находиться или в ***активном состоянии*** (то есть просматриваться ПУМ в данный момент модельного времени), или в ***состоянии задержки.***

В начальный момент (при выполнении оператора управления **START,** который начинает фазу интерпретации GPSS-модели) ПУМ обращается ко всем блокам **GENERATE** модели. Каждый из этих блоков планирует момент появления транзактов и заносит их в СБС, после чего ПУМ обращается к СТС. Так как в этом списке пока что отсутствуют транзакты, то ПУМ просматривает СБС и выбирает из него все транзакты, запланированные на ближайший момент времени и переносит их в СТС, после чего пытается продвинуть первый транзакт этого списка по блокам модели. Если перемещение транзакта было задержано по какой-либо причине, не связанной с блоком **ADVANCE**, то он остается в СТС и ПУМ пробует перемещать такой транзакт из этого списка далее по блокам. Если транзакт вошел в блок **ADVANCE**, то планируется его выход из этого блока и транзакт переносится в СБС.

Списки текущих и будущих событий можно увидеть на экране дисплея, если выдать команду **EVENTS** для GPSS/PC или в окне списков для GPSS World.

В организации эффективной процедуры просмотра важен порядок просмотра транзактов, движение которых заблокировано (например, ввиду занятости некоторого ресурса). Простейшим решением является пересмотр всех заблокированных транзактов для каждого нового значения модельного времени и выбор тех, для которых сняты условия блокировок. Если моделируемая система перегружена, то этот способ с точки зрения затрат компьютерного времени невыгоден, так как каждый транзакт пересматривается многократно до того, как выйдет из состояния блокирования.

Если причина перевода транзактов в состояние блокирования – состояние определенного ресурса системы в данный момент времени, то намного более эффективным является способ обработки, по которому заблокированные по этой причине транзакты вообще не пересматриваются до тех пор, пока не изменится состояние соответствующего ресурса. Такой способ может быть реализован, например, регистрацией для каждой единицы ресурса транзактов, движение которых заблокировано ввиду состояния именно этого ресурса в данный момент времени. В момент, когда изменяется состояние этого ресурса, необходимо сразу же просмотреть транзакты, которые этого ожидают, и продолжить их обработку.

Если транзакты находятся в активном состоянии, то процедура просмотра пытается переместить их к следующим блокам. Если перемещение транзакта блокируется каким-нибудь ресурсом ввиду его занятости, то вхождение в следующий блок невозможно и транзакт переводится в состояние задержки. Такие транзакты не просматриваются и размещаются в соответствующем списке задержки.

Если при обслуживании текущего активного транзакта произошло изменение состояния ресурса, пересмотр начинается сначала, и опять обслуживаются все транзакты из СТС, которые находятся в активном состоянии. Если изменение списков ресурсов не произошло, ПУМ опять обращается к СТС и проверяет, не остались ли в нем транзакты, которые необходимо обработать.

Список блокировок – это список транзактов, которые ожидают) изменения состояния ресурса. Существует шесть видов таких списков, связанных с устройствами; семь видов, связанных с МКУ, и два вида, связанных с логическими ключами. С устройствами используются списки для занятых и незанятых, доступных и недоступных устройств и устройств, работающих без прерываний и с прерываниями. С МКУ используются списки для заполненного, незаполненного, пустого, непустого, доступного, недоступного МКУ и транзактов, которые могут войти в МКУ. С логическими ключами связаны списки для включенных и выключенных ключей.

Кроме двух основных списков СТС и СБС существует ***список прерывании*** (СПР), содержащий прерванные во время обслуживания транзакты. **А** также транзакты, вызвавшие прерывание. Список прерываний используется для организации обслуживания одноканальных устройств по абсолютным приоритетам. Это дает возможность организовать приоритетные дисциплины обслуживания транзактов в устройствах.

Список синхронизации (СС) содержит транзакты. которые на данный момент времени сравниваются. Этот список работает с транзактами, полученными с помощью блока **SPLIT,** который создает транзакты-копии, принадлежащие одному семейству или ансамблю. Синхронизацию движения транзактов одного семейства выполняют следующие блоки: **MATCH** (синхронизирует движение транзактов с другим блоком), **ASSEMBLE** (собирает все транзакты-копии и выдает один начальный

транзакт), **GATHER** (собирает заданное количество транзактов и задерживает копии до тех пор, пока не соберется необходимое количество копий транзактов). Блок **SPLIT** можно использовать многократно.

Остановленные процессы находятся в СБС, СС и списках блокировок.

Список пользователя (СП) содержит транзакты, выведенные пользователем из СТС с помощью блока **LINK** и помещенные в СП как временно неактивные (переведенные пользователем в ***пассивное состояние).*** При работе ПУМ они ей недоступны до тех пор, пока не будут возвращены пользователем в СТС с помощью блока **UNLINK.**

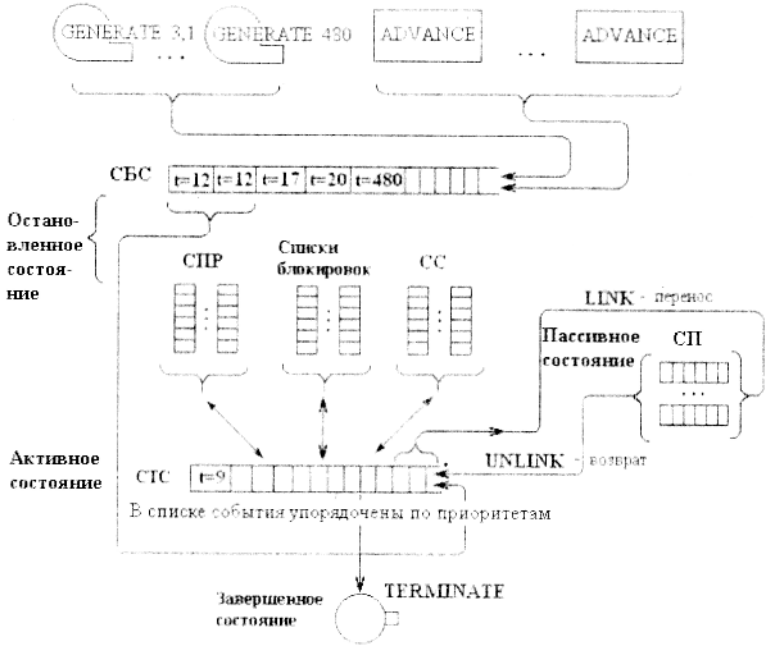


Рис. 4.14

Моделирование заканчивается тогда, когда счетчик завершения, инициализированный оператором управления **START,** будет сброшен в ноль или когда в СТС и СБС не будет ни одного транзакта.

4.23. Списки пользователей

Блок LINK (ВНЕСТИ В СПИСОК). Блок LINK собирает транзакты из СТС и помещает их в СП. Таким образом, интерпретатор их **Е** просматривает и не перемещает по блокам модели до тех пор, пока пользователь не возвратит их в модель. Формат блока:

LINK A,B[,C]

Операнд **А** задает номер или имя СП, в который будет помещен транзакт. Операнд **А** может быть положительным целым, именем, СЧА,СЧА*СЧА. Операнд **В** задает алгоритм упорядочивания СП.

Операнд **В** может быть LIFO, FIFO, целым, СЧА, СЧА*СЧА.

Допустимые значения операнда **В**:

FIFO – вошедший транзакт помещается в конец СП;

LIFO – вошедший транзакт помещается в начало СП;

номер параметра **–** входящие в СП транзакты располагаются в соответствии со значением указанного параметра;

PR – приоритет транзакта (транзакт помещается в список в соответствии с приоритетом);

M1 – время нахождения транзакта в модели;

Операнд **С** указывает альтернативный выход, который используется при описании разных ситуаций, возникающих в очередях. Операнд **С** может быть именем, положительным целым, СЧА, СЧА*СЧА.

Если операнд **С** не задан, индикатор, связанный с заданным СП, устанавливается в положение «1». Это приводит к тому, что все транзакты, безусловно входящие в блок, заносятся в СП, определенный операндом **А**, в порядке, который задан операндом **В**.

Если операнд **C** задан, проверяется индикатор СП. Если индикатор списка установлен в положение «1», вошедший транзакт, заносится в СП в порядке, заданном операндом **B**. Если же индикатор списка установлен в положение «0», он переводится в положение «3», и вошедший транзакт перемещается к блоку, заданному в операнде **C**.

Пример 4.53

LINK LIST,F IFO

В этом примере транзакт, вошедший в блок, помещается в конец списка с именем **LIST.**

Стандартные числовые атрибуты, связанные с блоком LINK:
CA<номер списка>, CA$<имя списка> – среднее число транзактов в СП;
CC<номер списка>, CC$<имя списка> – общее число транзактов в СП;
CH<номер списка>, CH$<имя списка> – – текущее число транзата в СП;
CM<номер списка>, CM$<имя списка> – максимальное число транзактов в СП;
CT<номер списка>, CT$<имя списка> – среднее время пребывания транзакта в СП.

Блок UNLINK (ВЫВЕСТИ ИЗ СПИСКА). Блок **UNLINK** удаляет транзакты из СП. После этого интерпретатор GPSS возобновляет их движение по модели.

Формат блока:

UNLINK (X| A,B,[C|,[D],[E],[F]

Операторы отношения, которые записываются во вспомогательном операнде **X**, определяют, какое условие (отношение) будет рассматриваться. Если этот оператор не задан, предполагается отношение равенства **E**.

Операторы отношения могут быть такими:

G ***(больше)*** **–** отношение истинно, если значение параметра, заданного в операнде **D**, больше значения, заданного в операнде **E**;

GE ***(больше или равно)*** **–** отношение истинно, если значение параметра, заданного в операнде **D,** больше значения, заданного в операнде **E**, или равно ему;

L ***(меньше)*** **–** отношение истинно, если значение параметра, заданного в операнде **D,** меньше значения, заданного в операнде **E**;

LE ***(меньше или равно)*** **–** отношение истинно, если значение параметра, заданного в операнде **D**, меньше значения, заданного в операнде **E**, или равно ему;

E ***(равно)*** – отношение истинно, если значение параметра, заданного в операнде **D**, равно значению, заданному в операнде **E**;

NE ***(не равно)*** **–** отношение истинно, если значение параметра, заданного в операнде **D**, не равно значению, заданному в операнде **Ё**.

Операнд **A** задает СП, из которого удаляются один или несколько транзактов. Операнд **A** может быть именем, положительным целым, СЧА или СЧА*СЧА.

В операнде **B** указывается номер блока, к которому переходят удаляемые из списка транзакты. Операнд **B** может быть именем, положительным целым, СЧА, СЧА*СЧА. Операнд **C** задает число транзактов, удаляемых из СП (счетчик удалений). Операнд **C** может быть именем, положительным целым, СЧА, СЧА*СЧА или «**ALL**» (означает удаление всех транзактов).

Операнд **D** может быть именем, целым, СЧА, СЧА*СЧА, ; «ВАСК». Действия, выполняемые при вхождении транзакта в блок , **UNLINK**, зависят от того, на что ссылается операнд **D**. В операнде **D** могут быть указаны номер параметра, булева переменная или слово «**ВАСК**».

Номер параметра. Если операнд **E** пропущен, значение заданного параметра вошедшего транзакта сравнивается со значением этого же параметра транзактов СП. Если **E** не пропущен, значение заданного параметра транзактов СП сравнивается со значением СЧА из операнда **E**. В обоих случаях транзакты, удовлетворяющие заданному отношению, будут удалены из списка и направлены в блок, указанный в операнде **B.**

Булева переменная **BVj** вычисляется отдельно для каждого транзакта из СП. Если для транзакта значение **BVj**=l, то он удаляется из СП (количество удаляемых транзакгов не может превышать значения операнда **C**). Если **BVj**=0 для всех транзактов списка, то вошедший транзакт пытается

переместиться в блок, заданный в операнде **F**. Если операнд **F** пропущен, транзакт пытается перейти в следующий по номеру блок.

Если в операнде **D** задана булева переменная, операнд **E** должен быть пустым. Если булева переменная **BVj** имеет ссылку на какой-либо параметр, то эта ссылка относится к параметрам транзактов из списка, **A** не к входящему в блок **UNLINK** транзакту.

Слово **«BACK»**. Из указанного списка, начиная с его конца, будет исключено столько транзактов, сколько задано операндом **C**. Операнд **E** в этом случае должен быть пустым.

Операнд **E** содержит СЧА, значение которого сравнивается со значением параметра транзактов СП (номер параметра указан в операнде D). Операнд **E** может быть именем, целым, СЧА, СЧА*СЧА.

Операнд **F** задает номер следующего блока длятого транзакта, который входит в блок **UNLINK** в случаях, когда соответствующий СП пустой или не выполнено заданное отношение, или же указанная в операнде **D** булева переменная равна нулю для всех транзактов списка (т.е. в случае, когда из СП нельзя ничего удалить). Операнд **F** может быть именем, положительным целым, СЧА, СЧА*СЧА.

Пример 4.54

```
UNLINK      LIST,FORW,1
```

Первый транзакт из СП с именем **LIST** помещается в блок с меткой **FORW**. Он заносится в СТС после транзактов с таким же приоритетом. Транзакт, вошедший в блок **UNLINK**, переходит в следующий блок.

Рассмотрим последовательность операций, выполняемых при входе транзакта в блок **UNLINK**. В зависимости от того, какие из операндов **A-F** блока **UNLINK** заполнены, можно выделить восемь основных вариантов.

Вариант 1. Операнды **A, B, C** заданы, операнды **D, E, F** не заданы. Вычисляются значения операнда **A** для определения номера (имени) СП. Проверяется, есть ли в списке транзакты. Если их нет, соответствующий этому списку индикатор устанавливается в «0», **A** транзакт, вошедший в блок, переходит к следующему по номеру блоку.

Если список не пуст, вычисляется значение операнда C (счетчика удалений), определяющего число транзактов, удаляемых из списка. Транзакты удаляются, начиная с первого в списке до тех пор, пока значение счетчика удалений не станет равным нулю или пока не будут исчерпаны все транзакты из списка. Удаленные из СП транзакты будут помещены в СТС и направлены к блоку, номер которого указан в операнде **B**. Транзакт, вошедший в блок **UNLINK**, перемещается к следующему по номеру блоку.

Вариант 2. Операнды **A, B, C** и **F** заполнены, операнды **D** и **E** не заданы. Этот вариант аналогичен варианту 1, за исключением случая, когда указанный СП пуст. При этом индикатор списка устанавливается в «0», **A** транзакт, вошедший в блок **UNLINK**, перемещается к блоку, указанному в операнде **F**, **A** не к следующему по номеру блоку.

Вариант 3. Операнды **A, B, C** и **D** заполнены, операнды **E** и **F** не заполнены, в операнде **D** задано значение параметра **Pj**. Этот вариант гакже аналогичен варианту 1. Однако из списка удаляются только те фанзакты, у которых значение параметра **Pj** равно значению этого же параметра **Pj** транзакта, вошедшего в блок **UNLINK**. Транзакты из списка удаляются до тех пор, пока значение счетчика удалений (определяемого операндом **C**) не станет равно нулю или пока не будут просмотрены все транзакты из списка. Все удаленные из СП транзакты перемешаются к блоку, указанному в операнде **B** блока **UNLINK.** Транзакт, вошедший в блок **UNLINK,** перемещается к следующему по номеру блоку.

Вариант 4. Операнды **A, B, C** и **D** заполнены, операнды **E** и **F** не заданы, **A** в операнде **D** записано слово «**BACK**». Этот вариант аналогичен варианту 1, за исключением того, что транзакты удаляются, начиная с конца списка.

Вариант 5. Операнды **A, B, C** и **D** – заполнены, операнды **E** и **F** – пусты, в операнде **D** задана булева переменная **BVj**. Этот вариант аналогичен варианту 3, за исключением того, что **BVj** вычисляется отдельно для каждого транзакта СП. Из СП удаляются только транзакты, для которых **BVj=l.**

1. Если **BVj** имеет ссылки на какой-либо параметр, то эти ссылки относятся к параметрам, связанным с транзактами из СП, **A** не с вошедшим транзактом.

2. В случаях, когда в операнде **D** блока **UNLINK** записана переменная **BVj** или «**BACK**», операнд **E** должен быть не заполненным, иначе – ошибка.

Вариант 6. Операнды **A, B, C, D** и **F** заполнены, операнд **E** не заполнен. Этот вариант аналогичен варианту 3, за исключением тех случаев, когда или заданный СП пуст (**CHj**=0), или заданное отношение не выполняется ни для одного транзакта списка, или **BVj**=0 для всех транзактов СП. В таких случаях транзакт, вошедший в блок **UNLINK,** перемещается к блоку, номер которого задан в операнде **F, A** не к следующему по номеру блоку. Индикатор списка устанавливается в «0» только в том случае, если список пуст.

Вариант 7. Операнды **A, B, C, D** и **E** заполнены, операнд **F** не заполнен.

Число удаляемых транзактов определяется значением операнда **C**. Номер блока, к которому направляются удаленные транзакты, определяется операндом **B**. Транзакты, для которых значение **Pj** (операнд **D)** равно СЧА из операнда **E**, удаляются из списка. Транзакты просматриваются и удаляются (если это возможно), начиная с начала списка. Удаление продолжается до тех пор, пока значение счетчика удалений (операнд **C**) не станет равным нулю или пока из СП не будут удалены все транзакты. Транзакт, вошедший в блок **UNLINK**, перемещается к следующему по номеру блоку.

Вариант 8. Операнды **A, B, C, D** и **F** заполнены. Этот вариант аналогичен варианту 5, за исключением тех случаев, когда СП пуст (то есть **CHj**=0) или заданное отношение не выполняется ни для одного из транзактов списка. Транзакт, вошедший в блок UNLINK, переходит к блоку, заданному в операнде F, **A** не к следующему по номеру блоку. Если заданное отношение выполняется для какого-нибудь транзакта из списка, транзакт, вошедший в блок, перемещается к следующему по номеру блоку. Если СП пустой, индикатор списка устанавливается в «0».

Пример4.55[13]

Рассмотрим модель обслуживания с алгоритмом FIFO выбора из очереди:

```
        GENERATE    1000,FN1
        QUEUE       QUE1
        LINK        CHAIN,FIFO,CAN
CAN     SEIZE       FC_1
        DEPART      QUE1
        ADVANCE     700,FN1
        RELEASE     FC_1
        UNLINK      CHAIN,CAN,1
        TERMINATE
```

Транзакты, выходящие из блока **GENERATE,** поступают в блок **QUEUE.** После внесения соответствующих изменений в статистику, собираемую по данной очереди, транзакт входит в блок **LINK.** Так как в блоке **LINK** существует альтернативный выход, проверяется состояние индикатора списка **CHAIN** (имя которого определятся операндом **A** блока **LINK).** Если индикатор этого списка установлен в «0», транзакт немедленно устанавливает его в «1» и переходит к альтернативному блоку, то есть в данной модели – к блоку **SEIZE**. Если индикатор списка установлен в «1», то транзакт, входящий в блок **LINK,** без дальнейших проверок заносится в СП **CHAIN.** Так как в операнде **B** задан алгоритм **FIFO,** транзакт помещается в юнец этого СП.

Отметим, что транзакты, занесенные в СП, не проходят дальнейшей обработки до тех пор, пока они не будут удалены из списка другими транзактами, входящими в блок **UNLINK.** После удаления из СП транзакт поступает к блоку, указанному в операнде **B** блока **UNLINK**.

Когда транзакт входит в блок **SEIZE**, проводится корректировка статистики, связанной с заданным в этом блоке устройством. Потом транзакт переходит к блоку **DEPART**, где корректируется статистика очереди, **A** потом переходит к блоку **ADVANCE**, в котором он остается на протяжении времени, определяемом операндами блока **ADVANCE**.

После выхода из блока **ADVANCE** транзакт входит в блок **RELEASE**. Освобождается указанное в блоке устройство и корректируется связанная с ним статистика. После этого транзакт перемещается к блоку **UNLINK**.

При перемещении транзакта к блоку **UNLINK** просматривается СП, имя которого задано в операнде **A**:

1) если СП пуст, то есть **CH$CHAIN** = 0, то индикатор списка устанавливается в «0» и транзакт переходит к следующему по номеру блоку, то есть к **TERMINATE**;

2) если СП не пуст, то есть **CH$CHAIN** ≠ 0, то первый транзакт этого списка удаляется, помещается в СТС и направляется к блоку, указанному в операнде **B** блока **UNLINK** (в данной модели это блок **SEIZE**, помеченный меткой **CAN**). А транзакт, вошедший в блок **UNLINK**, перемещается к блоку **TERMINATE**.

Рассмотренная модель не сложна, но хорошо иллюстрирует несколько важных моментов:

1) в этой системе активными могут быть только те транзакты, которые выходят из блока **GENERATE**, или тот транзакт, который в данный момент занимает устройство. Все остальные транзакты находятся в СП **CHAIN**;

2) так как все задержанные транзакты, то есть транзакты, находящиеся в очереди к устройству **FC_1**, будут помещены в СП **CHAIN**, интерпретатор не будет тратить время на изменение индикаторов задержки всех этих транзактов при каждом изменении положения устройства. Экономия времени зависит от длины очереди: чем длиннее очередь, тем больше времени будет сэкономлено благодаря блокам **LINK – UNLINK**, которые используются для управления очередями к разным .объектам;

3) пользователь имеет возможность динамически формировать свои списки независимо от списков задержки, которые поддерживаются системой GPSS;

4) использование блоков **LINK-UNLINK** дает возможность синхронизировать движение разных транзактов в модели, например, задерживать в списке транзакты до тех пор, пока какой-либо другой транзакт не выведет их из списка с помощью блока **UNLINK**.

Пример 4.56

Рассмотрим работу мультиплексора (см. главу 5), который подключен к высокоскоростному каналу (ВК) связи и работает в режиме разделения времени с четырьмя низкоскоростными каналами (НК), опрашивая их циклически. На один опрос каждого из НК мультиплексор тратит 50 мс, время переключения между НК – 10 мс. Если в опрашиваемом НК есть сообщение, оно передается по ВК в течение 500+60 мс. За один опрос передается одно сообщение. Время возникновения сообщений в четырех неравномерно распределено в интервале 1500±500 мс, и сообщения равновероятно могут появляться на любом канале.

Необходимо определить загрузку ВК при передаче по нему 1000 сообщений.

Программа:

```
*  За основу взята программа из книги Дж. Мартина
*  «Системный анализ передачи данных», т. 2
NUMBKS      FUNCTION    RN1,D4                  ; Определение номера НК, в
.25,1/.5,2/.75,3/.9999,4                        ; котором появилось сообщение
;
OPROS       FUNCTION    P1,D4   ;Функция, задающая
1,2/2,3/3,4/4,1                 ; последовательность опроса НК

*       Опрос каналов
            GENERATE    10,,,1  ; Генерация сообщений, опрашивающих каналы
            ASSIGN      1,4                     ; В первом параметре фиксируется
                                                ; номер последнего
                                                ; опрошенного НК
NEXT        ASSIGN      1,FN$OPROS              ; Номер опрашиваемого НК
            SEIZE       VK                      ; Занятие ВК
            ADVANCE     50                      ; Время опроса
            TEST NE     CH*1,0,MREL             ; Есть ли сообщения в НК?
            UNLINK      P1,MESS,1               ; Разрешить передачу по ВК
MREL        RELEASE     VK                      ; Освободить ВК
            ADVANCE     10                      ; Время переключения
            TRANSFER    ,NEXT                   ; Продолжить опрос НК
*       Передача сообщений по каналам
```

```
                GENERATE    1500,500        ; Генерация сообщений
                ASSIGN      1,FN$NUMBKS     ; Номер НК, в котором
                                            ; появилось сообщение
                LINK        P1,FIFO         ; Ожидать опроса канала
MESS            SEIZE       VK              ; Занять ВК для передачи
                ADVANCE     500,60          ; Передача сообщений
                RELEASE     MPD             ; Освободить ВК
                TERMINATE   1
                START       1000            ; Передать 1000 сообщений
```

```
10      EXP     FUNCTION    RN1,C24
0,0/.100,.104/.200,.222/.300,.355/.400,.509
.500,.690/.600,.915/.700,1.200/.750,1.380
.800,1.600/.840,1.830/.880,2.120/.900,2.300
.920,2.520/.940,2.810/.950,2.990/.960,3.200
.970,3.500/.980,3.900/.990,4.600/.995,5.300
.998,6.200/.999,7/1,8
;
20              GENERATE    55,FN$EXP
30              TEST L      CH$SP1,4,TER1       ; В списке SP1 меньше
                                                ; четырех транзактов?
                                                ; Нет – покинуть систему.
40              QUEUE       OCH1                ; Да – стать в очередь.
50              LINK        SP1,LIFO,MUS1
60      MUS1    SEIZE       USTR1
70              DEPART      OCH1
80              ADVANCE     30
90              RELEASE     USTR1
100             UNLINK      SP1,MUS1
110             ASSIGN      TOCH2,C1            ; Запомнить время прихода
120             ASSIGN      TOCH2+,40           ; Добавить 40 с
130             LINK        SP2,LIFO,MUS2
140     MUS2    SEIZE       USTR2
150             ADVANCE     20,5
160             RELEASE     USTR2
170             UNLINK LE   SP2,TER1,ALL,TOCH2,C1   ; Время ≤ 40 с?
```

Пример 4.57

На вход СМО поступает пуассоновский поток заявок со средним временем *55* с, которые последовательно обслуживаются двумя устройствами. Порядок выбора требований из очереди для обслуживания первым и вторым устройствами – LIFO. Заявки покидают систему не обслужившись, если на момент входа заявки в первую очередь в ней есть больше четырех заявок или время нахождения во второй очереди превышает 40 с. Время обслуживания заявки первым устройством составляет 30 с, вторым – 20 ± *5 с*. Необходимо смоделировать прохождение через СМО 100 заявок.

Программа:

```
180             UNLINK      SP2,MUS2,1          ; Идти на обслуживание
190     TER1    TERMINATE   1
200             START 100
```

В модели значения индикаторов списков пользователя позволяют определить состояние устройств обслуживания. Если в момент входа в блок **LINK** индикатор списка включен («1»), это означает, что соответствующее устройство занято и транзакт становится в начало списка **SP1** или **SP2** (LIFO), индикатор остается включенным. Если индикатор списка выключен («0» – соответствующее устройство свободно) – он включается, **A** транзакт переходит на обслуживание в блок **MUS1** или MUS2. В параметре с именем **TOCH** транзакта запоминается момент времени, к которому транзакт должен попасть на обслуживание устройством **USTR2,** в противном случае транзакт будет удален из модели блоком **UNLINK** (строка 170).

Пример 4.58

Рассмотрим модель эвакуации потерпевших и раненых во время боевых действий. Потерпевшие эвакуируются 5 – и 13-местными автомобилями. Если на пункте эвакуации собирается раненых больше чем мест в автомобиле, то вывозится столько раненых, сколько есть свободных мест. Иначе вывозятся все. Известными являются функции времени прибытия раненых к медицинскому пункту (МП) полка. В каждом полку есть четыре МП. Раненые эвакуируются в военно-полевой передвижной госпиталь (ВППГ), где им предоставляется необходимая помощь.

Фрагмент программы:

```
;       Author: V. Tomashevskiy
TPOLK       FUNCTION    ...             ; Функция распределения интервалов
            ...                         ; времени от момента ранения до
                                        ; подготовки в МП полка
                                        ; для следующей эвакуации
;
EXP1        FUNCTION    RN1,C21         ; Функция распределения
0,/. 0078,0/. 2556,2/. 4544,4/. 7081,6/. 8023,8     ; интервалов времени прибытия
8694,10/. 9151,12/. 9418,14/. 9625,16/. 9763,18     ; раненых в МП
9859,20/. 9896,22/. 9933,24/. 9955,26/. 9963,28
9974,30/. 9981,32/. 9989,34/. 9996,36/1,96
;
MEST_VP     FUNCTION    X$CARVP,D2      ; Эксплуатируется 9 автомобилей
9,5/18,13                               ; пятиместных и 9 – 13-местных
;
            GENERATE    ,,30,18         ; Эвакуация потерпевших
                                        ; 18 автомобилями. Момент выезда –
                                        ; через 30 мин после начала боя
            SAVEVALUE   CARVP+,1        ; Определение номера автомобиля
;
            ASSIGN      MESTOVP,FN$MEST_VP      ; Определение количества
                                                ; мест в автомобиле
VOZVR       ADVANCE     120,20          ; Длительность рейса к полку
EV          TEST LE     CH$EVAC_POL,P$MESTOVP,BOLVP
;       Если потерпевших меньше, чем мест в автомобиле, то эвакуируют всех
            UNLINK      EVAC_POL,VPPG,CH$EVAC_POL
            TRANSFER    ,VOZVR          ; Новый рейс к МП полка
BOLVP       UNLINK      EVAC_POL,VPPG,P$MESTOVP
;       Эвакуируют потерпевших в количестве, не превышающем числа мест в автомобиле
            TRANSFER    ,VOZVR          ; Новый рейс к МП полка
;       Эвакуация потерпевших с первого МП полка
            GENERATE    FN$EXP1,,100    ; Поступление потерпевших в МП полка:
                                        ; первый поступает не раньше, чем через
                                        ; 100 мин после начала боя
            SAVEVALUE   SV_P1+,1        ; Подсчет санитарных потерь
            TRANSFER    ,NAK            ; Направить на эвакуацию
;       Эвакуация потерпевших со второго МП полка
            GENERATE    8,FN$EXP,100
            ASSIGN      1,2
            ASSIGN      TRAN,FN$TPOLK
            SAVEVALUE   SV_P2+,1
            TRANSFER    ,NAK
;       Эвакуация потерпевших с третьего МП полка
            GENERATE    10,FN$EXP,100
            ASSIGN      1,3
            ASSIGN      TRAN,FN$TPOLK
            SAVEVALUE   SV_P3+,1
            TRANSFER    ,NAK
;       Эвакуация потерпевших с четвертого МП полка
            GENERATE    10,FN$EXP,100
            ASSIGN      1,4
            ASSIGN      TRAN,FN$TPOLK
            SAVEVALUE   SV_P4+,1
NAK         LINK        EVAC_POL,FIFO
;       Ожидание эвакуации из МП полка
VPPG        ADVANCE     60,10           ; Эвакуация в ВППГ
;       Прибытие медицинского транспорта на пост сортировки ВППГ
            ADVANCE     2,1             ; Сортировка в ВППГ
```

4.24. Блоки управления потоками транзактов LOGIC, GATE LR, GATE LS и GATE

Логические ключи (блок LOGIC). Логические ключи используются для моделирования объектов, имеющих всего два положения: «Включен» ***(set*** или 1) и «Выключен» ***(reset*** или 0).

Блок **LOGIC** используется для включения, выключения или инвертирования положения ключа. Положение ключа можно проверить любым транзактом в любой части модели.

Блок **LOGIC** имеет такой формат:

LOGIC X A

Операнд **A** – номер логического ключа; он может быть именем, положительным целым, СЧА или СЧА*СЧА.

Когда транзакт входит в блок **LOGIC,** положение логического ключа, номер которого задан в операнде **A**, изменяется в зависимости от значения вспомогательного оператора **X** следующим образом:

S – логический ключ устанавливается в положение «Включен»;

R – логический ключ устанавливается в положение «Выключен»;

I – логический ключ инвертируется, то есть положение его изменяется на противоположное.

Логический ключ имеет СЧА **LS<номер ключа>** или **LS$<HMH ключа>,** который возвращает значение 1, если ключ в положении «Включен», и 0 – если в положении «Выключен».

Для изменения направления движения сообщений в зависимости от положения логических ключей используются блоки **GATE LR** и **GATE LS**

Блоки GATE LR и GATE LS. Блоки **GATE LR, GATE LS** проверяют положение логического ключа. В операнде В задается номер блока, к которому переходит транзакт, если вспомогательный оператор **X** имеет значение «ложь». Если значение логического оператора – «истина», транзакт переходит к следующему по порядку блоку. Если операнд **B** пустой, блок **GATE LR** (**GATE LS**) работает в режиме условного вхождения, если заполнен – в режиме безусловного вхождения.

В *режиме условного вхождения* транзакты могут войти в блок **GATE** только в том случае, если логический оператор **(LRj** или **LSj)** имеет значение «истина». Если значение логического оператора – «ложь», транзакт помещается в список задержки и не обрабатывается интерпретатором до тех пор, пока значение не станет истинным. Пдинственным исключением являются транзакты, находящиеся в блоке **TRANSFER BOTH** (или **ALL**). Когда потом какой-нибудь другой транзакт проходит блок **LOGIC,** изменяющий состояние соответствующего ключа, и присваивает указанному в блоке **GATE** логическому оператору значение «истина», все транзакты, находящиеся в списке задержки, активизируются. После этого интерпретатор, просматривая СТС, получает возможность переместить один или несколько транзактов (включая и те, что находятся в блоке **TRANSFER BOTH** или **ALL)** в блок **GATE LR (GATE LS**), работающий в режиме условного вхождения.

Пример 4.59

Рассмотрим работу телефонной сети, имеющей 50 абонентских линий связи, причем одновременно может быть задействовано не более 10 связей между абонентами. Каждый абонент может соединиться с остальными, если свободны его входная линия связи и входная линия вызываемого абонента. Из 50 линий для организации связи могут использоваться любые две свободных линии. Необходимо промоделировать работу телефонной сети для 1000 вызовов. Интервалы между вызовами и длительность разговора распределены по экспоненциальному закону. Предусматривается, что первые 15 вызовов образуют переходной процесс в сети и эти данные не нужно учитывать при моделировании. Модель этой системы разработал Джеффри Гордон для GPSS/H. Приведенный текст программы модифицирован для GPSS World.

Программа:

```
*Origin: System simulation
*Author: Geoffrey Gordon
*Description: Simulation of a telephone system
*
*       SIMULATION OF A TELEPHONE SYSTEM - MODEL 2 (president line)
*
1       POISS           FUNCTION    RN1,C24             ; Функция интервалов
                                                        ; между вызовами
0.0,0.0/0.1,0.104/0.2,0.222/0.3,0.355/0.4,0.509/0.5,0.69
0.6,0.915/0.7,1.2/0.75,1.38/0.8,1.6/0.84,1.83/0.88,7.12
0.9,2.3/0.92,2.52/0.94,2.81/0.95,2.99/0.96,3.2/0.97,3.5
0.98,3.9/0.99,4.6/0.995,5.3/0.998,6.2/0.999,7/0.9997,8
2                       GENERATE    12,FN$POISS         ; Генерация вызова
3                       TEST G      V$FREELN,2,ABND     ; Система заполнена?
4       ASN1            ASSIGN      1,V$LINE            ; Нет, указать номер
                                                        ; входной линии абонента,
                                                        ; вызывающего другого
                                                        ; для разговора
5                       GATE LR     *1,ASN1             ; Проверка занятости
                                                        ; линии
6       ASN2            ASSIGN      2,V$LINE            ; Выбор адресата связи
```

```
7                TEST NE     P1,P2,ASN2          ; Повторить, если адресат
                                                 ; совпадает со входной
                                                 ; линией абонента
8                LOGIC S     *1                  ; Установить выходную
                                                 ; линию абонента занятой
9                TRANSFER    BOTH,,BLKD          ; Дождаться связи
10      GETL     ENTER       LNKS                ; Установить связь
11               GATE LR     *2,BUSY             ; Проверка занятости
12               LOGIC S     *2                  ; Установить адресата
                                                 ; занятым
13               ADVANCE     120,FN$POISS        ; Разговор
14               LOGIC R     *1                  ; Установить входную
                                                 ; линию абонента
                                                 ; свободной
15               LOGIC R     *2                  ; Установить адресата
                                                 ; свободным
16               LEAVE       LNKS                ; Освободить связь
17      CKCH     TEST G      CH$WAIT,0,MTRM      ; Есть ли ожидающие
                                                 ; вызовы?
18               GATE LR     1,GETF              ; Проверить, свободна ли
                                                 ; выходная линия?
19               UNLINK      WAIT,GETL,1,2,1,GETF
;       Установить связь с первым ожидающим вызовом
20      MTRM     TERMINATE   1
21      GETF     UNLINK      WAIT,GETL,1         ; Подключить первый
                                                 ; ожидающий вызов
22               TRANSFER    ,MTRM
23      ABND     TERMINATE                       ; Отказаться от вызова
24      BLKD     LINK        WAIT,P1             ; Ожидать в порядке
                                                 ; поступления вызовов
25      BUSY     LOGIC R     *1                  ; Освободить линию
26               LEAVE       LNKS                ; Освободить связь
27               TRANSFER    ,CKCH
*
28      LNKS     STORAGE     10                  ; Количество
                                                 ; одновременных связей
                                                 ; между абонентами
29      LINE     VARIABLE    X$NRLINES#RN1/1000+1
;       Выбор линии
30      FREELN   VARIABLE    X$NRLINES-2#S$LNKS-CH$WAIT
;       Номер свободной линии
*       Операторы управления GPSS World
*
31               INITIAL     X$NRLINES,50        ; Количество линий
                                                 ; для связи
                 START       15,NP               ; 15 проходов
                 RESET                           ; Сброс статистики
                 START       1000                ; Моделирование
                                                 ; 1000 вызовов
```

Блок GATE. Блок **GATE** управляет потоком транзактов с помощью логических операторов. Блок **GATE,** как и блок **TEST**, не изменяет никаких атрибутов транзактов. Он определяет номер следующего блока, к которому должен перейти транзакт из блока **GATE.** Блок **GATE** может задержать транзакт на входе, если не задан альтернативный выход. Блок **GATE** имеет такой формат:

GATEX A,|B|

Операнд **A** содержит имя или номер объекта, для которого производится проверка. Операнд **A** может быть именем, положительным целым числом, СЧА или СЧА*СЧА.

Операнд **B** содержит номер следующего блока для входящего транзакта, если логический оператор имеет значение «ложь». Операнд **B** может быть именем, положительным целым числом, СЧА или СЧА*СЧА. Если операнд **B** определен, то он должен содержать номер блока, допустимый для текущей модели.

В дополнительном операторе **X** задается один из следующих логических операторов:

1. Логические операторы, связанные с устройствами:

NU – устройство *j*, заданное в операнде **A**, свободно;

U – устройство *j,* заданное в операнде **A**, занято (в результате выполнения транзактом блока **SEIZE** или **PREEMPT);**

NI – устройство *j*, заданное в операнде **A**, не прервано;

I – устройство *j*, заданное в операнде **A**, обслуживает прерывания;

FV – устройство *j*, заданное в операнде **A**, доступно;
FNV – устройство *j*, заданное в операнде **A**, не доступно.
2. Логические операторы, связанные с МКУ:
SE – МКУ *j*, заданное в операнде **A**, пустое (S[j]=0);
SNE – МКУ *j*, заданное в операнде **A**, не пустое (S[j]<>0);
SF – МКУ *j*, заданное в операнде **A**, заполнено (R[t]=0);
SNF – МКУ *j*, заданное в операнде **A**, не заполнено (R[j]<>0);
SV – МКУ *j*, заданное в операнде **A,** доступно;
SNV – МКУ *j*, заданное в операнде **A**, не доступно.
3. Логические операторы, связанные с транзактами:
M – в блоке *j*, заданном в операнде **A** блока **GATE,** находится в состоянии синхронизации транзакт, принадлежащий тому же семейству, что и транзакт, который находится в блоке **GATE** или пытается войти в этот блок;
NM – в блоке *j*, заданном в операнде **A** блока **GATE,** в состоянии синхронизации нет ни одного транзакта, принадлежащего тому же семейству, что и транзакт, который пытается войти в блок **GATE.**
4. Логические операторы, связанные с логическими ключами:
LS – логический ключ *j*, заданный в операнде **A**, включен;
LR – логический ключ *j*, заданный в операнде **A**, выключен.

Режимы условного и безусловного входов в блок GATE. Блок **GATE,** как и блок **TEST,** может работать в режимах безусловного и условного вхождения.

В ***режиме безусловного вхождения*** транзакт никогда не задерживается на входе блока **GATE.** Если заданный логический оператор имеет значение «истина», транзакт пытается перейти к следующему no номеру блоку. Если логический оператор имеет значение «ложь», го транзакты будут пытаться перейти к блоку, номер которого задан в операнде **B** блока **GATE.** Выбор следующего блока производится один раз в момент вхождения транзакта в блок **GATE.**

В ***режиме условного вхождения,*** если операнд **B** блока **GATE** пустой (альтернативный выход не задан), транзакты не смогут войти в блок **GATE** до тех пор, пока указанный в этом блоке логический оператор не будет иметь значение «истина». Интерпретатор не проверяет значение логических операторов, за исключением операторов **M** и **NM**. В режиме условного вхождения задержанные транзакты находятся в списках задержки и, таким образом, исключаются из числа транзактов, обрабатываемых интерпретатором до тех пор, пока соответствующий логический оператор не примет значение «истина».

Пример 4.60

```
QUEUE      LINE
GATE SV    LINE1
DEPART     LINE
```

В данном случае транзакт помещается в список задержки, если МКУ **LINE1** не доступно в тот момент, когда транзакт пытается войти в блок **GATE.** Когда МКУ становится доступным, все транзакты выводятся из списка и делают попытку войти в МКУ.

Блоки **GATE –** очень мощный инструмент, но они могут приводить к значительным затратам компьютерного времени на тщетные попытки транзактов войти в блок. Чтобы уменьшить частоту бесполезных попыток вхождения в блок, можно с помощью блоков **LINK** и **UNLINK** поместить транзакты в СП.

4.25. Организация вывода временных рядов из GPSS-модели

При моделировании часто возникает необходимость получить с выхода имитационной модели временные последовательности, для последующего анализа, например, с помощью статистических пакетов. Для этого можно использовать запись результатов моделирования в стандартный файл отчетов.

Для формирования временного ряда по результатам моделирования в GPSS-программу вставляют специальные строки для запоминания этого ряда и вывода его в файл GPSS-отчета. Ниже приведена программа модели СМО вида M/M/1, в которую добавлены строки 11, 71-73, 91-94 для проверки модели на наличие циклов регенерации. Начало цикла регенерации отмечается флагом 1111 (если есть циклы регенерации). Цикл регенерации начинается, когда вновь прибывший транзакт застает СМО

пустой (в ней нет транзактов). Проверка этого условия осуществляется в строке 71 GPSS-программы. В результате прогона модели формируется столбец из сохраняемых величин, который может считываться (например, пакетом статистики). Это позволяет использовать полученные данные для построения параметрической модели и формирования функции распределения для выходных данных имитационной модели.

Пример 4.61

```
EXPDIS      FUNCTION    RN1,C24
0,0/.100,.104/.200,.222/.300,.355/.400,.509/.500,.690/.600,.915
.700,1.200/.750,1.380/.800,1.600/.840,1.830/.880,2.120
.900,2.300/.920,2.520/.940,2.810/.950,2.990/.960,3.200
.970,3.500/.980,3.900/.990,4.600/.995,5.300/.998,6.200/.999,7/1,8
10      BD          TABLE       M1,0,500,30
11                  INITIAL     X1,1            ; Начальное значение
                                                ; номера
20                  GENERATE    200,FN$EXPDIS
30      INP         QUEUE       B
40                  SEIZE       B
50                  DEPART      B
60                  ADVANCE     150,FN$EXPDIS
70      OUT         RELEASE     B
71                  TEST NE     N$OUT,N$INP,CIKL ; Есть ли цикл?
*       Цикл продолжается
72                  SAVEVALUE   1+,1            ; Номер следующего
                                                ; значения
73                  SAVEVALUE   X1,M1           ; Запоминание
                                                ; исследуемой переменной
80                  TABULATE    BD
90                  TERMINATE
*       Начало цикла
91      CIKL        SAVEVALUE   1+,1            ; Первый номер в цикле
92                  SAVEVALUE   X1,M1           ; Первое значение времени
                                                ; в цикле
93                  SAVEVALUE   1+,1            ; Номер для флага
94                  SAVEVALUE   X1,1111         ; Флаг (отмечает начало
                                                ; цикла)
100                 TABULATE    BD
110                 TERMINATE   1
```

4.26. Краткая характеристика языка PLUS

Язык GPSS можно отнести к языкам высокого уровня. В силу этого он имеет довольно слабые алгоритмические возможности. Для устранения этого недостатка в систему GPSS World добавлен PLUS – язык низкого уровня. Выражения, процедуры и эксперименты PLUS можно использовать в GPSS-моделях.

Рассмотрим основные элементы языка PLUS.

Алфавит языка PLUS (GPSS World) содержит алфавитно-цифровые и специальные символы. Для задания имен используются алфавитно-цифровые символы (прописные буквы *A-Z,* строчные буквы *a-z,* цифры 0-9 и знак подчеркивания «_»). Для обозначения операторов и пунктуации используются специальные символы («#», «*», «&», «+», «-», «/», «\», «,», «;»). В комментариях допускается использование символов русского алфавита «А-Я».

Имена – это созданные пользователем последовательности символов, используемые для обозначения объектов, переменных и процедур. Имя должно начинаться с буквы, в нем можно использовать от 1 до 250 алфавитно-цифровых символов. При этом имена не должны совпадать с ключевыми словами GPSS и с СЧА. Следует отметить, что GPSS World не различает регистр алфавита.

PLUS-выражение – это комбинация одного или нескольких элементов, называемых факторами. Выражения строятся с использованием операторов и вызовов процедур, обрабатывающих факторы. Выражения могут использоваться в PLUS-процедурах и в операндах операторов GPSS. Если выражение используется в операндах GPSS-блоков, то оно должно записываться в круглых скобках.

Типы данных. Переменные пользователя, элементы матриц, ячейки, параметры транзактов могут иметь значения различных типов данных. В GPSS World используются три основных типа данных:

целочисленный (Integer) – это 32-х битовые двоичные числа. Если при выполнении арифметической операции происходит переполнение целочисленного значения, производится автоматическое преобразование его в вещественное;

вещественный (Real) – это числа с плавающей точкой двойной точности. Для представления мантиссы используется 15 десятичных разрядов, **А** для порядка – значения в интервале от – 306 до 306;

строковый (String) – это последовательность символов ASCII; строки не ограничены в размерах (максимальная их длина определяется параметрами настройки ***Edit/Setting.../Simulation).***

Выражения могут быть вычисляемыми, численно вычисляемыми, вычисляемыми в виде строки.

Факторы. Факторы – это основные элементы выражений, которые в свою очередь используются в операндах операторов GPSS и PLUS-процедурах.

Факторами выражений GPSS могут быть:

1) строковые константы;
2) вещественные константы;
3) целочисленные константы;
4) имена;
5) элементы PLUS-матрицы;
6) обращения к процедурам (значения, возвращаемые процедурами);
7) стандартные числовые атрибуты.

Для объединения факторов в выражение используются операторы и вызовы процедур.

Операторы. В табл. 4.48 представлены арифметические операторы, используемые в выражениях GPSS World, перечисленные в порядке убывания их приоритетов.

Таблица 4.48

Оператор	***Действие***	***Результат***	***Унарный/бинарный (количество операндов)***
–	Отрицание	Аддитивная инверсия	Унарный
^	Возведение в степень	Арифметический показатель степени	Бинарный
NOT или	Инверсия	1 (TRUE) или О (FALSE)	Унарный
AND	Логическое И	1 (TRUE) или О (FALSE)	Бинарный
OR	Логическое ИЛИ	1 (TRUE) или О (FALSE)	Бинарный
G или >	Больше	1 (TRUE) или О (FALSE)	Бинарный
L или <	Меньше	1 (TRUE) или О (FALSE)	Бинарный
E или =	Равно	1 (TRUE) или О (FALSE)	Бинарный
NE или /=	Не равно	1 (TRUE) или О (FALSE)	Бинарный
LE или <=	Меньше или равно	1 (TRUE) или О (FALSE)	Бинарный
GE или =>	Больше или равно	1 (TRUE) или О (FALSE)	Бинарный
# или *	Умножение	Арифметическое произведение	Бинарный
/	Деление	Арифметическое частное	Бинарный
\	Целочисленное деление	Целочисленное частное	Бинарный
@	Деление по модулю	Целочисленный остаток	Бинарный
+	Сложение	Арифметическая сумма	Бинарный

–	Вычитание	Арифметическая разность	Бинарный

Все арифметические операторы автоматически преобразуют строковые операнды в числовые значения.

Перенастройку некоторых операторов можно выполнить через пункт меню ***Edit/Setting.. /Simulation.***

К операторам языка PLUS также относятся:

PROCEDURE – определяет PLUS-процедуру;

EXPERIMENT – определяет PLUS-эксперимент;

TEMPORARY – определяет и ограничивает область действия локальных переменных пользователя и локальных матриц (существующих только во время вызова конкретной процедуры);

BEGIN/END составной оператор, создает блок PLUS-операторов;

присваивание – предназначен для изменения значений переменных;

вызов процедуры – вызывает библиотечную процедуру;

помеченный оператор – класс операторов, начинающихся с метки;

IF/THEN – условный оператор, проверяет выражение и, если результат равен «TRUE» (истина), выполняет действие;

IF/THEN/ELSE – условный оператор проверяет выражение и в зависимости от результата производит то или иное действие;

WHILE/DO – оператор цикла, несколько раз выполняет действие;

GOTO – оператор безусловного перехода, передает управление к метке внутри процедуры;

RETURN – останавливает выполнение процедуры и возвращает результат ее выполнения (после чего память, используемая процедурой, освобождается).

