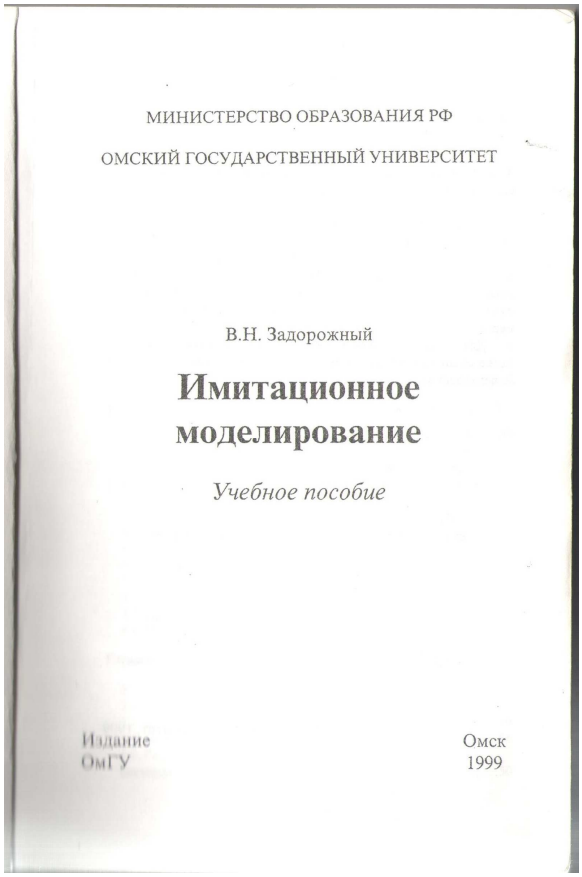
Предисловие автора

Цель, с которой я выкладываю данное пособие в Интернете, двоякая.

Во-первых, обнаружив, что оно «разобрано на цитаты» без должных ссылок, я понял, что более 15 лет назад поступил опрометчиво, выложив его черновик в виде doc-файлов на кафедральный сервер. Пособие разошлось по Интернету, и цитирующие его авторы, не зная реквизитов его официального издания, не смогли корректно на него сослаться. Судя по интенсивности цитирования пособия, ряд высказанных в нем положений и примеров до сих пор актуален, хотя используемая в нем версия языка GPSS давно устарела. Итак, выкладывая это пособие, я облегчаю возможность правильных ссылок на него.

Во-вторых, выяснилось, что некоторые уважаемые (судя по их регалиям) господа позволили себе издать попавший к ним черновик моего пособия, просто вставив в него свои имена в качестве авторов (см. http://www.rae.ru/monographs/39 или http://urss.ru/cgi-bin/db.pl?lang=Ru&blang=ru&page=Book&id=128178). Таким образом, выложив свое пособие, я даю этим господам возможность исправить их недостойный поступок. Для этого им достаточно переоформить опубликованное ими пособие просто как одну большую цитату из данного моего пособия.

Автор: Задорожный В.Н.



МИНИСТЕРСТВО ОБРАЗОВАНИЯ РФ

ОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

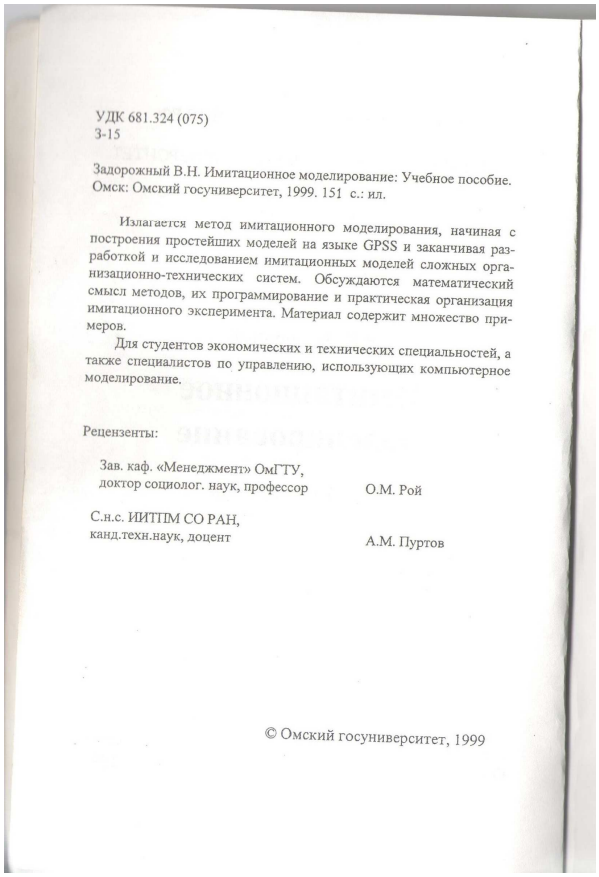
В.Н. Задорожный

Имитационное моделирование

Учебное пособие

Издание ОмГУ

Омск 1999



УДК 681.324 (075)
3-15

Задорожный В.Н. Имитационное моделирование: Учебное пособие. Омск: Омский госуниверситет, 1999. 151 с.: ил.

Излагается метод имитационного моделирования, начиная с построения простейших моделей на языке GPSS и заканчивая разработкой и исследованием имитационных моделей сложных организационно-технических систем. Обсуждаются математический смысл методов, их программирование и практическая организация имитационного эксперимента. Материал содержит множество примеров.

Для студентов экономических и технических специальностей, а также специалистов по управлению, использующих компьютерное моделирование.

Рецензенты:

Зав. каф. «Менеджмент» ОмГТУ, доктор социолог. наук, профессор О.М. Рой

С.н.с. ИИТПМ СО РАН, канд.техн.наук, доцент А.М. Пуртов

© Омский госуниверситет, 1999

Владимир Николаевич Задорожный

ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ

Учебное пособие

Редактор Л.Ф. Платоненко

Лицензия ЛР № 020380 от 29.01.97.
Подписано в печать 26.12.99. Формат бумаги 60x84 1/16.
Печ. л. 9,4. Уч.-изд. л. 10,9. Тираж 100 экз. Заказ 243.

Издательско-полиграфический отдел ОмГУ
644077, г. Омск-77, пр. Мира, 55а, госуниверситет

Оглавление

Введение.....6
Глава 1. Основные понятия.....8

1.1. Имитационное моделирование.....8
1.2. Типы имитационных моделей.....8
1.3. О различиях между моделями и реальными системами.....10
1.4. О различиях между моделями компьютерными и математическими.....12
1.5. О языках моделирования.....14

Глава 2. Простейшие модели на GPSS.....15

2.1. Основные объекты языка GPSS.....15
2.2. Пример задачи моделирования.....17
2.3. Модель на языке GPSS.....17
2.4. Сбор статистики об очереди клиентов.....19
2.5. Управление временем моделирования.....20
2.6. Модель парикмахерской с несколькими парикмахерами...22

Глава 3. Практическая работа на компьютере...24

3.1. Модель обслуживания с отказами.....24
3.2. Выполнение имитационного эксперимента.....25

Глава 4. Моделирование швейной фабрики.....32

4.1. Задача оптимизации производства.....32
4.2. Схема процесса и предварительный анализ.....32
4.3. Разработка моделирующей программы.....35
4.4. Планирование эксперимента.....37
4.5. Оптимизация управленческого решения.....39
4.6. Расчет экономического эффекта.....40
4.7. Другие применения модели.....41
4.8. Критический анализ показателей эффективности.....42

Глава 5. Задача моделирования конвейера.....45

5.1. Постановка задачи.....45
5.2. Схема процесса и предварительный анализ.....45
5.3. Имитационная модель.....46
5.4. Имитационный эксперимент.....48
5.5. Разработка мероприятий по совершенствованию системы.....50

3

5.6. Моделирование первого варианта мероприятий……..50
5.7. Моделирование второго варианта мероприятий……..51
5.8. Корректировка рекомендаций……52

Глава 6. Модель транспортного цеха……55

6.1. Постановка задачи……55
6.2. Схема процесса и предварительный анализ……55
6.3. Разработка модели……57
6.4. Планирование эксперимента……59
6.5. Имитационный эксперимент……59
6.6. Критический анализ результатов……60
6.7. Практические рекомендации……61

Глава 7. Описание языка GPSS\PC

7.1. О синтаксисе, семантике и прагматике языка GPSS\PC….63
7.2. Классы объектов языка GPSS\PC……64
7.3. Общеалгоритмические средства GPSS\PC……66
7.4. Применение стандартных числовых атрибутов……..67
7.5. Арифметические переменные……68
7.6. Булевские переменные……70
7.7. Функции……..72
7.8. Ячейки и матрицы ячеек……75
7.9. Задание начальных значений ячеек и матриц……..77
7.10. Объектно-ориентированные средства GPSS\PC……..78
7.11.Создание и уничтожение транзактов……79
7.12. Задержка транзактов в блоках ADVANSE……..82
7.13. Операции занятия и освобождения устройств……..83
7.14. Операции захвата и освобождения устройств……86
7.15. Операции блокирования и разблокирования устройств…89
7.16. Описание памятей и работа с ними……90
7.17. Операции с ключами……92
7.18. Блок GATE……93
7.19. Синхронизация транзактов……..94
7.20. Изменение параметров и стандартных числовых атрибутов транзактов……97
7.21. Управление маршрутами транзактов в модели……98
7.22. Сбор статистики с помощью очередей……..101
7.23. Сбор статистики с помощью таблиц……103
7.24. Стандартные числовые атрибуты (перечень)……..104
7.25. Блоки COUNT и SELECT……..107
7.26. Список операторов GPSS\PC……..108

4

Глава 8. Планирование имитационного эксперимента.....111

8.1. Параллельный и последовательный эксперимент.....111
8.2. Пояснения к выбору типа ИЭ.....112
8.3. Учет содержательного смысла задачи.....114

Глава 9. Задания для самостоятельного выполнения на GPSS\PC.....115

Глава 10. Расчет простейших систем массового обслуживания.....124

10.1. Одноканальная экспоненциальная СМО.....124
10.2. Многоканальная экспоненциальная СМО.....127
10.3. Разомкнутая экспоненциальная СеМО.....129
10.4. Расчет локальных характеристик СеМО.....131
10.5. Расчет системных характеристик разомкнутых экспоненциальных СеМО.....133

Глава 11. Применение календаря событий.....137

11.1 Предварительные пояснения.....137
11.2. Структура календаря событий.....138
11.3. Пример: разработка имитационной модели СМО.....141
11.4. Организация сбора статистики в модели.....144
11.5. Практические задания.....147

Литература.....150

5

Введение

Предлагаемое пособие соответствует программе дисциплины «Моделирование систем», входящей в типовые учебные планы специальностей «Автоматизированные системы обработки информации», «Информатика и вычислительная техника» и других, родственных им. В течение нескольких лет материал пособия апробирован автором также и в курсе «Информационные технологии в управлении», читаемом на экономических факультетах.

Здесь изложены методы и средства имитационного моделирования (ИМ), которое в последние десятилетия стало основным общепризнанным инструментом исследования на ЭВМ реальных сложных систем и процессов.

В гл. 1 разъясняются основные понятия ИМ. Во гл. 2 даются основные понятия языка GPSS, который наиболее широко распространен, по сравнению с другими языками имитационного моделирования, не только у нас в стране, но и в Европе, и в США. Основные понятия GPSS иллюстрируются на простом примере моделирования, который поэтапно превращается в реальную моделирующую программу. В гл. 3 с использованием полученной программы дается урок практической работы на компьютере, который должен быть выполнен с применением пакета программ GPSS\PC. Необходимые инструкции изложены по принципу «Key by key» (клавиша за клавишей).

В трех последующих (с 4 по 6) главах с помощью ИМ исследуются три конкретные системы: швейная фабрика, производственный конвейер и транспортный цех. В контексте исследования этих систем знание языка GPSS и методики моделирования естественным образом расширяется за счет привлечения необходимых дополнительных сведений. Во всех трех задачах основная линия исследования включает моделирование заданной системы, анализ качества ее функционирования и поиск решений по усовершенствованию системы. При этом найденные варианты решений также моделируются с целью определения их эффективности.

Рассмотренные в гл. 3-6 примеры моделирования можно использовать в качестве лабораторных работ по ИМ, так как все этапы моделирования описаны здесь достаточно подробно, включая выполнение имитационных экспериментов на компьютере.

Последовательность этапов моделирования любой конкретной системы, используемая и рекомендуемая в пособии, включает:
– постановку задачи;
– составление схемы основных причинно-следственных связей в системе;
– предварительный анализ системы;
– разработку моделирующей программы;
– планирование эксперимента;
– Анализ результатов эксперимента и разработку рекомендаций по совершенствованию системы;
– моделирование усовершенствованной системы и определение эффективности разработанных рекомендаций.

В гл. 7 дается описание языка GPSS, гл. 8 посвящена особенностям планирования имитационных экспериментов. В гл. 9 приводятся задачи для самостоятельного решения на GPSS\PC. Эти задачи могут использоваться как варианты заданий к курсовым работам по моделированию.

В гл. 10 излагаются методы и приводятся формулы для аналитического расчета некоторых характеристик СМО (систем массового обслуживания) и сетей СМО. Эти методы и формулы бывают весьма полезны на этапе предварительного анализа систем. Например, с их помощью нередко удается рассчитать или оценить коэффициенты загрузки ресурсов, что дает возможность правильно конкретизировать задачу исследования.

В гл. 11 рассматриваются принципы ИМ систем с помощью обычных языков программирования, таких, как Паскаль, Си и др. Естественно, что программы на таких языках получаются в десятки раз длиннее, чем на GPSS. Однако с их помощью можно моделировать более широкий класс систем, разрабатывать собственные сложные моделирующие пакеты, игровые, учебные, тренажерные комплексы и другие программы, в которых необходимо «сорганизовать» множество параллельных процессов. Рассматриваемые принципы и методы предназначены для студентов «компьютерных» специальностей. В конце главы приводятся задания для самостоятельного решения.

ГЛАВА 1. ОСНОВНЫЕ ПОНЯТИЯ

1.1. Имитационное моделирование

Имитационное моделирование (ИМ) применяется для исследования и проектирования таких сложных систем и процессов, как предприятия, информационные сети, мировые динамики в экономике или экологии и т.д.

Имитационная модель системы – это программа, в которой определяются все наиболее существенные элементы и связи в системе и задаются начальные значения параметров, соответствующие некоторому «нулевому» моменту времени, а все последующие изменения, происходящие в системе по закону причин и следствий, вычисляются на ЭВМ автоматически при выполнении программы.

Такой метод моделирования не требует составления уравнений и, тем более, не требует их решения. При этом он позволяет отображать и исследовать поведение системы с любой детальностью и точностью.

Выполнение имитационной модели называется имитационным экспериментом (ИЭ). В ходе ИЭ компьютер имитирует функционирование системы и вычисляет характеристики свойств, проявляемых системой.

ИЭ подобен натурному эксперименту. Однако он позволяет, в отличие от натурного метода, экспериментировать с системами, которых еще или уже нет, позволяет предсказывать поведение существующих систем в будущем, изучать их поведение в чрезвычайных условиях. Он дешевле и быстрее натурных экспериментов.

1.2. Типы имитационных моделей

По характеру возможных изменений переменных величин модели подразделяются на непрерывные модели и дискретные.

В непрерывных моделях величины представляют собой непрерывные функции времени (рис.1). В соответствии с этим продвижение во времени (пересчет значений переменных величин в ходе модельного времени), осуществляется в имитационной модели по принципу «малых Δt», т.е.

следующее состояние системы определяется по ее предыдущему состоянию с малым промежутком времени между этими состояниями.

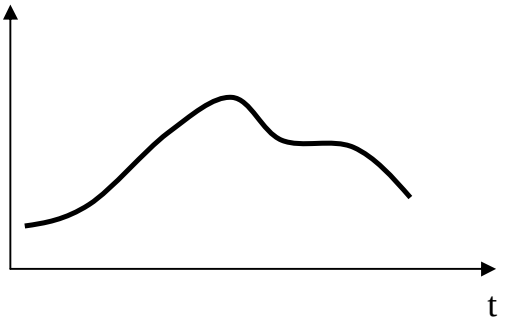


Рис.1

В дискретных моделях любые изменения происходят мгновенно, скачкообразно (рис.2), и между моментами изменений состояния элементов остаются постоянными. Изменения в дискретных моделях называют событиями.

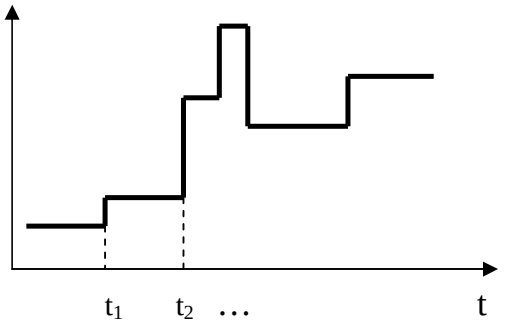


Рис.2

Продвижение в модельном времени в ходе ИЭ осуществляется по принципу «от события к событию», т.е. из нулевого момента времени модель перемещается сразу к моменту t_1, из него – к моменту t_2 и т.д.

Реальные системы не бывают непрерывными или дискретными. Просто для одних систем удобнее применять непрерывные модели, для других – дискретные.

По характеру проявления причинно-следственных связей имитационные модели подразделяются на детерминированные и стохастические, т.е. вероятностные.

Практически все языки моделирования имеют датчики случайных чисел и позволяют моделировать как детерминированные, так и стохастические процессы. Стохастичность модели не отражается на языке моделирования, но сказывается на организации ИЭ. В случае стохастических моделей приходится выполнять ИЭ многократно (или продолжать его достаточно долго), чтобы объем накопленных данных обеспечивал необходимую точность статистических оценок.

1.3. О различиях между моделями и реальными системами

Рассмотрим вначале соотношение между математическими представлениями о системах и реальными системами.

Представления о дискретности и непрерывности, так же как и представления о детерминизме и случайности, выработаны в рамках математики. И объекты, которые обладают названными свойствами, в строгом смысле слова встречаются только в математике. Значит, когда мы говорим, что некоторая система является дискретной, то тем самым уже имеем в виду не реальную систему, существующую в физическом мире, а некую математическую модель.

Реальные системы, объекты или процессы не являются детерминированными или непрерывными, или, скажем, непрерывно-детерминированными. Физики, например, долго выясняли вопрос о том, какое строение имеет материя, – в частности, электромагнитное поле, – корпускулярное или волновое. В конце – концов, выяснилось, что элементарные «частицы» могут в одних отношениях рассматриваться как частицы, т.е. корпускулы, а в других – как волны, но сами по себе «частицы» суть и не волны, и не корпускулы.

В то же время любой физический объект или процесс мы можем описывать и моделировать как непрерывный или как дискретный. И какой бы вариант мы ни избрали, мы можем достичь любой точности описания.

Например, живую речь человека можно описать в виде текста, т.е. дискретной моделью. Можно записать речь как непрерывную звуковую волну, т.е. как непрерывную функцию времени. Затем можно эту же звуковую волну

оцифровать, т. е. вновь представить дискретной моделью, и такая модель будет не менее точной, чем непрерывная.

Точно также реальные физические системы не являются ни случайными, ни детерминированными. Случайные величины и случайные события, - в точном смысле этих понятий, – есть объекты сугубо математические. Реальные события и величины могут лишь более или менее походить на случайные или детерминированные, но никогда не являются ни теми, ни другими.

И опять – таки каждое из этих двух противоположных понятий само по себе достаточно универсально, чтобы описывать реальные системы с любой степенью точности. Можно построить как угодно точную модель заданного объекта, которая будет чисто детерминированной. И можно построить точную модель этого же объекта, которая будет чисто вероятностной.

Например, при выбрасывании кубика мы получаем число очков от 1 до 6, которое можно рассматривать как случайную величину. Тем не менее, имеются также дифференциальные уравнения (их составил Яглом), которые позволяют точно определять результат выбрасывания кубика по начальным условиям броска, заданным с конечной погрешностью.

То же самое имеет место для рулетки и т.п. «случайных» объектов.

В природе нет ни случайности, ни детерминизма. Случайность и детерминизм – это математические абстракции, упрощения. Реальные процессы не случайные, не детерминированные, и не комбинированные из случайностей и детерминизмов.

Лаплас, например, исходил из того, что будущее Вселенной всецело определяется ее текущим состоянием, т.к. имеющиеся расположения и скорости движения всех ее частиц, подчиненные известным уравнениям теоретической механики, предопределяют расположение и скорости частиц на любой момент в будущем.

Но даже если настоящее в каком-то смысле предопределяет будущее, то это не детерминированное предопределение. Ведь в настоящем состоянии Вселенной находитесь, например, и Вы, уважаемый читатель, и ваши мысли и намерения также влияют на будущее, как и намерения многих миллионов существ. Следовательно, во Вселенной действует свободная воля как одна из реально наличествующих сил, и ее действие принципиально не может быть отождествлено ни с действием слепой случайности, ни с механическим причинно-следственным детерминизмом.

Не следует также забывать и то, что математические модели позволяют описывать и изучать только формальную сторону реальных объектов и явлений.

Грубо говоря, математические модели отражают содержание реальности примерно так же, как силуэт тени человека отражает самого человека.

Таково соотношение между математическими объектами и реальными. Математические модели всегда отражают лишь некоторый тонкий слой отношений, имеющихся в реальном объекте.

С другой стороны, именно благодаря этому «снятию» тонкого слоя отношений и «погружению» его в точный язык математики мы добиваемся грандиозного успеха в познании природы и овладении ее силами в форме машин и технологий. Ведь если модель отражает тончайший слой именно тех отношений, которые нас интересуют, то тем самым она концентрирует наше внимание, мышление, понимание, общение и практическую деятельность на самом существенном, бесконечно увеличивая этим эффективную «плотность мышления» и «пробивную силу» знания.

1.4. О различиях между моделями компьютерными и математическими

Рассмотрим теперь, как соотносятся модели математические и компьютерные.

При моделировании реальных систем мы вначале составляем некоторое представление о реальной системе, достаточно точное, чтобы можно было что-либо утверждать о системе доказательно. Это значит, что мы формируем математическую (в широком смысле слова) модель системы. Затем эту математическую модель мы превращаем в модель компьютерную. Следовательно, математические представления играют при моделировании роль своеобразного «интерфейса» между компьютерными моделями и реальными системами.

Но мы должны понимать, что объекты, реализованные в компьютерной программе, только лишь похожи на соответствующие математические объекты, но не идентичны им. Например, прямая линия на экране дисплея не есть то же самое, что прямая линия в математике. В математике прямая не имеет толщины. А на экране компьютера прямая не может не иметь определенную толщину, иначе она не была бы видна.

Это обычная компьютерная практика – моделировать математические объекты с известными свойствами посредством физических (компьютерных) объектов, которые сами этими свойствами не обладают. При этом, хотя точность моделирования математических объектов компьютерными может быть очень высокой, это все-таки не избавляет нас от необходимости

понимания замаскированных различий между исходными математическими объектами и их компьютерными образами.

Рассмотрим еще один пример. Вещественные величины в математике непрерывны, а в компьютере дискретны, т.к. длина разрядной сетки у компьютерных чисел принципиально ограничена. Точность представления конкретных вещественных чисел при этом может быть сколь угодно большой. Так, число π можно представить с двумя знаками после запятой, с десятью, с сотней знаков и т.д. Но такое свойство этого числа, как его иррациональность и трансцендентность, ни одно из компьютерных приближений «ухватить» не может. Это пример различия хотя и тонкого, но принципиального.

Другой подобный нюанс возникает при численном дифференцировании функций. В математике мы пользуемся для вычисления производной dy/dx пределом отношения $\Delta y/\Delta x$ при Δx, стремящемся к нулю. На компьютере приращения Δy и Δx можно брать весьма малыми, обеспечивая все более высокую точность вычисления производной. Однако, в отличие от математических законов, уменьшение приращений в компьютере будет приводить к повышению точности результата лишь до некоторого момента, после которого дальнейшее уменьшение приращений будет вызывать уже рост ошибки и закончится либо тем, что приближенное значение производной превратится в нуль, либо тем, что компьютер сообщит об ошибке «деление на нуль». Причина – ограниченность разрядной сетки, т.е., в конечном счете, – отличие компьютерных вещественных чисел от математических.

Из перечисленных примеров может сложится впечатление, что *некоторые* математические объекты и операции не реализуются компьютером в их совершенстве и полноте, но таких объектов как будто мало, и их нужно специально отыскивать. Однако такое впечатление, если бы оно возникло, было бы ошибочным. В действительности, пожалуй, нет таких математических объектов и операций, которые были воплощены в компьютере точно. Так, самая очевидная и простая операция – сложение целых чисел – может быть выполнена компьютером лишь для некоторых целых чисел. Если числа достаточно большие, то результат сложения переполнит разрядную сетку, т.е. компьютер не сможет сложить двух чисел. Выходит, что из бесконечного множества целых чисел «лишь некоторые», лишь конечное их множество, доступны компьютеру для сложения.

Подобные несовершенства в компьютерных представлениях математических объектов неисчислимы. Они приводят к тому, что правильное применение компьютера для решения серьезных задач может гарантироваться только совместными усилиями профессионалов высокой квалификации в области вычислительной техники, в области математики и в той предметной области, в которой компьютер применяется. Это соответствует специфике

тройственного отношения «реальная система – математическая модель – компьютерная модель».

1.5. О языках моделирования

Наиболее распространенным языком для имитационного моделирования непрерывных систем является язык DYNAMO, для дискретных – язык GPSS.

Система моделирования GPSS (General Purpose Simulation System) разработана фирмой IBM (International Business Machines). Основные составные части системы GPSS – это язык GPSS и соответствующий ему языковой процессор. Существует довольно много версий языка GPSS. Из них наиболее широко распространенными и доступными версиями являются GPSS/360 [1–3], GPSS-V [4] и GPSS\PC.

Пользователь, изучивший язык GPSS/360, с легкостью может перейти к применению любой другой версии GPSS. Язык GPSS получил наиболее широкое распространение по сравнению с другими языками моделирования. Он включен в институтские учебные курсы по моделированию систем у нас в стране и изучается в аналогичных курсах во многих колледжах и университетах США и других стран.

В последнее время фирмой Wolverine Software разработана новейшая версия языка GPSS, называемая GPSS/H+PROOF. В этой версии добавлены графические средства манипулирования с блок-схемами, возможность использования анимации и гибкий интерфейс связи с языком программирования C++ [5].

Язык DYNAMO проще, чем GPSS. Язык СЛАМ объединяет в одной оболочке языки GPSS и DYNAMO, т.е. позволяет строить непрерывно-дискретные модели систем и «имеет практически неограниченные возможности» [5].

ГЛАВА 2. ПРОСТЕЙШИЕ МОДЕЛИ НА GPSS

2.1. Основные объекты языка GPSS

Объекты языка GPSS – это наиболее простые математические модели, входящие в состав языка, с помощью которых можно конструировать более сложные модели, модели сложных реальных систем. Из этих объектов можно составлять, например, модели предприятий, модели транспортных систем города, модели сложных организационно-технических систем, таких как информационно-вычислительные сети, авиационные перевозки, торгово-финансовые комплексы и т.д.

Благодаря тому, что объекты GPSS отражают лишь определенный вид отношений в реальных системах, число основных объектов GPSS невелико.

К ним относятся блоки, транзакты, устройства, памяти и очереди.

Блоки представляют собой пространственные части или узлы реальных систем. Оказывается, что практически любую сложную систему можно разделить на части, каждая из которых функционирует по одному из типовых алгоритмов. Такие типовые части систем и представлены блоками GPSS. Выполняя пространственное разбиение реальных систем на блоки, следует иметь в виду не физическое пространство, в котором система существует как физическая данность, а функциональное, которое определяется организацией взаимодействия составных частей и элементов системы. Например, на предприятии составными частями, узлами функционального пространства являются места обработки и хранения: склады, транспортные средства, станки, сборочные конвейеры и т.д.

Транзакты – это динамические объекты, которые перемещаются в модели и вступают во взаимодействие с другими объектами. Число транзактов может быть любым, транзакты могут порождаться, размножаться, склеиваться и уничтожаться.

При моделировании дорожной сети города транзакты могут соответствовать автомобилям, при моделировании производственного процесса – изделиям, при моделировании финансовых потоков – деньгам и товарам.

При моделировании информационно-вычислительных систем транзактами обычно представляют программы и данные.

В самом абстрактном применении транзакт соответствует причинно-следственной связи, т.е. передаче некоторого взаимодействия.

В блоках могут выполняться такие операции, как создание транзактов (блок GENERATE), задержка транзактов на заданное время (блок ADVANCE), занятие транзактом некоторого устройства (блок SEIZE) и т.д.

Устройство – это объект, который может находиться в одном из двух состояний: *свободно* или *занято*. В каждый момент времени устройство может быть занято только одним транзактом. Если транзакт пытается занять уже занятое устройство, он *помещается в очередь*, т.е. переводится в состояние ожидания.

Когда занятое устройство освобождается, то проверяется наличие очереди транзактов к нему. Если такая очередь имеется, то первый из ожидающих транзактов извлекается из очереди и занимает устройство. Все эти операции выполняются в блоке SEIZE автоматически, т.е. их не нужно программировать. Кроме того, блок SEIZE автоматически собирает статистику об устройстве, в которую входит, например, *коэффициент загрузки* устройства, т.е. доля времени, в течение которой устройство было занято (работало) по отношению ко всему времени моделирования. Коэффициент загрузки можно определить и другими словами: он равен среднему числу транзактов, занимающих устройство. Очевидно, что коэффициент загрузки устройства всегда находится в пределах между 0 и1. Эту величину называют также *коэффициентом использования* устройства.

Память – это объект, имеющий заданную емкость. Емкость выражается целым числом единиц и описывается в модели *картой* STORAGE. Карты (от слова перфокарта) не являются блоками, т.е. транзакты в них не входят. Карты просто описывают перед началом моделирования те или иные объекты, которые будут входить в состав модели. Внешне карты, как и блоки, имеют вид программной строки и отличаются от блоков только по своему наименованию.

Память в отличие от устройства может быть занята одновременно несколькими транзактами. Когда транзакт пытается занять некоторое число единиц памяти, выполняется проверка, имеется ли в наличии достаточно свободных единиц. Если имеется, то транзакт занимает их. В противном случае он задерживается в очереди. Когда какой-либо транзакт освобождает несколько единиц памяти, выполняется проверка, есть ли транзакты в очереди к памяти. Если такие транзакты имеются, то первый из них пытается занять нужное ему число единиц, т.е. либо входит в память, либо продолжает оставаться в очереди (при недостаточном месте в памяти). Независимо от исхода этой проверки после нее выполняется проверка возможности войти в память второму транзакту из очереди, т.к. для второго транзакта места может оказаться достаточно. Далее проверки продолжаются до конца очереди. Тем самым реализуется рациональная дисциплина использования памяти. Эти операции выполняются в блоке ENTER. В нем также собирается статистика об использовании памяти.

Транзакт может освобождать иное число единиц памяти, чем занимал. В отличие от устройства, транзакт может освобождать память, которую не занимал.

Устройствами можно моделировать такие объекты реальных систем, как канал связи, процессор ЭВМ, функциональное место в банке (кассир, контролер и т.п.), станок, рабочего, продавца и т.д.

Память может изображать автостоянку, склад, память компьютера и т.п.

2.2. Пример задачи моделирования парикмахерской

В парикмахерскую с одним парикмахером приходят клиенты через 20 ± 10 минут друг за другом. Время стрижки одного клиента составляет 19 ± 5 мин. Требуется определить среднюю длину очереди клиентов и среднее время ожидания клиентами начала обслуживания.

Приведенная постановка задачи позволяет схематически изобразить процесс (функциональное пространство системы) в виде рис.3.

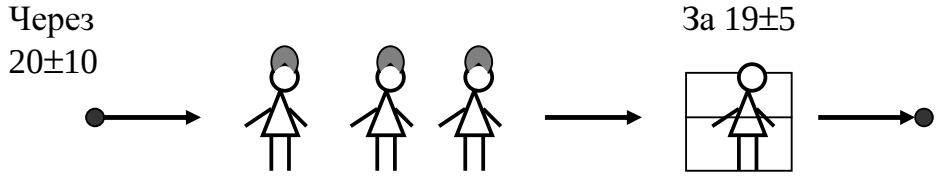


Рис. 3

3. Модель на языке GPSS

В соответствии со схематическим изображением функционального пространства парикмахерской (рис.3), ее модель на языке GPSS может быть написана следующим образом:

```
10          GENERATE        20,10
20          SEIZE           1
30          ADVANCE         19,5
40          RELEASE         1
50        TERMINATE
```

Здесь левая колонка – это *номера строк* модели: произвольные положительные числа в порядке возрастания (не обязательно целые).

Блок GENERATE порождает транзакты через каждые 20 ± 10 *единиц времени* (в данном примере мы считаем единицу времени минутой). Число 20 в первом операнде (*в поле A*) указывает интервал модельного времени, через который генерируются транзакты. В *поле B* записано число 10, которое задает *модификатор* интервала, т.е. интервал получается как случайное число в промежутке от 20 – 10 = 10 до 20 + 10 = 30.

Случайное значение 20 ± 10 математически интерпретируется как равномерно распределенная в интервале (10,30) случайная величина. На GPSS она реализуется при этом как целое число, т.к. *время в GPSS всегда выражается в целых числах.*

Таким образом, первый блок модели выдает через случайные интервалы времени транзакты, которые изображают приходящих в парикмахерскую клиентов.

Блок SEIZE 1, в который поступают транзакты из блока GENERATE, выполняет операцию занятия транзактами устройства номер 1. Устройство соответствует здесь креслу парикмахера или самому парикмахеру. Транзакты, появляющиеся в блоке GENERATE в моменты, когда устройство занято, остаются в этом блоке *в очереди* к устройству.

Блок ADVANCE 19, 5 задерживает транзакт, который занял устройство, на 19 ± 5 единиц времени, моделируя тем самым задержку клиента на время его обслуживания.

По истечении этого времени транзакт переходит в блок RELEASE 1, в котором выполняется освобождение устройства номер 1, и далее поступает в блок TERMINATE, в котором транзакты уничтожаются. Конечно, это не означает, что клиенты после стрижки тоже уничтожаются, просто клиент уходит из системы, значит, транзакт, моделирующий его, больше нам не нужен. Мы уничтожаем транзакт, чтобы не нужно было описывать его дальнейшее движение и чтобы освободить занимаемую им память компьютера.

Заметим, что в тот момент, когда один транзакт освобождает устройство, другой транзакт, стоящий в очереди, занимает это устройство. Оба действия выполняются в один и тот же момент модельного времени. Когда один транзакт находится в блоке ADVANCE, другие транзакты время от времени появляются

в блоке GENERATE и становятся в очередь к устройству. Следовательно, в модели одновременно в разных ее местах движутся несколько транзактов, выполняя те или иные операции, и могут влиять друг на друга и на другие объекты модели. *Параллельное выполнение нескольких процессов в одной программе* – существенное отличие языка GPSS от обычных алгоритмических языков, языков программирования. Это отличие делает языки имитационного моделирования мощным средством описания реальных систем, т.к. в реальных системах разные процессы в разных частях системы развиваются одновременно и при этом взаимодействуют между собой.

2.4. Сбор статистики об очереди клиентов

Чтобы при выполнении модели GPSS собирал статистику об очереди транзактов (клиентов), нужно включить в модель еще два блока – QUEUE (точка входа в очередь) и DEPART (точка выхода). Мы можем расставлять эти точки в своих моделях в принципе произвольным образом. Это зависит от того, о каком участке системы нам нужна статистика по движению через него потока транзактов.

В данной модели парикмахерской эти точки следует выбирать так:

```
    10          GENERATE        20,10
    15          QUEUE           1          ;точка входа в очередь номер 1
    20          SEIZE           1
    25          DEPART          1          ;точка выхода из очереди номер
1
    30          ADVANCE         19,5
    40          RELEASE         1
    50          TERMINATE
```

Добавленные в предыдущий вариант модели блоки выделены здесь жирным шрифтом. При такой расстановке блоков QUEUE и DEPART получается, что транзакт входит в очередь в момент появления его в системе, а выходит из очереди в момент, когда ему удалось занять устройство, т.е. пройти блок SEIZE. Следовательно, очередь 1 будет собирать статистику именно об очереди клиентов к парикмахеру, как она изображена на рис.3. И в результате

выполнения модели мы узнаем ответ на вопрос, поставленный в задаче: найти среднюю длину очереди клиентов и среднее время ожидания клиентами начала обслуживания.

Но можно расставить блоки QUEUE и DEPART иначе:

```
10          GENERATE        20,10
 15          QUEUE           1          ;точка входа в очередь номер 1
20          SEIZE           1
30          ADVANCE         19,5
40          RELEASE         1
 45          DEPART          1          ;точка выхода из очереди номер 1
50          TERMINATE
```

В этом случае статистика по очереди номер 1 будет соответствовать числу всех клиентов в парикмахерской вообще, включая клиента, обслуживаемого парикмахером. Таким образом, очереди, по которым мы можем собирать статистику, не обязательно должны совпадать с теми очередями, которые создаются транзактами, ожидающими освобождения устройств и памятей.

2.5. Управление временем моделирования

После того, как модель введена, она запускается на выполнение *командой*

START <n>

Команды вводятся с клавиатуры компьютера без номера строки и выполняются системой GPSS немедленно. Этим они отличаются от программных строк модели, т.е. от блоков и карт. Программные строки обязательно должны иметь номер строки и после ввода программные строки добавляются к модели, но не выполняются. Выполнение модели начинается только по команде START.

Целое число (обозначенное выше в виде <n>) в поле A карты START называется *счетчиком карты START*. С помощью счетчика карты START определяется момент, когда моделирование должно быть прекращено. Это осуществляется следующим образом.

В процессе выполнения модели каждый раз, когда какой-либо транзакт уничтожается в блоке TERMINATE, одновременно с этим GPSS вычитает из счетчика карты START число, *заданное в поле A этого блока TERMINATE*. Процесс моделирования прекращается в тот момент, когда значение счетчика карты START становится нулевым или отрицательным.

Одновременно с остановом модели начинается вывод всей собранной статистики на печать, точнее – вывод в текстовый файл с именем REPORT.GPS.

Такой механизм управления временем моделирования позволяет задавать останов модели по достижению заданного числового момента времени или же по заданному логическому условию, возникшему в модели.

Если мы введем один из предыдущих вариантов модели парикмахерской и запустим его на выполнение командой START 1 , то сможем наблюдать процесс обслуживания клиентов, смотреть значения тех или иных статистических показателей, но процесс моделирования не завершится, пока мы не прервем его сами. Ведь в рассмотренных версиях модели парикмахерской транзакты уничтожаются в блоке TERMINATE *с пустым полем A*, и поэтому счетчик карты START при уничтожении транзактов не уменьшается, т.е. никогда не станет нулевым или отрицательным.

