ФГБОУ ВО «Воронежский Государственный
Технический Университет»

Э. И. Воробьев, Б.Н. Тишуков

МОДЕЛИРОВАНИЕ ПРОЦЕССОВ И СИСТЕМ НА GPSS

Утверждено Редакционно-издательским советом университета в качестве учебно-методического пособия

Воронеж 2017

УДК 681.3

Воробьев Э.И., Тишуков Б.Н. Моделирование процессов и систем на GPSS: учебно-методическое пособие для студентов по направлению подготовки 09.03.01 «Информатика и вычислительная техника», дисциплине «Моделирование систем», по направлению подготовки 09.03.02 «Информационные системы и технологии», дисциплине «Моделирование процессов и систем» / Э.И. Воробьев, Б.Н. Тишуков. Воронеж: ФГБОУ ВО «Воронежский государственный технический университет», 2017.

В учебно-методическом пособии рассматриваются теоретические и практические сведения для изучения основных возможностей моделирования на языке GPSS, который используется для моделирования дискретных систем. Описаны основы работы с расширенным визуализированным пакетом GPSS.

Пособие соответствует требованиям Федерального государственного образовательного стандарта высшего образования по направлению подготовки 09.03.01 «Информатика и вычислительная техника», дисциплине «Моделирование систем» и по направлению подготовки 09.03.02 «Информационные системы и технологии», дисциплине «Моделирование процессов и систем».

Табл. 7. Рисунков 23. Библиогр.: 10 назв.

Рецензенты: кафедра вычислительной техники и информационных систем
Воронежского государственного
лесотехнического университета им. Г.Ф. Морозова
(зав. кафедры д-р. техн. наук, проф.
В.К. Зольников);
д-р техн. наук, проф. В.Ф. Барабанов

ВВЕДЕНИЕ

Моделирование — исследование объектов реального мира на их моделях; построение моделей реальных систем. Это мощный аппарат исследования систем, применяющийся в случаях, когда методы прямого исследования неприменимы, неэффективны или нереализуемы.

Строгая классификация методов моделирования невозможна ввиду многозначности этого понятия в науке и технике, однако существуют традиционные термины, выражающие тот или иной подход к построению модели.

Предметное моделирование основано на моделях, воспроизводящих основные геометрические, физические, химические характеристики «оригинала».

В случае, когда и модель, и объект моделирования имеют одинаковую физическую природу, речь идёт о *физическом моделировании.*

Возможно изучение системы с помощью моделей, имеющих иную природу, но описываемых одинаковыми математическими соотношениями (механические и электромагнитные колебания). Такой подход называют *предметно-математическим моделированием*.

Знаковое моделирование опирается на знаковые системы: чертежи, графики, схемы и прочее. Его разновидностью является *математическое моделирование*, базирующееся на языке математики и логики.

Любое моделирование опирается в своей основе на *мысленное моделирование*, протекающее на уровне понятий об объектах.

Модель может описывать структуру или поведение «оригинала». Во втором случае она представляет собой «чёрный ящик»: на вход подаётся воздействие, а на выходе контролируется реакция. Это наиболее характерно для *кибернетического моделирования.*

Стохастическое моделирование опирается на соотношения теории вероятностей. Используется при исследовании реакции объектов на некоторые случайные воздействия.

Моделирование, как метод познания, опирается на свойства подобия, изоморфности модели объекту по некоторым интересующим исследователя параметрам. Обычно, для корректного построения и обоснования модели используется какая-либо теория или гипотеза.

Для моделей одной физической природы с оригиналом создана теория подобия. Общая теория моделирования — теория больших систем находится ещё в процессе развития.

Сегодня имитационное моделирование – это целое научное направление. Активное внедрение имитационного моделирования в сферу решения различных задач организации и управления производством, интенсивная эксплуатация имитационного метода во всех областях инженерной деятельности, наконец, повсеместное вовлечение идей и методов машинного моделирования в процесс подготовки инженерных кадров для промышленности – все это требует широкого распространения и внедрения систем имитационного моделирования, доступных в обращении широкому кругу пользователей.

Одним из основных и наиболее распространенных инструментов построения и решения задач имитационного моделирования (ИМ) является язык моделирования GPSS. Язык GPSS представляет собой интерпретирующую языковую систему, применяющуюся для описания пространственного движения объектов. Такие динамические объекты в языке GPSS называются транзактам и представляют собой элементы потока. Функцию каждого из них можно представить как движение через модель с поочередным воздействием на ее блоки. Функциональный аппарат языка образуют блоки, описывающие логику модели, сообщая транзактам, куда двигаться и что делать дальше. Созданная GPSS-программа, работая в режиме интерпрета-

ции, генерирует и передает транзакты из блока в блок в соответствии с правилами, устанавливаемыми блоками. Каждый переход транзакта приписывается к определенному моменту системного времени. Самым современным интерпретатором данного языка является пакет GPSS World™ выпущенный компанией Minuteman Software в 2000-м году. Для большинства интерпретаторов языка GPSS созданы, так называемые, системы быстрого ввода моделей, так для GPSS/H существует пакет Prof Animation для GPSS/PC пакеты GPSS-IDE и Analiz. Своим рождением GPSS (General Purpose Simulation System — система моделирования общего назначения) обязан Гордону, предложившему язык блок-диаграмм. Разработан GPSS компанией IBM в начале 60-х годов.

При моделировании необходимо учитывать тот факт, что, возможно, построить несколько, в том числе и абсолютно противоречивых, моделей соответствующих известным экспериментальным фактам об объекте. Также необходимо учитывать то, что модель является идеализированной копией объекта, отражающей лишь основные характеристики.

Предназначение GPSS — реализация стохастических моделей дискретных процессов. Иными словами, моделирование дискретных систем (дискретной система называется в случае, когда её реакцию на воздействие можно представить в виде набора конечных состояний), подверженных случайным воздействиям.

1 Общие сведения о GPSS

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

номер _строки имя операция операнды ; комментарии

Все операторы исходной программы должны начинаться с номера 0_строки - целого положительного числа от 1 до 9999999. После ввода операторов они располагаются в исходной программе в соответствии с нумерацией строк. Обычно нумерация производится с некоторым шагом, отличным от 1, чтобы иметь возможность добавления операторов в нужное место исходной программы. Некоторые операторы удобно вводить, не включая их в исходную программу. Такие операторы вводятся без номера строки.

В настоящем издании при описании формата операторов и в примерах моделей номера строк будут опускаться для лучшей читаемости текста. Отдельные операторы могут иметь имя для ссылки на эти операторы в других операторах. Если такие ссылки отсутствуют, то этот элемент оператора не является обязательным.

В поле операции записывается ключевое слово (название оператора), указывающее конкретную функцию, выполняемую данным оператором. Это поле оператора является обязательным. У некоторых операторов поле операции включает в себя также вспомогательный операнд. В полях операндов записывается информация, уточняющая и конкретизирующая выполнение функции, определенной в поле операции. Эти поля в зависимости от типа операции содержат до семи операндов, расположенных в определенной последовательности и обозначаемых обычно первыми буквами латинского алфавита

от A до G. Некоторые операторы вообще не имеют операндов, а в некоторых операнды могут быть опущены, при этом устанавливаются их стандартные значения (по умолчанию). При записи операндов используется позиционный принцип: пропуск операнда отмечается запятой. Необязательные комментарии в случае их присутствия отделяются от поля операндов точкой с запятой. Комментарии не могут содержать букв русского алфавита.

Операторы GPSS записываются, начиная с первой позиции, в свободном формате, т.е. отдельные поля разделяются произвольным количеством пробелов. При вводе исходной программы в интегрированной среде GPSS размещение отдельных полей операторов с определенным количеством интервалов между ними производится автоматически. Каждый оператор GPSS относится к одному из четырех типов: операторы-блоки, операторы определения объектов, управляющие операторы и операторы-команды.

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

Операторы определения объектов служат для описания параметров некоторых объектов GPSS. Примерами параметров объектов могут быть количество каналов в многоканальной системе массового обслуживания, количество строк и столбцов матрицы и т.п.

Управляющие операторы служат для управления процессом моделирования (прогоном модели). Операторы-команды позволяют управлять работой интегрированной среды GPSS. Управляющие операторы и операторы-команды обычно не включаются в исходную программу, а вводятся непосредственно с клавиатуры ПК в процессе интерактивного взаимо-

действия с интегрированной средой. После трансляции исходной программы в памяти ПК создается, так называемая, текущая модель, являющаяся совокупностью разного типа объектов, каждый из которых представляет собой некоторый набор чисел в памяти ПК, описывающих свойства и текущее состояние объекта.

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

Динамические объекты, соответствующие заявкам в системах массового обслуживания, называются в GPSS *транзактами*. Они "создаются" и "уничтожаются" так, как это необходимо по логике модели в процессе моделирования. Сообщения движутся от блока к блоку так, как движутся элементы, которые они представляют (программы в примере с ЭВМ). Каждое продвижение считается событием, которое должно происходить в конкретный момент времени. Интерпретатор GPSS автоматически определяет моменты наступления событий. В тех случаях, когда событие не может произойти, хотя момент его наступления подошел (например, при попытке занять устройство, когда оно уже занято), сообщение прекращает продвижение до снятия блокирующего условия. С каждым транзактом может быть связано произвольное число параметров, несущих в себе необходимую информацию об этом транзакте. Кроме того, транзакты могут иметь различные приоритеты. Сообщения нумеруются последовательно, начиная с номера 1. Параметры сообщений принимают значения из множества целых чисел. Каждое сообщение имеет один или более параметров.