Процедуры языка PLUS могут записываться в любом месте модели (кроме тела другой процедуры). Остальные операторы PLUS могут появляться только внутри оператора **PROCEDURE.**

Различают процедуры ***пользовательские*** и ***встроенные (библиотечные).*** Пользовательские процедуры обычно используются для изменения значений глобальных переменных и поименованных величин или для вычисления выражения и выдачи результата. В первом случае результат вычисления не требуется. Во втором – требуется, в этом случае обязательно наличие оператора возврата **RETURN.** Результат, выдаваемый PLUS-процедурой, используется в операндах или других PLUS-выражениях. Если в операторе **RETURN** не задано какое-либо выражение или в процедуре отсутствует оператор возврата **RETURN,** то возвращается значение 0.

Для определения PLUS-процедуры необходимо поместить оператор **PROCEDURE** в файл модели и выполнить транслирование оператора вместе с моделью или передать оператор **PROCEDURE** процессу моделирования в интерактивном режиме. После этого уже определенную PLUS-процедуру можно вызывать при вычислении выражений или в PLUS-операторах присваивания.

Если процедура используется более, чем в одной модели, то ее можно сохранить в исходном файле, называемом ***библиотекой процедур пользователя.*** Командой **INCLUDE** можно включать эту библиотеку в каждую модель, в которой будет использоваться данная процедура.

Эксперимент – это особая разновидность PLUS-процедуры пользователя. Для ее определения используется оператор **EXPERIMENT.** Эта процедура применяется для управления несколькими повторяющимися имитациями.

Эксперименты обычно используются совместно с библиотечной процедурой **DoCommand()** для управления процессом моделирования, **А** также с библиотечной процедурой дисперсионного анализа **ANOVA** для анализа результатов моделирования. Обычно результаты эксперимента записываются в глобальную матрицу, которая передается процедуре **ANOVA.** Существует возможность автоматически создавать отсеивающие и оптимизирующие эксперименты при помощи генераторов экспериментов.

Эксперимент вызывается только командой **CONDUCT,** которую можно загрузить с помощью функциональной клавиши.

Эксперименты и обыкновенные процедуры, непосредственно или косвенно вызываемые экспериментом, могут использовать библиотечную процедуру **DoCommand()** для выполнения операторов GPSS, включая команды и блоки.

Библиотека процедур – это множество PLUS-процедур. Существует два типа библиотек: ***библиотека пользователя,*** представляющая собой совокупность процедур пользователя, и ***встроенная библиотека*** GPSS World, содержащая готовые к использованию строковые и математические процедуры. Для того, чтобы процедуру можно было использовать в PLUS-выражении, она должна находиться в библиотеке процедур.

Встроенная часть библиотеки содержит процедуры, которые могут быть разбиты на следующие группы:

Обслуживающие процедуры – процедуры, используемые для управления прогонами имитаций и анализа экспериментов.

1. Процедура **DoCommand.** Транслирует строку аргумента и передает результат трансляции выполняющейся модели. Эта процедура может вызываться только из экспериментов и процедур, вызванных из экспериментов.

2. Процедура **ANOVA.** Проводит многофакторный дисперсионный анализ, создает таблицу **ANOVA,** таблицу описательных статистик и выводит эти данные в журнал сессии.

3. Процедура **Exit()** завершает сеанс работы с GPSS World – останавливает процесс моделирования и закрывает все окна GPSS.

Файловые процедуры (процедуры потоков данных) **–** процедуры управления потоками данных внутри PLUS-процедуры. Потоки данных используются для чтения и записи файлов или для хранения и доступа больших объемов данных в памяти. Файловые процедуры выполняют следующие операции:

Open(DataStream, FileNameString) инициализирует поток данных;

Close(DataStream) – закрывает поток данных и извлекает код ошибки;

Read(DataStream) – считывает текстовую строку из потока данных;

Write(DataStream, String) – передает потоку данных текстовую строку;

Seek(DataStream, NewLinePosition) – устанавливает позицию текущей строки потока данных и извлекает предыдущую позицию строки.

Процедуры динамического вызова используются для вызова функций, хранящихся во внешних исполняемых файлах, включая динамически подключаемые библиотеки DLL. Соответствующие процедуры можно использовать для запуска функций, предоставляемых другими фирмами в отдельных исполняемых файлах и поддерживающих протокол CDECL.

Математические процедуры. К ним относятся:

ABS(Expression) – абсолютное значение;

ATN(Expression) – арктангенс (в радианах);

COS(Expression) – косинус;

EXP(Expression) – число *e,* возведенное в степень аргумента;

INT(Expression) – выделение целой части с отбрасыванием дробной части;

LOG(Expression) – натуральный логарифм;

SIN(Expression) – синус;

SQR(Expression) – квадратный корень;

TAN(Expression) – тангенс.

Вероятностные распределения процедуры теоретических вероятностных распределений (см. параграф 4.13).

Строковые процедуры выполняют операции со строками:

Align(InsertString, SourceString, Offset) – вставляет одну строку в другую с выравниванием по правому краю;

Catenate(Stringl, String2) – производит объединение строк;

Copies(SourceString, Count) – создает одну строку из нескольких копий исходной;

Datatype(Datum) – возвращает строку, содержащую тип данных аргумента;

Find(TestString, SourceString) – вычисляет смещение одной строки, содержащейся в другой строке;

Left(SourceString, MaxCount) – возвращает левую часть заданной строки;

Length(SourceString) – возвращает количество символов в строке;

Lowercase(SourceString) – переводит все прописные буквы в строке в строчные;

Place(InsertString, SourceString, Offset) – вставляет одну строку в другую с выравниванием по левому краю;

PolyCatenate(Stringl, Stringl,...) – производит объединение двух или более строк;

Right(SourceString, MaxCount) – возвращает правую часть заданной строки;

String(Datum) – преобразует данные в их строковый эквивалент;
StringCompare(Stringl, String2) – сравнивает две строки;
Substring(SourceString, Offset, MaxCount) – возвращает подстроку заданной строки;
Trim(SourceString) – удаляет начальные и конечные пробелы;
Uppercase(SourceString) – – переводит все строчные буквы в строке в прописные;
VaIue(Datum) – возвращает числовой эквивалент строки;
Word(SourceString, WordNumber) – возвращает заданное слово строки.

Процедуры запроса возвращают информацию о состоянии транзакта, находящегося в модели. К ним относятся такие процедуры:

QueryXNExist(TransactionNumber) – возвращает 1, если транзакт существует в модели, иначе – 0;

QueryXNParameter(TransactionNumber, Parameter) – возвращает значение параметра транзакта. Если искомого параметра не существует, происходит останов по ошибке;

QueryXNAssemblySet(TransactionNumber) – возвращает номер семейства, к которому принадлежит транзакт;

QueryXNPriority(TransactionNumber) – возвращает приоритет транзакта в виде целого числа;

QueryXNMl(TransactionNumber) – возвращает время входа транзакта в систему.

4.27. Команды GPSS WorId

Команды необходимы, как для построения программы модели, так и для интерактивного взаимодействия с моделью. Они включают операторы описания данных, операторы управления, которые включены в стандартный GPSS [10, 13], и команды, которые используются только в языке GPSS World.

Операторы описания данных и команды управления:

Команда START. Используется для инициации начала моделирования. Его формат:

```
START      A,[B],|C],[D]
```

В операнде **A** задается значение ***счетчика завершений,*** определяющего момент окончания прогона модели. Операнд **A** может быть только положительным целым числом.

Операнд **B** – операнд вывода статистики. Этот операнд может принимать значение «NP» или быть опущенным. Задание «NP» в операнде **B** приводит к блокированию вывода статистики. По умолчанию выводится стандартная статистика.

Операнд **C** не используется и сохранен для совместимости с предыдущими версиями GPSS.

Операнд **D** задает необходимость вывода содержания СТС и СБС. Операнд **D** может быть положительным целым. Если операнд **D** не равен 0, то СТС и СБС выводятся.

Моделирование продолжается до тех пор, пока счетчик завершения, определенный операндом **A**, не достигнет нулевого значения. Для уменьшения значения счетчика используется блок **TERMINATE** (подробнее см. в параграфе 4.4).

Стандартный числовой атрибут, который связан с этой командой, **TG1** – текущее значение счетчика завершения.

Оператор INITIAL. Позволяет задавать начальные значения сохраняемых величин (Savevalue), элементов матриц (Matrix) и логических ключей (Logicswitch). Имеет такой формат:

```
INITIAL    A|,B]
```

Значения операндов:

A – СЧА сохраняемых величин, элементов матриц или логических ключей. В операнде могут стоять:

LS<положительное целое>, LS$<имя> – имя логического ключа;

X<положительное целое>, X$<имя> – имя сохраняемой величины;

MX<положительное целое>( ) или **MX$<имя>( )** имя элемента матрицы.

B – устанавливаемое значение. По умолчанию равно единице. Операнд может быть числом, строкой, именем или **UNSPECIFIED.**

При выполнении оператора **INITIAL** значение, обусловленное операндом **B**, назначается логическому ключу, сохраняемой величине или элементу матрицы, определенному в операнде **A.**

Если операнд **A** определен как логический ключ, то операнд **B** может быть только нулем или единицей.

Если в операнде **B** использовано ключевое слово **UNSPECIFIED,** то сохраняемая величина, матрица или элемент матрицы устанавливается в «неопределенное» состояние. Обычно это ключевое слово используется, чтобы указать на отсутствие данных в матрице результатов, которая должна быть в дальнейшем проанализирована библиотечной процедурой **ANOVA**. Если матрица имеет размерность больше двух, то для установки ее в состояние **UNSPECIFIED** надо использовать язык PLUS.

С оператором **INITIAL** связаны блоки **LOGIC, SAVEVALUE** и **MSAVEVALUE.**

Пример 4.62

```
INITIAL     X88,12000
```

Значение 12000 записывается в сохраняемую величину с номером 88.

```
INITIAL     MX3(2,4),33
```

Значение 33 записывается в элемент (2, 4) матрицы 3.

```
INITIAL     XSQuote,"Now is the time ... "
```

Этот оператор назначает строчную константу сохраняемой величине **Quote.**

```
INITIAL     MX$Inventory(Part39,Stocklevel),200
```

Значение 200 записывается в строку **Part_39** столбца **Stocklevel** матрицы **Inventory.** Именам **Part39 и Stocklevel** должны быть предварительно назначены соответствующие целые числа в командах **EQU**

```
INITIAL     MainResult,UNSPECIFIED
```

Предварительно определяется матрица для экспериментов с именем **MainResuLt,** которая может не иметь данных.

Команда RESET. Сбрасывает в ноль статистику и СЧА системы, но не удаляет гранзакты из модели. Она используется для повторных экспериментов с моделью и сброса статистических данных переходного периода имитационного процесса. Имеет такой формат:

RESET

Действия команды **RESET:**

1) значение относительного модельного времени (**C1**) устанавливается в ноль;
2) значение абсолютного модельного времени (**AC1**) остается без изменений;
3) все датчики псевдослучайных чисел остаются неизменными;
4) значения сохраняемых величин и матриц, **A** также состояния логических ключей не изменяются;
5) счетчики числа входов в блоки (**Nj**) сбрасываются в ноль;
6) временные интегралы* устройств устанавливаются в ноль;
7) временные интегралы содержимого МКУ** устанавливаются в ноль;
8) счетчики числа входов в МКУ (**SCj**) и максимального содержимого МКУ (**SMj**) остаются неизменными;
9) временные интегралы всех очередей*** сбрасываются в ноль, счетчики вхождений в очередь **(QCj)** и максимального содержимого очереди **(QMj)** устанавливаются равными текущей длине очереди;
10) в таблицах стираются накопленные статистические данные;
11) временные интегралы СП устанавливаются в ноль, счетчики числа вхождений в списки (**CCj**) и максимального содержимого списков (**CMj**) устанавливаются равными текущей длине списка.

* *Временной интеграл устройства* – сумма длин всех интервалов времени, на протяжении которых устройство было занято.

** *Временной интеграл содержимого МКУ* рассчитывается так. В начале моделирования его значение равно нулю. При выполнении подпрограммы блоков **ENTER и LEAVE** вычисляются интервал времени, на протяжении которого содержимое МКУ не менялось, и соответствующее этому интервалу значение содержимого МКУ. Текущее значение временного интеграла содержимого МКУ увеличивается на значение произведения этих величин.

*** *Временной интеграл очереди* расчитывается так. В начале моделирования его значение равно нулю. При выполнении подпрограммы блоков **QUEUE и DEPART** вычисляется интервал времени, на протяжении которого содержимое очереди нс менялось, и соответствующее этому интервалу значение содержимого очереди. Текущее значение временного интеграла содержимого очереди увеличивается на значение произведения этих величин.

Команда CLEAR. Сбрасывает всю накопленную статистику, удаляет все транзакты из модели и устанавливает отсчет (нумерацию) транзактов, сгенерированных блоками **GENERATE,** начиная с единицы.

Формат записи команды **CLEAR:**

CLEAR [A]

Действия команды **CLEAR:**

1) все транзакты удаляются из модели;
2) содержимое всех блоков устанавливается в ноль;
3) текущие счетчики блоков (**Wj**) сбрасываются в ноль;
4) общие счетчики блоков (**Nj**) сбрасываются в ноль;
5) системное время (**C1** и **AC1**) устанавливается в ноль;
6) устройства становятся незанятыми и доступными;
7) МКУ становятся свободными и доступными;
8) временные интегралы устройств, МКУ, очередей и СП устанавливаются в ноль;
9) максимальные значения содержимого очередей, СП и МКУ устанавливаются равными их текущему значению;
10) состояние датчиков псевдослучайных чисел не изменяется;
11)внутренний счетчик транзактов, генерируемых в блоках **GENERATE,** устанавливается в ноль;
12) из всех числовых групп удаляются их члены;
13) содержимое всех сохраняемых величин и матриц принимает нулевое значение;
14) логические ключи сбрасываются.

Если используется **CLEAR OFF,** выполняется все перечисленное выше за исключением последних двух пунктов. То есть, если операнд **A=OFF,** то логические ключи, сохраняемые величины и матрицы остаются без изменений.

После выполнения всех названных операций команды **CLEAR** GPSS-модель просматривается интерпретатором в поиске блоков **GENERATE.** В каждом выявленном блоке **GENERATE** создается новый транзакт так же, как при первой интерпретации блока **GENERATE.** Заново вычисляется время начальной задержки и максимальное число транзактов, которые будут образованы в блоках **GENERATE.**

Команда RMULT. Моделирование часто требует нескольких различных последовательностей случайных чисел. Эти последовательности выдаются генераторами случайных чисел, которые действуют независимо друг от друга. При каждом запуске системы генераторы выдают одну и ту же последовательность чисел. Команда **RMULT** позволяет изменять такую последовательность путем изменения начальных множителей.

В системе GPSS World генераторы случайных чисел создаются по мере необходимости, их явное определение не обязательно. Несколько генераторов случайных чисел используются GPSS World для блоков **GENERATE, ADVANCE и TRANSFER.** Они определяются в пункте меню ***Edit/Setting/Random Numbers*** (случайные числа) настроек модели.

Формат команды **RMULT:**

RMULT [A],[Bl,[C],[D],|E],[F|,[Gj

В операнде **A** задается начальный множитель для первого генератора случайных чисел **RN1**.
В операнде **B** задается начальный множитель для второго генератора случайных чисел **RN2**.
В операнде **C** задается начальный множитель для третьего генератора случайных чисел **RN3**.
В операнде **D** задается начальный множитель для четвертого генератора случайных чисел **RN4**.
В операнде **E** задается начальный множитель для пятого генератора случайных чисел **RN5**.
В операнде **F** задается начальный множитель для шестого генератора случайных чисел **RN6**.
В операнде **G** задается начальный множитель для седьмого генератора случайных чисел **RN7**.

Стандартный числовой атрибут, связанный с этой командой, – **RN<номер генератора>.** Он возвращает случайное целое число из интервала от 0 до 999.

Операнды должны быть положительными целыми числами. В этом операторе должен быть задан хотя бы один операнд.

Пример 4.63

```
RMULT      875,1237,,,319
```

Устанавливаются начальные состояния множителей генераторов случайных чисел 1, 2 и 5. Остальные значения остаются без изменений.

Оператор EQU. Предназначен для присвоения числовых значений ***именам,*** которые используются в модели (аналог **#define** в языке программирования Си). Оператор имеет такой формат:

Таблица 4.49

Поле	***Информация в поле***
Метка	Имя
Операция	**EQU**
Операнд **A**	Выражение

Когда интерпретатор обрабатывает оператор **EQU**, он вычисляет выражение, заданное операндом **A**, после чего создает или переопределяет имя переменной. Имени присваивается результат вычисленного выражения. Полученное значение заменяет ссылки на это имя в операндах или выражениях, используемых в модели.

Значения имен могут использоваться как внутренние значения переменных пользователя, или они могут определять объекты, такие как метка. Именам, используемым как метки объектов, значения обычно не назначаются. Интерпретатор автоматически назначает индивидуальные значения именам, если они еще не появились в операторе **EQU**, в выражениях или операндах. Имена могут использоваться для определения объекта в СЧА.

Выражения, содержащиеся в операторе **EQU**, вычисляются согласно правилам для выражений и могут включать запросы к пользовательским или библиотечным процедурам PLUS. Выражения могут использовать любые из арифметических и логических операторов. Если в выражении используются параметры, они вычисляются для активного транзакта.

Имена, которым не были явно назначены значения, не могут использоваться в выражении. Необходимо назначить значение для имени прежде, чем будет вычислено выражение. Переменные пользователя могут быть заданы операторами **EQU** или в процедуре PLUS.

Если значение имени определено, то оно сохраняет свое значение на протяжении всего прогона модели. Имена переменных пользователя могут быть переопределены:

1) повторным определением и вводом новых операторов **EQU** в очереди команд;
2) в процедуре PLUS;
3) при выполнении интегрирования с помощью команды **INTEGRATE**

Значение метки блока не может быть изменено (переопределено после трансляции) оператором **EQU**. Если имя было определено для объекта, **A** затем переопределено, например, оператором **EQU**, то нет возможности обратиться к первоначальному имени.

Переменные **FVARIABLE** и **BVARIABLE** используют одну и ту же область имен.

Если необходимо использовать числовое имя для объекта, то оно должно быть назначено оператором **EQU** до определения объекта.

Пример 4.64

```
Mechanic    EQU        1
Mechanic    STORAGE    100
```

В СЧА и операндах блоков этого МКУ теперь может использоваться число 1 или имя **Mechanic.**

Пример 4.65

Рассмотрим модель одноканальной системы обслуживания:

```
10      GENERATE    30,6
20      SEIZE       DEVICE
30      ADVANCE     25,4
40      RELEASE     DEVICE
50      TERMINATE
60      GENERATE    24000
70      TERMINATE   1
        START       1
```

В этом случае статистика по устройствам имеет такой вид:

```
FACILITY  ENTRIES  UTIL.   AVE.TIME     AVAIL.   OWNER   PEND INTER RETRY DELAY
DEVICE      802    0.830    24.86         1       803      0    0     0     0
```

Используем оператор **EQU**:

```
10                  GENERATE    30,6
20                  SEIZE       DEVICE
30                  ADVANCE     25,4
40                  RELEASE     DEVICE
50                  TERMINATE
60                  GENERATE    24000
70                  TERMINATE   1
        DEVICE      EQU         1
                    START       1
```

Теперь статистика по устройствам имеет такой вид:

```
FACILITY  ENTRIES  UTIL.   AVE._TIME    AVAIL.   OWNER   PEND INTER RETRY DELAY
1           802    0.830    24.86         1       803      0    0     0     0
```

Кроме рассмотренных выше, к операторам описания объектов относятся:
BVARIABLE – определяет булеву переменную;
FUNCTION – определяет функцию;
FVARIABLE – определяет действительную переменную с фиксированной точкой;
MATRIX – определяет матрицу;
QTABLE – определяет таблицу для очереди;
STORAGE – определяет МКУ;
TABLE – определяет таблицу;
VARIABLE – определяет переменную. Все эти операторы были рассмотрены ранее. Отметим только, что в GPSS World возможности работы с матрицами расширены (увеличена максимальная размерность матрицы с 2-х до 6). Поэтому оператор **MATRIX** имеет следующий формат:

NAME MATRIX A,B,C,|D],[E],[F|[,G]

Метка **NAME** определяет имя матрицы.

Операнд **A** не используется (оставлен для совместимости с предыдущими версиями GPSS). Все остальные операнды должны быть только положительными целыми числами и задают размерность матрицы по соответствующему измерению:

B – задает максимальное значение индекса для первой размерности;
C – задает максимальное значение элементов для второй размерности:
D – задает максимальное значение элементов для третьей размерности;
E – задает максимальное значение элементов для четвертой размерности:
F – задает максимальное значение элементов для пятой размерности;
G – задает максимальное значение элементов для шестой размерности.

Только к первым двум размерностям матрицы можно обращаться в блоке **MSAVEVALUE**. В этом случае все отсутствующие размерности предполагаются равными единице. Процедуры PLUS могут обращаться ко всем элементам любой матрицы. Если необходимо использовать матрицу размерностью больше, чем 2, надо создать одну или несколько процедур PLUS для обращения к ним. Матрица, определенная в операторе **MATRIX,** имеет глобальную область действия, т.е. доступна всем процедурам PLUS. Кроме того, может быть создана временная матрица с локальной областью действия (доступна только в процедуре PLUS, в которой она объявлена).

Матрицы никогда не удаляются из модели, однако, они могут быть переопределены другим оператором **MATRIX.**

Когда матрица создается впервые или когда используется команда **CLEAR ON,** всем элементам задаются значения 0. При необходимости можно использовать оператор **INITIAL** для задания элемента матрицы неопределенного состояния – **UNSPECIFIED.** Это полезно для сохранения результатов эксперимента. Когда такая матрица результатов передается библиотечной процедуре **ANOVA,** элементы **UNSPECIFIED** будут обработаны как отсутствующие данные, вместо нулевых значений.

Максимальный размер памяти, выделяемый для матриц, определяется в настройках параметров системы GPSS World.

Команда EXIT (ВЫХОД). Предназначен для завершения работы с системой GPSS World. Имеет следующий формат:

EXIT [A]

В результате выполнения команды **EXIT** система немедленно завершает работу. Операнд **A** является необязательным и используется для управления сохранением модели и результатов моделирования. Он может принимать следующие значения:

0 или не указан – в окне сообщений появляется запрос на сохранение для каждого вновь созданного или измененного в сеансе моделирования файла;

1 – все файлы сохраняются без запроса;

– 1 – не сохраняются никакие файлы.

Диалоговые команды GPSS World. Могут быть включены в модель или введены в диалоговом окне команд – ***Command*** (в этом случае они называются диалоговыми командами). Любая имитация связана очередью команд. После трансляции модели при моделировании все команды выстраиваются в очередь команд и выполняются одна за другой, пока этот процесс не будет приостановлен или пока не будут выполнены все команды.

Команды могут быть немедленными или поставленными в очередь. Немедленные команды, как **HALT и SHOW,** выполняются, как только они получены в процессе моделирования, другие команды ставятся в очередь. Они помещаются в конце списка команд, которые еще не были выполнены. Если при моделировании нет немедленных команд, то выполняется следующая команда из очереди команд. Если имитация выполняется и получена немедленная команда, то моделирование временно приостанавливается для выполнения немедленной команды.

Команда **HALT –** особый случай. Мало того, что она немедленная команда, но она также удаляет все оставшиеся команды из очереди команд. После выполнения команды **HALT** моделирование прекращается.

Обычно удобно поместить список часто используемых команд в текстовый файл. Тогда можно использовать команду **INCLUDE** (ВКЛЮЧИТЬ) для посылки объекту моделирования целой последовательности команд. Еще проще, можно загружать в функциональную клавишу команду **INCLUDE** и выполнять целый список команд одиночным нажатием соответствующей клавиши.

Команда CONDUCT (ПРОВЕСТИ) начинает эксперимент. Ее формат:

CONDUCT |A|

Операнд **A** задает имя вызываемой процедуры эксперимента, написанной на языке PLUS. Операндом может быть только имя PLUS-процедуры. Данный операнд является необязательным.

Команда **CONDUCT** – немедленная команда, которая может быть послана только ***приостановленному*** процессу моделирования.

Эта команда запускает эксперимент и передает параметры к предварительно организованному эксперименту на языке PLUS для объекта моделирования. Если при моделировании используется тол: ко одиночный эксперимент без параметров, операнд **A** не требуется.

Пример 4.66

CONDUCT MyExperiment(NumberOfTellers, StartingReplicateNumber)

В этом примере PLUS-эксперимент **MyExperiment** запускается точно так же, как любая другая процедура. Глобальные переменные пользователя **NumberOfTellers** и **StartingReplicateNumber**

используются для того, чтобы определить, где эксперимент начинается или возобновляется при прогоне модели. Параметры вычисляются в глобальном контексте и передаются вызываемому эксперименту.

Как только начался эксперимент с командой **CONDUCT,** диалоговые возможности взаимодействия с выполняемой моделью ограничены. Можно только отображать часы модельного времени с помощью меню ***View/Simulation Clock.*** Если в процессе моделирования возникла необходимость взаимодействия с моделью, то эксперимент необходимо остановить командой **HALT.** Команда **CONDUCT** не может быть запущена обращением к процедуре **DoCommand.** В ходе эксперимента доступны только команда **HALT** и библиотечная процедура **DoCommand.** С командой **CONDUCT** не связаны никакие СЧА.

Команда CONTINUE (ПРОДОЛЖИТЬ) возобновляет процесс моделирования, который был ранее прерван либо остановлен командой **STOP.**

Формат команды:

CONTINUE

Моделирование считается ***остановленным,*** когда встречается условие, установленное командой **STOP** или **STEP.** Команда продолжает процесс моделирования, но не исключает повторные условия останова. Условие останова может быть исключено из модели опцией **OFF** команды **STOP** или в окне блоков.

Команда **CONTINUE** может использоваться после команды **HALT,** которая удаляет все команды из очереди команд. Только команда **CONTINUE** может продолжить процесс моделирования, если счетчик завершения остался положительным числом. В этом случае продолжается планирование появления транзактов в модели. В противном случае процесс моделирования прекращается, появление транзактов не планируется, генераторы случайных чисел в начальное состояние не сбрасываются.

Команда **CONTINUE** может быть выполнена с помощью «горячих клавиш» [Ctrl+Alt+C], если активно окно GPSS World.

Команда INCLUDE (ВКЛЮЧИТЬ) транслирует файл операторов модели.

Формат команды:

INCLUDE A

Операнд **A** обязательный, он определяет подключаемый дополнительный файл модели или файл со списком команд. Операнд представляет собой строку символов и определяет путь к файлу.

Пример 4.67

```
INCLUDE    "example_l.txt"
```

Когда транслятор получает команду **INCLUDE,** он считывает текст модели из файла с именем **example_l.txt.** Так как путь к файлу не указан, то предполагается, что файл находится в каталоге проекта модели. Команда **INCLUDE** является немедленной командой. При получении этой команды транслятор начинает транслировать подключаемый файл GPSS-модели **example_l.txt** так, как будто подключаемая модель заменила строку с командой **INCLUDE.** Допускается подключение не более пяти файлов.

Пример 4.68

```
INCLUDE    "A:\priml.txt"
```

В этом примере указан полный путь к подключаемому файлу, который находится на дискете в дисководе A.

Система GPSS World работает только с файлами, имеющими расширение «.txt». В этих файлах может находиться либо неоттранслированная модель GPSS-программы, либо последовательность команд. При попытке подключения с помощью этой команды оттранслированной модели с расширением «.gps» транслятор выдает сообщение о том, что подключаемый файл должен быть только с расширением «.txt».

Команда HALT прерывает моделирование и очищает очередь команд. Формат команды:

HALT

Команда **HALT –** немедленная команда и поэтому не помещается в очередь команд объекта моделирования. Она выполняется немедленно, переводя процесс моделирования в приостановленное состояние и удаляя любые оставшиеся команды из очереди команд. Моделирование может быть продолжено позже командой **CONTINUE.** Только после этой команды можно использовать команду GROUPS

Команда **HALT** может быть выполнена с помощью «горячих клавиш» [Ctrl+Al+H], если активно окно GPSS World.

Команда INTEGRATE устанавливает интегрирование и пороги непрерывной переменной. Формат команды:

NAME INTEGRATE A,[B],C,[D],[E]

Таблица 4.50

Операнд	***Значение***	***Результат по умолчанию***
A	Производная	Ошибка
B	Порог 1	Null
C	Метка 1	Null
D	Порог 2	Null
E	Метка 2	Null

NAME – имя переменной пользователя.

Значение операндов:

A – производная. Операндом **A** может быть имя, число, строка, выражение в скобках или СЧА.

B – первое пороговое значение (порог). Операндом **B** может быть имя, число, строка, выражение в скобках или СЧА.

C – первая метка. Операндом **C** может быть имя, положительное целое число, выражение в скобках или СЧА. Операнды **B** и **C**: или оба используются, или оба не используются.

D – второе пороговое значение (порог). Операндом **D** может быть имя, число, строка, выражение в скобках или СЧА.

E – вторая метка. Операндом **E** может быть имя, положительное целое число, выражение в скобках или СЧА. Операнды **D** и **E**: или оба используются, или оба не используются.

Когда объект моделирования принимает команду **INTEGRATE,** он размещает ее в конце очереди команд.

При выполнении команд **INTEGRATE** переменная пользователя автоматически интегрируется с течением системного времен. Для интегрирования используется модифицированный метод Рунге-Кутта-Фельберга пятого порядка с переменным размером шага.

Всем переменным пользователя, используемым в интегрировании, до выполнения моделирования должны быть даны начальные значения. Это можно сделать с помощью операторов **EQU** или операторов присваивания в PLUS-процедуре.

Операнд **A** команды **INTEGRATE** используется для производной переменной пользователя по времени. Он может быть очень простым или весьма сложным. В последнем случае можно определить процедуру PLUS и разместить команду вызова в вводимом выражении, используемом для операнда **A**.

Каждая команда **INTEGRATE** может иметь ноль, один или два числовых порога. Операнды **B** и **C** могут использоваться для определения первого порога, и (или) операнды **D** и **E** могут использоваться для определения второго порога. В любом случае, первый операнд пары определяет значение порога, второй указывает метку блока, который принимает генерируемые транзакты.

Если в течение интегрирования значение интегрируемой переменной достигает значение порога (с любого направления), создается новый транзакт. Этому транзакту устанавливается приоритет 0, и он переходит в блок, связанный с этим порогом в команде **INTEGRATE.** Время входа транзакта в модель определяется линейной интерполяцией. Для увеличения точности при приближении к порогу шаг интегрирования уменьшается. Транзакты, генерируемые при пересечении порога, могут использоваться для того, чтобы изменить значение порога.

Оба порога равноправны, нет необходимости определять один как нижний, **A** второй как верхний. Новый транзакт генерируется при пересечении любого порога с любой стороны. Если направление

пересечения важно для функционирования модели, то нужно следить за значением переменной либо проверять направление пересечения.

Интегрирование начинается автоматически в активном или «разрешенном» состоянии. Во время выполнения моделирования можно включать или выключать интегрирование, используя один или несколько блоков **INTEGRATION.**

Чтобы настроить автоматическое интегрирование переменной, необходимо:

1) объявить командой **INTEGRATE** интегрируемую переменную;

2) присвоить начальные значения используемым переменным.

Пусть имеется простое дифференциальное уравнение вида $y' = f(\Theta)$, где Θ – это выражение, в которое входят некоторые переменные и (или) значение системного времени (СЧА **AC1**). По определению $f(\Theta)$ – это производная переменной y по времени. Итак, для настройки интегрирования нужно:

1) объявить командой **INTEGRATE** интегрируемую переменную **Y_** (не забыв при этом заключить выражение производной в скобки):

```
Y_      INTEGRATE   (Θ)
```

2) присвоить переменной **Y_** начальное значение, например,

```
Y_      EQU         100.3
```

3) присвоить начальные значение всем переменным, входящим в выражение Θ (с помощью операторов **EQU** или операторов присваивания в PLUS-процедурах).

Производная показывает, как быстро изменяется переменная во времени. Она указывает, на какую величину GPSS World автоматически увеличивает числовое значение интегрируемой переменной в процессе моделирования.

Пусть, например, некоторый склад строится со скоростью 2 блока в минуту, **A** единица модельного времени равна одной секунде [20]. В этом случае настройка автоматического интегрирования осуществляется так:

```
Inventory     INTEGRATE   (2.0/3600)
Inventory     EQU         100
```

Естественно, переменной **Inventory** может быть придано любое другое начальное значение.

Вычисление производной требует большего времени, чем вычисление аналитического выражения. В таком простом примере, как этот, где переменная вычисляется как функция от времени, гораздо быстрее просто посчитать значение переменной во время моделирования, чем интегрировать переменную.

Интегрирование должно использоваться в случаях, когда не известно решение дифференциального уравнения.

Пример 4.68 [20]

```
X_      INTEGRATE   (Y_),0.707,Wake_Up
Y_      INTEGRATE   (–X_)
X_      EQU         1.0
Y_      EQU         0.0
```

Этот пример определяет систему из двух дифференциально-разностных уравнений первого порядка, точное решение которой таково:

$$\begin{cases} x = \cos(t) \\ y = -\sin(t) \end{cases} \qquad (4.23)$$

Выражения в командах **INTEGRATE** используются как производные по времени. Начальные значения непрерывных переменных установлены командами **EQU**.

В процессе моделирования интегрирование автоматически выполняется между моментами дискретного времени. Порог установлен для пользовательской переменной **X_** *. Когда эта переменная пересекает значение 0,707 (с любого направления), создается новый транзакт и направляется в блок, помеченный меткой **WAKE_UP.**

* Обозначение **X** используется для СЧА сохраняемых величин и не может использоваться как имя переменной, поэтому в данной модели используется имя **X_**.

При использовании интегрирования моделирование выполняется поочередно в непрерывных и дискретных стадиях. В процессе имитации при планировании событий выполняется дискретная стадия моделирования. В этом случае часы модельного времени продвигаются от события к событию. Между моментами наступления событий выполняется непрерывная стадия моделирования, в течение которой интегрирование осуществляется с маленькими приращениями времени, называемыми минишагами. График переменной интегрирования выводит промежуточные значения в концах минишагов. Когда происходит пересечение порога, генерируется транзакт и моделирование переходит в дискретную стадию. Непрерывная и дискретная стадии могут взаимодействовать. Например, интегрируемым переменным пользователя в дискретной стадии могут быть назначены новые значения. Это можно сделать, используя операторы **EQU** или вызывая предназначенные для этого PLUS-процедуры.

Пример 4.69

Пусть PLUS-процедура определена следующим образом:

```
PROCEDURE  SetPop(PopLevel) BEGIN
        Foxes=PopLevel;
END;
```

Можно повторно инициализировать переменную пользователя, введя, например, блок PLUS в модель

PLUS (SetPop (200))

или используя ввод выражения, которое вызывает **SetPop()** в некотором другом блоке.

Уравнения высокого порядка [20]. При необходимости использования в модели дифференциального уравнения высокого порядка, необходимо уменьшить его порядок до первого. Для этого нужно переписать это уравнение как систему уравнений первого порядка. Это выполняется достаточно просто путем ввода новой переменной для каждой промежуточной производной.

Например, пусть имеем дифференциальное уравнение третьего порядка:

$$25y''' - 6y' + y = 0.$$

Введем обозначения:

$$u = y',$$

$$v = u' = y''.$$

Для уравнений еще большего порядка нужно продолжить введение новых переменных. После подстановки новых переменных в начальное уравнение получим дифференциальное уравнение первого порядка:

$25v' - 6u + y = 0.$

Теперь у нас есть следующая система дифференциальных уравнений:

$$\begin{cases} y' = u, \\ u' = v, \\ v' = \frac{6}{25}u - \frac{1}{25}y. \end{cases}$$

В системе GPSS World это будет выглядеть так:

```
Y_      INTEGRATE   U_
U_      INTEGRATE   V_
V_      INTEGRATE   ((6/25) # U_-(1/25)#Y_)
Y_      EQU         10.0
U_      EQU         1.0
U_      EQU         1.0
```

Начальные значения для **U_, V_** и **Y_,** определенные при помощи оператора **EQU,** задаются в зависимости от начальных условий моделирования.

Пример 4.70 [20]

Постановка задачи «Хищник-Жертва». На маленьком острове бесконтрольно растет популяция кроликов. Проблема стоит так остро, что местные фермеры прикладывают значительные усилия, чтобы прекратить увеличение популяции кроликов. Для контроля над ситуацией они хотят завести популяцию лис.

Для моделирования этого явления используется модель «хищник – жертва». При моделировании изучается такой вопрос: «Что произойдет, если выпустить 80 лис?».

```
*****************************************************************
*       Модель Lotka-Volterra Хищник-Жертва
*       Действия:
*       График "Лисы и Кролики": X 12000; Y 0-3000
*       START 1
*****************************************************************
*
*       Не забывайте выражения производных заключать в скобки
Foxes           INTEGRATE   (FoxRate())
Rabbits         INTEGRATE   (RabbitRate())
*
*       Начальные условия
*
Foxes           EQU         80
Rabbits         EQU         1000
*
*       Параметры модели
*
K_              EQU         0.2000      ; Эффективность хищников
A_              EQU         0.0080      ; Уровень смертности хищников
B_              EQU         0.0002      ; Фактор охоты
C_              EQU         0.0400      ; Уровень рождаемости кроликов
*
*       Управляющий сегмент дискретного моделирования
*
                GENERATE    10000
                TERMINATE   1
*
PROCEDURE   FoxRate()       BEGIN
*
*       Скорость роста популяции лис
*
        TEMPORARY           BirthRate, DeathRate, TotRate;
*       Граничные значения для переменной
        IF (Foxes < 0) THEN Foxes = 0;
        IF (Foxes > 10e50) THEN Foxes = 10e50 ;
        BirthRate = K_ # B_ # Foxes # Rabbits;
        DeathRate = A_ # Foxes;
        TotRate = BirthRate - DeathRate;
        RETURN      TotRate;
END;
*
PROCEDURE RabbitRate() BEGIN
*
*       Скорость роста популяции кроликов
*
TEMPORARY BirthRate, DeathRate, TotRate;
*       Граничные значения для переменной
        IF (Rabbits < 0) THEN Rabbits = 0;
        IF (Rabbits > 10e50) THEN Rabbits = 10e50 ;
        BirthRate = C_ # Rabbits;
        DeathRate = B_ # Foxes # Rabbits ;
        TotRate = BirthRate - DeathRate;
        RETURN      TotRate;
END;
```

На рис.4.15 представлен отчет по результатам моделирования при условии, что начальное значение количества лис равно 80. На рис. 4.16 дан график изменения размеров популяций кроликов и лис для этого начального значения.

```
GPSS World Simulation Report - Integrate Model.4.1
                  Monday, February 10, 2003 07:25:31

          START TIME          END TIME  BLOCKS  FACILITIES  STORAGES
               0.000         10000.000     2         0          0

             NAME                          VALUE
          A_                                0.008
          BIRTHRATE                     10008.000
          B_                                0.000
            C_                                      0.040
          DEATHRATE                     10009.000
          FOXES                           530.069
          FOXRATE                       10001.000
          K_                                0.200
          RABBITRATE                    10003.000
          RABBITS                          52.949
          TOTRATE                       10010.000

 LABEL              LOC  BLOCK TYPE     ENTRY COUNT CURRENT COUNT RETRY
                     1    GENERATE             1              0       0
                     2    TERMINATE            1              0       0

FEC XN   PRI        BDT       ASSEM  CURRENT  NEXT  PARAMETER     VALUE
     2    0      20000.000       2        0     1
```

Рис. 4.15

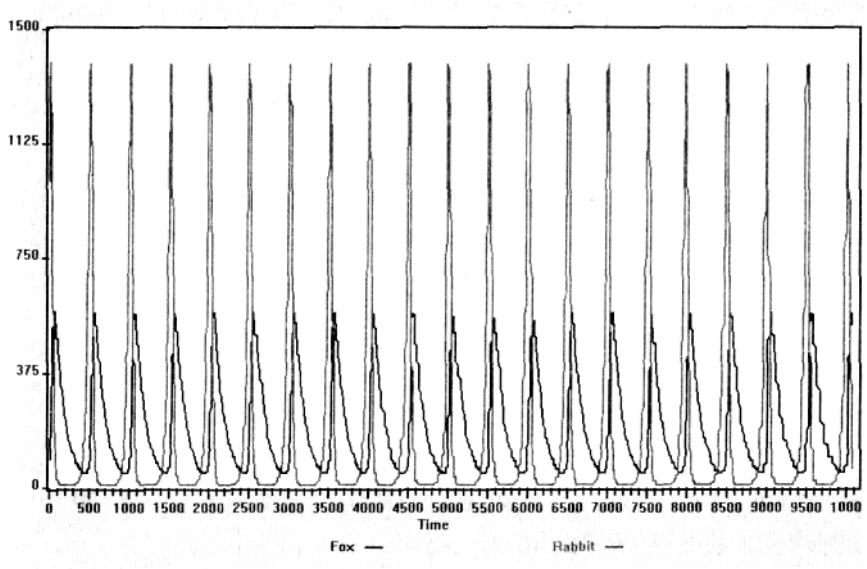


Рис. 4.16

Команда REPORT. Выводит стандартный статистической отчет о работе модели.
Формат команды:

REPORT [A][.B]

Операнд **A** не используется (оставлен для совместимости с предыдущими версиями).
Операнд **B** – признак немедленного создания стандартного отчета. Может иметь значение **NOW.**

Пример 4.71

REPORT

Эта команда немедленно создает стандартный отчет.

REPORT ,NOW

Этот формат команды сохранен для совместимости с версией GPSS/PC. Как и в предыдущем случае, она немедленно создает стандартный отчет.

Команда **REPORT** ставится в конец очереди команд. Способ выдачи информации в стандартный отчет может быть настроен в параметрах системы ***Edit/Settings/Report.*** Если в параметрах указан вывод в окно ***«In Windows»,*** то стандартный отчет создается в новом окне и может быть сохранен в файле. В

противном случае стандартный отчет создается, ему присваивается последовательный номер, и он автоматически помещается в файл.

Команда **REPORT** больше не использует операнд **A** и всегда предполагает, что операнд **B** имеет значение **NOW.** Она игнорирует параметр «Создать стандартный отчет» ***(CreateStandardReport),*** который используется для автоматического создания отчета, и операнд B команды **START,** используемой для текущей имитации. Обычно нет необходимости использовать команду **REPORT,** так как отчет создается автоматически в соответствии с параметрами настройки отчета, которые задаются в меню ***Edit/Settings/Report.***

Команда SHOW. Отображает значение выражения в строке состояния в окне модели. Эта команда выполняется немедленно, как только получена системой.

Формат команды:

SHOW <выражение>

Выражения в команде **SHOW** бывают арифметического и логического типа (см. параграф 4.12).

Пример 4.72

SHOW LOG(Q$BARBER)

Указанная команда вычисляет натуральный логарифм от СЧА **Q$BARBER** и выводит результат на экран дисплея.

SHOW X$PROFIT – X$expense

Если **X$PROFIT –** суммарный доход, а **X$EXPENSE –** суммарные затраты, то команда **SHOW** выведет их разность в окне модели в строке состояния.

Копия сообщения о выполнении этой команды с результатом вычисления посылается в окно журнала сессии. Если нет активных транзактов, т.е. имитация не выполнялась, или именам, используемым в команде **SHOW,** не были назначены значения, то выдается сообщение об ошибке. Эта команда отображает до 38 цифр или знаков.

Команда STEP. Задает возможность прерывания процесса моделирования при прохождении указанного количества блоков.

Формат команды:

STEP A

Операнд **A** – счетчик прохождения блоков. Операнд может быть положительным целым.

Команда **STEP** не является немедленной командой, **A** становится в конец очереди команд. Когда процесс моделирования прерывается, модель переходит в «приостановленное» состояние. В окне журнала сессии появляется сообщение трассировки. Когда используется команда **STEP,** в модели должен быть активный транзакт (то есть модель перед этим может быть запущена командой **START).**

Модель, которая стартовала с использованием команды **STEP,** не может быть завершена при нулевом счетчике завершения. Моделирование завершается при прохождении указанного в операнде **A** команды **STEP** количества блоков.

Если команда **STEP** выполняется, то счетчик завершения не устанавливается; блоки при необходимости перенумеровываются; все блоки, генерирующие транзакты, приводятся в исходное состояние; генераторы случайных чисел в начальное состояние не сбрасываются.

Пример 4.73

STEP 1

Процесс моделирования останавливается при прохождении одного блока модели.

Команду **STEP** используют для ***трассировки*** модели. Для этого необходимо выполнить следующие действия:

1. К тексту модели прибавить последовательность команд:

START 1

STEP 1

Моделирование начнется после команды **START** (счетчик завершения или длительность прогона модели выбирают так, чтобы на момент завершения моделирования в модели уже был активный транзакт). Активный транзакт после команды **STEP** продвинется к следующему блоку.

2. Перейти в диалоговое окно блоков ***(«Blocks Window»).*** В окне можно проследить движение каждого транзакта по блокам модели. В окне журнала сессии появится информация, подобная следующей:

```
01/20/03 16:39:53 Halt. XN: 1. Block 2 Next.
01/20/03 16:39:53   Clock:4.504044. Next: TEST. Line 9.
01/20/0316:39:53      TESTLE   Q$Barber,l,Finis;Waitiflinelorless
```

3. Нажать «мышкой» на поле **STEP** в меню команд окна блоков. Выполнится команда

```
STEP   1
```

и активный транзакт продвинется на один блок вперед.

4. Пункт 3 выполнить столько раз, сколько это необходимо. Для выполнения команды **STEP** можно использовать комбинацию клавиш [Ctrl+Alt+l].

Команда STOP. Устанавливает или снимает условие останова моделирования.

Формат команды:

```
STOP          [A],[B][,C1
```

Операнд **A** – номер транзакта, может быть положительным целым. Если операнд опущен, то любой транзакт удовлетворяет условию останова.

Операнд **B** – номер блока. Если операнд опущен, то любой блок удовлетворяет условию останова. Операнд может быть положительным целым или именем.

Операнд **C** – флажок состояния команды (может быть **ON** или **OFF).** По умолчанию – **ON**.

Пример 4.74

```
STOP          1000,50
```

Эта команда задает условие останова модели при входе транзакта с номером 1000 в блок с номером 50.

Команда **STOP** без операндов вызывает немедленный останов процесса моделирования, а с опцией **ON** устанавливает условие останова моделирования, но не стартует модель. Для запуска моделирования используют последовательность операторов и команд **START, STEP, CONTINUE.**

Команда **CONTINUE** позволяет выйти из состояния останова и продолжить моделирование, однако, условие останова, введенное ранее командой **STOP,** остается включенным. Для отключения условия останова необходимо выполнить в команду **STOP** с флагом **OFF.**

Условия останова могут быть заданы в окне блоков.

4.28. Диалоговые возможности GPSS World

Взаимодействие пользователя с системой GPSS World осуществляется с помощью оконного интерфейса в режиме активного диалога. Для этого в системе предусмотрены диалоговые окна, которые позволяют отображать информацию о состоянии отдельных объектов на экране дисплея. Эта информация может быть как статической, так и динамической. Главное окно, появляющееся при запуске системы, показано на рис. 4.17.

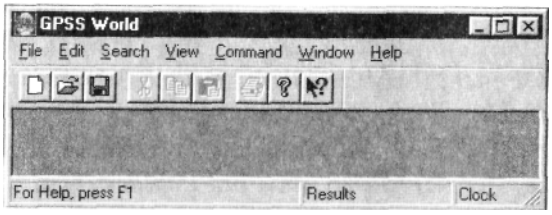


Рис. 4.17

Пункт меню ***File/New*** позволяет создать новую GPSS-модель или текстовый файл. Для удобства создания модели можно воспользоваться пунктом меню ***Edit/InsertGPSSBlock...,*** позволяющим выбрать из специального окна блоков и вставить в модель любой GPSS-Block (рис.4.18). При выборе блока открывается окно с его параметрами (рис. 4.19). Использование этого средства GPSS World гарантирует правильность формирования строки модели с выбранным блоком.

Текстовый файл обычно содержит последовательность команд, которые могут быть подключены к модели с помощью команды **INCLUDE,** как было описано в параграфе 4.27.

Пункт меню ***Edit/Insert Experiment*** позволяет вставить в модель эксперимент, написанный на языке PLUS.

Пункт меню ***Edit/Expression Window...*** предназначен для редактирования информации в окне выражений, если это окно использовалось в модели. Аналогично пункт меню ***Edit/Plot Window...*** предназначен для редактирования информации в окне графиков.

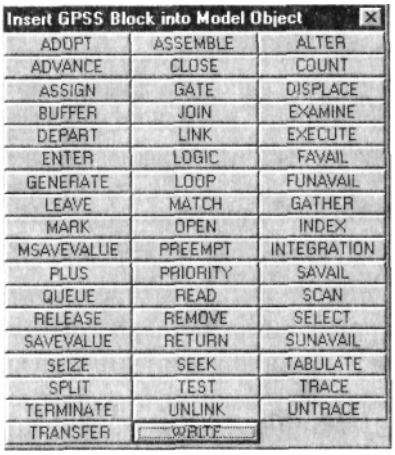


Рис. 4.18

Пункт меню ***Edit/Settings...*** позволяет задавать параметры имитации, отчетов, генераторов случайных чисел, функциональных клавиш и выражений.

Пункт главного меню ***Search*** помогает передвигаться внутри текстового объекта. Первый его пункт ***Find/Replace*** (Найти/Заменить) открывает обычный диалог для поиска и замены текстовой информации.

Последующий набор пунктов меню используется для работы с закладками, позволяя размещать невидимые маркировочные знаки, которые сохраняются с объектом. Они составляют циклический список, который можно просматривать с помощью соответствующих команд меню или с помощью клавиш.