Зададим для последней версии нашей модели парикмахерской время моделирования, равное одному году непрерывной работы:

```
10          GENERATE        20,10
 15          QUEUE           1          ;точка входа в очередь номер 1
20          SEIZE           1
 25          DEPART          1          ;точка выхода из очереди номер 1
30          ADVANCE         19,5
40          RELEASE         1
50         TERMINATE

**   Таймер модели  **

60          GENERATE        525600
70          TERMINATE       1
START   1
```

Карта, которая начинается звездочкой, рассматривается системой GPSS как *комментарий.*

Теперь в модель добавлены строки 60 и 70, которые обеспечивают выключение модели в момент модельного времени, равный 525600 единицам времени, т.е., в нашей интерпретации единиц времени, – 525600 минутам (или 1-му году).

Выполняются эти блоки следующим образом. Блок GENERATE 525600 генерирует транзакты с интервалом времени, заданным в поле A этого блока, т.е. через каждые 525600 единиц времени. Первый вышедший из него транзакт тут же уничтожается в блоке TERMINATE 1, и вычитается 1 из счетчика карты START. Поскольку счетчик был равен 1, то он в этот момент становится равным нулю и моделирование завершается.

Если мы запустим эту модель командой START 2, то моделирование прекратится только в момент выхода второго транзакта из блока GENERATE 525600, – т.е. будет выполнено моделирование двух лет работы парикмахерской.

Такую пару блоков, которая предназначена для останова модели по времени, как в конце приведенной здесь модели, называют *таймер модели.*

Таймер можно записывать разными способами. Например, представленный выше таймер модели эквивалентен по времени останова такому таймеру.

```
60      GENERATE        5256
70      TERMINATE       1
START   100
```

2.6. Модель парикмахерской с несколькими парикмахерами

Предположим, что клиенты появляются в парикмахерской чаще – через каждые 5 ± 5 минут, - и мы должны определить характеристики очереди клиентов при условии, что теперь их будут обслуживать четыре парикмахера (рис.4).

Четыре кресла можно изобразить в модели памятью емкостью 4 единицы, в которой каждый транзакт будет занимать одну единицу. Модель принимает тогда следующий вид.

```
10 KRES  STORAGE        4       ;память KRES имеет емкость 4 ед.
20       GENERATE       5,5
30       QUEUE          1       ;вход в очередь 1
40       ENTER          KRES    ;занять в памяти KRES одно место
 50       DEPART         1       ;выход из очереди 1
```

```
60          ADVANCE         19,5
70          LEAVE           KRES    ;освободить место в памяти KRES
80          TERMINATE
**  Таймер модели  **
90          GENERATE        525600  ;моделировать 1 год работы
100         TERMINATE       1
START   1
```

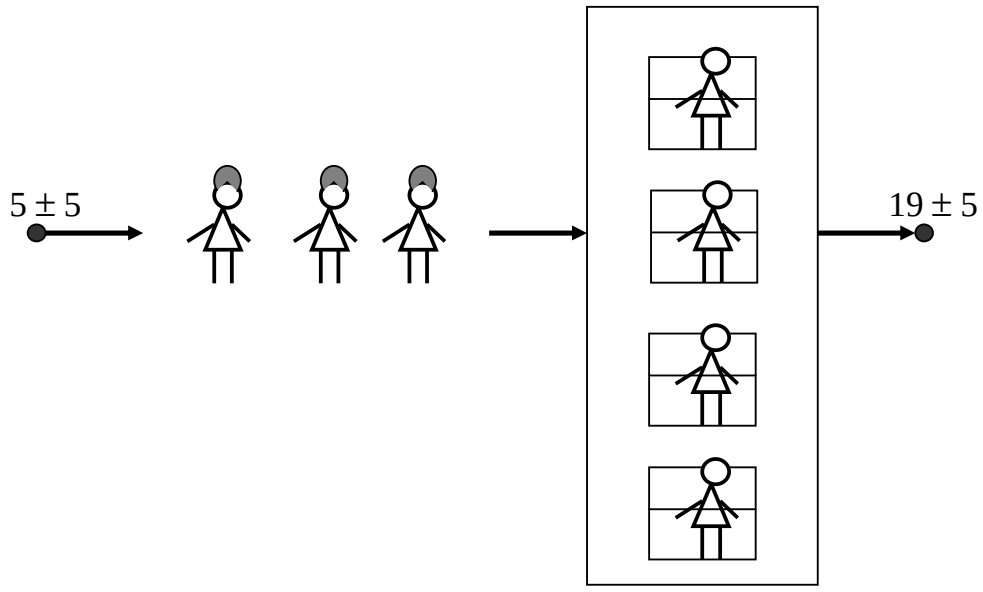


Рис.4

Транзакты в этой модели ведут себя точно так же, как клиенты в моделируемой парикмахерской. Когда в памяти KRES заняты все четыре единицы, то приходящие из блока GENERATE транзакты не могут войти в блок ENTER, и ожидают освобождения памяти, как клиенты ожидают освобождения одного из четырех кресел парикмахеров.

ГЛАВА 3. ПРАКТИЧЕСКАЯ РАБОТА НА КОМПЬЮТЕРЕ

3.1. Модель обслуживания с отказами

Усовершенствуем немного последнюю модель парикмахерской, чтобы учесть в ней возможные отказы клиентов от обслуживания. Будем считать, что когда приходящий клиент видит в очереди более двух клиентов, он уходит из парикмахерской, т.е. отказывается от обслуживания.

Условное изображение системы массового обслуживания, соответствующее этому варианту модели, представлено на рис.5. Линии изображают здесь возможные маршруты движения заявок (т.е. клиентов), прямоугольники изображают очередь заявок, надпись над очередью означает, что ее длина Q1 не может превышать трех заявок. Кружками изображены каналы, т.е. обслуживающие приборы (в данном случае – кресла парикмахеров).

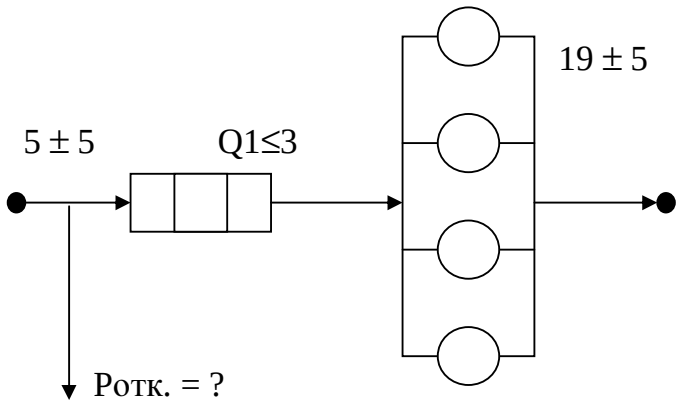


Рис. 5

В состав искомых данных наряду с характеристиками очереди клиентов включим вероятность отказа, т.е. вероятность того, что пришедший клиент застанет в парикмахерской очередь из трех человек и покинет парикмахерскую.

Модель для такого варианта функционирования парикмахерской может написана на GPSS так, как показано ниже.

```
10 KRES   STORAGE      4          ;память KRES имеет емкость 4 ед.
20        GENERATE     5,5
30        TEST  L      Q1,3,OTKAZ
40        QUEUE        1          ;вход в очередь 1
```

```
50          ENTER           KRES      ;занять в памяти KRES одно место
60          DEPART          1         ;выход из очереди 1
70          ADVANCE         19,5
80          LEAVE           KRES      ;освободить место в памяти KRES
90        TERMINATE
100 OTKAZ TERMINATE

**   Таймер модели  **

110          GENERATE          525600    ;моделировать 1 год работы
120         TERMINATE         1
START    1
```

Блок TEST выполняется следующим образом. При входе транзакта в этот блок проверяется условие, обозначенное после слова TEST буквой L, т.е. условие «меньше». Сравниваемые величины заданы в полях A и B блока TEST. Таким образом, проверяется условие Q1<3. Обозначение Q1 в GPSS стандартно соответствует текущей длине очереди номер 1.

Если проверяемое в момент входа транзакта в блок TEST условие выполняется (тест выполнен), то транзакт проходит дальше, к следующему блоку. В данной модели он входит в очередь 1. Если условие не выполняется, то транзакт переходит по альтернативной метке, записанной в поле C блока TEST. Значит, если длина очереди будет не меньше трех, т.е. если будет Q1=3 или Q1>3, то транзакт перейдет по метке OTKAZ –и уничтожится в строке 100 модели. Это соответствует уходу клиента, заставшего в очереди трех клиентов.

3.2. Выполнение имитационного эксперимента

Этот параграф следует читать, сидя за компьютером и выполняя последовательно все приведенные ниже инструкции. Текст инструкций выделяется жирным курсивом.

1. Войти в подкаталог GPSS и запустить программу gpss.exe.

Появляется заставка, а затем рабочий экран GPSS приблизительного такого вида (рис.6).

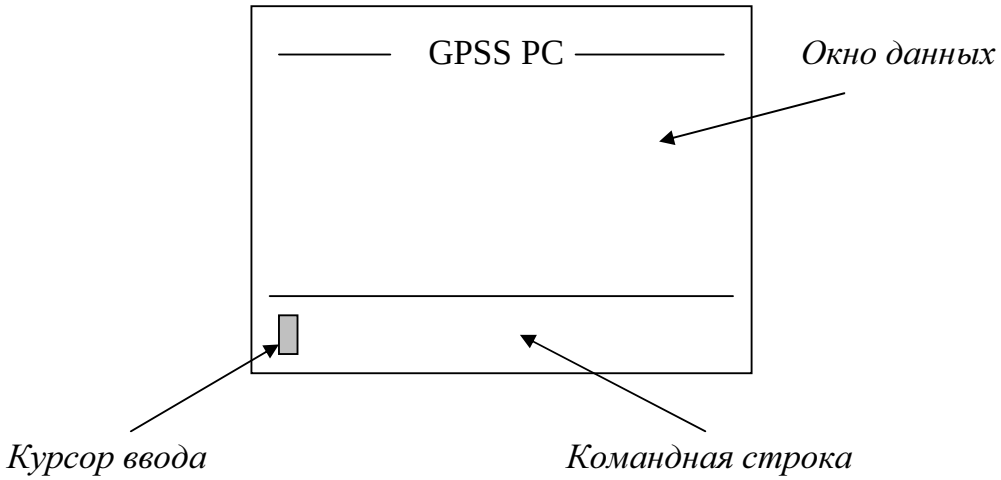


Рис.6

2. Набрать в командной строке построчно текст модели.

Имеется в виду последний вариант модели парикмахерской. Структура набираемых строк включает следующие поля:

```
Номер   Метка   Операция   Операнды A,B,…          ;Комментарии
Строки
```

Например, первая строка модели имеет номер 10, метку KRES, операцию STORAGE, операнд A задан числом 4. Комментарием считаются символы, начинающиеся после первого пробела в поле операндов.

Составные части строки отделяются друг от друга пробелами. После нажатия пробела курсор автоматически перескакивает в начало следующей части. Если в наборе поля допущена синтаксическая ошибка, то GPSS выдает звуковой сигнал и не позволяет продолжить набор, пока ошибка не будет исправлена.

Метки и комментарии в строках модели могут отсутствовать. Если метка отсутствует, то после ввода номера строки и нажатия пробела нужно нажать пробел еще раз, чтобы перевести курсор из поля метки в поле операции. Так, при наборе второй строки модели

```
20          GENERATE    5,5
```

необходимо после ввода числа 20 нажать пробел дважды.

В области операндов пробелы вставлять нельзя, т.к. любой пробел в этом поле означает, что ввод операндов закончен, поэтому следующие за пробелом символы будут рассматриваться всего лишь как комментарий.

Ввод операции в поле операции можно выполнять не полностью, достаточно ввести несколько первых символов и нажать пробел. Слово будет дописано до конца автоматически и курсор переместится в поле операндов.

Набор строки заканчивается клавишей ENTER. Набранная строка переместится из командной строки в рабочее поле GPSS. Это означает, что строка включена в программный файл модели.

Строки можно набирать в произвольном порядке. В рабочем поле их последовательность будет соответствовать порядку набора. Но в программном файле модели строки будут упорядочены по возрастанию номеров строк.

Номер строки не обязательно должен быть целым числом. Это позволяет легко вставлять новую строку между любыми двумя уже введенными строками.

При вводе модели можно использовать следующие команды:

EDIT <n>	- редактировать строку с номером n,
DISPLAY	- показать файл модели,
DISPLAY <n1>,<n2>	- показать файл, строки с n1 по n2,
DELETE <n>	- удалить строку n,
DELETE <n1>,<n2>	- удалить строки с n1 по n2.

Команды пишутся без номера строки и выполняются немедленно после их ввода. Названия команд также можно писать не полностью.

3. Сохранить введенную модель командой

SAVE PAR.GPS

Здесь PAR.GPS – имя файла, которое мы выбрали для модели. Имя файла может быть любым. Рекомендуется модели на GPSS сохранять под именами с расширением GPS.

4.Запустить модель на выполнение с помощью команды
START 1

5. Перейти в окно блоков с помощью сочетания клавиш (Alt+B), затем в окно памятей (Alt+S).

Приостановить модель (Esc).

Вернуться в текстовое окно (команда DISPLAY).

Продолжить моделирование (команда CONTINUE).

Для переходов между исходным окном данных и графическими окнами объектов модели используются следующие сочетания клавиш.

Окно блоков - Alt+B
Окно памятей - Alt+S
Окно устройств - Alt+F
Окно таблиц - Alt+T
Окно матриц - Alt+M
Окно данных - Alt+D

Буква в сочетании клавиш везде является первой буквой английского названия соответствующего объекта.

Объекты в графических окнах изображаются в динамике, т.е. в анимации, если модель находится в состоянии выполнения. Поэтому в графических окнах время выполнения модели компьютером возрастает по сравнению со временем, затрачиваемым в окне данных.

Состояние выполнения модели отображается в командной строке сообщением

Simulation in Progress …

Когда выполнение модели заканчивается, выводится звуковой сигнал и в командной строке появляется сообщение

Simulation complete, reporting …

Слово *reporting* …высвечивается до тех пор, пока результаты моделирования выводятся на диск в файл report.gps.

6. Когда слово reporting … погаснет, выйти из GPSS с помощью команды END

7. Просмотреть статистику, выданную в файл report.gps.

Статистика в файле report.gps имеет вид, неудобный для визуального просмотра, т.к. записывается туда во внутреннем формате GPSS. Отформатировать ее для просмотра можно с помощью служебной программы (утилиты) gpssrept.exe. После запуска этой утилиты следует подтвердить имя файла – источника данных для форматирования (нажатием клавиши ENTER) и ввести имя файла – приемника форматированных результатов. В данном случае, поскольку модель называется PAR.GPS, то файл результатами удобно назвать PAR.TXT. После нажатия пробела выполняется форматирование. Выйти из утилиты можно по клавише Esc.

8. Просмотреть созданный с помощью форматирования отчет PAR.TXT.

Основные опции информации в этом файле следующие.

(Трассировка:)

LINE	LOC	BLOCK_TYPE	ENTRY_COUNT	CURRENT_COUNT	RETRY
20	1	GENERATE	104985	0	0
30	2	TEST	104985	0	0
40	3	QUEUE	100032	2	0
50	4	ENTER	100030	1	0
60	5	DEPART	100029	0	0
70	6	ADVANCE	100029	3	0
80	7	LEAVE	100026	0	0
90	8	TERMINATE	100026	0	0
100	OTKAZ	TERMINATE	4953	0	0
110	10	GENERATE	1	0	0
120	11	TERMINATE	1	0	0

(Статистика по очередям:)

QUEUE	MAX	CONT.	ENTRIES	ENTRIES(0)	AVE.CONT.	AVE.TIME	AVE.(-0)	RETRY
1	3	3	100032	41854	0.70	3.67	6.30	0

(Статистика по памятям:)

STORAGE	CAP.	REMAIN.	MIN.	MAX.	ENTRIES	AVL.	AVE.C.	UTIL.	RETRY	DELAY
KRES	4	0	0	4	100030	1	3.62	0.904	0	2

Трассировка содержит следующую основную информацию.

В колонке ENTRY_COUNT (счетчик входов) показано число транзактов, прошедших за время моделирования через каждый блок модели. Так, из блока GENERATE вышло 104985 транзактов, – столько клиентов приходили в парикмахерскую за год. В блоке OTKAZ уничтожено 4953 транзакта. Столько клиентов отказались от обслуживания. Отсюда можно найти оценку вероятности отказа:

$$Ротк. = 4953/104985 = 0{,}047.$$

С математической точки зрения процесс появления клиентов представляет собой рекуррентный поток событий. Применяя соответствующие математические знания, можно видеть, что общее число приходящих за год клиентов, т.е. число 104985, является случайным. Его математическое

ожидание составляет величину 525600/5 = 105120. Значит, реализация этого случайного числа, полученная при моделировании, отличается от мат. ожидания более чем на 1000 (клиентов). Точно также число отказов 4953 представляет собой только одну из возможных случайных реализаций, т.е. величину весьма неточную. В то же время, привлекая анализ сходимости оценок стационарных вероятностных характеристик, мы можем заключить, что полученная выше *оценка вероятности отказа*, равная 0,047, имеет достаточно высокую точность. По крайней мере, можно утверждать, что все три цифры после запятой у этой оценки точные.

В колонке CURRENT_COUNT (счетчик текущих) показано число транзактов, задержанных в каждом блоке в момент останова модели.

Статистика по очередям содержит таблицу, в которой по каждой очереди в модели приводятся следующие основные данные.

В колонке QUEUE (очередь) содержится номер или имя очереди.

В колонке MAX (максимум) – максимальная длина очереди, которая достигалась за все время моделирования. Естественно, что в нашей модели длина очереди не могла превысить 3-х, т.к. клиенты, застающие очередь из трех клиентов, в нее не становятся, уходят.

В колонке CONT (содержимое) приводится текущая длина очереди в момент останова модели.

Колонка ENTRIES (входы) содержит число транзактов, вошедших в очередь. Видно, что это число совпадает с числом транзактов, вошедших в блок QUEUE, показанное в трассировке.

Колонка ENTRIES(0) содержит число транзактов, прошедших очередь без задержки (с *нулевой* задержкой). Т.о. 41854 клиента в момент прихода в парикмахерскую заставали свободного парикмахера, в очереди не стояли. Получается, что таких «везучих» клиентов было около 40% или почти в десять раз больше, чем «невезучих», отказавшихся от обслуживания.

В колонке AVE.CONT (среднее содержимое) выводится средняя длина очереди.

Столбец с заголовком AVE.TIME содержит среднее время прохождения транзакта через очередь. Соседний столбец AVE.(-0) – такое же среднее время, но рассчитанное только для тех транзактов, которые прошли очередь за ненулевое время, т.е. которые фактически задерживались в очереди.

Мы можем теперь ответить на вопрос, поставленный в задаче моделирования, следующим образом: при заданных параметрах парикмахерской средняя длина очереди клиентов к парикмахерам составляет 0,7 клиента, среднее время ожидания начала обслуживания равно 3,67 минуты, 4,7% клиентов отказываются от обслуживания из-за длинной очереди.

Статистика по памятям.

Содержимое статистики по памятям можно понять из перевода заголовков столбцов таблицы памятей, если при этом учитывать пояснения, приведенные выше для статистики по очередям. Заголовки основных столбцов переводятся так:

STORAGE - память
CAP. - емкость
MIN - минимум
MAX - максимум
ENTRIES - входы
AVE.C - среднее содержимое
UTIL - использование.

Отдельно требуется пояснить только заголовок UTIL. Под этим заголовком выводится коэффициент использования памяти. В отличие от коэффициента использования устройства он представляет собой не долю времени, в течение которой память была занята, а среднее содержимое памяти, деленное на емкость. Этот коэффициент также получается нормированным, т.е. находится всегда в пределах от 0 до 1.

С математической точки зрения (с точки зрения теории массового обслуживания) это есть не что иное, как коэффициент загрузки многоканальной СМО, в которой число каналов равно емкости памяти. Если выражаться точнее, это даже не коэффициент загрузки, а *статистическая оценка стационарного значения коэффициента загрузки.*

Этот показатель имеет тот экономический смысл, что выражает, насколько полно используется ресурс памяти, или, - в нашей модели, - насколько полностью загружены парикмахеры. Коэффициент загрузки получился равным 0,904. Это значит, что любой из четырех парикмахеров в среднем 90,4% времени занят непосредственным обслуживанием клиента. Пожалуй, такой режим работы может выдержать далеко не каждый!

8. Войти в GPSS с помощью команды

GPSS.EXE @PAR.GPS

Модель из файла PAR.GPS введется в GPSS автоматически. Если же вход выполнен по обычной команде загрузки GPSS.EXE, то, находясь в GPSS, можно загрузить в него модель PAR.GPS вводом в командной строке команды

@PAR.GPS

9. С помощью команды EDIT ввести в модель изменение, направленное на снижение вероятности отказа и на уменьшение загрузки парикмахеров.

Для этого можно увеличить число парикмахеров (т.е. емкость памяти KRES) на единицу.

10). Выполнить эту модель и определить, как изменились характеристики работы парикмахерской.

ГЛАВА 4. МОДЕЛИРОВАНИЕ ШВЕЙНОЙ ФАБРИКИ

4.1. Задача оптимизации производства

Задача моделирования швейной фабрики [1] содержит следующие исходные данные.

На трикотажной фабрике 50 швей работают на 50 машинах по 8 часов в день и по 5 дней в неделю. Любая из этих машин в любой момент во время работы может выйти из строя. В этом случае ее сразу заменяют резервной машиной. Если резервных машин нет, то швея простаивает до появления резервной машины, причем *почасовой убыток* от простоя одной швеи составляет 20 долларов. Тем временем сломанную машину передают в ремонтную мастерскую, где чинят в течение 7±3 ч и возвращают в цех уже в качестве резервной.

Известно, что *время наработки на отказ* для швейной машины составляет 157±25 ч. Оплата одного рабочего в ремонтной мастерской – 3,75 доллара в час.

Для создания запаса резервных машин предлагается кроме собственных 50 машин арендовать еще несколько по цене 30 долларов в день.

Управляющий должен решить, сколько рабочих следует нанять для работы в мастерской и сколько машин арендовать, чтобы ежедневные издержки производства были минимальны.

4.2. Схема процесса и предварительный анализ

Формально поставленную выше задачу можно записать в виде оптимизационной задачи:

$$COST(Np,Nм) \rightarrow \min,$$

где Np – число рабочих, нанимаемых для ремонта,

Nм – число машин, арендуемых для создания резерва,

COST – издержки, зависящие от выбора величин Np и Nм.

Схема процесса, в котором формируются эти издержки, приведена на рис.7.

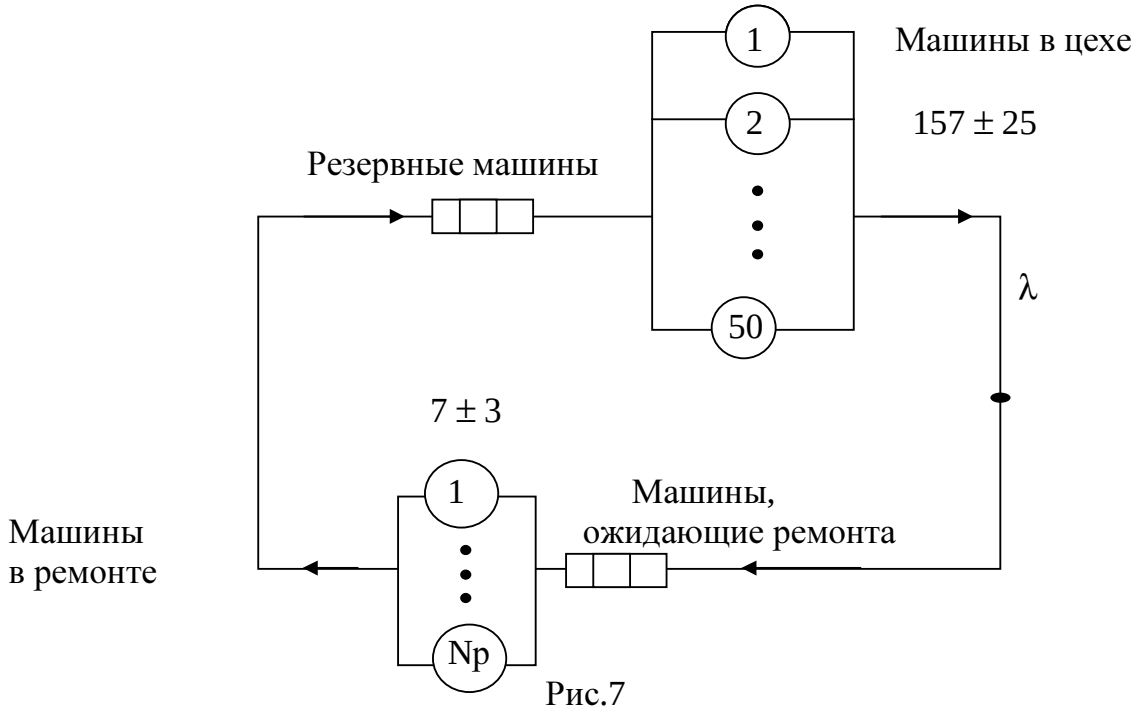


Рис.7

Поскольку процесс отказов машин случайный, то реальные издержки, возникающие ежедневно на швейной фабрике, будут случайными. Поэтому величину COST мы определим как *средние ежедневные* издержки производства.

Линии со стрелками на рис.6 показывают движение швейных машин в процессе их отказов и ремонта. Швейный цех изображен в виде многоканальной СМО с 50 местами. Можно поставить в соответствие этому цеху *память* GPSS, которая имеет емкость 50 единиц, а в соответствие машинам – *транзакты*, которые будут занимать места в этой памяти на время безотказной работы 157 ± 25 ч.

Резервные машины образуют очередь к швейному цеху. Сломанная машина освобождает место в цехе и переходит на участок ремонта. Здесь она может попасть в очередь, если все рабочие уже заняты. Отремонтированная машина вновь поступает в резерв.

Таким образом, система в целом представляет собой замкнутую сеть массового обслуживания, в которой циркулируют 50 + Nм заявок (машин).

Данная сеть может быть описана системой интегро-дифференциальных уравнений. Но получаются уравнения такого типа, решить которые математикам до сих пор не удается. По мнению известных математиков, нет надежды, что в ближайшие 100 лет появятся методы точного решения таких уравнений.

Имитационное моделирование позволяет находить все необходимые характеристики систем, не составляя и не решая уравнений.

Средние издержки за один рабочий день можно выразить следующим образом:

$$COST = 30(Np + Nм) + 8000(1- R),$$

где Np – число рабочих, нанятых для работы на участке ремонта,

Nм- число арендуемых для резерва машин (сверх 50-ти собственных).

R – коэффициент использования швейного цеха.

Величину (1-R) в формуле можно назвать *коэффициентом простоя* – это средняя доля времени, в течение которого каждое место в цехе простаивает (швея простаивает без работы). Поэтому величина **8***(1-R) равна среднему *времени* простоя одной швеи в течение 8-часового рабочего дня, а величина **20***8*(1-R) – это средние *издержки* за рабочий день от простоя одной швеи, а **50***20*8*(1-R) = 8000*(1-R) – средние издержки за рабочий день от простоя *всех 50-ти* швей.

Первое слагаемое в приведенной формуле для COST выражает издержки, которые включают оплату за Nм арендуемых машин (по 30 долларов в день за одну машину) и оплату одного дня работы всем Np рабочим, по 30 долларов в день каждому.

Моделирование приходится выполнять потому, что мы не можем иным путем определить величину R, входящую в формулу для издержек.

Естественно будет предположить, что в оптимальном варианте системы коэффициент загрузки швейного цеха будет близок к единице, т.е. что в оптимальном варианте в цехе будут практически без простоя работать все 50 швей. Тогда средняя интенсивность отказов машин λ составит величину

$$\lambda = 50*(1/157) \approx 1/3 \text{ (отказов в час)},$$

где 157 – среднее время между отказами одной работающей машины,

1/157 – средняя интенсивность отказов одной машины,

50 – число работающих машин в цехе,

λ - средняя интенсивность отказов всех работающих машин, т.е. среднее число машин, поступающих за 1 час на участок ремонта.

Интенсивность ремонта машин μ должна быть не меньшей чем λ (рабочие должны успевать ремонтировать за один час в среднем не меньше машин, чем за это время поступает):

$$\mu \geq \lambda,$$

где μ = Np *(1/7) – интенсивность ремонта машин всеми Np рабочими, 7 – среднее время ремонта машины одним рабочим, (1/7) – интенсивность ремонта машин одним рабочим. Следовательно

Np*(1/7) ≥ 1/3,
Np ≥ 7/3,
Np ≥ 2,3,
Np.min = 3.

Исходя из того, что в оптимальном решении Np ≥ 3, можно предположить, что и число резервных машин Nм также должно быть не меньше трех, т.к. для того, чтобы 50 машин работало, а 3 находились в ремонте, нужно иметь три резервных машины.

Выполненный здесь анализ является приближенным, но он позволяет начать перебор вариантов с более или менее разумного решения, в котором число рабочих и число резервных машин принимается равным трем.

Напишем модель для этого варианта управленческого решения и определим для него величину издержек COST.

4.3. Разработка моделирующей программы

Составим модель швейной фабрики на языке GPSS следующим образом.

```
10  COST1   VARIABLE    8000-8#SR$CEH
20  COST2   VARIABLE    30#(3+3)              ;30#(Np+Nм)
30  COST    VARIABLE    V$COST1+V$COST2
40  CEH     STORAGE     50
50  MEN     STORAGE     3                     ;Np
60          GENERATE    ,,,53                 ;50+Nм
70  PUSK    ENTER       CEH
80          ADVANCE     157,25
90          LEAVE       CEH
100         ENTER       MEN
110         ADVANCE     7,3
120         LEAVE       MEN
130         TRANSFER    ,PUSK
140         GENERATE    6240
150         TERMINATE   1
START 1
```

Здесь в строке 10 описана переменная с именем COST1, с помощью которой вычисляется та часть издержек, которая вызвана простоем швей. Швейный цех моделируется памятью CEH емкостью 50 единиц (описана в строке 40), а движение машин моделируется транзактами. Коэффициент

использования памяти в GPSS выражается стандартным числовым атрибутом SR. Обозначение SR$CEH – это обозначение коэффициента загрузки памяти CEH. Обозначение SR$CEH состоит из двух частей: из *названия атрибута* (*SR* – коэффициент загрузки памяти) и соединенного с ним (через значок $) *имени* конкретной памяти *CEH*. Однако коэффициент использования памяти SR$CEH выражается в GPSS не в долях от единицы, а в долях от тысячи, т.е.

SR$CEH = 1000*R.

Отсюда ясно, почему переменная COST1 описана в строке 10 программы таким выражением, – ведь именно так выражается слагаемое 8000*(1-R):

8000*(1-R) = 8000 – 8000*R = 8000 – 8*SR$CEH = 8000 – 8#SR$CEH.

Знак # обозначает в GPSS операцию умножения.

В строке 20 программы описана переменная COST2, равная доле издержек, связанной с арендной платой за машины и с оплатой труда рабочих. В данном варианте число рабочих Np = 3, число резервных машин Nм = 3.

В строке 30 переменная COST вычисляет суммарные издержки, т.е. равна сумме переменных с именами COST1 и COST2. Обозначения этих переменных состоят из обозначения типа числового атрибута (*V* – переменная VARIABLE) и из присоединенного к нему имени конкретной переменной (*COST1* или *COST2*).

Строки 40 и 50 описывают две памяти – память CEH и память MEN. Память MEN моделирует рабочих, ремонтирующих машины, и имеет емкость, равную числу рабочих.

Собственно процесс использования и ремонта швейных машин моделируется блоками в строках 60-130.

Блок

60 GENERATE ,,,53

генерирует 53 транзакта в нулевой момент времени, что соответствует вводу в действие 50-ти собственных машин и 3-х арендуемых для резерва.

Следующий блок

70 PUSK ENTER CEH

моделирует установку машин на рабочие места в цехе. Т.к. емкость памяти CEH равна 50, то в нее войдет 50 транзактов, а 3 транзакта останутся в состоянии ожидания в очереди к памяти, моделируя резервные машины.

Блок

80 ADVANCE 157,25

задерживает транзакты на время 157 ± 25 единиц, которое соответствует времени безотказной работы машин. Через такое случайное время транзакты, занимающие память, будут выходить из блока ADVANCE и в следующем блоке

90 LEAVE СЕН

освобождать место, занимаемое в памяти СЕН.

Аналогично строкам 70,80 и 90, описывающим прохождение машин через швейный цех, строки 100,110 и 120 описывают движение машин через участок ремонта.

Блок

130 TRANSFER ,PUSK

пересылает транзакты в блок с меткой PUSK, что соответствует переходу отремонтированной машины - в качестве резервной - в швейный цех.

Строки 140 и 150 задают таймер модели. Поскольку математическую единицу времени в данной модели мы интерпретируем как 1 час, то интервал времени 6240 в блоке GENERATE таймера соответствует 3-м годам работы фабрики.

Более подробная информация об алгоритмах выполнения и синтаксисе операторов GPSS приводится в главе 7.

Обратим внимание, что параметр Np – число рабочих – присутствует в тексте модели в двух местах (значение **3** этого параметра выделено жирным шрифтом): в строке 20, где вычисляются издержки на зарплату рабочих, и в строке 50, где определяется число рабочих мест на участке ремонта. Это следует учитывать при редактировании модели для перехода к новому варианту: если значение Np меняется, то его нужно изменить и в строке 20, и в строке 50. Аналогичное замечание имеет место и для параметра Nм – числа рабочих. Значение параметра Nм задано в строках 20 и 60 (в приведенном выше тексте модели его значение ***3*** выделение курсивом).

4.4. Планирование эксперимента

В решаемой задаче нас интересуют стационарные характеристики системы (различия между стационарными характеристиками и характеристиками переходного процесса рассматриваются в прил.1).

Следовательно, задача планирования заключается в том, чтобы найти время, задаваемое в таймере модели, при котором статистическая оценка издержек COST в достаточной мере стабилизируется. Найти такое время можно за несколько шагов, путем постепенного наращивания счетчика карты START и наблюдения за величиной получаемой оценки.

Для этого войдем в GPSS\PC, запустив программу GPSS.EXE.

Введем модель швейной фабрики, текст которой приведен в предыдущем параграфе.

Шаг 1: введем команду

START 1,

т.е. запустим модель для моделирования 3-х лет работы фабрики.

После завершения моделирования, не выходя из GPSS, введем команду

SHOW V$COST (показать значение переменной V$COST). Выпишем с экрана полученное значение издержек V$COST вместе со значением 1 счетчика карты START.

Введем команду CLEAR для сброса статистики, накопленной в результате выполненного прогона и для возвращения модели в исходное состояние.

Шаг 2: введем команду

START 2,

Т.е. запустим модель на моделирование 2*3 = 6 лет.

SHOW V$COST – снова выписываем значение счетчика START и V$COST.

CLEAR.

Шаг 3: START 3

SHOW V$COST

CLEAR

И т.д.

В результате дальнейшего продолжения этой последовательности шагов получается следующая зависимость оценки издержек V$COST от длины прогона (табл. 1):

Таблица 1

Счетчик START	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	...
V$COST	308	300	292	284	284	276	276	284	276	284	284	276	276	276	276	...

На основании выполненного испытания можно сделать вывод, что оценка средних издержек сходится к точному значению при счетчике карты START, равном 12 – это соответствует 12*3 = 36 годам работы фабрики.

Впредь, моделируя различные варианты управленческого решения, т.е. различные сочетания параметров Np и Nм, мы будем для каждого варианта выполнять прогон модели при счетчике карты START, равном как минимум 12 (ниже приводятся результаты *с запасом надежности* – при счетчике, равном 20).

Кроме вывода о длине прогона мы можем из выполненного здесь планирования эксперимента заключить, что при Np = 3 и Nм = 3 издержки COST составляют 276 долларов в день.

4.5. Оптимизация управленческого решения

Поиск оптимального варианта решения можно осуществить покоординатным методом. Для этого нужно изменять вначале только один параметр решения (например, Np), до тех пор, пока не будет найден минимум издержек по этому направлению изменений, а затем изменять только второй параметр (Nм), затем снова – только первый, и т.д.

Схема покоординатной оптимизации проиллюстрирована на рис.8.

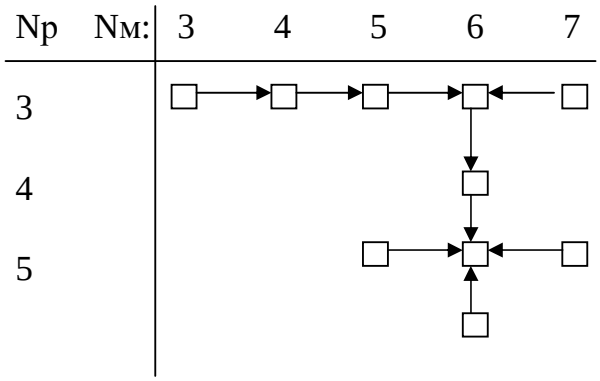


Рис.8

На этой схеме квадратики изображают числовые значения минимизируемого показателя (V$COST), а стрелки показывают направление уменьшения этого показателя. Мы продвигаемся вначале вдоль первой строки таблицы до тех пор, пока значение минимизируемого показателя уменьшается (т.е. до первого случая увеличения показателя). Затем от наименьшего значения движемся вниз также до тех пор, пока показатель уменьшается. Так продолжается движение то в одном, то в другом направлении до тех пор, пока мы не «опустимся» в точку, все направления из которой уже не приводят к уменьшению показателя.

При моделировании переход к новым значениям параметров Np и Nм выполняется с помощью команд:

```
CLEAR
EDIT  20
EDIT  50          } изменение параметров
   (или EDIT 60)
START  20
SHOW   V$COST  (результат выписывается в таблицу, см рис.8).
```

В данном примере получается таблица со следующими переходами (рис.9):

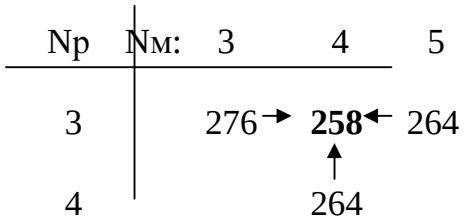


Рис.9

Таким образом, оптимальный вариант состоит для моделируемой швейной фабрики в том, чтобы нанять для ремонта машин 3-х рабочих и арендовать для создания резерва 4 машины.

4.6. Расчет экономического эффекта

После того, как найден оптимальный вариант управленческого решения, можно найти и получаемый годовой экономический эффект.

В качестве *базового варианта* будем рассматривать вариант, в котором число нанимаемых рабочих составляет 3 (как в оптимальном варианте), а машины для создания резерва не арендуются, – аренда машин составляет суть новшества, предложенного менеджером предприятия.