Операционные объекты GPSS, называемые *блоками*, соответствуют операторам-блокам исходной программы. Они, формируют логику модели, давая транзактам указания: куда идти и что делать дальше. Модель системы на GPSS можно представить совокупностью блоков, объединенных в соответствии с

логикой работы реальной системы в так называемую блок-схему. Блок-схема модели может быть изображена графически, наглядно показывая взаимодействие блоков в процессе моделирования. Ниже перечислены некоторые свойства этих блоков. В блоках могут происходить события четырех основных типов:

1) создание или уничтожение сообщений;

2) изменение числового атрибута объекта;

3) задержка сообщения на определенный период времени;

4) изменение маршрута сообщения в модели.

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

Одноканальные и многоканальные устройства соответствуют обслуживающим приборам в СМО. Одноканальное устройство, которое для краткости далее будем называть просто устройством, может обслуживать одновременно только один транзакт. Интерпретатор записывает информацию о том, какое сообщение в настоящий момент занимает устройство. Если другое сообщение попытается захватить устройство, то это сообщение задерживается до тех пор, пока устройство не освободится. Программа также автоматически подсчитывает общее время занятости устройства. Это значение позволяет определить коэффициент использования каждого устройства. Подсчитывается также общее число сообщений, занимавших устройство, что позволяет вычислить среднее время занятости устройства одним сообщением. В реальных системах объекты типа "устройство" могут иногда прерывать обслуживание одних элементов и начинать обслуживание других.

Многоканальное устройство (МКУ) может обслуживать одновременно несколько транзактов.

Многоканальные устройства используются для представления физического оборудования, например, зрительного зала театра, стоянки автомобилей и, в некоторых случаях, основной памяти в системах для обработки данных. Пользователь определяет емкость каждого многоканального устройства, используемого в модели, а интерпретатор ведет учет числа единиц многоканальных устройств, занятых в каждый момент времени. Если сообщение пытается занять больше единиц многоканального устройства, чем свободно в данный момент, обработка этого сообщения задерживается до того момента, пока в многоканальном устройстве освободится достаточный объем. Программа автоматически ведет подсчет числа сообщений, входящих в многоканальное устройство. Определяется также среднее число единиц многоканальных устройств, занятых одним сообщением, и среднее время пребывания сообщения в многоканальном устройстве. Эти статистические данные выдаются в конце счета и позволяют определить, насколько эффективно используются в системе объекты параллельной обработки и достаточна ли их емкость.

Логические переключатели (ЛП) используются для моделирования двоичных состояний логического или физического характера. ЛП может находиться в двух состояниях: включено и выключено. Состояние ключа может быть изменено любым другим сообщением, и любое сообщение может использовать состояние ключа для выбора одного из двух возможных путей или ожидать момента изменения состояния ключа.

Статистические объекты GPSS служат для сбора и обработки статистических данных о функционировании модели. К ним относятся *очереди* и *таблицы*. Каждая очередь обеспечивает сбор и обработку данных о транзактах, задержанных в какой-либо точке модели, например перед одноканальным устройством. Пользователь может определить специальные точки в модели, в которых нужно собирать

статистику об очередях. Тогда интерпретатор GPSS автоматически будет собирать статистику об очередях (длину очереди, среднее время пребывания в очереди и т.д.). Число задержанных сообщений и продолжительность этих задержек определяется только в этих заданных точках. Интерпретатор также автоматически подсчитывает в этих точках общее число сообщений, поступающих в очередь. Это делается примерно также, как и для устройств и памятей. В определенных счетчиках подсчитывается число сообщений, задерживающихся в каждой очереди, так как может представлять интерес число сообщений, прошедших какую-либо точку модели без задержки. Интерпретатор подсчитывает среднее время пребывания сообщения в очереди (для каждой очереди), а также максимальное число сообщений в очереди. Таблицы используются для получения выборочных распределений некоторых случайных величин, например, времени пребывания транзакта в модели. Таблица состоит из частотных классов, куда заносится число попаданий конкретного числового атрибута в каждый конкретный частотный класс. Для каждой таблицы вычисляется также математическое ожидание и среднеквадратичное отклонение. Эта статистика является стандартной для всех таблиц. При задании таблиц могут понадобиться различные режимы работы таблиц. Режим работы зависит от того, какая статистика нужна.

К вычислительным объектам GPSS относятся переменные (арифметические и булевские) и функции. Они используются для вычисления некоторых величин, заданных арифметическими или логическими выражениями либо табличными зависимостями. *Арифметические переменные* позволяют вычислять арифметические выражения, состоящие из стандартных числовых атрибутов (СЧА). Запись выражения переменной аналогична записи соответствующих операторов языка Фортран. В выражении переменной используют операторы, арифметические действия и вызовы библиотечных функций. *Булевские перемен-*

ные позволяют пользователю проверять в одном блоке GPSS одновременно несколько условий, исходя из состояния или значения объектов и их атрибутов. То есть, в данном блоке производится обращение к булевской переменной, выражение которой содержит в себе проверку нескольких условий. Булевская переменная имеет значение 1, если выражение переменной истинно, и 0, если выражение переменной ложно. Используя *функции*, пользователь может производить вычисления непрерывных или дискретных функциональных зависимостей между аргументом функции (независимая величина) и зависимым значением функции. Функции широко применяются, например, для задания случайного интервала времени между генерацией двух сообщений. Все функции в GPSS задаются табличным способом с помощью специальных операторов описания функции.

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

К объектам группирующего класса относятся списки пользователя и группы. Списки пользователя используются для организации очередей с дисциплинами, отличными от дисциплины "раньше пришел – раньше обслужен". Каждому объекту того или иного класса соответствуют числовые атрибуты, описывающие его состояние в данный момент модельного времени. Кроме того, имеется ряд так называемых системных атрибутов, относящихся не к отдельным объектам, а к модели в целом. Значения атрибутов всех объектов модели по окончании моделирования выводятся в стандартный отчет GPSS. Большая часть атрибутов доступна программисту и составляет, так называемые, стандартные числовые атрибуты (СЧА), которые могут использоваться в качестве операндов операторов исходной программы. Все СЧА в GPSS являются целыми числами. К системным числовым атрибутам относятся следующие величины:

RNj – число, вычисляемое j датчиком случайных чисел (где 1<=j<=7). Все датчики генерируют последовательность равномерно распределенных случайных чисел. Это целое число изменяется от 0 до 999 включительно, кроме двух случаев его использования - в качестве аргумента функции или элемента в переменной. В этих случаях RNj будет дробью от 0 до 0.999999;

C1 – текущее значение условного времени. Автоматически изменяется программой и устанавливается в 0 управляющими операторами CLEAR или RESET;

AC1 – текущее значение абсолютного времени. Автоматически изменяется программой. Эта величина не меняется под действием управляющего оператора RESET и устанавливается в 0 лишь под действием оператора CLEAR;

TG1 – число, равное текущему значению счетчика завершений. Сообщения, вошедшие в блоки TERMINATE с ненулевым операндом A, уменьшают значение этого счетчика на число, равное значению операнда A;

XN1 – возвращает номер активного сообщения;

Z1 – возвращает размер свободной оперативной памяти в байтах;

M1 – время пребывания в модели сообщения, обрабатываемого программой в данный момент. Эта величина может изменяться блоком MARK. Время пребывания вычисляется следующим образом: M1 равно разнице текущего значения абсолютного времени и отметки времени обрабатываемого сообщения;

PR – приоритет обрабатываемого в данный момент сообщения. Эта величина может изменяться блоком PRIORITY. По умолчанию приоритет равен 0.

Сообщения могут иметь СЧА:

Pj или ***j**, или ***<имя>**, или ***$<имя>** - значение параметра j текущего сообщения или значение параметра с именем <имя> текущего сообщения ;

MPj – значение времени, равное разности абсолютного модельного времени и содержимого j-го параметра текущего сообщения;

MBj – флаг синхронизации: 1, если сообщение в блоке j принадлежит тому же семейству , что и текущее сообщение; 0 - в противном случае.

Блоки имеют СЧА:

Nj – общее число сообщений, которое должно войти в j-й блок. Подсчет ведется программой автоматически. Например, N$MET1 - счетчик числа входов в блок MET1. Этот счетчик изменяется при каждом входе сообщения в блок MET1;

Wj – текущее число сообщений, которое находится в блоке j. Значение этого счетчика подсчитывается автоматически. Например, W$MET2 - счетчик текущего числа сообщений в блоке MET2.