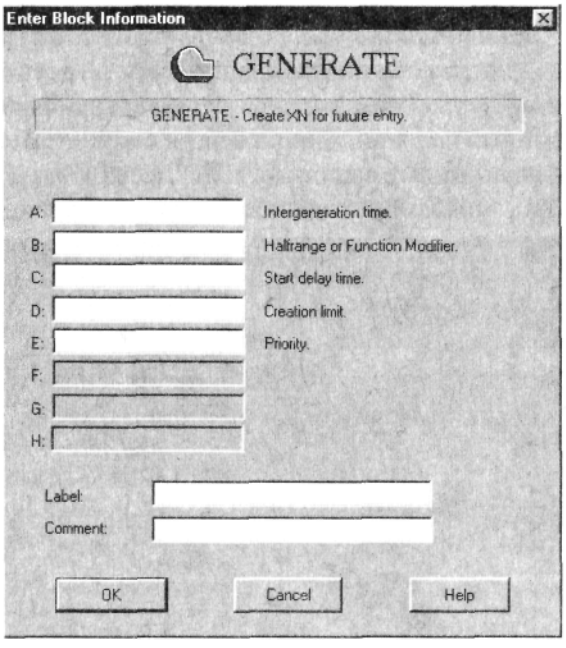


Рис. 4.19

Пункт меню ***Search/NextBookmark*** переводит к позиции следующей закладки в тексте. Пункт меню ***Search/Mark*** водит закладку в текущую позицию курсора, ***Search/Unmark –*** снимает выделение, удаляя текущую закладку, а ***Search/Unmark All*** снимает все закладки. Пункт меню ***Search/Select to Bookmark*** выделяет текст от текущей позиции курсора до текущей позиции закладки. Последние два пункта меню ***Search*** имеют дело с сообщениями об ошибках, которые возникают при трансляции GPSS-модели. Ошибки трансляции заносятся в циклический список. Этот список хранится вместе с GPSS-моделью и модифицируется при повторной трансляции. Для поиска ошибок используется пункты меню ***Search/Next Error*** (следующая ошибка) и ***Search/Previous Error*** (предыдущая ошибка). Для быстрого поиска с помощью клавиатуры используются комбинации клавиш [b+a+N] и [b+a+P] соответственно. Курсор останавливается перед ошибкой.

Пункт главного меню ***View*** (вид) управляет отображением информации в окнах. Первый пункт меню ***View/Notices*** (заметки) выводит информацию о текущей версии GPSS World и ее особенностях. Второй пункт меню ***View/Toolbar*** позволяет отображать или не отображать панель инструментов в главном окне. Третий пункт меню ***View/Entity Details*** управляет выдачей детальной информации для некоторых динамических окон. Например, в окне блоков может быть показана детальная информация по всем блокам модели (рис. 4.20) или отображаться только их графическое представление (рис. 4.21).

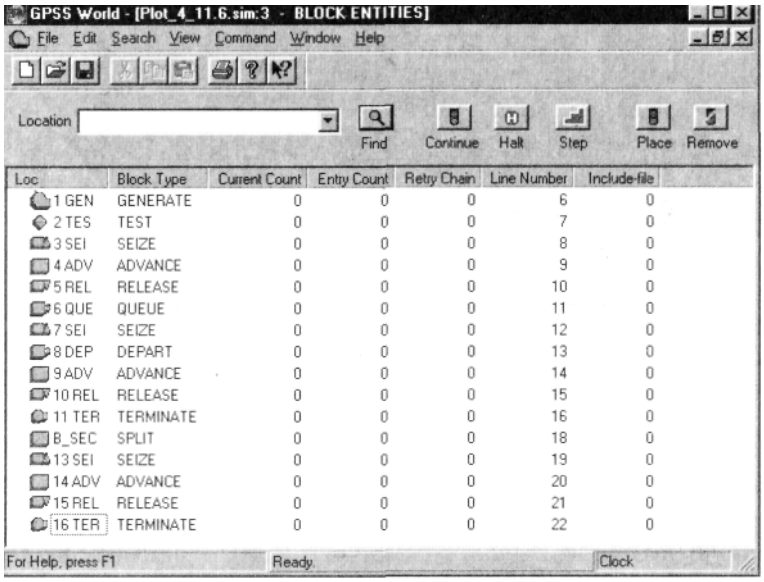


Рис. 4.20

Последний пункт меню ***View/Simulation Clock*** позволяет отображать часы модельного времени в нижнем правом углу главного окна.

Пункт главного меню ***Command*** (команда) используется для создания и управления объектами имитации. Пункт меню ***Command/Create Simulation*** (создать имитацию) вызывает транслятор для создания объекта имитации, который включает кроме GPSS-модели и файлы, связанные с ней. Ошибки трансляции должны быть исправлены (см. выше ***Search/Next Error).*** Команда меню ***Command/Retranslate*** доступна для выполнения повторной трансляции после исправления ошибок.

Команда ***Command/Repeat Last Command*** (повторить последнюю команду) – простой способ повторить некоторое действие для того же самого объекта. Остальные пункты меню ***Command*** выполняют команды, как описано выше в параграфе 4.27.

При работе с GPSS World пользователю доступны двадцать различных окон для наблюдения и взаимодействия с моделью в процессе имитации. Окна, отображающие визуальное состояние имитации, могут быть сохранены и распечатаны. Некоторые окна делают как бы мгновенный снимок состояния различных объектов имитации в некоторый момент модельного времени. Изображения в окнах изменяются динамически в интерактивном режиме взаимодействия с моделью. Следует отметить, что открытые динамические окна существенно замедляют скорость прогона модели.

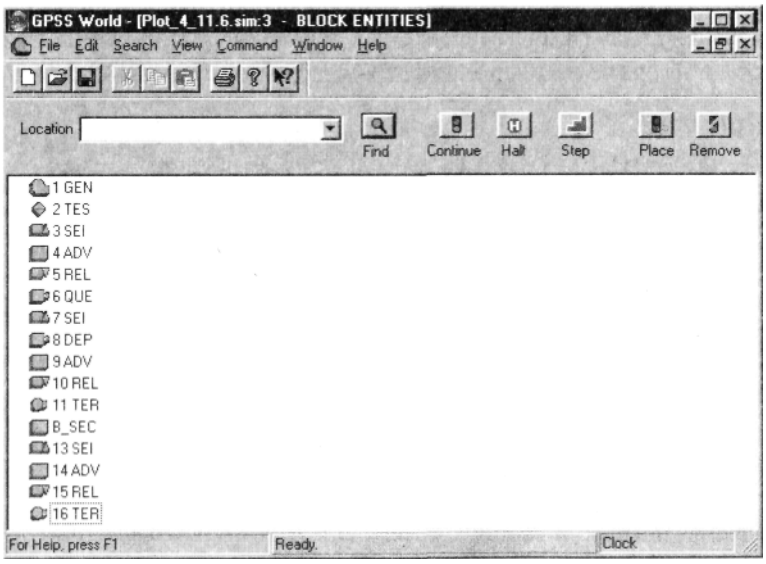


Рис. 4.21

Пользователю доступны следующие окна, позволяющие отображать мгновенное состояние системы:
Current Events Chain (список текущих событий);
Future Events Chain (список будущих событий);
Individual Transactions (отдельные транзакты);
Numeric Groups (числовые группы);
Transaction Groups (группы транзактов).

При моделировании может быть открыто любое число динамических окон для следующих объектов: блоков, устройств, МКУ, очередей, логических ключей, сохраняемых величин, матриц, таблиц.

Кроме того, можно воспользоваться окнами для графиков и выражений, что позволяет проследить изменения значений переменных во время имитации. Каждое окно графиков может отображать значения до восьми переменных, включая переменные интегрирования. Окна графиков имеют вертикальный и горизонтальный скроллинги для просмотра. Окно выражений может быть открыто в любой момент моделирования и позволяет просматривать значения любого числа PLUS-выражений.

В окне ***Future Events Chain*** отображается содержимое списка будущих событий. В нем отображаются транзакты с содержимым их параметров в порядке их расположения в таблице модельных событий.

Пример 4.75

```
TYP          FUNCTION    RN1,D4
.25,1/.50,2/.75,3/1,4
PROFIT       FUNCTION    P1,C2
1,5/4,20
             GENERATE    30,6,,,1
             SEIZE       DVC1
             ADVANCE     27,4
             RELEASE     DVC1
             PRIORITY    0
             ASSIGN      1,FN$TYP
             ASSIGN      PROF,FN$PROFIT
             SEIZE       DVC2
             ADVANCE     28,4
             RELEASE     DVC2
             TERMINATE

             GENERATE    24000
             TERMINATE   1
             START       1
```

Содержимое окна ***Future Events Chain*** показано на рис. 4.22. В отчете о работе модели информация об СБС будет выдана в таком виде:

FEC XN	PRI	BDT	ASSEM	CURRENT	NEXT	PARAMETER	VALUE
795	0	24005.221	795	9	10	1	3.000
						PROF	15.000
797	1	24009.728	797	0	1		
796	1	24015.417	796	3	4		
798	0	48000.000	798	0	12		

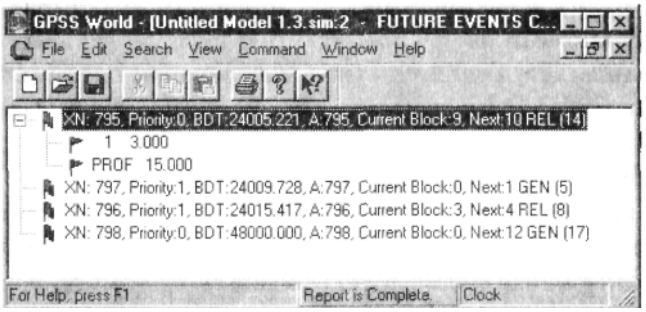


рис. 4.22

Содержание информации, которая выводится для транзактов из СБС следующее:

XN – номер транзакта;

PRI – приоритет транзакта;

BDT – таблица модельных событий – абсолютное модельное время выхода транзакта из СБС (и перехода транзакта в СТС);

ASSEM – номер семейства транзактов;

CURRENT – номер блока, где находится транзакт (0 – если транзакт еще не вошел ни в один блок модели);

NEXT – номер блока, в который должен войти транзакт;

PARAMETER – номер или имя параметра транзакта;

VALUE – значение параметра.

Окно графиков используется для графического отображения изменений значений СЧА, выбранных пользователем в процессе моделирования. Используя пункт меню ***Window/Simulation Window/ Plot Window...,*** можно построить до восьми таких графиков. Для организации вывода информации в графическом виде в GPSS World необходимо выполнить следующие шаги.

1. Создать модель, выбрав пункт меню ***Command/Create Simulation.***

2. Выбрать пункт меню ***Window/Simulation Window/ Plot Window...***

3.Заполнить поля в диалоговом окне ***Edit Plot Window*** (рис. 4.23) и нажать кнопку **ОК.**

4. Запустить процесс имитации, выбрав пункт меню ***Command/STAR T.***

Рассмотрим назначение полей диалогового окна ***Edit Plot Window*** (см. рис. 4.23):

1. Группа ***New Expression*** (новое выражение) предназначена для добавления нового выражения в список отображаемых выражений. После заполнения полей этой группы пользователь может нажать кнопку ***Plot*** для добавления введенного выражения в список отображаемых выражений. Пользователь может также нажать кнопку ***Memorize*** (запомнить) для сохранения выражения с целью дальнейшего использования. В поле ***Label*** задается имя выражения, **А** в поле ***Expression –*** само выражение пользователя.

2. В группе ***Window Contents*** (содержимое окна) отображается список выражений и задается ряд глобальных настроек графика. В поле ***Title*** задается заголовок графика, **А** в поле ***Time Range –*** длительность временного интервала, отображаемого на графике. Поля ***Min Value*** и ***Max Value*** определяют соответственно минимальное и максимальное значения отображаемой величины. Кнопка ***Remove*** используется для удаления выражения.

3.Группа ***Memorized Expressions*** (сохраненные выражения) содержит перечень сохраненных выражений пользователя.

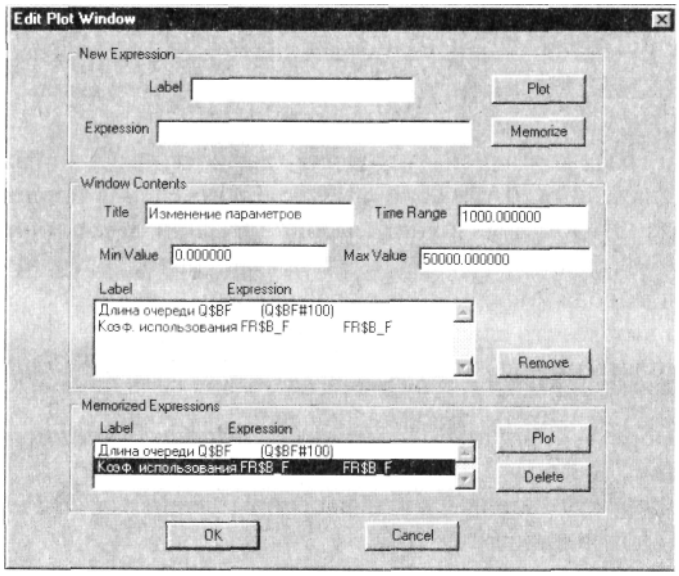


Рис. 4.23

Если значения отображаемого выражения выходят за пределы, указанные в параметрах ***Min Value*** **и** ***Max Value,*** либо длительность времени имитации превышает ***Time Range,*** пользователь может просмотреть интересующие его значения выражений, прокрутив изображение с помощью горизонтальных и вертикальных полос прокрутки.

Пример 4.76

Рассмотрим простейшую СМО вида D/D/1, работающую в режиме перегрузки. Система состоит из очереди и одного устройства, время обслуживания которого детерминировано и равно 11 единицам модельного времени. Транзакты поступают на вход системы каждые 10 единиц модельного времени. Продолжительность моделирования – 10000 единиц модельного времени.

Программа:

```
GENERATE    10
QUEUE       QB
SEIZE       B
DEPART      QB
ADVANCE     11
RELEASE     B
TERMINATE
GENERATE    10000
TERMINATE   1
```

В качестве выражения для визуализации выберем длину очереди **QB.** Заполнив окно ***Edit Plot Window*** (см. рис. 4.23) и выполнив команду **START 1,** получим график, представленный на рис. 4.24.

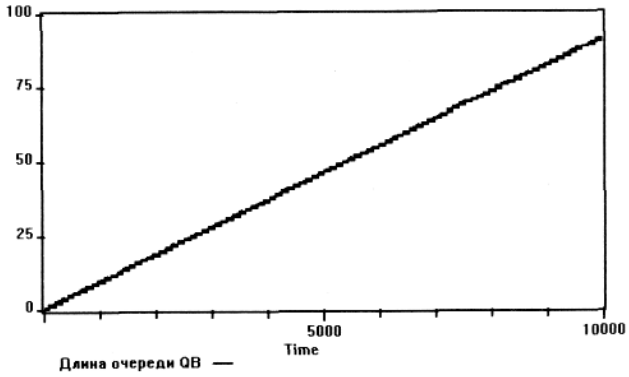


Рис. 4.24

Как и следовало ожидать, длина очереди в данной системе с течением времени неограниченно возрастает.

Пример 4.77

```
XPDIS       FUNCTION    RN1,C24
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915/.7,1.2/.75,1.38
.8,1.6/.84,1.83/.88,2.12/.9,2.3/.92,2.52/.94,2.81/.95,2.99/.96,3.2
.97,3.5/.98,3.9/.99,4.6/.995,5.3/.998,6.2/.999,7/.9998,8
*************************************************
            GENERATE    300,100
            TEST E      F$B_P,0,B_SEC
            SEIZE       B_P
            ADVANCE     190,FN$XPDIS
            RELEASE     B_P
            QUEUE       BF
            SEIZE       B_F
            DEPART      BF
            ADVANCE     300,FN$XPDIS
            RELEASE     B_F
            TERMINATE   1
*************************************************
B_SEC       SPLIT       2,,1
            SEIZE       B_S
            ADVANCE     500,FN$XPDIS
            RELEASE     B_S
            TERMINATE   1
```

Для получения информации об изменении длины очереди и коэффициента использования устройства **B_F** заполним диалоговое окно ***Edit Plot Window,*** как показано на рис. 4.25.

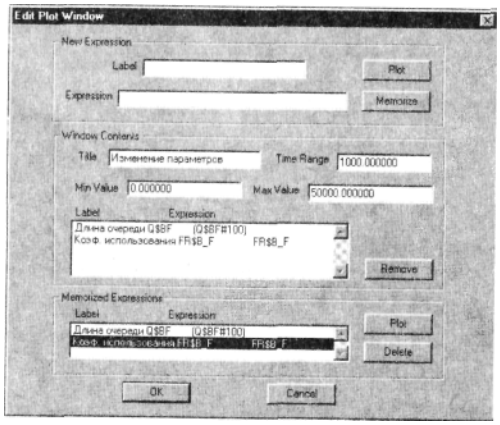


Рис. 4.25

Перейдем в окно графиков и начнем моделирование, выдав команду **START 200.** В окне графиков увидим, как изменяются во времени длина очереди **BF** (умноженная на 100) и коэффициент использования устройства **B_F** (рис. 4.26).

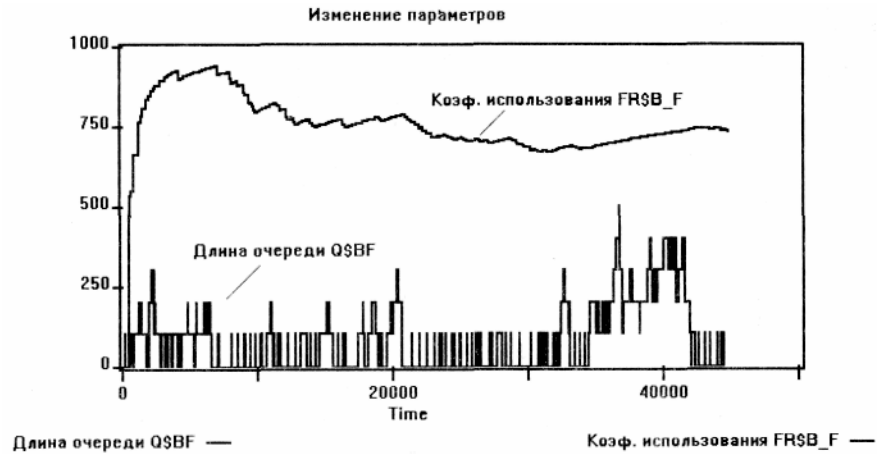


Рис. 4.26

4.29. Отличия между GPSS World и GPSS/PC

Язык GPSS World является преемником языка GPSS/PC, который был разработан в 1984 году для DOS. В отличие от GPSS/PC GPSS World работает в многозадачном режиме под управлением Windows и позволяет использовать многооконный интерфейс. Особенностью системы GPSS является то, что текст GPSS-модели создается и (или) изменяется, **А** потом транслируется для создания имитации, которая отличается от программного файла в GPSS/PC.

Возможно, самым броским отличием от GPSS/PC является отсутствие нумерации строк, фактически номеров строк. Это означает, что позиция блоков определяется не номером строки блока, **А** лишь ее относительной позицией в файле (файлах) модели при трансляции. Новая команда **INCLUDE** позволяет включать в модель или текстовые файлы моделей или последовательности команд, что может переопределить позицию блока при моделировании. Хотя блоки больше не могут быть вставлены в модель во время имитации, тем не менее, время трансляции настолько мало, что для большинства целей достаточно снова выполнить моделирование после внесения изменении в структуры модели. Даже в этом случае высокий уровень интерактивности «ручной имитации» сохранен путем использования меню команд.

В GPSS World при вычислениях не осуществляется автоматическое отбрасывание дробной части. Чтобы промежуточные числовые результаты были целыми, необходимо явным образом использовать процедуру **INT**( ). Эта процедура может применяться во всех выражениях, даже в операторах **VARIABLE** и **BVARIABLE.**

При использовании процедуры в операндах она должна заключаться в круглые скобки, например

```
SAVEVALUE   l,(INT(4.54))
```

Так СЧА могут теперь быть целыми, действительными числами или строками в зависимости от СЧА. Стандартно числовые атрибуты, которые возвращают значения в тысячных долях, дают действительные числа двойной точности от 0 и до 1000 включительно. Старые усеченные и целочисленные СЧА могут использоваться, если трансляция выполняется в режиме совместимости с GPSS/PC.

Мультизадачная архитектура GPSS World обусловила и другие изменения, например, для выполняемой модели могут посылаться сообщения или команды для изменения состояния имитации.

Блок **HELP** не используется, он заменен файловыми процедурами языка **PLUS**, которые поддерживают связь с внешними файлами и программами.

Команда управления **END** заменена командой **EXIT,** которая завершает сеанс работы (**END** – теперь ключевое слово языка PLUS).

Знак открытия файла **@** заменен инструкцией **INCLUDE,** которая использует имя файла как операнд.

Изменение старых программ GPSS/PC. Существует несколько способов, которые позволяют легко переносить GPSS/PC программы в новую систему. Для точного дублирования результатов необходимо определить правильность модели при выполнении в GPSS World, после чего просто перейти в новый режим работы, который предусматривает таймер с плавающей запятой и имеет другие особенности.

Если необходимо, можно убрать нумерацию строк (она теперь игнорируется). Не следует использовать старые программные файлы, которые позволяли редактировать, вставлять и удалять строки модели GPSS/PC с заданным номером.

В программу необходимо внести следующие изменения:

1) заменить команду **@** на команду **INCLUDE;**

2) названия имен файлов задавать в двойных кавычках в команде **INCLUDE;**

3) изъять все команды **REPORT;**

4) изъять команды **END** и любые имена типа **BEGIN, COUNT, NORMAL** и т.п. (они теперь совпадают с ключевыми словами GPSS World);

5) блоки **HELP** заменить файловыми процедурами языка **PLUS**;

6) удалить блоки **MOVE** (позиции в окне больше не поддерживаются при анимации);

7) полностью записать ключевые слова операторов **VARIABLE, FVARIABLE и BVARIABLE** (не допускается сокращение ключевых слов).

Наверное, следует считать самым безопасным методом изъятие всех старых команд запуска из старого GPSS/PC программного файла, пока не будут проведены все необходимые изменения в модели.

Строгое дублирование результатов. Большинство GPSS/PC программных файлов могут выдавать точно такие же результаты, которые выдает GPSS World. Через расхождение в округлении действительных чисел в вычислениях с плавающей запятой необходимо выполнить несколько дополнительных модификаций для GPSS/PC программного файла в дополнение к уже указанным особенностям. Чтобы получить результаты, которые статистически не отличаются от полученных в GPSS/PC, необходимо при использовании коммерческой версии GPSS World выполнить такие действия.

1. Установить режим совместимости GPSS/PC в параметрах настройки модели, который заставит GPSS World использовать целочисленное время и откидывать дробную часть:

1) выбрать ***File/Open,*** чтобы открыть объект модели;

2) выбрать ***Edit/Settings.***

Тогда на первой странице появится переключатель маркирования GPSS/PC совместимости.

2. Установить потоки случайных чисел для **GENERATE, ADVANCE** и **TRANSFER** в соответствии с файлом **settings.gps** для GPSS/PC. Потом установить поток случайных чисел, связанный со временем, в единицу. Для этого необходимо выбрать пункт меню ***Edit/Settings*** и закладку ***Random Numbers,*** а дальше установить входные поля случайных поточных чисел для **GENERATE, ADVANCE, TRANSFER,** а также связь со временем.

3. Убедиться, что все начальные числа в любых командах **RMULT** меньше, чем 100000000.

4. Изъять все зависимости от идентификатора и заданные по умолчанию системные значения. Например, **RN$IDENTIFIER** задает поток случайных чисел с отличными от системных значений, которые задаются по умолчанию. Описание этого объекта заменить.

5. Не вызывать случайную функцию из функции. Заменить все подобные операторы.

6. Заменить все дробные части, которые встречаются в блоках **TRANSFER,** целыми числами «частями тысячи».

7. Не изменять операторы описания координат функций **FN$SNORM и FN$XPDIS.**

Специальная программа DOS, которая называется **pcaid.exe,** поможет изменить GPSSA>C программный файл. Она исправит номера строк и напомнит, что при наличии инструкции **FUNCTION** в GPSS, их надо исследовать. Можно запустить эту программу в окне DOS.

В любом случае следует иметь в виду, что имитации – это не простые текстовые файлы в GPSS World. Они содержат параметры настроек, закладки и списки результатов, которые не могут интерпретироваться внешними программами обработки текстов. По этой причине по окончании изменений в GPSS/ЛРС программном файле необходимо использовать программу обработки текстов (например, Блокнот или Word) для копирования текста программы в буфер обмена Windows. Для этого в GPSS World надо открыть новую модель из пункта меню *File/New* и вставить текст программы, используя ***Edit/Paste.***

Этот процесс создает правильный объект модели GPSS World, основанный на начальной модели. Теперь можно получить соответствующие результаты для имитации GPSS World для модели GPSS/PC.

Некоторые версии GPSS World не совместимы между собой. Поэтому необходимо перетранслировать GPSS-модели для выполнения в других версиях. Для этого на компьютере необходимо иметь обе версии. Открыв в старой версии GPSS-модель, необходимо скопировать ее через буфер обмена в новую версию и перетранслировать.

ГЛАВА 5. МОДЕЛИРОВАНИЕ ВЫЧИСЛИТЕЛЬНЫХ И ОПЕРАЦИОННЫХ СИСТЕМ

Информационные технологии связаны с процессами передачи, обработки и хранения информации, которые выполняются компьютерными системами и системами передачи информации. Задания для самостоятельной работы в главе 7 включают задачи анализа простых компьютерных систем, сетей и систем передачи данных. Предполагается, что будет использоваться язык моделирования GPSS, хотя для этого имеются разные специализированные пакеты моделирования, например, предназначенные для моделирования передачи пакетов с разной организацией сетевых протоколов и технологий. Задача этой главы – дать краткие теоретические сведения о работе компьютеров, ОС и сетей для их успешного моделирования.

5.1. Операционные системы компьютеров

Любой компьютер имеет один или несколько микропроцессоров, память, устройства ввода-вывода или периферийные устройства типа дисплеев, клавиатуры, дисков, принтеров и сетевых устройств. Компьютерное моделирование применяется для оценки производительности компьютерных систем. В большинстве случаев для этого используют имитационное моделирование, но для решения некоторых задач анализа на предыдущем этапе применяют операционный анализ сетей СМО. Память моделируют для оценки операций с общей памятью или ее уровней (кэш-память, оперативная память, память второго или третьего уровней, которая организовывается на диске).

За организацию обработки информации в компьютерной системе отвечает ОС, которая распределяет ресурсы и руководит процессами распределения этих ресурсов между программами, выполняемыми компьютером. К этим программам принадлежат не только программы пользователей, но и системные программы ОС. Ресурсы ОС – это различные устройства (процессор, дисплей, клавиатура, накопители на магнитных дисках, модемы, принтеры и т.д.), **А** также память, файлы, программные модули. В случае занятости нужного ресурса, который требует программа, к нему создается очередь. Таким образом, если программы пользователей, которые выполняются компьютером, используют одинаковые ресурсы в некоторые промежутки времени, то выполнение таких программ будет существенно задерживаться.

Типичные функции ОС: выбор программы, которая будет выполняться процессором, управление общей памятью, управление операциями ввода-вывода, обработка прерываний.

Компьютерная система обычно работает в мультипрограммном или в многозадачном режиме. Классическое мультипрограммный режим построен на принципе совместимости операций, которые выполняет процессор, и операций ввода-вывода, которые выполняет программируемый контроллер или процессор ввода-вывода. Например, если компьютер выполняет вывод на печать результатов некоторой программы, то пользователь одновременно может выполнять другую программу.

Многозадачный режим построен на принципе распределения времени работы процессора между несколькими выполняемыми программами. Для каждой программы выделяется свой промежуток времени (квант времени), в течение которого она выполняется процессором. Если за это время программа не закончилась, то она прерывается и процессор начинает выполнять другую программу. Обычно прерванная программа становится в очередь к процессору последней. Таким образом, каждая выполняемая программа, несколько раз получает процессорное время для расчетов, до того как покидает компьютер. Если программа во время выполнения процессором прерывается для операций ввода-вывода (например, для записывания информации на диск), то она выводится из очереди к процессору, пока не закончит операцию записывания. Остаток процессорного времени от кванта не сохраняется, и по окончании операции записывания эта программа, когда ей будет предоставлен процессор, получает новый квант времени.

При моделировании режима распределения времени процессора возникает задача о назначении промежутков времени для определенных групп программ пользователей. Использование режима распределения времени содействует тому, что более короткие задания пользователей выполняются раньше, поскольку они набирают необходимое количество квантов процессорного времени быстрее, чем длинные задания. Это вынуждает пользователей делать задания для выполнения компьютером более короткими по времени выполнения.

Если взять очень маленький квант времени, то программы пользователей будут очень часто прерываться, **А** каждое прерывание требует определенного времени на его обработку компьютером. Если квант времени будет стремиться к бесконечности, то программы будут обрабатываться процессором по принципу «первая поступила – первая выполняется».

В любом режиме работы (мультипрограммном или многозадачном) компьютер с одним процессором в каждый момент времени выполняет только одну программу (одну команду). Если используется мультипроцессорная система с несколькими одинаковыми процессорами, то становится возможным выполнения нескольких программ на разных процессорах. Большей частью такая система моделируется как многоканальная СМО в отличие от СМО с одним устройством обслуживания, которая моделирует работу однопроцессорной системы.

Современные ОС (Unix, Windows и др.) используют виртуальную организацию памяти для сохранности выполняемых программ. Принцип виртуальной памяти основан на том, что только некоторые части программ находятся в оперативной памяти, другие – на диске. Во время работы

компьютера, если запрашивается область памяти, которая находится на диске, то эта часть программы загружается с диска в оперативную память. Если оперативной памяти недостаточно, из нее вытесняется некоторая часть другой программы, преимущественно той, к которой дольше всего не было обращений. Эти операции перемещения частей программ требуют некоторого времени, связанного с поиском нужной части программы на диске и ее считывания в память. Для ускорения операций поиска некоторые части программ могут находиться в кэш-памяти, которая обычно ускоряет операции поиска. Управление памятью требует разработки некоторой стратегии вытеснения и замещения частей программ, которая закладывается в ОС в виде алгоритма.

При выполнении операций ввода-вывода для записи или считывания данных с магнитных носителей информации, эти операции большей частью осуществляются через буфер ввода-вывода, где предварительно размещаются данные, что позволяет существенно ускорить эти операции. Выбор емкости буфера влияет на скорость выполнения операций ввода-вывода.

Обмен данными между памятью и внешними устройствами осуществляется через контроллеры ввода-вывода, но скорость их работы на порядок больше скорости электромеханических устройств. При моделировании можно считать, что эти операции выполняют сами устройства. Поиск информации на магнитных дисках (гибких или жестких) связан с перемещением головок на нужный сектор и с непосредственным выполнением операций записи или считывания данных, которые зависят от скорости вращения диска. Поэтому для моделирования времени поиска информации (перемещения головок) используют равномерное распределение, для которого минимальное и максимальное времена берут из технических характеристик устройств.

Таким образом, работу компьютерной системы, управляемой ОС, можно представить в упрощенном виде как работу сети СМО.

При моделировании обычно интересуются временем пребывания программы пользователя в компьютерной системе (от ввода программы или запроса на ее выполнение до получения результатов ее работы или ответа) или числом программ, выполненных системой за единицу времени. Типичная программа или запрос пользователя может описываться: идентификатором программы, временем поступления в систему, емкостью нужной памяти, количеством запросов к устройствам ввода-вывода и распределением их по времени, количеству данных, которые вводятся и выводятся на тех или иных устройствах, **А** также более детальной информацией в зависимости от цели моделирования.

5.2. Сети и системы передачи данных

Интеграция систем передачи информации (телефонная и радиосвязь) и компьютерных систем обусловила создание сетей поддержки связи между компьютерами (отдельные входы в системы, передача файлов, электронная почта и т.п.). В современных цифровых сетях передачи информации нет различия между передачей голоса и данных, поскольку они способны передавать данные, звуковой и видеосигнал как один пакет данных. Это так называемые широкополосные цифровые сети с предоставлением комплексных услуг.

Сети иногда делят по их географическому размещению на локальные, региональные и глобальные. Локальные сети имеют ограниченное количество компьютеров, и связь между ними осуществляется по физической линии связи (проводам, коаксиальному кабелю). Региональные сети объединяют несколько локальных сетей в рамках области или региона и используют, главным образом, телефонные сети или волоконно-оптические кабели. Глобальные сети передают данные не только по телефонным каналам, но и по спутниковым и радиоканалам.

Вычислительная сеть представляет собой систему станций на базе ЭВМ (у***злы сети),*** взаимодействующих между собой через каналы передачи данных, используемые для обмена информацией. Для обеспечения безошибочной и максимально удобной передачи информации сетевые операции регулируются набором правил и соглашений, названных ***сетевым протоколом.*** Протокол определяет типы разъемов и кабелей, сигналы, форматы данных и средства проверки ошибок, **А** также алгоритмы для сетевых интерфейсов и узлов по стандартным (в пределах сети) принципам подготовки сообщений, передачи и анализа на разнообразных уровнях детализации.

Сети имеют некоторые общие компоненты, функции и характеристики, **А** именно:

– серверы (server) – компьютеры, которые предоставляют свои ресурсы сетевым пользователям;

– клиенты (client) – компьютеры, которые осуществляют доступ к сетевым ресурсам, предоставленным сервером;

– среда – средства соединения компьютеров;

– совместно используемые данные – файлы, предоставленные серверами по сети;

– совместно используемые периферийные устройства (принтеры, приводы CD-ROM и прочие ресурсы, предоставленные серверами);

– ресурсы файлы, прикладные программные комплексы, принтеры и другие элементы, используемые в сети.

Принцип действия локальной сети зависит от расположения и соединения отдельных сетевых компонентов. Средство соединения компьютеров в сети называется топологией. В локальных сетях компьютеры могут работать в режимах:

– клиентов, которые используют сетевые ресурсы, но не предоставляют свои ресурсы другим компьютерам;

– одно-ранговых узлов, работающих с сетевыми ресурсами, которые разрешают доступ других машин к своим ресурсам;

– серверов, которые предоставляют ресурсы сети другим машинам. Каждый из этих режимов определяется типом применяемой на компьютере ОС. На серверах функционируют такие ОС, как Windows NT или Unix. Различают следующие типы сетей:

– псевдосети типа ПК-ПК;

– одно-ранговые сетевые среды (peer-to-peer), в которых нет серверов и ресурсы распределяются по независимым узлам;

– серверные (или клиент-серверные), которые имеют клиентов и обслуживают их на сервере;

– комбинированные (гибридные) сети – сети клиент-сервер и одноранговые с ресурсами, которые распределяются.

Большинство сетей на самом деле являются комбинированными.

Обмен информацией между двумя компьютерами может осуществляться через разъемы последовательного интерфейса обоих компьютеров с помощью специального кабеля нуль-модема. В этом случае для осуществления передачи данных нужно лишь соответствующее терминальное программное обеспечение, поскольку подобный принцип соединения двух компьютеров характерен, в первую очередь, для компьютеров типа ***laptop*** и ***notebook.*** При использовании параллельной передачи данных повышается скорость обмена информацией между обоими компьютерами, тем не менее, из технических соображений последовательный кабель может быть значительно длиннее, чем параллельный. Во время передачи данных оба процессора блокируются и не могут выполнять другие задания.

Минимальное количество линий для организации двусторонней связи компьютера без модемного устройства – три. Тогда перенесение информации становится возможным через непосредственный доступ к портам, для чего необходимо правильно их настроить и запрограммировать собственный драйвер с обработчиком прерываний. Схемы трехпроводного соединения компьютеров требуют для надежного перенесения данных организации программного подтверждения, которое снижает скорость передачи.

Серверные сети (сети с обслуживающими узлами) функционируют при наличии клиентов. Клиенты обращаются к серверу, который предоставляет им разные средства (например, печать или работу с файлами). Клиентские компьютеры обычно менее мощные, чем серверы.

Большинство сетей используют ***выделенные серверы.*** Выделенным называется такой сервер, который функционирует только как сервер и не выполняет функций клиента или рабочей станции. Такие серверы специально приспособлены для быстрой обработки запросов от сетевых клиентов и управления защитой файлов и каталогов. Сети на основе сервера стали промышленным стандартом.

С увеличением размеров сети и объема сетевого трафика необходимо увеличивать количество серверов. Распределение заданий между несколькими серверами гарантирует, что каждое задание будет выполняться наиболее эффективным средством из всех возможных. Наиболее распространенными серверами являются следующие:

– файловые серверы;

– серверы печати;

– серверы приложений;

– серверы сообщений;

– серверы баз данных.

Все сети строятся на основе трех базовых топологий: шины, звезды, кольца.

Если компьютеры подключены вдоль одного кабеля (сегмента), топология называется ***шиной,*** если же они подключены к сегментам кабеля, который выходит с концентратора (hub), топология называется ***звездой.*** Если кабель, к которому подключены компьютеры, замкнут в кольцо, такая топология называется ***кольцом.***

Базовые топологии сами по себе достаточно просты, но часто встречаются довольно сложные комбинации, которые объединяют свойства нескольких топологий.

В типичной сети с шинной топологией кабель имеет одну или более пар проводников, **А** активные схемы усиления сигнала или передачи его от одного компьютера к другому отсутствуют. Такая шинная топология – пассивная. Если один компьютер посылает сигнал по кабелю, все другие узлы получают эту информацию, но только один из них (адрес которого совпадает с адресом в сообщении) принимает ее. Другие не принимают сообщения.

В каждый момент времени отправлять сообщение может только один компьютер, поэтому количество подключенных к сети компьютеров значительно влияет на скорость сети. Перед передачей данных компьютер вынужден ожидать освобождения шины. Указанные факторы действуют также в кольцевой и звездообразной сетях. Примером недорогой сети с шинной топологией может быть Ethernet. Звездообразная топология применяется в сосредоточенных сетях, в которых конечные точки доступны из центрального узла, **А** также в случаях, когда предполагается расширение сети и нужна высокая надежность.

Каждый компьютер в сети с топологией типа «звезда» взаимодействует с центральным концентратором, который передает сообщения всем компьютерам (в звездоподобной сети с широковещательной рассылкой) или только компьютеру-адресату (в коммутированной звездоподобной сети).

Активный концентратор регенерирует электрический сигнал и посылает его всем подключенным компьютерам. Такой тип концентратора часто называют многопортовым повторителем. Для работы активных концентраторов и коммутаторов необходимо питание от сети. Пассивные концентраторы (например, коммутационная кабельная панель или коммутационный блок) действуют как точка соединения, не усиливая и не регенерируя сигнал.

В сети с кольцевой топологией каждый компьютер соединяется с тем компьютером, который ретранслирует его информацию, полученную им от первого. Благодаря такой ретрансляции сеть является активной, для нее не характерна потеря сигнала, как в сетях с шинной топологией.

Некоторые сети с кольцевой топологией используют метод эстафетной передачи. Специальное короткое сообщение – маркер – циркулирует по кольцу до тех пор, пока компьютер не передаст информацию другому узлу. Он модифицирует маркер, добавляет электронный адрес и данные, **А** потом отправляет его по кольцу. Каждый компьютер последовательно получает данный маркер с добавленной информацией и передает его соседнему компьютеру до тех пор, пока электронный адрес не станет совпадать с адресом компьютера-получателя или маркер не возвратится к отправителю. Компьютер, который получил сообщение, посылает отправителю ответ, который подтверждает, что сообщение принято. Тогда отправитель создает еще один маркер и отправляет его в сеть, давая возможность другим станциям перехватить маркер и начать передачу. Маркер циркулирует по кольцу до тех пор, пока любая из станций не будет готова к передаче и не захватит его.

Все эти события происходят очень быстро: маркер может пройти кольцо с диаметром 200 м приблизительно 10000 раз в секунду. В некоторых еще более быстрых сетях циркулирует сразу несколько маркеров. В других сетевых средах применяются два кольца с циркуляцией маркеров в противоположных направлениях. Такая структура обеспечивает восстановление сети в случае возникновения отказов.

Примером быстрой волоконно-оптической сети с кольцевой топологией является **FDDI (Fiber Distributed Data Interface) –** распределенный интерфейс передачи данных по волоконно-оптическим каналам: стандарт, разработанный ANSI и модифицированный IEEE/ISO.

Как среду передачи данных в электронной сети можно использовать: коаксиальный кабель, витую пару проводов, волоконно-оптический кабель, инфракрасное излучение или радиодиапазон эфира.

Сегодня преобладающая часть компьютерных сетей использует для соединения в большинстве случаев провода или кабели.

Модуль данных, передаваемый по сети, различен при различных технологиях организации сети. В датаграммных сетях переданный модуль данных – это полное сообщение, ограниченное заданным максимальным размером. Размер модуля данных изменяется также от одного сообщения к другому.

Можно разделять каждое сообщение на пакеты установленного размера и передавать их по сети. Сообщения, которые превышают отдельный пакет, разделяют на пакеты до того, как они посылаются в сеть. Пакеты можно передавать по сети по одному так, что они будут достигать адресата не в том порядке, в котором были отправлены.

Пакеты собираются у адресата после того, как сообщение получено пользователем. Такой же протокол предусмотрен и для ретрансляции утерянных пакетов.

Для передачи информации в пределах сети предназначена физическая среда. Она может использовать двухточечную связь или радиовещательную среду. В двухточечной связи – это специализированное физическое подключение (например, провод или волоконно-оптический кабель), между каждой парой компонентов, таких как переключатели, компьютеры, устройства ввода-вывода и др.

Если подключение поддерживает связь только в одном направлении и не разрешает одновременную передачу данных в обоих направлениях, то для организации связи не нужно никакого подтверждения. Если поддерживается связь во времени в обоих направлениях, то необходим протокол для переключения направления, чтобы обеспечить использование двусторонней связи.

Радиовещательная среда позволяет большому множеству компонентов использовать ее как среду связи. Но при этом нужен протокол, чтобы предотвратить влияние передатчиков друг на друга. В этом случае можно использовать протокол MAC как среду управления доступом. Например, широко используемая технология локальной сети базируется на радиовещательной среде Ethernet. Здесь все компоненты связаны с отдельным проводом, например, коаксиальным кабелем. Протокол MAC использует несколько простых правил: передатчик не передает данные до тех пор, пока провод используется другим передатчиком; если два передатчика одновременно начинают передавать данные, каждый из них знает об этом, контролируя провод и наблюдая за данными, переданными по проводу (определяется несоответствие переданных данных); при появлении столкновения передатчик останавливает передачу и ждет некоторое время, **А** потом ретранслирует (повторяет) сообщение. Подобный протокол часто используется для управления доступом к сети радиосвязи.

Другая широко используемая технология локальной сети – это эстафетное кольцо. В конфигурации кольцевой сети точка-точка маркер, как и маркер с данными, передаются в одном направлении по кольцу.

Кольцо подобно ленте конвейера последовательно перемещает данные от одного компьютера к другому по кругу. Лента конвейера разделена на «слоты», где каждый слот может содержать одиночную часть блока установленного размера, например, пакет данных. Каждый компьютер исследует данные, которые проходят по ленте конвейера. Если адреса данных соответствуют адресу компьютера, то данные отделяются от ленты и слот становится пустым. Если адреса не соответствуют компьютеру, то данные продолжают движение по ленте конвейера. Чтобы передать данные, компьютер находит пустой слот и заполняет его данными для передачи.

Упражнения для моделирования сети, предложенные в этой книге, содержат анализ простых сетей и непосредственную передачу данных в пределах сети. Используется универсальный язык моделирования GPSS World для разработки модели, хотя существуют коммерческие пакеты моделирования, разработанные для определенных стандартных сетей. Они содержат типичные модели передачи пакетов для разной организации сетевых протоколов и технологий, используемых в данное время, однако, могут требовать много времени для разработки имитационной модели сети.

5.3. Проблемы моделирования компьютеров и сетей

При проектировании сети необходимо обеспечить выполнение противоположных требований. Для решения этой проблемы используют имитационное моделирование. Первое задание проектировщика – разобраться в работе распределенной вычислительной системы, которая образует сеть из отдельных компонентов, таких как компьютеры, принтеры и прочие устройства ввода-вывода. Такую сеть можно моделировать как сеть СМО и делать расчеты с помощью операционного анализа, ориентированного на вычислительные системы [9].

Типичный вопрос, который можно было бы задать при моделировании: каким будет среднее время задержки при передаче большого файла, если добавить в систему пять новых компьютеров, которые создают дополнительный трафик в сети? Если цель заключается в том, чтобы оптимизировать работу

внутренних компонентов в пределах сети непосредственно (например, коммутатора), то ее надо моделировать как комплекс сетей СМО, который, возможно, содержит сотни или даже тысячи СМО.

Сетевого проектировщика волнуют, прежде всего, такие вопросы: как управлять перегрузкой в сети; какую емкость должен иметь буфер для каждого переключения направлений передачи; какие приоритеты должны назначаться на разные виды связи, которые используются в сети и т.п.

Некоторый другой набор вопросов касается лиц, которые конфигурируют сеть для определенного узла (например, компании или города), **А** именно: какая топология сети должна использоваться; какой ширины полосы частот должны быть распределены по индивидуальным каналам.

Проектировщикам, прежде всего, следует сосредоточиться на том, каким будет среднее время ответа на запрос в сети и как изменится время ответа сети на запрос в зависимости от разнообразного распределения пакетов с разными схемами обслуживания. Это требует определения времени ожидания, коэффициентов использования компонентов, времени переключения коммутаторов и времени задержки ответов на транзакцию, вызванную переключением и т.п. При использовании передачи пакетов целесообразно рассмотреть такие показатели работы, как время ответа, производительность, процессорное время для обработки пакета. Время ответа имеет несколько разных определений. Упрощенное определение может звучать так: время ответа – это время от запроса пользователя до ответа системы. Такое определение игнорирует время ввода запроса и вывода ответа. В системах с распределением времени для интерактивных пользователей определяется время ответа от окончания ввода данных до окончания вывода, **А** для пакетных систем – от предоставления запроса до завершения вывода.

Два важных показателя, которые часто используются для характеристики работы сети или работы системы в целом, – это время ожидания и производительность. Время ожидания означает задержку, связанную с выполнением действия, например, время от момента, когда первый бит сообщения передан в сеть, до момента, когда последний бит появился у конечного адресата. Точное определение зависит от целей изучения и моделирования системы.

Время ожидания часто упоминается как время ответа, в особенности, когда обсуждается работа системы. Простое определение времени ответа – это время от момента, когда пользователь запрашивает выполнение операции до момента, когда система выдает ответ (например, время от запроса печати файла до получения полного отпечатанного файла).

Производительность обычно определяется как скорость прохождения запросов или как промежуток времени, за который запрос обслуживается системой. Интерактивную производительность определяют в запросах за секунду, тогда как пакетную – в заданиях за секунду. Производительность процессора определяют в миллионах команд в секунду (MIPS) и миллионах операций с плавающей запятой в секунду (MFLOPS), производительность системы диалоговой обработки запросов – в транзакциях в секунду (TPS).

Производительность – это количество запросов пользователей, которое может быть удовлетворено за единицу времени. В сетях связи это относится к числу бит информации (всего или для каждого пользователя), которое может быть передано через сеть за секунду.

При исследовании работы системы производительность характеризуется количеством заданий (например, заданий для вывода на печать), выполняемых за единицу времени.

Время ожидания и производительность – важные показатели работы. В качестве примера рассмотрим трубу, которая подает воду в дом. Время ожидания пока вода достигнет дома от насосной станции зависит (среди других факторов) от расстояния между станцией и домом. Производительность определяется количеством воды, которая вытекает из крана каждую секунду, и размером (поперечным сечением) трубы. Продолжительное время ожидания не обязательно предполагает низкую производительность. Например, дом может быть очень далеко от насосной станции, но производительность трубы очень большая. Точно так же сети, которые используют спутники, могут обеспечить продолжительное время ожидания, большую ширину полосы частот и высокую производительность.

Важным показателем является использование ресурса. Использование ресурса рассматривают как часть времени, расходуемую на обслуживание запроса (например, часть времени принтера, занятого печатью задания). Остаток от этого времени называют временем простоя. Для процессоров можно определить коэффициент использования как отношение времени простоя ко времени занятости, для памяти – как часть ресурса, которая занята на текущий момент времени (использование определяется как средняя часть, используемая за интервал времени).

Понятно, что коэффициент использования должен быть между 0,0 (полностью не используется) и 1,0 (всегда занят). По этому показателю можно определять критические параметры в системе или ее предельные возможности. Некоторые ресурсы не могут полностью использоваться в любой момент времени (например, не вся память компьютера используется в данный момент времени). В этом случае коэффициент использования определяется как средняя часть занятого ресурса за интервал времени. Например, если память используется приблизительно на 50%, то есть половина памяти компьютера занятая, **А** половина – доступна для использования другими программами.

Другие часто используемые показатели связаны с надежностью (определяют, например, среднее время между ошибками) или с доступностью, которая определяет среднее время между неудачными попытками. Моделирование часто применяется для оценки надежности системы, то есть среднего времени между отказами компонента или системы.