Чтобы найти издержки в базовом варианте работы фабрики, перейдем к соответствующему варианту модели с помощью следующих команд:

```
CLEAR       сброс статистики по предыдущему варианту
EDIT …
EDIT …      переход к базовому варианту: Np = 3 и Nм = 0
EDIT …
START  20
SHOW   V$COST
```

Получаем в результате выполнения команды SHOW V$COST ответ: 514.

Таким образом, средние издержки за день в базовом варианте составляют 514 долларов.

Годовой экономический эффект можно рассчитать с помощью команды SHOW (которая позволяет вычислять выражения), умножая разницу между базовыми и оптимальными издержками в день на число дней в неделе и число недель в году:

```
SHOW   (514 – 258)#5#52
```

(ответ – 66560 долларов).

Теперь можно выйти из GPSS с помощью команды END. Заметим, что в решенной здесь задаче нам не пришлось обращаться к файлу отчетов REPORT.GPS – интересующие нас значения переменной V$COST мы получали с помощью интерактивной команды SHOW.

4.7. Другие применения модели

Любая модель имеет множество возможных применений, т.е. обладает некоторым *универсумом*.

Построенная выше модель швейной фабрики также обладает определенным универсумом. С ее помощью можно решать задачи, отличающиеся от решенной выше.

В качестве примеров приведем следующие две задачи.

Задача 1. После найма Np рабочих и аренды Nм машин менеджеру фабрики предлагается за 6 000 долларов обучить ремонтных рабочих и оснастить их специальными инструментами. В результате этого среднее время ремонта машин одним рабочим сократится с 7 часов до 5 часов, т.е. время ремонта машины составит 5 ± 2 ч.

Требуется определить, выгодно ли это предложение для фабрики, и, если выгодно, то каков будет годовой экономический эффект этого мероприятия.

Каков будет экономический эффект, если обучить только двоих рабочих за 6000 долларов и повысить их оплату до 35 долларов в день, а третьего рабочего уволить?

Задача 2. Менеджер фабрики думает о том, стоит ли *нанять* (после того, как взято в аренду 4 машины и нанято 3 рабочих - оптимальное решение на предыдущем этапе), *еще одного рабочего*, который выполнял бы профилактический осмотр и настройку машин в нерабочее время (во вторую смену). Зарплата этого рабочего за один день равна зарплате ремонтных рабочих – 30 долларов в день. В результате его труда по профилактическому осмотру машин они будут отказывать не через 157 ± 25 часов работы, а через 200 ± 25ч, - т.е. несколько реже, чем раньше. Или, может быть, лучше перевести одного из рабочих-ремонтников на такую профилактическую работу во вторую смену? Нужно проверить оба этих варианта и принять решение об их целесообразности на основе годового экономического эффекта этих вариантов.

Приведенные задачи рекомендуется решить самостоятельно.

4.8. Критический анализ показателей эффективности

При имитационном моделировании систем мы имеем возможность применять для оценки системы множество самых разнообразных показателей, т.к., в отличие от аналитического моделирования или расчета, практически любой разумный показатель эффективности может быть в имитационном эксперименте вычислен без труда.

Популярность «средних значений» в качестве показателей сложилась исторически и объясняется тем, что, хотя оценки «в среднем» часто бывают очень мало информативны, но зато в аналитических расчетах определяются достаточно просто, а более информативные показатели сложны для вычисления.

В решенной задаче о швейном цехе мы также, по вековой традиции, сравнивали варианты управленческого решения по средним издержкам за день. Реальный же процесс, как он и воспроизводится моделью, случайный. После того, как управленческое решение будет принято, по результатам года будут получены реальные, т.е. случайные издержки, а не средние.

Наиболее полное представление о возможных реальных издержках за год можно получить, если прогон модели, соответствующий одному году работы фабрики, выполняется многократно. Тем самым реализуется множество возможных случайных «сценариев» процесса и определяется финансовый результат по каждому сценарию.

Выполним, например, десятикратный прогон для базового варианта модели, когда на швейной фабрике нет специального резерва машин (Nм = 0) и число ремонтников равно трем (Np = 3).

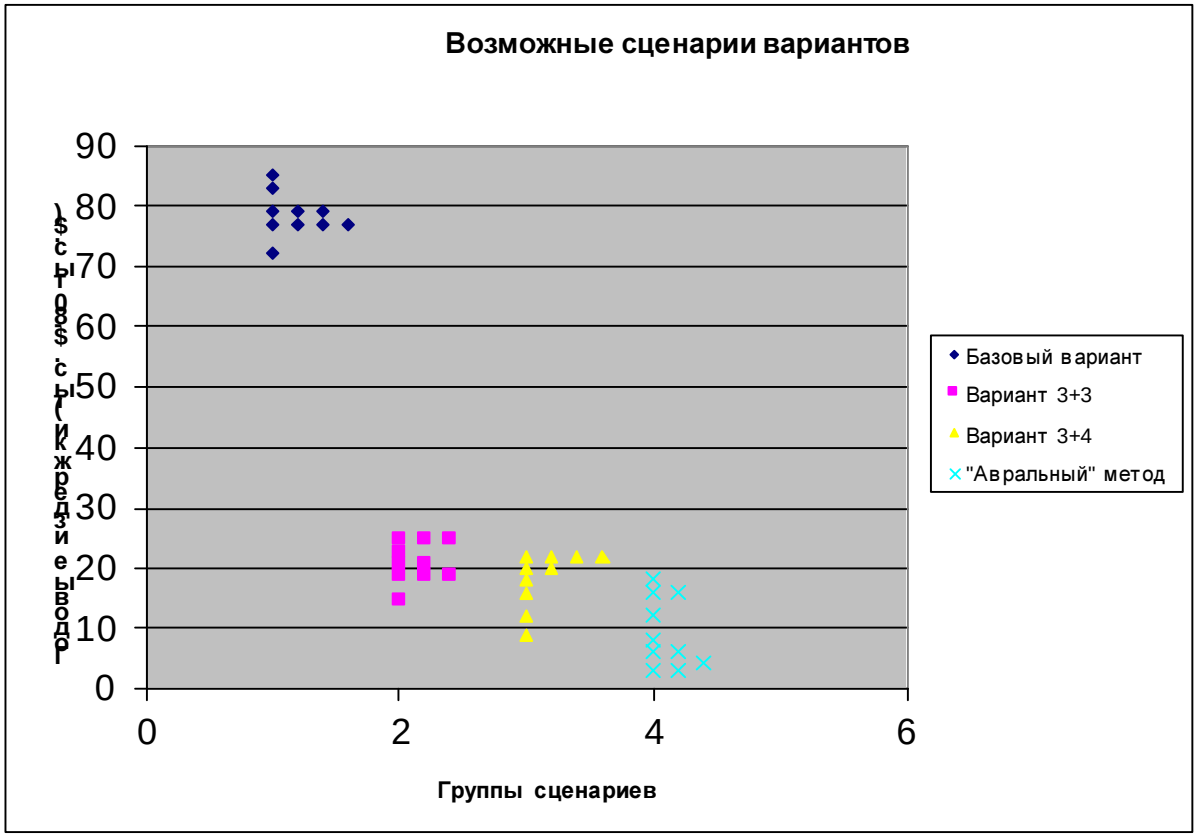


Рис.10. Сравнение вариантов по вероятным финансовым результатам

Результаты всех десяти реализаций базового сценария управления изображены на рис.10 десятью ромбиками (номер сценария равен единице). Каждый ромбик находится на высоте, соответствующей понесенным за год издержкам, выраженным в тысячах долларов и уменьшенным на 80 000 $ (для компактности графика). Чтобы определялись издержки за год, в модели на GPSS переменная с именем COST была скорректирована следующим образом: выражение, записанное в ней, заключается в скобки и к нему добавляются множители 5 (число дней в рабочей неделе) и 52 (число недель в году). Кроме того, в последнем блоке GENERATE, задающем момент окончания моделирования, в поле вместо 6240 нужно записать 2080 – число рабочих часов в году. Таким образом, модель завершает выполнение в момент, соответствующий одному году работы фабрики, и переменная V$COST дает значение издержек за год – реальных (т.е.случайных), а не средних. Десять прогонов дают десять возможных финансовых результатов (суммарных издержек) за один год.

Аналогично получены результаты для вариантов управления «3+3» и «3+4», которые отвечают первому промоделированному варианту управления (три машины и трое рабочих) и оптимальному варианту (четыре машины и трое

рабочих) соответственно. Сравнение этих двух вариантов с базовым (см. рис.9) показывает, что оба они имеют существенное преимущество перед базовым вариантом. Однако преимущество варианта «3+4» перед вариантом «3+3» выглядит теперь совсем не так, как при сравнении по средним издержкам. Вариант «3+3» характеризуется средними издержками за день в 276 $, а вариант «3+4» – в 258 $. Такая кажущаяся явной разница в реальности не обязательно проявится – из рис.10 видно, что возможные годовые издержки по обоим вариантам попадают практически в один и тот же интервал значений, и что вполне вероятно получить в варианте «3+3» издержки меньшие, чем в оптимальном варианте «3+4».

Понятно, что варианты «3+5» и «4+4» (см. рис. 10), более близкие к оптимальному по средним издержкам (равным 264 в обоих случаях), еще меньше будут отличаться от него в реальности, чем вариант «3+3». Это показывает, что решенная оптимизационная задача имеет в основном теоретический интерес, и не следует подражать ей на практике. На практике достаточно обосновать преимущества, которые появляются при аренде резервных машин, по сравнению с базовым вариантом.

На рис. 10 приведены также результаты моделирования «аврального» сценария управления ремонтом машин. Если моделировать швейную фабрику на GPSS и наблюдать ход процесса в окне памятей (Alt+S), то можно обратить внимание, что в начале процесса очередь сломанных машин на ремонт резко возрастает, затем уменьшается и далее колеблется на достаточно низком уровне. Причину этого явления понять несложно – она заключается в том, что время наработки машин на отказ находится в интервале от 157-25 = 132 ч до 157+25 = 182 часа. Это означает, что первые 132 ч ремонтники ничего не делают (а это около 16 рабочих дней), а потом в течение 50 часов к ним попадают все 50 машин из швейного цеха, т.е. в среднем по одной машине в час. С таким интенсивным потоком трое ремонтников не справляются. Возникает авральная ситуация.

«Авральный» метод управления в данном случае заключается в том, чтобы заключить контракт с ремонтными рабочими на работу, начиная с 17-го рабочего дня после установки новых машин в цехе. При этом с тремя рабочими контракт постоянный, и им выплачивается по 30 $ в день, и еще с тремя – контракт на 7 рабочих дней, и им выплачивается двойной тариф, т.е. по 60 $ в день. Или, что примерно то же самое, постоянный контракт заключается с тремя рабочими, и с ними же на 7 дней аврала заключается контракт на работу в две смены с оплатой второй смены по двойному тарифу.

Результаты моделирования «аврального» управления изображены на рис. 10 в виде крестиков. Можно видеть, что возможные исходы этого детально проработанного сценария в своей совокупности заметно предпочтительнее, чем результаты сценария «3+3».

ГЛАВА 5. ЗАДАЧА МОДЕЛИРОВАНИЯ КОНВЕЙЕРА

5.1. Постановка задачи

На комплектовочный конвейер сборочного цеха каждые 5 ± 1 мин поступают 5 изделий первого типа и каждые 20 ± 7 мин поступают 20 изделий второго типа. Конвейер состоит из секций, вмещающих по 10 изделий каждого типа. Комплектация начинается только при наличии деталей обоих типов в требуемом количестве и длится 10 мин. При нехватке деталей секция конвейера остается пустой.

Смоделировать работу конвейера сборочного цеха. Определить среднее число пропущенных секций за смену (за 8 часов). Найти средние и максимальные очереди по каждому типу изделий.

Предложить мероприятия по снижению числа пропускаемых секций. Оценить их экономическую целесообразность.

Данная задача представляет собой небольшую модификацию задачи из [6].

5.2. Схема процесса и предварительный анализ

Схема процесса изображена на рис.11. На схеме вверху показаны точки входа в систему деталей извне (из заготовительного цеха), рядом с которыми указан случайный интервал поступления деталей и, в квадратных скобках, размер партии деталей. Детали каждого типа накапливаются в очередях перед конвейером. Полоска с числами 10 и 10 на рис.11 – это символ синхронизации, выражающий условие разрешения комплектации: комплектация разрешается, если в момент сдвига конвейера в очередях имеется по 10 деталей обоих типов, и не разрешается в противном случае.

Конвейер изображен широкой полосой с обозначением секций, вмещающих по 10 деталей каждого типа. Отмечено время между сдвигами конвейера ΔT = 10 мин. Это означает, что конвейер стоит 10 мин, затем практически мгновенно сдвигается на одну секцию, снова стоит 10 мин, сдвигается и т.д. После каждого сдвига комплектация начинается либо сразу же, либо отменяется до конца 10-минутной фазы. Начинать комплектацию с задержкой нельзя – видимо, это привело бы к тому, что комплектация не завершилась бы вовремя, и все изделие, монтируемое на данной секции далее - до конца хода конвейера - оказалось бы бракованным.

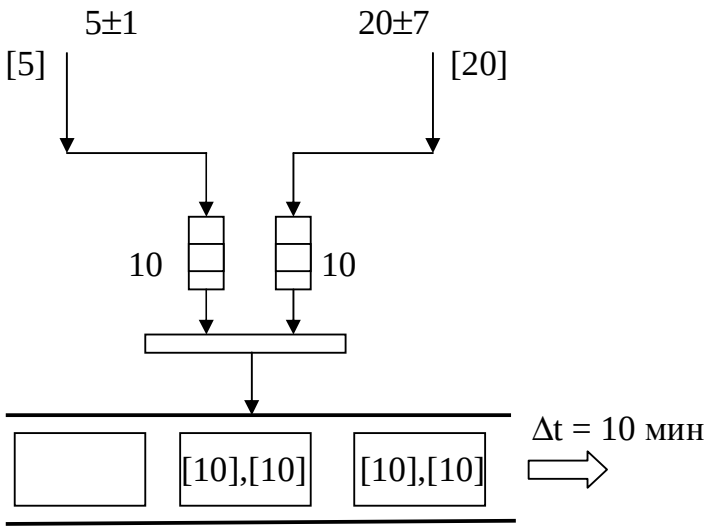


Рис.11

Предварительный анализ движения деталей на участке комплектации показывает следующее.

Во-первых, в среднем *участок сбалансирован*, т.е. средняя интенсивность поступления деталей на участок равна средней интенсивности их передачи конвейером на следующий участок. Действительно, детали первого типа поступают по 5 штук в среднем через пять минут, или одна деталь в минуту. Аналогично детали второго типа поступают в среднем также одна деталь в минуту. Конвейер способен увозить в среднем по 10 деталей каждого типа за 10 минут, т.е. по одной детали каждого типа в минуту.

Во-вторых, *в самом начале* процесса обязательно *будет пропущена, по крайней мере, одна секция,* т.к. в момент первого сдвига конвейера, равный 10 минутам, еще не появятся детали второго типа. Партия деталей второго типа придет не раньше, чем в момент времени, равный 20 – 7 = 13 мин.

В-третьих, когда возникает пропуск секции, то в течение последующих 10 минут детали на участок поступают, но с участка не уходят, и в течение этого времени очередь возрастает в среднем на 10 деталей каждого типа. После такого события вследствие увеличения очереди вероятность пропуска секции уменьшается. Поэтому логично предположить, что *пропуск секций конвейера происходит в основном в начале смены*. Эту гипотезу можно будет проверить в имитационном эксперименте и, если она будет верна, разработать соответствующие мероприятия, нацеленные на специфику начала смены.

5.3. Имитационная модель

Текст имитационной модели на GPSS имеет следующий вид.

```
KMPL    BVARIABLE    (Q1'GE'10)#(Q2'GE'10)
        GENERATE     5,1
        QUEUE        1,5
       TERMINATE
       GENERATE     20,7
       QUEUE        2,20
       TERMINATE
       GENERATE     10
       TEST E       BV$KMPL,1,PROP
       DEPART       1,10
       DEPART       2,10
       TERMINATE
PROP    TERMINATE
**** TIMER ****
        GENERATE     20
        TERMINATE    1
```

Номера строк в тексте модели для простоты опущены. Их можно задать при вводе модели в GPSS.

Модель имеет следующую структуру.

В первой строке описана булева переменная, которой присвоено имя KMPL. Значение булевой переменной равно единице, когда логическое выражение этой переменной истинно, и нулю – в противном случае. Логическое выражение переменной KMPL читается так: "первая очередь больше или равна 10 и вторая очередь больше или равна 10", или, короче, "деталей обоих типов достаточно для комплектации".

Далее следуют три блока GENERATE, QUEUE и TERMINATE, которые моделируют процесс наращивания очереди деталей первого типа: транзакты возникают каждые 5 ± 1 мин, наращивают длину первой очереди на 5 единиц и уничтожаются.

Следующие три блока моделируют процесс наращивания очереди деталей второго типа.

После этого идут блоки, начиная с GENERATE 10 и заканчивая блоком PROP TERMINATE, которые моделируют процесс укладки деталей из очереди на конвейер.

Блок GENERATE 10 выдает транзакт каждые 10 единиц времени, т.е. в моменты сдвига конвейера. Далее в блоке TEST транзакт проверяет, выполняется ли условие комплектации. Этот блок TEST читается следующим образом: "если равна булева переменная KMPL единице, то транзакт проходит через TEST к следующему блоку, иначе транзакт передается на блок с меткой PROP". Значит, если в данный момент – момент сдвига конвейера – условие комплектации выполняется, то транзакт проходит в блоки DEPART, где уменьшает на 10 единиц содержимое каждой очереди. Этим моделируется

начало комплектации деталей на конвейере. Если условие комплектации в данный момент не выполняется, то транзакт переходит из блока TEST в блок TERMINATE с меткой PROP. В результате очереди не уменьшаются. Число таких случаев пропуска секций мы сможем определить как число входов в блок с меткой PROP.

Таймер модели "заведен" на моделирование 20 минут работы конвейера. Благодаря этому мы сможем при моделировании видеть состояние системы через каждые 20 минут ее работы и проверить гипотезу о том, что большинство пустых секций проходит в начале смены.

5.4. Имитационный эксперимент

Войдем в GPSS, выполняя программу GPSSPC.EXE.

Введем модель.

Сохраним модель с помощью команды SAVE.

Перейдем в окно блоков, нажав пару клавиш Alt + B.

Создадим четыре микроокна для наблюдения значений следующих показателей:

- N$PROP (число входов в блок с меткой PROP, т.е. число пропущенных секций);
- Q1 (длина первой очереди деталей);
- Q2 (длина второй очереди деталей);
- C1 (модельное время в единицах модельного времени, здесь они соответствуют минутам);

Эти четыре микроокна создаются командами:

```
MICROWINDOW   1,N$PROP
MICROWINDOW    2,Q1
MICROWINDOW    3,Q2
MICROWINDOW    4,C1
```

Микроокна располагаются на правом краю экрана в порядке ввода снизу вверх.

Запустим процесс моделирования первых двадцати минут работы участка с помощью команды START 1.

Полученные показания выпишем из микроокон. В результате получим первую строку приведенной ниже таблицы (табл.2).

Запустим процесс моделирования следующих двадцати минут работы участка с помощью команды START 1. Поскольку после первой команды START мы не применяем команду CLEAR, то модель не переводится в начальное состояние – новая команда START просто продолжает процесс моделирования с той точки, в которой он был остановлен после предыдущего старта.

Полученные показания вновь выпишем, полчая таким образом вторую строку таблицы.

Продолжаем моделировать последовательные двадцатиминутные отрезки времени процесса, пока время С1 не достигнет значения 180 (мин), т.е. 3 ч.

Оставшийся 5-часовой отрезок смены (300 мин) моделируем одной командой START 15. Полученная в итоге таблица имеет следующий вид.

Таблица 2

Команда	N$PROP	Q1	Q2	C1
START 1	2	15	0	20
START 1	2	25	30	40
START 1	2	25	30	60
START 1	2	20	20	80
START 1	2	20	0	100
START 1	2	30	10	120
START 1	2	15	20	140
START 1	2	15	20	160
START 1	2	15	20	180
START 15	3	25	10	480

Вывод: в течение первых двадцати минут было пропущено две секции. Далее до конца смены была пропущена еще только одна секция. В течение смены всего пропущено три секции.

Промоделируем еще несколько смен, чтобы определить среднее число секций, пропускаемых за смену. Для восстановления начальных условий смены используем карту START.

С этой целью выполняем команды

CLEAR (восстановление начальных условий)

START 24 (моделирование 20*24 = 480 мин = 8 ч работы конвейера)

Для этой смены получаем 3 пропущенных секции.

Выполняем эти две команды еще два раза, моделируя тем самым еще две смены. Получаем для них соответственно 4 и 2 пропущенных секции.

Вывод: в среднем в течение смены получается около трех пропущенных секций. Поскольку за смену с конвейера сходит 480/10 = 48 готовых изделий, то

пропуск трех секций означает снижение производительности предприятия на (3/48)*100% ≈ 6 %.

5.5. Разработка мероприятий по совершенствованию системы

Эксперимент с моделью подтвердил гипотезу о том, что пропуски пустых секций возникают в основном в начале смены. Причина пропусков – недостаточный запас деталей. В связи с эти можно предложить следующие два варианта улучшения организации работы конвейера.

Вариант 1. Чтобы обеспечить больший запас деталей к началу работы конвейера, можно сдвинуть начало работы конвейера относительно начала работы заготовительного цеха на полчаса. За полчаса накопятся достаточные очереди деталей, и конвейер будет работать с меньшим числом пропусков.

Вариант 2. Запас деталей в очередях, складывающийся в конце смены, можно не расформировывать, а консервировать, сохраняя его неизменным для использования в начале следующей смены.

5.6. Моделирование первого варианта мероприятий

Чтобы промоделировать первый вариант мероприятий (конвейер начинает работать через полчаса после начала заготовки деталей), достаточно внести изменение в блок GENERATE 10, записав его следующим образом:

```
GENERATE      10,,30
```

В поле C этого блока задано время выдачи первого транзакта. Т.к. транзакты этого блока моделируют сдвиги конвейера, то такая редакция модели в точности соответствует первому варианту мероприятий.

Чтобы внести необходимое изменение в модель, следует перейти в окно данных (с помощью пары клавиш Alt + D), вывести текст модели на экран (по команде DISPLAY), и отредактировать нужную строку (указав ее номер в команде EDIT). После этого можно вновь вернуться в окно блоков и выполнить эксперименты с моделью усовершенствованной системы.

В результате моделирования конвейера, график работы которого сдвинут на полчаса, получаем для нескольких смен такие результаты (см. табл. 3).

Таблица 3

Команда	N$PROP	Q1	Q2	C1
START 24	1	25	10	480
CLEAR				
START 24	0	15	40	480
CLEAR				
START 24	1	30	30	480
CLEAR				
START 24	0	30	50	480
CLEAR				
START 24	0	35	30	480
CLEAR				
START 24	0	10	20	480
CLEAR				
START 24	3	40	10	480
CLEAR				
START 24	1	25	10	480

Отсюда находим, что среднее число секций, пропускаемых за смену, значительно снизилось по сравнению с и исходным вариантом и составляет теперь около $11/12 \approx 1$.

5.7. Моделирование второго варианта мероприятий

Чтобы промоделировать сохранение задела деталей, образованного в конце смены, достаточно просто не использовать команду очистки CLEAR при переходе от одной смены к другой. В результате этого состояние модели при переходе к моделированию следующей смены не будет изменяться – следовательно, и длины очередей в момент начала новой смены будут те же самые, которые сложились в момент окончания предыдущей смены.

Более того, теперь нет необходимости моделировать весь процесс по отрезкам длиной в одну смену. Можно задать в команде START такое значение счетчика, чтобы промоделировать сразу тысячу смен. Накопленное за это время число пропущенных секций нужно будет разделить на тысячу, чтобы найти среднее число секций, пропускаемых за одну смену.

Поэтому вариант мероприятий по сохранению задела можно смоделировать с помощью следующих действий.

Восстанавливаем старую форму блока GENERATE. Для этого переходим в окно данных, вызываем текст модели на экран и редактируем строку с этим блоком.

Сбрасываем старую статистику с помощью команды CLEAR.

Выполняем непрерывное моделирование 1000 смен. Для этого просто запускаем модель командой START 24000.

Мы не перешли из окна данных в окно блоков, чтобы не замедлять моделирование.

После завершения модели смотрим значение числа пропущенных секций с помощью команды SHOW N$PROP (ответ: 24).

Делим это число на число промоделированных смен.

Получаем, что среднее число секций, пропускаемых за одну смену, составляет теперь 0,024.

Вывод: консервация производственного задела экономически целесообразней, чем сдвиг графика работы конвейера. Такая консервация позволяет практически полностью исключить пропуск секций.

Следовательно, данное мероприятие повысит производительность предприятия (число готовых изделий, выпускаемых за смену) примерно на 6% по сравнению с исходным вариантом.

5.8. Корректировка рекомендаций

В выбранном варианте мероприятий (втором) имеется одна особенность, которая связана с ростом очередей к конвейеру: со временем очереди будут постоянно увеличиваться (в среднем), т.к. стационарная средняя длина их равна бесконечности. Это приводит к определенным трудностям при планировании экспериментов и интерпретации их результатов. Чтобы убедиться в этом, выполним следующий эксперимент.

Войдем в GPSS\PC и загрузим последний вариант модели из файла, в котором он был сохранен.

Введем операторы

```
PLOT Q1,200,0,4800
START 240.
```

Первый оператор определяет вывод на экран графика изменений СЧА Q1 (т.е. длины первой очереди). Три числа в этом операторе соответствуют максимальному значению СЧА по оси ординат (200), минимальному его значению (0) и моменту времени, ограничивающему график по оси времени (4800 – это соответствует десяти сменам).

В результате выполнения этих команд на графике отобразится изменение Q1 во времени. График имеет примерно следующий вид (рис. 12).

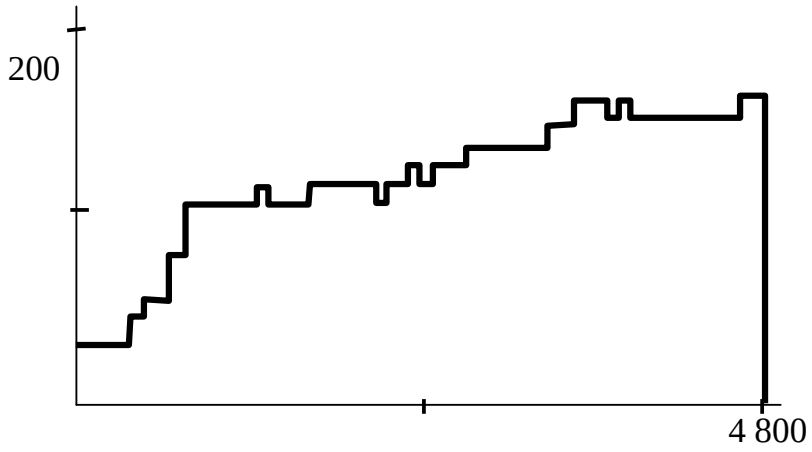


Рис 12.

Длину очереди Q1 после десяти смен можно посмотреть с помощью команды

```
SHOW Q1.
```

Она составляет в данном случае 150 деталей.

Если продолжать увеличивать время моделирования, то длина обеих очередей будет в среднем постоянно расти, достигая 1000 деталей, 2000 и т.д.

В реальных системах, однако, существуют факторы, которые обычно не учитываются простыми моделями, но в действительности ограничивают возможности изменения тех или иных характеристик в известных пределах. В случае с конвейером, очевидно, рост производственного задела в очередях тоже должен быть, так или иначе, ограничен.

Учитывая это, скорректируем модель конвейера таким образом, чтобы задел в очередях не превышал десятикратного запаса, т.е. запаса, достаточного для комплектации десяти секций конвейера. Это соответствует максимальной допустимой длине очередей Q1 и Q2, равной 100 деталям.

Исходя из этого, уточним наши рекомендации следующим образом. Задел деталей перед конвейером в конце каждой смены теперь рекомендуется сохранять, только если он не превышает 100 деталей (в каждой очереди). Если же в очереди в конце смены оказывается больше 100 деталей, то лишние детали (сверх сотни) из очереди изымаются.

Данную операцию по ограничению заделов учтем в модели, добавляя к ней следующие строки (после имеющихся строк).

```
      GENERATE     480
      TEST G       Q1,100,SECUND
      DEPART       1,Q1
      QUEUE        1,100
SECUND  TEST G       Q2,100,NEUTR
      DEPART       2,Q2
      QUEUE        2,100
NEUTR   TERMINATE
```

Здесь блок GENERATE каждые 480 единиц времени выдает транзакт, который проверяет длину каждой из очередей Q1 и Q2, и, если какая-либо из них больше 100, транзакт делает ее равной 100. Если длина очереди не превышает 100, то она не изменяется.

Выполняя модель в течение 10 000 смен, находим, что число пропусков за это время составило 2411, т.е. в среднем по 0,24 пропущенных секций за одну смену.

При уменьшении верхней границы задела с десятикратного запаса до четырехкратного, процент пропущенных секций возрастает более чем вдвое. Следовательно, рекомендация использовать ограничение очередей именно на уровне десятикратного запаса является существенной частью разработанного мероприятия по повышению производительности конвейера._

ГЛАВА 6. МОДЕЛЬ ТРАНСПОРТНОГО ЦЕХА

6.1. Постановка задачи

Транспортный цех объединения обслуживает три филиала A, B и C. Грузовики перевозят изделия из A в B и из B в C, возвращаясь затем в A без груза. Погрузка в A занимает 20 мин, переезд из A в B длится 30 мин, разгрузка и погрузка в B – 40 мин, переезд в C – 30 мин, разгрузка в C – 20 мин и переезд в A – 20 мин. Если к моменту погрузки в A или в B отсутствуют изделия, грузовики уходят дальше по маршруту. Изделия в A выпускаются партиями по 1000 штук через 20 ± 3 мин, в B – такими же партиями через 20 ± 5 мин. На линии работают 8 грузовиков, каждый перевозит по 1000 изделий. В начальный момент все грузовики находятся в A.

Смоделировать работу транспортного цеха объединения и определить частоту пустых перегонов грузовиков между A и B, B и C. Разработать мероприятия по уменьшению числа пустых перегонов.

Эта задача, с небольшой модификацией, взята из [6].

6.2. Схема процесса и предварительный анализ

Схема движения 8-ми грузовиков транспортного цеха изображена на рис.13.

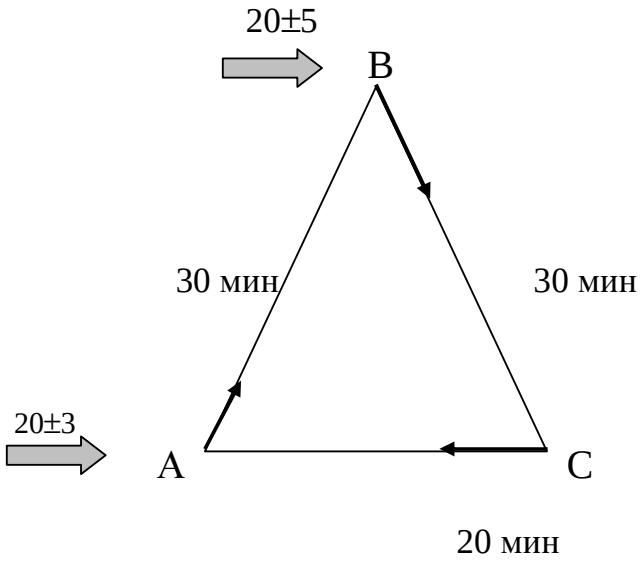


Рис.13

Серые стрелки показывают, что в пунктах A и B производятся партии изделий; время, через которое партии выставляются для перевозки, указано рядом со стрелками.

Предварительный анализ процесса перевозок показывает, что в самом начале обязательно несколько перегонов будут пустыми. Из пункта A все 8 грузовиков уйдут порожняком; пока они дойдут до пункта B, в нем появится одна партия. Один грузовик остановится на 20 мин для погрузки, остальные уйдут пустыми в пункт C и т.д.

Постепенно все грузовики рассосредоточатся по маршруту и частота пустых перегонов снизится.

В целом два взаимодействующих процесса – процесс выпуска партий и процесс их транспортировки – сбалансированы. Действительно, восьми грузовиков для развозки партий достаточно, но семи грузовиков было бы уже мало. Это можно видеть из следующих расчетов.

Предположим, что грузовики в стационарном режиме (т.е. достаточно времени спустя от начала процесса) движутся без пустых перегонов. Тогда один грузовик делает один круг за время, равное 160 минутам, т.к. 30 + 30 + 20 = 80 мин уходит на переезды между пунктами, и 20 + 40 + 20 = 80 мин – на погрузку и разгрузку в пунктах A, B и C. Итого время на один круг маршрута составляет 80 + 80 = 160 мин.

За 160 мин грузовик перевезет тогда 2 партии изделий, одну партию – в среднем за 80 мин. Следовательно, 8 грузовиков будут перевозить одну партию в среднем за 80/8 = 10 мин. Это время равно среднему времени между выпуском партий изделий: в пункте A партии выпускаются в среднем через 20 мин, и так же в пункте B. Вместе два пункта выдают одно изделие в среднем через 10 мин.

Таким образом, восьми грузовиков достаточно, чтобы успевать перевозить все производимые изделия (при условии, что грузовики не делают пустых перегонов). Очевидно, что если будут пустые перегоны, то грузовики не будут успевать развозить груз и он будет накапливаться в пунктах его производства.

Обратим внимание на то, что интенсивность выпуска партий и интенсивность их перевозки - при условии отсутствия пустых перегонов - в точности одинаковы. Это напоминает ситуацию, когда в СМО коэффициент загрузки равен единице. Поэтому можно предположить, что рассматриваемая система перевозок будет иметь соответствующую специфику, т.е. что стационарный режим для нее существует, но средняя длина очереди (в данном случае – число не перевезенных партий) равна бесконечности. Практически наличие такого стационарного режима означало бы, что мы можем моделировать только переходный процесс, т.к. длительность последнего бесконечна. В таком случае реальная система никогда не войдет в стационарный режим, хотя он - аналитически - существует. В аналитическом

смысле система войдет в стационарный режим лишь через время, равное бесконечности.

Если высказанное предположение подтвердится, то необходимо будет рассмотреть возможность создания некоторого резерва для ресурса перевозок, т.е. для снижения загрузки системы.

Увеличить ресурс перевозок можно, увеличивая число грузовиков или длительность одной смены грузовиков по сравнению со сменой производства изделий, или, например, формируя время от времени дополнительные рейсы грузовиков.

6.3. Разработка модели

Текст модели с комментариями приводится ниже.

В начальный момент времени все блоки GENERATE начинают выполняться одновременно. Поэтому во всех сегментах модели (разделенных строками-комментариями) транзакты движутся - в модельном времени - одновременно. Взаимодействия между первым и вторым сегментами осуществляются через ячейки X1 и X2. Во втором сегменте эти ячейки пополняются (товар производится), в первом их содержимое разбирается транзактами (продукция развозится). В первом сегменте из блока GENERATE *одновременно* выходит 8 транзактов в нулевой момент времени. Но *логически* один из них приходит в блок TEST прежде других (хотя и в один и тот же момент времени). Он может «забрать груз». Следующий транзакт, приходящий в тот же момент модельного времени, но логически после первого, может «не застать груз» и из блока BEG TEST сразу уйти «в путь к В».

```
5 EMPTY     VARIABLE     N3-N4+N12-N13 ;число пустых перегонов
10          GENERATE     ,,,8          ;выход 8-ми грузовиков
20          ASSIGN       1,0           ;помечаем грузовики как пустые
30 BEG      TEST G       X1,0,GOTOB    ;проверка наличия груза в п.А
40          SAVEVALUE    1-,1          ;забрать груз для погрузки
50          ADVANCE      20            ;задержка 20 мин для погрузки
60          ASSIGN       1,1           ;помечаем: грузовик с грузом
70 GOTOB    ADVANCE      30            ;30 мин переезд в п.В
80 RAZGR    TEST G       P1,0,POGR     ;проверка: нужно разгружаться?
90          ADVANCE      20            ;разгрузка 20 мин
100         ASSIGN       1,0           ;помечаем грузовик как пустой
110 POGR    TEST G       X2,0,GOTOC    ;проверка наличия груза в п.В
120         SAVEVALUE    2-,1          ;забрать груз для погрузки
130         ADVANCE      20            ;разгрузка 20 мин
```

```
140          ASSIGN       1,1             ;помечаем: грузовик с грузом
150 GOTOC    ADVANCE      30              ;30 мин переезд в п.С
160          TEST G       P1,0,GOTOA      ;проверка: нужно разгружаться?
170          ADVANCE      20              ;разгрузка 20 мин
180          ASSIGN       1,0             ;грузовик пуст
190 GOTOA    ADVANCE      20              ;20 мин переезд в п.А
200          TRANSFER     ,BEG            ;начинаем новый круг
```

*** Производство перевозимой продукции ***

```
210          GENERATE     20,3            ;выпуск партий в п. А
220          SAVEVALUE    1+,1            ;число партий - в ячейке X1
230          TERMINATE
240          GENERATE     20,5            ;выпуск партий в п. В
250          SAVEVALUE    2+,1            ;число партий - в ячейке X2
260          TERMINATE
```

*** Таймер для останова процесса моделирования ***

```
270          GENERATE     480             ;одна смена = 480 мин.
280          TERMINATE    1
```

*** Создание микроокон для наблюдения за показателями ***

```
290          MICROWINDOW   1,V$EMPTY      ;окно числа пустых перегонов
300          MICROWINDOW   2,X1           ;окно остатка в п. А
310          MICROWINDOW   3,X2           ;окно остатка в п. В
```

Ячейки X1 и X2 общедоступны для всех транзактов, как и продукция в п. А и В является общей для всех грузовиков. Значения этих двух ячеек изменяются в блоках SAVEVALUE. Значение параметра транзакта P1 у каждого транзакта свое, как и состояние каждого грузовика - индивидуальное.

В строке 5 описана переменная, значение которой в любой момент модельного времени равно числу пустых перегонов грузовиков. Эта переменная вычисляется через СЧА Nj, который равен числу транзактов, вошедших в блок номер j. Число пустых перегонов равно сумме числа пустых перегонов на пути АВ и числа пустых перегонов на пути ВС. Число пустых перегонов на пути АВ определяется как N3 –N4, т.е. как число грузовиков,

прибывших в п.А и проверивших наличие в нем груза, минус число грузовиков, заставших в п.А груз и, соответственно, загруженных. Аналогично определяется число пустых перегонов между пунктами B и C, равное N11 – N12.