Многоканальные устройства имеют следующие СЧА:

Sj – текущее содержимое многоканального устройства j. Содержимое многоканального устройства может изменяться блоками ENTER и LEAVE. Например, S$OPER - текущее содержимое многоканального устройства OPER;

Rj – число свободных единиц многоканального устройства j. Эта величина может изменяться блоками ENTER и LEAVE. Например, R$MACH - свободный объем многоканального устройства MACH;

SRj – коэффициент использования многоканального устройства j в тысячных долях, т.е. если коэффициент равен 0.65, то SRj равно 650;

SAj – среднее содержимое многоканального устройства j (целая часть);

SMj – максимальное содержимое многоканального устройства j;

SCj – общее число входов в многоканальное устройство j;

STj – среднее время пребывания сообщений в многоканальном устройстве j.

SEj – флаг незанятости многоканального устройства j: 1 - свободно, 0 - занято;

SFj – флаг заполненности многоканального устройства j: 1 - заполнено, 0 - не заполнено;

SVj – флаг готовности многоканального устройства j: 1 - готово, 0 - не готово;

Устройства имеют следующие СЧА:

Fj – текущее состояние устройства j. Эта величина равна 0, если устройство свободно, и 1 - во всех остальных случаях. Этот атрибут изменяется блоками SEIZE, RELEASE, PREEMPT и RETURN. Например, F$ACPU - состояние устройства ACPU;

FIj – флаг прерывания устройства:1, если устройство находится в состоянии прерывания, 0 - в противном случае;

FVj – флаг готовности устройства к использованию:1, если готово, 0 - в противном случае;

FRj – коэффициент использования устройства j в тысячных долях, т.е., если коэффициент равен 0.88, то FRj равен 880;

FCj – общее число входов в устройство j;

FTj – среднее время использования устройства одним сообщением.

Очереди имеют следующие СЧА:

Qj – длина соответствующей очереди j. Эта величина может изменяться блоками QUEUE и DEPART. Например, Q2 соответствует очереди 2;

QMj – максимальная длина очереди j. Это значение автоматически определяется и сохраняется программой;

QCj – общее число входов в очередь j. Это значение автоматически определяется и сохраняется программой;

QZj – число нулевых входов в очередь j. Это значение подсчитывается автоматически;

QTj – среднее время пребывания сообщения в очереди j (включая нулевые входы);

QXj – среднее время пребывания сообщения в очереди j (без нулевых входов).

Таблицы имеют следующие СЧА:

TBj – вычисленное среднее таблицы j. Для занесения в таблицу используется блок TABULATE;

TCj – общее число включений в таблицу j;

TDj – вычисленное среднеквадратичное отклонение для таблицы.

Ячейки и матрицы ячеек сохраняемых величин имеют следующие СЧА:

Xj – содержимое ячейки j;

MXj(a,b) – содержимое элемента матрицы ячеек j, расположенного в строке a, столбце b;

Вычислительные объекты имеют следующие СЧА:

FNj – вычисленное значение функции j. От значения функции берется целая часть, за исключением тех случаев, когда это значение используется в качестве модификатора в блоках GENERATE, ADVANCE или ASSIGN или в качестве аргумента другой функции;

Vj – вычисленное значение переменной j. При вычислении значения переменной с фиксированной точкой получается целое число. При вычислении значения переменной с плавающей точкой дробная часть конечного результата отбрасывается;

BVj – вычисленное значение (1 или 0) булевской переменной.

Списки и группы имеют следующие СЧА:

GNj – текущее число членов в числовой группе j;

GTj – текущее число членов в группе сообщений с номерами j;

CHj – текущее число сообщений в j-том списке пользователя;

CAj – среднее число сообщений в j-том списке пользователя;

CMj – максимальное число сообщений в j-том списке пользователя;

CCj – общее число сообщений в j-том списке пользователя;

CTj – среднее время пребывания сообщения в j-том списке пользователя;

LSj – возвращает состояние логического ключа j:1 - становлен, 0 - не установлен.

Стандартные числовые атрибуты могут использоваться в качестве операндов практически в любом типе блоков.

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

имя EQU номер

Блокам присваиваются их порядковые номера в исходной программе (не путать с номерами строк!). Для ссылки на какой-либо стандартный числовой атрибут некоторого объекта соответствующий операнд оператора исходной программы записывается одним из следующих способов:

СЧА $имя;

СЧА j,

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

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

В начальный момент времени в GPSS-модели нет ни одного транзакта. В процессе моделирования симулятор генерирует транзакты в определенные моменты времени в соответствии с теми логическими потребностями, которые возникают в моделируемой системе. Подобным же образом транзакты покидают модель в определенные моменты времени в зависимости от специфики моделируемой системы. В общем случае в модели одновременно существует большое число транзактов, однако в каждый момент времени симулятор осуществляет продвижение только какого-либо одного транзакта. Если транзакт начал свое движение, он перемещается от блока к блоку по пути, предписанному блок-схемой. В тот момент, когда транзакт входит в некоторый блок, на исполнение вызывается подпрограмма симулятора, соответствующая типу этого блока, а после ее выполнения, при котором реализуется функция данного блока, транзакт "пытается" войти в следующий блок. Такое продвижение транзакта продолжается до тех пор, пока не произойдет одно из следующих возможных событий:

1) транзакт входит в блок, функцией которого является удаление транзакта из модели;

2) транзакт входит в блок, функцией которого является задержка транзакта на некоторое определенное в модели время;

3) транзакт "пытается" войти в следующий блок, однако блок "отказывается" принять его. В этом случае транзакт оста-

ется в том блоке, где находился, и позднее будет повторять свою попытку войти в следующий блок. Когда условия в модели изменятся, такая попытка может оказаться успешной, и транзакт сможет продолжить свое перемещение по блок-схеме.

Если возникло одно из описанных выше условий, обработка данного транзакта прекращается, и начинается перемещение другого транзакта. Таким образом, выполнение моделирования симулятором продолжается постоянно.

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

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

Таймер GPSS/PC имеет следующие особенности:

1) регистрируются только целые значения (все временные интервалы в модели изображаются целыми числами);

2) единица модельного времени определяется разработчиком модели, который задает все временные интервалы в одних и тех же, выбранных им единицах;

3) симулятор не анализирует состояние модели в каждый следующий момент модельного времени (отстоящий от текущего на единицу модельного времени), а продвигает таймер к моменту времени, когда происходит ближайшее следующее событие.

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

Центральной задачей, выполняемой симулятором, является определение того, какой транзакт надо выбрать следующим для продвижения в модели, когда его предшественник прекра-

тил свое продвижение. С этой целью симулятор рассматривает каждый транзакт как элемент некоторого списка. В относительно простых моделях используются лишь два основных списка: список текущих событий и список будущих событий.

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

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

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

2 Основные блоки GPSS/PC и связанные с ними объекты

2.1 Блоки, связанные с транзактами

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

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

имя GENERATE A,B,C,D,E

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

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

Например,

GENERATE 100,40

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

Модификатор-функция используется, если закон распределения интервала поступления отличен от равномерного. В

этом случае в **поле B** должна быть записана ссылка на функцию (ее СЧА), описывающую этот закон, и случайный интервал поступления определяется, как целая часть произведения **поля A** (среднего значения) на вычисленное значение функции.

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

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

В поле **E** задается приоритет, присваиваемый генерируемым транзактам. Число уровней приоритетов неограниченно, причем самый низкий приоритет - нулевой. Если поле E пусто, то генерируемые транзакты имеют нулевой приоритет. Важными стандартными числовыми атрибутами транзактов являются значения их параметров. Любой транзакт может иметь неограниченное число параметров, содержащих те или иные числовые значения. Ссылка на этот СЧА транзактов всегда относится к активному транзакту и имеет вид Pj или P$ имя, где j и имя - номер и имя параметра соответственно. Такая ссылка возможна только в том случае, если параметр с указанным номером или именем существует, т.е. в него занесено какое-либо значение.

Когда сообщение покидает блок **GENERATE**, счетчик общего числа прошедших через блок сообщений (Nj) увеличивается на единицу. Так как последующее сообщение не генерируется до тех пор, пока предыдущее сообщение не покидает блок **GENERATE**, то содержимое счетчика текущего числа находящихся в блоке сообщений (Wj) никогда не превышает 1. При повторном описании блока **GENERATE** при помощи нового оператора описания блока интерпретатор GPSS просматривает все находящиеся в данный момент модели сообщения и проверяет, есть ли среди них сообщения, связанные с повторно описываемым блоком **GENERATE**

(таких сообщений может и не быть, если данный блок уже создал заданное число сообщений). Эти сообщения, если они есть, уничтожаются. Операнды нового блока **GENERATE** заменяют операнды предыдущего блока **GENERATE**, и затем создается новое сообщение, используя спецификации нового блока.

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

имя ASSIGN A,B,C

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

Например, блок

ASSIGN 5,0

записывает в параметр с номером 5 значение 0, а блок

ASSIGN COUNT+,1

добавляет 1 к текущему значению параметра с именем COUNT.

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

имя MARK A

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

Например, если на входе участка модели поместить блок

MARK MARKER,

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

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

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

имя PRIORITY A,B

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

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

имя TERMINATE A

Значение поля **A** указывает, насколько единиц уменьшается содержимое так называемого счетчика

завершений при входе транзакта в данный блок **TERMINATE**. Если поле **A** не определено, то оно считается равным 0, и транзакты, проходящие через такой блок, не уменьшают содержимого счетчика завершений. Каждый раз, когда сообщение входит в блок **TERMINATE,** общее число сообщений, вошедших в блок **TERMINATE** (Nj), увеличивается на единицу. Число сообщений, находящихся в данный момент времени в блоке **TERMINATE**, всегда равно нулю, т.е. Wj = 0.