ГЛАВА 6. ОСНОВЫ МОДЕЛИРОВАНИЯ ПРОЦЕССОВ

6.1. Производственные процессы

Производственные процессы ориентированы на выпуск разнообразной продукции или изделий, которые могут изготавливаться в дискретном или непрерывном поточном режиме. Такие операции, как разделение на группы, объединение групп, сборка, разборка, монтаж, контроль качества и устранение брака представляют собой типичные функции, реализованные дискретными производственными процессами. Для того, чтобы точно промоделировать эти функции, модель должна отслеживать информацию об отдельных объектах потока, а также их атрибуты. Кроме того, в процессе создания модели важно учитывать правила построения очередей, последовательность выполнения операций, которые задаются маршрутными технологическими картами, а также моделирование простоя. Иногда при моделировании важно воссоздать расписание работы персонала во времени.

Примерами производственных процессов служат: выполнение заказов, работа отдела снабжения, расчеты нужных мест складирования и управления запасами, определение технологического маршрута выполнения заказов, формирование счетов к оплате или обработка требований и др. Таким образом, при имитации необходимо наиболее точно воссоздавать процессы, которые наблюдаются на практике.

Имитационное моделирование производственных процессов нацелено или на определение оптимальных параметров и режимов работы, или на проектирование наиболее подходящей структуры производственного процесса. Используя моделирование, можно уточнять и определять вероятностные закономерности, выявлять конкретные взаимосвязи и взаимоотношения (например, между степенью загрузки производственного участка, средним временем простоя заготовок и необходимым средним временем технологической операции и др.).

Технологические производственные системы разделяют на отдельные элементы: технологические агрегаты, испытательные стенды, транспортные средства, рабочие места, склады и т.п. Производственный процесс «движется» сквозь эти элементы и «потребляет» производственные ресурсы во времени (при выполнении операции).

Таким образом, производственный процесс отображается как некоторый материальный поток, объединяющий все без исключения продукты, которые есть в производственной системе. Все операции, с задержками по времени выполнения операции, независимо от того, изменяется ли состояние продукта в потоке. Например, если такие операции, как транспортирование или испытание, на состояние продукта не влияют, то выполнение таких операций, как монтаж или демонтаж, обработка на станках, сварка и прочие, изменяет состояние продукта.

Для моделирования ресурсов средствами языка GPSS в производственном процессе обычно используют устройства обслуживания (блоки **SEIZE, RELEASE**), если это отдельный ресурс, или МКУ (блоки **ENTER, LEAVE**), если это несколько однотипных ресурсов, склад или бункер.

Продукт в потоке бывает единичный (например, заготовка) или групповой (например, поддон с несколькими заготовками, которые двигаются по конвейеру). Групповой материальный поток моделируются с помощью ансамблей транзактов (транзактов, которые принадлежат одному семейству).

Операции сборки и демонтажа воссоздают с помощью блоков **MATCH, GATHER, ASSEMBLE.** Для управления транзактами используются блоки **GATE M** и **GATE NM.**

Технологические маршруты в GPSS задают с помощью функций [18] или матриц [10] (блок **MSAVEVALUE).**

Моделирование производственных процессов предусматривает создание устойчивой технологической схемы, поскольку последовательность выпускаемой продукции повторяется. Важной процедурной концепцией анализа эффективности является определение периода неустойчивой работы и устранение статистических данных, собранных за этот период по результатам моделирования.

6.2. Распределительные процессы

Распределительные процессы включают транспортирование и доставку, которые обеспечивают перемещение продукции или людей между разными пунктами в сети распределения. Характерным отличием транспортирования от доставки является то, что поточные объекты при транспортировании – это люди, а не товары.

Типичные процессы транспортирования можно найти в системах общественного транспорта. На типичных процессах доставки построены системы сбыта изготовленной продукции, доставки почты и товаров покупателю.

При моделировании распределительных процессов необходимо наблюдение таких характеристик, как место назначения, скорость доставки или затраты, а также свойства поточных объектов. При моделировании перемещения иногда целесообразнее представлять ресурсы для транспортирования как поточные объекты. Транспортные средства могут характеризоваться количеством мест, скоростью передвижения, маршрутом движения. Для синхронизации поточных объектов и транспортных средств целесообразно использовать списки пользователей (блоки **LINK** и **UNLINK**).

Большинство распределительных процессов носят переходный характер. Поэтому продолжительность моделирования должна быть достаточной, чтобы охватить весь цикл процесса. Кроме того, для анализа показателей эффективности прогон модели необходимо выполнить несколько раз.

6.3. Процессы обслуживания клиентов

Процессы обслуживания клиентов представляют собой одну из важнейших областей применения имитационного моделирования, поскольку в типичном процессе обслуживания суммарное время ожидания может достигать 95% от общего времени обработки.

Процессами обслуживания клиентов могут быть: предоставление услуг по телефону (справочные центры), работа «фабрик» услуг (рестораны, центры копирования, фотолаборатории), «магазинов» услуг (госпитали, ремонтные мастерские) и универмагов.

Имитационное моделирование процессов обслуживания клиентов считается сложной задачей, поскольку люди – это поточные объекты, но также могут быть ресурсами. Людям присуще сложное и непредвиденное поведение. Проще моделировать создание продуктов, документов, работу оборудования или перемещение транспортных средств. Например, клиенты, которые стоят в очереди, могут спорить, хитрить или вообще уйти. Моделирование подобных ситуаций требует незаурядной гибкости программирования.

В большинстве случаев время обслуживания клиентов и время появления клиентов являются случайными величинами. Поэтому для корректного представления необходимо использовать вероятностные распределения.

Поскольку поступление в систему носит циклический и случайный характер, системы обслуживания редко находятся в устойчивом состоянии. Поэтому было бы правильным моделировать осуществление операций в такой системе за некоторый период времени и соответствующим образом описывать работу элементов модели.

6.4. Процессы управления разработками проектов

Подобные процессы осуществляются одним человеком или группой людей. Типичными примерами являются разработка нового продукта или такие административные мероприятия, как подбор и расстановка кадров. Обычно в анализе подобных процессов задействован программный инструментарий управления проектами. Тем не менее, оценки времени полного цикла процесса и требований к ресурсам, которые получают в результате анализа, выполненного по методике имитационного моделирования, являются более точными, поскольку временные параметры выполнения

проекта очень нестабильны, **А** общее использование ресурсов приводит к появлению большого количества взаимосвязей.

Создавая корректную имитационную модель проекта, в первую очередь рассматривают такие элементы, как приоритеты выполнения срочных работ, разделение на смены, простои, ненормированные работы, графики обучения и т.п.

Важным моментом, на который следует обратить внимание при имитационном моделировании работ над проектом, является процедура повторения прогонов модели и получение результатов моделирования. Поскольку временные параметры неустойчивы, один прогон модели даст только один вариант развития бизнес-процессов. На основании множества повторных прогонов модели можно получить несколько вариантов сценария, которые дадут возможность получить более точные оценки и выделить наиболее уместные интервалы для определения показателей эффективности.

ГЛАВА 7. ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Задание 1. Моделирование разливной линии

Имеется некоторая конвейерная автоматизированная линия по выпуску баночек фруктового сока (рис.7.1). Пустые баночки для фруктового сока поступают в накопитель 1 автоматизированной линии каждые **А** ± *B* секунд. После этого в них автоматически заливается сок. Одновременно может заливаться лишь одна баночка, на что расходуется *E* секунд. Потом баночки поступают в накопитель 2 для выполнения операции закупоривания. Для этого расходуется *C* секунд времени на каждую баночку. Одновременно может обрабатываться одна баночка. Потом они попадают в накопитель 3 для следующей операции. В конце конвейера баночки устанавливаются в ящики. Время установки одной баночки представляет собой равномерно распределенную случайную величину в интервале *D* ± E секунд. Одновременно может устанавливаться в ящик не больше двух баночек.

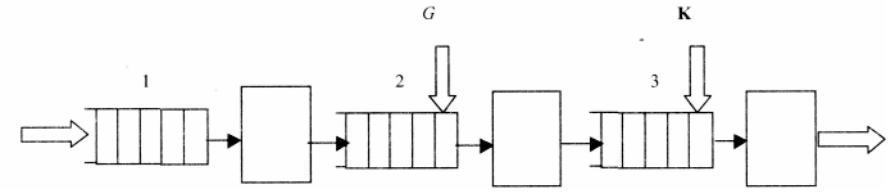


Рис. 7.1

Начальные условия: в начале смены в накопителе 2 находится *G* баночек, **А** в накопителе 3 – *К* баночек.

Определить, какие размеры должны иметь накопители с номерами, указанными в табл. 7.1 соответственно варианту. Промоделируйте работу линии на протяжении одной смены *(N* часов). В табл. 7.1 указаны варианты и значения параметров.

Таблица 7.1

Параметр		***Вариант***	
	1	*2*	*3*
A± *B*	3,5 ± 1,1	_4,5 ± 2,0	3,2 ± 1,3
F	1,5	1.2	2,3
C	1,6	1,3	2,4
D±*E*	2,0 ± 0,8	1,7 ± 0,5	2,6 ± 0,4
G	20	26	35
K	36	36	30
N __	8	6	7
Номера анализируемых накопителей	1,3	2,3	1,2

Задание 2 [10]. Моделирование контроля и настройки телевизоров

Собранные телевизоры проходят серию испытаний на станции технического контроля. Если оказывается, что функционирование телевизора ненормально, то отбракованный телевизор передают в цех наладки, где заменяют неисправные блоки. После наладки телевизор возвращают на станцию контроля и снова проверяют. Со станции технического контроля телевизоры после одной или нескольких проверок поступают в цех упаковки (рис. 7.2).

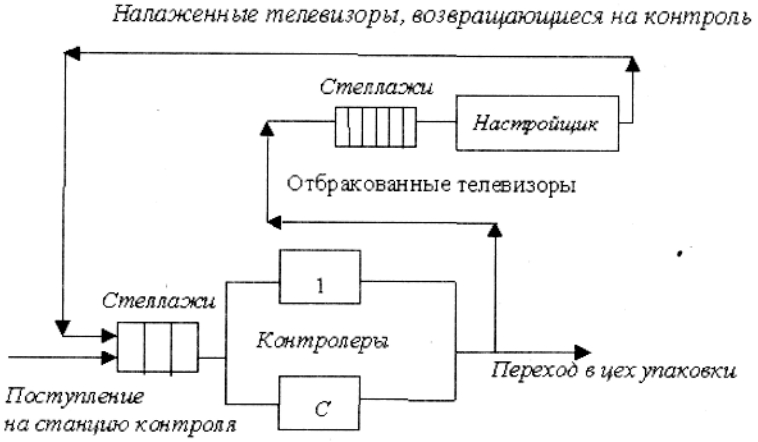


Рис. 7.2

Телевизоры попадают на станцию технического контроля каждые **A** ± *B* минут. На станции работают *C* контролеров одинаковой квалификации. Операция контроля одного телевизора состоит из двух проверок:

1) для первой проверки каждому контролеру необходимо *D* ± **E** минут;

2) для второй проверки на всех *C* контролеров имеется один тестовый прибор (продолжительность тестирования – *F* минут).

Приблизительно *G* процентов телевизоров успешно проходят проверку и попадают в цех упаковки, **A** другие *K* процентов – в цех наладки, в котором находится один рабочий – наладчик. Время наладки (замены) неисправных блоков распределено в соответствии с равномерным законом в интервале *N*±M минут.

Написать на GPSS модель функционирования этого параграфа производственной линии. Время моделирования – 8 ч.

Определить, сколько мест на стеллажах необходимо предусмотреть на входе станции контроля и в цехе наладки. В табл. 7.2 приведены варианты заданий и значения параметров.

Таблица 7.2

Параметр		***Варианты***	
	1	***2***	***3***
A ± B	5,5 ± 2,0	7,0 ± 1,5	6,5 ± 2,0
C	2	3	2
D±E	9± 3	6± 5	12 ± 3
F	1,2	1,5	1,0
G	85	95	85
K	15	5	15
N±M	30 ±7	35 ±9	30 ±8

Задание 3. Моделирование работы кафе

В небольшом кафе работают две официантки *(A* и *B)* обслуживая по *N* четырехместных столиков. Официантка **A** пользуется большей популярностью, чем официантка *B*. Приходя в кафе, клиент садится за столик официантки *B* только в том случае, если все места за столиками, которые обслуживает

официантка *A,* заняты. Клиенты приходят в кафе через $a \pm b$ минут и, если не застают свободных мест, становятся в очередь.

Когда клиент садится на освободившееся место, он ждет, пока к нему подойдет официантка и примет у него заказ. Время приема заказа у официантки **А** занимает $c \pm d$ секунд, у официантки *B* соответственно Е $\pm f$ секунд. Приняв заказ у клиента, официантки сразу же его выполняют. Время выполнения заказа обеими официантками составляет $g \pm h$ секунд. После получения заказа клиент на протяжении $k \pm m$ минут обедает и уходит из кафе. Официантки обслуживают клиентов по принципу FIFO и в каждый момент времени могут обслуживать не более одного клиента.

Определить время ожидания в очереди и время, которое клиент проводит за столиком кафе. Промоделируйте работу кафе на протяжении 10 ч. В табл. 7.3 приведены варианты заданий и значения параметров.

Таблица 7.3

Параметр	***Варианты***		
	1	***2***	***3***
N	*S*	7	6
$a \pm b$	*2*± 1	2± 1	3±2
$c \pm d$	45 ± 15	35 ±6	40 ± 10
$e \pm f$	17 ±4	22 ±6	35 ±8
$g \pm h$	160 ± 20	180 ±30	200 ± 50
$k \pm m$	16 ±4	10 ± 3	12 ± 3

Задание 4. Моделирование работы обрабатывающего цеха

В обрабатывающий цех через $a \pm b$ минут поступают детали двух типов: с вероятностью p_1 – первого типа, с вероятностью p_2 – второго типа.

Детали первого типа обрабатываются станком **А** (время обработки $c \pm d$ минуты, в каждый момент времени может обрабатываться только одна деталь). С вероятностью p_3 деталь не отвечает требованиям качества и возвращается на повторную обработку на станок, в противном случае она поступает на станок *C*.

Детали второго типа обрабатываются станком *B* (время обработки Е $\pm f$ минут, в каждый момент времени может обрабатываться только одна деталь). С вероятностью p_3 деталь не отвечает требованиям качества и возвращается на повторную обработку на станок *B,* в противном случае она поступает на станок С. Станок *C* может обрабатывать до *g* деталей одновременно, время обслуживания одной детали составляет $k \pm m$ минут.

Промоделировать работу цеха на протяжении *N* часов.

Определить время нахождения детали на обработке в цехе. В табл. 7.4 приведены варианты заданий и значения параметров.

Таблица 7.4

Параметр	***Варианты***		
	1	***2***	***3***
$a \pm b$	5±1	6±2	*7+2*
P_1	0,4	0,5	0,7
P_2	0,6	0,5	0,3
$c \pm d$	15±5	16±6	14±10
P_3	0,1	0,05	0,075
$e+f$	8±4	12±6	16±8
g	5	4	3
$k+m$	6±2	8±3	9±3
N	10	11	8

Задание 5. Моделирование работы обрабатывающего цеха

На вход некоторого цеха, который состоит из трех участков, поступает случайный поток деталей. Интервалы поступления имеют экспоненциальное распределение со среднимзначением 4 мин. С вероятностью 0,65 поступает деталь первого типа, с вероятностью 0,35 – второго типа. После того, как детали поступили в цех, они направляются на участок $У_1$, где обрабатываются последовательно одна за другой (время обработки распределено равномерно в интервале 2-5 мин). При этом детали второго типа имеют больший приоритет, чем детали первого типа.

Далее, после обработки на участке $У_1$ детали первого типа поступают на участок $У_2$, ***А*** детали второго типа – на участок $У_3$. На участке $У_2$ есть три идентичных станка. Время обработки детали станком имеет экспоненциальное распределение со средним значением 11 мин. На участке $У_3$ есть два станка (время обработки на каждом из них имеет экспоненциальное распределение со средним значением 7 мин).

Промоделировать работу цеха на протяжении 40 ч.

Определить статистические характеристики очереди деталей перед участками $У_2$ и $У_3$.

Задание 6. Моделирование работы обрабатывающего цеха

В цех поступает пуассоновский поток деталей с интенсивностью 20 дет./ч. С вероятностью 0,4 деталь поступает на первый участок, а с вероятностью 0,6 – на второй участок. На первом участке детали обрабатываются на одном из двух станков. Время обслуживания имеет экспоненциальное распределение со средним значением 48 мин. На втором участке детали обрабатывают одним станком за время, которое равномерно распределено в интервале 2 ± 1 мин. После обработки на одном из двух участков детали направляются к третьему участку с одним станком, на котором время обработки имеет экспоненциальное распределение со средним значением 2 мин.

Промоделировать обработку 1000 деталей.

Определить количество деталей, которые прошли через первый участок, и максимальную длину очереди перед третьим участком.

Построить GPSS-модель цеха, которая состоит: 1) из одного сегмента* (с использованием параметров транзактов); 2) из двух сегментов.

* Сегмент – часть GPSS-модели, которая начинается блоком GENERATE и заканчивается блоком TERMINATE.

Задание 7. Моделирование работы СМО

7.1. На вход одноканальной обслуживающей системы с интенсивностью λ(1/ед. времени) поступает пуассоновский поток требований. С вероятностью p_1 требование имеет первый тип, с вероятностью p_2 – второй тип. Требования второго типа при выборе из очереди имеют больший приоритет, чем требования первого типа. Время обслуживания требования прибором имеет экспоненциальное распределение со средним значением t_1, ед. времени для требования первого типа, t_2 – для требования второго типа. Промоделировать обслуживание K требований.

Оценить длину очереди требований перед прибором. В табл. 7.5 приведены варианты заданий и значения параметров.

Построить GPSS-модель, которая состоит: 1) из одного сегмента; 2) из двух сегментов.

7.2. На вход одноканальной обслуживающей системы поступает поток требований, время поступления которых равномерно распределено в интервале от **А** до B единиц времени. С вероятностью p_1 требование имеет первый тип, с вероятностью p_2 – второй тип. Требования второго типа при выборе из очереди имеют больший приоритет, чем требования первого типа. Время обслуживания требования прибором имеет экспоненциальное распределение со средним значением t_1 единиц времени для требования первого типа, t_2 – для требования второго типа. Промоделировать обслуживание K требований.

Оценить длину очереди требований перед прибором.

Таблица 7.5

Вариант	**λ**	***A***	***B***	p_1	p_2	t_1	t_2	***K***
1	0,05,	20	40	0,4	0,6	12	16	100
2	0,03	30	70	0,2	0,8	28	26	200
3	0,005	200	300	0,3	0,7	100	190	300
4	0,006	180	260	0,65	0,35	70	200	400

Задание 8. Моделирование функций

8.1. Задать дискретную GPSS-функцию, приведенную в табл. 7.6.
Построить график функции. Использовать эту функцию в блоке **GENERATE.**

Таблица 7.6

Значение функции	2	3	4	8	10
Вероятность	0,3	0,1	0,2	0,1	0,3

8.2. Задать кусочно-непрерывную GPSS-функцию, которая моделирует случайную величину, заданную в табл. 7.7. Внутри каждого интервала случайная величина равновероятно приобретает одно из целых значений этого интервала. Аргументом функции является случайное число, равномерно распределенное в интервале 0-1. Построить график функции. Использовать эту функцию в блоке **ADVANCE.**

Таблица 7.7

Номер интервала	**1**	***2***	***3***
Вероятность того, что случайная величина примет значения из интервала	0,4	0,4	0,2
Значения интервала	2-4	5-12	13-40

8.3. Средствами GPSS задать пуассоновский поток требований:
1) с параметром 0,25 мин$^{-1}$;
2) со средним значением времени поступления 5 с.
8.4. Задать дискретную GPSS-функцию, приведенную в табл. 7.8. Построить график функции. Использовать эту функцию в блоке **GENERATE.**

Таблица 7.8

Значение функции	3	2	1	4	8	5
Вероятность	0,05	0,05	0,1	0,2	0,3	0,3

8.5. Задать кусочно-непрерывную GPSS-функцию, которая моделирует случайную величину, заданную в табл. 7.9. Внутри каждого интервала случайная величина равновероятно приобретает одно из целых значений этого интервала. Аргументом функции является случайное число, равномерно распределенное в интервале 0—1. Построить график функции. Использовать эту функцию в блоке **ADVANCE.**

Таблица 7.9

Номер интервала	1	2	*3*	*4*
Вероятность того, что случайная величина примет значения из интервала	0,5	0,2	0,2	0,1
Отрезок	3-8	9 – 13	14 – 40	41 – 50

8.6. Средствами GPSS задать пуассоновский поток требований:
1) с параметром 0,33 $ч^{-1}$;
2) со средним значением времени поступления 25 мин.

8.7. Задать дискретную GPSS-функцию, приведенную в табл. 7.10. Построить график функции. Использовать эту функцию в блоке **GENERATE.**

Таблица 7.10

Значение	6	8	12	14	20	25	30
Вероятность	0,1	0,05	0,05	0,2	0,2	0,3	0,1

8.8. Задать кусочно-непрерывную GPSS-функцию, которая моделирует случайную величину, заданную в табл. 7.11. Внутри каждого интервала случайная величина равновероятно приобретает одно из целых значений этого интервала. Аргументом функции является случайное число, равномерно распределенное в интервале 0—1. Построить график функции. Использовать эту функцию в блоке **ADVANCE**.

Таблица 7.11

Номер интервала	*1*	*2*	*3*	*4*
Вероятность того, что случайная величина примет значения из интервала	0,2	0,3	0,15	0,35
Отрезок	100-150	151-200	201-225	226-250

8.9. Средствами GPSS задать пуассоновский поток требований:
1) с параметром 0,2 ед./год;
2) со средним значением времени поступления четыре года.

8.10. Задать дискретную GPSS-функцию, приведенную в табл. 7.12. Построить график функции. Использовать эту функцию в блоке **GENERATE.**

Таблица 7.12

Значение функции	2	3	6	5	8	10
Вероятность	0,1	0,2	0,2	0,3	0,1	0,1

8.11. Задать кусочно-непрерывную GPSS-функцию, которая моделирует случайную величину, заданную в табл. 7.13. Внутри каждого интервала случайная величина равновероятно приобретает одно из целых значений этого интервала. Аргументом функции является случайное число, равномерно распределенное в интервале 0-1. Построить график функции. Использовать эту функцию в блоке **ADVANCE.**

Таблица 7.13

Номер интервала	*1*	*2*	*3*	*4*
Вероятность того, что случайная величина примет значения из интервала	0,38	0,12	0,25	0,25
Отрезок	1 – 5	6 – 10	11 – 15	16 – 20

8.12. Средствами GPSS задать пуассоновский потоктребований:
1) с параметром 0,5 ед./сутки;
2) со средним значением времени поступления четверо суток.

Задание 9 [10]. Моделирование системы обслуживания

На вход одноканальной системы обслуживания поступает два потока требований**.** Первый из них – пуассоновский с интенсивностью λ 1/мин. Во втором потоке интервалы поступления распределены равномерно на отрезке *a-b* мин. Интенсивность обслуживания требования устройством зависит от длины очереди на обслуживание. Если длина очереди меньше или равняется *d* (больше, чем *d),* то время обслуживания имеет экспоненциальное распределение со средним значением t_1 (t_2) минут. Промоделировать работу системы на протяжении *К* часов. В табл. 7.14 приведены варианты заданий и значения параметров.

Таблица 7.14

Вариант	**λ**	***a***	***b***	***d***	t_1	t_2	***К***
1	0,2	16	40	3	2	4	100
2	0,25	6	20	4	1	2.5	250
3	0,1	3	9	6	2	4	350
4	0,15	2	7	5	1	4	400

Самостоятельно задать соответствующую функциональную зависимость двумя способами: через дискретную и непрерывную GPSS-функции с числом отрезков не меньше трех, если интенсивность обслуживания требования устройством зависит от: 1) времени функционирования системы;

2) числа нулевых входов в очередь;

3) числа ненулевых входов в очередь;

4) средней длины очереди;

5) текущей длины очереди;

6) среднего времени пребывания в очереди;

7) коэффициента использования устройства;

8) числа входов в устройство (сколько раз использовалось устройство).

Задание 10 [16]. Моделирование системы автоматизации проектирования

Система автоматизации проектирования состоит из ЭВМ и трех подключенных к ней терминалов. За каждым терминалом работает один проектировщик, который формирует задания на расчет в интерактивном режиме. Набор строки задания занимает 10 ± 5 с. Анализ строки требует 3 с работы ЭВМ и 5 с работы терминала. В каждый момент времени может анализироваться только одна строка. После набора десяти строк считается, что задание сформировано и поступает на решение, которое занимает 10 ± 3 с работы ЭВМ (решение заданий имеет больший приоритет, чем анализ строк). Вывод результата решения требует 8 с работы терминала, **А** анализ результата проектировщиком – 30 ± 10 с, после чего цикл повторяется.

Промоделировать работу системы на протяжении 6 ч.

Определить вероятность простоя проектировщика из-за занятости ЭВМ, коэффициент загрузки ЭВМ и параметры очереди к ЭВМ.

Задание 11 [16]. Моделирование работы транспортного цеха

Транспортный цех обслуживает три филиала *A, B* и *C.* Грузовики перевозят изделия из **А** в *B* и из *B* в С, возвращаясь потом в **А** без груза. Погрузка изделий в филиале **А** занимает 20 мин, переезд из **А** в *B* длится 30 мин, разгрузка и загрузка в филиале *B* – по 20 мин, переезд в *C* – 30 мин, разгрузка в С – 20 мин и переезд в **А** – 20 мин. Если на момент загрузки в филиалах **А** и *B* изделия отсутствуют, грузовики уходят дальше по маршруту пустыми. Изделия в **А** выпускаются партиями по 1000 шт. через 20 ± 3 мин, в *B* – такими же партиями через 20 ± 5 мин. На линии эксплуатируется восемь грузовиков, каждый может перевозить по 1000 изделий. В начальный момент четыре грузовика находятся в *A,* четыре – в *B.*

Промоделировать работу транспортного цеха на протяжении 1000ч.

Определить частоту пустых перегонов грузовиков между филиалами **А** и *B, B и C.*

Задание 12 [16]. Моделирование системы передачи разговора

В системе передачи цифровой информации разговор передается в цифровом виде. Речевые пакеты поступают через 6 ± 3 мс и передаются через два *последовательно* соединенных канала. В каждый момент времени каждый из каналов может передавать только один пакет. В случае занятости канала пакеты сохраняются в накопителях перед каждым каналом. Время передачи пакета по каждому из каналов имеет экспоненциальное распределение со средним значением 5 мс. Пакеты, время передачи которых больше 10 мс (без учета времени ожидания), на выходе системы уничтожаются, поскольку длительное время передачи значительно снижает качество передаваемой речи. Уничтожение свыше 30% пакетов недопустимо. При достижении такого уровня система за счет ресурсов ускоряет передачу в каналах до среднего значения времени 4 мс. При снижении уровня до приемлемого значения происходит отключение ресурсов.

Промоделировать 10 с работы системы.

Определить частоту уничтожения пакетов, частоту подключения ресурсов и среднее время нахождения одного пакета в системе передачи информации (с учетом времен ожидания).

Задание 13 [16]. Моделирование системы передачи данных

Система передачи данных обеспечивает передачу пакетов данных из пункта **А** в пункт *С* через транзитный пункт *В.* В пункт **А** пакеты поступают через 10± 5 мс. Здесь они сохраняются в накопителе с максимальной вместительностью 25 пакетов и с равной вероятностью передаются по одной из двух линий: *АВ1* – за 20 мс; *АВ2* – за 20 ± 5 мс. В пункте *В* пакеты снова буферизируются в накопителе с максимальной вместительностью 20 пакетов и дальше передаются по линии *ВС1* за 20 ± 3 мс и по линии *ВС2* за 25 мс. Причем пакеты, которые передавались по *АВ1,* поступают в *ВС1,* а те, которые передавались по *АВ2, –* в *ВС2.* При достижении предельного значения количества пакетов в накопителе (максимальной вместительности) пакет, который пытается попасть в этот накопитель, уничтожается.

Промоделировать работу системы на протяжении 1 мин.

Оценить вероятность уничтожения пакетов.

Задание 14 [16]. Моделирование узла коммутации сообщений

В узел коммутации сообщений, который состоит из входного буфера, процессора, двух выходных буферов и двух выходных линий, поступают сообщения с двух направлений (по каждому через интервалы времени 15±7мс). Сообщения с первого направления поступают во входной буфер, обрабатываются в процессоре, накапливаются в выходном буфере первой линии и передаются по первой выходной линии. Сообщения со второго направления обрабатываются аналогично, но накапливаются в выходном буфере второй линии и передаются по второй линии. Примененный метод контроля потоков разрешает одновременное присутствие в системе не больше трех сообщений с каждого направления. Если при наличии в системе трех сообщений с некоторого направления поступает сообщение с этого же направления, то оно получает отказ. Время обработки в процессоре равняется 7 мс на сообщение, время передачи по каждой из выходных линий – 15 ± 5 мс.

Промоделировать работу узла коммутации на протяжении 10 с.

Определить загрузку устройств и вероятность отказов в обслуживании.

Задание 15 [16]. Моделирование процесса сборки

На сборочный участок цеха предприятия из трех независимых источников через интервалы времени, которые имеют экспоненциальное распределение со средним значением 10 мин, поступают детали. Каждая деталь с вероятностью 0,5 должна пройти обработку на протяжении 7 мин. На сборку подаются одна обработанная и одна необработанная детали. В результате получают готовое изделие. Процесс сборки занимает 6 мин. В каждый момент времени может собираться только одно изделие. Потом изделие поступает на регулирование, которое продолжается в среднем 8 мин (экспоненциальное распределение).

Промоделироватьработу цеха на протяжении 24 ч.

Оценить загруженность операций и распределение времени пребывания в системе.

Задание 16 [16]. Моделирование работы цеха

Детали, необходимые для работы цеха, находятся на цеховом и центральном складах. На цеховом складе может храниться до 20 комплектов деталей, потребность в которых возникает через 60 ± 10 мин и составляет один комплект. В случае уменьшения запасов до трех комплектов на протяжении 60 мин формируется требование на пополнение запасов цехового склада до полного объема (20 комплектов), которая посылается на центральный склад, где на протяжении 60 ± 20 мин происходит комплектование и за 60 ± 5 мин осуществляется доставка деталей в цех.

Промоделировать работу цеха на протяжении 400 ч.

Оценить вероятность простоя цеха из-за отсутствия деталей.

Задание 17 [16]. Моделирование системы управления производством

Пусть имеется некоторая система управления производством, в которой ЭВМ циклически опрашивает три датчика информации (рис. 7.3). Информация в датчиках появляется через 12±3 с и имеет размер 3000±1000 символов; ЭВМ поочередно каждому датчику предоставляет 3 с:

- в первые 3 с обрабатывается информация из первого датчика;
- во вторые 3 с обрабатывается информация из второго датчика;
- в третьи 3 с обрабатывается информация из третьего датчика;
- в четвертые 3 с обрабатывается информация из первого датчика и т.д.

Если на момент начала опрашивания у датчика нет информации для обработки, имеем свободный цикл.

Если за соответствующие 3 с ЭВМ успевает обработать информацию датчика, то обслуживание завершается, если – нет, то остаток необработанной информации становится в специальную очередь. Задания, которые находятся в этой очереди, обрабатываются во время свободных циклов.

Скорость обработки информации ЭВМ равна 1000 символов в секунду.

Промоделировать 5 ч работы ЭВМ.

Определить загрузку ЭВМ, параметры специальной очереди неоконченных заданий.

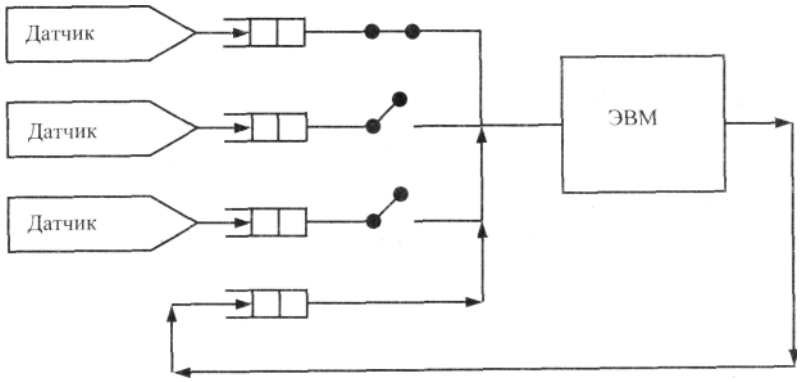


Рис. 7.3

Задание 18. Моделирование производственного процесса

Имеется некоторый производственный процесс, который реализуется линией с тремя последовательно установленными агрегатами: Л, *Б* и *В*. Поток продукции, который поступает от агрегата А, является пуассоновским со средней нормой выработки 10 изделий за час. Агрегат *Б* функционирует по равномерному закону, продолжительность обработки изделия составляет 4 ± 6 мин. Закон распределения времени обслуживания изделий агрегатом *В* приведен в табл. 7.15.

Таблица 7.15

Вероятность	0,1	0,2	0,4	0,2	0,1
Продолжительность обслуживания, мин	2	3	4	5	6

При скоплении на входе агрегата *B* двух или более изделий в технологической линии возникает затор.

Промоделировать функционирование линии на протяжении 100 ч.

Определить общее время затора на входе агрегата *B*. Построить гистограмму распределения продолжительности заторов.

Задание 19. Моделирование работы заправочной станции

На заправке есть три вида топлива для автомобилей: низкооктановый, высокооктановый бензины и дизельное топливо. Для каждого вида топлива есть свои колонки. Характеристики заправки приведены в табл. 7.16. Прибытие автомобилей на заправку распределено согласно закону Эрланга второго порядка со средним значением 2,2 мин. В 10% автомобилей после заправки доливают от 0,5 до 2 л масла. Доливание 0,5 л масла занимает 2 мин. Стоимость одного литра масла – 40 руб.

Оценить среднее время обслуживания автомобилей на заправке и выручку за пять дней работы.

Таблица 7.16

Вид топлива	***Количество колонок***	***Часть автомбилей, которые заправляются, %***	***Количество топлива, которым заправляют автомобиль, л***	***Скорость заправки, л/мин***	***Стоимость топлива за литр, руб***
Низко-октановый бензин	1	30	Равномерно распределено в интервале 5-60 л (через 5 л)	12	17,0
Высоко-октановый бензин	*2*	50	Равномерно распределено в интервале 5-40 л (через 5 л)	15	21,5
Дизельное опливо	1	20	Равномерно распределено в интервале 10-60 л (через 5 л)	18	15,0

Задание 20. Моделированиеработы станции технического обслуживания

На станцию технического обслуживания (СТО) согласно закону Эрланга второго порядка со средним временем прибытия 14 мин прибывают автомобили для технического обслуживания (36% автомобилей) и ремонта (64% автомобилей). На СТО есть два бокса для технического обслуживания и три бокса для ремонта. Выполнение простого, средней сложности и сложного ремонтов – равновероятно.

Время и стоимость выполнения работ по техническому обслуживанию и ремонту зависит от категории выполняемых работ (табл. 7.17).

После технического обслуживания 12% автомобилей поступают для выполнения ремонта средней сложности.

Построить гистограмму времени обслуживания автомобилей.

Оценить выручку СТО за пять дней работы.

Таблица 7.17

Категория работ	***Время ремонта, мин***	***Стоимость ремонта, руб.***

Техническое обслуживание	Равномерно распределено в интервале 10-55	Равномерно распределено в интервале 100-400
Простой ремонт	Равномерно распределено в интервале 12-45	Равномерно распределено в интервале 50-450
Ремонтсредней сложности	Нормально распределено со средним 45 и среднеквадратичным отклонением 5	Равномерно распределено в интервале 100-1400
Сложный ремонт	Равномерно распределено в интервале 80-150	Равномерно распределено в интервале 350-2550

Задание 21. Моделирование работы станции скорой помощи

На станцию скорой помощи поступают вызовы по телефону. Станция имеет пять каналов для одновременного приема вызовов. Время между попытками вызова скорой помощи распределено согласно закону Эрланга второго порядка (среднее время – 1,5 мин). Абоненты тратят 15 с на набор номера и, если застают все каналы занятыми, через 20 с повторяют вызов. Так происходит до тех пор, пока вызов не будет принят. Время приема вызова составляет 1 мин.

На станции скорой помощи для обслуживания вызовов имеется 15 автомобилей. Время, затраченное на проезд к больному, зависит от расстояния до его дома. Распределение расстояния приведено в табл. 7.18. После предоставления помощи автомобили возвращаются на станцию. Скорость движения автомобилей равномерно распределена в интервале 35-55 км/ч.

Таблица 7.18

Вероятность	0,15	0,22	0,17	0,28	0,18
Расстояние, км	5	8	12	15	20

Время оказания помощи больному распределено в соответствии с нормальным законом со средним значением 25 мин и среднеквадратическим отклонением 4 мин.

Оценить среднее время от момента начального вызова скорой помощи до окончания помощи больному и средний пробег автомобиля за пять дней работы.

Задание 22. Моделирование работы госпиталя

В госпиталь на протяжении суток поступают раненые и потерпевшие от катастрофы, которых доставляют на пятиместных (70%) и трехместных (30%) автомобилях. Время прибытия автомобилей распределено согласно закону Эрланга второго порядка со средним значением 45 мин.

В госпитале бригада из трех терапевтов и одного хирурга на протяжении 4 ± 2 мин осматривают раненых и потерпевших, определяют необходимый вид предоставления медицинской помощи и направляют в соответствующую палату (табл. 7.19).

После операционной 55% больных направляют в палату реанимации, **А** 45% – в палату интенсивной терапии.

Промоделировать работу госпиталя на протяжении 10 суток.

Оценить среднее время пребывания пострадавших в госпитале и необходимое количество мест в палатах.

Таблица 7.19

Вероятность направления	***Палата***	***Количество мест***	***Время предоставления помощи, мин***
0,15	интенсивной терапии	20	Распределено равномерно в интервале 1440-2060

0,25	операционная	6	Распределено равномерно в интервале 20-120
0,35	реанимации	20	Распределено равномерно в интервале 2880-3660
0,15	хирургическая	25	Распределено нормально со средним значением времени 1800 мин и среднеквадратическим отклонением 60
0,1	терапии	30	Распределено равномерно в интервале 1200-2200

Задание 23. Моделирование работы маршрутных такси

На некотором городском маршруте по кольцевому маршруту с десятью остановками работают пять 11-местных и десять 14-местных микроавтобусов. Время движения между остановками имеет равномерное распределение в интервале 5±8 мин. На каждую остановку в соответствии с экспоненциальным законом распределения со средним значением 2 мин прибывают пассажиры и ждут микроавтобуса. Микроавтобус подъезжает к остановке и забирает столько пассажиров, сколькс имеется свободных мест. Если свободных мест больше, чем пассажиров, то микроавтобус забирает всех. Если на остановке никто не выходит и в микроавтобусе нет свободных мест, он не останавливается. Вероятность того, что пассажир проедет некоторое количество остановок, задана в табл. 7.20. Стоимость проезда – 30 руб.

Таблица 7.20

Вероятность	0,1	0,15	0,2	0,25	0.3
Количество остановок	3	4	5	6	7

Промоделировать работу микроавтобусов на протяжении 16 ч.

Оценить загруженность микроавтобусов, распределение времени поездки пассажиров и выручку со всех микроавтобусов.

Задание 24. Моделирование работы печатной системы

В компьютерной сети издательского дома используются два сетевых высокопроизводительных принтера: цветной и черно-белый, которые подключены к одному принт-серверу. От сотрудников на печать поступает пуассоновский поток документов с интенсивностью N документов/мин. Количество страниц в документах имеет нормальное распределение с математическим ожиданием m и среднеквадратичным отклонением σ (≠1) (объем страниц имеет экспоненциальное распределение со средним значением а Кб), причем с вероятностью p_1 эти документы предназначены для распечатки на черно-белом принтере и с вероятностью $(1 – p_1)$ – на цветном.

Сначала документы обрабатываются на принт-сервере и становятся в его очередь, размер которой равен P Мб. При превышении этого числа принт-сервер приостанавливает прием документов на обработку и возвращает отправителям сообщение об ошибке. Время печати одной страницы имеет экспоненциальное распределение со средним значением b минут для черно-белой печати и c минут – для цветной.

Промоделировать работу печатной системы издательского дома на протяжении R часов.

Оценить время, проходящее от посылки документа на печать до окончания печати.

Определить, на сколько надо изменить размеры очереди принт-сервера, чтобы сотрудники не получали соответствующих сообщений об ошибках.

Параметры задать самостоятельно.

Задание 25. Моделирование процесса сборки ПК

Радиозавод выполняет заказы мелких компьютерных фирм по сборке персональных компьютеров (ПК) под их торговыми марками. Сборка производится на конвейере.

На вход конвейера поступают полные наборы комплектующих с интенсивностью $a \pm b$ мин. На первом участке производится параллельная сборка n_1 ПК по $c \pm d$ мин каждый. Затем каждый собранный ПК проходит настройку и проверку на предмет работоспособности аппаратной части по **E** $\pm$ f мин каждый. Эту проверку не проходят p_1% ПК. Отбракованные ПК отправляют обратно на участок сборки для устранения неисправностей, которое занимает $g \pm h$ мин.

По желанию заказчиков на собираемые ПК может быть установлено программное обеспечение (операционная система и прочее). Поэтому только p_2% собранных ПК направляются на участок упаковки, **A** остальные – на участок установки и настройки программного обеспечения (ПО), на котором параллельно работают n_2 инженеров. Установка ПО на один компьютер занимает $k \pm l$ мин. В процессе этого на P_3% ПК могут обнаружиться незамеченные ранее аппаратные проблемы, вследствие чего эти ПК отправляются на первый участок для устранения неисправностей, которое занимает $g \pm h$ мин. Исправные компьютеры поступают на участок упаковки.

На участке упаковки все ПК предварительно складируются, **A** затем поступают на один из n упаковочных станков, упаковка на каждом из которых занимает m минут.

Промоделировать работу завода на протяжении K часов.

Определить среднее время выполнения заказа и максимальный размер склада для участка упаковки.

ГЛАВА8. ПРОЕКТИРОВАНИЕ ИМИТАЦИОННЫХ МОДЕЛЕЙ С ПОМОЩЬЮ ИНТЕРАКТИВНОЙ СИСТЕМЫ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ

8.1. Структура интерактивной системы имитационного моделирования

Интерактивная система имитационного моделирования (ИСИМ) [5] – это средство автоматизации процесса создания GPSS-моделей. Модели представляются в виде сетей массового обслуживиния с произвольным количеством узлов для проведения экспериментов с ними без написания программных реализаций. Структура программного генератора показана на рис. 8.1. Дадим описание объектов показанных на рис. 8.1:

ОРМ – объекты реального мира, которые моделируются.

СМПМ – система меню проектирования моделей, предназначенная для взаимодействия проектировщика с программным генератором. Она предполагает реализацию функций манипулирования объектами модели или их компонентами (добавление объектов, модификация, удаление);

СПНМ – система параметрической настройки модели, отображающая формальное многоуровневое представление стохастических сетевых моделей в виде концептуальной, логической и программной структурных схем. Для этого задаются узлы сети и связи между узлами как некоторые объекты и определяются свойства объектов и связей. После определения всей сети задаются условия эксперимента. Совокупность реализаций СПНМ для модели подготавливает всю необходимую информацию для создания имитационной модели в среде GPSS. Эта информация поступает на вход лингвистического процессора (ЛП). Система параметрической настройки моделей ориентирована на текстовый файл, который представляет собой описание элементов вершин стохастической сети.

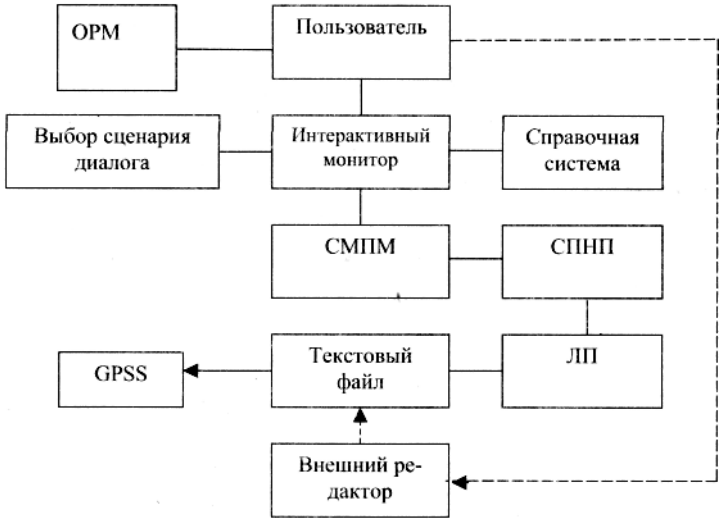


Рис. 8.1

Лингвистический процессор отображает совокупности подмоделей каждого узла и связей для конкретной GPSS модели. Он строит модель таким образом, чтобы из допустимого множества блоков были выбраны необходимые, **А** потом выстраивает их в логическую последовательность. Использование блоков и операций языка GPSS зависит от модели и методов параметрической настройки.

На концептуальном уровне модель задается графом, вершины которого представляют собой множество таких объектов, как генераторы требований, одно– или многоканальные устройства обслуживания и терминаторы, уничтожающие требования.

Логический уровень представления модели объединяет объект и выходящую из него связь. На этом уровне определяются свойства объектов и связей модели.

Программный уровень представления модели содержит готовый текст GPSS-программы модели, который создается после компиляции проекта.

Программный генератор полностью автоматизирует процесс создания имитационной модели и проведение экспериментов с ней, но если пользователь знает язык GPSS, то с помощью внешнего редактора он может изменить или дописать код программы. Новая версия ISS 2000 расширяет возможности ИСИМ, включает генерацию формул и применение операционного анализа для поиска узких мест в сети.

8.2. Построение концептуальной схемы модели

В терминах ИСИМ выполняемые с моделью работы называются проектом. Для создания нового проекта нужно выбрать пункт меню **Файл/Создать.** При этом открывается рабочее окно проекта, которое содержит:

- область проектирования модели;
- линейку шаблонов объектов;
- набор кнопок настройки модели.

На первом этапе построения модели необходимо задать ее концептуальную схему, то есть определить узлы и связи стохастическое сети. Другими словами, определяется модель в форме концептуальной структуры. Линейка шаблонов объектов имеет шесть шаблонов, на базе которых строятся узлы стохастической сети. Рассмотрим эти шаблоны.

Генератор – служит для создания потока транзактов.

Одноканальное, многоканальное устройство – база для построения узлов в виде СМО.

Терминатор – уничтожает транзакты.

Построитель формул – задает сложные логические и математические выражения для атрибутов модели.

Для добавления узла к модели необходимо нажать мышью на кнопку с изображением необходимого шаблона и потом перетянуть его на область проектирования (drag-and-drop). К модели добавляется необходимый узел, **А** его изображение появляется в области проектирования.

Таким образом, можно определить все вершины стохастической сети. Кроме того, используя механизм drag-and-drop, можно расположить узлы в области проектирования в наиболее эргономичном и удобном виде.

После определения всех вершин модели необходимо задать связи между узлами. Для этого в правом верхнем углу над областью проектирования есть кнопка **Связать узлы.** При нажатии на нее пользователю предлагается выбрать узел-источник, **А** после – узел-приемник транзактов. На связи определения маршрутов движения транзактов наложены определенные ограничения. Так, источником транзактов не может быть выбранный узел ***Терминатор*** хотя бы потому, что в нем транзакты уничтожаются, **А** приемником не может быть узел ***Генератор.***

Созданные узлы и связи можно удалять из модели. Для этого нужно выделить необходимый объект и нажать правую кнопку мыши – появится всплывающее меню, из которого нужно выбрать пункт **Удалить.**

После определения узлов сети и связей между ними концептуальная структура является полностью заданной и концептуальный этап проектирования модели завершается.

8.3. Параметрическая настройка модели

На логическом уровне выполняется параметрическая настройка элементов стохастической сети, которая состоит в определении свойств узлов, связей, **А** также правил движения транзактов между узлами. Для задания свойств узла нужно выделить необходимый объект и нажать правую кнопку мыши, после чего появится всплывающее меню, из которого надо выбрать пункт **Свойства.** Содержание меню **Свойства** зависит от типа узла.

Меню генератора содержит следующие пункты:

Закон поступления – выбор закона распределения для времени поступления транзактов в модель;

Параметры – определение параметров закона распределения;

Задержка первого сообщения – задерживать ли первый транзакт, если да, то на сколько;

Задание приоритета – задавать ли приоритет обслуживания транзактам, если да, то какой;

Ограничение числа транзактов – ограничено ли количество транзактов, которые генерируются, если да, то каким образом;

Параметры транзактов – определение параметров транзактов, с возможностью добавления, переименования, редактирования и удаления.

Пункты меню одноканального устройства:

Имя устройства – ввод имени устройства;

Закон обслуживания – выбор закона распределения для времени обслуживания транзактов;

Параметры – определение параметров закона распределения;

Приоритет обслуживания – выбор дисциплины обслуживания транзактов;

Изменение значения приоритета изменение приоритета транзактов после обслуживания.

Пункты меню для очереди:

Имя очереди – ввод имени очереди;

Правило выбора из очереди – выбор порядка обслуживания транзактов (FIFO или LIFO);

Ограничения – задание ограничений, которые накладываются на очередь;

Статистика – получение статистики по очереди и по времени пребывания в очереди.

Пункты меню МКУ:

Имя устройства – ввод имени устройства;

Закон обслуживания – выбор закона распределения для времени обслуживания транзактов;

Количество каналов – количество каналов в устройстве;

Количество занимаемых каналов – количество каналов, который занимает один транзакт;

Параметры – определение параметров закона распределения;

Приоритет обслуживания – выбор дисциплины обслуживания транзактов;

Изменение значения приоритета изменение приоритета транзактов после обслуживания.