6.4. Планирование эксперимента

В процессе планирования эксперимента следует решить два вопроса. Во-первых, необходимо определить, какой процесс мы должны исследовать – переходный или стационарный. Во-вторых, для выбранного типа процесса нам следует найти длину прогона модели (если исследуется стационарный процесс) или число независимых реализаций процесса (если исследуется переходный процесс), чтобы обеспечить достаточно малую для решения задачи статистическую погрешность результатов моделирования.

Решение о типе процесса обычно принимается на основе содержательного анализа задачи. Так, в данном случае очевидно, что моделируемая система функционирует исключительно в состоянии переходного процесса. Каждый день грузовики выходят из п. А в начале смены одновременно. В конце смены производство продукции завершается, и грузовики остаются в гараже (в п. А) до начала следующей смены. Затем все повторяется сначала.

6.5. Имитационный эксперимент

Таким образом, мы можем получить представление о работе моделируемой системы, выполняя модель несколько раз (моделируя несколько смен) и сопоставляя результаты разных смен между собой.

Моделируя таким образом смену за сменой (используя для возврата модели в исходное состояние команду CLEAR), находим, что число пустых перегонов грузовиков равно 11 или 12. При этом остаток не перевезенной продукции X1 + X2 составляет от 3 до 5 партий изделий.

Выполненный выше предварительный анализ показал, что большое число пустых перегонов обусловлено тем, что все восемь грузовиков выходят в рейс одновременно. Конечно, это довольно нелепо – отправлять в рейс сразу восемь грузовиков, зная заранее, что для перевозки будет готова максимум одна партия изделий.

Поэтому посмотрим, что будет, если мы будем выпускать грузовики в рейс один за другим с интервалом в 20 мин, – т.е. через такое время, которое в среднем нужно для выпуска новой партии продукции.

Для этого в блок GENERATE ,,,8 внесем изменение – запишем его как GENERATE 20,,,8, что соответствует выходу грузовиков друг за другом через 20 мин.

Моделируя этот вариант расписания, находим, что число пустых перегонов за смену сократилось до 2-х, но число не перевезенных партий за смену в среднем несколько возросло – примерно на 1 – 2 партии по той, очевидно, причине, что каждый грузовик в среднем вышел в смену более чем на час позже, чем в первом варианте.

6.6. Критический анализ результатов

Особенности планирования эксперимента с моделями СМО при коэффициенте загрузки, равном единице, подсказывают нам, что в задаче о перевозках имеется тонкость, состоящая в том, что средняя длина очереди, – т.е. среднее число не перевезенных партий изделий – с течением времени будет стремиться к бесконечности. Чтобы проверить это предположение, проведем моделирование процесса перевозок с условием, что производство продукции и движение грузовиков осуществляются непрерывно в течение бесконечного времени. Это соответствует тому, как если бы грузовики, перевозя продукцию, в конце смены «замирали» в том месте пути, в котором их застало окончание смены. Следующую смену они начинали бы тогда именно из этой точки. Кроме того, при таком моделировании сохраняется задел изделий в пунктах производства.

Моделируя смену за сменой без сбрасывания результатов командой CLEAR, мы получим для 100 смен подряд следующие результаты. Число пустых перегонов - во всех 100 сменах - в каждой смене равно 2. Число не перевезенных партий продукции меняется от смены к смене следующим образом: оно составляет последовательно 5,9,9, … ,10,11,14, …,7.

Из этого опыта может сложиться впечатление, что в стационарном процессе при таком расписании, – когда грузовики выходят в первый рейс через 20 мин друг за другом – среднее число пустых перегонов за смену сократилось в среднем с 12 до 2, и что количество не перевезенных партий составляет в среднем около 9 – 10 за 100 смен.

Но если мы продолжим опыт при большей длине прогона модели, то обнаружим рост как числа пустых перегонов, так и – главное - числа не перевезенных партий.

Так, число пустых перегонов за 10000 смен составляет уже около 49, а число не перевезенных партий изделий – около 300.

Разумеется, обнаруженная тонкость, связанная с ростом очередей, имеет исключительно теоретическое значение. Вряд ли кто-либо из сотрудников транспортного цеха на практике обеспокоился бы из-за того, что имеется вероятность накопления очереди на 20 поездок восьми грузовиков, если цех проработает в неизменных условиях в течение 10000 дней, - т.е. почти 30 лет.

6.7. Практические рекомендации

Мы можем разрабатывать десятки вариантов модели перевозки грузов по филиалам A,B и C.

Например, мы можем обратить внимание на то, что смена по производству продукции может закончиться раньше, чем смена грузовиков по ее перевозке. Это значит, что необходимо уточнить правило моделирования перевозок. Тогда мы должны будем выпускать грузовики друг за другом через 20 мин, филиалы в пп. A и B должны будут работать в модели ровно 8 час, и грузовики должны будут делать *ровно три круга*, начинающихся и кончающихся в п. A. Это означает, что грузовики могут заканчивать свою смену позже, чем филиалы.

Однако, моделируя такой вариант, мы обнаружим, что грузовики в среднем делают свою смену – три круга – ровно за 480 мин (т.е. 8 час), и что со временем по прежнему возрастают как число пустых перегонов, так и число не перевезенных партий изделий.

Мы можем также промоделировать вариант, когда грузовики в пунктах A и B, не обнаруживая груза, останавливаются до тех пор, пока груз не появится (естественно, исключая случай, когда соответствующий филиал уже закончил свою смену). Но и в этом случае со временем будут расти как число пустых перегонов, так и число не перевезенных партий продукции.

Продолжая такие поиски стратегий, мы, в конце концов, убеждаемся, что ни одна стратегия не позволяет полностью исключить накопление числа пустых перегонов и числа не перевезенных партий. При этом мы будем наблюдать такую закономерность, что некоторое снижение числа пустых перегонов, достигаемое в какой-либо конкретной стратегии, вызывает рост не развезенной продукции. Практические рекомендации, вытекающие из этих опытов, оказываются довольно тривиальными. Они сводятся к следующему.

Грузовики следует выпускать через 20 мин друг за другом.

Каждый грузовик должен сделать три круга и вернуться в исходный п. A.

Время от времени могут понадобиться дополнительные рейсы грузовиков, чтобы развезти накопившуюся в пп. А и В продукцию.

Все эти рекомендации вытекают, по сути дела, из того, что загрузка ресурса перевозок в данной системе равна единице. Эти выводы можно было бы сделать и без имитационного моделирования.

Подобная ситуация весьма типична при моделировании реальных систем: мы нередко приходим к выводам, которые выглядят для нас совершенно тривиальными.

Однако следует обратить внимание на то, что эти выводы возникают не иначе, как в процессе моделирования: мы формализуем задачу, анализируем ее, программируем имитационную модель. … И вдруг, на каком-то этапе, обнаруживаем, что мы уже знаем решение задачи, хотя еще не получили всех необходимых численных результатов имитационного эксперимента.

Получается, что для решения задачи как будто и не нужно было выполнять имитационное моделирование.

И все же во всех подобных случаях решение задачи получается у нас не иначе, как в процессе разработки имитационной модели системы и в ходе каких-то экспериментов с нею. Следовательно, польза имитационного моделирования не сводится только лишь к получению точных численных результатов экспериментов. Имитационное моделирование позволяет нам лучше понять исследуемую систему уже на самых первых этапах, дает нам опыт анализа, умение обращать внимание на самые существенные свойства и параметры систем, воспитывает нашу «системную» интуицию.

Перефразируя Ломоносова, можно сказать так: модель уже затем строить нужно, что она ум в порядок приводит.

ГЛАВА 7. ОПИСАНИЕ ЯЗЫКА GPSS\PC

7.1. О синтаксисе, семантике и прагматике языка GPSS\PC

GPSS\PC – язык имитационного моделирования, близкий по своим возможностям к языку GPSS-V. Можно рассматривать его как версию GPSS-V, адаптированную для персональных ЭВМ.

В алфавит языка GPSS\PC входят следующие символы:

A, B, ..., Z - большие буквы латинского алфавита,
1, 2, ..., 9, 0 - цифры,
? – вопросительный знак (для вызова помощи),
@ - коммерческое по (знак остатка от деления),
# - решетка (номер) - используется как *знак умножения*,
$ - знак доллара,
^ - «карет» - знак возведения в степень,
+ - плюс,
- - минус,
_ - подчеркивание,
***** - звездочка,
/ - косая черта (знак деления),
(- левая скобка,
) - правая скобка,
‘ - апостроф,
. - точка,
, - запятая,
␣ - пробел.

Формальные правила написания программ на языке GPSS\PC образуют его *синтаксис*.

При моделировании между системой, описываемой на языке GPSS *(GPSS-системой),* и моделируемой реальной системой устанавливают определенное логическое соответствие, которое позволяет из сведений, получаемых при анализе GPSS-системы, делать определенные выводы о реальной системе. Такие выводы называют *интерпретацией* GPSS-систем. Правила интерпретации GPSS-систем составляют *семантику* языка GPSS. В отличие от синтаксиса семантика является неформальной стороной языка и потому не может быть описана с математической однозначностью. Как правило, семантику описывают и изучают с помощью примеров интерпретации разнообразных языковых конструкций.

Наряду с синтаксическим и семантическим аспектами языка большое значение имеет его прагматический аспект. *Прагматика* языка включает сведения по его практическому использованию, в том числе типичные приемы написания программ на этом языке, методы организации их отладки и выполнения и т.д. Необходимые сведения такого рода будут приводиться ниже по мере надобности.

7.2. Классы объектов языка GPSS\PC

Описываемые на GPSS\PC абстрактные системы строятся из набора простых объектов. Эти объекты делятся на четыре класса: динамические, оборудование, статистические и операционные.

Динамические объекты называются *транзактами*. Эти объекты могут возникать и уничтожаться. Транзакты движутся в системе и производят при движении ряд действий. Транзакты могут также «размножаться» и «склеиваться».

Каждый транзакт имеет определенное число параметров, значения которых могут меняться в результате выполняемых действий.

*Пример*1. В табл. 4 приводятся некоторые интерпретации транзактов.

Таблица 4

Моделируемая система	Интерпретация транзакта
Кинотеатр	Зритель
ЭВМ	Задание
Универсам	Покупатель
Сеть перекрестков	Автомобиль
Сеть ЭВМ	Сообщение
Производственный участок	Изделие

Оборудование в GPSS\PC подразделяется на следующие типы: устройства, памяти и логические ключи. Оборудование подвергается определенным воздействиям со стороны транзактов.

Устройство может находиться в одном из трех состояний: свободном, занятом или захваченном. В начальный момент все устройства свободны. Переходы устройства из одного состояния в другое происходят в результате воздействия транзактов, которые могут соответственно освобождать, занимать или захватывать устройства. Устройство в каждый момент времени может быть занято или захвачено не более чем одним транзактом.

*Пример*2. В табл.5 приводятся некоторые из возможных интерпретаций устройств.

Таблица 5

Моделируемая система	Интерпретация устройства
Кинотеатр	Кассир
ЭВМ	Процессор
Универсам	Продавец
Сеть ЭВМ	Канал связи
Производственный участок	Станок

Память имеет емкость, выражаемую целым числом единиц. Ее состояние определяется числом занятых единиц. Транзакт может занимать и освобождать любое число единиц памяти, лишь бы оно было неотрицательным и не превышало объема памяти. Память могут занимать одновременно несколько транзактов. В исходном состоянии системы все памяти свободны.

*Пример*3. В табл.6 приводятся элементы реальных систем, которые могут соответствовать памяти.

Таблица 6

Моделируемая система	Интерпретация памяти
Кинотеатр	Зрительный зал
ЭВМ	Запоминающее устройство
Универсам	Место для очереди
Сеть ЭВМ	Буферная память
Производственный участок	Бункер для изделий

Логический ключ имеет 2 состояния: S (включен) и R (выключен). Состояния ключа устанавливаются одними транзактами и могут проверяться другими. Результаты проверки могут учитываться в последующих действиях транзактов. В начальном состоянии все ключи в системе выключены.

Пример 4. При моделировании сети перекрестков ключ может соответствовать светофору.

Представителями *статистических* объектов являются очереди и таблицы. Эти объекты вводятся в модель в подавляющем числе случае не с целью отображения каких-либо реальных объектов, а только для сбора необходимой пользователю статистики о модели. Следовательно, статистические объекты ориентированы не на семантику, а на прагматику языка GPSS\PC и для них нельзя привести таблицу, аналогичную табл. 4-6.

Каждая *очередь* в GPSS-системе имеет выход и вход. Очередь подсчитывает статистические данные о проходящих через нее транзактах: число проходящих транзактов, среднее время задержки транзактов в очереди и т.д.

Таблица может для любой указанной пользователем величины фиксировать и обрабатывать последовательность ее значений. По наблюдаемым значениям указанной величины таблица вычисляет определенную статистическую информацию об этой величине: ее математическое ожидание, среднее квадратичное отклонение и т.д.

Операционные объекты в GPSS\PC называются *блоками*. Блоки являются ячейками пространственной структуры GPSS-системы: ее функциональное пространство имеет вид последовательности блоков. Транзакты в системе всегда находятся в каких-либо ее блоках. Движение транзактов в системе есть их перемещение из одних блоков в другие. В то же время каждый блок представляет собой некоторый алгоритм преобразования объектов системы. Этот алгоритм выполняется, если транзакт находится в блоке. Число типов блоков в GPSS\PC составляет 46.

Динамика GPSS-системы характеризуется тем, как проходят изменения состояний ее объектов во времени.

Все объекты системы изменяют свои состояния скачкообразно. Модельное время складывается из интервалов постоянства состояний объектов и из отдельных моментов мгновенного изменения состояний. Изменения состояний объектов называются *событиями*. Каждое событие происходит в момент перемещения какого-либо транзакта из одного блока в другой. Последовательность разделенных интервалами времени мгновенных событий, происходящих при движении в модели фиксированного транзакта, называется процессом.

7.3. Общеалгоритмические средства GPSS\PC

Прежде чем начать изучение основных объектно-ориентированных средств языка GPSS\PC, целесообразно ознакомиться с его общеалгоритмической частью. Общеалгоритмические средства GPSS\PC включают способы записи постоянных и переменных величин, способы указания вычислений и запоминания их результатов и т.д.

Для любого общеалгоритмического средства GPSS\PC можно найти аналог почти в любом алгоритмическом языке. Поэтому автор рекомендует изучающим язык GPSS\PC «проводить параллели» с каким-либо алгоритмическим языком высокого уровня. Такое проведение параллелей

может заметно облегчить и одновременно углубить ознакомление с общеалгоритмическими возможностями языка GPSS.

7.4. Стандартные числовые атрибуты

Алгоритм, выполняемый при входе транзакта в блок, определяется типом блока, указанным в поле операции, и операндами, которые записаны в поле операндов. Операнды дают выполняемому алгоритму информацию о текущем состоянии объектов системы в виде значений *стандартных числовых атрибутов* (СЧА).

В общем случае любой СЧА имеет вид:

tj

где t - идентификатор выражаемой атрибутом характеристики, j - указание, к какому именно объекту относится данная характеристика. Например, СЧА, значение которого равно текущему содержимому (числу занятых единиц) памяти номер 1, имеет вид:

Q1 ,

где буква Q обозначает «текущая длина очереди», а число 1 указывает, что имеется в виду очередь номер 1.

В общем случае идентификатор характеристики t состоит из одной или двух букв, а указатель j является либо номером объекта (как в S1), либо присоединенным именем объекта, либо косвенным номером объекта.

Присоединенное имя состоит из знака $ и метки объекта. Так, значение атрибута

S$MEM

равно текущему содержимому памяти, которая имеет метку MEM.

Косвенный номер объекта состоит из знака * и номера параметра транзакта. При этом имеются в виду параметры того транзакта, который обрабатывается в данный момент времени. В качестве косвенного номера объекта принимается содержимое указанного параметра транзакта. Например, значение атрибута

Q*2

равно текущей длине очереди, номер которой записан во втором параметре транзакта.

Примеры СЧА:

R2 - число свободных единиц памяти номер 2;

SM$MEM1 - максимальное содержимое памяти, имеющей метку MEM1;

F70 - состояние устройства 70; F70 равно нулю, если устройство 70 свободно и единице в остальных случаях;

P3 - содержимое третьего параметра транзакта;

P*3 - содержимое того параметра транзакта, номер которого записан в третьем параметре этого параметра;

PR - приоритет транзакта;

C1 - текущее значение модельного времени;

W179 - число транзактов, находящихся в данный момент в 179 -ом блоке модели;

W$ABC00 - число транзактов в блоке, имеющем метку ABC00;

Q1 - длина очереди номер 1;

FN$NORM - значение функции с меткой NORM;

V$COST - значение арифметической переменной с меткой COST;

RN1 - псевдослучайное число, равномерно распределенное в интервале от 0 до 1000. В GPSS\PC имеется практически неограниченное число таких СЧА (RN1, RN2, …, RN100…). Для вычисления каждого из этих СЧА используется свой генератор псевдослучайных чисел.

7.5. Арифметические переменные

Для большей универсальности языка GPSS\PC у некоторых СЧА стандартизировано лишь обозначение, а смысл этих СЧА и алгоритмы их вычисления определяются пользователем при описании модели. К таким СЧА относятся арифметические переменные. Обозначение арифметической переменной имеет вид V$*name*. Использовать любой конкретный СЧА вида V$*name* в качестве операнда можно лишь при условии, что этот СЧА описан пользователем в данной модели.

Описание атрибута V$*name* осуществляется в строке описания арифметической переменной. В одной строке описывается одна переменная. В поле метки обязательно записывается имя описываемой переменной, в поле операции записывается слово VARIABLE или FVARIABLE, а в поле операндов - арифметическое выражение, составляемое из СЧА, знаков арифметических операций и круглых скобок.

В арифметическом выражении можно использовать следующие знаки арифметических операций:

\+ - сложить,

\- - вычесть,

\# - умножить,

/ - разделить,

@ - взять остаток от деления,

^ - возведение в степень.

Пусть, например, переменная V$SUM описана строкой

```
SUM     VARIABLE    S1+S2.
```

Тогда значение V$SUM будет определяться как суммарное содержимое первой и второй памяти. Если переменную V$CAP описать в виде

```
CAP     VARIABLE    R*8+5*8,
```

то её значением будет сумма числа свободных и числа занятых единиц той памяти, номер которой записан в восьмом параметре транзакта.

При описании арифметических переменных необходимо иметь в виду следующее.

1. Равноприоритетные операции в арифметическом выражении при отсутствии скобок выполняются слева направо. Операции #, /, @ имеют приоритет по отношению к операциям +, -.

2. Если в строке описания арифметической переменной в поле операции записано VARIABLE, переменная считается целой, в случае записи FVARIABLE - вещественной. При вычислении *целой* переменной от результатов всех промежуточных операций берется целая часть. При вычислении *вещественной* переменной этого не делается. Однако окончательный результат округляется в меньшую сторону, как в целой, так и в вещественной переменной. Например, если в модель включены строки

```
ABC1    FVARIABLE     10#(11/3)
ABC2    VARIABLE      10#(11/3)
ABC3    FVARIABLE     1/5
```

то описанные в этих строках переменные будут иметь следующие значения:

```
V$ABC1 =  36,
V$ABC2 =  30,
V$ABC3 =  0.
```

3. Синтаксис арифметических выражений соответствует синтаксису, принятому в языках программирования высокого уровня. Например, аргументы стандартных функций типа SIN, EXP,SQR, и т.п. должны заключаться в скобки.

4. Синтаксис арифметических выражений в переменных такой же, как в команде SHOW.

7.6. Булевские переменные

Обозначение булевской переменной имеет вид BV$*name*. Этот СЧА, как и арифметическая переменная, описывается в тексте модели пользователем. Булевская переменная описывается в *строке описания булевской переменной*. В одной строке описывается одна переменная. В поле метки обязательно записывается имя переменной, в поле операций - слово BVARIABLE, а в поле операндов - булевское выражение. Булевское выражение составляется из элементов пяти типов: СЧА, отношений, *стандартных логических атрибутов* (СЧА), знаков логических операций и скобок. Значением булевской переменной является число 1, когда булевское выражение истинно и число 0 - в противном случае. Подчеркнем, что значением булевской переменной являются именно числа.

Отношение представляет собой запись двух СЧА, соединенных одним из знаков:

‘G’ - больше чем,
‘L’ - меньше чем,
‘E’ - равно,
‘NE’ - не равно,
‘LE’ - меньше или равно,
‘GE’ - больше или равно.

Отношение имеет значение “истинно” в том и только в том случае, если значения входящих в него СЧА отвечают смыслу соединяющего знака. Например, переменные BV$LOG1 и BV$LOG2 , описанные строками:

```
LOG1    BVARIABLE    V$ABC1’G’5
LOG2    BVARIABLE    V$ABC1’NE’V$ABC2,
```

всегда равны единице, если V$ABC1 и V$ABC2 описаны так, как в предыдущем параграфе.

Если СЧА в логическом выражении не входит в отношение, то ему приписывается логическое значение (в этом случае СЧА должен быть заключен в скобки). Логическое значение “ложно” приписывается в случае, если численно СЧА равен нулю, во всех остальных случаях приписывается значение “истинно”.

Стандартные логические атрибуты (СЛА) в GPSS\PC отражают состояние оборудования в модели. Это следующие атрибуты.

СЛА устройств:

FIj - устройство j захвачено,
FNIj - устройство j не захвачено,

FUj - устройство j занято,
FNUj - устройство j свободно,
Fj - устройство j не свободно,
FVj – устройство j доступно,
FNVj – устройство j недоступно.

СЛА памятей:

SFj - память j заполнена,
SNFj - память j не заполнена,
SEj - память j пуста,
SNEj - память j не пуста,
SVj – память j доступна,
SNVj – памятьj недоступна.

СЛА ключей:

LSj - логический ключ j включен,
LRj - логический ключ j выключен.

СЛА блоков MATCH:

M – в j-ом блоке синхронизации есть транзакт того же семейства,
NM – в j-ом блоке синхронизации нет транзакта того же семейства.

Последние два СЛА могут быть использованы только в блоке GATE.

СЛА имеет значение “истинно” тогда и только тогда, когда выполняются соответствующие ему условие - в приведенном перечне оно записано напротив вида индикатора. Указатель j в обозначении индикатора задается так же, как и в обозначении СЧА, т.е. может быть номером, косвенным номером или именем объекта.

Знаки логических операций в описании булевской переменной - это знак логического сложения (+) и знак логического умножения (#). Операция умножения имеет приоритет по отношению к операции сложения. Например, булевская переменная BV$RESURS, описанная строкой:

```
RESURS    BVARIABLE     FNU1#FNU2+SNF1#SNF2,
```

равна единице тогда и только тогда, когда устройства 1 и 2 оба свободны, или памяти 1 и 2 обе не заполнены.

Следует иметь в виду, что, хотя все промежуточные операции в булевской переменной имеют *логические* значения «истинно» или «ложно», сама булевская переменная имеет *числовое* значение 0 или 1.

7.7. Функции

Функция - это СЧА, обозначаемый в виде FN$*name* и описываемый пользователем в виде численной зависимости FN$*name* от другого СЧА. Рассмотрим на примерах, как это делается.

Пусть необходимо, чтобы значение FN$KL1 вычислялось в модели через S$MEM в соответствии с графиком, изображенным на рис. 14.

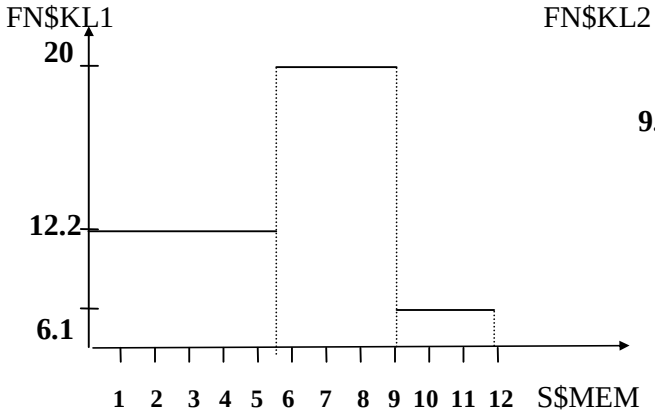


Рис. 14

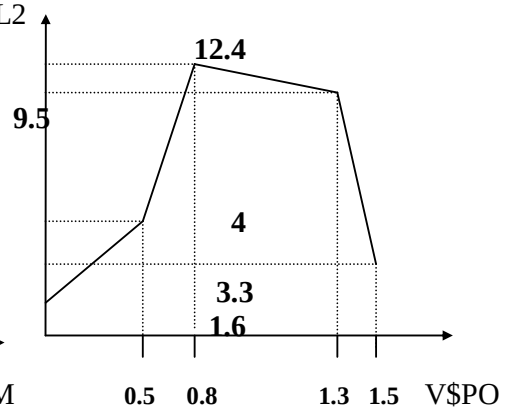


Рис. 15

Функция, вычисляемая в соответствии с таким кусочно-постоянным графиком, называется дискретной функцией. Она описывается следующими строками:

```
    KL1     FUNCTION        S$MEM,D3
5.5,12.2,/9,20/12,6.1.
```

Первая строка описания функции должна содержать метку функции в поле метки, слово FUNCTION в поле операции и два операнда в поле операндов. Первый операнд является обозначением того СЧА, который выбран в качестве аргумента функции. В данном примере это S$MEM – количество занятых единиц памяти MEM. Второй операнд состоит из буквы D, обозначающей, что данная функция дискретная, и из числа точек, которыми задается график функции. Координаты точек, задающих график, записываются в последующих строках описания функции. Из примера видно, что каждая точка задается парой координат, разделенных запятой, причем вначале указывается абсцисса точки, потом ордината. Координаты разных точек разделяются косой чертой. Функция может задаваться любым числом точек.

Запись координат точек начинается с 1-й позиции строки и может быть продолжена на последующих строках.

Перенос записи координат на следующую строку следует производить, начиная с новой пары координат. При этом ту пару координат, которая оказывается на строке последней, не обязательно заканчивать разделяющей косой чертой.

Абсциссы точек графика, указываемые в строках описания функции, должны быть все упорядочены по возрастанию.

Следует обратить внимание на то, что в описании дискретной функции задаются только те точки, которые расположены на правых концах горизонтальных отрезков.

Кроме того, если значение аргумента в процессе моделирования окажется за пределами диапазона, охваченного графиком, то соответствующий крайний отрезок графика автоматически “продолжается” до нужного места. Так что описанная выше дискретная функция FN$KL1 будет иметь в зависимости от S$MEM следующие значения:

$$FN\$KL1 = \begin{cases} 12.2, & \text{при } -\infty < S\$MEM < 5.5, \\ 20, & \text{при } 5.5 < S\$MEM < 9, \\ 6.1, & \text{при } 9 < S\$MEM < \infty. \end{cases}$$

График функции FN$KL2, изображенный на рис.15, получен путем соединения нескольких точек ломаной линией. Функция, задаваемая таким графиком, называется непрерывной. Функция FN$KL2 в данном случае описывается строками:

```
    KL2     FUNCTION            V$PO,C5
0,1.6/0.5,4/0.8,12.4/1.3,9.5/1.5,3.3.
```

Как видим, отличие описания непрерывной функции от дискретной состоит лишь в том, что в первой строке вместо признака D дискретной функции должен быть признак C непрерывной функции.

Если при вычислении непрерывной функции значение ее аргумента выходит за диапазон, заданный в графике, то график автоматически “продолжается” горизонтальными прямыми: влево - на уровне первой заданной точки графика, вправо - на уровне последней.

Важным частным случаем рассмотренных видов функций являются функции с аргументом RNj. Если в строке FUNCTION в поле A записан аргумент RNj , то подразумевается, что в действительности используется не сам

СЧА RNj , а полученное с его помощью псевдослучайное число, заключенное в диапазоне от 0 до 0,999999 включительно, т.е. RNj берется в масштабе 1:1000. Когда аргументом функции является случайное число, то функция также будет принимать случайные значения. Функция от аргумента RNj используется для моделирования случайных величин, имеющих требуемые законы распределения вероятностей. Например, дискретная функция, описанная строками:

```
    DIS   FUNCTION    RN1,D4
0.1,1/0.3,5/0.4,7/1.0,8,
```

будет принимать значение 1 с вероятностью 0,1, значение 5 с вероятностью 0,2, значение 7 с вероятностью 0,1 и значение 8 с вероятностью 0,6.

Действительно, график функции FN$DIS имеет вид, показанный на рис.16. Поэтому функция FN$DIS примет при ее очередном вычислении значение 1, если RN1 попадет в интервал от 0 до 0,1. Вероятность этого равна 0,1. Значение 5 для FN10 будет получено с вероятностью попадания RN1 в интервал от 0,1 до 0,3. Эта вероятность равна 0,3 - 0,1 = 0,2. Аналогично определяются вероятности значений 7 и 8.

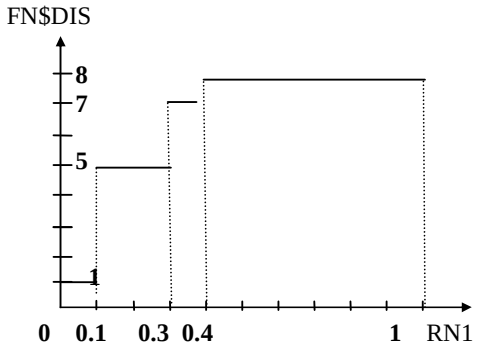


Рис.16

Дискретные функции от RNj применяются для моделирования дискретных случайных величин, непрерывные для непрерывных. Реализация непрерывных случайных величин осуществляется методом обращения [8]. Функция от RNj в этом случае является обратной по отношению к требуемой функции распределения вероятностей моделируемой случайной величины.

*Пример*5. Функция с меткой EXP , если ее описать так, как показано ниже, будет иметь случайные значения, распределенные по экспоненциальному закону с математическим ожиданием единица:

```
EXP          FUNCTION        RN1,C24
0,0/1,.104/.2,.222/.3,.355/.4,.509/.5,.69
.6,.915/.7,1.2/.75,1,38/.8,1.6/.84,1.83/.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/.998,7/.9997,8.
```

Поскольку в GPSS\PC алгоритмические средства богаче, чем в первых версиях языка GPSS, и содержат стандартную операцию вычисления натурального логарифма, то можно задавать преобразование RN1 в экспоненциальную случайную величину без такого громоздкого набора чисел, как в приведенном примере функции FN$EXP. Аналитически такое преобразование выражается формулой, приводимой в [8]:

$$y = - M \times ln\,(z),$$

где z – равномерная случайная величина, заданная на интервале (0,1),

M – произвольный положительный числовой множитель,

y – функция от z, (т.е. тоже случайная величина). Распределение случайной величины y будет экспоненциальным с математическим ожиданием M [8].

Такое преобразование равномерной случайной величины в экспоненциальную можно осуществить с помощью арифметической переменной FVARIABLE.

С другой стороны применение в GPSS функций FUNCTION позволяет значительно ускорить вычисления. Это ускорение обеспечивается за счет того, что числовая информация о функции FUNCTION представляется в памяти ЭВМ в виде таблицы, в которой для возможных значений аргумента уже хранятся готовые, заранее заданные значения функции или же они могут быть определены с помощью простой линейной интерполяции.

В GPSS\PC наряду с рассмотренными двумя видами функций (дискретной и непрерывной) имеются ещё несколько более специальных видов функций.

7.8. Ячейки и матрицы ячеек

Для того чтобы в модели можно было организовать запоминание каких-либо данных, в языке GPSS\PC имеются специальные СЧА, называемые *ячейками* сохраняемых величин или ячейками SAVEVALUE. Ячейка имеет обозначение вида Xj, где j – номер или имя ячейки.

Запись значений в ячейку осуществляется при входе транзакта в блок SAVEVALUE. Поле A этого блока интерпретируются как номер, косвенный номер или имя ячейки, поле B - как записываемое в нее значение.

Если в поле A записан СЧА, то его значение рассматривается как номер ячейки.

Пусть, например, транзакт проходит в модели через следующую цепочку блоков:

```
SAVEVALUE  OPERATION_NUMBER,157
SAVEVALUE  1,X$OPERATION_NUMBER
SAVEVALUE  X1,C1.
```

Тогда будут выполнятся следующие действия:
- запись числа 157 в ячейку X$OPERATIONS_NUMBER;
- запись числа 157 в ячейку 1;
- запись текущего времени в ячейку номер 157.

В начальный момент времени, когда в ячейки еще ничего не записывалось, все они имеют нулевое значение.

Если к обозначению СЧА в поле A блока SAVEVALUE приписать знак + или -, то в ячейке вместо *записи* значения будет производиться *добавление* или, соответственно, *вычитание* этого значения. Например, прохождение транзактом блоков

```
SAVEVALUE     11+,S1
          SAVEVALUE     12-,V$DIFFERENCIAL
```

вызовет добавление числа занятых единиц памяти номер 1 в ячейку 11 и вычитание переменной DIFFERENCIAL из ячейки номер 12.

Наряду с ячейками в моделях на языке GPSS\PC можно использовать матрицы ячеек. В отличие от простых ячеек матрицы перед использованием должны быть описаны. Для описания матрицы применяется *строка описания матрицы*. В поле метки этой строки записывается имя описываемой матрицы, в поле операции - слово MATRIX, в поле операндов - параметры матрицы: в поле A записывают любое слово или оставляют поле пустым, в поле B указывают число строк матрицы, в поле C - число столбцов. Например, описать матрицу ITOGI размерами 2×4 можно с помощью строки:

```
ITOGI    MATRIX    ,2,4.
```

В начальный момент любая матрица содержит только нулевые значения.

После того как матрицы описаны, можно использовать в качестве СЧА индексированные переменные - ячейки этих матриц. Ячейка матрицы имеет обозначение вида MX$*name*(a,b), где *name* – имя матрицы , a - СЧА, значение которого задает номер строки матрицы, b - СЧА, задающий номер столбца. Например, обозначение MX$ITOGI(1,2) соответствует ячейке в первой строке и втором столбце матрицы ITOGI.

Запись значений в ячейки матриц осуществляется с помощью блоков MSAVEVALUE. Поле A этого блока задает имя матрицы, поле B - номер строки, поле C - номер столбца, поле D - записываемое значение.

Например, при входе транзакта в блок

MSAVEVALUE ITOGI,2,4,V$ABC

в матрицу ITOGI коротких ячеек (в ячейку, расположенную на пересечении второй строки и четвертого столбца) записывается значение переменной V$ABC.

В поле A блока MSAVEVALUE после имени матрицы может быть приписан знак + или - . В этом случае значение операнда D добавляется или соответственно вычитается в ячейке матрицы, аналогично тому, как это делается в блоке SAVEVALUE.

9. Задание начальных значений ячеек и матриц

Начальные значения ячеек можно сделать отличными от нуля. Для этого применяется строка описания ячеек INITIAL. В поле A этой строки указывается СЧА, обозначающий ячейку или ячейку матрицы, в поле B – задаваемое начальное значение.

Например, строка

```
INITIAL X1,-10
```

установит в качестве начального значения ячейки номер 1 число –10, а с помощью строки

```
INITIAL MX$ITOGI(2,3),555
```

устанавливается начальное значение 555 в ячейку матрицы MX$ITOGI, расположенную во второй строке, третьем столбце.

7.10. Объектно-ориентированные средства GPSS\PC

Если общеалгоритмические средства языка GPSS\PC предоставляют нам набор возможностей, которые в том или ином виде имеются в подавляющем большинстве языков программирования, то объектно-ориентированные средства - это специфическая часть языка GPSS\PC, характерная лишь для него и его различных версий, как бы лицо этого языка.

Основой объектно-ориентированных средств является набор типовых моделей определенного класса реальных объектов, т.е. абстрактные объекты типа транзактов, устройств и т.д. В программах на языке GPSS\PC мы можем обращаться с такими абстрактными объектами и операциями над ними так же, как в языках алгоритмических обращаемся с числовыми переменными и арифметическими операциями. В то же время абстрактные объекты языка GPSS\PC, несомненно, соответствуют более "крупным" фрагментам реального мира, чем числовые переменные. Благодаря этому мы можем писать на GPSS\PC компактные и легко обозримые модели сложных систем.

На стадии выполнения модели компьютером каждый абстрактный объект представляется определенной группой ячеек оперативной памяти. Путем изменения содержимого таких ячеек имитируется изменение состояний объектов. Поскольку все абстрактные объекты взаимодействуют во времени, то представление времени на этапе машинной обработки модели играет особую роль. Понимание пользователем механизма представления модельного времени чрезвычайно важно для успешного применения языка.

Модельное время представляется в памяти ЭВМ содержимым фиксированной ячейки C1. Численное значение времени C1 логически никак не связано с физическим временем, в котором работает применяемая ЭВМ.

В начальном состоянии модели время C1 - равно нулю. Симулятор анализирует модель и вычисляет, когда должно произойти первое событие, которое изменит состояние модели. После этого симулятор наращивает содержимое ячейки C1 - сразу до значения, соответствующего моменту наступления этого события, и имитирует событие путем соответствующего изменения состояния модели. Затем анализируется полученное новое состояние модели, вновь определяется, когда произойдет очередное событие, и время C1 устанавливается равным моменту его наступления. Эта процедура наращивания времени C1 по принципу "от события к событию" повторяется до тех пор, пока не будет достигнуто заданное конечное состояние модели. Метод построения этой процедуры более подробно описан в гл.11.

Подобный принцип изменения модельного времени применяется почти во всех языках имитационного моделирования дискретных систем. Модельное время измеряется в безразмерных единицах. Определение размерности единицы времени относится к вопросу интерпретации модели. В тексте модели

размерность единицы времени нигде не указывается. Пользователь должен все интервалы времени в тексте модели задавать в одних и тех же единицах.

Кроме того, модельное время в GPSS\PC всегда выражается целым числом единиц. Хотя известно, что в дискретных системах время не обязательно должно быть дискретным, в GPSS\PC, тем не менее, введено дискретное время. Это сделано для того, чтобы результаты выполнения модели на ЭВМ не зависели от типа ЭВМ. Ведь в случае непрерывного времени его значения в ЭВМ неизбежно бы округлялись, а результат операции округления зависит от типа ЭВМ. В некоторых случаях это приводило бы к нарушениям последовательности событий в системе.

Так как время в модели может принимать только значения 0, 1, 2, ..., то любой интервал времени, меньший единичного, представляется нулем. Следовательно, если моделируемые события разделены менее чем единичным интервалом времени, то в модели они наступят одновременно, как бы "слипнутся" во времени. Отсюда вытекает важный вывод, что единицу времени необходимо выбирать достаточно мелкой, чтобы в модели не происходило "слипания" слишком большого числа событий.

7.11. Создание и уничтожение транзактов

Создание транзактов осуществляется блоком GENERATE (генерировать). Этот блок имеет 5 операндов, которые определяют моменты создания транзактов, количество транзактов и их приоритет.

Моменты модельного времени, в которые из блока GENERATE будут выходить транзакты, задаются операндами A,B и C.