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

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

Участок блок-схемы модели, связанный с парой блоков **GENERATE -TERMINATE,** называется *сегментом*. Простые модели могут состоять из одного сегмента, в сложных моделях может быть несколько сегментов. Например, простейший сегмент модели, состоящий всего из двух блоков **GENERATE** и **TERMINATE** в совокупности с управляющим оператором **START,** моделирует процесс создания случайного потока транзактов, поступающих в модель со средним интервалом в 100 единиц модельного времени, и уничтожения этих транзактов. Начальное значение счетчика завершений равно 1000. Каждый транзакт, проходящий через **блок TERMINATE**, вычитает из счетчика единицу, и, таким образом, моделирование завершится, когда тысячный по счету транзакт войдет в **блок TERMINATE**. При этом точное значение таймера в момент завершения прогона непредсказуемо. Следовательно, в приведен-

ном ниже примере продолжительность прогона устанавливается не по модельному времени, а по количеству транзактов, прошедших через модель.

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

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

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

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

В приведенных примерах транзакты, входящие в модель через блок **GENERATE**, в тот же момент модельного времени уничтожались в блоке **TERMINATE**. В моделях систем массо-

вого обслуживания заявки обслуживаются приборами (каналами) СМО в течение некоторого промежутка времени прежде, чем покинуть СМО. Для моделирования такого обслуживания, т.е. для задержки транзактов на определенный отрезок модельного времени, служит **блок ADVANCE** (задержать), имеющий следующий формат:

имя ADVANCE A,B

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

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

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

В рассмотренных выше примерах случайные интервалы времени подчинялись равномерному закону распределения вероятностей. Для получения случайных величин с другими распределениями в GPSS/PC используются вычислительные объекты: переменные и функции. Как известно, произвольная случайная величина связана со случайной величиной **R**, имеющей равномерное распределение на отрезке [0;1], через свою обратную функцию распределения. Для некоторых случайных величин уравнение связи имеет явное решение, и значение случайной величины с заданным распределением вероятностей может быть вычислено через **R** по формуле. Так, например, значение случайной величины **E** с показательным (экспоненциальным) распределением с параметром **d** вычисляется по формуле:

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

Напомним, что параметр d имеет смысл величины, обратной математическому ожиданию E, а, следовательно, 1/d - математическое ожидание (среднее значение) случайной величины E. Для получения случайной величины R с равномерным распределением на отрезке [0;1] в GPSS/PC имеются встроенные генераторы случайных чисел. Для получения случайного числа путем обращения к такому генератору достаточно записать системный СЧА RN с номером генератора, например RN1. Правда, встроенные генераторы случайных чисел GPSS/PC дают числа не на отрезке [0;1], а целые случайные числа, равномерно распределенные от 0 до 999, но их нетрудно привести к указанному отрезку делением на 1000.

Проще всего описанные вычисления в GPSS/PC выполняются с использованием арифметических переменных. Они могут быть целыми и действительными. Целые переменные определяются перед началом моделирования с помощью опе-

ратора определения **VARIABLE** (переменная), имеющего следующий формат:

имя VARIABLE выражение

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

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

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

Действительные переменные могут быть использованы для получения случайных интервалов времени с показательным законом распределения. Пусть в модели из рассмотренного ранее примера распределения времени поступления транзактов и

времени задержки должны иметь показательный закон. Это может быть сделано так:

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

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

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

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

```
имя   FUNCTION   A,B
```

Здесь **имя** - имя функции, используемое для ссылок на нее; **A** – стандартный числовой атрибут, являющийся аргумен-

том функции; **B** – тип функции и число точек таблицы, определяющей функцию.

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

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

```
EXP    FUNCTION    RN1,C24
```

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

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

Как уже говорилось, при использовании функции в поле **B** блоков **GENERATE** и **ADVANCE** вычисление интервала поступления или времени задержки производится путем умножения операнда A на вычисленное значение функции. Отсюда следует, что функция, используемая для генерирования случайных чисел с показательным распределением, должна описывать зависимость **y=-ln(x)**, представленную в табличном виде.

Оператор **FUNCTION** с такой таблицей, содержащей 24 точки для обеспечения достаточной точности аппроксимации, имеет следующий вид:

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

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

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

Функции всех типов имеют единственный **СЧА** с названием **FN**, значением которого является вычисленное значение функции. Вычисление функции производится при входе тран-

закта в блок, содержащий ссылку на СЧА FN с именем функции. Заменим в рассмотренном выше примере переменные TARR и TSRV на функцию EXP(результат показан ниже).

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

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

Особенностью непрерывных функций является то, что они принимают "непрерывные" (но только целочисленные) значения в диапазоне от Y1 до Yn , где n - количество точек таблицы. В отличие от них дискретные числовые функции, тип которых кодируется буквой **D** в операнде **B** оператора определения функции, принимают только отдельные (дискретные) значения, заданные координатами Yi в строках, следующих за оператором определения **FUNCTION**. При вычислении дискретной функции текущее значение СЧА-аргумента, указанного в поле A оператора **FUNCTION**, сравнивается по условию <= последовательно со всеми значениями упорядоченных по возрастанию координат Xi до выполнения этого условия при некотором i. Значением функции становится целая часть соответствующего значения Yi.

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

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

В блоке **ASSIGN** в параметр **TYPE** каждого сгенерированного транзакта заносится тип заявки, получаемый с помощью дискретной функции **CLASS**. Аргументом функции является генератор случайных чисел **RN1**, а координаты ее таблицы представляют собой обратную функцию распределения дискретной случайной величины "класс заявки" с одинаковыми вероятностями каждого из трех значений случайной величины. Поле **A** блока **ADVANCE** содержит ссылку на списковую функцию **MEAN**, аргументом которой служит параметр **TYPE** входящих в блок транзактов. В зависимости от значений этого параметра (типа заявки) среднее время задержки принимает одно из трех возможных значений функции **MEAN**: 70, 80 или 90 единиц.

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

```
GENERATE   100,FN$EXP
ASSIGN     TYPE,FN$CLASS
ADVANCE    FN$MEAN,FN$EXP
TERMINATE  1
```

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

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

имя SPLIT A,B,C

В поле **A** задается число создаваемых копий исходного транзакта (родителя), входящего в блок **SPLIT**. После выхода из блока **SPLIT** транзакт-родитель направляется в следующий блок, а все транзакты-потомки поступают в блок, указанный в поле **B**. Если поле **B** пусто, то все копии поступают в следующий блок. Транзакт-родитель и его потомки, выходящие из блока **SPLIT**, могут быть пронумерованы в параметре, имя или номер которого указаны в поле **C**. Если у транзакта-родителя значение этого параметра при входе в блок **SPLIT** было равно k, то при выходе из блока оно станет равным k+1, а значения этого параметра у транзактов-потомков окажутся равными k+2, k+3 и т.д. Например, блок

```
SPLIT   5,MET1,NUM
```

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

2.2 Блоки, связанные с аппаратными объектами

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

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

имя SEIZE A

имя RELEASE A

В поле **A** указывается номер или имя устройства. Если транзакт входит в блок **SEIZE**, то устройство, указанное в поле **A**, становится занятым и остаётся в этом состоянии до тех пор, пока этот же транзакт не пройдёт соответствующий блок **RELEASE**, освобождая устройство. Если устройство, указанное

в поле **A** блока **SEIZE**, уже занято каким-либо транзактом, то никакой другой транзакт не может войти в этот блок и остаётся в предыдущем блоке. Транзакты, задержанные (заблокированные) перед блоком **SEIZE**, остаются в списке текущих событий и при освобождении устройства обрабатываются с учетом приоритетов и очередности поступления. Каждое устройство имеет следующие СЧА:

F - состояние устройства (0 - свободно,1 - занято);

FR - коэффициент использования в долях 1000;

FC - число занятий устройства;

FT - целая часть среднего времени занятия устройства.

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

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

Для моделирования захвата (прерывания) одноканального устройства вместо блоков **SEIZE** и **RELEASE** используют-

ся соответственно блоки **PREEMPT** (захватить) и **RETURN** (вернуть). Блок **PREEMPT** имеет следующий формат:

имя PREEMPT A,B,C,D,E

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

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

Блок **RETURN** имеет единственный операнд **A**? содержащий имя или номер устройства, подлежащего освобождению от захвата, и имеет следующий формат:

RETURN <A>

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

с предварительно вычисленным временем занятия устройства для продолжения обслуживания.

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

имя STORAGE A

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

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

имя ENTER A,B

имя LEAVE A,B

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

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

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

текущее содержимое МКУ не должно становиться отрицательным.

Многоканальные устройства имеют следующие СЧА**:**

S - текущее содержимое МКУ;

R - свободная емкость МКУ;

SR - коэффициент использования в долях 1000;

SA - целая часть среднего содержимого МКУ;

SM - максимальное содержимое МКУ;

SC - число занятий МКУ;

ST - целая часть среднего времени занятия МКУ.