Пункты меню для связи:

Кратность обслуживания – задание кратности обслуживания (однократная или многократная);

Прекращение обслуживания – задание условия прекращения обслуживания;

Обеспечение вложенности устройств – определение вложенности устройств одно в другое (не доступно для объектов ***Генератор*** **и** ***Терминатор);***

Задержка транзактов – определение закона и задание параметров закона для времени задержки при передаче транзактов.

В узле типа ***Терминатор*** транзакты уничтожаются, поэтому свойства ему не задаются.

Функция распределения времени генерации или обслуживания транзакта в узле иногда бывает неизвестной. Если известен характер распределения, то по виду функции можно приблизительно аппроксимировать ее вероятностными законами распределений. Для учета такой ситуации во всех сценариях, где нужно выбрать закон распределения, есть кнопка **Тест функции.** При нажатии на нее пользователю представляются графики выбранной функции распределения и плотности вероятности. Это позволяет варьировать законами распределения и их параметрами для получения желаемого характера распределения.

В ИСИМ предусмотрены следующие законы распределений:

- детерминированный;
- экспоненциальный;
- равномерный;
- нормальный;
- логнормальный;
- бета-распределение;
- гамма-распределение;
- распределение Вейбула.

Пользователь может самостоятельно задавать функции, используя пункт меню **Проект/Определить функцию.** В окне определения функции можно создать новую или загрузить сохраненную раньше функцию распределения. Выбирается тип функции, имя, аргумент и количество точек, после чего задаются ее значения. Если все точки функции определены, можно посмотреть графики. Это осуществляется нажатием кнопки **Нарисовать.** После этого функция добавляется в проект, и ее можно выбирать во всех диалоговых окнах, где нужно выбирать закон распределения. Для дальнейшего использования созданную функцию можно сохранить в файле с именем <имя_функции>.ufn.

8.4. Генератор формул

При создании модели той или другой системы часто возникает потребность вычисления некоторых величин в процессе прогона модели, например, экономических показателей (размера прибыли или убытков) или различных временных величин (время занятости или простоя для определенных участков модели) и т.д. Для этого используется генератор формул. При построении модели системы пользователю предоставляется возможность задавать сложные логические и математические выражения с использованием СЧА. Для этого используется объект **Построитель формул.**

8.5. Управление экспериментом

Для задания условий проведения эксперимента и его прекращения нужно выбрать пункт меню **Проект/Условия эксперимента.**

Есть возможность прекратить эксперимент по времени моделирования или при прохождении через модель определенного числа транзактов. К операциям управления экспериментом относится и сбор статистики по результатам моделирования. Частично эта проблема решается на логическом этапе проектирования. Действительно, задание статистических условий для одноканальных устройств и МКУ, **А** также очередей транзактов к ним осуществляется при определении свойств узлов. Тем не менее, можно отслеживать время прохождения транзактами определенных участков модели. Для этого вводится понятие ***коллекции,*** которая собирает в таблицы времена продвижения транзактов по участкам сети с последующим построением гистограммы. Для создания коллекции надо задать определенный маршрут движения транзактов по модели, а также свойства таблицы для времени прохождение этого маршрута. Создание коллекции во многом похоже на определение связи между узлами сети. Сначала нужно нажать кнопку **Указать выдачу статистики о времени прохождения,** после этого отметить начальный узел коллекции, а затем конечный. Система построения коллекций проверяет допустимость маршрута. Если удается построить маршрут, то создается коллекция и ниже области проектирования появляется окно списка существующих коллекций. Коллекции можно удалять

и редактировать. В обоих случаях нужно нажать правую кнопку мыши над необходимой коллекцией и в выпадающем меню выбрать нужную операцию. В свойствах коллекции задаются:

- имя таблицы, которая представляет собой идентификатор данной коллекции;
- верхняя граница первого частотного интервала;
- ширина интервала;
- количество интервалов.

На рис. 8.2 показана графическая интерпретация оси действительных значений и ее деление на ряд интервалов для построения таблицы *(A* – верхняя граница первого частотного интервала; *B* – ширина интервала; *N* – количество интервалов).

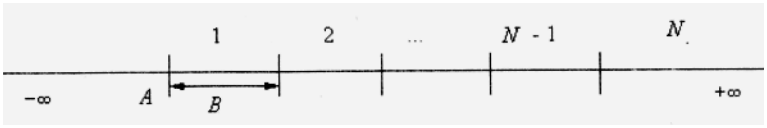


Рис. 8.2

Таким образом, коллекция создает гистограмму распределения случайной величины.

8.6. Запуск эксперимента и обработка результатов моделирования

Процесс создания модели предусматривает несколько этапов: концептуальный, логический, определение условий моделирования, сбор статистики и построение исходного программного файла GPSS-программы. Для пользователя это означает компиляцию данных проекта с помощью пункта меню **Проект/Построить.** Система ИСИМ проверяет корректность заданной модели и выполняет генерацию файла в текстовом формате GPSS. После компиляции пользователь имеет возможность посмотреть информацию о построенной модели и выполнить моделирование.

Квалифицированный пользователь, который знаком с языком системы GPSS, может просмотреть исходный текст программы, выбрав меню **Проект/Текст программы.** Есть также возможность просматривать, запускать и обрабатывать результаты моделирования внешних моделей GPSS программ. Под внешними понимают отлаженные GPSS-модели, сформированные не системой.

При запуске эксперимента формируется bat-файл для выполнения программ GPSSPC.EXE и GPSSREPT.EXE [14] с построенной моделью. После выполнения программ формируется отчет об эксперименте в удобном для пользователя виде.

Окно результатов моделирования имеет две закладки: на первой из них представлены статистические данные об устройствах (одно – и многоканальные) и об очередях, на второй – информация о времени пребывания транзактов в очередях и статистических коллекциях.

8.7. Управление проектами и общей настройкой системы

Проекты ИСИМ сохраняются в виде стандартных INI-файлов Windows. Это разрешает при навыках работы с системой редактировать модели, не входя в ИСИМ. Проекты можно создавать, сохранять, переносить, редактировать и т.д. Для этого предназначено подменю **Файл,** в котором собраны все необходимые команды управления проектами.

К общим настройкам системы относятся:

– задание директорий, где находятся необходимые системные файлы (например, файлы системы GPSS/PC и файлы с функциями);

– подтверждение на открытие файлов проекта, GPSS-программы и отчета, которое необходимо для поддержки работы системы с внешними моделями.

Общие настройки сохраняются в каталоге WINDOWS в файле **gsgen.ini.**

8.8. Пример построения модели средствами ISS 2000

Цель исследования. Определение наилучшего варианта технологического процесса ремонта и замены оборудования при обеспечении минимальной себестоимости производства.

Постановка задачи. Некоторый производственный участок имеет L станков, которые работают 24 ч в сутки. Всего в системе есть $M > L$ станков (из них L – собственных, **А** остальные арендуют для резерва). Любой из станков может выйти из строя в любое время. Если станок сломался, его заменяют другим, резервным, **А** сломанный направляют для ремонта в мастерскую. Отремонтированный станок возвращается уже как резервный.

В мастерских есть три специализированных участка для ремонта станков. Технологический цикл ремонта начинается на участке диагностики, где определяются причина выхода из строя оборудования и необходимый вид ремонта. Ремонт выполняется в механических и электронных мастерских. Статистические данные анализа выхода из строя станков показали, что 75% случаев составляет отказ электронного оборудования станков, **А** 25% – механического. Диагностикой занято m_1 рабочих, ремонтом механического оборудования – m_2, а ремонтом электронного оборудования – m_3 рабочих.

Заработная плата рабочих в ремонтной мастерской – W руб. за час, плата за арендованные станки – 5руб. в сутки. Почасовой убыток при использовании менее L станков в производстве составляет Q руб. на станок. Убытки возникают вследствие спада производства.

Опыт эксплуатации показывает, что на диагностику расходуется $A_1 \pm B_1$ часов, на ремонт сломанного электронного оборудования станка – $A_2 + B_2$ часов, **А** на ремонт механического оборудования – $A_3 + B_3$, часов (распределение равномерное). Если станок используется в производстве, время наработки иа отказ имеет экспоненциальное распределение с параметром T часов. Время для перевозки станков из цеха в мастерскую и в обратном направлении незначительно, и его не учитывают. Между рабочими в мастерских нет никаких отличий, как и между станками.

Заработная плата за аренду станков не зависит от того, используют их или нет. Руководителю необходимо определить, сколько рабочих надо нанять для работы в мастерские и сколько станков арендовать, т. е. сколько станков надо иметь в резерве, чтобы можно было бы подменять ими имеющиеся на случай поломок. Цель – минимизация стоимости производства. Длительность времени моделирования Н дней.

Описание модели. Система имеет три основные составляющие:

1) количество рабочих в мастерских;

2) максимальное количество станков, которые одновременно находятся в производстве;

3) общее количество станков, которые находятся в системе.

Для моделирования двух первых составляющих используется МКУ. Общее количество станков моделируется транзактами, то есть станки являются динамическими объектами, которые «перемещаются» с одного места в другое в процессе их использования в системе.

Рассмотрим состояние некоторого станка в процессе полного цикла в системе. Предположим, что станок в данный момент находится в резерве. Тогда МКУ NOWON, используемое для моделирования работающих станков, заполнено, то есть резервные станки не имеют возможности войти в МКУ. Станки, которые находятся в резерве, могут иметь возможность работать, и транзакт, который их моделирует, может сделать это после многочисленных попыток войти в МКУ NOWON, пока одна из них не будет успешной. Проходя сквозь блок **ENTER** в блок **ADVANCE,** транзакт моделирует время работы этого станка, пока последний не сломается.

После того, как станок вышел из строя, транзакт покидает МКУ NOWON, давая возможность другим резервным станкам начать работать, и ждет (если в этом есть необходимость) на входе устройства DIAGN (участок диагностики). В этом случае транзакт сыграет роль ремонтируемого станка. После диагностики он входит или в МКУ ELEK, или в МКУ MEX, то есть для ремонта электронного или механического оборудования. После выполнения ремонта транзакт снова возвращается к той части модели, где он пробует войти в МКУ NOWON.

Так как общее количество станков, которое находится в системе, равняется M(L – собственных и некоторое количество арендованны: для резерва), то это количество задается в начале прогона модели, используя ограничитель блока **GENERATE.**

Данные для моделирования приведены в табл. 8.1.

Таблица 8.1

L	***T***	***$A_1 \pm B_2$***	***$A_2 \pm B_2$***	***$A_3 \pm B_3$***	***H***	***W***	***S***	***Q***
50	160	2±1	30+10	45±5	360	*7,75*	650	120

Построение имитационной модели. Создадим новый проект. По умолчанию он носит имя **Untitled.gsg** (рис. 8.3). Зададим концептуальную структуру модели в виде замкнутой сети СМО, как показано на рис. 8.4. Она состоит из одного генератора и четырех МКУ.

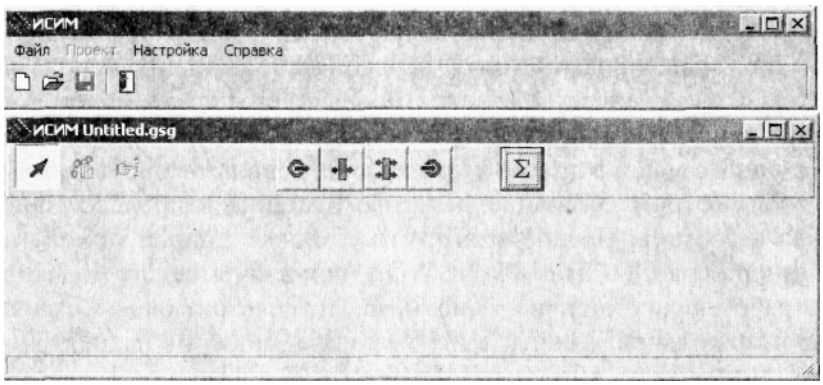


Рис. 8.3

Определим свойства генератора. Зададим два арендованных станка, то есть *M=52.* Генератор только вводит в начальный момент моделирования определенное количество транзактов (52), которые постоянно находятся в модели. Поэтому зададим в свойствах генератора (рис. 8.5) детерминированный закон распределения с нулевым временем и ограниченным количеством транзактов (52).

Определим свойства МКУ с именем NOWON. Для времени обслуживания выберем экспоненциальный закон распределения с параметром 160 и зададим количество устройств – 50 (рис. 8.6).

Определим свойства МКУ с именем DlAGN (рис. 8.6). Зададим два ремонтника на участке диагностики как начальное количество. Определим равномерное распределение времени для ремонта с параметрами 2,1 (рис. 8.7).

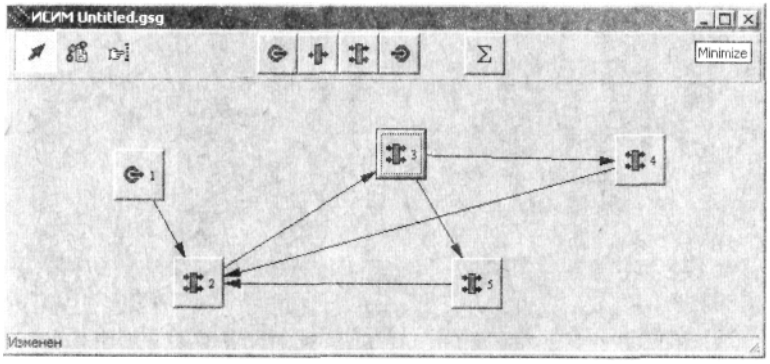


Рис. 8.4

Аналогично определим свойства МКУ с именем ELEK. Зададим три ремонтника на участке ремонта электронного оборудования как первоначальное количество. Определим равномерное распределение времени для ремонта с параметрами 30, 10.

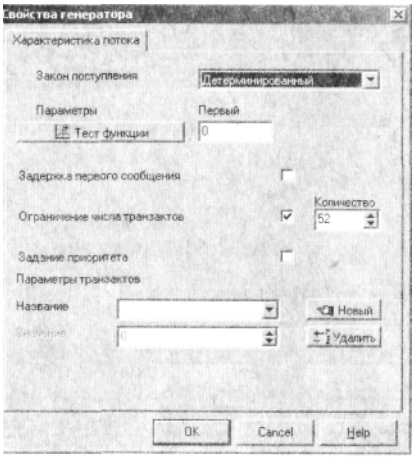


Рис. 8.5

Аналогично определим свойства МКУ с именем МЕХ. Зададим три ремонтника на участке ремонта механического оборудования как начальное количество. Определим равномерное распределение времени для ремонта с параметрами 45, 5.

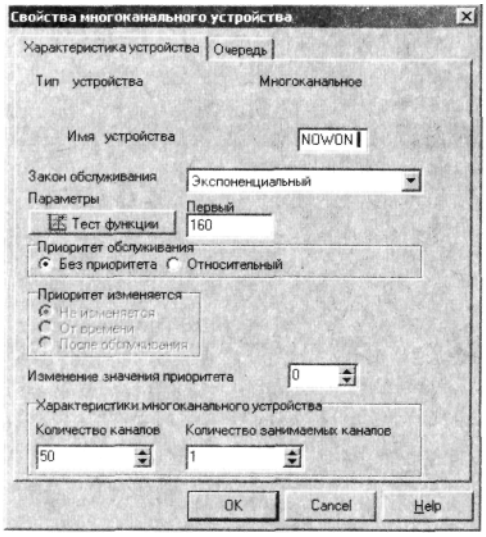


Рис. 8.6

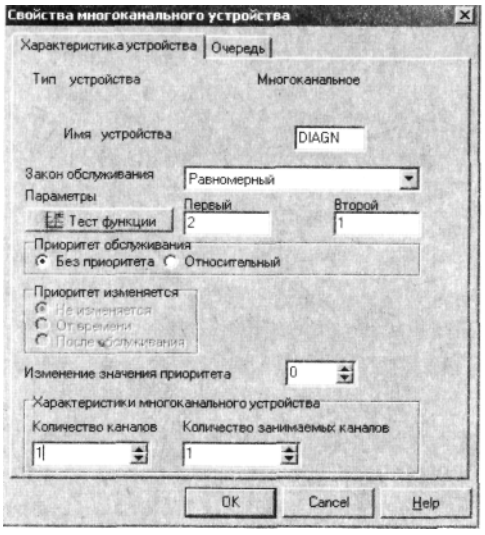


Рис.8.7

Зададим свойства связи для МКУ DIAGN. Выберем передачу транзактов по вероятностному распределению, как показано на рис. 8.8.

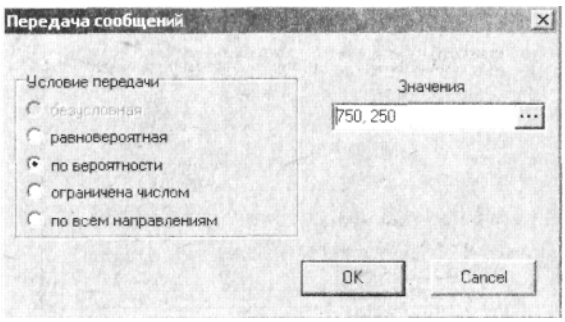


Рис. 8.8

Зададим в меню **Условия эксперимента** время моделирования 8640 часов, как показано на рис. 8.9. Сохраним проект построенной имитационной модели в файле **C:\Мои документы\proekt.gsg.**

Времена ремонта для электронного и механического оборудования будем заносить в таблицы (коллекции), как показано на рис. 8.10. Для этого обозначим маршруты движения транзактов от узла 2 к узлу 4 и от узла 2 к узлу 5.

Зададим верхнюю границу первого частотного интервала 20, ширину интервала 50 и количество интервалов 20 для обоих маршрутов, как показано на рис. 8.11.

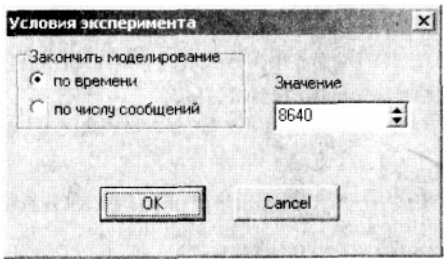


Рис. 8.9

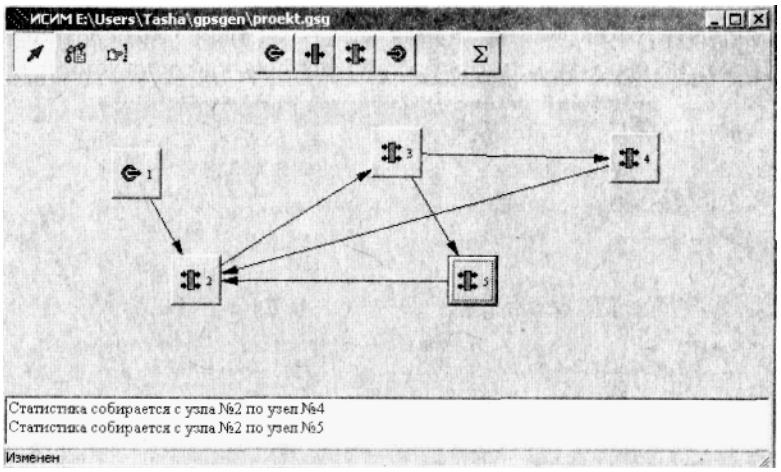


Рис. 8.10

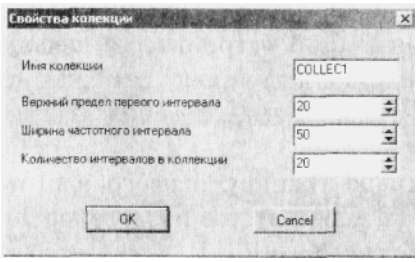


Рис. 8.11

Для расчетов потерь производства необходимо добавить в построенную модель такие переменные:
* Обшее количество станков (собственные и арендованные) **OBL N(NODE1)**
* Количество ремонтников
POTO R(DIAGN)+S(DIAGN)+R(MEN)+S(MEN)+R(ELEKT)+S(ELEKT)
*Плата за арендованные станки
POT1 (OBL-R(NOWON)-S(NOWON)#65#360
* Плата за аренду и зарплата рабочих в мастерских **POT2 POn+POTO#2.75#24#360**
* Общие затраты
NEG_PROFIT POT2+(R(NOWON)+S(NOWON)-SA(NOWON))#120#24#360

Для задания этих переменных следует вызвать построитель формул  . На рис. 8.12 приведен пример формирования переменной POTO.

После задания переменной следует нажать на кнопку **Просмотр кода** и убедится, что переменная задана верно (см. рис. 8.13).

Если переменная задана верно, то нажимаем кнопку **Построитель** и возвращаемся в окно, показанное на рис. 8.12. Нажимаем **ОК.** Задаем свойство формулы – **Общая формула** (рис. 8.14).

Аналогично задаем переменные OBL, POT1, POT2, NEG PROFIT.

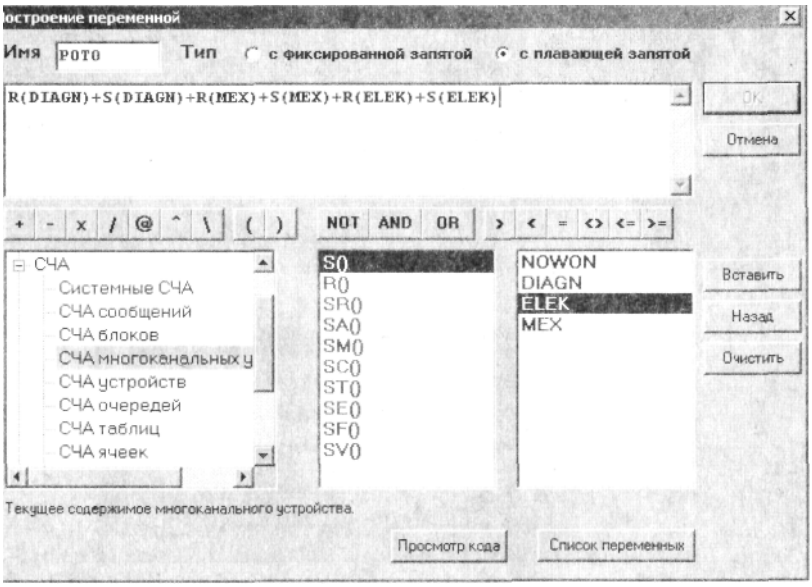


Рис.8.12

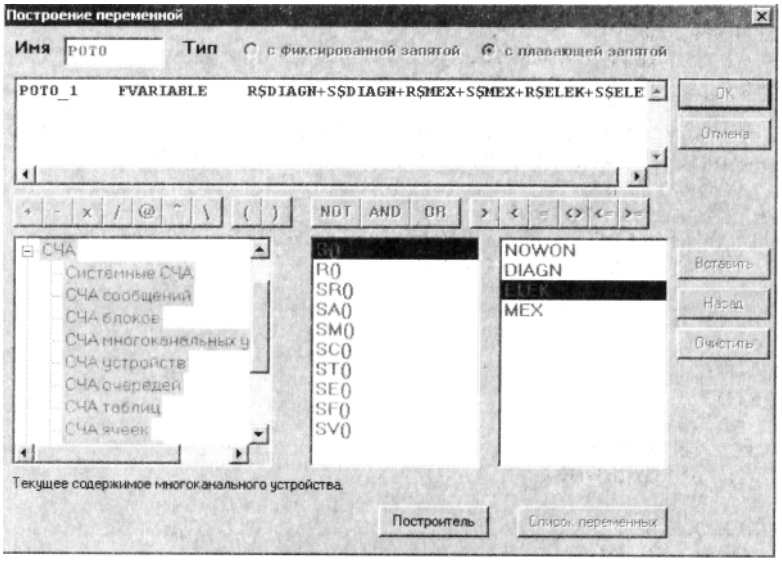


Рис. 8.13

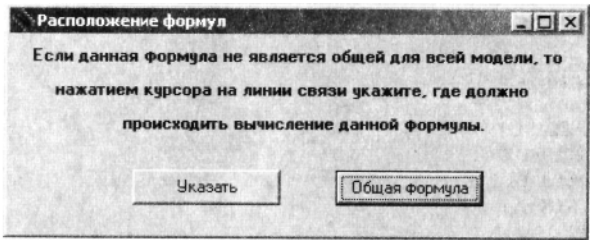


Рис.8.14

Выберем пункт меню **Проект/Построить** для построения проекта **proekt.gsg** и рассмотрим текст GPSS-программы.

```
 NOR      FUNCTION    RN1,C25
0,-5/.00003,-4/.00135,-3/.00621,-2.5/.02275,-2
.06681,-1.5/.11507,-1.2/.15866,-1/.21186,-.8/.27425,-.6
.34458,-.4/.42074,-.2/.5,0/.57926,.2/.65542,.4
.72575,.6/.78814,.8/.84134,1/.88493,1.2/.93319,1.5
.97725,2/.99379,2.5/.99865,3/.99997,4/1,5
XPDIS     FUNCTION    RN1,C24
0,0/.100,.104/.200,.222/.300,.355/.400,.509
.500,.690/.600,.915/.700,1.200/.750,1.380
.800,1.600/.840,1.830/.880,2.120/.900,2.300
.920,2.520/.940,2.810/.950,2.990/.960,3.200
```

```
,970,3.500/.980,3.900/.990,4.600/.995,5.300
,998,6.200/.999,7/1,8
COLLEC1      TABLE M1,20,50,20
COLLEC2      TABLE M1,20,50,20
*************************************************************
POT0_1   FVARIABLE   R$DIAGN+S$DIAGN+R$MEX+S$MEX+R$ELEK+S$ELEK
NEG_PROFIT   FVARIABLE V$POT2_1+(R$NOWON+S$NOWON-
SA$NOWON)#120#24#360
OBL_1   VARIABLE   N$NODE1
POT1_1   FVARIABLE   (V$OBL_1-R$NOWON-S$NOWON)#650#360
POT2_1   FVARIABLE   V$POT1_1+V$POT0_1#360#7.75
*************************************************************
NOWON  STORAGE 50
DIAGN  STORAGE 2
ELEK  STORAGE 3
MEX  STORAGE 3

10 NODE1  GENERATE 0,,,52,
20      TRANSFER ,NODE2

30 NODE2 MARK
40      QUEUE QDIAGN
50      ENTER NOWON,1
60      DEPART QDIAGN
70      ADVANCE 160,FN$XPDIS
80      LEAVE NOWON,1
90      TRANSFER ,NODE3

100 NODE3 QUEUE QUEUE3
110      ENTER DIAGN,1
120      DEPART QUEUE3
130      ADVANCE 2,1
140      LEAVE DIAGN,1
150      TRANSFER .750,NODE4,NODE5

160 NODE4 TABULATE COLLEC1
170      QUEUE QUEUE4
180      ENTER ELEK,1
190      DEPART QUEUE4
200      ADVANCE 30,10
210      LEAVE ELEK,1
220      TRANSFER ,NODE2

230 NODE5 TABULATE COLLEC2
240      QUEUE QUEUE5
250      ENTER MEX,1
260      DEPART QUEUE5
270      ADVANCE 45,5
280      LEAVE MEX,1
290      TRANSFER ,NODE2
300 COMTER TERMINATE

310      GENERATE 8640
320      SAVEVALUE   POT0,V$POT0_1
330      SAVEVALUE   NEG_PROFIT,V$POTERY_1
340      SAVEVALUE   OBL,V$OBL_1
350      SAVEVALUE   POT1,V$POT1_1
360      SAVEVALUE   POT2,V$POT2_1
370      TERMINATE 1
START 1
REPORT REPORT.GPS,NOW
END
```

Функция NOR и блок **COMTER TERMINATE** всегда вставляются в модель. Этот блок необходим для уничтожения транзактов, для которых не выполняются условия (например, превышено время пребывания в очереди). Каждый программный блок узла модели МКУ помечен меткой NODE с номером узла.

Если выполнить эксперимент с построенной моделью, то получим статистические данные (рис. 8.15), значения переменных (рис. 8.16) и таблицы распределения времени для ремонта (рис. 8.17).

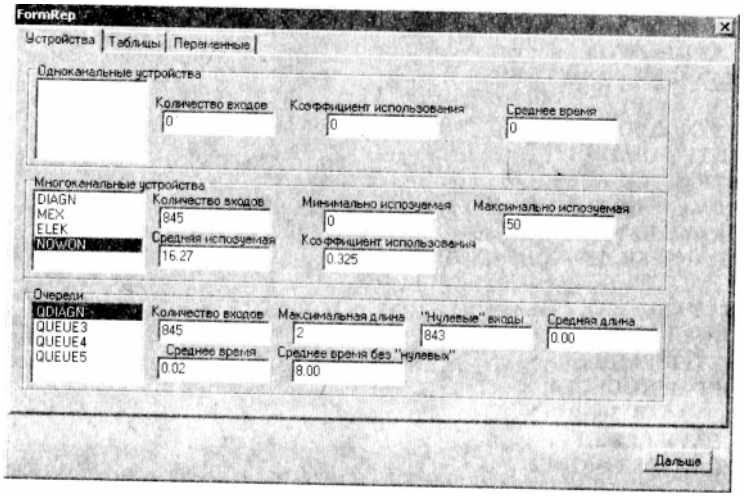


Рис.8.15

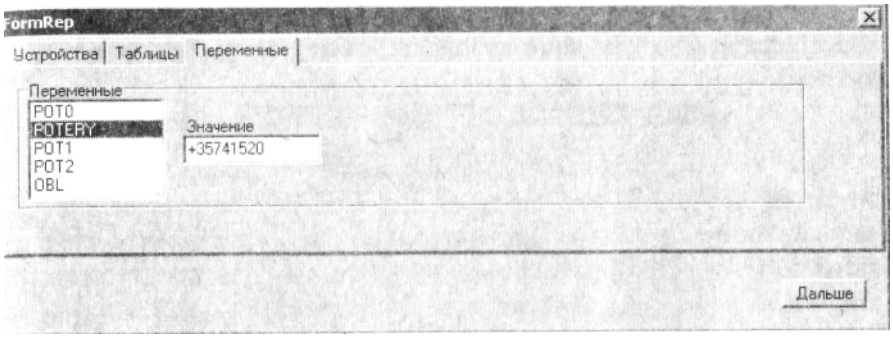


Рис. 8.16

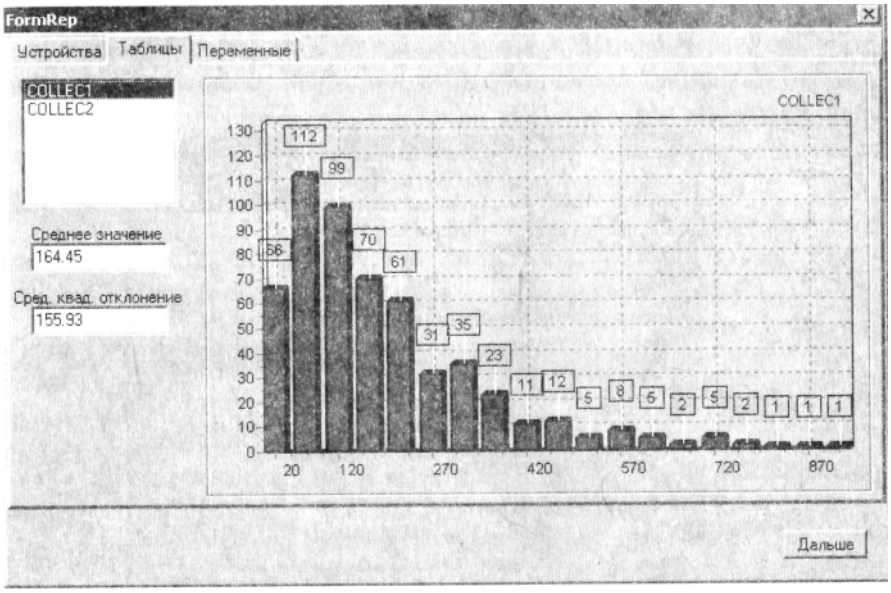
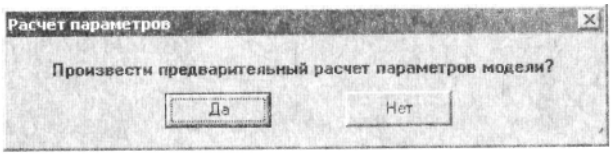


Рис. 8.17

Кроме того, ISS 2000 предоставляет возможность нахождения «узкого места» модели. Для этого выбираем пункт меню **Проект/Построить** и нажимаем кнопку «Да» в появившемся диалоге:



Далее выбираем участок модели, в котором требуется найти узкое место (рис. 8.18):

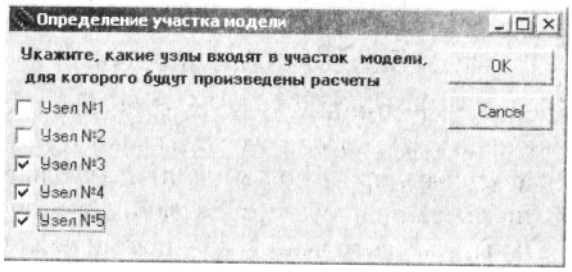


Рис. 8.18

Получаем данные о загрузке каждого из узлов определенного участка, о среднем времени пребывания в каждом узле и о потенциально узком месте (рис. 8.19):

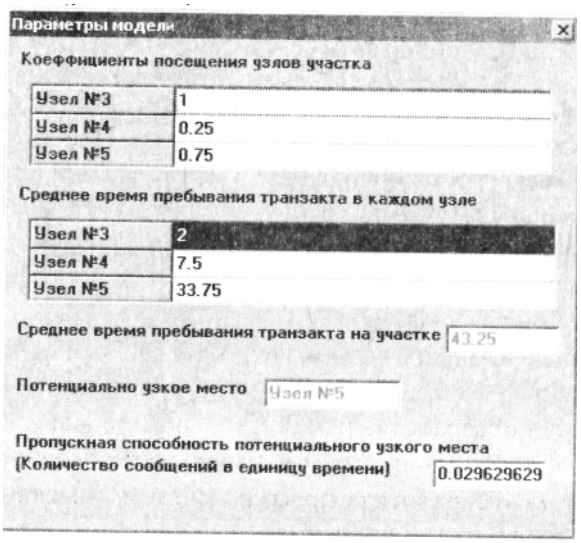


Рис.8.19

Как видим, в нашем примере узким местом оказался узел № 5 – ремонт механического оборудования.

ГЛАВА 9. ТЕХНОЛОГИЯ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ

9.1. Имитационные проекты

Главная ценность имитационного моделирования состоит в том, что в его основу положена методология системного анализа. Она дает возможность исследовать проектируемую или анализируемую систему по технологии операционного исследования, включая такие взаимосвязанные этапы, как содержательная постановка задачи; разработка концептуальной модели; разработка и программная реализация имитационной модели; оценка адекватности модели и точности результатов моделирования; планирование экспериментов; принятие решений. Благодаря этому имитационное моделирование можно применять как универсальный подход для принятия решений в условиях неопределенности и для учета в моделях трудно формализуемых факторов.

Имитационное исследование оформляется в виде документированного проекта, пояснительная записка которого состоит из следующих структурных элементов:

- титульный лист;
- реферат;
- содержание;
- постановка задачи;
- раздел «Анализ возможных методов решения поставленной задачи»;
- раздел «Разработка концептуальной модели»;
- раздел «Выбор программных средств моделирования»;

– раздел «Разработка структурной схемы имитационной модели и описание ее функционирования»;
– раздел «Оценка адекватности модели»;
– раздел «Организация экспериментов с моделью»;
– выводы и рекомендации относительно применения модели;
– перечень ссылок;
– приложения.

Реферат. Реферат предназначен для ознакомления с имитационным проектом. Он должен быть кратким и информативным.

Текст реферата передает полное библиографическое содержание проекта, который выполняют в соответствии с требованиями действующего стандарта относительно библиографического и издательского дела.

Реферат содержит:

– сведения об объекте исследования и количестве иллюстраций, таблиц, приложений и использованных источников (в соответствии с перечнем ссылок на них);
– текст реферата;
– перечень ключевых слов.

В тексте реферата необходимо отобразить приведенную в проекте информацию в такой последовательности:

– объект исследования;
– цель работы;
– методы исследования;
– результаты;
– основные конструктивные, технологические характеристики и показатели;
– значимость работы и выводы;
– прогнозы и предположения относительно развития объекта исследования или разработки.

Части текста реферата, в которых отсутствуют сведения, опускают.

Реферат рекомендуется выполнять в объеме не больее 500 слов.

Ключевые слова или словосочетания, которые являются определяющими для раскрытия содержания (если такие необходимы), размещают после текста реферата в именительном падеже в строку через запятую.

Постановка задачи. Приводится содержательная постановка задачи, определяются цели исследования, внешние воздействия и ограничения, которые накладываются на систему.

Анализ возможных методов решения поставленной задачи.

В данном разделе главе анализируются методы решения поставленной задачи, указываются их преимущества и недостатки, дается четкое обоснование выбора метода решения, указываются источники, по которым проводится обзор методов решения. Приводятся конкретные причины, по которым задача не может быть решена аналитическими методами.

Разработка концептуальной модели. В данном разделе необходимо:

– определить цели моделирования;
– разработать структурную схему модели;
– описать входные, выходные переменные и параметры модели;
– представить функциональные зависимости, описывающие поведение переменных и параметров;
– описать ограничения на возможные изменения величин;
– выбрать степень детализации представления модели;
– сформулировать целевые функции (критерии эффективности) моделируемой системы.

Выбор программных средств моделирования. При предварительном выборе программных средств необходимо определить:

– существует ли хорошо написанные руководство или инструкция для пользователя;
– обеспечивается ли хорошая диагностика ошибок;
– знакомо ли средство программирования модели.

При кратком описании выбранного средства необходимо указать:

- имеющиеся средства генерации случайных чисел и переменных;
- возможности отладки программной реализации модели;
- организацию сбора статистических данных о работе модели;
- возможности отображения структуры моделируемой системы;
- возможности редактирования модели;
- наличие средств автоматизации создания программ.

Разработка структурной схемы имитационной модели и описания ее функционирования.

Описание имитационной модели. В данном подразделе разрабатывается алгоритм моделирования, приводится схема имитационной модели в терминах алгоритма моделирования или выбранного средства моделирования и описывается программная реализация модели. Приводится таблица определений, содержательное значение всех используемых статических и динамических объектов с описанием их свойств.

Описание программной реализации имитационной модели. В данном подразделе дается описание каждого блока модели с комментариями к ним. Для оценки правильности программной реализации имитационной модели проводится пробный эксперимент (прогон модели с тестовыми данными) с целью проверки правильности функционирования программы. Приводятся данные по тестированию модели.

Оценка адекватности модели. В этом разделе выполняется предварительный расчет ожидаемых от модели результатов с помощью операционного анализа сетей СМО (см. главу 2) или метода средних величин. Полученные результаты сравниваются с результатами пробного прогона модели. Обосновывается правильность построения модели путем обратных преобразований (программная модель преобразуется в алгоритм моделирования или логическую схему, **А** затем в концептуальную модель и постановку задачи). Осуществляется подбор тестовых данных для проверки функционирования модели во всем диапазоне исходных данных. Приводятся данные трассировки модели.

Организация экспериментов с моделью.

План экспериментов. Основная цель планирования экспериментов – изучение поведения моделированной системы при наименьших затратах на экспериментирование. Для этого строится план экспериментов. Чаще всего используют такие эксперименты:

- сравнение средних значений и дисперсий разных альтернатив;
- определение важности учета или значимости влияния переменных и ограничений, которые накладываются на эти переменные;
- поиск оптимальных значений переменных на некотором множестве возможных значений.

Разрабатывается ***план экспериментов*** *с* моделью для достижения поставленной цели. При необходимости используют отсеивающий или оптимизирующий эксперименты. В случае оптимизации числового критерия формулируют ***гипотезы*** о выборе наилучших вариантов структур моделируемой системы или режимов ее функционирования, определяют ***диапазон значений параметров*** (режимов функционирования) модели, в границах которых осуществляется поиск оптимального решения.

Оценка точности результатов моделирования. Для оценки точности стохастических моделей строятся доверительные интервалы для получаемых выходных переменных. Если модель работает в переходном режиме, то используют метод повторений экспериментов и дисперсионный анализ. Для стационарных эргодических и регенерирующих процессов определяют длительности прогонов модели, при которых гарантирована точность полученных оценок.

В конце этого раздела указывают затраты компьютерного времени на моделирование, приводят соображения о возможных улучшениях в работе системы.

Анализ и оценка результатов. Приводятся результаты компьютерных экспериментов в виде графиков, таблиц, распечаток, **А** также даются качественные и количественные оценки результатов моделирования.

Поиск наилучших решений. За один прогон модели невозможно определить наилучшие показатели системы или выбрать ее структуру. Процедура поиска наилучших решений всегда оказывается итерационной и циклической. Если осуществляется поиск оптимальных значений на поверхности

отклика, то используют оптимальное планирование экспериментов и численные методы оптимизации. Для выбора наилучшего решения из нескольких альтернатив обычно используют проверки гипотез и выявляют гипотезу победительницу.

Выводы и рекомендации по использованию модели. По полученным результатам формулируются выводы по проведенным исследованиям и определяются рекомендации по использованию модели. Описываются сценарии принятия решений.

Перечень ссылок. В списке литературы необходимо указывать только те источники, на которые есть ссылка в проекте.

Приложения. Приложения содержат тексты программ и другие вспомогательные материалы. Объем приложений не ограничивается.

9.2. Организация экспериментов

С точки зрения представления поведения моделируемой системы имитационные модели относятся к классу описательных. Если в модели учитываются случайные факторы, то в процессе имитации обычно осуществляется большое число прогонов модели, как с разными входными данными, так и с разными значениями последовательностей случайных чисел. Для детерминированной модели (без учета случайностей) достаточно одного прогона модели для каждой комбинации входных данных, однако, в жизни такие модели встречаются крайне редко.

В результате экспериментирования с моделью получают большое количество выходных данных, которые должны быть структурированы и интерпретированы так, чтобы их можно было использовать для принятия решений по результатам моделирования. Для правильной интерпретации полученных от модели выходных данных необходимо организовать эксперименты с моделью.

Организация эксперимента **–** это разработка плана проведения экспериментов, который дает возможность за минимальное число прогонов модели и при минимальной стоимости работ сделать статистически значимые выводы или найти наилучшее решение. При организации эксперимента обычно определяют:

– входные данные для каждого эксперимента;
– количество прогонов имитационной модели;
– длительность одного прогона модели;
– длительность переходного процесса моделирования, после которого необходимо собирать выходные данные;
– стратегию сбора данных для каждого прогона модели;
– методы оценки точности выходных данных с построением доверительных интервалов;
– чувствительность модели к входным данным, различным видам распределений, сценариям поведения моделируемой системы;
– условия эксперимента и сценарии;
– условия генерации потоков случайных чисел внутри системы моделирования и для вероятностных входных данных;
– стратегию достижения цели эксперимента (например, сравнение альтернативных вариантов или оптимизация целевой функции).

Конечная цель проведения экспериментов – это получение достаточной статистической информации для принятия решений по результатам моделирования. Как правило, моделирование проводится с целью нахождения некоторых экстремальных значений характеристик моделируемой системы (оптимизирующий эксперимент) или для выявления важных факторов, влияющих на моделируемую систему (отсеивающий эксперимент). Оба эти эксперимента используют факторные планы и аппроксимируют поверхность отклика полиномами разного порядка, а для поиска экстремальных значений применяются численные методы оптимизации. Для этих экспериментов необходима некоторая функциональная зависимость значений выходной переменной (отклика) от входных переменных или факторов, которая, как правило, отражает критерий эффективности моделируемой системы. Таким образом, поиск наилучшего решения выражается численной характеристикой этого критерия, а для нахождения экстремальных значений необходимо исследовать поверхности отклика (проводить эксперименты) в разных точках. От выбора начальной точки в факторном пространстве во многом зависит эффективность экспериментов.

Другой вид экспериментов, проводимых с моделью, – это структурная оптимизация [21], под которой будем понимать поиск наилучшей структуры моделируемой системы. В этом случае эксперименты проводятся с разными моделями, а не с одной, как в предыдущем случае, причем модели могут отличаться структурой, параметрами и принятыми алгоритмами поведения. Для таких экспериментов нет единого числового критерия оптимизации, что затрудняет использование классических методов. Однако количество рассматриваемых вариантов, как правило, невелико, поэтому для структурной оптимизации можно использовать метод выдвижения гипотез с перебором вариантов. Оптимизация каждого варианта моделируемой системы обычно осуществляется с помощью поиска узких мест и их устранения, т.е. балансировки моделируемой системы. Узкие места определяют пропускную способность всей системы (см. параграф 2.4). Поиск наилучшего решения осуществляется сравнением рассмотренных вариантов.

9.3. Проблемы организации имитационных экспериментов

Перечислим основные проблемы, возникающие при экспериментировании с имитационными моделями.

1. ***Задание начальныхусловии эксперимента.*** Обычно эксперимент начинают из состояния «пусто и свободно», т.е. когда в модели нет транзактов. Если рассматривать достаточно длительный период моделирования, то можно наблюдать так называемый период «разогрева» модели или ***переходной период,*** после которого модель может перейти в стационарный режим работы. Учет данных переходного периода для выходных переменных модели будет вносить смещение в статистические оценки. Чтобы уменьшить влияние данных переходного процесса на конечные результаты, можно поступать следующим образом:

– запускать модель с модальных (наиболее вероятных) значений установившегося режима;

– запускать модель со средних значений установившегося режима.

Эти способы обычно обеспечивают уменьшение длительности переходного процесса модели. При этом они дают эффект только в том случае, если загрузка обслуживающих устройств в модели невелика. При стремлении коэффициентов загрузки устройств к единице, на выходе модели может наблюдаться стационарный процесс, в котором нельзя четко выделить данные переходного режима, как показано на рис. 9.1.

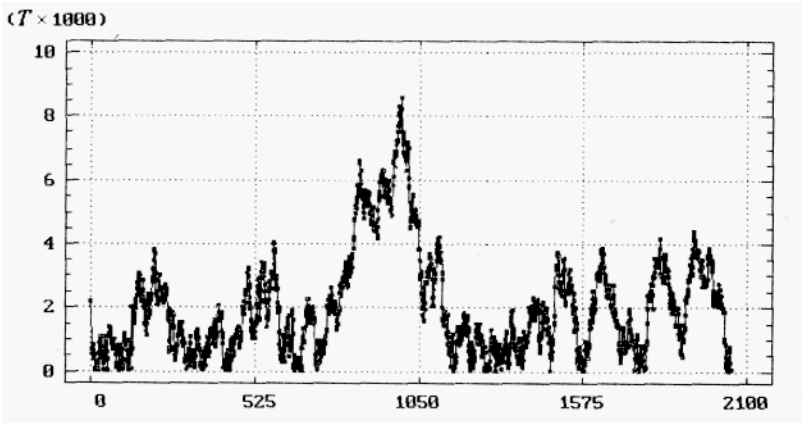


Рис. 9.1

При оценивании выходной величины рекомендуется не учитывать данные переходного процесса, так как онимогут давать существенное смещение искомых оценок. Это достигается путем удаления данных переходного процесса (с помощью команды **RESET).** Лучший способ определения установившегося процесса – это использование графиков для наблюдения за изменением переходного процесса во времени.

Команда **RESET** оставляет транзакты в модели, обнуляет статистику и освобождает устройства обслуживания. Стандартный числовой атрибут С1 – дает значение модельного времени с момента выдачи последней команды **RESET, A** СЧА АС1 – абсолютное модельное время с начала моделирования.

2. ***Правило останова*** определяет длительность имитационного прогона. От продолжительности прогона зависит точность результатов моделирования.

3. ***Состояния модели в момент прекращения прогона.*** Часто при моделировании возникает вопрос: «Что делать с оставшимися компонентами (транзактами) модели в момент окончания ее работы?» Учет оставшихся компонентов может привести к смещению оценки в большую сторону. Например, при моделировании работы некоторого цеха использовалось правило, что наиболее короткие работы запускаются раньше. На момент окончания в модели останутся незавершенные работы с длительными временами выполнения. Если их не учитывать, то оценка средней длительности работ в цеху будет занижена.

4. ***Определение длительности прогона модели при наличии в модели процессов с различными скоростями протекания.*** Оценку точности результатов моделирования обычно выполняют для самого медленного процесса в модели. В этом случае оценки для быстрых процессов будут заведомо намного лучше, чем для медленного процесса, т.е. доверительные интервалы для них будут меньше. При разработке имитационной модели обычно выбирают степень детализации модели так, чтобы скорости протекающих в ней процессов не различались более, чем на два порядка. В случаях моделирования редких событий (медленные процессы), например, отказов оборудования, необходимо укрупнять состояния для быстрых процессов. Для этого обычно используют аналитико-имитационные модели.

9.4. Оценка точности результатов моделирования

Оценка точности результатов моделирования связана с построением доверительных интервалов для выходных переменных (откликов) модели. Количество реализаций (прогонов модели) и время прогона для каждой реализации модели определяют точность результатов. Если модель детерминированная, то для получения точных результатов моделирования достаточно одного прогона. В общем случае данные одного прогона модели представляют единичную выборку или временной ряд. ***Временной ряд*** **–** это конечная реализация случайного процесса, т.е. в результате каждого прогона модели образуются временные ряды для каждого значения отклика модели исследуемых стохастических процессов. Для стохастических моделей рассматривают два режима работы: переходный и стационарный. Стационарный режим определяется стационарным процессом на выходе модели.