Операнд A задает длительность интервалов времени между появлениями транзактов. Например, блок

```
GENERATE   100
```

будет генерировать транзакты через каждые 100 единиц модельного времени: первый транзакт выйдет в момент 100, второй – в момент 200, третий – в момент 300 и т.д. Число генерируемых транзактов будет неограниченным.

В поле B записывается *модификатор*. Если записанный в поле B СЧА является функцией FN_j, то его называют "*модификатор-функция*", в противном случае - "*модификатор-интервал*". Модификатор используется для модификации длительности интервалов времени, заданной полем A.

Рассмотрим вначале применение модификатора-интервала. Если он задан, то для каждого временного интервала поступления транзактов

длительность определяется как значение случайной величины, равномерно распределенной от “A-B” до “A+B”. Например, блок

```
        GENERATE    10,5
```

будет генерировать транзакты через интервалы времени, длительность каждого из которых выбирается случайно в пределах от 5 до 15. Необходимо помнить, что время в GPSS\PC всегда целое. Поэтому в пределах от 5 до 15 могут выбираться только целые значения времени. Каждое из этих 11-ти целых значений будет выбираться с одинаковой вероятностью. Таким образом, если в поле B задан модификатор-интервал, то блок GENERATE генерирует случайный поток транзактов, в котором время между транзактами равномерно распределено в диапазоне “A±B” и имеет среднее значение “A”.

Если же в поле B задан модификатор-функция, то для каждого временного интервала поступления транзактов длительность определяется как произведение “A*B”.

Пусть, например, в модели имеются строки

```
    DIS    FUNCTION    C1,D3
3000,3/5000,2/6000,1
           GENERATE    10,FN$DIS
```

В этом случае блок GENERATE будет генерировать транзакты вначале через каждые 30 единиц времени, пока модельное время не достигнет 3000 единиц, затем - через каждые 20 единиц времени, пока время не достигнет 5000 единиц, а после этого - через каждые 10 единиц времени.

Если использовать блок

```
           GENERATE    1000,FN$EXP,
```

где функция с меткой EXP описана так, как в параграфе 7 (пример 5), то интервал времени между генерируемыми транзактами будет псевдослучайной величиной, распределенной экспоненциально и имеющей математическое ожидание 1000. Следовательно, данный блок GENERATE будет моделировать простейший (пуассоновский) поток транзактов.

В поле C записывается момент создания первого транзакта, в поле D - число генерируемых транзактов.

Ко всему сказанному об использовании полей A и B в блоке GENERATE необходимо добавить, что:

- операнды A - D могут быть не заданы; в этом случае принимаются их значения по умолчанию: A = 0, B = 0, C=1, D = ∞.

- если при вычислении значения операнда C получен нуль, то симулятор игнорирует это значение и принимается C = 1;
- если операнд A не задан или равен нулю, то обязательно должен быть задан операнд D.

Приоритет создаваемых транзактов задается операндом E. По умолчанию приоритет равен нулю.

*Пример*6. Задания поступают в ЭВМ через каждые 5±5 мин.
Принимая минуту за единицу модельного времени, поступление заданий можно смоделировать блоком

```
GENERATE     5,5    ;ПОСТУПЛЕНИЕ ЗАДАНИЙ.
```

Однако при таком описании потока заданий возникает заметная численная ошибка дискретизации: Время между поступлениями реальных заданий распределено непрерывно в интервале от 0 до 10 мин, а модельное время между транзактами может принимать только значения 1,2,3,...,10. Для уменьшения роли такой численной ошибки можно в качестве единицы времени выбрать секунду. Тогда блок, моделирующий поступление заданий, примет вид:

```
GENERATE     300,300 ;ПОСТУПЛЕНИЕ ЗАДАНИЙ.
```

*Пример*7. Задания поступают в ЭВМ через каждые 5 мин; первое задание поступает в 0 ч, последнее - в 6 ч. Принимая за единицу времени секунду, можно поступление заданий представить в модели следующим блоком:

```
GENERATE     3000,,0,73.
```

Уничтожение транзактов осуществляется блоком TERMINATE (уничтожить). Никаких других действий этот блок не выполняет, если единственный возможный операнд A в блоке не задан. Если же задан операнд A, то его значение вычитается из содержимого *счетчика транзактов*.

Начальное значение счетчика транзактов задается в команде START. Когда в результате входа очередного транзакта в блок TERMINATE значение счетчика становится нулевым или отрицательным, симулятор прекращает выполнение модели и передает управление программе вывода, которая распечатывает накопленные симулятором данные о модели в файл REPORT.GPS.

7.12. Задержка транзактов в блоках ADVANSE

Блок ADVANSE (задержать) предназначен для задержки транзактов на определенные интервалы модельного времени.

Обязательный операнд A задает время задержки транзакта в блоке ADVANSE. Необязательный операнд B является модификатором-функцией или модификатором-интервалом. Значение операнда B используется здесь для модификации значения операнда A так же, как в блоке GENERATE.

Любой транзакт входит в блок ADVANSE беспрепятственно. В нем транзакт задерживается на период модельного времени, величина которого определяется операндами A и B. После этого транзакт направляется из блока ADVANSE к следующему блоку.

В поле A задержка транзакта может быть задана нулевой. В этом случае транзакт будет задержан в блоке ADVANSE на нулевое время, т.е. момент выхода транзакта из блока будет равен моменту его входа в блок.

Например, в блоке

```
ADVANSE     0
```

транзакты не будут задерживаться.

Блок

```
ADVANSE     1000
```

будет задерживать транзакты ровно на 1000 ед. модельного времени.

В блоке

```
ADVANSE     1000,P1
```

любой транзакт будет задерживаться на случайное время, выбранное из диапазона “1000 ± значение первого параметра транзакта”.

Пусть функция FN$EXP описана в модели так же, как в примере 5. Тогда в блоке

```
ADVANSE     1000,FN$EXP
```

задержка транзакта представляет собой случайную величину, распределенную по экспоненциальному закону и имеющую среднее значение 1000 ед. времени.

Очевидно, суммарная задержка транзакта в двух блоках

```
ADVANSE     1000,FN$EXP
ADVANSE     1000,FN$EXP
```

есть сумма двух независимых экспоненциальных случайных величин (сл.в.), которые имеют математическое ожидание 1000 ед. времени. Отсюда следует, что эта суммарная задержка есть эрланговская сл.в. второго порядка, которая имеет математическое ожидание 2000 ед. времени [8].

Рассмотрим суммарную задержку транзактов в следующих 12-ти блоках:

```
ADVANSE    100,100
ADVANSE    100,100
ADVANSE    100,100
ADVANSE    100,100
ADVANSE    100,100
ADVANSE    100,100
ADVANSE    100,100
ADVANSE    100,100
ADVANSE    100,100
ADVANSE    100,100
ADVANSE    100,100
ADVANSE    100,100.
```

Задержка в каждом из них имеет равномерное распределение вероятностей на интервале (0,200). Ее среднее значение составляет M = 100 ед. времени, дисперсия $D=200^2*(1/12)$. Поэтому сумма двенадцати таких задержек имеет среднее 12*M = 1200 и среднее квадратичное отклонение $\sqrt{12*D}$ = 200. По центральной предельной теореме теории вероятностей заключаем, что закон распределения суммарной задержки приблизительно нормальный.

Заметим, что с помощью СЧА RN_j такую задержку можно задать проще, если использовать арифметические переменные:

```
                    ОПИСАНИЕ ПЕРЕМЕННЫХ:

D        VARIABLE    RN1+RN1+RN1+RN1
NORM     VARIABLE    (V$D+V$D+V$D)/5

              НОРМАЛЬНО РАСПРЕДЕЛЕННАЯ ЗАДЕРЖКА:

         ADVANSE     V$NORM.
```

7.13. Операции занятия и освобождения устройств

Операции занятия и освобождения устройств выполняются в блоках SEIZE (занять) и RELEASE (освободить).

При вводе транзакта в блок SEIZE выполняется операция занятия устройства, номер или имя которого задается операндом A блока SEIZE. Номер устройства может быть задан в виде любого СЧА.

Занятие устройства транзактом выполняется следующим образом. Когда транзакт направляется из какого-нибудь блока в блок SEIZE, симулятор проверяет, свободно ли требуемое устройство. Если устройство не свободно, т.е. занято или захвачено, то транзакт не может войти в блок SEIZE. Он остается в предыдущем блоке до тех пор, пока устройство не освободится. Если же устройство свободно, то симулятор без задержки передвигает транзакт в блок SEIZE и транзакт занимает устройство. В этот же момент времени транзакт направляется к следующему за SEIZE блоку.

В последующем движении по модели транзакт может освободить занятое устройство. Освобождение устройства происходит в блоке RELEASE, в поле A которого указывается номер освобождаемого устройства.

При составлении моделей пользователь должен соблюдать правило: освободить устройство может только тот транзакт, который его занимает. Если транзакт попытается освободить устройство, занятое или захваченное другим транзактом, симулятор прервет выполнение модели и выдаст сообщение об ошибке.

Если несвободное устройство пытаются занять несколько транзактов с помощью одного или разных блоков SEIZE, то все они до момента освобождения устройства задерживаются в тех блоках, в которых находились в момент попытки перейти в блок (блоки) SEIZE. В момент освобождения устройства должен быть решен вопрос о том, какой из задержанных транзактов имеет право первым занять устройство. В GPSS\PC этот вопрос решается следующим образом. Когда транзакты задерживаются на входе блока SEIZE, они регистрируются симулятором в списке. В этом списке транзакты упорядочиваются по приоритетам: любой транзакт с более высоким приоритетом находится впереди любого другого транзакта, имеющего более низкий приоритет. Если же приоритеты у каких-либо транзактов одинаковы, то такие транзакты упорядочиваются между собой по времени прихода: впереди оказывается транзакт, который раньше обратился к устройству. Такое упорядочение задерживаемых транзактов будем называть *приоритетным*. Приоритетное упорядочение является в GPSS\PC основным способом упорядочения задерживаемых транзактов и применяется не только при обращениях к несвободному устройству, но и во многих других случаях. В момент освобождения устройства его занимает тот из задержанных транзактов, который находится в списке первым.

Транзакт может занимать любое число устройств. Освобождать занятые устройства транзакт может в любом порядке.

Пример8. Пусть транзакты имеют одинаковые приоритеты и изображают кинозрителей, а устройство представляет кассира в кассе кинотеатра.

Каждый кинозритель приобретает у кассира билеты в течение 20 ± 5с, в зависимости от числа билетов и необходимости получения сдачи. Кинозрители

приходят через каждые 20 ± 10с, знакомятся в течение 15 ± 15с с обстановкой и занимают очередь. Общее число кинозрителей 200.

Тогда имитационную модель процесса обслуживания кинозрителей кассиром мы можем описать так.

```
*    МОДЕЛЬ РАБОТЫ КАССИРА
          GENERATE    20,10,,200      ;ПРИХОД ЗРИТЕЛЕЙ
          ADVANSE     15,15           ;ЗНАКОМСТВО С
ОБСТАНОВКОЙ
          SEIZE       1               ;ОБРАЩЕНИЕ К КАССИРУ
          ADVANSE     20,5            ;ПОКУПКА БИЛЕТА
          RELEASE     1               ;ОСВОБОЖДЕНИЕ
КАССИРА
          TERMINATE   1               ;УХОД ЗРИТЕЛЯ
          START       200             ;СЧЕТЧИК=200.
```

Рассмотрим логику выполнения этой модели. Первая строка модели является комментарием и симулятором не воспринимается. Блок GENERATE изображает приход зрителей: он генерирует транзакты через каждые 20 ± 10 единиц времени до тех пор, пока не будет выдано 200 транзактов. Каждый транзакт-зритель, поступивший из блока GENERATE, попадает в блок ADVANSE 15,15 и задерживается на время “ознакомления с обстановкой”. Заметим, что, поскольку эта задержка случайная и может для разных транзактов отличаться на 30 единиц времени, то возможны случаи, когда позже попавшие в этот блок транзакты будут выходить из него раньше.

Если транзакт направляется из блока ADVANSE 15,15 в блок SEIZE 1, в то время как устройство 1 свободно, то он входит в этот блок, занимает устройство, переходит в блок ADVANSE 20,5 и задерживается в нем. Таким образом, в течение 20 ± 5 ед. модельного времени устройство 1 будет занято. Транзакты, пытающиеся в это время перейти из блока ADVANSE 15,15 в блок SEIZE, будут задержаны и упорядочены по приоритетам. Поскольку приоритеты у всех транзактов одинаковы и равны нулю (см. блок GENERATE), то транзакты будут упорядочены в порядке времени их обращения к устройству. Следовательно, задерживаемые транзакты будут в точности изображать живую очередь к кассиру.

Из блока ADVANSE 20,5 транзакт переходит в блок RELEASE 1 и освобождает устройство 1. В этот же момент времени транзакт переходит в блок TERMINATE и уничтожается. В результате из содержимого счетчика строки START вычитается 1. Моделирование будет продолжаться до тех пор, пока счетчик не станет равным нулю. В строке START задано начальное

значение счетчика, которое равно 200. Ввод строки START является командой симулятору начать выполнение модели.

7.14. Операции захвата и освобождения устройств.

Операция захвата устройства выполняется транзактом с помощью блока PREEMPT (прервать). Номер захватываемого устройства задается в этом блоке в поле A. Если транзакт захватил устройство посредством блока PREEMPT, то освобождать его он должен в блоке RETURN (восстановить). Номер освобождаемого устройства задается в блоке RETURN в поле A.

Блок PREEMPT позволяет транзакту использовать устройство на более льготных условиях, чем блок SEIZE.

Рассмотрим наиболее простой режим работы блока PREEMPT. В этом режиме блок работает в случае, если задан только операнд A. Здесь возможны следующие три ситуации.

1. *При входе транзакта в блок PREEMPT устройство свободно*. В этом случае транзакт беспрепятственно входит в блок и захватывает устройство. Устройство из состояния "свободно" переходит в состояние "захвачено". Когда транзакт войдет в соответствующий блок RETURN, устройство из состояния "захвачено" вновь перейдет в состояние "свободно".

2. *При переходе транзакта в блок PREEMPT устройство занято другим транзактом, прошедшим блок SEIZE*. В этом случае транзакт беспрепятственно входит в блок PREEMPT, а устройство прерывает обслуживание занимающего его транзакта и переключается на обслуживание транзакта, вошедшего в блок PREEMPT. При этом из состояния "занято" устройство переходит в состояние "захвачено". Когда транзакт, захвативший устройство, войдет в блок RETURN, устройство освободится, возобновит прерванное обслуживание другого транзакта и перейдет в состояние "занято".

Если прерываемый транзакт в момент прерывания находится в блоке ADVANCE, то, начиная с момента прерывания, отсчет времени пребывания транзакта в этом блоке прекращается до тех пор, пока не будет восстановлено обслуживание транзакта. Таким образом, в момент восстановления прерванного обслуживания транзакта время, оставшееся этому транзакту до выхода из блока ADVANCE, такое же, каким оно считалось непосредственно в момент прерывания. Такое прерывание обслуживания называется прерыванием с последующим дообслуживанием.

3. *При входе транзакта в блок PREEMPT устройство захвачено другим транзактом, прошедшим блок PREEMPT* . В этом случае транзакт задерживается в предыдущем блоке до момента освобождения устройства.

Все транзакты, задержанные при обращении к устройству, упорядочиваются по приоритету.

Более сложные режимы прерывания задаются с помощью большего числа операндов. Кроме операнда A в блоке PREEMPT могут быть заданы какие-либо из операндов B,C,D,E.

Операнд B записывается в виде обозначения PR, задающего приоритетный режим работы блока PREEMPT. В этом режиме транзакт захватывает устройство, если оно свободно или если оно обслуживает менее приоритетный транзакт. Прерывание обслуживания менее приоритетного транзакта происходит с его последующим дообслуживанием. Прерванные транзакты упорядочиваются по приоритету совместно с вновь приходящими транзактами, т.е. образуют с ними одну очередь к устройству. Вновь приходящий транзакт задерживается на входе блока PREEMPT только тогда, когда устройством обслуживается не менее приоритетный транзакт.

Для определения последующего движения прерванных транзактов блок PREEMPT предоставляет пользователю следующие возможности .

1. В поле C может быть указано имя какого-либо блока, на который будет передан прерванный транзакт. При этом прерванный транзакт продолжает претендовать на данное устройство.

2. В поле D блока PREEMPT может быть задан номер параметра транзакта. Тогда, если прерванный транзакт находится в блоке ADVANCE, то вычисляется остаток времени обслуживания (время дообслуживания), и полученное значение помещается в параметр, заданный в поле D. Прерванный транзакт при этом будет послан в блок, указанный в поле C. Прерванный транзакт продолжает претендовать на данное устройство.

3. Если в поле E блока PREEMPT записано обозначение RE, то будут производиться те же операции, что и в случае 1), за исключением того, что прерванный транзакт больше не претендует на данное устройство .

Пример 9. Оперативная память однопроцессорной ЭВМ разбита на три раздела с номерами 0, 1, 2. В каждом разделе находится по одной программе. Приоритет программы тем выше, чем меньше номер ее раздела. Программы вызываются на выполнение по требованиям пользователей, поступающим в случайные моменты времени .

Программа нулевого раздела вызывается с интервалами длительностью 10 ± 8 с. Время ее выполнения процессором в зависимости от обрабатываемых данных колеблется в пределах 3 ± 2с.

Программа первого раздела вызывается через каждые 2мин. и выполняется процессором в течение 30 ± 15с.

Через полчаса после начала поступления требований в нулевой и в первый разделы начинают поступать требования во второй раздел с интервалами по 5 ± 4 мин. Такие требования обрабатываются 1 мин.

В случае конфликтных ситуаций, когда при поступлении какого-либо требования процессор выполняет другое требование, используется дисциплина абсолютных приоритетов .

Требуется написать модель на языке GPSS\PC, воспроизводящую первый час описанного процесса. При моделировании необходимо подсчитывать число поступивших и число обслуженных требований каждого вида .

Будем представлять требования на выполнение программ транзактами.

Присвоим требованиям приоритеты 0, 1, 2. Согласно условию задачи нулевой - наименьший - приоритет должны иметь требования к третьему разделу, а приоритет 2 - требования к нулевому разделу. Процессор представим в модели устройством с номером 1. За единицу времени примем 1с. Для подсчета требований воспользуемся ячейками SAVEVALUE. Тогда модель может быть написана в следующем виде.

```
* МОДЕЛЬ НУЛЕВОГО РАЗДЕЛА:
    GENERATE  10,8,,,2     ;ПЕРВЫЙ ПОТОК ТРЕБОВАНИЙ
    SAVEVALUE 1+,1         ;ПОДСЧЕТ ИХ ЧИСЛА
    PREEMPT   1,PR         ;ЗАХВАТ ПРОЦЕССОРА
    ADVANCE   3,2          ;ВЫПОЛНЕНИЕ ТРЕБОВАНИЯ
    RETURN    1            ;ОСВОБОЖДЕНИЕ ПРОЦЕССОРА
    SAVEVALUE 2+,1         ;ПОДСЧЕТ ОБСЛУЖЕННЫХ ТРЕБ.
    TERMINATE
* МОДЕЛЬ ПЕРВОГО РАЗДЕЛА:
    GENERATE  120,,,,1     ;ВТОРОЙ ПОТОК ТРЕБОВАНИЙ
    SAVEVALUE 3+,1         ;ПОДСЧЕТ ИХ ЧИСЛА
    PREEMPT   1,PR         ;ЗАХВАТ ПРОЦЕССОРА
    ADVANCE   30,15        ;ВЫПОЛНЕНИЕ ТРЕБОВАНИЯ
    RETURN    1            ;ОСВОБОЖДЕНИЕ ПРОЦЕССОРА
    SAVEVALUE 4+,1         ;ПОДСЧЕТ ОБСЛУЖ-Х ТРЕБ-Й
    TERMINATE
* МОДЕЛЬ ВТОРОГО РАЗДЕЛА:
    GENERATE  300,240,1800;ТРЕТИЙ ПОТОК ТРЕБОВАНИЙ
    SAVEVALUE 5+,1         ;ПОДСЧЕТ ИХ ЧИСЛА
    PREEMPT   1,PR         ;ЗАХВАТ ПРОЦЕССОРА
    ADVANCE   60           ;ВЫПОЛНЕНИЕ ТРЕБОВАНИЯ
    RETURN    1            ;ОСВОБОЖДЕНИЕ ПРОЦЕССОРА
    SAVEVALUE 6+,1         ;ПОДСЧЕТ ОБСЛУЖ-Х  ТРЕБ-Й
    TERMINATE
* ОСТАНОВ. МОДЕЛИ ПРИ T=1 ЧАСУ = 3600 СЕКУНДАМ:
    GENERATE  3600
    TERMINATE 1
    START     1.
```

В результате моделирования наряду с информацией об устройстве 1 в файл REPORT.GPS автоматически будет выведено значение ячеек X1-X6 в виде таблицы, содержащей номера этих ячеек и их содержимое.

Значение ячейки X1 будет равно числу требований, поступивших в нулевой раздел оперативной памяти, а ячейки X2 - числу обслуженных таких требований. Аналогично интерпретируются значения ячеек X3, X4 для требований первого раздела и ячеек X5, X6 - для второго.

7.15. Операции блокирования и разблокирования устройств

Транзакты имеют возможность блокировать устройства, т.е. делать их недоступными, не занимая этих устройств. При этом могут задаваться разные режимы управления теми транзактами, которые будут пытаться занять или захватить блокированное устройство.

Блокирование устройств выполняется с помощью блока FUNAVAIL (устройство недоступно):

FUNAVAIL A [, B, C, D, E, F, G, H]

у которого поле A обязательно, а поля B – H могут отсутствовать.

В полях задаются:

A - устройство или диапазон устройств;

B – это обозначение RE или CO. Если задано RE, то транзакт, который в момент блокирования занимает (возможно) устройство,с обслуживания снимается и в дальнейшем не претендует на его занятие. В этом случае в поле C задается метка, на которую пересылается снятый с обслуживания транзакт. Если в поле B записано CO, то обслуживаемый транзакт продолжает обслуживаться до конца;

C – метка или номер блока;

D – номер параметра управляющего транзакта,

E – определяет поведение ранее прерванных транзактов на период недоступного состояния устройства. Здесь может быть записано RE или CO: при RE они исключаются из борьбы за устройство и передаются по метке поля F, а при CO продолжают претендовать на дообслуживание после восстановления доступности устройства и его освобождении от более приоритетеных транзактов;

F – номер или метка блока;

G – определяет поведение транзактов, которые будут обращаться к устройству в период его недоступности. Запись CO означает, что они будут

ожидать устройство в обычном порядке, запись RE – что они исключаются из борьбы и передаются по метке поля H;

H – метка или номер блока.

Разблокирование устройств выполняется блоком FAVAIL (устройство доступно):

FAVAIL A,

где A – номер или диапазон номеров устройств (например, FUNAVAIL 4-8).

7.16. Описание памятей и работа с ними

В отличие от устройства память в GPSS\PC имеет параметр, от которого зависят результаты операций с нею. Этот параметр - емкость памяти - должен быть известен симулятору. Поэтому для каждой применяемой памяти пользователь должен указать ее емкость.

Для указания емкостей памятей применяется строка STORAGE . Эта строка является строкой описания объекта, а не блоком , поэтому транзакты в нее не входят .

Поле метки строки STORAGE содержит номер или метку памяти, поле операции - слово STORAGE, а поле A указывает емкость памяти. Емкость памяти должна выражаться целым числом. Например, строка описания памяти MEM емкостью 40 единиц может быть записана (с комментариями) следующим образом:

```
MEM      STORAGE  40   ;ПАМЯТЬ MEM ИМЕЕТ ЕМКОСТЬ 40 ЕД.
```

В начальный момент времени все памяти пустые. Если в ходе моделирования транзакт обращается к неописанной памяти, ее объем принимается равным 2^{31} - 1 единиц.

Транзакты обращаются к памяти посредством блоков ENTER (войти) и LEAVE (выйти).

Блок ENTER применяется для занятия памяти. В поле A указывается номер памяти, в поле B - число занимаемых единиц. Когда транзакт входит в блок ENTER, определяется число свободных единиц памяти, равное емкости памяти минус число занятых единиц. Если значение операнда B не превышает числа свободных единиц памяти, то число занятых единиц увеличивается на значение операнда B. В этом случае транзакт входит в блок ENTER без задержки. Если же значение операнда B превышает число свободных единиц памяти, то

транзакт задерживается перед входом в блок ENTER. Задержанные при обращении к памяти транзакты упорядочиваются по приоритету .

Число занимаемых единиц памяти должно быть целым .

Если поле B в блоке ENTER пустое, то число занимаемых единиц памяти принимается равным 1 .

Пусть транзакт “x” задержан перед входом в блок ENTER. Из описания блока ENTER видно, что если для транзакта “y”, приходящего после “x”, свободной емкости памяти достаточно, то “y” войдет в блок ENTER без задержки.

Блок LEAVE служит для освобождения памяти. В поле A указывается номер памяти, в поле B - число освобождаемых единиц. В случае пустого поля B число освобождаемых единиц памяти принимается равным единице .

При входе транзакта в блок LEAVE количество занятых единиц памяти, указанной в поле A, уменьшается на значение операнда B. Перед входом в блок LEAVE транзакты не задерживаются. Транзакт не должен освобождать большее число единиц памяти, чем их всего занято. Если же транзакт пытается это сделать, то симулятор выдает на печать сообщение об ошибке и прекращает выполнение модели .

В тот момент модельного времени, когда транзакт освобождает память, симулятор просматривает от начала до конца список задержанных у памяти транзактов, если они есть. Для каждого очередного транзакта проверяется, может ли он теперь быть обслужен памятью. Если такая возможность есть, то симулятор перемещает этот транзакт в блок ENTER, и в результате число занятых единиц памяти соответствующим образом увеличивается.

Транзакт не обязан освобождать такое же число единиц памяти, какое занимал. Он может также освобождать память, которую не занимал. Транзакт имеет право занимать и освобождать любое количество памятей и операции занятия и освобождения могут при этом чередоваться в произвольном порядке.

Число освобождаемых единиц памяти должно быть целым.

Пример10. Оперативная память (ОП) однопроцессорной ЭВМ имеет емкость 16 Мбайт. Из них 3 Мбайт постоянно занимают управляющие программы операционной системы (ОС). Остальной объем ОП используется для выполнения заданий абонентов ЭВМ.

В каждом задании абонентом записан приоритет задания и требуемый для его выполнения объем ОП. При поступлении задания управляющие программы ОС проверяют, достаточен ли свободный объем ОП для загрузки. Если свободного места в ОП достаточно, то задание загружается и занимает часть памяти требуемого объема. В противном случае задание записывается на НМД в очередь, упорядоченную по приоритетам.

При завершении задания занимаемая им часть памяти освобождается. Управляющие программы ОС обеспечивают непрерывность всего свободного

объема ОП за счет возможности сдвига адресов загрузки у выполняемых заданий. После освобождения части ОП завершенным заданием и обеспечения непрерывности свободной памяти управляющие программы просматривают по порядку все задания в очереди на НМД и загружают из них те, для которых в ОП находится достаточное место. Время выполнения управляющих программ по сравнению со временем выполнения заданий можно считать равным нулю .

ОП, используемая заданиями, имеет объем 16 - 3 = 13 Мбайт. При моделировании ЭВМ эта ОП может быть описана с помощью строки *name* STORAGE 13. Если заданиям поставить в соответствие транзакты, то вход заданий в ОП можно моделировать с помощью блока ENTER, а выход из ОП - с помощью блока LEAVE .

Памяти могут также блокироваться и разблокироваться, как и устройства.

Блокирование памятей выполняется с помощью блока

SUNAVAIL A,

где A – номер (имя) или диапазон номеров памятей. Во время недоступности памяти блок ENTER задерживает транзакты, а блок LEAVE – нет.

Доступность памятей восстанавливается посредством блока

SAVAIL A,

В котором в поле A задается номер (имя) памяти или диапазон номеров памятей.

7.17. Операции с ключами

Для управления состоянием логических ключей применяется блок LOGIC.

Все ключи в модели в нулевой момент времени выключены, т.е. находятся в состоянии R. При входе транзакта в блок LOGIC состояние логического ключа, указанного в поле A, изменяется одним из трех способов. Способ изменения определяется символом, записанным в позиции 14 поля операции. Символ R означает, что ключ требуется установить в состояние R (выключить), символ S - установить в состояние S (включить), а символ I - инвертировать состояние ключа, т.е. включить его, если он выключен и выключить, если он включен.

В табл.7 приведены примеры записи блока LOGIC с комментариями.

Таблица 7

	МЕТКА		ОПЕРАЦИЯ	A,B,...→
			LOGIC S	41 Включить ключ 41
			LOGIC R	155 Выключить ключ 155
			LOGIC I	4 Инвертировать ключ 4

Начальное состояние логических ключей можно задавать с помощью строки INITIAL.

18. Блок GATE

Состояние оборудования может опрашиваться транзактом в блоке GATE и учитываться для выбора дальнейшего способа движения транзакта.

В поле операции блока GATE записывается слово GATE и через пробел - символ проверяемого условия.

Существует десять условий, которые проверяются в блоке GATE для оборудования:

NU- устройство свободно (т.е. не используется),
U - устройство не свободно (т.е. используется),
NI - устройство не захвачено,
I - устройство захвачено,
SE - память пуста (все единицы памяти свободны),
SNE - память не пуста,
SF - память заполнена (все единицы заняты),
SNF - память не заполнена,
LR - ключ выключен,
LS - ключ включен.

Кроме этих условий, смысл которых здесь ясен, в блоке GATE применяются также следующие два *условия синхронизации транзактов*:

M,
NM,

смысл которых будет пояснен в п. 3.9.

В поле A блока GATE записывается номер или имя проверяемого объекта.

Поле B определяет режим работы блока GATE. Блок GATE может работать в режиме *отказа* или в режиме *условного перехода*.

1. Если поле B пустое, блок работает в режиме отказа. В этом случае направляемый в блок GATE транзакт при невыполнении условия задерживается перед входом в блок до момента , когда условие будет выполнено. Когда условие выполнится, транзакт войдет в блок GATE и перейдет к следующему в тексте модели блоку .

2. Если поле B не пустое, то в нем должна быть записана метка какого-нибудь блока. В этом случае блок GATE работает в режиме условного перехода. Транзакт входит в такой блок беспрепятственно. При выполнении требуемого условия транзакт тут же переходит к следующему за GATE блоку, а при невыполнении - к блоку, метка которого указана в поле B.

7.19. Синхронизация транзактов

Любые элементы в системах прямо или опосредованно связаны, взаимодействуют. Зависимость между процессами, протекающими в разных частях системы, нередко выражается в форме синхронизации, т.е. в форме взаимного согласования этих процессов по времени. Для моделирования различных видов синхронизации в GPSS\PC применяются блоки SPLIT, ASSEMBLE, GATHER, MATCH, GATE .

Блок SPLIT предназначен для моделирования одновременного начала нескольких процессов. В момент входа транзакта в блок SPLIT создается несколько копий этого транзакта. Число копий задается в поле A. Исходный (порождающий) транзакт переходит в блок, определенный в поле B. Все копии переходят к блоку, следующему за SPLIT. Если поле C блока SPLIT пустое, то все копии идентичны породившему их транзакту. Например, при входе транзакта в блок

```
SPLIT       4,NEXT
```

порождаются четыре транзакта, идентичные вошедшему, и передаются в следующий по тексту блок. Исходный транзакт переходит в блок, в поле метки которого записано NEXT. Всего из этого блока SPLIT выходит пять транзактов.

Если поле C не пустое, то его значение интерпретируется как номер параметра транзакта. Пусть N - значение этого параметра в момент входа транзакта в блок SPLIT. Тогда в момент выхода из SPLIT данный параметр у исходного транзакта будет иметь значение N+1 , а у копий соответственно N+2, N+3, ... , N+K , где K - общее число вышедших из блока SPLIT транзактов. Например, если транзакт, имеющий нуль в десятом параметре, войдет в блок

```
SPLIT       2,ABCDE,10,
```

то параметр P10 у этого транзакта приобретет значение 1, а у копий соответственно 2 и 3 .

Транзакты - копии могут двигаться в модели независимо друг от друга. Копии могут проходить блоки SPLIT и порождать новые копии .

Множество, состоящее из исходного транзакта и всех его копий, называется *семейством* транзактов. Копия члена семейства является членом того же семейства. Любой транзакт - член одного и только одного семейства .

Блок ASSEMBLE используется для моделирования одновременного завершения нескольких процессов. Он собирает заданное в поле A число транзактов одного семейства и превращает их в один транзакт. Первый из транзактов какого-либо семейства, вошедший в блок, задерживается до тех пор, пока в этом блоке не накопится заданное число транзактов того же семейства. После этого первый транзакт выходит из блока ASSEMBLE, а остальные транзакты этого семейства уничтожаются.

В одном блоке ASSEMBLE могут одновременно проходить сборку транзакты, принадлежащие к разным семействам. Например, если в блок

ASSEMBLE 4

поступают транзакты разных семейств, то транзакты каждого семейства собираются по четыре и каждая четверка превращается в один транзакт.

Аналогично применяется и блок GATHER. Отличие состоит лишь в том, что транзакты в этом блоке не уничтожаются. Когда в нем накапливается заданное в поле A число транзактов одного семейства, все они переходят к следующему блоку.

Блок MATCH синхронизирует продвижение двух транзактов, принадлежащих к одному семейству и двигающихся в модели по различным путям. В поле A блока MATCH задается имя другого блока, который называется *сопряженным* блоком MATCH. Два транзакта, принадлежащие к одному семейству и поступающие в пустые сопряженные блоки MATCH в моменты времени T1 и T2 ≥ T1, выходят из этих блоков одновременно в момент T2.

Пример11. Нужно промоделировать сборку изделий рабочими A,B и C. Изделия в разобранном виде поступают каждые 300 ± 100 мин. Каждое из них разделяется между рабочими A и B, которые параллельно готовят свою часть изделия к сборке. Подготовка изделия состоит из двух фаз, причем после окончания первой фазы производится сверка с одновременным участием обоих рабочих A и B. После сверки рабочие независимо друг от друга выполняют вторую фазу. На выполнение первой фазы рабочий A тратит 100 ± 20 мин, а рабочий B 80 ± 20 мин.

Вторую фазу рабочий A выполняет 50 ± 5 мин, а рабочий B 70 ± 10 мин.

После окончания работы рабочими A и B рабочий C выполняет сборку изделия. Сборка занимает 50 ± 5 мин. и не может быть начата до тех пор, пока рабочие A и B не закончили выполнение своих операций .

Модель описанного процесса может быть написана с применением блоков синхронизации транзактов:

```
*
*   МОДЕЛЬ СБОРКИ ИЗДЕЛИЙ
          GENERATE    300,100   ;ПОСТУПЛЕНИЕ ИЗДЕЛИЙ
          SPLIT       1,MANB    ;РАЗДЕЛЕНИЕ ИЗДЕЛИЙ
          SEIZE       1         ;ЗАНЯТЬ РАБОЧЕГО A
          ADVANCE     100,20    ;ПЕРВАЯ ФАЗА
  ARB1    MATCH       ARB2      ;ЖДАТЬ РАБОЧЕГО B
          ADVANCE     50,5      ;ВТОРАЯ ФАЗА
          RELEASE     1         ;ОСВОБОДИТЬ РАБОЧЕГО A
          TRANSFER    ,MANC     ;ПЕРЕДАЧА ИЗДЕЛИЯ В СБОРКУ
  MANB    SEIZE       2         ;ЗАНЯТЬ РАБОЧЕГО B
          ADVANCE     90,20     ;ВРЕМЯ ПЕРВОЙ ФАЗЫ
  ARB2    MATCH       ARB1      ;ЖДАТЬ РАБОЧЕГО A
          ADVANCE     70,10     ;ВТОРАЯ ФАЗА
          RELEASE     2         ;ОСВОБОДИТЬ РАБОЧЕГО B
  MANC    ASSEMBLE    2         ;ЖДАТЬ ОБЕ ЧАСТИ ИЗДЕЛИЯ
          SEIZE       3         ;ЗАНЯТЬ РАБОЧЕГО C
          ADVANCE     50,5      ;СБОРКА ИЗДЕЛИЯ
          RELEASE     3         ;ОСВОБОДИТЬ РАБОЧЕГО C
          TERMINATE   1         ;СБОРКА ЗАВЕРШЕНА
          START       1000      ;МОДЕЛИРОВАТЬ 1000 СБОРОК
```

Синхронизация транзактов может производиться и с помощью уже известного нам блока GATE, если в нем указано условие синхронизации M или NM, а в поле A - имя блока ASSEMBLE, GATHER или MATCH. Условие M означает “в указанном блоке находится транзакт одного семейства с тем, который обращается к блоку GATE”. Условие NM противоположно условию M. Например, если в модели имеются строки:

```
  CPU     GATE M      PROG1
    .
    .
  PROG1   MATCH       PROG2
    .
    .
  DISK    GATE NM     PROG2
    .
```

```
    .
PROG2    MATCH        PROG1,
```

то в блок CPU транзакт сможет войти лишь при условии, что в блоке PROG1 в состоянии синхронизации находится другой транзакт из этого же семейства. В блок DISK транзакт какого - либо семейства сможет войти при условии, что в блоке PROG2 нет транзакта этого семейства .

7.20. Изменение параметров и стандартных числовых атрибутов транзактов

Основным средством задания параметров транзактов является блок ASSIGN (назначить). В поле A этого блока указывается номер параметра, в поле B определяется записываемое в параметр значение.

Приписывая к номеру параметра в поле A символ + или -, можно обеспечить не запись значения поля B в параметр, а добавление или вычитание этого значения из значения параметра.

Если в поле C указано какое-либо имя, оно интерпретируется как имя функции. В этом случае вычисляется указанная функция и значение поля B модифицируется путем умножения на значение функции. Примеры:

```
ASSIGN      1,475          ;ЗАНЕСТИ 475 В P1
ASSIGN      12+,45         ;ДОБАВИТЬ 45 В P1
ASSIGN      11,C1          ;ЗАПИСАТЬ ВРЕМЯ В P11
ASSIGN      8,42,FFF       ;ЗАПИСАТЬ 42*FN$FFF В P8.
```

В блоке ASSIGN P1,P2 содержимое второго параметра будет записано в параметр, номер которого нужно прочесть в первом параметре. В блоке ASSIGN P*1,P2 содержимое второго параметра будет записано в параметр, номер которого записан в другом параметре, номер которого нужно прочесть в параметре P1.

При моделировании на GPSS\PC часто возникают случаи, когда нужно менять стандартные числовые атрибуты транзактов. К атрибутам транзактов относятся приоритет и отметка времени.