Воспользуемся блоками **ENTER-LEAVE** и оператором **STORAGE** для моделирования двухканальной СМО с ожиданием. Если текущее содержимое МКУ с именем STO2 меньше 2, т.е. в блоке **ADVANCE** находится один или ни одного транзакта, то очередной транзакт, поступающий в модель через блок **GENERATE**, может войти в блок **ENTER** и затем в блок ADVANCE. Если же текущее содержимое МКУ равно 2, то очередной транзакт остается в блоке **GENERATE**, образуя очередь в списке текущих событий. По истечении задержки одного из двух обслуживаемых транзактов в блоке **ADVANCE** и после входа его в блок **LEAVE** первый из заблокированных транзактов сможет войти в блок **ENTER**.

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

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

INITIAL LS$ имя

INITIAL LS j

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

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

имя LOGIC X A

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

LOGIC S 9

LOGIC R FLAG

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

2.3 Блоки для сбора статистических данных

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

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

имя QUEUE A,B

имя DEPART A,B

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

Для сбора статистики о транзактах, заблокированных перед каким-либо блоком модели, блоки **QUEUE** и **DEPART** помещаются перед и после этого блока соответственно. При прохождении транзактов через блоки **QUEUE** и **DEPART** соответствующим образом изменяются следующие СЧА очередей**:**

Q - текущая длина очереди;

QM - максимальная длина очереди;

QA - целая часть средней длины очереди;

QC - общее число транзактов, вошедших в очередь;

QZ - число транзактов, прошедших через очередь без ожидания (число "нулевых" входов);

QT - целая часть среднего времени ожидания с учетом "нулевых" входов;

QX - целая часть среднего времени ожидания без учета "нулевых" входов.

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

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

Очень часто исследователя интересует не только среднее значение времени ожидания в очереди, но и дисперсия этого времени, а также статистическое распределение выборки времени ожидания, представляемое обычно графически в виде ги-

стограммы. Имея такое распределение, можно оценить вероятность того, что время ожидания превысит или не превысит некоторое заданное значение. Для сбора и обработки данных о выборочном распределении времени ожидания в очереди служат статистические объекты типа **Q-таблица.**

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

имя QTABLE A,B,C,D

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

Диапазон всевозможных значений времени ожидания в очереди, указанной в поле **A**, разбивается на ряд частотных интервалов, количество которых указано в поле **D**. Первый из этих интервалов имеет ширину от минус бесконечности до величины, указанной в поле **B**, включительно. Второй интервал включает значения, большие, чем величина первой границы в поле **B**, но меньшие или равные **B+C**, и т.д. Все промежуточные интервалы имеют одинаковую ширину, указанную в поле **C.** Наконец, последний интервал включает все значения, большие, чем последняя граница. Значения операндов **B, C** и **D** должны задаваться целыми константами. Операнд **B** может быть неположительным, хотя для **Q-таблицы** это не имеет смысла, так как время не может быть отрицательным. Операнды **C** и **D** должны быть строго положительными**.**

При прохождении транзакта через блоки **QUEUE** и **DEPART** его время ожидания фиксируется, и к счетчику частотного интервала таблицы, в который попало это время, добавляется 1. Одновременно в таблице накапливается информа-

ция для вычисления среднего значения и среднеквадратического отклонения (корня из дисперсии) времени ожидания. По окончании моделирования среднее значение и среднеквадратическое отклонение времени ожидания, а также счетчики попаданий в различные частотные интервалы выводятся в стандартный отчет GPSS/PC. Таблицы, как и другие объекты GPSS/PC, имеют СЧА**:**

TC - общее число транзактов, вошедших в очередь, связанную с таблицей;

TB - целая часть среднего времени ожидания в очереди;

TD - целая часть среднеквадратического отклонения времени ожидания в очереди.

Дополним модель из приведенного выше примера оператором **QTABLE** для получения распределения времени ожидания в очереди с именем LINE.

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

Оператор определения таблицы с именем WTIME разбивает ось времени на 10 частотных интервалов. Первый интервал включает значения от 0 до 50, второй - от 50 до 100, третий - от 100 до 150 и т.д. Последний, десятый, интервал включает значе-

ния, превышающие 450. Если, например, время ожидания некоторого транзакта в очереди составило 145 единиц модельного времени, то к счетчику третьего частотного интервала будет добавлена 1. Следует заметить, что информация в таблицу с именем WTIME заносится автоматически, при входе транзактов в блоки **QUEUE** и **DEPART**, и никаких специальных мер для этого принимать не требуется.

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

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

имя TABULATE A

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

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

время ожидания **TB** и среднеквадратическое отклонение времени ожидания **TD**.

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

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

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

2.4 Блоки, изменяющие маршруты транзактов

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

Блок **TRANSFER** (передать) служит для передачи входящих в него транзактов в блоки, отличные от следующего. Блок имеет девять режимов работы:

- безусловный (пробел);
- статистический (.);
- BOTH;
- ALL;
- PICK;
- функция (FN);
- параметр (P);
- подпрограмма (SBR);
- SIM.

Кроме того, операнд **A** может быть дробным числом, именем, положительным целым числом, СЧА или СЧА*<параметр>. Поля **B** и **C** задают возможные значения номеров следующих блоков или их положение. Использование значений описано при рассмотрении определенных режимов выбора. Операнды могут быть именем, положительным целым числом, СЧА или СЧА*<параметр>. Если поле **B** пусто, ассемблер записывает в нем номер блока, следующего за блоком **TRANSFER**. Рассмотрим подробнее режимы работа блока.

Безусловный режим выбора. Если операнд **A** пропущен, то блок **TRANSFER** работает в безусловном режиме. Входящее в блок **TRANSFER** сообщение переходит к блоку, указанному в поле **B**. Если сообщение в этот блок войти не может, попытка

направить сообщение к какому-либо другому блоку не производится. Например,

```
XFER  TRANSFER ,NEXT
NEXT  SEIZE    1
```

Сообщения, входящие в блок **TRANSFER** XFER, переходят к блоку NEXT.

```
TRANSFER , V$TER
```

Сообщения, которые входят в вышеприведенный блок **TRANSFER**, сразу переходят в блок, номер которого определяется переменной TER.

Статистический режим выбора. Когда операнд **A** не является зарезервированным словом, блок **TRANSFER** работает в статистическом режиме выбора. Значение аргумента, записанного после точки (.) в поле **A**, рассматривается как трехзначное число, показывающее (в частях от тысячи), какой процент входящих в блок сообщений следует направить к блоку, указанному в поле **C**. Остальные сообщения направляются к блоку, указанному в поле **B**, или к следующему по номеру блоку, если операнд **B** пропущен. Для каждого сообщения выбирается один из двух возможных вариантов; после того как выбор сделан, второй вариант для этого сообщения не рассматривается. Числовое значение может быть задано при помощи любого стандартного числового атрибута. Если вычисленное значение аргумента меньше или равно нулю, будет происходить безусловная передача сообщений к блоку, указанному в поле **B**. Если же значение аргумента больше или равно 1000, то будет происходить безусловная передача сообщений к блоку, указанному в поле **C**. Например,

```
BCD  TRANSFER .709, BLK1, BLK2
```

Из общего числа сообщений, входящих в блок BCD, в среднем .709 будут пытаться войти в блок BLK2. Остальные .209 будут пытаться войти в блок BLK1.

BCD TRANSFER .P1, BLK3, BLK4

Трехзначное число, записанное в параметре 1 сообщений, входящих в блок BCD, интерпретируется как вероятность (в частях от тысячи) того, что сообщение попытается войти в блок BLK4. В остальных случаях сообщение попытается войти в блок BLK3.

CDE TRANSFER .X1, BLK5, BLK6

Если в момент входа сообщений в блок CDE в ячейке SAVEVALUE 1 записано число 30, то в среднем 3 % от общего числа сообщений будет направлено к блоку BLK6, а остальные 97 % попытаются войти в блок BLK5.

Режим BOTH. Если в поле **A** стоит зарезервированное слово **BOTH**, блок **TRANSFER** работает в режиме **BOTH**. В этом режиме каждое входящее сообщение сначала пытается перейти к блоку, указанному в поле **B**. Если это сделать не удается, сообщение пытается перейти к блоку, указанному в поле **C**. Если сообщение не сможет перейти ни к тому, ни к другому блоку, оно остается в блоке **TRANSFER** и будет повторять в том порядке попытки перехода при каждом просмотре списка текущих событий, до тех пор, пока не сможет выйти из блока **TRANSFER**. Ниже приведен фрагмент программы, в котором сообщение сначала пытается перейти к блоку TRY1. Если оно не может войти в этот блок, оно пытается войти в блок TRY2. Если сообщение не может войти и в этот блок, оно остается в списке текущих событий и повторяет эти попытки при каждом просмотре списка до тех пор, пока не выйдет из блока **TRANSFER**.

```
TRANSFER  BOTH, TRY1, TRY2
TRY1  SEIZE       1
TRY2  SEIZE       2
```

Режим ALL. Если в поле **A** стоит зарезервированное слово **ALL**, блок **TRANSFER** работает в режиме **ALL**. В этом режиме каждое входящее сообщение прежде всего пытается перейти к блоку, указанному в поле **B**. Если сообщение в этот блок войти не может, то последовательно проверяются все блоки в определенном ряду в поисках первого, способного принять это сообщение, включая блок, указанный операндом **C**. Номер каждого проверяемого блока вычисляется как сумма номера предыдущего блока и шага, заданного операндом **D**:

N + M, N + 2M, N + 3M, ... L,

где **N** - номер блока, указанного в поле **B**;
M - значение шага, заданного в поле **D**;
L - номер блока, указанного в поле **C**.