Если модель работает в переходном режиме, то необходимое количество прогонов модели можно рассчитать по тем же формулам, что и для метода статистических испытаний. Количество прогонов модели определяется в соответствии с формулами (3.23) и (3.26). Необходимую точность **ε** можно задать равной ± 5% от среднего значения величины, для которой строится доверительный интервал при α = 0,96. Если модель реализована на языке GPSS, то необходимо после последнего прогона выполнить процедуру **ANOVA.**

Если число прогонов небольшое (менее тридцати), то при построении доверительного интервала используют распределение Стьюдента (t-распределение). При большем числе прогонов можно использовать функцию нормального распределения.

Если критерием оценки является стоимостная характеристика (доход, прибыль, затраты и т. п.), которая определяется для стационарного режима работы модели, то длина прогона может быть определена по результатам наблюдения за изменением величины, равной отношению оцениваемого показателя за весь период моделирования к продолжительности моделирования (например, затраты за единицу времени). Для этого в окне **PLOT** строят график изменения этой величины. Длина прогона должна соответствовать стационарному режиму функционирования модели (рис. 9.2). Статистические данные переходного периода работы модели не должны учитываться. Для этого используют команду **RESET** (см. параграф 4.27).

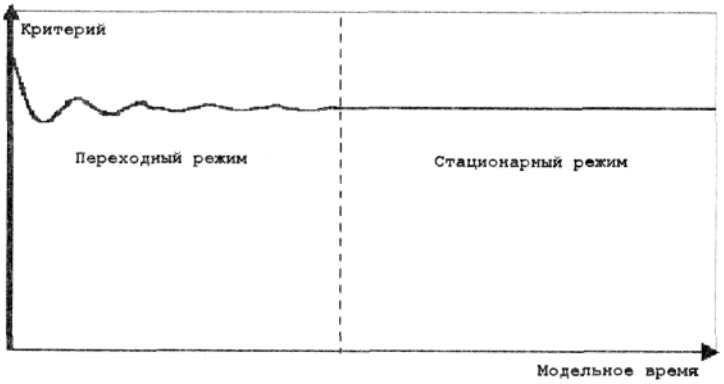


Рис. 9.2

Стационарность процесса с выхода модели можно также проверить, наблюдая за его *автоковариационной функцией.* Если она имеет тенденцию к затуханию, то это свидетельствует в большинстве случаев о том, что в первом приближении процесс стационарный. Поскольку автоковариационная функция случайного процесса неизвестна, то ее можно только оценить. Учитывая то, что полученные из имитационной модели временные ряды достаточно длинные, оценивают автоковариационную функцию cov_h по формуле [12]

$$\hat{cov}_h = \frac{1}{N}\sum_{i=1}^{N-h}(X_i X_{i+h}) - \frac{1}{N-h}\left(\sum_{i=1}^{N-h} X_i\right)\left(\sum_{i=1}^{N-h} X_{i+h}\right),$$

где N — количество точек во временном ряду; h – сдвиг по ряду; X_i – i-значение измеряемой переменной (i-я точка).

Особое значение при имитационном моделировании имеют стационарные эргодические процессы, свойства которых могут быть оценены по результатам одного временного ряда.

Последовательность выборочных средних значений $\{\overline{X}_N\}, N = 1,2,...\infty$ будет ***эргодической,*** если дисперсия величины $\overline{X}_N$ стремится к нулю при неограниченном увеличении s :

$$VAR[\overline{X}_N] \xrightarrow[N\to\infty]{} 0 \ .$$

Таким образом, выборочное среднее асимптотически стремится к математическому ожиданию, если дисперсия выборочного среднего стремится к нулю. В этом случае установившийся стационарный режим работы модели не будет зависеть от начальных условий моделирования.

Чтобы проверить на практике, является процесс эргодическим или нет, необходимо резко изменить начальные условия моделирования. Если модель сначала запускалась при условиях, что очереди были пустые, а устройства свободные, то второй прогон модели следует провести при наличии транзактов в очередях и устройствах. Если на выходе модели будут получены близкие результаты, то это обычно свидетельствует об эргодичности процесса.

В системе GPSS изменять условие моделирования можно, задав несколько блоков **GENERATE и TRANSFER:**

```
GENERATE    ,,,N
TRANSFER    ,LABEL
```

Здесь метка **LABEL** указывает на вход устройства, а N – на количество транзактов, которые посылаются в это устройство. Таким образом, в начальный момент в модели будут находиться уже N транзактов.

Для стационарных режимов работы системы, модель которой регенерирует (повторяется в вероятностном смысле), используют метод построения доверительных интервалов [15]. Пример 9.1 построения доверительного интервала для регенерирующих процессов приведен для СМО с одним устройством обслуживания в программе **Regenerative method.**

В языке GPSS статистика о работе модели собирается автоматически или по желанию пользователя и выводится в файл результатов. Однако при этом не гарантируется надежность полученных оценок. Этим вопросом должен заниматься сам пользователь.

Для имитационных стохастических моделей, которые работают в переходном режиме, необходимо выполнить несколько прогонов модели, каждый из которых должен отличаться своей последовательностью псевдослучайных чисел. Для этого используется последовательность команд, аналогичная такой:

```
RMULT     713
START     1
...
CLEAR
...
RMULT     617
START     1
```

Команда **RMULT** разрешает устанавливать начальные значения множителей для генераторов случайных чисел. Команда **CLEAR** осуществляет сброс собранной статистики в предыдущем прогоне модели и удаляет транзакты из модели, но не устанавливает множители генераторов случайных чисел в начальные значения.

При моделировании стохастических систем, работающих в стационарном режиме, может быть использован регенерирующий анализ, если эти системы регенерируют. Для СМО моменты регенерации определяются номерами тех требований, которые будут, например, заставать устройство обслуживания свободным. Класс регенерирующих систем довольно большой. К нему относятся стохастические сети СМО, система управления запасами и прочие системы. Поэтому приведем алгоритм построения 100(1-δ)% доверительного интервала с использованием этого метода [15].

1. Провести n циклов регенерации.

2. Вычислить Y_j и α_j для любого j-го цикла, где Y_j – сумма исходных значений переменной, полученной от имитационной модели на j-м цикле регенерации, и α_j – длина j-го цикла (количество транзактов, которые попали в j-й цикл).

3. Вычислить выборочные статистики

$$\bar{Y} = \frac{1}{n}\sum_{j=1}^{n} Y_j, \qquad \bar{\alpha} = \frac{1}{n}\sum_{j=1}^{n} \alpha_j, \qquad \hat{r} = \frac{\bar{Y}}{\bar{\alpha}};$$

$$S_{11} = \frac{1}{n-1}\sum_{j=1}^{n} Y_j^2 - \frac{1}{n(n-1)}\left(\sum_{j=1}^{n} Y_j\right)^2;$$

$$S_{22} = \frac{1}{n-1}\sum_{j=1}^{n} \acute{a}_j^2 - \frac{1}{n(n-1)}\left(\sum_{j=1}^{n} \acute{a}_j\right)^2;$$

$$S_{12} = \frac{1}{n-1}\sum_{j=1}^{n} Y\acute{a}_j - \frac{1}{n(n-1)}\left(\sum_{j=1}^{n} Y_j\right)\left(\sum_{j=1}^{n} \acute{a}_j\right);$$

$$S^2 = S_{11} - 2\hat{r}^2 S_{12} - \hat{r}^2 S_{22},$$

где $\hat{r}$ – среднее значение переменной, которая оценивается в имитационной модели; s_{11}, s_{22}, s_{12} – соответственно выборочные дисперсии значений Y_j и α_j и выборочный второй смешанный момент значений (Y_j, α_j).

4. Сформировать доверительный интервал

$$\hat{r} \pm \frac{z_\delta^*}{\bar{\alpha}\sqrt{n}},$$

где $z_\delta^* = \hat{O}^{-1}(1 - \frac{\delta}{2})$; Φ – функция стандартизированного нормального распределения. Значение z^*_δ = 1,645 для δ = 0,9; z^*_δ = 1,96 для δ = 0,95.

Если начало первого цикла не совпадает с началом моделирования, то данные, предшествующие первому циклу, необходимо отбросить.

Пример 9.1 [18]

Покажем, как можно использовать данный алгоритм в GPSS-программе. Ниже приводится GPSS-программа для моделирования СМО вида M/M/1, для которой оценивается значение времени пребывания требования в системе и строится доверительный интервал с 90-процентным уровнем доверия.

Программа:

```
* Regenerative method
* Author V. Tomashevskiy
EXPD      FUNCTION      RN1,C24
0,0/.100,.104/.200,.222/.300,.355/.400,.509
.500,.690/.600,.915/.700,1.200/.750,1.380
.800,1.600/.840,1.830/.880,2.120/.900,2.300
.920,2.520/.940,2.810/.950,2.990/.960,3.200
.970,3.500/.980,3.900/.990,4.600/.995,5.300
.998,6.200/.999,7/1,8
          INITIAL     X1,0
          INITIAL     X2,0
          INITIAL     X3,0
          INITIAL     X4,0
          INITIAL     X5,0
          INITIAL     X6,0
          INITIAL     X7,0
          INITIAL     X8,0
          INITIAL     X9,0
41 BAD2   TABLE       M1,0,500,30                ; Таблица времени пребывания
51 KBA    VARIABLE    X2^2                       ; Расчет α
61 KBY    VARIABLE    X1^2                       ;
71 ALY    VARIABLE    X1#X2                      ; Расчет Y
81 YSR    FVARIABLE   X3/X4                      ; Расчет Ȳ
91 NNN    FVARIABLE   1/(X8-1)                   ; Расчет 1/(n-1)
101 NN1   FVARIABLE   X8/V$NNN                   ; Расчет n/(n-1)
111 SS11  FVARIABLE   X6/(X8-1)-X3/X8#X3/(1-X8)  ; Расчет S11
121 SS22  FVARIABLE   X5/(X8-1)-X4/X8#X4/(1-X8)  ; Расчет S22
131 SS12  FVARIABLE   X7/(X8-1)-X3/X8#X4/(X8-1)  ; Расчет S12
*                                                Расчет S^2
141 SKB   FVARIABLE   V$SS11-2#V$YSR#V$SS12+V$YSR^2#V$SS22
142*
151       GENERATE    200,FN$EXPD                ; Генерация потока требований
161 INP   QUEUE       BAD1                       ; Постановка в очередь
171       SEIZE       BAD1                       ; Занятие устройства
181       DEPART      BAD1                       ; Освобождение очереди
191       ADVANCE     180,FN$EXPD                ; Обслуживание
201 OUT   RELEASE     BAD1                       ; Освобождение устройства
211       TEST NE     N$OUT,N$INP,CIKL           ; Проверка начала цикла
221       SAVEVALUE   2+,1                       ; Подсчет j в цикле
231       SAVEVALUE   1+,M1                      ; Подсчет α_j в цикле
241       TABULATE    BAD2                       ; Табулирование времени пребывания
251       TERMINATE
261 CIKL  SAVEVALUE   2+,1                       ; Подсчет последнего j в цикле
271       SAVEVALUE   1+,M1                      ; Подсчет последнего α_j в цикле
281       SAVEVALUE   3+,X1                      ; Σ Y_j
291       SAVEVALUE   4+,X2                      ; Σα_j
301       SAVEVALUE   5+,V$KBA                   ; Σα_j^2
311       SAVEVALUE   6+,V$KBY                   ; Σ Y_j^2
321       SAVEVALUE   7+,V$ALY                   ; Σ α_j *Y_j
331       SAVEVALUE   1,0                        ; Начало нового цикла
341 REG   SAVEVALUE   2,0                        ;
351       TEST NE     X8,N$REG,ENDREG            ; Закончить моделирование?
*
361       SAVEVALUE   8+,1                       ; Подсчет числа циклов
371       TABULATE    BAD2                       ; Табулирование последнего
*                                                ; значения времени пребывания
*                                                ; в цикле
          TEST E      TG1,1,END_REG              ; Последний цикл?
          SAVEVALUE   MEIN,(X3/X4)               ; Запомнить среднее значение
*         Запомнить величину разброса 90%-го доверительного интервала
          SAVEVALUE   DELTA,(1.645#SQR(V$SKB)#X8/X4/SQR(X8))
*
381 END_REG TERMINATE 1
391 ENDREG  SAVEVALUE 8+,1                       ; Учет последнего цикла
401         SAVEVALUE 9,V$SKB                    ; Запомнить значение S^2

411       TERMINATE   1

START     1000        Закончить моделирование через 1000 циклов
```

Так как моделируется СМО вида M/M/1, то согласно формуле (1.7) при C=1 теоретическое среднее время пребывания в СМО T= 1800, а результаты моделирования дают T = 1833,053±440,258 для 90% доверительной вероятности (MEIN ± DELTA). Теоретический коэффициент загрузки равен 0,9, а по результатам моделирования 0,901. Как видно, полученные результаты близкие к теоретическим.

Приведенная программа может быть использована для лвзбой регенерирующей модели. Для этого между метками INP и OUT нужно вставить программу регенерирующей модели. Проверка циклов регенерации осуществляется по количеству транзактов, которые находятся между этими метками. Цикл регенерации начинается, если в модели между указанными метками нет ни одного транзакта.

При моделировании систем управления запасами цикл начинается, когда достигается минимальная величина запаса, поэтому для таких моделей необходимо изменить условие проверки начала цикла регенерации.

9.5. Факторный план

При экспериментировании с моделью различают входные и выходные переменные. Входные переменные называются ***факторами.*** Выходные переменные называются ***откликами.*** Каждый фактор в эксперименте может принимать одно или несколько значений, называемыми ***уровнями фактора.*** Множество уровней факторов определяет одно из возможных состояний моделируемой системы и представляет условия проведения одного из возможных экспериментов. Существует определенная связь между уровнями факторов и откликами системы, которая обычно заранее неизвестны. Эту связь можно определить следующим образом:

$$y_l = \psi_l(x_1, x_2, ..x_i .., x_m), \; l = \overline{1,n},$$

где y_l –l-й отклик, n – число анализируемых откликов, x_i – i-й фактор, m – число факторов.

Функция ψ в правой части называется ***функцией отклика*** или реакции. Ее геометрический образ – поверхность отклика. Так как функция ψ заранее не известна, то используют другую приближенную функцию:

$$\hat{y}_l = \varphi_l(x_1, x_2, ..x_i .., x_m), \; l = \overline{1,n}.$$

Эти функции φ_l находят по данным эксперимента и представляют в виде степенного полинома первого, второго и, реже, третьего порядка. После проведения экспериментов аппроксимирующие полиномы заменяют уравнениями регрессии и методом наименьших квадратов находят статистические оценки их неизвестных коэффициентов.

Факторный эксперимент может быть отсеивающий, когда из всего множества факторов определяются те факторы, которые существенно влияют на отклики модели. Второй вид факторного эксперимента используется для определения экстремальных значений на поверхности отклика. В этом случае серия факторных экспериментов планируется так, чтобы достичь экстремума на поверхности отклика.

Факторный эксперимент представляет собой план, в котором все уровни каждого фактора встречаются в сочетании со всеми уровнями всех других факторов. Различные уровни некоторого фактора могут соответствовать качественным значениям (например, разные дисциплины обслуживания в устройстве) или количественным значениям (например, число устройств обслуживания). Если фактор f, (f= 1,...,K) имеет L_f уровней, то общее число комбинаций уровней определяется произведением:

$$L_1,..., L_k = \prod_{f=1}^{k} L_f. \qquad (9.1)$$

Если число уровней для каждого из факторов одинаково, то общее число комбинаций будет L^k.

Левая часть выражения (9.1) используется для обозначения факторного плана.

Применение факторного плана вместо классической схемы, согласно которой каждый раз изменяется только один фактор, имеет ряд преимуществ.

- Становится более полной картина влияния каждого фактора, поскольку они изучаются в самых различных условиях (вследствие одновременного изменения других факторов).
- Большое число комбинаций факторов, используемых в эксперименте, облегчает предсказание результатов, которые могут быть достигнуты при определенной комбинации условий.
- Если эффекты, вызываемые каждым фактором, статистически независимы, то о каждом факторе можно получить не меньше информации, чем при изменении в экспериментах только одного фактора при фиксации остальных.
- Если (как это часто бывает) различные факторы не являются независимыми, а вызывают эффекты, которые в большей или меньшей степени коррелированны, то в этом случае только факторный эксперимент может дать информацию о характере этих взаимодействий. При наличии нескольких

взаимосвязанных существенных факторов обойтись без постановки факторного эксперимента невозможно. Для ряда часто встречающихся специальных задач разработано большое число стандартных факторных планов.

Рассмотрим пример 2-х факторного эксперимента, с двумя факторами на 2-х уровнях и с двумя наблюдениями в каждом опыте, т.е. план 2^2. Факторы принято обозначать буквами латинского алфавита *A,B,* С и т.д.

Результаты экспериментов сведем в таблицу 9.1.

Таблица 9.1

Фактор A	***Фактор B***	
	Уровень 1	***Уровень 2***
Уровень 1	y_{111} y_{112}	y_{121} y_{122}
Уровень 2	y_{211} y_{212}	y_{221} y_{222}

В этой таблице y_{ijg} обозначает *g-e* наблюдение *(g* = 1,2) в ячейке *i*, *j*. Количество наблюдений (прогонов модели) *g* определяется желаемой точностью получения оценок откликов.

В общем случае в 2-х факторном эксперименте число уровней факторов **A** и *B* равно соответственно *I* и *J*. Обозначим математическое ожидание $E(y_{ijg})=\eta_{ij}$, тогда в планировании эксперимента предполагается верной следующая модель:

$$y_{ijg}=\eta_{ij}+e_{ijg} \quad \left(i=\overline{1,I};\ \ j=\overline{1,J};\ \ g=1,2,3..\right),$$

где e_{ijg} – ошибка опыта. Предполагается, что все эти ошибки являются независимыми нормально распределенными случайными величинами с математическим ожиданием 0 и дисперсией σ^2. При имитации ошибки опытов можно сделать независимыми, применяя различные последовательности случайных чисел при прогонах модели.

9.6. Дисперсионный анализ ANOVA в планировании экспериментов

Для определения, является фактор значимым или нет, используется дисперсионный анализ ANOVA (analysis of variance), который применим только к количественным факторам. С помощью него определяются количественные отклонения наблюдений от среднего значения. Если какой-либо фактор не оказывает влияние на отклик, то он является незначимым. С другой стороны, если фактор влияет на отклик, то его (фактора) количественное значение сравнивают с оценкой изменчивости наблюдения, то есть со стандартной ошибкой.

Это делается для исключения эффектов, которые являются не более чем случайной флуктуацией.

Неявно в ANOVA используется аддитивная математическая модель, которая определяет компоненты изменения в наблюдениях. Ее называют статистической моделью. Самая простая статистическая модель:

$$y_{ig} = \mu + e_{ig},$$

т.е. каждое *i*-е наблюдение представляет собой общее среднее по всем опытам μ, и случайную ошибку e_{ig}. В этой модели общее среднее не изменяется от опыта к опыту, в отличие от ошибки.

Статистическая модель для анализа данных экспериментов с одним фактором **A** имеет следующий вид:

$$y_{ig} = \mu + \alpha_i^A + e_{ig},$$

где α^A_i – главный эффект фактора **A** на уровне *i*. Все наблюдения на данном уровне обработки анализируются, используя то же самое значение для α^A. Так как в этом эксперименте имеется только один фак гор, число комбинаций обработки определяется числом уровней *I* этого фактора.

Для двух факторов общая модель факторного плана такова:

$$y_{ijg} = \mu + \alpha_i^A + \alpha_j^B + \alpha_{ij}^{AB} + e_{ijg}, \qquad (9.2)$$

где α^{B}_{j} – главный эффект фактора *B* на уровне *j*; α^{AB}_{ij} – взаимодействие фактора **A** на уровне *i* и фактора *B* на уровне *j*. Сумма эффектов двух факторов не равна сумме их отдельных эффектов из-за взаимодействия между ними. Главный эффект фактора определяет долю участия фактора в значении функции отклика во время перехода его с нижнего уровня к верхнему.

Дисперсионный анализ, основанный на статистической модели (9.2), заканчивается построением таблицы ANOVA, в которой анализируется влияние факторов *A, B,* взаимодействие между факторами *AB* и случайные помехи наблюдения.

С помощью ANOVA проверяется гипотеза об отсутствии влияния фактора. Если справедлива гипотеза об отсутствии влияния фактора, то считается, что все наблюдения получены из одной генеральной совокупности. Для проверки гипотезы используется *F*-распределение Фишера. Критерий Фишера определяет отношение двух выборочных дисперсий. Если фактор существенно влияет на отклик, то значения F-распределения принимает большие значения и F-статистика становится значимой. Таким образом, большие значения *F* приводят к отбрасыванию гипотезы об отсутствии влияния фактора, т.е. фактор является ***значимым.***

9.7. Библиотечная процедура ANOVA

Библиотечная процедура ANOVA системы GPSS World анализирует эксперименты от 1 до 6 факторов, включая взаимодействия 2-го и 3-го порядка между факторами.

На рис. 9.3 [19] представлена таблица ANOVA, полученная в GPSS World. Сначала рассмотрим среднюю часть таблицы. Полная сумма квадратов (Total) отделена от компонентов, связанных с эффектами факторов и их взаимодействиями (A, B, AB). В строке Error (ошибка) приведена остаточная сумма квадратов. Средняя сумма квадратов (Mean Square) остаточного члена используется для оценки стандартной ошибки эксперимента (в данном случае это величина 2,5).

Каждая сумма квадратов делится на число степеней свободы для уровней. Из статистических соображений степени свободы – это делитель, который должен использоваться для получения несмещенной оценки стандартной ошибки. Для наших целей, достаточно представлять степени свободы как соответствующий делитель, связанный с суммой квадратов в таблице ANOVA. Система GPSS World всегда вычисляет степени свободы.

Каждый фактор и взаимодействие в статистической модели представлены отдельной строкой в верхней части таблицы ANOVA. В каждой строке указана сумма квадратов и число степеней свободы, связанные с оценкой факторов и их взаимодействий. Это – основы, из которых получены другие числа. Частное от деления определяет средний квадрат ошибки, и в предпоследнем столбце таблицы выдается F-статистика для этого эффекта.

Сделаем некоторые заключения. Необходимо решить, достаточно ли большое значение F-критерия получено для объявления эффекта значимым. Пороговое значение, которое используется для сравнения, называется «критическим значением F» и помещено справа от F-статистики в той же самой строке. Если полученное значение *F* превышает критическое значение, то делаем заключение, что имеем дело со значимым эффектом фактора. Если нет, то считаем, что эффект фактора незначимый и игнорируем любое связанное *с* ним изменение в наблюдениях, считая, что оно вызвано случайными помехами. Представленная на рис. 9.3 таблица ANOVA показывает, что эффект фактора **A** значимый, **A** эффект фактора *B* и взаимодействие *AB* незначимы.

Иногда при выполнении эксперимента невозможно обнаружить эффект даже в том случае, если он фактически существует. Одна из задач эксперимента заключается в том, чтобы сделать это маловероятным. Из таблицы ANOVA видно, что для получения лучших результатов необходимо иметь или большую F-статистику или меньшее значение F-критерия. Желательно удалить часть суммы квадратов ошибки из-за какого-либо важного эффекта, не включенного в анализ. Если это можно сделать, то F-статистика будет больше. Для этого определяют дополнительные факторы, которые должны быть включены в эксперимент.

ANOVA

Source of Variance	Sum of Squares	Degrees of Freedom	Mean Square	F	Critical Value of F (p=.05)
A	28.000	2	14.000	5.600	3.89
B	21.000	3	7.000	2.800	3.49
AB	11.000	6	1.833	0.733	3.00
Error	30.000	12	2.500		
Total	90.000	23			

Treatment Level A B	Count	Mean	Minimum	Maximum	95% C.I. (SE)
1 1	2	3.000	2.000	4.000	(0.564, 5.436)
1 2	2	1.000	1.000	1.000	(-1.436, 3.436)

01/27/01 13:26:44 1.5811388

Рис. 9.3

Для увеличения степени свободы остаточного члена можно использовать два подхода. Первый просто увеличивает число повторений в эксперименте. Этот подход обычно более дорогой, но может быть весьма эффективным. Второй касается плана эксперимента и статистической модели дисперсионного анализа. Средний квадрат ошибки – фактически остаточный член, оставшийся после удаления других квадратов. Если можно найти приемлемый способ, который позволит большему количеству данных оставаться после удаления эффектов, то получим оценку стандартной ошибки с большим числом степеней свободы. Окончательное значение F-критерия будет уменьшаться при увеличении мощности анализа. При этом фактически игнорируются некоторые из взаимодействий.

В многофакторных экспериментах для упрощения статистической модели и уменьшения количества экспериментов игнорируют взаимодействия самого высокого порядка. Например, статистическая модель с двумя факторами без учета их взаимодействия имеет следующий вид:

$$y_{ijg} = \mu + \alpha_i^A + \alpha_j^B + e_{ijg}.$$

Если действительно эффектами этих взаимодействий можно пренебречь, то это позволит использовать дополнительные степени свободы для получения лучших оценки F-статистики. Кроме того, большее число степеней свободы означает также и меньшее значение F-критерия. Однако необходимо понимать, что удаление членов из статистической модели предполагает, что не существует взаимодействий более высоких порядков.

В GPSS World от учета взаимодействий высоких порядков можно отказаться, используя третий параметр процедуры ANOVA. Иногда целесообразней учитывать только одну случайность. В этом случае для улучшения статистических оценок просто добавляются повторения наблюдений. Число повторений задается во втором параметре процедуры ANOVA.

Для работы процедуры ANOVA ей необходимо передать имя GPSS-матрицы с сохраненными результатами всех прогонов модели. Можно иметь несколько матриц результатов и для каждой их них выполнить эту процедуру. Любые матрицы GPSS World могут иметь максимальную размерность 6, эта величина ограничивает максимальное число анализируемых факторов.

Прежде чем начать эксперимент, необходимо инициализировать элементы матрицы в НЕОПРЕДЕЛЕННОЕ (UNSPECIFIED) состояние с помощью оператора:

INITIAL MyResuItMatrix,UNSPECIFIED

Этот оператор сообщает программе ANOVA, что эксперимент не был закончен.

Позиция в матрице результатов для каждого результата эксперимента определена комбинацией уровней обработки, как было показано в таблице 9.1. Например, если в эксперименте рассматривается 3

вида рабочих и два значения их численности (минимальное и максимальное) для каждого вида, то результат моделирования для третьего вида с максимальной численностью помещается в матрицу результатов в позицию [3, 2]. Так как число повторений указывается в третьем индексе матрицы, то результат первого прогона этой комбинации уровней обработки помещается в элемент [3, 2, 1] матрицы результатов. Каждое измерение в матрице результатов анализируется библиотечной процедурой ANOVA, как фактор. Размерность матрицы результатов должна быть, по крайней мере, такой же, как и число уровней обработок этого фактора или максимальное рассчитанное число повторений. Нет никакого предела для числа уровней обработок факторов (вернее, предел накладывается только виртуальной памятью используемого компьютера).

Пример 9.2

```
MyResults   MATRIX     ,8,5,7,10
INITIAL     MyResults,UNSPECIFIED
```

В этом примере созданная матрица результатов может содержать результаты эксперимента с 4-мя факторами (по числу заданных операндов – B, C, D, E) или с 3-мя факторами (операнды B, C, D) и 10-ю повторениями прогонов (последний операнд). Фактор **A** может иметь до 8-х уровней обработок, фактор *B* может иметь до 5 и так далее.

Оператор INITIAL устанавливает все элементы в матрице в состояние UNSPECIFIED.

При обращении к процедуре ANOVA надо задать 3 параметра. Первый – имя матрицы результатов, например, MyResults.

Второй необязательный параметр – размерность (индекс) матрицы результатов, которую нужно использовать для повторений. Каждый уровень в этой размерности представляет прогон с различными начальными значениями случайных чисел. Процедура ANOVA использует информацию, связанную с числом повторений прогонов для оценки стандартной ошибки. Чем больше повторений, тем больше число степеней свободы и тем точнее статистические результаты моделирования.

Некоторые экспериментальные планы не используют размерность повторения. В том случае второй параметр равен 0.

Третий параметр задает взаимодействия между факторами, которые будут включены в статистическую модель. Если параметр равен 2, то взаимодействия второго порядка будут включены в анализ. Если параметр равен 1, то взаимодействия не будут учитываться. При равенстве значения параметра 3 – взаимодействия 2-го и 3-го порядков будут включены в статистическую модель. Взаимодействия более высоких порядков GPSS World не поддерживает. Еще раз отметим, что если при экспериментировании с моделью не учитываются взаимодействия между факторами, то это увеличивает число степеней свободы и улучшает оценки стандартной ошибки.

Главная цель процедуры ANOVA состоит в создании стандартной таблицы ANOVA, где указывается окончательная F-статистика и ее критическое значение. Кроме того, при вызове процедуры ANOVA из программы PLUS при завершении процедуры возвращается стандартная ошибка.

9.8. Технология проведение дисперсионного анализа в системе GPSS World

Процедура ANOVA выполняет дисперсионный анализ, определяет значимость факторов по критерию Фишера (F-критерию) и выполняет расчет доверительных интервалов исследуемых величин. Если значение F-критерия не превышает критическое значение, то наблюдаемый фактор незначимый, выборочные последовательности принадлежат одной генеральной совокупности.

При выполнении дисперсионного анализа для каждого уровня обработки исследуемого фактора необходимо выполнить несколько прогонов модели. Количество прогонов устанавливается в зависимости от задаваемой точности (см. параграф 3.5). Для уменьшения корреляции результатов прогонов рекомендуется, чтобы каждый из прогонов использовал свою последовательность псевдослучайных чисел. После каждого прогона модели необходимо сохранить результаты моделирования, обнулить собранную за предыдущий прогон статистику и удалить транзакты из модели. Для выполнения этих действий необходимо воспользоваться следующими операторами GPSS World:

RMULT – устанавливает начальные значения множителей для генераторов случайных чисел (см. параграф 4.27);

CLEAR – осуществляет обнуление собранной статистики за предыдущий прогон модели и удаляет транзакты из модели, но не устанавливает множители генераторов случайных чисел в начальные значения. Операнд A = OFF запрещает обнуление матрицы результатов (см. параграф 4.27);

START – используется для инициирования процесса моделирования. Если задать второй операнд NP, то статистика не будет выводиться на экран (см. параграф 4.27);

MSAVEVALUE – записывает полученные результаты моделирования в специальную матрицу результатов (см. параграф 4.17).

Все перечисленные операторы должны записываться в специальный текстовый файл, который размещается в одном каталоге с файлом модели.

Перед началом моделирования необходимо определить матрицу результатов. Например:

RES MATRIX *R,C*

где **RES** – имя матрицы, в которой накапливаются результаты прогонов; *R* – количество строк матрицы (Treatment levels), должно соответствовать количеству уровней обработки; *C* – количество столбцов матрицы (Replicates), должно соответствовать числу прогонов модели для каждого уровня фактора. Числа *R* и *C* должны быть целыми.

Затем нужно создать командный текстовый файл, например, с такой последовательностью команд и операторов:

```
FACTOR  EQU  N1                          первое значение уровня фактора
RMULT Z11
START 1,NP
MSAVEVALUE RES,1,1,X$Rez
CLEAR OFF
...                                      C таких блоков команд
RMULT Z1C
START 1,NP
MSAVEVALUE RES,1,C,X$Rez
CLEAR OFF
...
FACTOR  EQU  Ni                          i-е значение уровня фактора
RMULT Z i1
START 1,NP
MSAVEVALUE RES,i,1,X$Rez
CLEAR OFF
...                                      C таких блоков команд
RMULT ZiC
START 1,NP
MSAVEVALUE  RES,i,C,X$Rez
CLEAR OFF
...
FACTOR  EQU  NR                          последнее значение уровня фактора
RMULT ZR1
START 1,NP
MSAVEVALUE  RES,R,1,X$Rez
CLEAR OFF
...                                      C таких блоков команд
RMULT ZRC
START 1,NP
MSAVEVALUE  RES,R,C,X$Rez
CLEAR OFF
```

Целые числа Z_{11} ... Z_{RC} задают начальные значения множителей для генераторов случайных чисел. В сохраняемой величине X$Rez в каждом прогоне формируется значение критерия эффективности системы (результат моделирования). В процессе анализа исследуется степень влияния фактора на значение этого критерия эффективности. Индекс *i* – номер текущего уровня обработки (изменяется от 1 до *R)*. Сохраним этот файл с именем test.txt.

После этого нужно открыть файл модели (*.gps) в системе GPSS World и создать имитацию (оттранслированную модель) с помощью пункта меню **Command/Create Simulation,** а затем выбрать пункт меню **Command/Custom** (рис. 9.4). В результате откроется диалоговое окно **Simulation Command** (рис. 9.5). В поле для ввода команд нужно ввести текст

include "test.txt"

и нажать кнопку **ОК.**

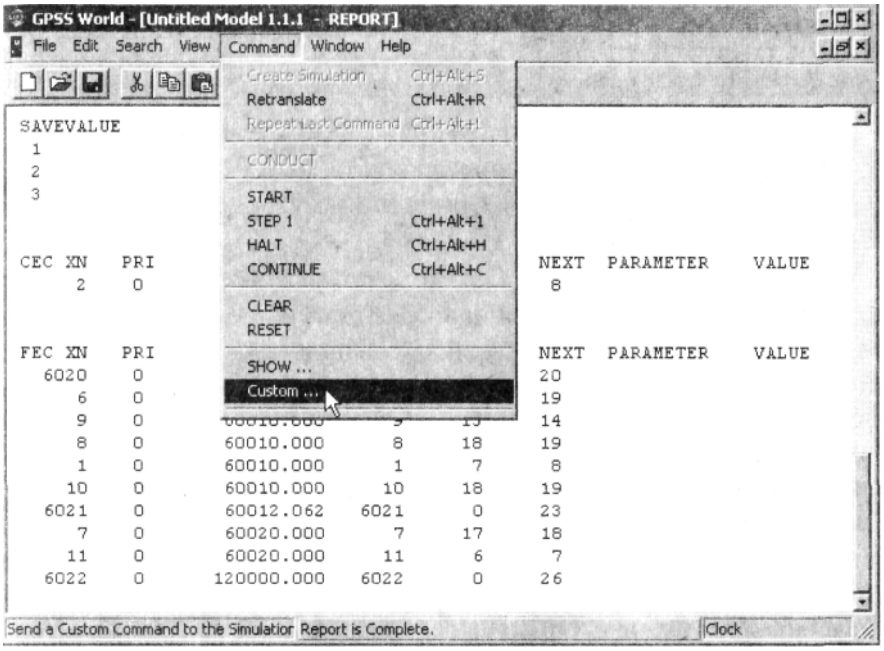


Рис. 9.4

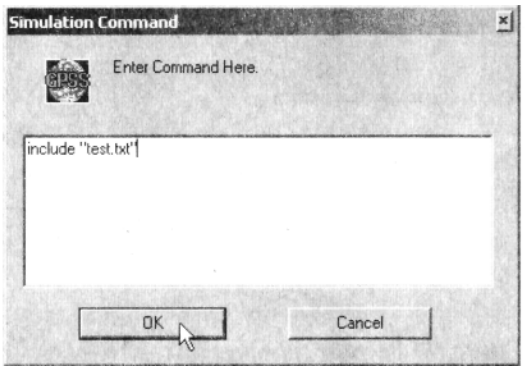


Рис. 9.5

Если по какой-либо причине файл test.txt размещен не в одном каталоге с файлом модели, то необходимо указывать полный путь к этому файлу.

Процесс моделирования может занять продолжительное время в зависимости от количества прогонов модели. После его завершения в нижней строке состояния (рис. 9.6) главного окна появится сообщение «The Simulation has ended».

Рис. 9.6

Затем необходимо с помощью пункта меню Command/SHOW открыть диалоговое окно Show Command и в строке ввода написать

ANOVA(RES,A,B)

где **A** равно **2**, если результаты повторяющихся прогонов модели для каждого уровня фактора со своей последовательностью псевдослучайных чисел записываются в одну строку (как принято по умолчанию в ранних версиях GPSS World), и **1**, если в один столбец (матрица результатов транспонируется). Если **A = 0,** то повторяющиеся прогоны не производятся.

Параметр **B,** согласно документации, – это максимальное количество анализируемых взаимодействий между факторами. Может принимать значения 1, 2 или 3. Для выполнения процедуры **ANOVA** нужно

нажать кнопку **ОК**. После этого в окне журнала сессии (рис. 9.7) можно просмотреть результаты дисперсионного анализа.

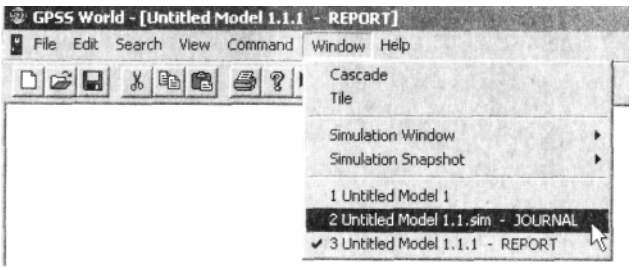


Рис. 9.7

Данный формат процедуры **ANOVA** можно использовать только для системы GPSS World, начиная с версии 4.2.1 включительно. Для версии 4.1.1 формат команды следующий:

ANOVA(RES)

Пример 9.3

Пусть имеем модель некоторой системы. При этом величина **TIMEA** является входной переменной, **A** в сохраняемой величине **X3** формируется значение выходной величины (результат моделирования).

```
          TIMEA EQU 20
          TIMEB EQU 20
          INITIAL   X1,0
          INITIAL   X2,0
          INITIAL   X3,0

          GENERATE  ,,,4
          TRANSFER        ,PUNKTB

          GENERATE  ,,,4
PUNKTA    TEST GE         X1,1,PAP
          SAVEVALUE 1-,1
          ADVANCE         20
          ADVANCE         20
          TRANSFER        ,GOB
PAP        SAVEVALUE      3+,1
GOB        ADVANCE        30
PUNKTB    TEST GE         X2,1,PBP
          SAVEVALUE 2-,1
          ADVANCE         20
          ADVANCE         20
          TRANSFER        ,GOC
PBP       SAVEVALUE 3+,1
GOC       ADVANCE         30
          ADVANCE         20
          TRANSFER        ,PUNKTA

          GENERATE  TIMEA,3
          SAVEVALUE 1+,1
          TERMINATE 0

          GENERATE  TIMEB,5
          SAVEVALUE 2+,1
          TERMINATE 0

          GENERATE  60000
          TERMINATE 1
          START     1
```

Целью моделирования является оценка влияния фактора TIMEA на значение выходной переменной X3 модели. Сформируем командный файл (a.txt):

```
RES MATRIX,9,10

TIMEA EQU 20
RMULT 401
START 1,NP
MSAVEVALUE RES,1,1,X3
CLEAR OFF
RMULT 411
START 1,NP
MSAVEVALUE RES,1,2,X3
CLEAR OFF
RMULT 421
START 1,NP
MSAVEVALUE RES,1,3,X3
CLEAR OFF
RMULT 431
START 1,NP
MSAVEVALUE RES,1,4,X3
CLEAR OFF
RMULT 441
START 1,NP
MSAVEVALUE RES,1,5,X3
CLEAR OFF
RMULT 451
START 1,NP
MSAVEVALUE RES,1,6,X3
CLEAR OFF
RMULT 461
START 1,NP
MSAVEVALUE RES,1,7,X3
CLEAR OFF
RMULT 471
START 1,NP
MSAVEVALUE RES,1,8,X3
CLEAR OFF
RMULT 481
START 1,NP
MSAVEVALUE RES,1,9,X3
CLEAR OFF
RMULT 491
START 1,NP
MSAVEVALUE
MSAVEVALUE RES,1,10,X3
CLEAR OFF

TIMEA EQU 40
RMULT 401
START 1,NP
MSAVEVALUE RES,2,1,X3
CLEAR OFF
RMULT 411
START 1,NP
MSAVEVALUE RES,2,2,X3
CLEAR OFF
RMULT 421
START 1,NP
MSAVEVALUE RES,2,3,X3
CLEAR OFF
RMULT 431
START 1,NP
MSAVEVALUE RES,2,4,X3
CLEAR OFF
RMULT 441
START 1,NP
MSAVEVALUE RES,2,5,X3
CLEAR OFF
RMULT 451
START 1,NP
MSAVEVALUE RES,2,6,X3
CLEAR OFF
RMULT 461
START 1,NP
MSAVEVALUE RES,2,7,X3
CLEAR OFF
RMULT 471
START 1,NP
MSAVEVALUE RES,2,8,X3
CLEAR OFF
RMULT 481
START 1,NP
MSAVEVALUE RES,2,9,X3
CLEAR OFF
RMULT 491
START 1,NP
MSAVEVALUE RES,2,10,X3
CLEAR OFF
```

Компилируем модель и запускаем моделирование, после чего выбираем пункт меню **Command/Custom** и вводим в диалоговое окно команду

include "a.txt"

После этого выдаем команду

SHOW ANOVA(RES,2,1)

Для ранних версий – **SHOW ANOVA(RES).**
В окне журнала сессии получим результаты работы процедуры ANOVA.

ANOVA

Source of Variance	Sum of Squares	Degrees of Freedom	Mean Square	F
Treatments	2416489530.089	8	302061191.261	33.023
Error	740907977.300	81	9147012.065	
Total	3157397507.389	89		

Treatment	Count	Mean	Std. Dev.	Minimum	Maximum	95% Conf.
1	10	36.90	10.43	19.00	47.00	7.46
2	10	16492.60	9073.20	3008.00	29995.00	6490.13

33.0229357

9.9. Особенности планирования экспериментов

Опишем последовательность действий, выполняемых при проведении экспериментов.
1. Определение откликов (выходных переменных) системы.
2. Определение факторов, которые предположительно влияют на отклик системы.

Большинство систем работает по ***принципу Парето.*** Этот принцип гласит, что с точки зрения характеристик системы существенными являются лишь некоторые из множества факторов. В большинстве систем 20% факторов определяют 80% свойств системы.

3. Определение уровней факторов. Минимальное количество уровней два – нижняя и верхняя границы значений факторов. При использовании этого числа уровней можно определить только линейные эффекты. Для учета квадратичных эффектов необходимо использовать три уровня, для

кубических эффектов – четыре и т.д. Анализ значительно упрощается, если брать только равноотстоящие друг от друга значения уровней. В этом случае имеем так называемое ***ортогональное*** разбиение уровней.

Для множественных экспериментов с числом факторов больше одного дисперсионный анализ требует, чтобы для заключительного анализа использовался ортогональный эксперимент. Это означает, что оценки в пределах анализа должны быть некоррелированные. На практике использование одних и тех же чисел при выполнении экспериментов в пределах каждой комбинации уровней обработки гарантирует ортогональность.

Планирование эксперимента обычно применяется для определения важных – существенно влияющих на отклик – факторов (отсеивающий эксперимент). Так как с ростом числа факторов число комбинаций быстро растет, то необходимо выделить наиболее важные факторы, т.е. провести предварительное отсеивание. Для этого используются планы 2^{k-p}, считая взаимодействия более высокого порядка равными нулю.

При использовании планирования экспериментов для поиска экстремальных значений отклика применяются планы типа 2^{k-p} и дополнительные комбинации, позволяющие оценить отклик как функцию (полином первого или второго порядка) от независимых переменных. В этом случае все k факторов должны быть количественными.

Планы можно применять последовательно, т.е. сначала получить наблюдения для одних комбинаций уровней факторов, затем для других и после анализа этих наблюдений решить, для какой комбинации (старой или новой) провести дополнительные наблюдения. Эти новые наблюдения опять анализируются (обычно вместе с предыдущими) для принятия решения о дальнейших наблюдениях и так далее. В планах типа 2^{k-p} можно сначала выполнить малую часть эксперимента, проанализировать наблюдения, **А** затем, если этот анализ показывает, что данная часть эксперимента слишком мала для оценки всех возможных эффектов, расширить эксперимент таким образом, чтобы он позволил оценить все эти эффекты.

Если целью анализа поверхности отклика является поиск оптимальной комбинации уровней k количественных факторов, то в этом случае выполняют следующие этапы:

1. Планирование эксперимента, т.е. выбор комбинации уровней факторов. В этом случае используются полные типа 2^k , дробные типа 2^{k-p} планы экспериментов и некоторые специальные планы.
2. Проведение экспериментов и построение по их результатам уравнения регрессии поверхности отклика.
3. Подъем по поверхности отклика к вершине. Для нахождения направления увеличения отклика обычно используется метод скорейшего подъема (крутого восхождения). В направлении, в котором ожидается увеличение отклика, этапы 1, 2 повторяются до тех пор, пока не будет достигнута область максимума.
4. Проведение канонического анализа в области максимума функции поверхности отклика. При этом определяется: имеется ли один максимум, несколько максимумов, седловая точка или гребень.

9.10. Нахождение экстремальных значений на поверхности отклика

Запишем уравнение поверхности отклика в следующем виде

$$y = F(x_1,...x_k),$$

где $x_1,...,x_k$ – независимые переменные, k – число факторов. Во многих случаях цель имитационного моделирования заключается в поиске таких величин или уровней независимых переменных, при которых отклик достигает экстремального значения. Для определения направления движения к экстремальной точке в случае использования количественных, непрерывных и измеряемых величин применяют ряд небольших, полных и неполных факторных экспериментов. Так как поверхность отклика неизвестна, то ее аппроксимируют какой-то гладкой функцией, в качестве которой обычно используют полином первого порядка

$$y = a_0 + a_1x_1 + a_2x_2 + ... + a_kx_k$$

или второго порядка

$$y = a_0 + a_1x_1 + a_2x_2 + ... + a_kx_k + a_{12}x_1x_2 + a_{13}x_1x_3 + ... + a_{k-1,k}x_{k-1}x_k + \\ + a_{11}x_1^2 + a_{22}x_2^2 + ... + a_{kk}x_k^2.$$

Параметры $a_0,a_1,...,a_k,...$ оценивают по результатам факторного эксперимента.

Для поиска экстремума наиболее часто используют метод скорейшего подъема. Он основан на линейной аппроксимации поверхности отклика в окрестности рассматриваемой точки P с помощью факторного эксперимента.

По построенной линейной функции определяется направление скорейшего подъема $\vec{Q}$ к точке оптимума (рис. 9.8). В направлении $\vec{Q}$ делается небольшой шаг, после чего описанная процедура повторяется снова. Метод не позволяет определить длину шага, однако, указывает направление движения.

Предположим, что исследователь провел в точке P эксперимент с 2^k комбинациями плюс два наблюдения в центре. Эксперимент позволяет определить коэффициенты a_0, a_1, a_2 (для случая $k = 2$), которые определяют наклон плоскости аппроксимации. Направление скорейшего подъема показывает относительные величины изменения факторов, обеспечивающих максимальное увеличение отклика. Поднявшись по этому направлению до некоторой точки P_1, необходимо повторить всю процедуру. Такой итерационный процесс позволяет достигать все лучших и лучших значений отклика. Однако вблизи точки экстремума эта процедура неэффективна, так как коэффициенты a_1 и a_2, определяющие наклон аппроксимирующей плоскости, становятся небольшими и точность их оценивания низка. Это означает, что вблизи экстремальной точки линейная аппроксимация поверхности отклика является недостаточной и надо переходить к аппроксимации полиномом более высокой степени.

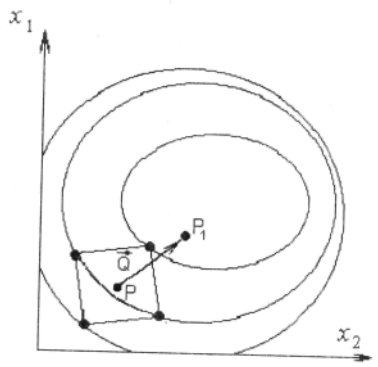


Рис. 9.8

Для рассматриваемого примера эксперимент с 2^k комбинациями достаточен для оценивания коэффициентов a_0,a_1,a_2. Однако два добавочных наблюдения в геометрическом центре P позволяют не только уточнить уравнение регрессии, но и получить несколько дополнительных степеней свободы для проверки статистической значимости оценок параметров регрессии. То же самое можно сделать с помощью повторного эксперимента. Вблизи экстремума поверхности желательно аппроксимировать поверхности отклика, по меньшей мере, полиномом второго порядка. Для этого используют приближение:

$$y = a_0 + a_1x_1 + a_2x_2 + a_{11}x_1^2 + a_{22}x_2^2 + a_{12}x_1x_2$$

Для оценки коэффициентов регрессии этой модели необходимо измерить каждый фактор, по крайней мере, на трех уровнях, то есть использовать 3^k-факторный эксперимент. Однако этот эксперимент дает довольно низкую точность оценок коэффициентов регрессии. Поэтому специально для квадратичных полиномов используют другие способы построения эксперимента. Из них наиболее полезными являются ***центральный композиционный*** или ***рототабельный*** планы. Они получаются путем добавления дополнительных точек к данным, полученным из 2^k факторных экспериментов. Для рототабель-ного построения стандартная ошибка одинакова для равноудаленных от центра области точек. Такие построения разработаны для любого числа факторов и представляют собой правильные геометрические фигуры с центральными точками.

9.11. Организация экспериментов в GPSS WorId

Система GPSS World обеспечивает два вида автоматически проводимых экспериментов: разработанные для пользователя и разработанные пользователем. Разработанные для пользователя и встроенные в GPSS World автоматические генераторы отсеивающих и оптимизирующих экспериментов не предполагают повторных прогонов, т.е. эти эксперименты проводятся с моделями, работающими в стационарном режиме. Для таких моделей оценки точности результатов моделирования могут быть получены за один достаточно длительный прогон модели (см. параграф 9.4). Данные переходного периода в этих экспериментах могут не учитываться. Эти модели не предусматривают повторений экспериментов для каждого уровня сочетания факторов.