Для задания приоритетов транзактов кроме блока GENERATE можно использовать блок PRIORITY. Поле A этого блока определяет значение присваиваемого приоритета. Например, при прохождении через блок

```
PRIORITY  3
```

транзакту будет присвоен приоритет 3.

Отметка времени - это запись в транзакте текущего момента времени, которая автоматически производится в момент порождения транзакта блоком GENERATE и может меняться только в блоке MARK. Если транзакт не проходил блока MARK, то время существования транзакта в модели может быть определено как разность между текущим значением модельного времени и отметкой времени транзакта. При прохождении блока MARK с пустым полем A отметка времени транзакта заменяется на новую, равную моменту прохождения этого блока.

Если в поле A указано некоторое значение, то оно рассматривается как номер параметра транзакта и текущее время записывается в этот параметр. Отметка времени транзакта при этом не меняется, т.е. сохраняет свое старое значение.

Например, блок

```
MARK
```

записывает текущее время в отметку времени проходящего через него транзакта, а блок

```
MARK    1
```

записывает текущее время только в P1.

Блок MARK можно применять для измерения времени прохождения транзактом определенного участка модели. В начале этого участка блоком MARK делается отметка времени в транзакте, а в конце участка определяется разность между текущим модельным временем и отметкой времени, сделанной в транзакте.

7.21. Управление маршрутами транзактов в модели

В тех случаях, когда естественный порядок движения транзактов сверху вниз по блокам модели необходимо определенным образом нарушить, можно воспользоваться блоками TRANSFER (переслать), LOOP (повторить) или TEST (проверить). В некоторых частных случаях для этого могут подойти, конечно, и ранее изученные блоки SPLIT и GATE.

Блок TRANSFER имеет в общем случае четыре операнда. Операнд A задает режим выполнения блока, операнды B и C являются метками блоков.

Наиболее простым является режим *безусловного перехода* блока TRANSFER. В этом режиме поле A в блоке пустое. Все транзакты переходят к блоку, указанному в поле B. Пример:

```
TRANSFER    ,NEXT.
```

Если в поле A записана десятичная дробь, начинающаяся точкой, то производится случайный выбор между блоками, определенными в полях B и C. Десятичная дробь в поле A задает вероятность перехода к блоку поля C. Например, блоком

```
TRANSFER    .370,CCC,BBB
```

37% всех транзактов будет передано в блок с меткой BBB, остальные 63% - в блок CCC. Описанный режим называется *статистическим*.

Если в поле A блока TRANSFER записано мнемоническое обозначение BOTH, то для каждого транзакта, поступающего в этот блок, проверяются два пути. Сначала проверяется возможность перехода к блоку поля B, и, если транзакт не может войти в этот блок, он пытается перейти к блоку поля C. Если транзакт не может войти и в этот блок, он задерживается в блоке TRANSFER и опрос блоков, заданных в полях B и C, повторяется при каждом изменении модельного времени, пока не появляется возможность выхода.

Примечание 1. Этот режим выбора может существенно увеличить время обработки модели, т.к., если оба выхода заблокированы, то проверка их состояния производится симулятором очень часто.

Примечание 2. Если в блоке TRANSFER поле B, интерпретируемое как адрес блока, пустое, то считается, что оно указывает на блок, следующий за данным блоком TRANSFER.

С помощью режима BOTH блока TRANSFER можно промоделировать, например, выбор заявками одного из двух каналов в двухканальной СМО:

```
          TRANSFER    BOTH,,SAM   ;ВЫБОР СВОБ-ГО КАНАЛА
          SEIZE       175         ;ЗАНЯТИЕ  1-ГО КАНАЛА
            .
            .
            .
SAM       SEIZE       17      ;ЗАНЯТИЕ  2-ГО КАНАЛА.
```

Обобщением режима BOTH является режим ALL, задаваемый мнемоническим обозначением ALL в поле A блока TRANSFER. Первый из блоков, возможность перехода к которым проверяется, определяется полем B, последний - полем C. В поле D задается индексная константа N. Транзакт проверяет по порядку возможность входа в блоки с номерами B, B+N, B+2N,..., B+MN=C и входит в первый из них, который это разрешает. Если же такой возможности не представляется, проверка повторяется при каждом изменении модельного времени.

С помощью режима ALL можно промоделировать, например, выбор заявками одного из трех каналов в трехканальной СМО.

Кроме описанных - безусловного, статистического режима, а также режимов BOTH и ALL возможны еще пять режимов выполнения блока TRANSFER: SIM, PICK , SBR, FN и P. Эти режимы более специальные и могут быть изучены при необходимости с помощью документации и «помощи», имеющихся в составе пакета программ GPSS\PC.

Для организации циклического прохождения транзактов по определенному сегменту модели служит блок LOOP. В поле A указывается номер параметра транзакта, а в поле B задается метка какого-нибудь блока. Каждый раз, когда транзакт входит в блок LOOP, значение указанного параметра транзакта уменьшается на 1. Если после этого параметр не равен нулю, транзакт пересылается по заданной метке, в противном случае - к блоку, записанному под данным блоком LOOP.

В следующем примере транзакты проходят через блок ADVANCE по 4 раза.

```
          ASSIGN     1,4        ;ЗАНЕСТИ  4 в P1
WAIT    ADVANCE    15,FN$EXP  ;ЖДАТЬ
        LOOP       1,WAIT  ;ИДТИ  К WAIT при P1≠ 0.
```

Заметим, что, поскольку счетчик числа циклов задается в параметре транзакта, то он может у разных транзактов иметь разные начальные значения и, таким образом, число прохождений одного и того же цикла может быть у всех транзактов разным.

Необходимо также учитывать, что в случае, когда в LOOP входит транзакт с нулевым или отрицательным значением параметра, указанного в поле A, симулятор выдает сообщение об ошибке и прекращает выполнение модели.

Наиболее универсальными возможностями для управления маршрутами сообщений располагает блок TEST. В поле операции этого блока через пробел после слова TEST записывается мнемоническое обозначение одного из шести отношений: L, LE, E, NE, G или GE. В полях A и B записываются обозначения сравниваемых СЧА. Движение транзакта зависит от того, задана ли метка какого-нибудь блока в поле C.

Если поле C пустое, то блок TEST выполняется в режиме отказа. В этом случае транзакт будет задерживаться на входе блока TEST при невыполнении указанного отношения между СЧА в полях A и B, или передаваться к следующему после TEST блоку - при выполнении отношения. Например, в следующих блоках транзакт будет задерживаться при невыполнении условий, расшифрованных в комментариях:

```
TEST E      V1,X342    ;ЗНАЧЕНИЯ V1 И X342 РАВНЫ
TEST L      S$DD,10    ;СОДЕРЖИМОЕ ПАМЯТИ DD<10
TEST LE     Q6,17          ;В 6-Й ОЧЕРЕДИ≤17 ТРАНЗ-В
TEST NE     X903,X104 ;X903≠X104
```

Если поле C содержит метку блока, то транзакт при невыполнении отношения между операндами A и B будет передан к указанному блоку. При выполнении отношения транзакт перейдет к следующему после TEST блоку. Режим выполнения блока TEST с меткой в поле C называется режимом условного перехода.

Например, в следующих блоках транзакты будут передаваться по метке, записанной в поле C, если *не выполняются* условия, расшифрованные в комментариях:

```
TEST GE     V$A,V$D,OTHER    ;V$A≥V$D
TEST L      RN1,600,ALTRN    ;RN1<600.
```

В блоке TEST можно в качестве СЧА использовать и булевские переменные, например:

```
TEST E      BV$PROV,1   ПРОПУСТИТЬ,ЕСЛИ BV$PROV=1
TEST E      BV7,0,ABC   ЕСЛИ BV7=1, ИДТИ К ABC.
```

Применение булевских переменных позволяет проверять в блоке TEST условия произвольной сложности.

7.22. Сбор статистики с помощью очередей

Некоторые виды статистических данных накапливаются симулятором автоматически. Другие виды данных могут быть получены при помощи блоков для сбора статистики . Эти блоки связаны с объектами типа очередь и таблица .

Очередь - это множество, в которое транзакты включаются в момент прохождения ими блока QUEUE (войти в очередь) и из которого они исключаются в момент прохождения блока DEPART (выйти из очереди). В поле A этих блоков задается номер очереди.

Очередь характеризуется стандартным числовым атрибутом, называемым *длиной очереди*. В начальный момент времени, когда очередь пуста, ее длина равна нулю. Длина очереди меняется в моменты входа транзактов в блоки QUEUE и DEPART. Если поле B пустое, то при входе транзакта в блок QUEUE

длина соответствующей очереди увеличивается на единицу, а при входе в блок QUEUE - на единицу уменьшается. Следовательно, при использовании пустого поля B длина очереди равна текущему числу транзактов в этой очереди. В поле B блоков QUEUE и DEPART можно задавать произвольное число единиц, на которое должна увеличиваться или уменьшаться длина очереди. Транзакты могут проходить любое число блоков QUEUE и DEPART с произвольными значениями полей A и B, чередующихся в любом порядке, лишь бы это не приводило к отрицательным длинам образуемых очередей.

Симулятор следит за очередями в модели и подсчитывает для них определенную статистику, которую можно рассмотреть на следующем примере.

Пример12.Модель, построенную в условиях примера 8, изменим путем добавления блока QUEUE 1 перед блоком SEIZE 1 и блока DEPART 1 после блока SEIZE 1. В результате эта модель примет вид:

```
*   МОДЕЛЬ РАБОТЫ КАССИРА
        SIMULATE
        GENERATE        20,10,0,200     ;ПРИХОД ЗРИТЕЛЕЙ
        ADVANCE         15,15           ;ЗНАКОМСТВО С ОБСТ-КОЙ
        QUEUE           1               ;ВКЛЮЧЕНИЕ В ОЧЕРЕДЬ
        SEIZE           1               ;ОБРАЩЕНИЕ К КАССИРУ
        DEPART          1               ;ВЫХОД ИЗ ОЧЕРЕДИ
        ADVANCE         20,5            ;ПОКУПКА БИЛЕТА
        RELEASE         1               ;ОСВОБОЖДЕНИЕ КАССИРА
        TERMINATE 1                 ;УХОД ЗРИТЕЛЯ
        START           200             ;СЧЕТЧИК=200
```

В этой модели момент включения каждого транзакта в очередь 1 совпадает с моментом обращения его к блоку SEIZE, т.к. блок QUEUE выполняется в модельном времени мгновенно. В очереди каждый транзакт находится до тех пор, пока не займет устройство 1. Момент занятия устройства совпадает с моментом выхода транзакта из очереди. Таким образом, очередь 1 состоит из транзактов, задержанных при обращении к блоку SEIZE из-за занятости устройства 1. Следовательно, в данной модели очередь 1 имеет естественную интерпретацию как очередь зрителей к кассиру, а длина очереди 1 интерпретируется как число зрителей в очереди .

При наличии в модели очередей симулятор выдает статистику по очередям в файл REPORT.GPS в виде таблицы.

Применение блоков QUEUE и DEPART в рассмотренном примере является типичным. Другой типичный способ применения этих блоков состоит в обхватывании ими произвольного сегмента модели, по пребыванию транзактов в котором нужна статистика. Так, рассмотренную в примере модель можно расширить еще одной парой блоков QUEUE и DEPART: блок QUEUE 2 можно

вставить непосредственно после блока GENERATE, а блок DEPART 2 - перед блоком TERMINATE .

7.23. Сбор статистики с помощью таблиц

В отличие от очередей таблицы позволяют определять любые статистические характеристики любых случайных величин в модели .

Объект типа таблица представляет собой эквивалент понятия “статистическая гистограмма”. Гистограммы применяются для статистического анализа такой случайной величины, функция распределения которой неизвестна, но зато имеется достаточно большое число независимых реализаций этой величины.

Гистограмма задается набором каких-либо выделенных интервалов изменения случайной величины и количеством попаданий ее реализованных значений в каждый из интервалов. Графически гистограмма изображается набором прямоугольников, основания которых совпадают с выделенными интервалами, а высоты изображают соответствующие количества попаданий. Например, для некоторой случайной величины X реализовано 200 значений и в результате получена следующая гистограмма (рис.17).

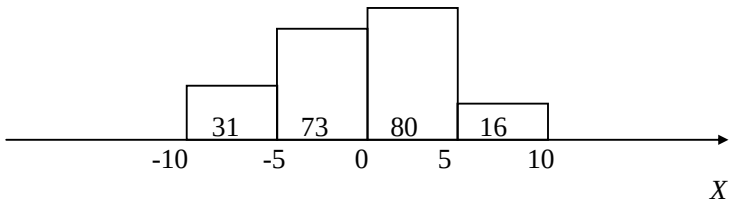


Рис.17

Выделяемые в таблице интервалы называются частотными, а количество попаданий случайной величины в эти интервалы - частотами.

Для рис.17 частотные интервалы - это интервалы x≤-10, -10< x ≤-5, ..., x≥10, а частоты - это величины 0,31,73,80,16 и 0. Исследуемая случайная величина, которая фиксируется в таблице, называется аргументом таблицы.

Для того, чтобы таблицы можно было использовать в модели, они должны быть описаны.

Для описания таблицы применяется строка TABLE. В поле метки этой строки задается номер или имя таблицы, в поле A - аргумент таблицы в виде СЧА, в поле B - верхняя граница первого частотного интервала, в поле C - ширина интервалов, а в поле D - их число, включающее оба полубесконечных интервала. Например, если аргументом табл. GIS должна быть переменная V$ABC, а интервалы выделяются так, как на рис.17, то строка описания таблицы будет иметь вид

```
GIS      TABLE        V$ABC,-10,5,6.
```

Если таблица описана, то транзакты могут фиксировать в ней информацию с помощью блока TABULATE. В поле A этого блока указывается имя таблицы, в которой накапливается информация. Если поле B пустое, то при входе транзакта в блок TABULATE вычисляется значение аргумента указанной таблицы и определяется, в какой из интервалов таблицы это значение попадает. После этого счетчик соответствующей интервальной частоты увеличивается на 1. В поле B может быть задано некоторое число - тогда счетчик частоты увеличивается на это число.

В результате моделирования по каждой таблице выдается информация в файл REPORT.GPS. Графическое представление гистограмм таблиц отображается в окне таблиц, в которое можно перейти с помощью клавиш Alt+T.

7.24. Стандартные числовые атрибуты

В приведенных ниже обозначениях СЧА индекс j соответствует номеру объекта, косвенному номеру объекта или присоединенному имени объекта.

СЧА могут записываться в качестве операндов практически в любом типе блоков.

Если конкретный операнд конкретного блока имеет ограничения на вид используемых в нем СЧА, то при его вводе осуществляется автоматический контроль допустимости СЧА. Кроме того, при вводе СЧА можно обратиться к контекстной помощи (клавиша «?»), которая покажет перечень всех СЧА, допустимых для данного операнда.

1) Атрибуты транзактов

P_j - параметр транзакта.

M1 - время прохождения участка модели транзактом, обрабатываемым в данный момент. Вычисляется по формуле M1 =(текущее время)-(отметка времени).

MP_j - время прохождения участка модели транзактом, обрабатываемым в данный момент. Вычисляется по формуле MP_j = (текущее время) - P_j. Для использования СЧА MP_j нужно в P_j предварительно занести отметку времени блоком MARK.

PR - приоритет обрабатываемого в данный момент транзакта.

2) Атрибуты блоков

N_j - число входов транзактов в блок j.

W_j - число транзактов, находящихся в блоке j.

3) Системные атрибуты

RN_j - псевдослучайное число. Если RN_j используется как аргумент функции, то $0 \le RN_j < 1$.

В остальных случаях $0 \le RN_j < 1000$.

4) Атрибуты оборудования

S_j - содержимое памяти j.

R_j - свободный объем памяти j.

SA_j - среднее содержимое памяти j (целая часть).

SR_j - коэффициент использования памяти j в частях от тысячи. Рассчитывается по формуле: $SR_j = 1000* SA_j$ / (емкость памяти j).

SM_j - максимальное содержимое памяти j.

SC_j - число входов в память j.

ST_j - среднее время пребывания транзактов в памяти j (целая часть).

F_j - состояние устройства j (0 - свободно, 1 - в остальных случаях).

FR_j - коэффициент использования усройства j в частях от 1000.

FC_j - число входов в устройство j.

FT_j - среднее время использования устройства j одним транзактом (целая часть).

5) Статистические атрибуты

Q_j - длина очереди j.

QA_j - средняя длина очереди j (целая часть).

QM_j - максимальная длина очереди j.

QC_j - число входов в очередь j.

QZ_j - число нулевых входов в очередь j.

QT_j - среднее время пребывания в очереди j с учетом нулевых входов (целая часть).

QX_j - среднее время пребывания в очереди j без учета нулевых входов (целая часть).

TB_j - среднее значение аргумента таблицы j (целая часть).

TC_j - число входов в таблицу j.

TD_j - оценка стандартного отклонения для таблицы j (целая часть).

6) Атрибуты ячеек SAVEVALUE

X_j - содержимое ячейки j.

MX_j (a, b) - содержимое ячейки j-ой матрицы ячеек, a - номер строки матрицы, b - номер столбца (a и b могут быть любыми СЧА).

7) Функции и переменные

FN_j - значение функции j. Если это значение не используется в качестве модификатора в блоках GENERATE, ASSIGN и ADVANCE или в качестве аргумента вещественной переменной, то сохраняется только целая часть.

V_j - значение j-ой арифметической переменной.

BV_j - значение булевской переменной (0 или1).

8) Атрибуты списков пользователя

CH_j - число транзактов в j-ом списке пользователя.

CA_j - среднее число транзактов в j-ом списке пользователя (целая часть).

CM_j - максимальное число транзактов в j-ом списке пользователя.

CC_j - число входов в j-ый список пользователя.

CT_j - среднее время пребывания транзакта в j-ом списке пользователя (целая часть).

9) Атрибуты группы

G_j -число элементов j-ой группы.

10) Системные атрибуты

AC1 – абсолютное время (сбрасывается командой CLEAR).

TG1 – количество окончаний прогонов, поступающее в счетчик текущих запусков.

XN1 – номер активного транзакта.

В приведенных выше обозначениях СЧА индекс j соответствует номеру объекта, косвенному номеру объекта или присоединенной метке объекта.

7.25. Блоки COUNT и SELECT

Блок COUNT служит для определения количества элементов, удовлетворяющих указанному условию. Он может содержать до пяти полей:

COUNT lo A, B, C [, D, E]

где lo – логический оператор. Если поля D, E отсутствуют, то lo представляет собой мнемонику СЛА, т.е. для устройств это будет FU, FNU, и т.д., для памятей – SE, SNE и т.д., для ключей – LR или LS;

A – номер параметра входящего транзакта, в который помещается результат подсчета объектов;

B – нижний предел номер анализируемых объектов;

C – верхний предел номеров объектов.

Тип объектов задавать не нужно, т.к. он однозначно определяется мнемоникой логического оператора. Множество объектов просматривается и подсчитывается число таких из них, для которых указанный СЛА имеет значение «истина». Полученное число заносится в параметр транзакта, заданный в поле A.

Если заданы также поля D и E, то логический оператор lo является оператором сравнения, т.е. имеет вид L, или LE, E, NE и т.д. В поле D тогда задается некоторая величина, а в поле E – СЧА объектов, сравниваемый с величиной поля E. СЧА задается в виде мнемоники без указания номера объекта, т.к. диапазон номеров объектов определен в полях B и C. Рапример, СЧА может быть задан в виде обозначения SR – тогда будут просматриваться памяти и их коэффициент использования будет сравниваться с величиной в поле E.

Пример:

COUNT LE 1, 1, 5, X10, FC.

Здесь в параметр 1 будет записано число устройств из множества от 1-го до 5-го устройства, у которых атрибут FC (счетчик входов) меньше или равен X10.

Блок SELECT применяется для нахождения номера первого объекта (из определенного диапазона объектов), удовлетворяющего заданному условию. Операции и операнды этого блока такие же, как и у блока COUNT, с той лишь разницей, что вместо числа соответствующих условию объектов находится номер первого такого объекта. И еще одно отличие состоит в том, что

логический оператор здесь может быть задан не только теми обозначениями, которые предусмотрены для блока COUNT, но также еще двумя: MIN и MAX.

Пример:

SELECT MIN 1,1,8,,Q.

Здесь будет найден номер самой короткой очереди из восьми (с первой по восьмую) и записан в первый параметр транзакта.

7.26. Список операторов GPSS\PC

В списке операторов, который приводится в данном параграфе, блоки выделены жирным шрифтом.

Остальные операторы – это операторы описания объектов модели и команды. Большинство из них могут входить как в состав текста модели, т.е. могут вводиться с номерами строк, так и выполняться в интерактивном режиме, - т.е. вводиться без номера строки. В интерактивном режиме введенный оператор исполняется немедленно и в исходный текст модели не включается.

Основные операторы достаточно подробно описаны в данном пособии. Необходимая информация о других операторах может быть получена из файлов документации, входящих в состав пакета GPSS\PC, и из контекстной помощи, вызываемой нажатием клавиши «вопросительный знак».

ADVANCE – задержка транзакта,

ALTER – изменение значений атрибутов транзактов, принадлежащих к заданной группе,

ANITRACE – сопоставить статистические ряды,

ANOVA - произвести анализ вариабельности в файле результатов,

ASSEMBLER – сборка транзактов,

ASSIGN – присвоить значение параметру транзакта,

BUFFER – переход к продвижению следующего транзакта в списке текущих событий,

BVARIABLE – описание булевской переменной,

CLEAR – очистка модели, переход в начальное состояние,

CONTINUE – продолжить выполнение модели,

COUNT – подсчитать число элементов в группе,

DELETE – удалить строку (строки) модели,

DEPART – выход транзакта из очереди,

DISPLAY – вывести текст модели на экран, данных,

EXAMINE – выбор пути транзакта в зависимости от того, принадлежит ли он к указанной группе,

DOS - приостановить сеанс и временно обратиться к DOS,

EDIT – редактировать строку модели,

END – выход из GPSS\PC,

ENTER – выход транзакта из памяти,

EQU – установка эквивалентности имени объекта и числа (тип объекта – в поле B),

FVARIABLE – описание вещественной переменной,

EVENTS - показать списки текущих и будущих событий в окне

EXECUTE – выполнить указанный блок,

FAVAIL – разблокирование устройства,

FUNAVAIL – блокирование устройства,

FUNCTION – описание функции,

GATE – проверка состояния оборудования или блоков MATCH,

GATHER – синхронизация движения транзактов,

GENERATE – генерация транзактов,

GROUPS - поместить члены групп транзактов и чисел в окно данных,

HELP – подключение модуля, написанного на другом языке программирования,

INDEX – увеличение параметра на заданную величину и запись результата в P1,

INITIAL – задание начальных значений ячеек, матриц и логических ключей,

JOIN – добавление числа или транзакта к группе,

LEAVE – выход транзакта из памяти,

LINK – перевод транзакта в список пользователя,

LOGIC – установка состояния логического ключа,

LOOP – повторить цикл,

MARK – отметить время в транзакте,

MATCH – парная синхронизация,

MATRIX – описание матрицы,

MICROWINDOW- открыть или закрыть микроокно,

MOVE - ,

MSAVEVALUE – сохранить значение в матрице ячеек,

PLOT- выдавать график СЧА в окне данных во время моделирования,

PREEMPT – захватить устройство,

PRIORITY – задать приоритет транзакта,

QTABLE – описать таблицу для очереди,

QUEUE – вход транзакта в очередь,

RELEASE – освобождение занятого устройства,

REMOVE – удаление транзакта или числа из группы,

RENUMBER - перенумеровать операторы в исходной программе,

REPORT - объявить файл для записи следующего отчета системы GPSS/PC,

RESET – сброс статистики без изменения состояния системы,

RESULT - добавить числовой результат в файл DOS,

RETURN – освобождение захваченного устройства,

RMULT – установка начального множителя генератора случайных чисел,

SAVAIL – разблокирование памяти,

SAVE - создать файл для исходной программы в целом или частично,

SAVEVALUE – сохранение значения в ячейке,

SCAN – сканирование группы для поиска транзакта с заданными атрибутами,

SEIZE – занятие транзактом устройства,

SELECT – выбор объекта, который отвечает указанному условию,

SHOW –- выполнить оператор и выдать результат в верхнее окно,

SIMULATE – объявления режима исполнения модели (рудимент от GPSS-360),

SPLIT – создать копии транзакта,

START – выполнить модель,

STEP - смоделировать несколько входов транзактов в блок, затем остановиться,

STOP -- остановиться, когда это запланировано транзактом или блоком,

STORAGE – описание емкости памяти,

SUNAVAIL – блокирование памяти,

TABLE – описание таблицы-гистограммы,

TABULATE – табулировать ,

TERMINATE – уничтожение транзакта,

TEST – проверка произвольного условия,

TRACE – выводить трассу транзакта в модели,

TRANSFER – пересылка транзакта,

UNLINK – удаление транзакта из цепи пользователя,

UNTRACE – прекратить вывод трассы транзакта,

USERCHAINS - поместить элементы списка пользователей в окно данных,

VARIABLE – описание целой арифметической переменной,

WINDOW - открыть основное окно системы GPSS/PC.

ГЛАВА 8. ПЛАНИРОВАНИЕ ИМИТАЦИОННОГО ЭКСПЕРИМЕНТА

8.1. Параллельный и последовательный эксперимент

Основная задача планирования имитационного эксперимента (ИЭ) заключается *в контроле достоверности* (точности) результатов моделирования.

Поскольку имитируемые процессы содержат обычно случайные факторы, то необходимо обоснование того, что результаты моделирования – а это *оценки* средних значений, вероятностей и других вероятностных характеристик (в.х.) - достаточно близки к истинным, теоретическим значениям этих в.х.

В практике имитационного моделирования систем сложились два основных способа осуществления имитационного эксперимента (ИЭ), называемые соответственно *параллельным* экспериментом и *последовательным* экспериментом [18].

Параллельный ИЭ – это «классический» эксперимент, в котором процесс воспроизводится многократно, и каждая реализация процесса (реплика) является статистически независимой от других реплик.

Параллельный эксперимент осуществляется путем многократного прогона имитационной модели при одних и тех же начальных условиях. Наблюдаемая с.в. в силу действия случайных факторов в каждом прогоне изменяется по разному. Ее зависимость от времени на протяжении одного прогона и представляет собой одну реплику.

При этом мы можем запомнить все значения интересующей нас с.в., которые она имеет в разных репликах в один и тот же фиксированный момент времени t, и рассматривать этот набор значений как независимую выборку. Действительно, все эти значения статистически независимы, т.к. независимы реплики. Следовательно, для этих значений можно находить оценку среднего, оценку дисперсии, доверительные интервалы оценок (например, по правилу «трех сигм» [7]), и т.д., основываясь на классических методах анализа независимых выборок.

Последовательный ИЭ применяется в случае моделирования *эргодических* процессов и основан на том, что среднее значение эргодического процесса, найденное «по времени», равно среднему, найденному «по множеству». Это значит, что если достаточно долго моделировать зависимость случайной величины от времени и определять ее среднее по времени (как интеграл по времени, деленный на время моделирования), то это среднее совпадет со средним, найденным по множеству реплик в параллельном эксперименте.

8.2. Пояснения к выбору типа ИЭ

Чтобы правильно ориентироваться в выборе типа эксперимента (параллельный или последовательный ?), надо ясно понимать две вещи.

Во-первых, важно уяснить, что такое эргодический процесс.

Во-вторых, - и это связано с понятием эргодического процесса, - следует знать, что последовательный ИЭ позволяет рассчитать оценки в.х. только для *стационарной* фазы процесса. Поэтому последовательный ИЭ, в отличие от «классического» параллельного, не обладает универсальными возможностями – он позволяет найти только небольшую часть в.х. системы.

Эргодическим называется такой случайный процесс, для которого стационарное среднее, найденное по времени (при $T \to \infty$) по любой реплике, совпадает со стационарным средним, найденным по множеству реплик.

Чтобы распознать отсутствие эргодичности у того или иного процесса, часто бывает достаточно обратить внимание на то, имеется ли в начальных фазах этого процесса фактор, который существенно влияет на всю его последующую историю.

Например, на рис.18 изображена замкнутая сеть массового обслуживания, в которой две заявки в начальный момент времени находятся в СМО 1. Ясно, что в.х. стационарного процесса, найденные по одной реплике, будут зависеть от того, как переместятся эти заявки с выхода первой СМО, - т.е. попадут ли они обе во вторую СМО, обе в третью, или разделятся между ними.

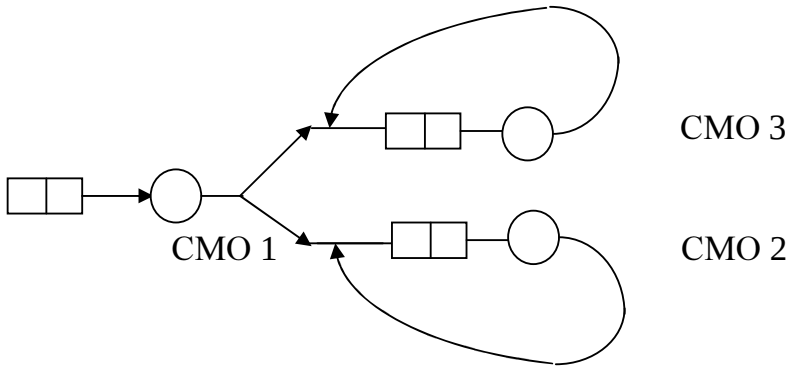


Рис.18

Таким образом, изображенная система *не эргодическая*, и ее в.х. следует определять с помощью параллельного ИЭ.

Вообще говоря, *сам случайный процесс* никогда не бывает стационарным. Скажем, длина очереди (*фактическая*, а не средняя) в экспоненциальной СМО (см.) всегда изменяется, и ее изменения случайны. Поэтому она не может быть стационарной.

Стационарными могут быть только те или иные *в.х.* случайного процесса. Так, *средняя* длина очереди в первый момент времени, когда еще ни одна заявка не поступила в СМО, равна нулю. Затем она постепенно увеличивается и приближается к стационарному значению, соответствующему времени $t = \infty$. Средняя длина очереди может иметь стационарное значение.

Когда мы говорим о *существовании стационарного режима* функционирования системы, то выражаемся не вполне корректно. Существование стационарного режима должно было бы означать, что при $t \to \infty$ все в.х. системы сходятся к некоторым постоянным значениям, не зависящим от времени.

Однако на практике чаще всего бывает так, что одни в.х. сходятся к постоянным значениям, а другие - нет. При этом скорость сходимости у разных в.х. разная. Поэтому, если судить с практической точки зрения, то по одним в.х. длительность переходного процесса бывает одна, по другим – другая. По одним в.х. стационарный процесс существует, а по другим – нет.

Например, для простейшей одноканальной экспоненциальной СМО классическое правило звучит так: если коэффициент загрузки ρ не превышает единицы, то стационарный режим существует, иначе – нет. Но в действительности при $\rho>1$ хотя и нет стационарного значения для средней длины очереди (она постоянно растет *в среднем*), однако есть стационарное значение *вероятности* застать канал занятым (оно равно единице), есть стационарное значение *средней скорости прироста* длины очереди (оно равно отношению интенсивности прихода заявок к интенсивности их обслуживания), и т.д.

Обратим внимание также и на то, что при $\rho<1$, когда стационарный режим называют в теории «*существующим*», на самом деле *не имеется* стационарного среднего значения у таких переменных, как общее число заявок, обслуженных в СМО, суммарное время ожидания заявками обслуживания и т.д.

Очевидно, в любой системе всегда имеются переменные, которые с ростом времени стремятся к стационарным значениям, и имеются другие переменные, которые с ростом времени не сходятся к константам.

Добавим к этому, что существует особый класс стационарных решений – периодические. Так, в практических заданиях (гл. 11) имеются СМО, у которых в стационарном режиме средняя длина очереди и вероятность отказа представляют собой периодические функции времени.

8.3. Учет содержательного смысла задачи

При планировании имитационного эксперимента обычно весьма существенное значение имеют не только свойства конкретной исследуемой системы, но и такие содержательные моменты, как цель исследования, мощность используемого компьютера, ограничения по срокам моделирования, требуемая точность оценок и т.д.

Как правило, при этом ни один из важных факторов, влияющих на план эксперимента, не бывает известен достаточно хорошо до тех пор, пока какая-то часть экспериментов не будет выполнена. Это означает, что планирование экспериментов фактически осуществляется в ходе самих этих экспериментов, с учетом получаемой в них конкретной информации.

Это можно ясно видеть из примеров моделирования, рассмотренных в данном пособии. По сути дела, все приведенные в гл. 4 – 6 задачи решались по индивидуальному сценарию, целесообразному только для конкретной решаемой задачи. Эти сценарии «нащупывались» опытным путем, который в тексте пособия явным образом не отражен, так как содержит множество пробных действий, которые впоследствии отбрасывались как неудачные.

Единственный случай корректировки хода эксперимента приводится в явном виде в примере моделирования конвейера. Здесь исходная система после оптимизации (вариант с сохранением производственного задела между сменами) превратилась из стационарной в не стационарную. Из-за этого продолжение экспериментов с нею по методу последовательного эксперимента стало ошибочным. Ясно, что в этой ситуации необходимо либо перейти к параллельному эксперименту, либо внести в систему еще одно изменение, которое сделало бы ее стационарной. Предпочтение было отдано второму решению, – т.е. стационарному, - т.к. в.х. не стационарной системы слишком нестабильны, чтобы можно было ориентироваться на них в практике.

Достаточно простое и полное описание математических методов планирования имитационного эксперимента приводится в [8]. Здесь наряду с прочими описаны методы ускорения последовательного эксперимента с помощью отбрасывания статистики, собранной на этапе переходного процесса, методы ускоренного сравнения систем за счет введения положительной корреляции в выборку, метод существенной выборки (метод взвешивания), который позволяет ускорить нахождение средних значений без потери точности, а также метод стратификации (метод расслоения выборки), с помощью которого достигается многократное сокращение затрат машинного времени на эксперимент. Последние два метода (метод существенной выборки и метод стратификации) изложены также в [7].

ГЛАВА 9. ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОГО ВЫПОЛНЕНИЯ НА GPSS\PC

Ниже приводятся задачи, для решения которых требуется выполнить небольшое исследование. В основном эти задачи заимствованы из [6].

Рекомендуется следующий порядок их решения.

На первом этапе следует построить **схему процесса** функционирования системы, на которой отражалось бы движение перемещающихся в системе элементов (которые будут моделироваться транзактами), временные задержки этих элементов и места возможного возникновения очередей. Символы, используемые в схеме, не обязаны соответствовать каким-либо стандартам. Можно в качестве примеров руководствоваться обозначениями, которые применялись в данном пособии при моделировании швейной фабрики, конвейера и т.д. На этом этапе следует добиться четкого представления о логике движения транзакта, обо всех взаимодействиях, возникающих в системе. На этом же этапе желательно приблизительно определиться с тем, какие элементы системы будут моделироваться какими объектами языка GPSS.

На втором этапе рекомендуется выполнить приблизительное **аналитическое исследование** системы. Как правило, можно приближенно или точно рассчитать коэффициенты загрузки ресурсов системы, оценить вероятности тех или иных событий, максимальные и минимальные возможные длительности определенных этапов и т.д.

Здесь невозможно дать какие-либо общие методы анализа. Примеры предварительного анализа систем также можно найти в данном пособии. Хорошо, если такой предварительный анализ позволяет понять, где могут возникнуть «узкие места» (обычно они возникают там, где высока загрузка ресурсов) и сформулировать гипотезу о возможных недостатках системы и способах их устранения.

Результативность предварительного анализа несколько зависит от соответствующей математической подготовки и от интуиции, воспитанной на опыте анализа аналогичных систем. Поэтому в следующей главе пособия приводится краткая теория анализа простейших систем массового обслуживания с примером приблизительного анализа сложной системы.

Что касается анализа систем в приведенных ниже задачах, то нет такой теории, которая могла бы быть применена к ним. Методы точного аналитического решения этих задач (кроме двух-трех из них) никому не известны и, судя по всему, в ближайшие лет сто не появятся.

Однако те или иные характеристики систем могут быть точно или приближенно найдены, и это весьма полезно сделать по следующим причинам.

Во-первых, сам ход аналитического исследования дает такую степень понимания исследуемого процесса, которая позволяет качественно предвосхитить результаты последующего имитационного моделирования и избежать возможных принципиальных ошибок.

Во-вторых, в ходе анализа устанавливается множество точно определенных взаимосвязей между величинами, что позволит обнаруживать мелкие ошибки, в случае появления которых эти взаимосвязи обычно нарушаются.

В-третьих, найденные аналитически характеристики или их оценки можно сравнить с численными результатами моделирования. Соответствие аналитических и модельных результатов косвенно подтвердит их достоверность; несоответствие явно укажет на наличие ошибок и на необходимость их поиска и исправления.

По своему личному опыту имитационного моделирования автор может определенно утверждать, что предварительный анализ делает исследуемую систему и ее свойства «прозрачными», даже если собственно числовых результатов анализа удается найти немного. Разумеется, такая прозрачность достигается только путем достаточно упорных и длительных усилий по анализу, путем достаточно настойчивой концентрации внимания на исследуемой системе.

На третьем этапе разрабатывается **моделирующая программа,** и выполняются пробные эксперименты, подтверждающие работоспособность имитационной модели. Здесь могут применяться любые методы и подходы, распространенные в программировании.

Четвертым этапом рекомендуется сделать этап **планирования эксперимента**. Целью планирования является экспериментальное определение такого объема статистической выборки, который гарантирует достижения требуемой точности результатов моделирования.

Требования к точности зависят от решаемой задачи.

Для планирования эксперимента важно сразу определиться с тем, какие данные требуется получить: данные о переходном процессе или только данные о стационарном режиме функционирования системы. В ряде случаев приходится моделировать системы, которые в принципе не работают в стационарном режиме. Соответственно этому приходится планировать параллельный эксперимент или можно удовлетвориться экспериментом последовательным. Примеры планирования эксперимента и соответствующей его организации на GPSS\PC приводились в данном пособии. Как видно из примера с моделированием конвейера, планирование эксперимента может понадобиться выполнить несколько раз – для разных вариантов системы по разному.

На пятом этапе выполняется **имитационный эксперимент,** интерпретируются его результаты и проверяются гипотезы, сформулированные на этапе аналитического исследования. Предлагаются пути улучшения функционирования системы, усовершенствования ее организации.

Можно изменять параметры системы или ее структуру (организацию функционирования). Необходимо иметь в виду, что, как правило, параметры внешней среды (такие, как интенсивность поступления транзактов из внешней среды, их свойства и т.п.) на практике мы менять не можем. Изменение параметров системы, таких, как время обработки транзактов в устройствах, объемы памятей и т.д. мы можем свободно изменять, когда система проектируется. Но, если моделируются уже существующая система, то изменение ее параметров связано с определенными затратами. Так, чтобы изменить скорость работы устройства, нужно обычно купить или сделать новое устройство и поместить его на место старого. Поэтому, если речь идет о существующей системе, то предпочтительны изменения в ее организации, т.е. в стратегии обработки транзактов, в способе выбора ими маршрутов и т.п. Пример такого усовершенствования системы был рассмотрен при моделировании конвейера.