Этот номер должен быть больше номера блока, указанного в поле **B**, на величину, кратную шагу M. Если операнд **D** не задан, то проверяется каждый блок, номер которого принадлежит этому ряду, включая блок, определенный операндом **C**. Блоки, номера которых выше номера блока, указанного в поле **C**, не проверяются. Как только первый блок, способный принять сообщение, будет найден, сообщение входит в этот блок и оттуда продолжает свое дальнейшее движение. Если сообщение не может перейти ни к одному из указанных блоков, оно остается в блоке **TRANSFER** и повторяет описанную выше процедуру при каждом просмотре списка текущих событий до тех пор, пока не выйдет из блока.

Поскольку обычно в полях **B** и **C** записываются символические метки блоков, блоки следует располагать таким образом, чтобы при присвоении номеров разность между номерами бло-

ков, указанных в полях **B** и **C**, была кратна шагу, указанному в поле **D**. Например,

TRANSFER ALL, 60, 120, 10

В этом примере сообщение будет последовательно пытаться перейти к блокам 60, 70, 80, ... 120.

TRANSFER ALL, NEXT1, NEXT2, 5

Здесь режим **ALL** допустим только в том случае, если разность между номерами, присвоенными блокам NEXT1 и NEXT2, кратна 5.

TRANSFER ALL, 60, 120, 25

В данном примере режим **ALL** недопустим, потому что разность между номерами блоков, записанных в полях **B** и **C**, не является кратной шагу, указанному в поле **D**.

Условными являются только режимы **BOTH** и **ALL**. Во всех остальных режимах выбор следующего блока производится в момент входа сообщения в блок. В режимах **BOTH** и **ALL** выбор следующего блока производится в момент снятия блокирующего условия. Следует отметить, что каждый раз, когда интерпретатор при просмотре списка текущих событий обнаруживает сообщение, задержанное в блоках **TRANSFER BOTH** или **TRANSFER ALL**, он пытается продвинуть сообщение, начиная с блока, указанного в поле **B**. Следовательно, в режиме **BOTH** в тех случаях, когда возможен переход к обоим блокам (**B** и **C**), блок **B** имеет некоторое преимущество. Аналогично, в режиме **ALL** в случае, когда возможен переход к нескольким блокам, блоки с меньшими номерами имеют некоторое преимущество перед блоками с большими номерами.

Режим PICK. Если в поле **A** стоит зарезервированное слово **PICK**, блок **TRANSFER** работает в режиме **PICK**. В этом режиме из последовательности блоков с номерами N, N+1, N+2,...M (N - номер блока, указанного в поле **B**, а M - номер блока, указанного в поле **C**) случайным образом выбирается один блок, к которому должно быть направлено сообщение. Все блоки, включая указанные в полях **B** и **C**, выбираются с одинаковой вероятностью, равной **1/(M-N)+1**. Сообщение пытается перейти только к выбранному для него блоку. Если сообщение не может сразу перейти к следующему блоку, то оно будет ждать в блоке **TRANSFER** до тех пор, пока не будет снято блокирующее условие. Номер блока в поле **C** должен быть больше или равен N+1. Например,

TRANSFER PICK,30,39

Сообщение, вошедшее в блок **TRANSFER**, пытается войти в один из 10 блоков (30,31,...39) с равной вероятностью: 1/10.

Режим "функция" (FN). Если в поле **A** стоит зарезервированное слово **FN**, блок **TRANSFER** работает в режиме "функция". Вычисляется значение функции, номер которой задан в поле **B** блока **TRANSFER**; если результат нецелый, от него берется целая часть. Для определения номера следующего блока полученное целое число складывается с аргументом поля **C** (в поле **C** может быть записан ноль). Сообщение пытается перейти только к блоку с вычисленным номером. Сообщение остается в блоке **TRANSFER** до тех пор, пока не сможет перейти именно к этому блоку. Например,

TRANSFER FN,3,PH3

Номер следующего блока = Значение функции FN3+Значение параметра 3 формата "полуслово"

Режим "параметр". Если в поле **A** стоит зарезервированное слово **P**, блок **TRANSFER** работает в режиме "параметр". Значение аргумента поля **B** интерпретируется как номер j параметра входящего сообщения. Для определения следующего номера блока для данного сообщения, значение этого параметра складывается со значением аргумента поля **C**. Если операнд **C** не задан, номер следующего блока будет равен значению параметра. Например,

TRANSFER P,12,37

Номер следующего блока = Значение параметра 12, вошедшего в блок сообщения + Значение, записанное в ячейке 37

Режим "подпрограмма" (SBR). Если в поле **A** стоит зарезервированное слово **SBR**, блок **TRANSFER** работает в режиме "подпрограмма". Вошедшее в блок **TRANSFER** сообщение будет пытаться перейти к блоку, указанному в поле **B**. Значение аргумента поля **C** интерпретируется как номер параметра; в этом параметре записывается номер j данного блока **TRANSFER**. Если такого параметра нет, то он создается. Этот режим блока **TRANSFER** обычно используется для перехода к подпрограмме, началом которой является блок, указанный в поле **B**. Например,

TRANSFER SBR,NEXT,10

Если в конце подпрограммы записать блок

TRANSFER P,10,1,

то сообщение сможет вернуться к блоку, следующему за блоком **TRANSFER SBR**, где следующий блок равен текущему значе-

нию, записанному в параметре под номером 10 (в данном случае это номер блока **TRANSFER SBR**) плюс 1.

Режим SIM. Если в поле **A** стоит зарезервированное слово **SIM**, то блок **TRANSFER** работает в режиме **SIM**. Режим введен для случая, когда требуется одновременное выполнение нескольких условий. Каждое сообщение имеет свой индикатор задержки (назовем его индикатором **SIM**). В этом индикаторе записывается результат любой попытки сообщения войти в следующий блок. Если интерпретатор обнаруживает условия, препятствующие входу сообщения в блок, то индикатор **SIM** этого сообщения устанавливается в единицу. Если все условия перехода к следующему блоку удовлетворяются, то индикатор **SIM** остается равным нулю. Если не выполняется хотя бы одно из условий, то индикатор **SIM** данного сообщения устанавливается в единицу (в режимах **BOTH** и **ALL** индикатор **SIM** устанавливается в единицу только в том случае, когда переход невозможен ни к одному из указанных блоков). При входе сообщения в блок **TRANSFER** проверяется значение индикатора **SIM**. Если он равен нулю, сообщение направляется к следующему блоку, указанному в поле **B**. Если индикатор **SIM** равен единице, сообщение направляется к блоку, указанному в поле **C**, а индикатор **SIM** устанавливается в "0". В любом случае сообщение будет пытаться перейти только к выбранному для него блоку и будет находиться в блоке **TRANSFER** до выполнения соответствующих условий. В момент, когда создаются условия для выхода сообщения из блока **TRANSFER**, значение индикатора **SIM** не проверяется. Состояние индикатора **SIM** отмечается символом X в колонке DELAY распечатки информации о сообщениях. При задержке сообщений в блоках **ASSEMBLE, GATHER** или **MATCH** индикатор **SIM** в единицу не устанавливается.

Изменение индикатора SIM в блоке ADVANCE. Каждый раз, когда сообщение выходит из блока **ADVANCE** с нулевым временем задержки, индикатор **SIM** становится равным нулю.

После того, как сообщение покинуло блок **ADVANCE,** оно может быть снова задержано по каким-либо причинам прежде, чем дойдет до тех блоков, в которых проверяется одновременность выполнения ряда условий. Следовательно, индикатор **SIM** может быть установлен в единицу до того, как начнется проверка условий. В таком случае перед блоками, в которых проверяется одновременное выполнение условий, следует поместить блок **TRANSFER SIM**, в котором в полях **B** и **C** указан один и тот же блок - первый из блоков, проверяющих условия. После прохождения через этот блок, индикатор **SIM** сообщения снова станет равным нулю. Например,

TRANSFER SIM,10,10

Этот блок позволяет установить в "0" индикаторы всех сообщений, входящих в блок 10.