Отсеивающие и оптимизирующие эксперименты также не могут быть использованы в тех случаях, когда в процессе экспериментирования необходимо изменять количество устройств в МКУ, которое задается оператором **STORAGE.**

Пользовательские эксперименты могут проводиться с моделями, работающими в переходном режиме, и позволяют организовать повторение экспериментов. В этом случае необходимо заранее до планирования экспериментов для каждого сочетания уровней факторов определить количество повторений экспериментов (см. параграф 9.4). Наличие встроенного языка программирования PLUS позволяет разрабатывать пользовательские эксперименты любой сложности, включая и переопределение емкости МКУ.

Отсеивающий эксперимент в GPSS World. Этот эксперимент используется для определения факторов, существенно влияющих на систему. Полученная информация важна для проведения дальнейших экспериментов с целью создания более эффективных оптимизирующих экспериментов. Результаты отсеивающего эксперимента показывают, какие факторы менее эффективны и должны в дальнейшем получить меньший приоритет при рассмотрении. Для получения достоверных результатов необходимо правильно выбирать значения факторов, с которыми будет проводиться эксперимент. В случаях, когда область допустимых значений фактора велика, может возникнуть необходимость проведения нескольких экспериментов для разных значений факторов из этой области. В общем случае значения каждого фактора желательно выбирать равноудаленно от границ области и друг от друга.

Для того, чтобы воспользоваться встроенным генератором отсеивающих экспериментов GPSS World, необходимо выполнить пункт меню **Edit/Insert Experiment/Screening,** в результате чего появится диалоговое окно **Screening Experiment Generator.** Отсеивающий эксперимент проводится только для двух значений каждого уровня факторов **Value 1 и Value 2**, так как в его основе лежит линейная регрессионная модель.

Для организации эксперимента необходимо иметь отлаженную имитационную модель, которая должна быть проверена при всевозможных значениях входных переменных. Необходимо гарантировать, что модель будет правильно работать при любой комбинации уровней факторов.

В результате заполнения полей диалогового окна в эту модель автоматически будет добавлен код на языке PLUS. По выбору пользователя для запуска эксперимента может быть назначена функциональная клавиша (обычно **F11,** но ее можно переопределить). В этом случае после трансляции модели (команда пункта меню **Command/Create Simulation или Ctrl+Alt+S)** для запуска отсеивающего эксперимента достаточно будет нажать функциональную клавишу **F11.**

Пример 9.4

Отсеивающий эксперимент для модели работы компьютерной сети.

Локальная сеть (ЛС) имеет кольцевую топологию. В ней используется 15 рабочих станций и один сервер, который также подключен к глобальной сети.

Пользователи рабочих станций посылают на сервер запросы для выполнения локальных заданий (обращения к базе данных, находящейся на сервере, составляют 15% запросов) и для обращения в глобальную сеть (поиск информации составляет 85% запросов).

Опрос рабочих станций в ЛС происходит с помощью маркера по кольцевому алгоритму каждые 1,5 с. Время переключения маркера с одной рабочей станции на другую составляет 0,1 с. Если у рабочей станции есть сообщение, оно передается на сервер.

Поток запросов на обслуживание от всех рабочих станций является пуассоновским с интенсивностью 1 запрос за 30 с. Длина возникающих сообщений (кбайт) имеет гамма-распределение с параметрами a =

88, P = 0,4. Каждое сообщение в ЛС разбивается на пакеты размером 1 Кбайт. Каждый пакет передается на сервер в течение 10 мс. Для сборки пакетов на сервере затрачивается по 1 мс на пакет.

Для обработки запросов по обращению к базе данных сервера требуется 50 ± 40 мс. Объем данных, которые нужно будет передать пользователю с сервера, равномерно распределен в интервале от 0,01 до 1 Мбайт.

Связь с глобальной сетью осуществляется по полудуплексному каналу (прием и передача не могут происходить в один и тот же момент времени) со скоростью 3,6 кбайт/с. Время поиска информации в глобальной сети в секундах имеет гамма-распределение с параметрами $\alpha = 10$, $\beta = 0{,}2$. Объем передаваемых данных от сервера на рабочие станции распределен по равномерному закону и составляет 100±50кбайт.

Канал связи занят во время передачи данных от рабочих станций к серверу и в обратном направлении. Одновременно могут передаваться данные только по одному запросу. Если в момент, когда канал занят, возникает запрос на использование канала, он становится в общую очередь запросов. После освобождения канала сначала проверяется содержимое очереди запросов. Если она пустая, то продолжается опрос рабочих станций, начиная со следующей после освободившей канал станции. Данные могут передаваться по каналу связи, даже когда процессор сервера обрабатывает запрос. Запросы обрабатываются центральным процессором (ЦП) сервера по правилу FIFO. В то время, как ЦП выполняет обращение к базе данных или глобальной сети, он освобождает канал связи. Результаты обработки запросов передаются на рабочую станцию сразу же, как только освободится канал.

Необходимо промоделировать работу сети на протяжении 24 часов, оценить среднее время обработки запросов от рабочих станций и определить факторы, влияющие на это время.

Программа:

```
* Определение номера рабочей станции, которая сгенерировала запрос
wsnum  function        rn1,d15
.066,1/.133,2/.199,3/.266,4/.332,5/.399,6/.465,7/.532,8/.598,9
.665,10/.731,11/.798,12/.864,13/.931,14/1,15
* Определение порядка опроса:
qorder  function       p1,d15
1,2/2,3/3,4/4,5/5,6/6,7/7,8/8,9/9,10/10,11/11,12/12,13/13,14/14,15/15,1

 * Предварительное задание значений факторов для эксперимента
 q_time     equ 100                 ; Время опроса
 t_time     equ 10                  ; Время передачи одного пакета
 q_int      equ 30000               ; Интенсивность запроса на рабочих станциях
 * Время передачи 1 Кбайта из глобальной сети при скорости 3,6 Кбайт/с
 gn_time    equ 278                 ; 278 мс

            generate    ,,,1        ; Генерация опросного маркера
            assign      1,1         ; Номер опрашиваемой рабочей станции

 next  seize channel                ; Опрос рабочей станции
            advance     q_time
            test ne     ch*1,0,ifempty      ; Есть ли запрос на этой рабочей станции?

            unlink      p1,send_f,1         ; Передать запрос, если он есть
 ifempty  release  channel          ; Занять канал для опроса
            assign      1,fn$qorder ; Опрос следующей рабочей станции
            transfer    ,next
            generate    (exponential(3,0,q_int)),,,1    ; Запрос с рабочей станции
            assign      1,fn$wsnum  ; № рабочей станции, сгенерировавшей запрос
            assign      2,(gamma(4,0,.4,88)\1+1)    ; Размер запроса, Кбайт
            queue       network     ; Фиксация появления запроса
            link        p1,fifo     ; Ждать опроса станции
 send_f     seize       channel     ; Занять канал
            advance     (t_time#p2) ; Передать запрос
            release     channel     ; Освободить канал
            seize       cpu         ; Начать обработку запроса на сервере
            advance     p2          ; Время сборки запроса из пакетов
            transfer    .85,bdq,gnq ; Запрос к БД или к сети?
 bdq        advance     50,40       ; Время обработки запроса к БД
            assign      2,(duniform(5,10,1024))
            transfer    ,send_b
```

* Обработка запроса к глобальной сети

* Поиск в глобальной сети. Время поиска зависит от скорости соединения с сетью

```
gnq       advance   (gamma(6,0,gn_time/1390,10))
          assign    2,(duniform(7,50,150))
          advance   (gn_time#p2)        ; Время получения данных из сети
          transfer  ,send_b
* Передача результатов обработки запроса на рабочую станцию
send_b    release   cpu
          priority  2                   ; Повысить приоритет для ответа
          seize     channel
          advance   (t_time#p2)
          release   channel
          depart    network             ; Фиксация времени пребывания в сети
          terminate

          generate  86400000            ; Таймер (24 часа = 86400 с = 86400000 мс)
          terminate 1
```

Создадим отсеивающий эксперимент, в ходе которого оценим влияние на критерий эффективности модели следующих факторов:

- время опроса (фактор *A);*
- время передачи одного пакета по каналу связи (фактор *B);*
- интервал времени между возникновениями запросов на рабочих станциях (фактор C);
- время доступа к глобальной сети (фактор *D).*

Для изменения значений этих факторов воспользуемся операторами EQU, которые определены в начале программы.

Для создания отсеивающего эксперимента выберем команду в меню **Edit/Insert Experiment/Screening...** и заполним поля диалогового окна (рис. 9.9).

Назначение элементов диалогового окна:

Experiment Name (название эксперимента). В данном случае «ScreenNetwork».

'Run Procedure' Name (название процедуры прогона). В данном случае «DoTheRun».

Factors (факторы), которых должно быть не более шести. В соответствующие поля необходимо ввести имена факторов, заданных в модели оператором EQU, и два значения уровней факторов **(Value 1** и **Value** 2).

Fraction (доля) определяет, какая часть от полного факторного эксперимента будет выполнена, т.е. план 2^{k-p}. В этом случае прогоны выполняются для каждого варианта значений уровней факторов, заданных в полях **Value 1** и **Value 2.**

Result, Expression (выражение для выходной переменной, для которой проводится эксперимент). В это поле могут быть введены названия переменных пользователя, СЧА или выражения.

Generate Run Procedure, Load F11 with CONDUCT command (флажки «Генерация процедуры прогона» и «Загрузка команды CONDUCT в клавишу F11») необходимо установить для автоматизации и упрощения запуска эксперимента.

Alias Groups (группы смешивания). Эта кнопка открывает диалоговое окно **Defining Relation** (определение отношений, рис. 9.10).

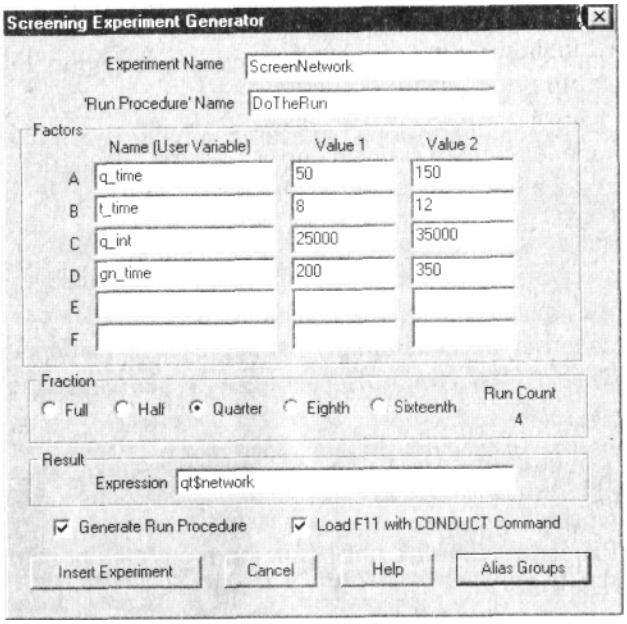


Рис. 9.9

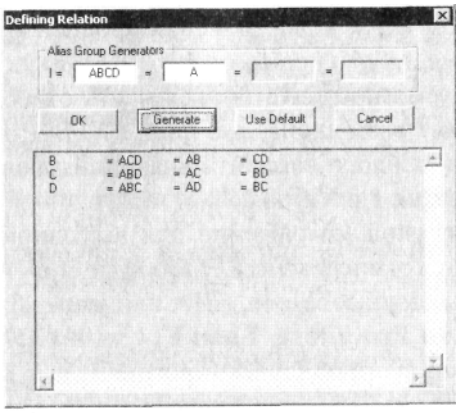


Рис. 9.10

Часто время одного прогона модели настолько велико, что делать все 2^k экспериментов с g повторениями не представляется возможным (особенно, когда факторов 4...6, а g = 100 или больше). В таком случае генератор отсеивающего эксперимента позволяет задать меньшее количество экспериментов (**Fraction**), т.е. 2^{k-p}. Этого можно добиться, благодаря смешиванию некоторых эффектов (Alias Groups). Следует, однако, учитывать, что при этом будет получена только часть информации. Кроме того, при превышении определенного предела числа прогонов уже нельзя будет определить эффекты отдельных факторов. В таких случаях можно сделать предположение о влиянии определенного фактора по значению эффекта, полученного для групп, в которые он входит и для которых это значение удалось рассчитать даже при меньшем количестве прогонов. Однако следует учитывать, что влияние факторов из одной группы может быть взаимно противоположными. Может создаться впечатление, что эти факторы этой группы не оказывают на модель существенного влияния, в то время как в действительности каждый в отдельности оказывает такое влияние. Рекомендуется так группировать факторы, чтобы в каждой группе оказывался только один существенный фактор.

В приведенном примере в качестве критерия эффективности выбрано среднее время (в мс) обработки запроса от момента появления его на рабочей станции до момента окончания приема результатов запроса на этой станции. Для получения этого времени используется регистратор очереди network. В качестве критерия выбран СЧА qt$network – среднее время нахождения транзактов в системе, которое будет стремиться к истинному среднему только тогда, когда система будет работать в стационарном режиме. Если же запросы будут поступать с частотой, превышающей частоту их обработки, то длина очереди на обработку будет стремиться к бесконечности.

После заполнения диалога в модель будут вставлены такие процедуры:

ScreenNetwork – собственно отсеивающий эксперимент;

ScreenNetwork_GetResult – процедура выполнения прогонов, служит для фиксации результатов каждого прогона и перехода к следующему прогону;

DoTheRun – процедура прогона, которая служит для установки начальных значений генераторов случайных чисел и запуска каждого нового прогона.

Генератор отсеивающего эксперимента GPSS World до версии 4.3.5 не всегда прописывает название процедуры **DoTheRun** на языке PLUS. Необходимо проверить добавляемый код и в случае, если эта процедура оказалась без названия, ввести его вручную.

Отсеивающий эксперимент в GPSS World предполагает, что модель работает в стационарном режиме и оценки 1 выходной величины или критерия эффективности получаются по одному достаточно длинному прогону, поэтому в процедуре для отсеивающего эксперимента нет повторений для каждой комбинации уровней факторов.

Итак, для того, чтобы результаты эксперимента были достоверными, необходимо экспериментировать с моделью, которая находится в стационарном режиме. Поэтому сначала необходимо исследовать модель на стационарность. Для этого воспользуемся графиком изменения

величины qt$network – среднего времени обработки запроса (рис. 9.11) в модельном времени (выполнив пункт меню **Window/Simulation Window/Plot Window...).**

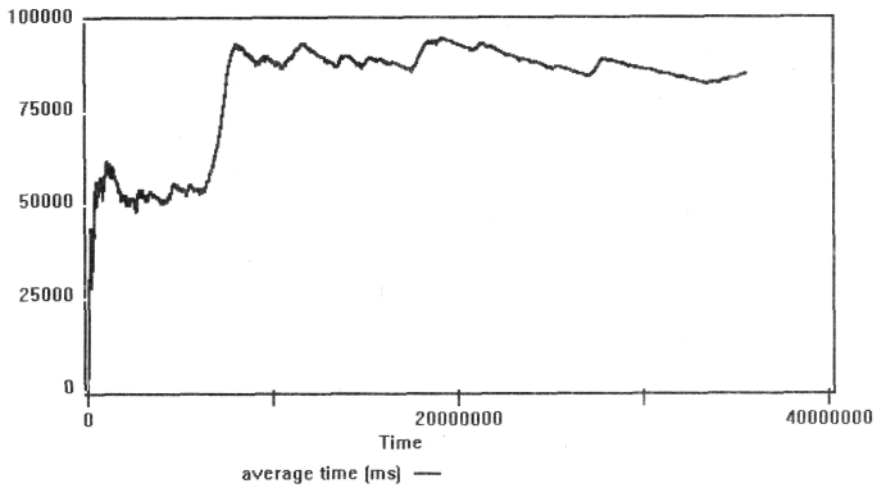


Рис. 9.11

Из графика видно, что время переходного процесса равно примерно 20мин (1200000 мс) и оказалось, как минимум, в 72 раза меньше времени одного прогона системы. Окончательную длительность прогона для стационарного процесса надо определять в зависимости от требуемой точности получения оценок для выходной переменной. Это можно сделать для эргодических или регенерирующих процессов, как описано в параграф 9.4.

Для того, чтобы статистические данные, собираемые во время переходного процесса, не давали смещения для выходной переменной, их необходимо удалять с помощью оператора **RESET.** Автоматически сгенерированный отсеивающий эксперимент в конце процедуры **DoTheRun** предполагает, что переходный период в модели закончится после прохождения 100 транзактов (Get past the Startup Period).

```
/* SET UP YOUR OWN RUN CONDITIONS. */
   DoCommand("START 100,NP");   /* Get past the Startup Period. */
   DoCommand("RESET");          /* Begin the Measurement Period. */
   DoCommand("START 1000,NP"); /* Run the Simulation. */
```

После этого выполняется оператор **RESET** и начинается период измерений (Begin the Measurement Period). Полный прогон модели (Run the Simulation) заканчивается после прохождения через нее 1000 транзактов.

В данном примере моделирование заканчивается через 24 часа (86400000 мс). Учитывая то, что один прогон модели занимает значительное время и для 4-х факторов на двух уровнях надо выполнить для полного факторного эксперимента 2^4 прогонов, то сократим длительность прогона до 10 длительностей переходного периода. Для этого установим таймер в модели на 1200000 мс.

GENERATE 1200000 С учетом этого внесем изменения в процедуру **DoTheRun.**

```
/* SET UP YOUR OWN RUN CONDITIONS. */
   DoCommand("START 1,NP");   /* Get past the Startup Period. */
   DoCommand("RESET");        /* Begin the Measurement Period. */
   DoCommand("START 10,NP"); /* Run the Simulation. */
```

После трансляции модели (пункт меню **Command/Create Simulation** или с помощью комбинации клавиш **Ctrl+Alt+S)** выполним полный **(Full)** отсеивающий эксперимент, нажав функциональную клавишу F11. В результате будут получены такие данные.

```
02/14/03 17:31:09

Alias  Effect        Sum of      Degrees of    F - for Only   Critical Value
Group             Squares Freedom Main Effects   of F (p=.05)

  A     0.125          0.063        1              0.000         4.84
  B     5.625        126.563        1              0.155         4.84
```

```
AB    -5.375
 C   -47.625      9072.563       1            11.079       4.84
AC    -0.625
BC    -4.625
ABC    6.375
 D    46.375      8602.563       1            10.505       4.84
AD    -0.625
BD     7.875
ABD   -3.125
CD   -44.875
ACD    1.125
BCD   -6.875
ABCD   5.125

02/14/03 17:31:09       Error                    9008.188          11
02/14/03 17:31:09       Total                   26809.938          15
02/14/03 17:31:09       Grand Mean       25.563

02/14/03 17:31:09   Experiment ended.
```

Как видно из результатов, только два эффекта – C и D – значимы, причем рост интенсивности запросов на рабочих станциях отрицательно сказывается на среднем времени пребывания в сети, **A** увеличение скорости передачи пакетов уменьшает это время.

Оптимизирующий эксперимент. Этот эксперимент предназначен для построения уравнения поверхности отклика для заданных факторов модели и поиска численного значения оптимума (сочетания таких значений факторов, при котором заданная функция принимает экстремальное значение – максимум или минимум). Максимально возможное количество изменяемых факторов – пять. Эксперимент проводится с уже созданной моделью. Факторы, которые будут изменяться, должны быть в модели представлены в виде констант, которые определяются с помощью оператора EQU или переменных. В ходе эксперимента производится многократный прогон модели, фиксация результатов и использование их для получения поверхности отклика и поиска оптимума.

Оптимизирующий эксперимент добавляется в созданную и отлаженную модель. Для этого необходимо выбрать пункт меню **Edit/Insert Experiment/Optimizing,** после чего откроется диалоговое окно **Optimizing Experiment Generator** (рис. 9.12).

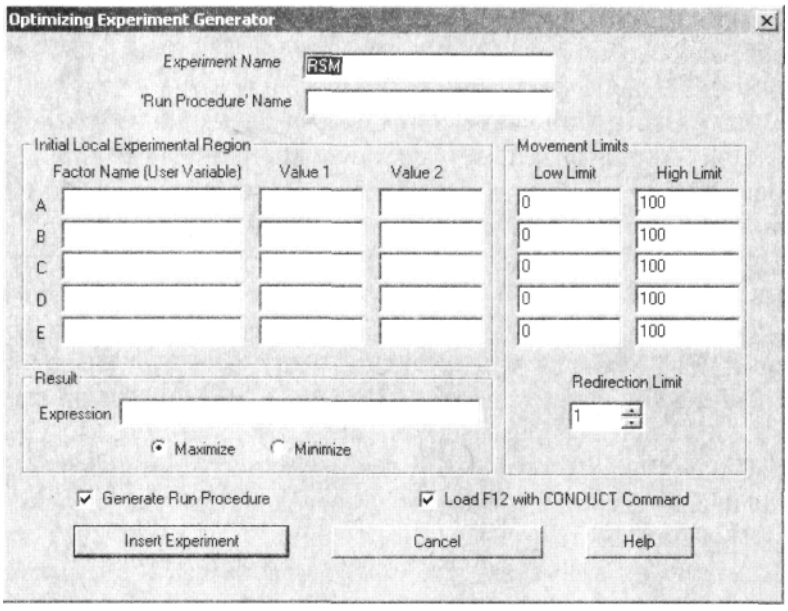


Рис. 9.12

В нем указываются необходимые для выполнения эксперимента параметры. В полях **Experiment Name и Run Procedure Name** задаются соответственно название эксперимента и Run-процедуры прогонов модели. По умолчанию название эксперимента – RSM (от **R**esponse **S**urface **M**odel). Это имя будет добавлено ко всем названиям процедур и переменных, которые автоматически генерируются для этого эксперимента. Пользователю доступны для изменений все процедуры, относящиеся к эксперименту, кроме одной RSM_FitSurfaceToData (имя процедуры фиксировано и не изменится при

задании другого названия эксперимента). В качестве названия Run-процедуры рекомендуется использовать DoTheRun, хотя можно использовать и любое другое имя.

В полях **Factor Name A, B, C, D, E** указываются изменяемые величины (факторы) и задаются значения их верхних и нижних уровней **Value 1 и Value 2.**

Поля **Low Limit, High Limit для Movement Limits** задают максимальные границы перемещения (могут иногда помочь при поиске оптимума, но в большинстве случаев можно оставлять значения по умолчанию 0 и 100). Значение поля **Redirection Limit** ограничивает количество изменений направлений при поиске оптимума. В некоторых случаях достаточно 1, но обычно нужно выбирать 2 или 3.

В поле **Expression** указывается целевая функция – выражение на языке PLUS, заключенное в круглые скобки, или переменная модели, в которой задана эта функция. С помощью переключателя **Maximize-Minimize** выбирается направление оптимизации.

При выборе **Generate Run Procedure** создается процедура прогонов модели, в противном случае придется ее написать. Выбор **Load F12...** позволяет запускать эксперимент с помощью нажатия одной клавиши F12 вместо того, чтобы выбирать в меню **Command/Conduct** и в нем вводить CONDUCT RSM() или другое выбранное название эксперимента.

Для запуска автоматической генерации эксперимента нужно нажать на кнопку **Insert Experiment,** но сначала необходимо задать перечисленные параметры.

Покажем на примерах, как можно использовать оптимизирующий эксперимент.

Пример 9.5

Исследуем работу алгоритма поиска экстремума функции с двумя переменными $z = \sqrt{9 - x^2 - y^2}$, которая представляет собой верхнюю часть сферы с радиусом 3. Для того, чтобы можно было создать оптимизирующий эксперимент, воспользуемся простой моделью:

```
GENERATE 10
TERMINATE 1
```

В ней через 10 единиц модельного времени приходит транзакт, завершающий процесс моделирования, который должен начинаться с помощью оператора управления START 1.

Диалоговое окно эксперимента заполним так, как показано на рис. 9.13 (при задании имен констант, переменных лучше использовать символ подчеркивания – это гарантирует избежание конфликтов имен).

В данном случае полезно в Movement Limits поменять пределы на 0 и 1, вследствие чего намного уменьшается количество итераций и увеличивается точность. Результат получается идеальным.

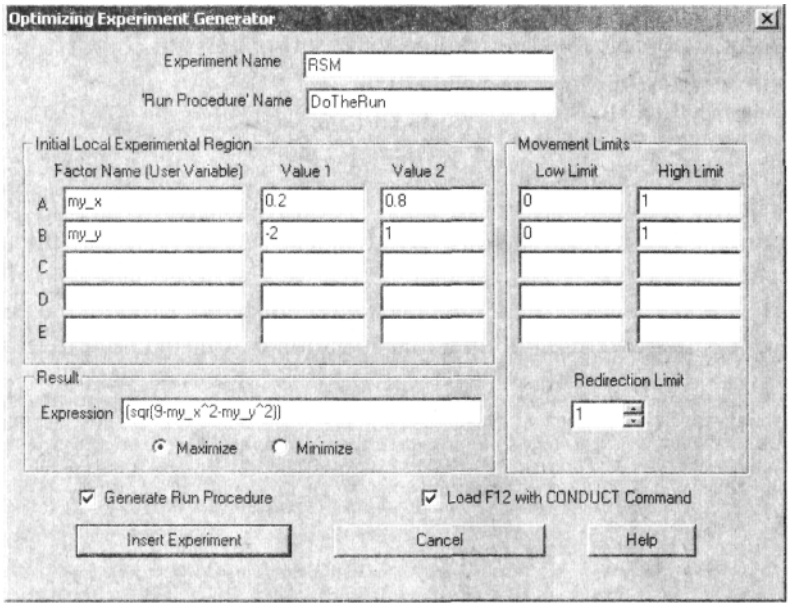


Рис. 9.13

После нажатия кнопки **Insert Experiment** откроется окно **Run Procedure Generation** (рис. 9.14), в котором отображается автоматически сгенерированный текст Run-процедуры, названной именем Do-TheRun.

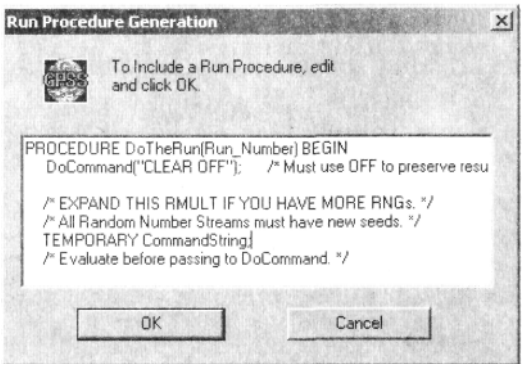


Рис. 9.14

Нажмем кнопку **OK**, в результате чего код на языке PLUS будет добавлен в текст модели. В него нужно внести исправления, в данном случае необходимо следующим образом изменить процедуру **DoTheRun:**

```
PROCEDURE DoTheRun(RunNumber) BEGIN
   DoCommand("CLEAR OFF");
   DoCommand("START 1");
END;
```

Теперь надо оттранслировать модель, выполнив пункт меню **Command/Create Simulation.** При этом мы увидим, что все процедуры регистрируются.

Нажмем клавишу **F12** и начнем эксперимент. В журнале сессии будет представлена следующая информация о ходе эксперимента.

```
01/20/03 22:59:07   **** Experiment in Progress. ****
01/20/03 22:59:07  Simulation in Progress.
01/20/03 22:59:07  A Simulation in an Experiment has ended.  Clock is 10.000000.
01/20/03 22:59:07   "Run 1.   Yield=2.227105745132009.  my_x=.2; my_y=-2;"
01/20/03 22:59:07  Simulation in Progress.
01/20/03 22:59:07  A Simulation in an Experiment has ended.  Clock is 10.000000.
01/20/03 22:59:07   "Run 2.   Yield=2.821347195933177.  my_x=.2; my_y=1;"
01/20/03 22:59:07  Simulation in Progress.
01/20/03 22:59:07  A Simulation in an Experiment has ended.  Clock is 10.000000.
01/20/03 22:59:07   "Run 3.   Yield=2.08806130178211.  my_x=.8; my_y=-2;"
01/20/03 22:59:07  Simulation in Progress.
01/20/03 22:59:07  A Simulation in an Experiment has ended.  Clock is 10.000000.
01/20/03 22:59:07   "Run 4.   Yield=2.712931993250107.  my_x=.8; my_y=1;"
01/20/03 22:59:07  Simulation in Progress.
01/20/03 22:59:07  A Simulation in an Experiment has ended.  Clock is 10.000000.
01/20/03 22:59:07   "Run 5.   Yield=2.91547594742265.  my_x=.5; my_y=-.5;"
01/20/03 22:59:07  Simulation in Progress.
01/20/03 22:59:07  A Simulation in an Experiment has ended.  Clock is 10.000000.
01/20/03 22:59:07   "Run 6.   Yield=2.91547594742265.  my_x=.5; my_y=-.5;"
01/20/03 22:59:07  Simulation in Progress.
01/20/03 22:59:07  A Simulation in an Experiment has ended.  Clock is 10.000000.
01/20/03 22:59:07   "Run 7.   Yield=2.91547594742265.  my_x=.5; my_y=-.5;"
01/20/03 22:59:07           Second order model fails. Saddle point detected.
01/20/03 22:59:07            Using Model:
01/20/03 22:59:07               Y = 2.86125 -0.206216  A +0.203185  B
01/20/03 22:59:07            Optimum could not be predicted.
01/20/03 22:59:07            Saddle point detected. Critical point is not opti-
mal.
01/20/03 22:59:07    RSM_FitSurfaceToData() returns 2.
01/20/03 22:59:07   "Moving ..."
01/20/03 22:59:07  Simulation in Progress.
01/20/03 22:59:07  A Simulation in an Experiment has ended.  Clock is 10.000000.
01/20/03 22:59:07   "Run 8.   Yield=3.  my_x=0; my_y=0;"
01/20/03 22:59:07   "Move ending."
01/20/03 22:59:08  Simulation in Progress.
01/20/03 22:59:08  A Simulation in an Experiment has ended.  Clock is 10.000000.
01/20/03 22:59:08   "Run 9.   Yield=2.580697580112788.  my_x=-.3; my_y=-1.5;"
01/20/03 22:59:08  Simulation in Progress.
01/20/03 22:59:08  A Simulation in an Experiment has ended.  Clock is 10.000000.
01/20/03 22:59:08   "Run 10.   Yield=2.580697580112788.  my_x=-.3; my_y=1.5;"
01/20/03 22:59:08  Simulation in Progress.
01/20/03 22:59:08  A Simulation in an Experiment has ended.  Clock is 10.000000.
01/20/03 22:59:08   "Run 11.   Yield=2.580697580112788.  my_x=.3; my_y=-1.5;"
01/20/03 22:59:08  Simulation in Progress.
```

```
01/20/03 22:59:08  A Simulation in an Experiment has ended.  Clock is 10.000000.
01/20/03 22:59:08   "Run 12.   Yield=2.580697580112788.  my_x=.3; my_y=1.5;"
01/20/03 22:59:08  Simulation in Progress.
01/20/03 22:59:08  A Simulation in an Experiment has ended.  Clock is 10.000000.
01/20/03 22:59:08   "Run 13.   Yield=3.  my_x=0; my_y=0;"
01/20/03 22:59:08  Simulation in Progress.
01/20/03 22:59:08  A Simulation in an Experiment has ended.  Clock is 10.000000.
01/20/03 22:59:08   "Run 14.   Yield=3.  my_x=0; my_y=0;"
01/20/03 22:59:08  Simulation in Progress.
01/20/03 22:59:08  A Simulation in an Experiment has ended.  Clock is 10.000000.
01/20/03 22:59:08   "Run 15.   Yield=3.  my_x=0; my_y=0;"
01/20/03 22:59:08  Simulation in Progress.
01/20/03 22:59:08  A Simulation in an Experiment has ended.  Clock is 10.000000.
01/20/03 22:59:08   "Run 16.   Yield=2.98496231131986.  my_x=.3; my_y=0;"
01/20/03 22:59:08  Simulation in Progress.
01/20/03 22:59:08  A Simulation in an Experiment has ended.  Clock is 10.000000.
01/20/03 22:59:08   "Run 17.   Yield=2.98496231131986.  my_x=-.3; my_y=0;"
01/20/03 22:59:08  Simulation in Progress.
01/20/03 22:59:08  A Simulation in an Experiment has ended.  Clock is 10.000000.
01/20/03 22:59:08   "Run 18.   Yield=2.598076211353316.  my_x=0; my_y=1.5;"
01/20/03 22:59:09  Simulation in Progress.
01/20/03 22:59:09  A Simulation in an Experiment has ended.  Clock is 10.000000.
01/20/03 22:59:09   "Run 19.   Yield=2.598076211353316.  my_x=0; my_y=-1.5;"
01/20/03 22:59:09            Using Model:
01/20/03 22:59:09               Y = 3.00049 +0 A +0 B
01/20/03 22:59:09                    +0 A B
01/20/03 22:59:09                    -0.180775 A^2 -0.17918 B^2
01/20/03 22:59:09            Predicted optimum yield is 3.00049.
01/20/03 22:59:09            Optimum is in the local Experimental Region.
01/20/03 22:59:09    RSM_FitSurfaceToData() returns 4.
01/20/03 22:59:09   "Running the predicted Optimum."
01/20/03 22:59:09  Simulation in Progress.
01/20/03 22:59:09  A Simulation in an Experiment has ended.  Clock is 10.000000.
01/20/03 22:59:09   "Run 20.   Yield=3.  my_x=0; my_y=0;"
01/20/03 22:59:09    Experiment ended.
```

Полученные оптимальные значения (входные переменные my_x = 0, my_y = 0, выходная переменная Yield = 3) совпадают с теоретическими.

Это как раз тот случай, когда можно увидеть, что сначала была попытка применить линейную модель Y = 2.86125 – 0.206216A + 0.203185B, она не удалась и поэтому была применена модель второго порядка $Y = 3.00049 + 0A + 0B + 0AB – 0.180775A^2 – 0.17918B^2$. Также этот пример продемонстрировал, что оптимум не обязательно должен находиться между указанными значениями полей **Value 1, Value** 2 диалогового окна **Optimizing Experiment Generator.**

Пример 9.6

Рассмотрим задачу нахождения оптимального количества работников производственного участка (аналог задачи о печи [10]).

Имеется производственный участок, на котором работают несколько человек. Чтобы изготовить изделие, работники сначала его собирают в течение 30±5 мин (процесс сборки осуществляется параллельно), **А** потом обрабатывают с помощью общей машины в течение 8+2 мин. В каждый момент времени на машине можно обрабатывать только одно изделие. После этого изделие считается готовым, и работник начинает собирать новое изделие. Доход от одного изделия составляет 5 единиц стоимости, использование машины обходится в 400 единиц стоимости в неделю, зарплата одного рабочего 150 единиц стоимости в неделю. Нужно определить, сколько нужно рабочих, чтобы доход от производства был максимальным. Моделирование необходимо выполнить для пятидневной недели с восьмичасовым рабочим днем.

Программа:

```
NWorkers    EQU 3                                   ; Количество рабочих
PRIBY       FVARIABLE   N$OUT#5-400-NWorkers #150    ; Расчет дохода
            GENERATE    ,,,NWorkers                 ; Транзакты моделируют
*                                                   ; рабочих
BACK1       ADVANCE     30,5                        ; Работник собирает изделие
            SEIZE       Machine                     ; Работник использует машину
            ADVANCE     8,2
            RELEASE     Machine
OUT         TRANSFER    ,BACK1                      ; Переход к сборке нового
*                                                   ; изделия
            GENERATE    2400                        ; Таймер (60мин*8час*5дней)
            TERMINATE   1
```

Нужно найти такое значение параметра **NWorkers,** при котором доход достигал бы максимального значения. В выражение дохода **PRIBY** входит количество изготовленных изделий, равное значению

СЧА **N$OUT.** Понятно, что доход может быть как отрицательным (убытки от производства), так и положительным (когда производство становится прибыльным).

Диалоговое окно эксперимента заполним, как показано на рис. 9.15. Используем ту же процедуру **DoTheRun,** что и в примере 9.5.

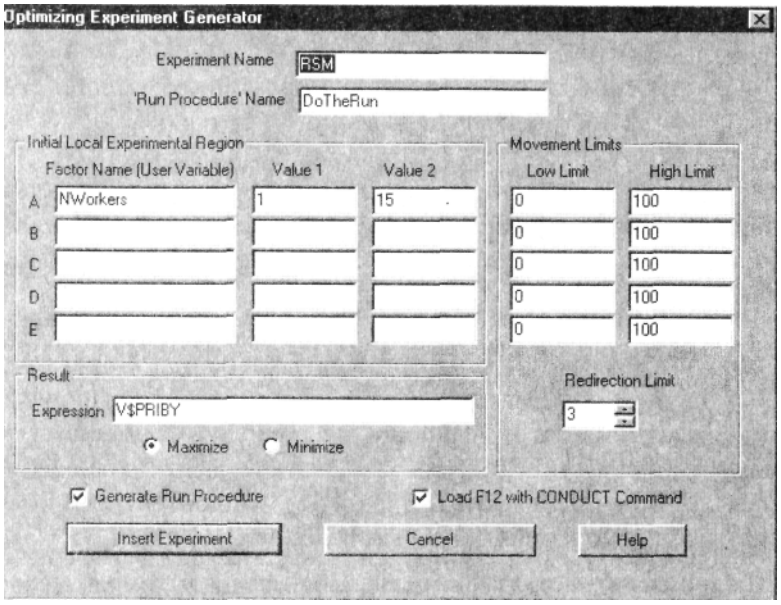


Рис. 9.15

```
02/15/03 20:27:35  CONDUCT RSM()
02/15/03 20:27:35   **** Experiment in Progress. ****
02/15/03 20:27:35  Simulation in Progress.
02/15/03 20:27:35  A Simulation in an Experiment has ended.  Clock is
2400.000000.
02/15/03 20:27:35   "Run 1.   Yield=-235.  NWorkers=1;"
02/15/03 20:27:35  Simulation in Progress.
02/15/03 20:27:35  A Simulation in an Experiment has ended.  Clock is
2400.000000.
02/15/03 20:27:35   "Run 2.   Yield=-1185.  NWorkers=15;"
02/15/03 20:27:36  Simulation in Progress.
02/15/03 20:27:36  A Simulation in an Experiment has ended.  Clock is
2400.000000.
02/15/03 20:27:36   "Run 3.   Yield=-130.  NWorkers=8;"
02/15/03 20:27:36  Simulation in Progress.
02/15/03 20:27:36  A Simulation in an Experiment has ended.  Clock is
2400.000000.
02/15/03 20:27:36   "Run 4.   Yield=-120.  NWorkers=8;"
02/15/03 20:27:36  Simulation in Progress.
02/15/03 20:27:36  A Simulation in an Experiment has ended.  Clock is
2400.000000.
02/15/03 20:27:36   "Run 5.   Yield=-140.  NWorkers=8;"
02/15/03 20:27:36           Using Model:
02/15/03 20:27:36              Y = -344.694 +121.531 A -11.8367 A^2
02/15/03 20:27:36           Predicted optimum yield is -32.7478.
02/15/03 20:27:36           Optimum is in the local Experimental Region.
02/15/03 20:27:36    RSM_FitSurfaceToData() returns 4.
02/15/03 20:27:36   "Running the predicted Optimum."
02/15/03 20:27:36  Simulation in Progress.
02/15/03 20:27:36  A Simulation in an Experiment has ended.  Clock is
2400.000000.
02/15/03 20:27:37   "Run 6.   Yield=254.9568965517241.  NWork-
ers=5.133620689655173;"
02/15/03 20:27:37   Experiment ended.
```

Таким образом, получен следующий результат: в процессе поиска решения была построена модель второго порядка $Y = -344.694 + 121.531A - 11.8367A^2$.

Оптимальные значения: входная переменная NWorkers=5.13, выходная переменная Yield = 254.96. С учетом условия целочисленности количества работников, имеем: оптимальное число работников для этой модели равно 5 .

При некоторых значениях полей Value 1, Value 2 диалогового окна **Optimizing Experiment Generator** процедура оптимизирующего эксперимента не выполняет последний прогон «Running the predicted Optimum» и не выдает искомый результат. Также может быть выдано такое сообщение «Goodness of fit test fails. Standard Error is 0. Cannot perform F test". В этих случаях нужно изменить значения Value 1 и Value 2.

Эксперименты пользователя. Пользовательский эксперимент в GPSS World – очень гибкий и мощный инструмент, так как с помощью встроенного языка PLUS можно задать выполнение разнообразных, достаточно сложных действий. Кроме того, выполняющийся эксперимент позволяет вызывать любые команды GPSS из процедур на языке PLUS. Подпрограммы на языке PLUS делятся на процедуры (в заголовке используется ключевое слово **PROCEDURE)** и эксперименты **(EXPERIMENT)** Эти подпрограммы должны находиться в файле модели. Разница между этими двумя видами подпрограмм заключается в следующем. Процедуры с помощью оператора **RETURN** могут возвращать значение, которое можно посмотреть с помощью команды **SHOW,** либо использовать в других подпрограммах. Для этого сначала нужно оттранслировать модель вместе с процедурой или же просто процедуру, если модель не используется. При этом процедура регистрируется. Затем в меню нужно выбрать пункт **Command/SHOW** и написать там название процедуры, указав в скобках нужные параметры или ничего не указывая (если параметров нет). После этого процедура выполняется и выдается результат.

Процедуры на языке PLUS также могут использоваться в командах GPSS в качестве операндов. В этом случае они заключаются в круглые скобки. Эти процедуры, в свою очередь, в качестве параметров могут использовать вызовы других процедур. Это очень мощное новшество GPSS World значительно повышает возможности и гибкость модели.

Эксперимент отличается от процедуры тем, что он может быть вызван (выполнен) только пользователем. Для этого нужно выбрать пункт меню **Command/CONDUCT** и написать там название эксперимента, указав в круглых скобках параметры, если они имеются, или ничего не указывая, если параметров нет. Что нового дает эксперимент? Ключевым моментом является то, что с помощью библиотечной процедуры **DoCommand** можно вызывать любую команду GPSS, а это позволяет многократно выполнять прогоны модели и в каждом прогоне получать доступ к текущим результатам. Из подпрограммы эксперимента могут вызываться процедуры, которые в свою очередь могут вызывать другие процедуры и т.д. Причем из любой процедуры, на какой бы глубине вложенности вызовов она ни была, можно вызывать процедуру **DoCommand.** Поэтому структура выполняемого кода эксперимента может быть какой угодно сложной. Таким образом, пользовательский эксперимент объединяет в себе набор процедур на языке PLUS, в котором одна из процедур имеет в заголовке ключевое слово **EXPERIMENT**. Выполнить такой эксперимент можно с помощью команды **CONDUCT.** Разные эксперименты могут иметь общие вызываемые процедуры.

В пользовательском эксперименте для задания последовательности команд и исходных данных каждого прогона модели удобно использовать PLUS-процедуру. Такая процедура может инициализировать генераторы случайных чисел, выполнять команды и управлять имитацией.

В руководстве по GPSS World пользовательские эксперименты используются исключительно с процедурой ANOVA, однако, их можно использовать для решения совершенно разных задач. Рассмотрим несколько практических примеров пользовательского эксперимента.

В этих задачах нужно найти значения параметров системы, при которых ее критерий эффективности достигает оптимума. Алгоритм поиска решения базируется на том, что экономические показатели реальных систем обычно представляют собой функции, имеющие единственный локальный экстремум, который и является глобальным. Рассмотрим еще раз задачу из примера 9.6. Использование в этом примере оптимизирующего эксперимента было не совсем корректным. Оптимизирующий эксперимент рассчитан на модели, работающие в стационарном режиме, а для данного примера нет гарантий, что за пять дней работы моделируемая система войдет в такой режим.

Поэтому следует считать, что система работает в переходном режиме, и для каждого значения количества работников необходимо провести некоторое число прогонов модели.

Воспользуемся программой из примера 9.6 и организуем для этой модели пользовательский эксперимент, но прежде рассмотрим алгоритм поиска оптимума. Предполагаем, что целевая функция, определяющая прибыль, имеет единственный оптимум. Если бы времена сборки и обработки детали на машине были бы детерминированными, то оптимальное количество работников находилось бы вблизи величины, равной отношению времени сборки к времени обработки. В этом случае машина была бы максимально загружена и работники не простаивали бы в ожидании ее освобождения. В действительности, в силу случайного характера этих времен, оптимальное число работников будет отличаться от значения, рассчитанного таким образом. Так как количество вариантов невелико, то можно воспользоваться методом их перебора, проводя эксперименты с разным числом работников,

начиная с одного. Таким образом, предполагаемый доход будет вначале увеличиваться до некоторого момента, а когда на каком-то шаге (при большем количестве работников) доход станет меньше, чем на предыдущем шаге, нужно остановиться. Оптимальное количество работников соответствует шагу с самым большим значением дохода.

Для оценки дохода на каждом шаге при некотором фиксированном количестве работников будем выполнять 10 прогонов с разными начальными множителями генераторов случайных чисел и рассчитывать среднее арифметическое получаемого дохода. В общем случае число прогонов надо рассчитывать с учетом требуемой точности оценки прибыли.

Добавим к модели программный текст на языке PLUS, реализующий вышеприведенный алгоритм поиска оптимального количества работников.

```
EXPERIMENT Go(Par1) BEGIN
    TEMPORARY CurYield,ShowString,CommandString;
    NWorkers=Par1;
    CurYield=GetResult();
    ShowString = PolyCatenate(" Profits  ",String(CurYield),". ");
    ShowString = PolyCatenate(ShowString," Number of workers ",String(NWorkers),".
    CommandString = PolyCatenate("SHOW """,ShowString,""""", "" );
    DoCommand(CommandString);
END;

PROCEDURE GetResult() BEGIN
    TEMPORARY Sum,CurYield,Ind_i;
    Ind_i=1;
    Sum=0;
    WHILE (Ind_i<10) DO BEGIN
          CurYield=DoTheRun(Ind_i);
          Sum=Sum+CurYield;
          Ind_i=Ind_i+1;
    END;
    RETURN (Sum/(Ind_i-1));
END;

PROCEDURE DoTheRun(Run_Number) BEGIN
    DoCommand("CLEAR OFF");
    TEMPORARY CommandString;
    CommandString = Catenate("RMULT ",Run_Number#3);
    DoCommand(CommandString);
    DoCommand("START 1");
    RETURN (FC$Machine#5-400-NWorkers#150);
END;

EXPERIMENT Seek_Opt() BEGIN
    TEMPORARY ShowString,CommandString;
    TEMPORARY Prev1,Cur1,Flag1;
    NWorkers=0;
    Prev1=-100000;
    Flag1=1;
    WHILE (Flag1 'NE' 0) DO BEGIN
       NWorkers=NWorkers+1;
       Cur1=GetResult();
       IF (Cur1<Prev1) THEN BEGIN
            /*show results*/
     ShowString = PolyCatenate(" Optimum. Profits ",String(Prev1),". ");
 ShowString=PolyCatenate(ShowString," Number of workers ",String(NWorkers-
1),".");
        CommandString = PolyCatenate("SHOW """,ShowString,""""", "" );
        DoCommand(CommandString);
        Flag1=0;
        END;
        ELSE Prev1=Cur1;
    END;
END;
```

Переменная Flag1 используется для управления циклом. Сначала ей присваивается значение 1, условие выхода из цикла: Flag1 = 0. В переменной Prev1 хранится значение дохода на предыдущем шаге. Вначале ей присваивается очень большое по модулю отрицательное значение, чтобы получаемый доход на первом шаге заведомо был больше. На каждой итерации цикла количество работников NWorkers увеличивается на 1. Перед входом в цикл оно устанавливается равным 0.

Переменная Curl получает значение дохода на текущем шаге. Если предыдущее значение дохода больше текущего, то выдается результат и осуществляется выход из цикла.

Для запуска эксперимента сначала нужно оттранслировать модель вместе с написанным программным текстом. Для этого выполняем пункт меню **Command/Create Simulation.** Все написанные процедуры регистрируются в системе. Затем в меню нужно выбрать пункт

Command/CONDUCT. В результате появится диалоговое окно **'Conduct Experiment' Command** (рис. 9.16).

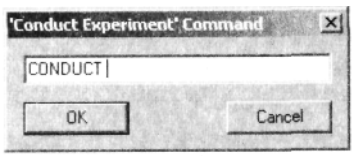


Рис. 9.16

Для запуска пользовательского эксперимента, осуществляющего поиск оптимального числа работников, нужно в поле окна дописать название эксперимента Seek_Opt(). Если же нас интересует оценка величины прибыли для конкретного числа рабочих, например, для трех, то необходимо вызвать эксперимент Go(3).

В результате выполнения эксперимента Seek_Opt() в журнале сессии появится множество записей, соответствующих выполняемым прогонам и, в конце концов, такой результат:

```
02/10/03 01:15:21    " Optimum. Profits 298.8888888888889. Number of workers 5."
```

Понятно, что при желании можно выдавать любые промежуточные данные. Таким образом, оптимальное количество рабочих составляет 5, **А** доход равен 298,89 единиц стоимости, что совпадает с оптимизирующим экспериментом (отличие в величине дохода объясняется разными подходами к формированию этой величины).

Подобным образом можно искать оптимум в задачах с любым количеством изменяемых параметров, нужно только вложить друг в друга циклы WHILE и перед поиском наилучшего значения целевой функции для конкретного значения параметра запоминать значение отклика для предыдущего значения этого параметра.