На последнем, шестом этапе рекомендуется проверить, что дает применение предложенного варианта усовершенствования системы. Для этого соответствующее изменение вносится в моделирующую программу и выполняется **моделирование улучшенного варианта системы**.

1. На сборочный участок цеха предприятия через интервалы времени, распределенные экспоненциально со средним значением 10 мин, поступают партии, каждая из которых состоит из трех деталей. Половина всех поступающих деталей перед сборкой должна пройти предварительную обработку в течение 7 мин. На сборку подаются обработанная и необработанная детали. Процесс сборки занимает всего 6 мин. Затем изделие поступает на регулировку, длящуюся в среднем 8 мин (время выполнения ее распределено экспоненциально). В результате сборки возможно появление 4% бракованных изделий, которые не поступают на регулировку, а направляются снова на предварительную обработку.

Смоделировать работу участка. Определить возможные места появления очередей и их вероятностно-временные характеристики. Выявить причины возникновения очередей, предложить меры по их уменьшению и смоделировать скорректированную систему.

2. На обрабатывающий участок цеха поступают детали в среднем через 50 мин. Первичная обработка деталей производится на одном из двух станков. Первый станок обрабатывает деталь в среднем 40 мин и имеет до 4% брака,

второй - соответственно 60 мин и 8% брака. Все бракованные детали возвращаются на повторную обработку на второй станок. Детали, попавшие в разряд бракованных дважды, считаются отходами. Вторичную обработку проводят также два станка в среднем 100 мин каждый. Причем первый станок обрабатывает имеющиеся в накопителе после первичной обработки детали, а второй станок подключается при образовании в накопителе задела больше 3 деталей. Все интервалы времени распределены по экспоненциальному закону.

Смоделировать обработку деталей на участке. Определить загрузку второго станка на вторичной обработке и вероятность появления отходов. Определить возможность снижения задела в накопителе и повышения загрузки второго станка на вторичной обработке.

3. На регулировочный участок цеха через случайные интервалы времени поступают по два агрегата в среднем через каждые 30 мин. Первичная регулировка осуществляется для двух агрегатов одновременно и занимает около 30 мин. Если в момент прихода агрегатов предыдущая партия не была обработана, поступившие агрегаты на регулировку не принимаются. Агрегаты после первичной регулировки, получившие отказ, поступают в промежуточный накопитель. Из накопителя агрегаты, прошедшие первичную регулировку, поступают попарно на вторичную регулировку, которая выполняется в среднем за 30 мин, а не прошедшие первичную регулировку поступают на полную, которая занимает 100 мин для одного агрегата. Все величины, заданные средними значениями, распределены экспоненциально.

Смоделировать работу участка. Определить вероятность отказа в первичной регулировке и загрузку накопителя агрегатами, нуждающимися в полной регулировке. Ввести в систему накопитель, обеспечивающий безотказное обслуживание поступающих агрегатов, и определить его параметры.

4. 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 буферизируются в накопителе емкостью 20 пакетов и передаются по любой из двух линий АВ1 - за время 20 мс или АВ2 - за время 20±5 мс. В пункте В они снова буферизуются в накопителе емкостью 25 пакетов и далее передаются по линиям ВС1 (за 25±3 мс) и ВС2 (за 25 мс). Причем пакеты из АВ1 поступают в ВС1, а из АВ2 - в ВС2. Чтобы не было переполнения накопителя, в пункте В вводится пороговое значение его емкости - 20 пакетов. При достижении очередью порогового значения происходит подключение резервной аппаратуры, и время передачи снижается для линий ВС1 и ВС2 до 15 мс.

Смоделировать прохождение пакетов через систему передачи данных. Определить вероятность подключения резервной аппаратуры и характеристики очереди пакетов в пункте В. В случае возможности переполнения накопителя определить необходимое для нормальной работы пороговое значение его емкости.

5. Система обработки информации содержит мультиплексный канал и три мини ЭВМ. Сигналы от датчиков поступают на вход канала через интервалы времени 10±5 мкс. В канале они буферизируются и предварительно обрабатываются в течение 10±3 мкс. Затем они поступают на обработку в ту мини ЭВМ, где имеется наименьшая по длине входная очередь. Емкости входных накопителей во всех мини- ЭВМ рассчитаны на хранение величин 10 сигналов. Время обработки сигнала в любой мини ЭВМ равно 33 мкс.

Смоделировать процесс обработки сигналов, поступающих с датчиков. Определить средние времена задержки сигналов в канале и мини ЭВМ и вероятности переполнения входных накопителей. Обеспечить ускорение обработки сигнала в ЭВМ до 25 мкс при достижении суммарной очереди сигналов значения 25 единиц.

6. На участке термической обработки выполняются цементация и закаливание шестерен, поступающих через 10±6 мин. Цементация занимает 10±7 мин, а закаливание - 10±6 мин. Качество определяется суммарным временем обработки. Шестерни со временем обработки больше 25 мин покидают участок, со временем обработки от 20 до 25 мин передаются на повторную закалку и при времени обработки меньше 20 мин должны пройти повторную полную обработку. Детали с суммарным временем обработки меньше 20 мин считаются вторым сортом.

Смоделировать процесс обработки шестерен на участке. Определить функцию распределения времени обработки и вероятности повторения полной и частичной обработки. При выходе продукции без повторной обработки менее 90% обеспечить на участке мероприятия, дающие гарантированный выход продукции первого сорта 90 %.

7. Магистраль передачи данных состоит из двух каналов (основного и резервного) и общего накопителя. При нормальной работе сообщения передаются по основному каналу за 7±3с. В основном канале происходят сбои через интервалы времени 200±35с. Если сбой происходит во время передачи, то за 2 с запускается запасной канал, который передает прерванное сообщение с самого начала. Восстановление основного канала занимает 23±7с. После восстановления резервный канал выключается и основной канал продолжает

работу с очередного сообщения. Сообщения поступают через 9±4с и остаются в накопителе до окончания передачи. В случае сбоя передаваемое сообщение передается повторно по запасному каналу.

Смоделировать работу магистрали передачи данных в течение 1ч. Определить загрузку запасного канала, частоту отказов канала и число прерванных сообщений. Определить функцию распределения времени передачи сообщений по магистрали.

8. В системе передачи данных осуществляется обмен пакетами данных между пунктами А и В по дуплексному каналу связи. Пакеты поступают в пункты системы от абонентов с интервалами времени между ними 10±3 мс. Передача пакета занимает 10 мс. В пунктах имеются буферные регистры, которые могут хранить два пакета (включая передаваемый). В случае прихода пакета в момент занятости регистров пунктам системы предоставляется выход на спутниковую полудуплексную линию связи, которая осуществляет передачу пакетов данных за 10±5 мс. При занятости спутниковой линии пакет получает отказ.

Смоделировать обмен информацией в системе передачи данных. Определить частоту вызовов спутниковой линии и ее загрузку. В случае возможности отказов определить необходимый для безотказной работы системы объем буферных регистров.

9. Специализированная вычислительная система состоит из трех процессоров и общей оперативной памяти. Задания, поступающие на обработку через интервалы времени 5±2 мин, занимают объем оперативной памяти размером в страницу. После трансляции первым процессором в течение 5±1 мин их объем увеличивается до двух страниц, и они поступают в оперативную память. Затем после редактирования во втором процессоре, которое занимает 2,5±0,5 мин на страницу, объем возрастает до трех страниц. Отредактированные задания через оперативную память поступают в третий процессор на решение, требующее 1,5±0,4 мин на страницу, и покидают систему, минуя оперативную память.

Смоделировать работу вычислительной системы. Определить характеристики занятия оперативной памяти по всем трем видам заданий.

10. К мини ЭВМ подключено четыре терминала, с которых осуществляется решение задач. По команде с терминала выполняют операции редактирования, трансляции, планирования и решения. Причем, если хоть один терминал выполняет планирование, остальные вынуждены простаивать из-за нехватки оперативной памяти. Если два терминала выдают требование на решение, то

оставшиеся два простаивают, и если работают три терминала, выдающих задания на трансляцию, то оставшийся терминал блокируется. Интенсивности поступления задач различных типов равны. Задачи одного типа от одного терминала поступают через экспоненциально распределенные интервалы времени со средним значением 160с. Выполнение любой операции длится 10с.

Смоделировать работу мини ЭВМ. Определить загрузку процессора, вероятности простоя терминалов и частоту одновременного выполнения трансляции с трех терминалов.

11. В системе передачи цифровой информации передается речь в цифровом виде. Речевые пакеты передаются через два транзитных канала, буферизуясь в накопителях перед каждым каналом. Время передачи пакета по каналу составляет 5 мс. Пакеты поступают через 6±мс. Пакеты, передававшиеся более 10 мс, на выходе системы уничтожаются, так как их появление в декодере значительно снизит качество передаваемой речи. Уничтожение более 30% пакетов недопустимо. При достижении такого уровня система за счет ресурсов ускоряет передачу до 4 мс на канал. При снижении уровня до приемлемого происходит отключение ресурсов.

Смоделировать работу системы. Определить частоту уничтожения пакетов и частоту подключения ресурса.

12. В узел коммутации сообщений, состоящий из входного буфера, процессора, двух исходящих буферов и двух выходных линий, поступают сообщения с двух направлений. Сообщения с одного направления поступают во входной буфер, обрабатываются в процессоре, буферизируются в выходном буфере первой линии и передаются по выходной линии. Сообщения со второго направления обрабатываются аналогично, но передаются по второй выходной линии. Применяемый метод контроля потоков требует одновременного присутствия в системе не более трех сообщений на каждом направлении. Сообщения поступают через интервалы 15±7 мс. Время обработки в процессоре равно 7 мс на сообщение, время передачи по выходной линии равно 15±5 мс. Если сообщение поступает при наличии трех сообщений в направлении, то оно получает отказ.

Смоделировать работу узла коммутации. Определить загрузки устройств и вероятность отказа в обслуживании из-за переполнения буфера направления. Определить изменения в функции распределения времени передачи при снятии ограничений, вносимых методом контроля потоков.

13. Из литейного цеха на участок обработки и сборки поступают заготовки через 20±5 мин. Треть из них обрабатывается в течение 60 мин и поступает

на комплектацию. Две трети заготовок обрабатывается за 30 мин перед комплектацией, которая требует наличия одной детали первого типа и двух деталей второго. После этого все три детали подаются на сборку, которая занимает 60±2 мин для первой детали и 60±8 мин для двух других, причем они участвуют в сборке одновременно. При наличии на выходе одновременно всех трех деталей изделие покидает участок.

Смоделировать работу участка. Определить места образования и характеристики возможных очередей.

14. Детали, необходимые для работы цеха, находятся на цеховом и центральном складах. На цеховом складе хранится 20 комплектов деталей, потребность в которых возникает через 60±10 мин и составляет один комплект. В случае снижения запасов до трех комплектов формируется в течение 60 мин заявка на пополнение запасов цехового склада до полного объема в 20 комплектов, которая посылается на центральный склад, где в течение 60±20 мин происходит комплектование и за 60±5 мин осуществляется доставка деталей в цех.

Смоделировать работу цеха. Определить вероятность простоя цеха из-за отсутствия деталей и среднюю загрузку цехового склада. Определить момент пополнения запаса цехового склада, при котором вероятность простоя цеха будет равна нулю.

15. Для обеспечения надежности АСУ ТП в ней используется две ЭВМ. Первая ЭВМ выполняет обработку данных о технологическом процессе и выработку управляющих сигналов, а вторая находится в "горячем резерве". Данные в ЭВМ поступают через 10±2с, обрабатываются в течение 3с, затем посылается управляющий сигнал, поддерживающий заданный темп процесса. Если к моменту посылки следующего набора данных не получен управляющий сигнал, то интенсивность выполнения технологического процесса уменьшается вдвое и данные посылаются через 20±4с. Основная ЭВМ каждые 30с посылает резервной ЭВМ сигнал о работоспособности. Отсутствие сигнала означает необходимость включения резервной ЭВМ вместо основной. Характеристики обеих ЭВМ одинаковы. Подключение резервной ЭВМ занимает 5с, после чего она заменяет основную до восстановления, а процесс возвращается к нормальному темпу. Отказы ЭВМ происходят через 300±30с. Восстановление занимает 100с. Резервная ЭВМ абсолютно надежна.

Смоделировать работу системы. Определить среднее время нахождения технологического процесса в заторможенном состоянии и среднее число пропущенных из-за отказов данных.

16. На вычислительный центр через 300±100 поступают задания длиной 500±200 байт. Скорость ввода, вывода и обработки заданий - 100 байт/мин. Задания проходят последовательно ввод, обработку и вывод, буферируясь перед каждой операцией. После вывода 5% заданий оказываются выполненными неправильно вследствие сбоев и возвращаются на ввод. Для ускорения обработки задания в очередях располагаются по возрастанию их длины, т.е. короткие сообщения обслуживают в первую очередь. Задания, выполненные неверно, возвращаются на ввод и во всех очередях обслуживаются первыми.

Определить необходимую емкость буферов и функцию распределения времени обслуживания заданий.

17. Вычислительная система включает три ЭВМ. В систему в среднем через 30с поступают задания, которые попадают в очередь на обработку к первой ЭВМ, где они обрабатываются около 30с. После этого задание поступает одновременно во вторую и третью ЭВМ. Вторая ЭВМ может обработать задание за 140±5с, а третья - за 16±1с. Окончание обработки задания на любой ЭВМ означает снятие ее с решения с той и другой машины. В свободное время вторая и третья ЭВМ заняты обработкой фоновых задач.

Смоделировать работу системы. Определить необходимую емкость накопителей перед всеми ЭВМ, коэффициенты загрузки ЭВМ и функцию распределения времени обслуживания заданий. Определить производительность второй и третьей ЭВМ на решении фоновых задач при условии, что одна фоновая задача решается 2 мин.

Наряду с приведенными здесь задачами для лучшего усвоения языка GPSS и технологии имитационного моделирования можно воспользоваться задачами, которые приводятся вместе с подробным изложением их решения в фундаментальной, но, к сожалению, весьма редкой книге [1]. Обратим внимание в связи с этим на то, что в пакет GPSS\PC входят 27 файлов с именами schr2a.gps, schr2b.gps, schr2c.gps, schr2d.gps, schr2e.gps, schr2f.gps, …, schr7d.gps, которые являются моделями к примерам с номерами соответственно 2a, 2b,2c,2d,2e,2f, … ,7d из книги Шрайбера [4]. Модели написаны на языке GPSS\PC, в то время как в [1] они даны на GPSS/360. Это позволяет тем, кто имеет практику работы на GPSS/360, ознакомиться с приемами более эффективного построения моделей, которое возможно в GPSS\PC.

Несколько задач по моделированию с их решениями можно найти также в [7]. В [9-16] приводится обширная информация как по языку GPSS, так и по его применениям. В частности, в [13-16] разработаны концептуальные модели имитационного моделирования информационно-вычислительных систем, и эти модели доведены до программной реализации на языке GPSS.

ГЛАВА 10. РАСЧЕТ ПРОСТЕЙШИХ СИСТЕМ МАССОВОГО ОБСЛУЖИВАНИЯ

10.1. Одноканальная экспоненциальная СМО

Одноканальная система массового обслуживания (СМО) задается следующими свойствами. СМО имеет *канал*. В СМО приходят *заявки.* Если СМО пустая (нет заявок), то приходящая заявка занимает канал. Заявка, приходящая в непустую СМО (канал занят), становится в очередь к каналу. Любая заявка, занявшая канал, обслуживается, освобождает канал и уходит из СМО. Если в момент ухода очередь не пустая, то первая в ней заявка выходит из очереди и занимает канал.

Символическое изображение одноканальной СМО приведено на рис.19.

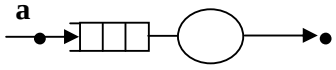


Рис.19

Кружком (рис. 19) обозначен канал, прямоугольниками – очередь к каналу. Стрелки указывают направление движения заявок. Точки у стрелок - вход и выход СМО. Буква **a** обозначает среднее время между приходами заявок, буква **T -** среднее время обслуживания заявки.

В *экспоненциальной* СМО приходы заявок образуют пуассоновский поток событий. Это означает, что время *x* между приходами любых двух последовательных заявок есть такая независимая случайная величина (с. в.), которая имеет экспоненциальное распределение вероятностей. Кроме того, в экспоненциальной СМО время обслуживания заявок τ также распределено экспоненциально.

Таким образом, в экспоненциальной СМО функция распределения вероятностей (ф.р.в.) случайных величин *x* и τ имеет следующий вид:

$$F(t) = 1 - e^{-\lambda t} \qquad (1)$$

где λ - параметр распределения. Как известно, этот параметр есть величина, обратная математическому ожиданию распределения. Следовательно, для с.в. *x* параметр λ равен 1/**a** и представляет собой *интенсивность поступления* заявок, т.е. среднее число заявок, приходящих в единицу времени. Для с.в. τ параметр λ равен 1/**T** и представляет *интенсивность обслуживания* заявок. В

дальнейшем интенсивность обслуживания заявок в СМО будем обозначать через μ.

Цель анализа СМО заключается в расчете ее *характеристик*, важнейшие из которых следующие:

- коэффициент загрузки ρ;
- средняя длина очереди **L**;
- среднее число заявок в СМО **N**;
- среднее время ожидания обслуживания **W**;
- среднее время пребывания заявки в СМО **U**.

Коэффициент загрузки рассчитывается по формуле

$$\rho = \lambda \mathbf{T}. \qquad (2)$$

Если выполняется условие

$$\rho \le 1, \qquad (3)$$

то существует *стационарный режим* функционирования СМО.

В стационарном режиме такие вероятностные характеристики происходящих в СМО процессов, как **L, M, W, U** не зависят от времени, т.е. являются постоянными величинами. Сами происходящие в СМО процессы остаются при этом случайными.

Если соотношение (3) не выполняется, то стационарного режима в СМО не существует. Действительно, если $\rho > 1$, то, как вытекает из (2), $\lambda \mathbf{T} > 1$, т.е. $\lambda > 1/\mathbf{T}$, или, что то же самое, $\lambda > \mu$. Это значит, что интенсивность поступления заявок в СМО при $\rho > 1$ превышает интенсивность их обслуживания каналом (канал обслуживает заявки в среднем медленнее, чем те поступают).Поэтому длина очереди заявок будет возрастать каждую единицу времени в среднем на величину ($\lambda - \mu$). Со временем очередь будет постоянно возрастать и стационарного значения у средней длины очереди не будет.

В *стационарном режиме* среднее число **N** заявок в СМО постоянно. Поэтому среднее число заявок, поступающих в СМО в единицу времени, равно среднему числу заявок, в единицу времени из СМО уходящих. Следовательно, в стационарном режиме интенсивность потока уходящих заявок равна λ. Эта величина меньше, чем интенсивность обслуживания μ, т.к. канал в стационарном режиме определенную долю времени простаивает. Эта доля времени равна $(1-\rho)$ – коэффициенту простоя СМО. Интенсивность выходного потока заявок равна интенсивности обслуживания, когда канал занят, и равна нулю, когда канал свободен. В среднем же она равна интенсивности поступления заявок. Поэтому среднее число заявок в СМО в стационарном режиме со временем не изменяется, остается постоянным.

Требование (3) является условием стационарности не только для экспоненциальных СМО, но и для любых других. «Физический смысл» этого требования состоит в том, чтобы «скорость поступления работы» в систему не превышала «скорости ее выполнения» системой.

Коэффициент загрузки ρ в стационарном режиме можно интерпретировать следующими способами:

а) как среднее значение той части единицы времени, в течение которой канал занят;

б) как вероятность застать канал занятым при обращении к нему в случайный момент времени;

в) как среднее число заявок в канале.

Если речь идет об экспоненциальной СМО, то коэффициент загрузки равен также вероятности того, что приходящая в СМО заявка застанет канал занятым.

В последующем речь в данной главе будет идти только о стационарных значениях характеристик.

Средняя длина очереди (среднее число заявок в очереди) в одноканальной экспоненциальной СМО рассчитывается по формуле

$$L = \frac{\rho^2}{1 - \rho}. \qquad (4)$$

График зависимости **L**(ρ) приведен на рис.20. Среднее число **N** заявок в СМО равно сумме среднего числа **L** заявок в очереди и среднего числа ρ заявок в канале :

$$M=L+R \qquad (5)$$

Действительно, как видно из рис.19, число заявок в СМО складывается из числа заявок в очереди и числа заявок в канале. И, поскольку среднее значение суммы чисел равно сумме их средних, то отсюда получается соотношение (5).

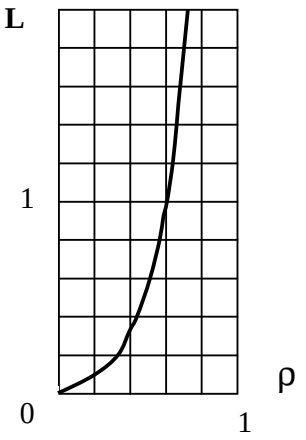


Рис.20

Зная среднее число заявок в какой-либо части системы, можно легко определить и среднее время пребывания заявок в этой части. Данные средние величины связаны знаменитой *формулой Литтла*, которая представляет собой своеобразный «закон природы» для систем массового обслуживания любого вида, не обязательно только экспоненциальных. Эта формула достаточно проста:

$$n_{ср} = \lambda_{вх}\, t_{ср}\,,$$

где $\lambda_{вх}$ – интенсивность поступления заявок в выделенную часть системы, $t_{ср}$ – среднее время пребывания заявки в этой части, $n_{ср}$ – среднее число заявок в ней.

Формула Литтла справедлива для стационарного режима. Ее смысл можно

пояснить следующим образом.

Поскольку в стационарном режиме среднее число заявок *ncp* в заданной области системы постоянно, то в среднем число заявок, поступающих в нее в единицу времени, т.е. $\lambda_{вх}$, равно числу заявок, уходящих из нее в единицу времени, т.е. $\lambda_{вых}$. При этом интенсивность ухода заявок $\lambda_{вых}$ складывается из интенсивностей ухода каждой из *ncp* заявок, т.е. $\lambda_{вых} = ncp\ (1/tcp)$.Отсюда получаем

$\lambda_{вх} = ncp\ (1/tcp)$ – т.е. формулу Литтла.

Применяя формулу Литтла к очереди в СМО, находим среднее время ожидания обслуживания через среднюю длину очереди:

$$\mathbf{W} = \mathbf{L}/\lambda. \qquad (6)$$

Аналогично вычисляется среднее время пребывания заявки во всей СМО:

$$\mathbf{U} = \mathbf{N}/\lambda. \qquad (7)$$

Если учесть, что время прохождения заявки через СМО складывается из времени ее прохождения через очередь и времени обслуживания в канале, то величину U можно вычислить и как сумму соответствующих средних времен:

$$\mathbf{U} = \mathbf{W} + \mathbf{T}.$$

Характеристики (2) - (7) могут давать ценную информацию о системе, моделируемой в виде СМО. Возможные интерпретации эих показателей при моделировании конкретных систем рассмотрены в примерах моделирования на GPSS\PC.

Заметим, что только формула (4) для средней длины очереди связана здесь с предположением о том, что СМО экспоненциальная. Остальные формулы справедливы для любых СМО, т.к. вытекают из общих свойств случайных величин, установленных теорией вероятностей.

10.2. Многоканальная экспоненциальная СМО

Многоканальная экспоненциальная СМО отличается от одноканальной следующим. Число каналов в ней более одного. Приходящая заявка становится в очередь, если все каналы заняты. В противном случае заявка занимает свободный канал.

Многоканальная экспоненциальная СМО задается тремя параметрами:

интенсивностью λ прихода заявок,

средним временем обслуживания **Т**

числом **К** каналов.

На рис.21 изображена двухканальная СМО.

Формулы для расчета характеристик многоканальной экспоненциальной СМО немногим сложнее формул (2)-(7).

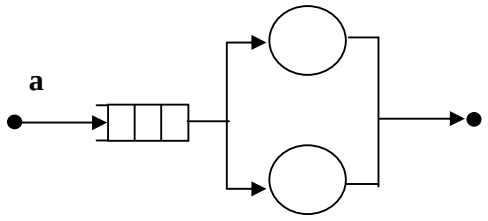


Рис.21

Коэффициент загрузки определяется в виде

$$\rho = \lambda \mathbf{T}/\mathbf{K}. \qquad (8)$$

Его значение должно отвечать условию (3), чтобы многоканальная СМО имела стационарный режим. Формула (8) справедлива и для не экспоненциальных СМО.

Средняя длина очереди в *экспоненциальной* многоканальной СМО вычисляется с помощью достаточно громоздкого выражения:

$$L = \frac{(K\rho)^{K+1}}{K!\,K\left(1-\rho\right)^2}\pi_0, \qquad (9)$$

где $\pi 0$ - стационарная вероятность того, что в СМО нет заявок. Эта вероятность определяется в виде

$$\pi_0 = \frac{1}{\frac{(\rho K)^K}{K!(1-\rho)} + \sum_{m=0}^{K-1}\frac{(\rho K)^m}{m!}}. \qquad (10)$$

При К = 1 многоканальная СМО превращается в одноканальную, вероятность $\pi 0$ в (10) становится равной 1- ρ, и формула (9) превращается в формулу (4).

Остальные характеристики многоканальной СМО вычисляются следующим образом:

$$\mathbf{M=L+KR}, \qquad (11)$$

$$\mathbf{W=L/\lambda}, \qquad (12)$$

$$\mathbf{U=W+T}. \qquad (13)$$

Основная характеристика многоканальной экспоненциальной СМО – средняя длина очереди – изображена на рис.22 в зависимости от ρ для различных значений К.

Как видно из рис.22, при фиксированном коэффициенте загрузки СМО средняя длина очереди тем меньше, чем больше число каналов в СМО.

Рис.22 показывает также, что коэффициент загрузки является наиболее важным с практической точки зрения параметром СМО. Так, изменяя число каналов в СМО в диапазоне от 1 до 5, мы не добиваемся существенного изменения средней длины очереди (а значит, и среднего времени ожидания обслуживания), если показатель ρ фиксирован. С другой стороны, небольшие изменения ρ существенно влияют на характеристики СМО, особенно, если ρ близко к единице. В реальных системах обычно необходимость моделирования возникает в тех случаях, когда показатель ρ бывает близок к единице. Это отражается, в частности, в примерах моделирования, рассмотренных в гл. 1- 6.

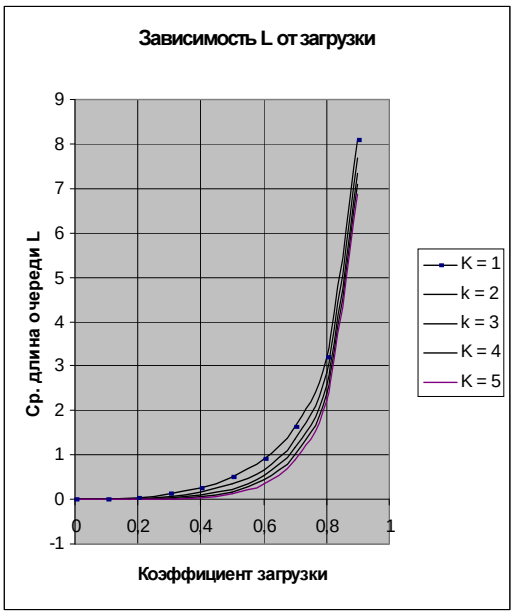


Рис.22

10.3. Разомкнутая экспоненциальная СеМО

Сетью массового обслуживания (СеМО) в общем случае называют совокупность СМО, в которой заявки с выходов одних СМО могут поступать на входы других. Входным потоком заявок в СеМО называют поток заявок, приходящих на вход отдельной СМО из внешней среды СеМО, т.е. не с выхода какой-либо СМО, входящей в данную СеМО. Число входных потоков в СеМО может достигать числа СМО.

Экспоненциальной будем называть СеМО, которая отвечает следующим требованиям:

- входные потоки в СеМО пуассоновские;
- во всех **n** СМО время обслуживания заявок имеет экспоненциальное распределение вероятностей;
- переход заявки с выхода i-й СМО на вход j-й является независимым случайным событием, имеющим вероятность **P** ij; i,j=1,**n**; При этом

выполняется условие **P** i1 + … + **P** in = 1 – **P**i0, где **P**i0 – вероятность ухода заявки, покидающей i-ю СМО, из сети.

Если заявки приходят в сеть и уходят из нее, то, сеть называется *разомкнутой.* Если заявки не приходят в сеть и из нее не уходят , то сеть называется *замкнутой.* Число заявок в замкнутой цепи постоянное.

Разомкнутая экспоненциальная СеМО задается следующими параметрами:

1) числом **n** СМО;
2) числом **К**1,...,**К**n каналов в СМО с номерами 1,...,**n**;
3) матрицей **P** = || **P**ij || вероятностей передач, i = 1,…,**n**, j = 0,…,**n**;
4) интенсивностями Λ1, … , Λn входных потоков заявок в СеМО;
5) средними временами обслуживания **T**1 ,...,**T**n заявок вСМО.

Например, СеМО, изображенная на рис 23, будет задана численно следующими параметрами:

1) числом СМО в сети: **n**=3;

2) числом каналов в каждой СМО: **К**1 =1, **К** 2 =1, К 3 =2;

3) матрицей переходных вероятностей: **P** =

	0	1	2	3
1	0,1	0	0,5	0,4
2	0	1	0	0
3	0	1	0	0

;

4) интенсивностями входных потоков заявок: Λ1 = 1, Λ2 = 0, Λ3 = 0;

5) средними временами обслуживания в каждой СМО: **T**1 = 0,07, **T** 2 = 0,06, **T** 3 = 0,35.

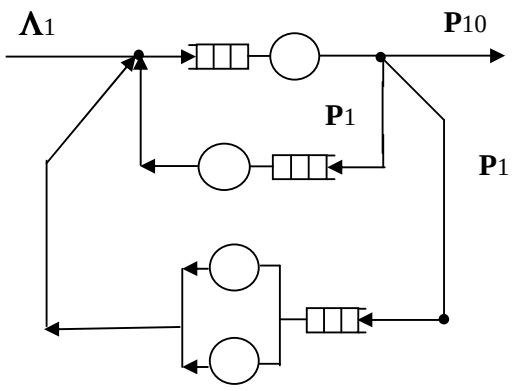


Рис.23

С помощью СеМО можно промоделировать, например, вычислительную систему в АСУ. Тогда входные потоки заявок СеМО будут изображать запросы, поступающие на вход вычислительной системы, отдельные СМО будут соответствовать этапам обработки запросов на устройствах (процессорах, периферийных устройствах и др.),

выходные заявки СеМО - результатам обработки запросов.

В СеМО поток заявок на входе любой *СМО* в общем случае складывается из *входного потока СеМО*, поступающего из внешней среды, и из потоков, поступающих с выходов отдельных СМО. Входной поток СМО в *экспоненциальной* СеМО в общем случае получается *не пуассоновским*. Это значит, что отдельные СМО в ней в общем случае не экспоненциальные. Тем не менее доказано (теорема Джексона), что СМО ведут себя в ней во многом как экспоненциальные.

В частности, характеристики отдельных СМО могут быть рассчитаны по формулам (2) – (13). Поэтому для их расчета в заданной экспоненциальной СеМО достаточно найти интенсивности λ1 ,..., λn входных потоков отдельных СМО.

10.4. Расчет локальных характеристик СеМО

Характеристики СеМО делятся на локальные и системные. К локальным характеристикам СеМО относятся характеристики всех входящих в нее СМО. Системные характеристики отражают свойства всей СеМО как единого целого.

Для расчета локальных характеристик СеМО следует найти вначале интенсивности $\lambda 1, \ldots, \lambda n$ потоков заявок, поступающих на входы отдельных СМО.

Нахождение интенсивностей $\lambda 1, \ldots, \lambda n$ осуществляется на основе *уравнений баланса потоков* в сети. При составлении уравнений баланса потоков учитываются простые свойства слияния и разветвления потоков.

Если происходит *слияние* нескольких потоков заявок, то образуется один общий поток, интенсивность которого равна сумме интенсивностей всех сливающихся потоков.

Если поток *разветвляется* на несколько направлений, то образуются потоки, сумма интенсивностей которых равна интенсивности исходного потока. При этом, если разветвление происходит путем случайного выбора заявкой одного из возможных выходных направлений, то интенсивность каждого выходного потока равна произведению интенсивности исходного потока на вероятность выбора данного направления.

В стационарной СеМО среднее число заявок в любой ее фиксированной части постоянное (свойство стационарности). Отсюда следует, что суммарная интенсивность входящих в эту часть потоков равна суммарной интенсивности выходящих. Запись данного закона в математической форме называется уравнениями баланса потоков.

Выделяя различные части в СеМО и составляя для них уравнение баланса, можно получить систему уравнений, связывающую неизвестные интенсивности $\lambda 1, \ldots, \lambda n$ с известными $\Lambda 1, \ldots, \Lambda n$. Обычно при этом в качестве отдельных частей СеМО выделяют отдельные СМО. В этом случае для n неизвестных получается n уравнений Можно добавить к ним уравнение баланса, составленное для входных и выходных потоков всей СеМО. Тогда получится n+1 уравнение, и одно из них можно использовать в качестве проверочного.

Например, баланс интенсивностей в сети (рис. 23) можно учесть, обозначая интенсивности на входах и выходах СМО и СеМО так, как показано на рис. 24. Применяя свойства слияния и разветвления потоков, запишем следующую систему уравнений:

$$\begin{cases} \lambda_1 = \Lambda_1 + \lambda_2 + \lambda_3 \\ \Lambda_1 = p_{10}\lambda_1 \\ \lambda_2 = p_{12}\lambda_1 \\ \lambda_3 = p_{13}\lambda_1 \end{cases} \qquad (14)$$

.

При известных $\Lambda 1 = 1$, p10 = 0,1, p12 = 0,5 и p13 = 0,4 из последних трех уравнений находим $\lambda 1 = 10$, $\lambda 2 = 5$, $\lambda 3 = 4$. Используя первое уравнение в (14) для проверки, подставляем в него найденные значения интенсивностей и получаем тождество 10=1+5+4, подтверждающее правильность произведенных вычислений.

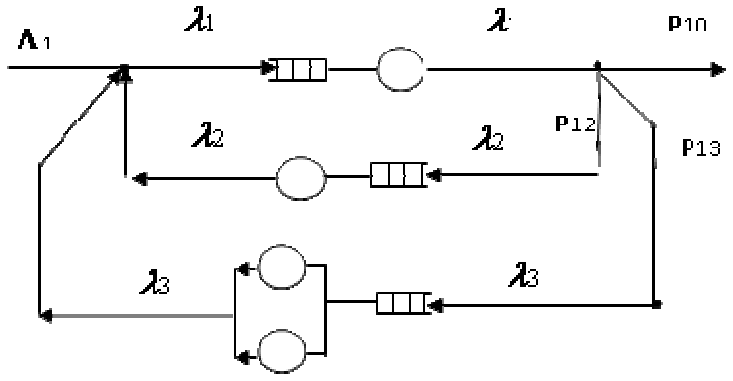


Рис. 24

Расчет входных интенсивностей завершается проверкой выполнения условий стационарности.

СеМО стационарна, если стационарны все ее СМО т.е. если

$$\rho j \le 1, \qquad j = 1, \ldots, n \qquad (15)$$

Проверить эти условия после того, как определены все λ j , не представляет труда. Например, для СеМО (рис. 23) находим:

ρ 1 = 10·0,07 = 0,7, ρ 2 =5·0,06=0,03, ρ 3 = 4·0,35/2=0,7,

т.е. условие (15) выполняется.

Если бы условия стационарности СеМО не выполнялись, то это означало бы, что расчет входных интенсивностей СМО выполнен неверно, т.к. уравнения баланса потоков справедливы только для стационарной СеМО. В этом случае следует использовать методы расчета нестационарных режимов. Заметим, что во многих случаях установление факта нестационарности СеМО приводит к завершению расчетов, т.к. свидетельствует о непригодности моделируемой системы (нехватке ресурсов для выполнения требуемых работ). В таких случаях те СМО, у которых загрузка превышает единицу, заменяют в реальной системе более мощными.

Для стационарной экспоненциальной СеМО с известными интенсивностями λj расчет локальных характеристик сводится к применению формул (2) - (13).

Например для СеМО (рис.23) находим:

ρ 1=0,7, **L**1=1,63, ; **N**1 =2,33, **W**1 =0,163, **U**1 =0,233,
ρ 2=0,3, **L**2 =0,13, **N**2 =0,43, **W**2 =0,026, **U**2 =0,086;
ρ 3 =0,7, **L**3 =0,402, **N**3 =1,802, **W**3 =0,1, **U**3 =0,45.

10.5. Расчет системных характеристик разомкнутых экспоненциальных СеМО

Если локальные характеристики СеМО относятся к ее отдельным СМО, то системные характеристики отражают свойства сети в целом, рассматриваемой как единая, неделимая на части система.

Наиболее важными системными характеристиками СеМО являются следующие:

1) *Среднее время* **U**s *пребывания* заявки в сети.

Временем пребывания заявки в сети называется время между моментом прихода заявки в сеть и моментом ее ухода из сети. Это величина случайная. Ее среднее значение и есть показатель **U**s.

2) *Входовые средние времена* **F**1 ,...,**F**n *пребывания* в сети. Величина **F**i определяется как среднее время пребывания в сети заявки, поступающей из i-го входного потока (i = 1,…,n).

3) *Абсолютные пропускные способности* **A**i .Предположим, что в заданной СеМО интенсивности всех входных потоков, кроме i-го, заменены на нулевые, а интенсивность i-го входного потока Λi заменена на предельное значение **A**i, при котором сеть еще стационарна. Это значение **A**i будем называть абсолютной пропускной способностью сети по i-му входу.

Если сделать Λi > **A**i , то сеть будет не стационарна, каковы бы ни были интенсивности остальных входных потоков. Если Λi = **A**i, то сеть стационарна лишь при условии, что остальные входные потоки (кроме i-го) не создают никакой нагрузки на ресурсы сети.

4) *Условные пропускные способности* **B**1 ,...,**B**n . Предположим, что в заданной СеМО значение интенсивности Λi заменено на максимальное значение **B**i, при котором сеть еще стационарна, при условии, что интенсивности по остальным входам в сеть оставлены неизменными (такими, как заданы при построении модели). Это значение **B**i будем называть условной пропускной способностью по входу i.

При заданных Λk (k≠i) сеть стационарна для любых значений Λi ≤ **B**i.