Внутренние операции блока TRANSFER. При входе сообщения в блок **TRANSFER** (за исключением блоков, работающих в режимах **BOTH** и **ALL**) вычисляется номер следующего блока, к которому сообщение должно перейти, и сообщение пытается перейти к этому блоку. Вычисленный номер блока должен быть допустимым номером блока в текущей модели. Если сообщение не может перейти в этот блок, то номер блока запоминается. Номер вычисляется только один раз. Интерпретатор все время пытается продвинуть сообщение только к этому блоку. Если блок **TRANSFER** работает в режиме **BOTH** или **ALL,** запоминается номер последнего блока, указанного в поле **C.** Этот номер также вычисляется только один раз. Если сообщение не сможет перейти ни к одному из указанных в режимах **BOTH** или **ALL** блоков, интерпретатор будет повторять попытки для всех указанных блоков при каждом просмотре списка текущих событий. Эти попытки будут повторяться даже в том случае, если блокирующие условия не будут сняты. По этой причине, использование блока **TRANSFER** в режимах **BOTH**

или **ALL** может увеличить время обработки. В этом случае сообщение можно поместить в список пользователя на время, пока не будет найден блок, способный принять сообщение. Это может быть сделано с помощью блоков **LINK** и **UNLINK.**

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

имя TEST X A,B,C

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

TEST LE P$TIME,C1

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

TEST L Q$LINE,5,OUT

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

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

имя GATE X A,B

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

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

Например, блок

GATE SNE BUF3

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

GATE LR 4,BLOK2

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

Блоки рассматриваемой группы используются при моделировании различных СМО с потерями заявок. Воспользуемся, например, блоками **TRANSFER** для моделирования двухканальной СМО с отказами и повторными попытками.

```
STO2  STORAGE   2
EXP   FUNCTION  RN1,C24
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915
.7,1.2/.75,1.38/.8,1.6/.84,1.85/.88,2.12/.9,2.3
```

```
.92,2.52/.94,2.81/.95,2.99/.96,3.2/.97,3.5/.98,3.9
.99,4.6/.995,5.3/.998,6.2/.999,7/.9998,8
        GENERATE   100,FN$EXP
ENT1    TRANSFER   BOTH,,REFUS
        ENTER      STO2
        ADVANCE    160,FN$EXP
        LEAVE      STO2
        TERMINATE  1
REFUS   TRANSFER   .1,,OUT
        ADVANCE    250,FN$EXP
        TRANSFER   ,ENT1
OUT     TERMINATE  1
```

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

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

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

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

При использовании блока **GATE** модель принимает вид, показанный ниже. В этом варианте транзакт проходит в блок **ENTER**, если условие "МКУ STO2 не заполнено" истинно.

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

```
        ENTER     STO2
        ADVANCE   160,FN$EXP
        LEAVE     STO2
        TERMINATE 1
REFUS   TRANSFER  .1,,OUT
        ADVANCE   250,FN$EXP
        TRANSFER  ,ENT1
OUT     TERMINATE 1
```

2.5 Блоки, работающие с памятью

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

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

INITIAL X$ имя,значение

INITIAL Xj ,значение

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

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

имя SAVEVALUE A,B

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

SAVEVALUE 5,Q$LINE

SAVEVALUE NREF+,1

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

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

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

имя MATRIX A,B,C

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

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

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

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

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

```
INITIAL    MX$ имя (a,b),значение

INITIAL    MXj    (a,b),значение
```

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

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

имя MSAVEVALUE A,B,C,D

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

MSAVEVALUE 5,3,2,X1

MSAVEVALUE MTAB+,PROW,PCOL,1

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

MX$ имя (a,b)

MXj (a,b)

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

MX5(2,1)

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

2.6 Блоки для работы со списками пользователя

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

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

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

имя LINK A,B

В поле **A** задается имя или номер списка пользователя, в который безусловным образом помещается транзакт, вошедший в блок. Поле **B** определяет, в какое место списка пользователя следует поместить этот транзакт. Если в поле **B** записано

ключевое с**лово FIFO**, то транзакт помещается в конец списка, если **LIFO** - в начало списка. В других случаях транзакты упорядочиваются в соответствии с вычисленным значением поля **B**, где обычно записывается один из СЧА транзактов, таких как **PR, M1 или P**. Если поле **B** содержит СЧА **PR**, то транзакты упорядочиваются по убыванию приоритета. В остальных случаях производится упорядочение по возрастанию указанного СЧА. Например, блок

```
LINK   5,FIFO
```

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

```
LINK   BUFER,P$ORDER
```

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

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

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

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

имя UNLINK X A,B,C,D,E,F

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

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

UNLINK 5,NEXT,1

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

UNLINK BUFER,ENT1,1,BACK

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

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

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

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

Каждый список пользователя имеет следующие СЧА:

CH - текущая длина списка;

CA - средняя длина списка (целая часть);

CM - максимальная длина списка;

CC - общее число транзактов, вошедших в список;

CT - целая часть среднего времени пребывания транзакта в списке.

Воспользуемся рассмотренными блоками для моделирования многоканальной СМО с ожиданием транзактов в списке пользователя. Если МКУ с именем STO2 не заполнено, блок

GATE впускает вновь прибывший транзакт в блок **ENTER**, и в МКУ занимается один канал. Если же МКУ заполнено, то блок **GATE** направляет транзакт в блок **LINK** с именем WAIT, помещающий транзакт в конец списка пользователя с именем BUFER, моделирующего очередь к МКУ. Каждый транзакт, покидающий МКУ по завершении обслуживания и освобождающий один канал, проходит блок **UNLINK** и выводит один транзакт с начала списка (если список не пуст), направляя его в блок с именем ENT1 на занятие канала в МКУ.

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

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

Рассмотрим еще один пример, иллюстрирующий использование списков пользователя для организации нестандартных дисциплин обслуживания. Пусть в одноканальной СМО с ожиданием требуется организовать такую дисциплину, при которой приоритет отдается заявкам с наименьшим временем обслуживания.

В параметр TSRV поступающих в модель транзактов в блоке **ASSIGN** записывается случайное время обслуживания, вычисляемое с использованием функции EXP. Если устройство SYSTEM свободно, то блок **GATE** впускает транзакт в блок **SEIZE,** и устройство занимается на время P$TSRV. Если же в момент поступления транзакта устройство занято, то блок **GATE** направляет транзакт в блок **LINK**, который вводит транзакт в список пользователя LINE, упорядочивая транзакты по возрастанию времени обслуживания, записанного в параметре P$TSRV. Блок **UNLINK** по освобождении устройства выводит с начала списка транзакт с наименьшим временем обслуживания, обеспечивая тем самым заданную дисциплину.

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

3 Управляющие операторы **GPSS/PC**

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

START A,B,C,D

Поле **A** содержит константу, задающую начальное значение счетчика завершений. В поле **B** может быть записано ключевое слово **NP** - признак подавления формирования стандартного отчета по завершении моделирования. Если поле **B** пусто, то по окончании прогона модели формируется отчет со стандартной статистической информацией о всех объектах модели. Поле **C** не используется и сохранено для совместимости со старыми версиями GPSS. Поле **D** может содержать 1 для включения в отчет списков текущих и будущих событий. Если поле **D** пусто, то выдача в отчет содержимого этих списков не производится.

Оператор **SIMULATE** (моделировать) устанавливает предел реального времени, отводимого на прогон модели. Если прогон не завершится до истечения этого времени, то он будет прерван принудительно с вы дачей накопленной статистики в отчет. Оператор **SIMULATE** имеет единственный операнд A, содержащий предельное время моделирования в минутах, задаваемое константой. Оператор размещается перед оператором **START**, начинающим лимитированный прогон.

Оператор **RMULT** (установить значения генераторов) позволяет перед началом прогона установить начальные значения генераторов случайных чисел RN, определяющие генерируемые ими последовательности. Поля **A-G** оператора могут содержать начальные значения генераторов соответственно RN1-

RN7, задаваемые константами. Начальные значения генераторов, не установленные операторами **RMULT**, совпадают с номерами генераторов.

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

С оператором **RESET** связано различие между относительным (СЧА С1) и абсолютным (СЧА АС1) модельным временем. Таймер относительного времени С1 измеряет модельное время, прошедшее после последнего сброса статистики оператором **RESET**, а таймер абсолютного времени АС1 - модельное время, прошедшее после начала первого прогона модели. Если не использовалось ни одного оператора **RESET**, то значения этих таймеров совпадают. Оператор **RESET** устанавливает таймер С1 в ноль и не влияет на таймер АС1.

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

Пусть, например, в модели необходимо отбросить статистику, собираемую на первой тысяче транзактов. Это может быть сделано способом, показанным ниже. Первый оператор **START** начинает прогон модели длиной 1000 транзактов (переходный период). Поскольку статистика, накопленная на этом периоде, не используется, в поле **B** оператора указан признак подавления формирования отчета NP. Оператор **RESET** сбрасывает накопленную статистику, не изменяя состояния модели. Второй оператор **START** начинает основной прогон модели с формированием отчета по завершении прогона.

```
EXP   FUNCTION   RN1,C24
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915
.7,1.2/.75,1.38/.8,1.6/.84,1.85/.88,2.12/.9,2.3
.92,2.52/.94,2.81/.95,2.99/.96,3.2/.97,3.5/.98,3.9
.99,4.6/.995,5.3/.998,6.2/.999,7/.9998,8
      GENERATE   100,FN$EXP
      ASSIGN     TSRV,80,EXP
      GATE NU    SYSTEM,WAIT
SFAC  SEIZE      SYSTEM
      ADVANCE    P$TSRV
      RELEASE    SYSTEM
      UNLINK     LINE,SFAC,1
      TERMINATE  1
WAIT  LINK       LINE,P$TSRV
      START      1000,NP
      RESET
      START      10000
```

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

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

Пусть, например, требуется повторить прогон модели три раза при емкости МКУ, равной 1, 2 и 3. После каждой очистки модели оператором **CLEAR** оператор **STORAGE** устанавливает новое значение емкости МКУ с именем STO2.