Пример 9.7 [10]. Рассмотрим швейное производство, в котором используются 40 собственных станков для изготовления продукции. Эти станки работают 150 ± 26 часов, после чего ломаются. Поломавшийся станок забирается в ремонтную мастерскую, где работает некоторое число рабочих, каждый из которых может ремонтировать один станок. Ремонт осуществляется в течение 8 ± 3 часов, после чего станки снова готовы к работе. Недозагрузка производственных мощностей обходится в 140 единиц стоимости в час за один станок из-за потерь, которые несет производство по недовыпуску продукции. Оплата одного рабочего составляет 4,75 единиц стоимости в час.

Кроме 40 собственных станков, имеющихся у предприятия, можно для подмены ломающихся арендовать аналогичные станки, Это обходится в 2,5 единицы стоимости в час за станок. Если в работе уже задействовано 40 станков, то отремонтированный станок становится в резерв.

Определить количество арендуемых станков и количество работников в ремонтной мастерской, при которых средние затраты на производство были бы минимальными.

Для поиска оптимума будем «разрезать» поверхность отклика плоскостями, которые соответствуют фиксированным значениям количества арендуемых станков и нанятых ремонтников. При фиксированном значении одного параметра график функции затрат будет представлять собой кривую в двумерном пространстве с одним минимумом (его можно найти точно так же, как в предыдущем примере). Таким образом, если мы будем последовательно изменять один параметр, по которому производится разрезание поверхности, то получаемые значения функции затрат при этом фиксируемом параметре будут сначала уменьшаться, **А** потом в какой-то момент снова пойдут на увеличение. Самое маленькое значение и будет наилучшим. Это следует из предполагаемого вида поверхности.

Программа:

```
M_HIRED EQU 0                          ; Количество арендуемых машин
N_WRK     EQU 1                        ; Количество ремонтников
REP       STORAGE 3                    ; Начальное количество ремонтников
MANUF     STORAGE 40                   ; Количество станков

          GENERATE ,,,(40+M_HIRED)     ; Общее число станков
BACK1     ENTER MANUF
          ADVANCE 150,26               ; Время до отказа станка
```

```
LEAVE MANUF                ; Передать на ремонт
ENTER REP                  ; Начать ремонт
ADVANCE 8,3                ; Время ремонта
LEAVE REP                  ; Закончить ремонт
TRANSFER ,BACK1            ; Вернуть станок в производство

GENERATE 2080              ; Таймер (8часов*5дней*52недели)
TERMINATE 1
```

Тексты пользовательских PLUS-процедур:

```
EXPERIMENT Go(Par1,Par2) BEGIN
    TEMPORARY CurYield,ShowString,CommandString;
    M_HIRED=Par1;
    N_WRK=Par2;
    CurYield=GetResult();
    ShowString = PolyCatenate(" Losses  ",String(CurYield),". ");
    ShowString =PolyCatenate(ShowString," Hired machines
",String(M_HIRED),"." );
    ShowString = PolyCatenate(ShowString," Repairers  ",String(N_WRK), "." );
    CommandString = PolyCatenate("SHOW """,ShowString,""""", "" );
    DoCommand(CommandString);
END;

PROCEDURE GetResult() BEGIN
    TEMPORARY Sum,CurYield,Ind_i;
    Ind_i=1;
    Sum=0;
    WHILE (Ind_i<10) DO BEGIN
          CurYield=DoTheRun(Ind_i);
          Sum=Sum+CurYield;
          Ind_i=Ind_i+1;
    END;
    RETURN (Sum/(Ind_i-1));
END;

PROCEDURE DoTheRun(Run_Number) BEGIN
    DoCommand("CLEAR OFF");
    TEMPORARY CommandString;
    CommandString = Catenate("RMULT ",Run_Number#3);
    DoCommand(CommandString);
    CommandString = Catenate("REP STORAGE ",String(N_WRK));
    DoCommand(CommandString);
    DoCommand("START 1");
    RETURN ((40-SA$MANUF)#140+4.75#N_WRK+2.5#M_HIRED);
END;

EXPERIMENT Seek_Opt() BEGIN
    TEMPORARY ShowString,CommandString;
    TEMPORARY Prev1,Cur1,Flag1;
    TEMPORARY Prev2,Cur2,Flag2,Prev_N_WRK;
    M_HIRED=0-1;
    Prev1=100000;
    Flag1=1;
    Prev_N_WRK=0;
    WHILE (Flag1 'NE' 0) DO BEGIN
        M_HIRED=M_HIRED+1;
```

```
        N_WRK=1-1;
        Prev2=100000;
        Flag2=1;

        WHILE (Flag2 'NE' 0) DO BEGIN
            N_WRK=N_WRK+1;
            Cur2=GetResult();
            IF (Cur2>Prev2) THEN BEGIN
            Flag2=0;
            Show_1(Prev2);
            END;
            ELSE Prev2=Cur2;
        END;

        Cur1=Prev2;
        IF (Cur1>Prev1) THEN BEGIN
            /*show results*/
            ShowString = PolyCatenate(" Optimum. Losses  ",String(Prev1),".
");'
ShowString = PolyCatenate(ShowString," Hired machines  ",String(M_HIRED-1),
"." );
    ShowString = PolyCatenate(ShowString," Repairers  ",String(Prev_N_WRK),
"." );
            CommandString = PolyCatenate("SHOW """,ShowString,""""", "" );
            DoCommand(CommandString);
            Flag1=0;
        END;
        ELSE BEGIN
            Prev1=Cur1;
            Prev_N_WRK=N_WRK-1;
        END;
    END;
END;

PROCEDURE Show_1(Losses) BEGIN
    TEMPORARY ShowString,CommandString;
    ShowString = PolyCatenate(" Cur. losses  ",String(Losses),". ");
ShowString =PolyCatenate(ShowString," Cur. hired machines
",String(M_HIRED),"." );
  ShowString = PolyCatenate(ShowString," Cur. repairers  ",String(N_WRK-
1),"." );
    CommandString = PolyCatenate("SHOW """,ShowString,""""", "" );
    DoCommand(CommandString);
END;
```

Оптимизировать в данном случае нужно средние затраты на производство за час, которые вычисляются следующим образом:

```
(40-SA$MANUF)#140+4.75#N_WRK+2.5#M_HIRED
```

Среднее число работающих станков за все время моделирования, определяет СЧА SA$MANUF. Начальное количество ремонтников в модели может не задаваться. Оно нужно только для автономного запуска модели без подключенного эксперимента.

Внешний цикл эксперимента Seek_Opt() организует изменение количества арендуемых станков. Внутренний цикл последовательно изменяет количество работников (чтобы найти минимум функции затрат, становящейся двумерной). Переменные Flagl, Flag2 управляют выходом из циклов. Переменные Prevl, Curl, Flagl соответствуют внешнему циклу; Prev2, Cur2, Flag2 – внутреннему. Переменная Prev_N_WRK хранит значение количества рабочих на предыдущем шаге внешнего цикла (при меньшем фиксированном значении количества арендуемых станков), соответствующее наименьшему значению функции затрат. Процедура Show_1 предназначена для отображения промежуточных наименьших значений функции затрат и при каких значениях параметров они достигаются.

Для данной модели выполним два пользовательских эксперимента. Первый (Go) позволяет определить затраты на производство при конкретных значениях числа арендуемых станков и ремонтников, которые задаются как параметры в процедуре. Как и в предыдущем примере, средние затраты рассчитываются по 10 прогонам модели. Для выполнения этого эксперимента необходимо оттранслировать модель с подключенными процедурами, выполнить пункт меню **Command/CONDUCT** и в диалоговом окне **'Conduct Experiment' Command** вызвать процедуру GO

CONDUCT Go(5,3),

где 5 – число арендуемых станков и 3 – число ремонтников. Результат этого эксперимента:

```
02/20/03 09:28:32   " Losses  100.3012130170936.  Hired machines  5. Repairers  3."
```

Второй эксперимент Seek_Opt() запускается командой

CONDUCT Seek_Opt()

и позволяет методом направленного перебора найти вариант, соответствующий минимальным затратам. Для каждого нового варианта сочетания числа станков и рабочих при старте в операторе STORAGE переопределяется емкость МКУ для нового значения количества ремонтников. Опустив промежуточные результаты, приведем конечный результат для этого эксперимента.

```
02/20/03 10:44:20 " Cur. losses  52.7758888547009.  Cur. hired machines  8. Cur. repairers  6."
02/20/03 10:44:20 " Optimum. Losses  52.54681494444466.  Hired machines  7. Repairers 6."
```

Из приведенных результатов видно, что на предыдущем шаге рассматривался вариант с 8 станками и 6 рабочими, однако, затраты при этом сочетании оказались больше, чем при 7 станках и 6 рабочих.

Поэтому в качестве окончательного был принят вариант с 7 станками и 6 рабочими при средних часовых затратах на производство 52.5568 единиц стоимости.

9.l2. Выбор наилучшего варианта структуры системы

Во многих случаях при моделировании приходится искать наилучший вариант структуры моделируемой системы или алгоритмов ее функционирования. По полученным результатам экспериментов с моделью обычно обнаруживается наличие некоторой проблемы, которую надо локализовать и определить причины ее возникновения. Для этого формулируют несколько гипотез, которые потом проверяют.

Не существует единого подхода к формулированию гипотез, поскольку такая процедура зависит от конкретной модели и проблемы, которая решается с помощью этой модели. Поэтому можно лишь предложить некоторые общие методы формулирования гипотез [17].

1. **Идентификация похожих ситуаций.** Используется существующий опыт проведения подобных работ и формулируются гипотезы, похожие на уже известные.
2. **Выявление резко отличающихся значений.** Такие значения часто могут приводить к правильным гипотезам. Их следует игнорировать только после тщательного изучения и объяснения.
3. **Выявление закономерностей.** В модели определяют интересные закономерности во времени, такие как циклы или тенденции. Для этого целесообразно применять графические методы: диаграмму состояний системы, временные ряды и диаграммы Ганта.
4. **Выявление корреляций.** Корреляция между параметрами и показателями критерия эффективности системы может привести к правильным гипотезам.

Выявление и анализ несоответствий. Существование очевидных взаимосвязей между данными, которые проверяются, с данными, взятыми из разных источников; между данными, связанными с системой; между полученными данными на модели и их ожидаемыми значениями. Выявленные несоответствия с большой вероятностью приводят к гипотезам.

Решение задачи анализа системы с помощью имитационной модели сводится большей частью к выявлению так называемых «узких мест». Пусть S – система, P – показатель эффективности, $x_1,x_2,..., x_i ..., x_n$ – n параметров модели. Величины x_i – это параметры окружающей среды и внутренние параметры модели. Допустим, что увеличение P приводит к улучшению показателя. Функция $P(x_i)$, $i = 1, ..., n$ обычно характеризуется явно выраженными нелинейностями, которые приобретают форму узких мест.

Система S ***имеет узкое место*** относительно P в определенной области пространства параметром D, если в этой области значение P существенным образом возрастает при изменении одного или немногих параметров. Даже большие изменения других параметров не приводят к ощутимому изменению P, если они не выводят систему за границы области D. В этом случае не рассматриваются области, которые окружают глобальные или локальные максимумы P. Узкие места возникают тогда, когда к некоторым ресурсам системы выстраиваются большие очереди из-за нарушения баланса между потоком запросов к этому ресурсу и возможностями ресурса удовлетворять их.

Признаком наличия узкого места в системе может быть большая разница между коэффициентами использования компонентов системы, в особенности, если один из коэффициентов стремится к единице. Другим признаком может стать выявление несоответствия производительности системы ожидаемой.

Один из подходов к улучшению показателей эффективности P систем связан с последовательным устранением узких мест. В общем случае изменение одного или нескольких параметров x_i, модели не всегда может привести к балансированию системы, так как могут обнаруживаться новые узкие места, которые до этого были скрыты только что устраненными. Кроме того, внесенные изменения в модель

могут перевести систему S в другую область пространства параметров, в которой критерий P ограничивается другим узким местом.

Если моделируемая система отображается сетью СМО, то для предварительного анализа ее работы и поиска узких мест используют операционный анализ (см. параграф 2.4). В соответствии с операционным анализом пошаговое устранение узких мест в системе и балансирование коэффициентов загрузки узлов в сети позволяет получить минимальное время пребывания требований в системе, то есть получить ее максимальную пропускную способность. Во многих случаях это дает возможность найти начальное решение для определения необходимого количества устройств в узлах сети, которые потом уточняются при имитационном моделировании.

ГЛАВА 10. ПРИМЕРЫ ПРИНЯТИЯ РЕШЕНИЙ С ПОМОЩЬЮ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ

10.1. Моделирование производственного участка

Процедуру принятия решений рассмотрим на примере производственного участка с заданными маршрутами движения деталей. Этот пример аналогичный примеру, описанному в работе [10], за исключением процедур принятия решений и включением конвейера для передачи деталей между станками.

Некоторый производственный участок имеет четыре типа станков: токарный, сверлильный, шлифовальный, фрезерный. На участке обрабатываются детали четырех типов. Каждый тип детали требует выполнения операций на определенных типах станков в последовательности, которая задается маршрутной картой. Структурная схема концептуальной модели изображена на рис. 10.1.

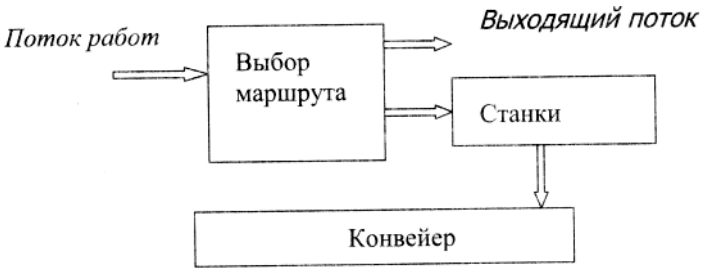


Рис. 10.1

Количество этапов обработки, последовательность прохождения и среднее время обработки для всех типов деталей приведены в маршрутной карте движения деталей по участку (табл. 10.1). Станки в маршрутной карте размещены по порядку выполнения работ.

Заготовки деталей поступают на участок с других участков по закону Пуассона со средним значением 24 заготовки деталей за 8 часов работы участка. Появление любого типа деталей равновероятно и не зависит от других типов работ. Характеристики изготовления одной детали каждого типа и доход приведены в табл. 10.2.

Целью моделирования работы производственного участка является определение наилучших управленческих решений усовершенствования технологического участка по критерию увеличения дохода от выполненных работ.

Концептуальная структура модели представляет собой виртуальную сеть СМО, в которой каждый тип детали имеет свой маршрут движения. Сеть имеет один виртуальный узел, в котором обслуживающие устройства (станки) меняют свои номера в зависимости от типа детали и ее этапа обработки.

Для разработки модели можно было бы использовать матрицы [10], однако, применение функций облегчает внесение изменений в маршрутную карту. Для реализации виртуального узла сети СМО используется прохождение одной и той же последовательности блоков **SEIZE – ADVANCE – RELEASE** и параметрическая настройка модели на конкретный станок с помощью косвенной адресации устройств обслуживания и функций. В функциях отображаются конкретные параметры типа детали, число этапов обработки, начальное значение этапа для каждого типа детали, маршрут прохождения детали через станки и время обработки на каждом станке.

Таблица 10.1

Тип детали	***Количество этапов обработки***	***Последовательность прохождения деталей через станки***	***Время обработки, мин***
1	6	Токарный Фрезерный Сверлильный Шлифовальный Сверлильный Токарный	8,8 12 12 13 10,5 11,5
2	4	Фрезерный Шлифовальный Фрезерный Сверлильный	20 14 14,5 16
3	5	Токарный Сверлильный Фрезерный Токарный Шлифовальный	17,6 19 14 11,6 30
4	4	Сверлильный Токарный Фрезерный Шлифовальный	19 16,8 13 19

Таблица 10.2

Тип деталей	***Доход, руб.***	***Себестоимость, руб.***	***Допустимое время изготовления, мин***	***Штраф за задержку изготовления свыше допустимого срока, руб.***
1	1550	350	1890	80
2	1850	420	1600	120
3	1350	280	2300	160
4	1450	315	1400	100

Рассмотрим подробнее эти функции. Функция EXPDIS задает пуассоновский поток поступления работ на участок. Тип детали определяется при помощи функции TYP, в которой задается вероятность появления деталей одного из четырех типов. Функция JTAP задает начальные значения маршрутов для каждого типа деталей, которые определяются функцией маршрутов ROUTE. Аргументом функции служит параметр транзакта P1, который определяет тип детали. Поскольку общее число маршрутов по всем деталям равняется 19, то функция JTAP задает начальный номер маршрута для каждого типа детали. Маршруты определяются функцией ROUTE последовательно, начиная с детали первого типа и кончая четвертым. Например, для третьего типа деталей начальное значение маршрута будет равняться сумме числа маршрутов для первого и второго типов деталей (6+4) плюс 1, то есть 11.

Функция JOB определяет количество этапов обработки для каждого типа детали. В качестве аргумента в ней используется параметр транзакта P1, который задает тип детали.

Функция ROUTE в качестве аргумента применяет параметр P3, который задает номер этапа обработки для каждого типа детали. Начальное значение параметра P3 определяется функцией JTAP. Значение функции ROUTE задает номер станка, то есть тип станка, обрабатывающего деталь, который запоминается в параметре P4.

Функция TIME аналогична функции ROUTE, но задает время обработки на каждом станке, значение которого запоминается в параметре P5.

В данной модели транзакт – это деталь определенного типа, задаваемого параметром P1. Второй параметр задает количество этапов обработки для детали данного типа и используется как счетчик,

работающий в режиме уменьшения. Если его значение равняется нулю, то над деталью полностью выполнены все операции на участке. Третий параметр траизакта задает номер этапа, который выполняется, и используется как счетчик, работающий в режиме увеличения. Четвертый параметр задает тип станка, **A** пятый – продолжительность обработки на станке детали данного вида.

За единицу модельного времени примем 0,1 мин.

Данные о распределении времени изготовления деталей собираются в таблицы 1–4 GPSS-программы модели, соответственно, для каждого типа детали. Первые интервалы таблиц задают допустимое время изготовления деталей, указанное в табл. 10.2. Это дает возможность определить, штрафуются ли изготовленные детали соответствующего типа или нет.

Учитывая подробное описание модели и комментарии, приведенные в тексте программы, логику работы модели можно не описывать. Однако укажем, что в любой момент времени транзакты-работы могут находиться в модели или в блоке **ASSIGN,** ожидая входа в блок **SEIZE,** или в блоке **ADVANCE,** где они задерживаются на время обработки детали.

По окончании моделирования печатается СБС, то есть список работ, которые выполняются на участке в конце рабочего дня.

Процедура определения наилучших решений относительно управления и усовершенствования технологического участка итерационная и связана с внесением изменений в технологию обработки деталей на участке. С этой целью выполняемые работы можно разделить на такие этапы:

1) выявление причин снижения производительности участка и уменьшение дохода от выполненных работ;
2) выдвижение гипотез и предварительный анализ их правильности;
3) проверка гипотез и сравнение полученных результатов;
4) выдача рекомендаций относительно усовершенствования технологического участка.

Порядок работы с имитационной моделью следующий:

1) осуществить пробный прогон модели и устранить ошибки, если они есть (при пробном прогоне желательно уменьшить время моделирования с целью сокращения счета);
2) выполнить полный прогон модели;
3) проанализировать результаты прогона и выдвинуть гипотезы относительно усовершенствования технологического участка.

Предлагаются такие рабочие гипотезы:

Гипотеза А. Перейти на новые режимы работы оборудования, то есть увеличить скорость выполнения работ на станках. Такие изменения скоростных режимов могут привести к некоторой потере качества, которая уменьшит доход, но прибыль может возрасти из-за увеличения общей производительности участка и сокращения незавершенного производства к концу рабочего дня. Кроме того, могут сократиться штрафы, вследствие несоблюдения допустимых сроков изготовления деталей. Возможные изменения режимов работы станков и размера дохода приведены в табл. 10.3. Благодаря переходу станков на скоростной режим скорость их работы можно увеличить на 20%.

Исходные данные для проверки гипотезы **A** приведены в табл. 10.3.

Для проверки гипотезы **A** необходимо в функции TIME изменить соответствующие времена обработки деталей.

Гипотеза В. Увеличение количества однотипных станков на участке. Это изменение приводит к таким же последствиям, что и в случае гипотезы *A,* однако, при этом не будут ухудшаться показатели качества изготовления деталей. Вместе с тем, доход уменьшится из-за амортизационных отчислений на новые станки (табл. 10.4).

Таблица 10.3

Увеличение скорости обработки на станках	**Уменьшение цены по типам деталей, %**			
	первый	**второй**	**третий**	**четвертый**
Одном	1,5	1,2	2,8	2,0
Двух	2,5	1,5	3,0	2,8

Трех	3,0	2,0	3,5	3,2
Четырех	3,5	2,8	3,8	3,6

Чтобы проверить гипотезу *B,* необходимо ввести в программу накопители и перейти от одноканальных устройств (блоки **SEIZE** и **RELEASE**) к МКУ (блоки **ENTER** и **LEAVE**). Вместительность накопителя (оператор **STORAGE**) задать в описательной части модели. Исходные данные для проверки гипотезы *B* приведены в табл. 10.4.

Таблица 10.4

Количество однотипных станков	**Увеличение себестоимости детали, изготовленной на станке, %**			
	Сверлильном	**Токарном**	**Фрезерном**	**Шлифовальном**
2	5	8	7	10
3	10	16	14	20
4	15	24	21	30

Гипотеза С. Предположим, что если упорядочить работы перед станком по уменьшению отношения величины штрафа детали ко времени ее обработки на станке, то уменьшается суммарный штраф за нарушения допустимых сроков обработки деталей.

Для проверки этой гипотезы необходимо ввести в модель новую переменную (FVARIABLE) для вычисления отношения величины штрафа для данного вида детали ко времени обработки, то есть ввести функцию штрафов с именем FINE, в зависимости от типа детали, и вычислить величину

VARIABLE FN$FINE/P$5#100

Эту переменную необходимо использовать для задания приоритета (блок **PRIORITY)** перед захватом станка после метки NEXT в программе.

Гипотезы A, B, C можно использовать одновременно.

С помощью моделирования необходимо проверить гипотезы, выбрать наилучший вариант усовершенствования технологического участка, описав стратегию выбора этого варианта, и вычислить доход. Вот программа:

```
************************************
*** Модель производственного участка ***
***          Начальная структура          ***
***  Введение дополнительных станков ***
************************************
SVER            EQU  1 ; присвоение эквивалентных значений
TOK             EQU  2
FREZ            EQU  3
SHLI            EQU  4
KNV             EQU  5
BAD1            EQU  1
BAD2            EQU  2
BAD3            EQU  3
BAD4            EQU  4
```

```
ROUTE       EQU  5
TIME        EQU  6
*
KNV         STORAGE  200        ; Вместительность конвейера
SVER        STORAGE  2          ; Определение
TOK         STORAGE  2          ; количества
FREZ        STORAGE  2          ; однотипных
SHLI        STORAGE  2          ; станков
** Функция себестоимости для каждого типа деталей **
CEB         FUNCTION  P1,D4
1,350/2,420/3,280/4,315
** Функция штрафов для каждого типа деталей **
MSHT        FUNCTION  P1,D4
1,80/2,120/3,160/4,100
** Функция дохода для каждого типа деталей **
MDOH        FUNCTION  P1,D4
1,1550/2,1850/3,1350/4,1450
** Функция допустимого времени пребывания  на участке **
**                для каждого типа деталей                **
DTIM        FUNCTION  P1,D4
1,1890/2,1600/3,2300/4,1400
** Функция распределения времени поступления деталей на участок  **
EXPDIS      FUNCTION  RN1,C24
0,0/.100,.104/.200,.222/.300,.355/.400,.509
.500,.690/.600,.915/.700,1.200/.750,1.380
.800,1.600/.840,1.830/.880,2.120/.900,2.300
.920,2.520/.940,2.810/.950,2.990/.960,3.200
.970,3.500/.980,3.900/.990,4.600/.995,5.300
.998,6.200/.999,7/1,8
**  Функция типов деталей
TYP         FUNCTION  RN1,D4
.25,1/.40,2/.75,3/1,4
**  Функция начальных значений маршрутов для каждого типа деталей  **
JTAP        FUNCTION  P1,D4
1,1/2,7/3,11/4,16
**  Функция количества этапов обработки для каждого типа деталей   **
JOB         FUNCTION  P1,D4
1,6/2,4/3,5/4,4
**  Функция типов станков для каждого типа деталей   **
ROUTE       FUNCTION  P3,D19
1,2/2,3/3,1/4,4/5,1/6,2
7,3/8,4/9,3/10,1
11,2/12,1/13,3/14,2/15,4
16,1/17,2/18,3/19,4
**  Функция времени наработки для каждого этапа
TIME        FUNCTION  P3,D19
1,88/2,120/3,120/4,130/5,105/6,92
7,200/8,140/9,145/10,160
11,176/12,190/13,140/14,116/15,300
16,190/17,168/18,130/19,190
** Функция времени движения деталей по участку
```

```
GO          FUNCTION  P3,D19
1,16/2,23/3,23/4,23/5,55/6,38/7,35
8,38/9,29/10,23/11,16/12,40/13,55
14,55/15,33/16,55/17,38/18,23/19,38
** Функция времени для выхода детали с участка
EXIT_       FUNCTION  P1,D4
1,68/2,33/3,16/4,16
** Сбор данных о времени пребывания по типам деталей
BAD1        TABLE     M1,1890,24000,2
BAD2        TABLE     M1,1600,24000,2
BAD3        TABLE     M1,2300,24000,2
BAD4        TABLE     M1,1400,24000,2
*  -й сегмент модели
            GENERATE 200,FN$EXPDIS
            ASSIGN    1,FN$TYP       ; Тип детали
            ASSIGN    2,FN$JOB       ; Количество этапов
            ASSIGN    3,FN$JTAP      ; Номер первого этапа
            ASSIGN    6,FN$DTIM      ; Допустимое время
            ENTER     KNV            ; Поступление на конвейер
NEXT        ASSIGN    4,FN$ROUTE     ; Тип станка
            ASSIGN    5,FN$TIME      ; Время наработки
            ASSIGN    7,FN$GO        ; Время движения к станку
            ADVANCE   P7             ; Время движения по конвейеру
            QUEUE     P4             ; Очередь к станку
            ENTER     P4             ; Занятие станка
            DEPART    P4             ; Выход из очереди
            ADVANCE   P5             ; Обработка на станке
            LEAVE     P4             ; Освобождение станка
            ASSIGN    3+,1           ; Увеличение количества этапов
            LOOP      2,NEXT         ; Этапы – все? Нет – NEXT
            ASSIGN    7,FN$EXIT_     ; Время выхода с участка
            ADVANCE   P7             ; Выход с участка
            LEAVE     KNV            ; Выход с конвейера
            TABULATE  P1             ; Сбор статистики
            TEST L    P6,M1,DDD      ; Штрафовать? Нет – DDD
            SAVEVALUE      SHTRAF+,FN$MSHT  ; Штраф
            SAVEVALUE      DOHOD-,FN$MSHT   ; Определение дохода
DDD         SAVEVALUE      DOHOD+,FN$MDOH   ; Определение дохода
            SAVEVALUE      DOHOD-,FN$CEB    ; Определение дохода
            SAVEVALUE      CEBECT+,FN$CEB   ; Определение себестоимости
*
            TERMINATE
* 2-й сегмент модели
            GENERATE 4800                   ; Таймер модели
            TERMINATE      1
```

За один эксперимент с моделью невозможно определить оптимальную структуру производственного участка. Эта процедура неминуемо оказывается итеративной и требует генерации и проверки множества гипотез. Для каждой гипотезы следует провести несколько экспериментов с моделью, чтобы получить результаты с нужной точностью.

Перед проведением экспериментов множество гипотез упорядочивают по величине увеличения материальных затрат на внедрение гипотезы. Для данного примера упорядоченный список гипотез такой: начальная структура участка; введение нового режима работы оборудования; введение приоритетов в очередях к станкам всех типов; введение приоритетов и нового режима работы; введение новых дополнительных станков. Наилучшую гипотезу следует выбирать с учетом того, что загрузка оборудования не должна превышать критического значения 75-85%.

Если коэффициенты загрузки станков превышают критическое значение, то нужно ввести дополнительные станки этого типа. Если введенные станки будут загружены на 60-70%, то это является условием быстрой окупаемости.

В результате решения данной задачи наилучшей оказалась гипотеза перехода на новый режим работы оборудования производственного участка и введение приоритетов при обработке деталей на станках. На рис. 10.2 показаны приблизительные графики величин доходов для начальной структуры участка и для улучшенной структуры при моделировании 11 дней работы участка. В табл. 10.5 приведены варианты задач, которые можно использовать для выполнения самостоятельных работ с учетом приведенной программы.

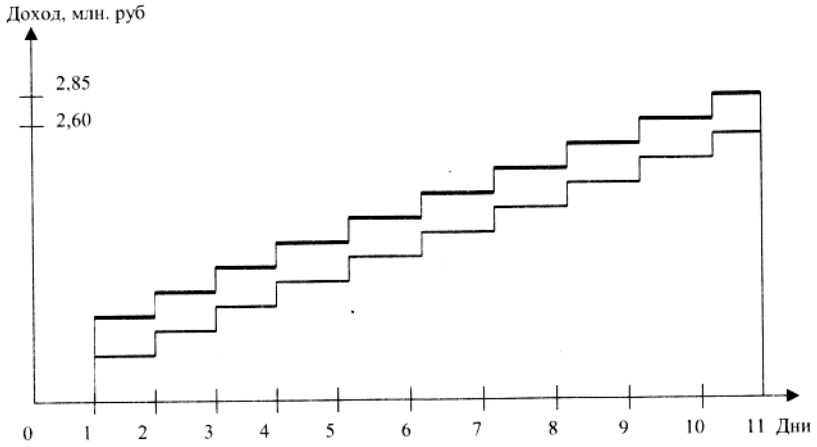


Рис. 10.1

При выполнении самостоятельной работы с приведенной программой необходимо дать ответы на следующие вопросы:

1. Какие изменения надо внести в модель для производственного участка, чтобы задать новые типы деталей с заданными маршрутами их движения?

2. Какие изменения надо внести в модель для производственного участка, чтобы задать новый станок – строгальный?

3. Предложите новые гипотезы относительно улучшения технологического процесса на участке.

Таблица 10.5

Вариант	***Средний интервал времени между поступлениями работ на участок, мин***	***Разделение работ по типам, %***			
		первому	***второму***	***третьему***	***четвертому***
1	180	30	15	40	15
2	220	20	30	20	30
3	190	15	30	55	10
4	170	25	20	35	20
5	210	10	50	30	10
6	165	40	15	25	20
7	240	20	25	30	25
8	175	35	35	15	15
9	185	20	35	25	20
10	220	40	15	30	15
11	210	15	55	10	20
12	230	25	35	20	20
13	180	35	35	15	15
14	200	20	30	15	35
15	190	30	10	50	20
16	240	40	25	25	10
17	225	15	35	30	20
18	185	30	20	30	20
19	165	20	40	10	30
20	170	35	15	25	25
22	210	25	30	40	5
23	220	40	10	30	20
24	190	20	40	20	40
25	230	25	25	20	30
26	200	15	35	35	15

27	175	20	10	40	30
28	210	30	30	30	10
30	225	40	20	10	30
31	230	20	10	25	30
32	200	35	20	40	15
33	175	25	30	30	30

10.2. Моделирование технологического процесса ремонта и замены оборудования

Рассмотрим пример построения модели средствами ISS 2000, (см. параграф 8.8). В соответствии с целью исследования необходимо определить наилучший вариант технологического процесса ремонта и замены оборудования для обеспечения минимальной себестоимости производства. Варианты данных для моделирования приведены в табл. 10.6.

Таблица 10.6

Вариант	***L***	***T***	$A_1 \pm B_1$	$A_2 + B_2$	$A_3 \pm B_3$	***H***	***W***	***S***	***Q***
1	50	160	2±1	30±10	45±5	360	*7,75*	650	120
2	40	120	3±1	25±7	40+3	300	7,25	500	140
3	44	135	|3±2	22±12	38±7	365	7,35	480	130
4	48	125	2+1	18±6	35±2	310	7,7	750	155
5	55	165	3±2	25±7	47±4	320	7,05	540	125
6	50	140	3±1	33+5	48+3	340	8,0	580	160
7	45	145	2±1	31±8	40±3	315	7,55	630	110
8	50	150	4+2	24±6	37±5	315	7,45	460	100
9	45	135	2±1	28±7	40±5	310	7,35	520	112
10	60	145	3±1	20+10	51±8	320	7,45	710	132
11	48	125	3±2	27+4	43+6	330	7,15	670	144
12	52	130	4±2	23±5	40+4	360"	7,5	540	150
13	44	140	2±1	31+6	41±5	365	7,55	510	123
14	38	150	3±1	33±4	39+4	312	7,65	480	135
15	60	160	3±2	30+8	50±5	322	7,55	550	140
16	55	155	12+1	26±7	41+5	300	7,45	600	130
17	50	145	4±2	25±8	43±5	315	7,35	650	110
18	45	155	3±1	24±7	47+5	314	7,25	530	130
19	47	128	2+1	31±1	44±5	310	7,5	560	140
20	52	135	3±1	22±8	48±7	350	7,75	620	120
21	35	145	4±2	30±5	50±3	324	7,55	580	145
22	40	140	3+1	33±6	43±5	314	7,65	480	125
23	30	130	2±1	24+7	42±6	334	7,45	500	120
24	35	155	3+2	28±4	41±3	285	7,3	550	143
25	45	160	2±1	22+4	40+6	320	7,35	560	132
26	53	136	3±2	31±10	38±9	310	8,05	610	147
27	58	128	2±1	28±9	45+8	300	7,75	584	157
28	48	150	3+2	35+6	40+10	315	7,85	615	166
29	52	138	2±1	30±9	35±6	310	8,1	600	160
30	55	140	2±1	28±5	36±8	300	7,9	585	155
31	60	150	3±1	22±8	45±3	315	8,0	500	140
32	55	220	2±1	30±5	50±6	350	9,5	550	120
33	60	200	3±2	33±6	55±9	285	7,5	560	145

Имитационная модель в виде стохастической сети СМО построена средствами ИСИМ. Последовательность действий для организации поиска наилучшего варианта технологического процесса:

- рассчитать среднее время ремонта етанков *R* с помощью операционного анализа сетей СМО;
- сформулировать гипотезу о потенциально узком месте системы и определить его;
- описать стратегию поиска решения задачи, определить необходимое количество арендованных станков и ремонтников для проведения моделирования;
- разработать программу проведения экспериментов, предварительно определив количество прогонов модели для каждой комбинации «количество рабочих – количество станков» с записью стоимости затрат в матрицу результата.

Используя процедуру ANOVA, провести анализ результатов моделирования и сделать выводы относительно наилучшего варианта технологического процесса ремонта и замены оборудования.

Стратегия поиска решения задачи. Для поиска наилучшего решения воспользуемся методом структурной оптимизации [16, 17]. Модель используется для оценки сочетания «нанимать – арендовать», которая минимизировала бы средние дневные затраты на производство.

При фиксированном количестве нанятых рабочих в мастерских средние дневные затраты будут изменяться в зависимости от количества арендованных станков. Эта зависимость имеет вид вогнутой вниз кривой. Аналогично при заданном количестве арендованных станков влияние количества нанятых рабочих на дневные затраты имеет тот же вид. Если вообразить рассматриваемую ситуацию в трехмерном пространстве: количество арендованных станков – количество рабочих – дневные затраты, то можно предположить, что поверхность дневных затрат будет также вогнутой вниз поверхностью и будет иметь одну точку минимума. Таким образом, поиск сочетания «количество нанятых – количество арендованных» является поиском этой точки.

Сравним между собою почасовую заработную плату рабочих $W(m_1+m_2+m_3)$, почасовую стоимость аренды одного станка *S* и убытки из-за нехватки одного станка $Q \cdot H$. Последний показатель значительно превышает два других. Это означает, что оптимальное соотношение надо искать среди тех значений, которые обуславливают минимальные затраты из-за простоев станков.

Рассмотрим «идеальную» систему, в которой станок, вышедший из строя, никогда не простаивает перед тем, как рабочий начнет ремонт. Найдем нижнюю оценку необходимого количества ремонтников. Каждый станок имеет средний цикл $T+R$ часов, который состоит из фазы работы (среднее время работы до выхода из строя *T)* и фазы ремонта (среднее время ремонта *R).*

Минимальное среднее время ремонта станков можно определить с помощью операционного анализа по коэффициентам посещаемости узлов сети V_j СМО, которая моделирует мастерскую, и среднего времени ремонта на каждом рабочем месте R_j.

$$R=\sum_j V_j R_j, \quad j=1,2,3.$$

В соответствии с формулой (2.6): $V_1 = 1$, $V_2 = q_{12} = 0{,}25$, $V_3 = g_{13} = 0{,}75$.

Пропускная способность мастерской определяется узким местом, то есть местом, где коэффициент загрузки ремонтников U_j приближается к единице. Потенциально узкое место определяют по формуле

$$V_d R_d = \max\{V_1R_1, V_2R_2, V_3R_3\}.$$

Необходимо сбалансировать систему, то есть добиться приблизительного равенства средних времен ремонта, поскольку время ремонта станков полностью определяется узким местом. Это можно сделать за счет увеличения количества ремонтников на рабочих местах с $i \neq i_0$, где i_0 – номер наименее загруженного рабочего места, которое определяется по формуле:

$$V_{i_0} R_{i_0} = \min\{V_1R_1, V_2R_2, V_3R_3\}.$$

Приблизительное равенство средних значений времени ремонтов приводит к выполнению равенства

$$\frac{V_1R_1}{m_1} \approx \frac{V_2R_2}{m_2} \approx \frac{V_3R_3}{m_3}, \qquad (10.1)$$

где $m_{i_0} = 1$.

Таким образом, пропускная способность мастерской будет сбалансирована, если коэффициенты загрузки рабочих мест в мастерской будут между собой приблизительно равными. Тогда коэффициент использования станков в такой идеальной системе

$$k = \frac{T}{T+R}.$$

По условиям задачи имеем L собственных станков. Для того, чтобы исключить затраты вследствие снижения объема производства, общее количество станков, которое используется в системе, должно равняться

$$M = \left]\frac{L}{k}\right[,$$

где выражение $]x[$ означает операцию округления **X** до ближайшего целого с избытком.

Пусть

$$L_r = M - L. \qquad (10.2)$$

Тогда при L работающих станках L_r. станков должно находиться в резерве. Однако, учитывая случайность времени отказа и времени ремонта, станки могут простаивать в тех местах, где потенциально возможно ожидание. Таким образом, учитывая заданный критерий затрат, необходимо арендовать больше, чем L_r станков.

Схема алгоритма поиска. Наилучшую комбинацию «количество арендованных станков L_r - количество рабочих $m_1+m_2+m_3$» можно найти таким образом:

1. Считаем, что элементы комбинации «количество арендованных станков L_r – количество рабочих $m_1+m_2+m_3$» определяются соотношениями (10.1) и (10.2).

2. Проводим серию экспериментов для комбинаций «количество арендованных станков L_r – количество рабочих $m_1+m_2+m_3$». После каждого прогона сохраняем в соответствующей матрице результатов коэффициенты загрузки U_j, (j = 0, 1, 2, 3) и величину затрат на производство.

3. После проведения экспериментов данные, полученные в серии прогонов после выполнения процедуры ANOVA, заносим в табл. 10.7. Приближение значения коэффициента загрузки станков k_0 к единице свидетельствует о том, что станки быстро возвращаются в рабочее состояние, то есть не простаивают, что свидетельствует о достаточном количестве отремонтированных и (или) арендованных станков.

Таблица 10.7

Комбинации	***Количество ремонтников***			***Количество станков Lr***	***Средние значения коэффициентов загрузки***				***Среднее значение стоимости производства***
	m1	***M2***	***m3***		***U0***	***U1***	***U2***	***U3***	

4. По результатам моделирования определяем ***узкое место*** системы. Выдвигаем гипотезу об увеличении количества ремонтников в этом узле (увеличиваемое значение может быть больше единицы). Если же узких мест нет, увеличивается количество станков Lr и для новой комбинации «количество арендованных станков i_r – количество рабочих $m_1+m_2+m_3$» возвращаемся к п. 2.

5. Прекращаем моделирование, если никакие изменения значений L_r, m_1, m_2, $m3$ не приводят к уменьшению среднего значения стоимости производства. Комбинация «количество арендованных станков L_r – количество рабочих $m_1+m_2+m_3$», которой отвечает наименьшее среднее значение затрат на производство, является решением задачи.

По результатам процедуры ANOVA делаем вывод о значимости расхождений полученных результатов, приводим значения критерия Фишера для наилучшего решения, анализируем построенный доверительный интервал.

Если получено несколько значений, близких к оптимальному значению функции затрат на производство, и эта функция пологая, необходимо увеличить количество прогонов в сериях соответствующих комбинаций «L_r - $m_1+m_2+m_3$», ***А*** потом для них выполнить процедуру ANOVA.

Приведенный алгоритм может быть реализован с помощью пользовательского эксперимента.

ПРИЛОЖЕНИЕ

Системные СЧА

RN_j – число, вычисляемое *j*-м генератором случайных чисел. Все генераторы выдают последовательность равномерно распределенных случайных чисел. Это число целое и изменяется от 0 до 999 включительно, кроме двух случаев его использования – в качестве аргумента функции или выражения переменной (VARIABLE, FVARIABLE). В этих случаях RN_j будет дробью от 0 до 0.999999;

C1 –текущее значение относительного (условного) времени. Автоматически изменяется программой и устанавливается в 0 управляющими операторами CLEAR или RESET;

AC1 – текущее значение абсолютного времени. Эта величина не меняется под действием управляющего оператора RESET и устанавливается в 0 лишь под действием оператора CLEAR;

TG1 – текущее значение счетчика завершений;

XN1 – номер активного сообщения;

Z1 – размер свободной оперативной памяти в битах,

M1 – время пребывания в модели транзакта, обрабатываемого программой в данный момент. Эта величина может изменяться блоком MARK;

PR – приоритет транзакта, обрабатываемого в данный момент. Эта величина может изменяться блоком PRIORITY. По умолчанию приоритет равен 0.

СЧА транзактов

Pj или **j* или *<имя>, или *$<имя> – значение параметра *j* текущего транзакта или значение параметра с именем <имя> текущего транзакта;

MP*j* – значение времени, равное разности относительного модельного времени и содержимого *j*-го параметра текущего транзакта;

MB*j* – флаг синхронизации: 1, если транзакт в блоке; принадлежит тому же семейству, что и текущий транзакт; 0 – в противном случае.

СЧА блоков:

N*j* — общее число транзактов, вошедших в блоку;

Wy – текущее число транзактов, находящихся в блоке/.

СЧА МКУ

S*j* – текущее значение содержимого многоканального устройства *j*. Содержимое многоканального устройства изменяется блоками ENTER и LEAVE;

R*j* – число свободных единиц многоканального устройства *j*. Эта величина изменяется блоками ENTER и LEAVE;

SRj – коэффициент использования многоканального устройства *j* в тысячных долях;

SA*j* – среднее содержимое многоканального устройства *j*;

SM*j* – максимальное содержимое многоканального устройства *j*;

SC*j* – общее число входов в многоканальное устройство *j*;

ST*j* – среднее время пребывания транзактов в многоканальном устройстве *j*;

SE*j* – флаг незанятости многоканального устройства *j*: 1 — свободно, 0 – занято;

SF*j*- флаг заполнения многоканального устройства *j*: 1 – заполнено, 0 – не заполнено;

SV*j* – флаг готовности многоканального устройства *j* к использованию: 1 — готово, 0 – не готово.

СЧА одноканальных устройств:

F*j* – текущее состояние устройства *j*. Эта величина равна 0, если устройство свободно, и 1 – во всех остальных случаях. Этот атрибут изменяется блоками SEIZE, RELEASE, PREEMPT и RETURN.;

FI*j* – флаг прерывания устройства: 1, если устройство находится в состоянии прерывания , 0 — в противном случае;

FR*j* – коэффициент использования устройства *j* в тысячных долях;
FV*j* – флаг готовности устройства к использованию: 1 – готово, 0 – в противном случае;
FC*j* – общее число входов в устройство *j*;
FT*j* – среднее время использования устройства транзактами.

СЧА очередей

Q*j* – длина очереди *j*;
QA*j* – средняя длина очереди *j*;
QM*j* – максимальная длина очереди *j*;
QC*j* – общее число входов в очередь *j*;
QZ*j* – число нулевых входов в очередь *j*;
QT*j* – среднее время пребывания транзактов в очереди *j* (включая нулевые входы);
QX*j* – среднее время пребывания транзактов в очереди *j* (без нулевых входов).

СЧА таблиц

TB*j* – вычисленное среднее таблицы *j*;
TQ*j* – общее число включений в таблицу *j*;
TD*j* – вычисленное среднеквадратичное отклонение для таблицы.

СЧА ячеек и матриц ячеек сохраняемых величин:

X*j* – содержимое ячейки *j*;
MX*j*(a,e) – содержимое элемента матрицы ячеек *j*, расположенного в строке **A** *и* столбце *b.*

СЧА вычислительных объектов

FN*j* – вычисленное значение функции *j*. От значения берется целая часть, за исключением тех случаев, когда это значение используется в качестве модификатора в блоках GENERATE, ADVANCE, ASSIGN или в качестве аргумента другой функции;
V*j* – вычисленное значение переменной *j*. При вычислении значения переменной с фиксированной запятой получается целое число. При вычислении значения переменной с плавающей запятой дробная часть конечного результата отбрасывается;
BVj – вычисленное значение булевой переменной.

СЧА списков и групп

GN*j*– текущее число членов в числовой группе
GT*j* – текущее число членов в группе транзактов с номерами *j*
CH*j* – текущее число транзактов в *j*-м списке пользователя;
CA*j* – среднее число транзактов в *j*-м списке пользователя;
CMj – максимальное число транзактов в *j*-м списке пользователя;
CC*j* – общее число транзактов в *j*-м списке пользователя;
CT*j* – среднее время пребывания транзакта в *j*-м списке пользователя;
LS*j* – состояние логического ключа *j*: 1 – включен , 0 – выключен.

СПИСОК ЛИТЕРАТУРЫ

1. *Томашевский В.Н., Жданова Е.Г.* Имитационное моделирование средствами системы GPSS/PC: Учеб. пособие. – К.: ІЗМН, НТТУ КПИ, 1998.-123с.

2. *Томашевский В.Н., Жданова Е.Г., Жолдаков А.А.* Решение практических задач методами компьютерного моделирования: Учеб. Пособие – К.: Изд-во "НАУ", 2001. – 268 с.

3. *Beasley J. E.* and *Whitchurch G.* O. R. education – a survey ofyoung O. R. workers. // *Journal of the Operational Research Society.* – 1984. – № 35. -P.281-288.

4. *Методы* построения имитационных систем / В.В. Литвинов, Т.П. Марянович – К.: Наук, думка, 1991. – 120 с.

5. *Томашевский В.Н., Павлишин В.А.* Формальный метод проектирования программного генератора имитационных моделей / Труды первой международной конференции с программирования "УкрПРОГ 98" - К.: Кибернетический центр НАН Украины, 1998.- С. 563-571.

6. *O'Keefe Robert* and *Roach Joan W.* Artificial Intelligence Approach to Simulation. *//Journal of the Operational Research Society.* – 1987. -№ 38. -P.713-722.

7. *Новиков О.А., Петухов С.И.* Прикладные вопросы теории массового обслуживания. – М.: Сов. радио, 1969. – 400 с.

8. *Клейнрок Л.* Теория массового обслуживания. – М.: Машиностроение, 1979.-432с.

9. *Литвин В.Г., Аладышев В.П., Винниченко А.И.* Анализ производительности мультипрограммных ЭВМ. – М.: Финансы и статистика, 1984.-159с.

10. *Шрайбер Т.Дж.* Моделирование на GPSS. – М.: Машиностроение, 1980.-593с.

11. *Прикладная статистика.* Основы моделирования и первичная обработка данных. Справочное изд. Г.А. Айвазян, И.С. Енкжов, Л.Д. Мешалкин. – М.: Финансы и статистика, 1983. – 471 с.

12. *Прицкер А.* Введение в имитационное моделирование и язык СЛАМ II – М.: Мир, 1987. – 646 с.

13. *Шеннон Р.* Имитационное моделирование систем – наука и искусство. – М.: Мир, 1978. – 418 с.

14. *Система* программного обеспечения для имитационного моделирования на GPSS/PC / Версия 2. – Калинин: Центрпрограммсистем, 1989. – 200с.

15. *Крэйн М., Лемуан О.* Введение в регенеративный метод анализа моделей. – М.: Наука, 1982. – 104 с.

16. Советов Б.Я. Яковлев С.А. Моделирование систем. Курсовое проектирование. – М.: Высш. шк., 1988. – 135 с.

17. Феррари Д. Оценка производительности вычислительных систем. -М.: Мир, 198l.-576с.

18. Томашевский В.Н. Имитационное моделирование систем и процессов: Учеб. пособие. – К.: 1СДО, "ВІПОЛ", 1994. – 124 с. (язык украинский).

19. GPSS World reference manual. Fourth Edition 2001. Copyright Minuteman Software. Holly Springs, NC, U.S.A. 2001.

20. GPSS World Tutorial Manual. Copyright Minuteman Software. Holly Springs,NC,U.S.A.2001.

21. Томашевский В.Н., Жданова Е.Г. Метод структурной оптимизации с использованием имитационной модели / МКИМ-202. Международная конференция с индуктивного моделирования – Т. 2 – Львов: Державный НДИ информационной инфраструктуры, 2002, с. 224-227 (язык украинский).

Срдержание