5) *Запасы* **D**1,..,**D**n *по пропускным способностям*. Запас **D**i = **B**i -Λi, i=1,n. Запас **D**i показывает, насколько может быть увеличена интенсивность прихода заявок на i-ом входе (при заданных остальных) без нарушения условия стационарности.

Если в виде СеМО моделируется некоторая реальная система, то характеристики 1)-5) могут дать ценную информацию о свойствах этой реальной системы. Например, если СеМО изображает информационно-вычислительную сеть, то среднее время пребывания **U**s характеризует среднее время ответа системы, а запасы **D**i выражают готовность системы продолжать устойчивое функционирование при увеличении нагрузки (интенсивности запросов) по тому или иному входу.

Среднее время Us **пребывания заявки** в СеМО рассчитывается по формуле

$$\mathbf{U}s = \mathbf{N}/\Lambda, \qquad (16)$$

где **N** = **N**1 + … +**N**n – среднее число заявок в СеМО, Λ = Λ1 + … + Λn – интенсивность поступления заявок в сеть по всем входам. Эта формула есть просто формула Литтла, примененная ко всей СеМО в целом.

Расчет системных характеристик СеМО можно выполнять после определения ее локальных характеристик. Например, в формуле (16) используются локальные характеристики **N**i.

Для СеМО, заданной на рис.23, среднее время прохождения заявки через сеть составляет

$$\mathbf{U}s = (2{,}33+0{,}43+1{,}8)/(1+0+0) = 4{,}56 \text{ с.} \qquad (16)$$

Входовое среднее время пребывания определяется с помощью уравнений, которые составляются с учетом трех простых свойств случайных величин по

топологии (графу, изображению) СеМО. Три свойства с.в. называют соответственно свойством суммы, свойством смеси и свойством суммы случайного числа слагаемых.

Свойство суммы случайных величин состоит в том, что математическое ожидание (м.о.) суммы случайных величин равно сумме математических ожиданий этих с.в. Т.е. если $\boldsymbol{y} = \boldsymbol{x}1 + \ldots + \boldsymbol{x}k$, то $M(\boldsymbol{y}) = M(\boldsymbol{x}1) + \ldots + M()$.

Свойство смеси с.в. касается м.о. смеси с.в. *Смесью* с.в. $\boldsymbol{x}1, \ldots, \boldsymbol{x}k$ называется с.в. $\boldsymbol{z}$, которая принимает значение с.в. $\boldsymbol{x}1$ с вероятностью p1, …, значение с.в. $\boldsymbol{x}k$ с вероятностью pk. При этом выбор одной из с.в. $\boldsymbol{x}i$ и ее значение статистически независимы. Смесь с.в. обладает следующим свойством:

$$M(\boldsymbol{z}) = p1 \cdot M(\boldsymbol{x}1) + \ldots + pk \cdot M(\boldsymbol{x}k).$$

Суммой τ случайного числа слагаемых называют сумму вида $\tau = \boldsymbol{x}1 + \boldsymbol{x}\Upsilon$, в которой число слагаемых Υ случайно. При этом все с.в. $\boldsymbol{x}i$ независимы и имеют одинаковые средние $M(\boldsymbol{x}1) = \ldots = M(\boldsymbol{x}k) = M(\boldsymbol{x})$. Тогда справедливо выражение

$$M(\tau) = M(\Upsilon) \cdot M(\boldsymbol{x}).$$

Рассмотрим теперь СеМО (рис.23) и проследим, как формируется входовое время пребывания в сети заявки первого потока. Из рис.23 видно, что это время состоит из двух слагаемых. Первое слагаемое есть время пребывания в СМО1, составляющее в среднем $\mathbf{U}1$. Второе слагаемое есть смесь трех с.в.: с вероятностью p10 оно равно нулю (т.к. заявка уходит из сети), с вероятностью p12 равно входовому времени пребывания для входа 2 (заявка входит в сеть через СМО2) и с вероятностью p13 - входовому времени пребывания для входа 3.

Из свойства смеси вытекает, что в среднем второе слагаемое составляет величину $p10 \cdot \mathbf{0} + p12\ \mathbf{F}2 + p13\ \mathbf{F}3 = p12\ \mathbf{F}2 + p13\ \mathbf{F}3$. В целом среднее входовое время пребывания $\mathbf{F}1$ (по свойству суммы) равно сумме средних значений первого и второго слагаемых:

$$\mathbf{F}1 = \mathbf{U}1 + p12\ \mathbf{F}2 + p13\ \mathbf{F}3 \qquad (17)$$

Рассуждая аналогично о входовых средних временах пребывания $\mathbf{F}2$ и $\mathbf{F}3$, можно записать для них сходные с (17) уравнения, которые вместе с (17) составят следующую систему уравнений:

$$\begin{cases} \mathbf{F}1 = \mathbf{U}1 + p12\ \mathbf{F}2 + p13\ \mathbf{F}3 \\ \mathbf{F}2 = \mathbf{U}2 + \mathbf{F}1 \\ \mathbf{F}3 = \mathbf{U}3 + \mathbf{F}1\ . \end{cases} \qquad (18)$$

Из этой системы при известных **U**j (они найдены при расчете локальных характеристик) нетрудно найти, что **F**1 =4,56, **F**2 =4,646, **F**3 =5,01.

По аналогии с (18) можно составить уравнения относительно **F**i для любой экспоненциальной СеМО.

Абсолютная пропускная способность по i-му входу **A**i может быть вычислена следующим образом.

В уравнениях баланса нужно вместо всех заданных входных интенсивностей Λk, k≠i, подставить значение 0, а обозначение Λi сохранить в буквенном виде, как неизвестное. После решения системы уравнений баланса все входные интенсивности СМО λk будут выражены через Λi с числовым коэффициентом (который определяется значениями переходных вероятностей). После этого коэффициенты загрузки всех СМО ρ k выражаются через буквенное обозначение с числовым коэффициентом. Максимальный из ρ k приравнивается к единице и из этого равенства находится Λi, равное пропускной способности **A**i. Аналогично можно найти абсолютные пропускные способности по другим входам в СеМО, отличным от i-го, если они представляют интерес.

Например, для СеМО, заданной на рис.23, у которой только первый входной поток имеет ненулевую интенсивность, условие стационарности сохраняется до тех пор, пока Λ1 ≤ 10/7. Отсюда **A**1 = 10/7.

Относительная пропускная способность по i-му входу СеМО может быть найдена по такому же алгоритму, как и абсолютная, с тем лишь отличием, что в системе уравнений баланса все интенсивности Λk, k≠i, сохраняются численно равными заданным в исходной модели. Интенсивность Λi также сохраняется в буквенном виде.

Для СеМО, изображенной на рис.23, относительная пропускная способность по первому входу совпадает с абсолютной, т.к. кроме первого входного потока в СеМО все остальные имеют нулевую интенсивность (отсутствуют). Следовательно, относительная пропускная способность **B**1 для этой СеМО равна 10/7.

Запасы по пропускным способностям вычисляются в соответствии с их определением. Для СеМО на рис.23 запас по пропускной способности первого входного потока составляет $\mathbf{D}_1 = 10/7 - 1 = 3/7$.

ГЛАВА 11. ПРИМЕНЕНИЕ КАЛЕНДАРЯ СОБЫТИЙ

11.1. Предварительные пояснения

В любой дискретной системе в различных ее частях могут развиваться сразу несколько различных процессов, которые имеют свою внутреннюю логику и, в то же время, взаимодействуют друг с другом. Поэтому при программировании имитационных моделей на обычных алгоритмических языках, таких, как Паскаль или Бэйсик, возникает проблема: ***как с помощью программы, выполняемой последовательно, воспроизвести логику нескольких взаимодействующих процессов, протекающих во времени параллельно?***

Суть этой проблемы состоит в том, что в любой момент выполнения данного процесса на него может воздействовать какой-нибудь другой процесс, и тогда ход данного процесса должен измениться. Но когда мы пишем программу данного процесса, то не знаем заранее, в какие моменты будут появляться воздействия на него, и каковы будут эти воздействия.

Один из способов программирования модели для таких параллельных процессов состоит в том, чтобы программа любого процесса моделировала его продвижение по времени только на достаточно малый шаг Δt, после чего передавала бы управление программе следующего процесса. Следующая программа также должна продвинуть свой процесс на время Δt и передать управление другой программе, и т.д., пока все процессы не продвинутся на один и тот же малый шаг. На каждом таком маленьком шаге каждый процесс должен проверять, не появилось ли воздействия на него со стороны других процессов, и после этого принимать решение о том, каким будет его развитие на следующем шаге. Такое продвижение процессов во времени должно продолжаться до тех пор, пока не будет достигнут конец моделируемого периода времени.

Этот подход, по существу, представляет собой не что иное, как алгоритм численного решения дифференциальных уравнений. Поэтому он с успехом используется при имитационном моделировании непрерывных систем, но совершенно непригоден для моделирования дискретных.

Действительно, все изменения в дискретных системах мгновенны, поэтому шаг Δt должен быть весьма мал, чтобы не перепутался порядок близких по времени событий и чтобы конечный интервал Δt можно было бы считать «мгновением». С другой стороны, на интервалах времени между событиями никаких изменений у переменных в системе не происходит, поэтому подавляющая часть мелких шагов по времени – т.е. все шаги между событиями – будут просчитываться “впустую” вместе со всеми необходимыми на каждом

шаге каждого процесса проверками на наличие всевозможных воздействий со стороны других процессов.

Поэтому проблема моделирования дискретных процессов решается другим способом, который основан на применении специальной структуры данных - **календаря событий**. Календарь событий позволяет весьма эффективно продвигать процессы по времени: продвижение выполняется сразу от одного события к следующему событию. При этом в программе процесса не требуется предусматривать никаких проверок на появление внешних воздействий со стороны других процессов, т.к. календарь событий учитывает все внешние воздействия автоматически.

11.2. Структура календаря событий

Всякий процесс в дискретной системе представляет собой логически связную последовательность событий. Каждое событие происходит мгновенно и проявляется как скачкообразное изменение каких-либо величин.

Например, в системе массового обслуживания (СМО) имеются такие события, как «приход заявки в систему». В момент прихода заявки либо возрастает на единицу длина очереди (если заявка попадает в очередь), либо канал СМО переходит из состояния «свободно» в состояние «занято» (если заявка занимает этот канал), либо число потерянных заявок возрастает на единицу (если заявка теряется). Таким образом, когда в какой-то момент времени происходит событие «приход заявки в систему», то в этот же момент «скачком» изменяются некоторые из переменных, характеризующих состояние системы, т.е. изменяется длина очереди, состояние канала или число потерянных заявок. Какие конкретно переменные изменятся и как они изменятся – это зависит от конкретной ситуации, которая сложилась в системе на момент появления данного события.

В общем случае число *типов событий* в конкретной дискретной системе невелико. Их все можно систематически рассмотреть и для каждого типа событий написать программу, которая с учетом конкретной ситуации будет вычислять и выполнять все необходимые изменения переменных, вызываемые любыми конкретными событиями данного типа.

Например, в СМО часто бывает достаточно выделить два типа событий – это «приход заявки» и «уход заявки».

Всякое событие кроме того, что изменяет значения переменных, может также повлечь за собой некоторое следствие – другое событие, которое должно будет произойти через некоторое время после данного события. Однако передвинуть время сразу к моменту наступления этого следствия в общем

случае мы не имеем права, т.к. в других процессах могут произойти какие-нибудь события, которые наступят раньше, чем рассматриваемое следствие данного события.

Например, если в данный момент времени произошло занятие заявкой канала, то через определенное время произойдет следствие – освобождение занятого канала. Однако мы не имеем права сразу перейти к моменту освобождения канала чтобы имитировать это освобождение канала, т.к. мы не знаем, не придет ли раньше какая-нибудь заявка на вход СМО. Если такая заявка придет раньше, то она попадет в очередь, и тогда при освобождении канала он должен быть сразу же занят заявкой из очереди.

Календарь событий используется для корректного решения именно таких ситуаций, когда известно, что в определенный момент времени в будущем произойдет следствие данного события и что мы обязаны обеспечить моделирование этого следствия, но, тем не менее, не можем перейти к моделированию следствия непосредственно из программы данного события. Выход состоит в том, что будущее событие просто записывается в календарь событий и в нем делается запись времени, когда это событие должно произойти.

TIME	EVENT

Рис.25

Календарь событий имеет следующую структуру. Он представляет собой последовательность записей, которые состоят из двух полей (рис.25). Одно поле записи, назовем его TIME, содержит момент времени, в который произойдет будущее событие. Во втором поле, которое назовем EVENT, записывается само событие. Событие в поле EVENT удобно записывать в виде имени программы, которая это событие имитирует. Если программа события имеет список формальных параметров, то вместе с именем программы могут быть записаны также фактические параметры, с которыми это событие (эта программа) должно выполняться. Поле TIME в записях календаря числовое, поле EVENT можно кодировать как символьное или как-то иначе, в зависимости от используемого языка программирования.

Условимся в дальнейшем обозначать календарь событий сокращенно KS. Эта структура данных может быть представлена в конкретной программе в виде списка, в виде массива из двух столбцов или иным образом.

Любое будущее событие в KS занимает одну строку. События записываются в KS в порядке возрастания времени. Поэтому ближайшее по времени будущее событие всегда находится в первой строке KS.

Когда будущее событие записывается в календарь, то оно вставляется в него после всех событий, у которых время меньше времени данного события. Если в календаре имеются события с точно таким же временем наступления, то данное

событие вставляется после них последним. Если у событий имеются приоритеты, которые отражают логику упорядочивания совпадающих по времени событий, то данное событие ставится впереди всех, запланированных на один с ним момент времени, но имеющих меньший приоритет, чем у данного.

Операцию записи в KS события E, планируемого на момент времени t, будем обозначать в виде (t, E) → KS. Запись события в KS производится по описанным выше правилам.

Операцию извлечения события из списка обозначим в виде KS → (t, E). Извлечение события из календаря KS означает, что управляющая программа прочитала время t первого события в календаре, и имя программы события E вместе со списком фактических параметров, которые должны быть переданы этой программе. Кроме того, первая запись в календаре после извлечения первого события удаляется, а все остальные записи «подтягиваются» вверх на одну позицию.

При использовании календаря событий правильный логический и временной порядок выполнения событий, который учитывает все причинно - следственные связи между ними, обеспечивается с помощью весьма простой управляющей программы, которая в самом начале моделирования определяет все необходимые начальные значения переменных – т.е. устанавливает модельное время равным нулю, устанавливает начальные значения параметров тех или иных объектов, включенных в модель, начальные значения переменных, используемых для сбора статистической и иной информации о ходе моделируемых процессов и т.п. Это традиционное начало выполнения любых программ называется инициализацией переменных.

После этого управляющая программа должна занести в календарь событий KS все события, являющиеся непосредственными следствиями ситуации, которая задана в качестве начального состояния системы. Следствия следствий, т.е. следствия событий, записанных в календарь, не планируются, т.к. они попадут в календарь после имитации этих уже записанных в календарь событий, т.е. после того, как эти события наступят.

Дальнейшее выполнение управляющей программы имеет циклический характер: она просто извлекает первое событие из календаря, наращивает модельное время до момента наступления этого события и передает управление программе, имитирующей событие. Вызванная программа выполняется, т.е. имитирует событие, записывает свои непосредственные следствия, если они есть, в календарь и возвращает управление управляющей программе. Последняя вновь извлекает очередное событие из календаря, передает ему управление, и так продолжается до тех пор, пока не будет достигнуто заданное условие завершения модели. После этого управление обычно передается

программе вывода накопленных данных о системе. Такой метод построения имитационных моделей полезно рассмотреть на конкретном примере, который приводится в следующем параграфе.

11.3. Пример: разработка имитационной модели СМО

Для того, чтобы составить более подробное и ясное представление о применении календаря событий, рассмотрим пример моделирования конкретной СМО.

Требуется написать имитационную модель двухканальной СМО типа R/R/2/3 (см. рис.26). Обозначение данной СМО говорит о том, что во входном потоке время между приходами заявок распределено равномерно, время обслуживания заявок каналом также распределено равномерно, что СМО имеет два канала и максимальная длина очереди равна трем.

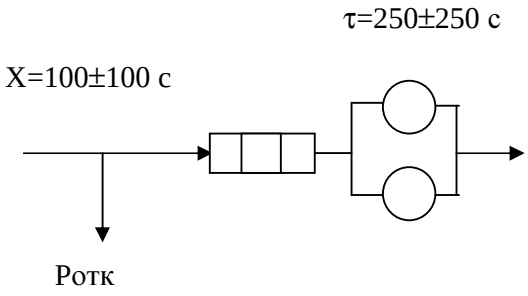


Рис.26

Когда заявка приходит в СМО, выполняется проверка на наличие свободного канала. Если первый канал свободен, заявка занимает его. Если он занят, то проверяется, свободен ли второй канал. Если второй канал свободен, то заявка занимает его. Если оба канала заняты, заявка попадает в очередь. Но если в очереди перед этим уже находятся три заявки, то пришедшая заявка покидает СМО, т.е. теряется.

Время между приходами заявок распределено равномерно в интервале от 0 до 200 с. Время обслуживания заявки каналом (любым) распределено равномерно в интервале от 0 до 500 с.

Введем обозначения переменных:

τ – время от прихода одной заявки до прихода следующей - случайная величина (с.в.);

x - время обслуживания заявки каналом – с.в.;

Ротк – вероятность отказа, т.е. того, что пришедшая заявка потеряется;

l – текущая длина очереди;

S(k) – состояние k-го канала, k = 1,2;

t – текущее время.

Выделим в моделируемой СМО два типа событий: PUT – приход заявки, GET(k) – уход заявки с k-го канала.

Календарь событий организуем в виде двух линейных массивов TIME и EVENT, так, как это описано в предыдущем параграфе.

Разработаем теперь алгоритм события типа «приход заявки», т.е. события PUT.

В соответствии с тем, что происходит в моделируемой СМО в момент прихода заявки, алгоритм должен содержать операции проверки состояния первого канала S(1) и состояния второго канала S(2) (см. рис.27).

Если есть свободный канал, то его номер записывается в переменной k, а состояние переводится из «свободно», т.е. из S(k) = 0 в «занято», т.е. S(k) = 1.

Если оба канала заняты, то проверяется длина очереди, которая после этого либо увеличивается на единицу, либо – при потере заявки – остается неизменной.

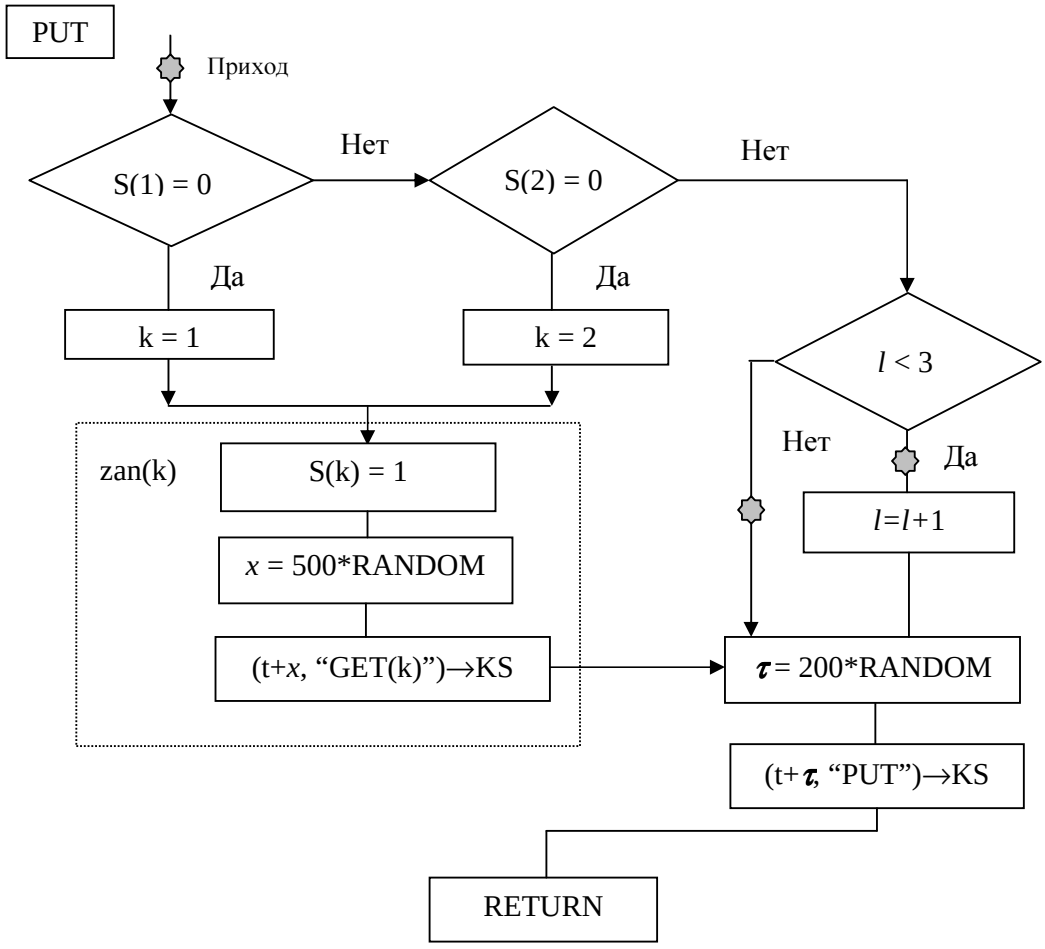


Рис.27

Звездочками на схеме обозначены точки, в которые в последующем мы будем добавлять операции для сбора статистических данных о моделируемой СМО.

Пунктирной линией обведен набор действий, выполняемых программой при имитации события занятия k–го канала. Этот набор действий обозначен zan(k). Он включает запись значения 1 в переменную S(k), что означает, что k-й канал переводится в состояние занятости, и планирование следствия из данного события, т.е. планирование освобождения канала. Т.к. канал освободится через случайное время, равномерно распределенное в интервале от 0 до 500 с, то интервал времени *x* до освобождения определяется как с.в. RANDOM (стандартная с.в., распределенная равномерно от 0 до 1), умноженная на 500. В календарь событий заносится событие освобождения только что занятого канала и время, когда это событие наступит.

В конце процедуры PUT, имитирующей приход заявки, в календарь записывается следствие данного события прихода заявки. Оно состоит в том, что через случайное время τ, равномерно распределенное в интервале от 0 до 200 с, на вход СМО поступит следующая заявка.

Алгоритм имитации события GET(k) – ухода заявки с k-го канала – приведен на рис.28. Здесь освобождаемый канал переводится в состояние

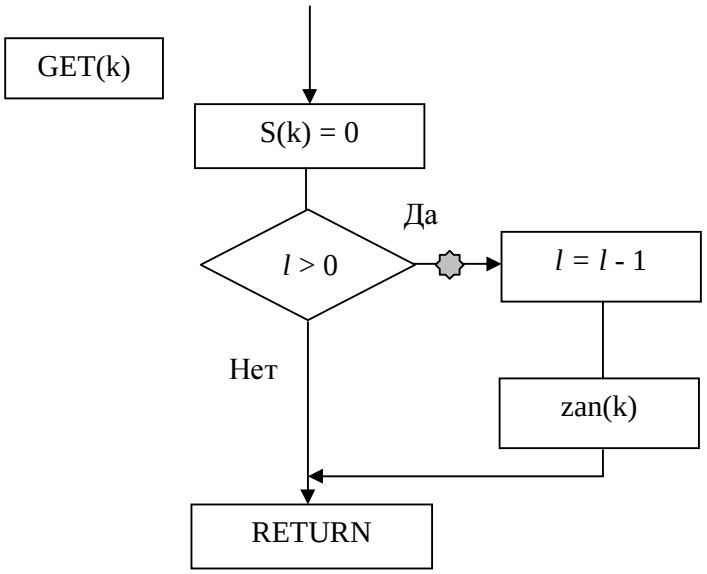


Рис.28

«свободно» и, если очередь в СМО не пустая, то ее длина уменьшается на единицу и выполняется процедура zan занятия освобожденного канала.

Схема управляющей программы приведена на рис.29.
После инициализации переменных управляющая программа заносит в

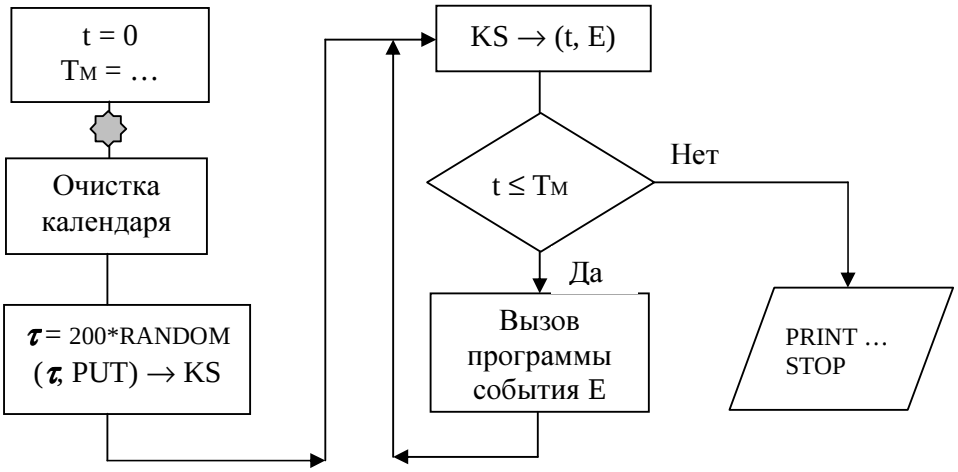


Рис.29

календарь событий KS первое событие, которое состоит в приходе заявки в СМО, и переходит к основному циклу. Основной цикл включает три операции – извлечение из KS первого события (время t при этом становится равно времени извлеченного события), проверку условия завершения модели, т.е. условия достижения времени моделирования ТМ, и вызов программы, имитирующей извлеченное событие Е.

11.4. Организация сбора статистики в модели

Поскольку все переменные изменяются лишь в отдельные моменты времени и не изменяются в промежутках между этими моментами, то сбор статистических сведений о ходе процесса можно осуществлять также лишь в те отдельные моменты, когда происходят события.

Рассмотрим этот принцип на примере расчета средней длины очереди *l*ср и вероятности отказа Ротк, т.е. потери заявки (имеются в виду стационарные значения этих характеристик).

В алгоритм управляющей программы (рис.29), в то его место, которое помечено восьмиконечной звездой, добавим следующие операции инициализации статистических переменных:

N = 0,
N1 = 0,
*l*_SUM = 0,
t_end = 0,

где N – число заявок, пришедших в систему, N1 – число заявок, потерянных из-за переполнения очереди, *l*-SUM – интеграл от длины очереди по времени, t_end – момент времени, когда произошло последнее изменение длины очереди.

Для расчета вероятности отказа Ротк добавим в алгоритм события PUT (рис.27) операции для подсчета числа пришедших заявок и числа потерянных заявок.

Число пришедших заявок будет подсчитываться с помощью операции N=N+1, которую нужно вставить вместо первой звездочки (помеченной на рис.27 словом «Приход»). Тогда каждый раз, когда будет вызвано событие PUT – приход заявки в СМО – эта пришедшая заявка подсчитается в переменной N.

Аналогично этому число потерянных заявок можно подсчитывать, поместив вместо второй звездочки (на рис.27 она расположена на дуге «Нет») операцию N1=N1+1.

Тогда вероятность отказа Ротк можно определить в конце выполнения модели как долю потерянных заявок относительно числа пришедших:

$$\text{Ротк} = N1/N.$$

Точнее говоря, мы находим не вероятность отказа, а ее статистическую оценку, точность которой зависит от длительности наблюдения за СМО, т.е. от времени моделирования ТМ. Учитывая приведенные в данном пособии рекомендации по планированию имитационных экспериментов, можно обеспечить любую точность данной оценки.

Средняя длина очереди *l*ср определяется как среднее значение функции *l*(t), т.е.

$$l\text{ср} = (1/T)\int_0^T l(t)dt.$$

Длина очереди *l*(t) изменяется во времени t скачкообразно (рис.30), как, впрочем, и любые другие переменные в модели дискретной системы. Поэтому ее интеграл по времени можно определить просто как сумму площадей прямоугольников, высоты которых равны текущей длине очереди, а ширина пределяется как длина промежутка времени, на котором очередь не изменяется.

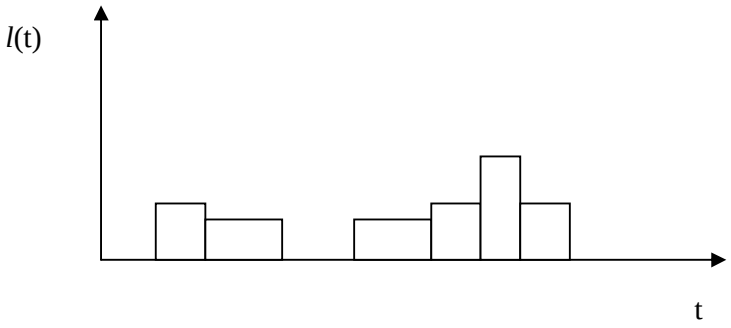


Рис.30

Как видно из сделанных замечаний, интеграл от длины очереди по времени можно рассчитать, накапливая в переменной *l*_SUM площади соответствующих прямоугольников. Для этого достаточно поместить следующую пару операторов в каждой из точек алгоритма имитации, в которых происходит изменение длины очереди:

$$l_SUM = l_SUM + l*(t - t_end)$$
$$t_end = t.$$

Таких точек в алгоритме две: одна в процедуре GET (помечена звездочкой; здесь очередь уменьшается), другая в процедуре PUT (помечена звездочкой и находится на дуге со словом «Да»; здесь очередь увеличивается). Первый из операторов добавляет к интегралу площадь очередного прямоугольника, второй сохраняет значение текущего времени в переменной t_end, чтобы можно было определить ширину очередного прямоугольника при следующем изменении очереди.

В конце моделирования можно теперь определить оценку средней длины очереди *l*ср по формуле

$$l\text{ср} = l_SUM / t.$$

Такой же простой принцип расчета средних, который рассмотрен здесь на примере средней длины очереди, может применяться и для расчете любых других средних (по времени) значений, например, для расчета коэффициентов загрузки каналов, среднего числа занятых каналов и т.п.

11.5. Практические задания

Весьма полезный опыт программирования имитационных моделей на обычных алгоритмических языках можно приобрести путем практического моделирования различных вариантов СМО. Наряду с опытом программирования при этом приобретаются также ценные умения по организации последовательного и параллельного имитационного эксперимента, которые позволяют корректно определять разнообразные вероятностно-временные характеристики стационарного и переходного режимов функционирования дискретных систем.

Таблица с вариантами СМО приводится ниже (табл.8). В этой таблице первая колонка содержит обозначение типа СМО, которое состоит из четырех позиций.

Первая позиция содержит буквенное обозначение вида распределения, которое имеет время между приходами заявок. Вторая позиция задает распределение времени обслуживания заявки каналом. При этом буква M соответствует экспоненциальному закону распределения (от слова Марковиан), буква E – эрланговскому, R – равномерному и D – детерминированному конкретному значению, т.е. константе. Цифра рядом с буквой E указывает порядок эрланговского распределения. Равномерное распределение задается на интервале от нуля до двух математических ожиданий с.в.

Третья позиция задает число каналов в СМО. Во всех вариантах число каналов равно двум. Четвертая позиция определяет максимальную длину очереди. Когда приходящая заявка застает в СМО очередь, имеющую максимальную длину, то она теряется.

Во второй колонке таблицы для каждого варианта приводится среднее значение $\boldsymbol{a}$ времени между приходами заявок, в третьей колонке – среднее время $\boldsymbol{b}$ обслуживания заявки каналом.

Таблица 8

Номер варианта	Тип СМО	a = M(x)	b = M(τ)
1	**R/R/2/3**	**10**	**25**
2	**R/R/2/0**	**10**	**15**
3	**M/M/2/10**	**10**	**35**
4	**M/E2/2/1**	**10**	**16**
5	**E2/E3/2/3**	**10**	**45**
6	**M/R/2/5**	**10**	**20**
7	**M/D/2/10**	**10**	**30**
8	**R/D/2/3**	**10**	**40**
9	**R/E3/2/1**	**10**	**17**
10	**R/M/2/5**	**10**	**28**
11	**M/M/2/0**	**10**	**18**
12	**E2/R/2/3**	**10**	**32**
13	**E3/M/2/3**	**10**	**37**
14	**E3/D/2/1**	**10**	**19**
15	**E4/R/2/10**	**10**	**22**
16	**D/R/2/3**	**10**	**24**
17	**D/E2/2/3**	**10**	**36**
18	**D/E4/2/1**	**10**	**20**
19	**D/M/2/3**	**10**	**27**
20	**D/M/2/0**	**10**	**21**

В результате моделирования выбранного варианта СМО необходимо определить и оформить в виде отчета следующие данные.

Во-первых, это должны быть стационарные значения (оценки) коэффициентов загрузки первого и второго каналов СМО, вероятности отказа (потери заявки) и средней длины очереди. Решение о том, какое время моделирования ТМ достаточно, чтобы можно было считать найденные оценки достаточно точными, следует принимать на основе предварительных (пробных) прогонов модели с различными значениями ТМ.

Во-вторых, нужно определить переходные значения перечисленных вероятностных характеристик. Для этой цели требуется осуществить параллельный эксперимент. Рекомендуется выполнить около (10 … 100) тысяч независимых прогонов модели на начальном этапе функционирования СМО. В течение одного прогона значения каждого наблюдаемого процесса, например, значения $l(t)$ (отсчеты) следует заносить в массив через один и тот же

небольшой интервал времени. Можно интервал отсчетов выбирать, например, так, чтобы за один прогон получилось 10 … 50 отсчетов процесса. Тогда массив должен содержать, соответственно, 10 … 50 элементов. Чтобы эти отсчеты выполнялись в нужные моменты времени, можно сформировать дополнительное событие, которое выполняет запись отсчетов наблюдаемых процессов и планирует само себя в календаре событий через заданный интервал времени отсчетов. Тогда такое событие будет автоматически запускаться управляющей программой модели.

За один прогон будут получены отсчеты по одной реализации каждого наблюдаемого процесса. В каждом прогоне новые значения отсчетов следует прибавлять к соответствующим элементам массива, т.е. к уже имеющимся отсчетам. Тогда в каждой фиксированной точке времени за n прогонов будет накоплена сумма n независимых отсчетов, и среднее значение процесса в этой точке можно определить путем деления полученной суммы на число прогонов n. Так, путем усреднения по прогонам, можно определить зависимость среднего значения каждого процесса от времени, т.е. переходные процессы в среднем.

Для вероятности отказа такая схема сбора статистики не подходит, т.к. событие отказа не является непрерывным во времени процессом, а может происходить только в моменты прихода заявок, которые случайны. Поэтому для определения того, как меняется вероятность отказа во времени, а точнее, как она меняется с ростом числа пришедших в СМО заявок, нужно фиксировать отсчеты по наличию отказа в момент прихода каждой заявки. Тогда номер элемента массива отсчетов будет равен порядковому номеру пришедшей заявки. В этот элемент следует добавить единицу, если заявка получила отказ, или нуль, - если заявка вошла в СМО. Очевидно, первая и вторая заявки никогда не получат отказа, т.к. СМО имеет два канала. Третья заявка в некотором числе прогонов может получить отказ, тогда за n прогонов в третьем элементе массива накопится несколько единиц, и разделив это число единиц на n, мы определим вероятность отказа для третьей заявки. Аналогично оценивается вероятность отказа для всех последующих заявок, для которых предусмотрено место в массиве отсчетов.

Все переходные процессы по средним и по вероятностям отказа рекомендуется вывести на печать в форме графиков. Наглядную статистическую информацию можно получить также путем вывода на график нескольких конкретных реализаций самих наблюдаемых случайных процессов.

Заметим, что стационарные значения, к которым приближаются рассчитанные переходные характеристики, должны быть такими же, какие получены при проведении последовательного эксперимента. Кроме того, эти значения можно легко проконтролировать путем моделирования СМО на языке GPSS.

ЛИТЕРАТУРА

1. Шрайбер Т.Дж. Моделирование на GPSS: Пер. с анг. /Пер. В.И.Гаргер, И.Л.Шмуйлович; Ред. М.А.Файнберг. – М.:Машиностроение, 1980. – 512 с.

2. Общецелевая система моделирования GPSS/360. Вводные руководящие материалы для пользователей: Пер. с англ./ Под ред. О.В. Голованова. М.: НИИТЭХИМ, 1974,183 с.

3. *Голованов О.В. и др.* Моделирование сложных дискретных систем на ЭВМ третьего поколения. Опыт применения GPSS. – М.:Энергия, 1978. – 178с.

4. Разработка системы моделирования дискретных процессов (ППП СМДП-В). Руководство программиста. Часть 1-4. АПК.00035-01 33 35. – 1981.

5. *Бахвалов Л*: «Компьютерное моделирование – длинный путь к сияющим вершинам?» / ж. «Компьютерра», 6.10.1997г., №40 (217), с.26-36.

6. *Советов Б.Я., Яковлев С.А*. Моделирование систем. Курсовое проектирование: Учеб. пособие для вузов по спец. «Автоматизированные системы обработки информации и управления». М., Высш. школа, 1980. 80 с.

7. *Задорожный В.Н.*. Статистическое моделирование: Учеб. пособие. – Омск: Изд-во ОМГТУ, 1996.– 92с. – ISBN 5-230-13865-3.

8. *Клейнен Дж.* Статистические методы в имитационном моделировании: Пер. с англ./Под ред. Ю.П.Адлера и В.Н.Варыгина. – М.:Статистика,1978.-Вып.1.-221с.; Вып.2 – 335с.

9. ЕС ЭВМ. ППП для моделирования и исследования на ЭВМ дискретных систем. Руководство программиста ПРО 309.007 Д2. 1977.-51с.

10. Общецелевая система моделирования GPSS/360. Вводные руководящие материалы для пользователей: Пер. с англ./Под ред. О.В.Голованова. М.:НИИТЭХИМ, 1974. – 183с.

11. *Шеннон Р*. Имитационное моделирование систем - искусство и наука. М.: Мир, 1978.-418с.

12. *Клейнрок Л.* Вычислительные сети с очередями. -М.: Мир, 1979.-600с.

13. *Задорожный В.Н.* Программирование имитационных моделей. Учеб. пособие. – Омск, 1987, 84с.

14. *Задорожный В.Н.* Разработка и исследование методов имитационного моделированияпроизводительности вычислительных систем: Автореф. Дис. Канд. Техн. Наук. Омск, 1983, 16с.

15. *Потапов В.И., Задорожный В.Н. Кузнецова Е.М.* Методология имитационного моделирования вычислительных систем: Учеб. пособие. Омск, 1983, 80 с.

16. *Потапов В.И., Задорожный В.Н. Кузнецова Е.М.* Программные средства моделирования вычислительных систем: Учеб. пособие. Омск, 1983, 80 с.