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

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

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

4 Некоторые приемы конструирования GPSS-моделей

4.1 Косвенная адресация

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

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

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

```
ASSIGN     TYPE,FN$CLASS
PRIORITY   V$PRIOT
QUEUE      LINE
QUEUE      P$TYPE
ENTER      STO2
DEPART     P$TYPE
DEPART     LINE
ADVANCE    FN$MEAN,FN$EXP
LEAVE      STO2
TABULATE   TTIME
TERMINATE  1
```

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

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

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

дый транзакт "отмечается" в блоках **QUEUE** в двух очередях. Очередь с именем LINE является общей для транзактов всех классов. Входя в следующий блок **QUEUE**, транзакт отмечается в очереди с номером 1, 2 или 3 в зависимости от класса заявки, записанного в параметре TYPE. Аналогичным образом фиксируется уход из очередей в блоках **DEPART**. Таким образом, в модели создается четыре объекта типа "очередь": одна очередь с именем LINE и три с номерами 1, 2 и 3. При этом три последние очереди создаются одной парой блоков

QUEUE-DEPART. В этом и заключается эффект косвенной адресации.

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

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

LINE EQU 4

4.2 Обработка одновременных событий

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

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

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

Предположим теперь, что первым в списке текущих событий расположен транзакт, поступающий на вход модели. Так как условие "МКУ STO2 не заполнено" ложно, то блок **GATE** направит этот транзакт в блок с именем REFUS. Таким образом, в модели будет зафиксирован отказ в обслуживании, хотя в

этот же момент модельного времени, после обработки транзакта, освобождающего канал, МКУ станет доступным.

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

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

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

Транзакты, поступающие в модель через блок **GENERATE**, имеют нулевой приоритет. Такой же приоритет имеют транзакты, получившие отказ в обслуживании, направ-

ленные в блок с именем REFUS и затем повторно поступающие в блок с именем ENT1. Те же транзакты, что поступают на обслуживание, повышают приоритет до 1 в блоке **PRIORITY**, и после выхода из блока ADVANCE возвращаются из списка будущих в список текущих событий, располагаясь в начале списка. Таким образом, нужный порядок транзактов обеспечивается, и временной узел будет обработан правильно.

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

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

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

Проведенный анализ показывает, что для правильной обработки временного узла необходимо обеспечить такой порядок расположения транзактов в списке текущих событий,

чтобы первым всегда располагался вновь пришедший транзакт. В рассматриваемом случае этого можно добиться, используя блок **PRIORITY** с операндом BU.

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

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

5 Технология работы с пакетом GPSS-World

5.1 Загрузка интегрированной среды

Для запуска программы необходимо выбрать и дважды щелкнуть на файле GPSS World.exe. При запуске появляется окно, представленное на рисунке 1.

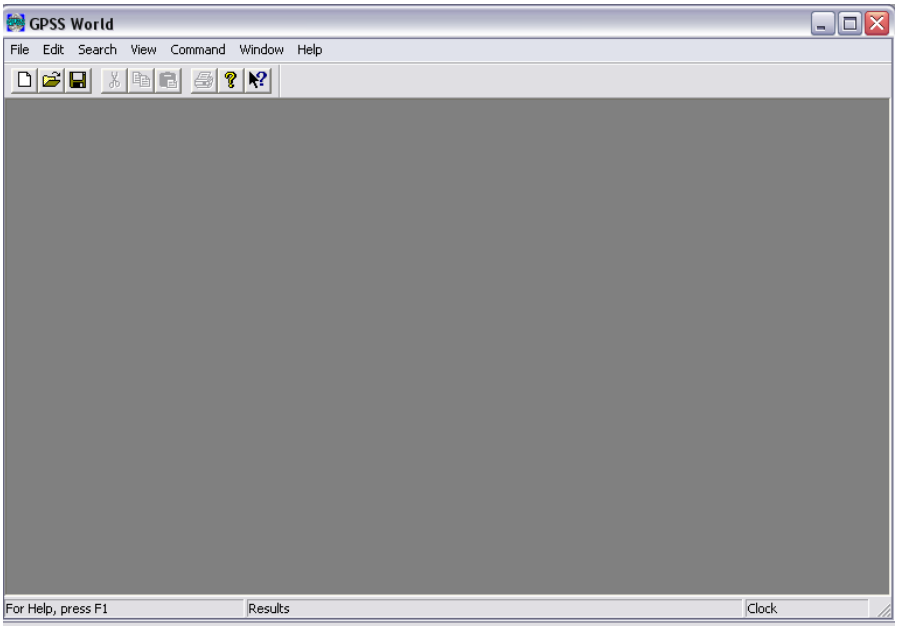

Рис.1. Окно GPSS World

С помощью команд на панели инструментов возможен детальный анализ модели проектирования, рассмотрение его функций, реализованных в программе. Рассмотрим подробнее состав панели инструментов.

В меню File находятся такие функции как открытие (закрытие) файла, сохранение, печать файла, а также связь с Интернетом. Меню Edit предназначено для редактирования. Меню

earch необходимо для поиска необходимого знака, блока строки. В меню View расположены кнопки, относящиеся к панели инструментов и системным часам. Меню Command предназначено для симуляции, трансляции проекта как по шагам, так и единовременно, а также в этом меню находятся стандартные блоки CLEAR и RESET. В меню Window находятся подменю, отвечающие за графический вывод в виртуальных окнах. Существует несколько окон, каждое из которых предназначено для определенных целей. Главные из них:

- *Окно «Blocks» («Блоки»)* - оперативный обзор динамики блоков.

- *Окно «Plot» («График»)* - оперативный обзор изменения значений системных числовых атрибутов (СЧА) и выражений с помощью графиков.

- *Окно «Expression» («Выражения»)* - оперативный обзор значений выражений.

- *Окно «Facilities» («Устройства»)* - оперативный обзор динамики изменения состояния устройств.

- *Окно «Table»(«Таблица»)* - оперативный обзор динамики изменения таблицы или Q-таблицы в виде гистограммы.

- *Окно «Queues»(«Очереди»)* - оперативный обзор динамики изменения очередей.

- *Окно «Storages»*(«Памяти») - оперативный обзор динамики изменения содержимого памятей.

5.2 Ввод новой модели

Если исходная программа с моделью еще не введена и не записана на диске, то необходимо ввести ее с клавиатуры. Ввод производится в программный файл. Для ввода или удаления блоков и их операндов также можно использовать стандартное подменю « Insert GPSS blocks». После ввода новой модели необходимо оттранслировать программу.

5.3 Редактирование текста модели

Удалить строки из исходной программы можно при помощи команды Delete line. При этом удаляется та строка, в которой находится курсор. Для удаления нескольких строк достаточно выделить их и нажать на кнопку Delete.

При необходимости вставить в текст новый оператор, поместив его между уже введенными операторами, достаточно ввести его с промежуточным номером строки. Вы можете перенумеровать строки, нажав на уже введенный номер и удалив старый. Подводя курсор к нужным позициям строки, вы можете внести в нее необходимые изменения.

5.4 Запись и считывание модели с диска

Если работа с моделью предполагается и по окончании данного сеанса, то после ввода и редактирования исходную программу имеет смысл записать на диск. Для этого необходимо ввести команду SAVE (сохранить). Файл должен иметь расширение .GPS. Записав модель в файл, вы сможете в следующем сеансе работы с GPSS не вводить ее заново с клавиатуры, а считать с диска.

5.5 Прогон модели и наблюдение за моделированием

После того, как исходная программа модели введена с клавиатуры или считана с диска и оттранслирована, в памяти ПК создалась текущая модель, и теперь можно выполнить ее прогон. Для этого в меню Command выбирается оператор Create Simulate. Если прогон модели достаточно длинный, то можно наблюдать за процессом моделирования, открывая те или иные графические окна. Это производится путем выбора необходимого окна.

5.6 Получение и интерпретация стандартного отчета

По завершении прогона модели раздается звуковой сигнал, и появляется окно со стандартным отчетом. Отчет, создаваемый по завершении моделирования, можно записать в файл со стандартным именем REPORT.GPS. Это имя может быть изменено командой **REPORT** (создать отчет), имеющей следующий формат:

REPORT A,B

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

Выход из интегрированной среды (завершение сеанса) производится путем ввода управляющего оператора END (закончить) или нажатием кнопкой на значке закрытия рабочего окна.

Отчет содержит следующую информацию:

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

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

3) сведения о блоках модели, включающие для каждого блока номер строки исходной программы (LINE), номер или имя

блока (LOC), название блока (BLOCK_TYPE), количество транзактов, прошедших через блок (ENTRY_COUNT), текущее количество транзактов в блоке в момент завершения моделирования (CURRENT_COUNT), количество транзактов, заблокированных перед блоком в момент завершения моделирования (RETRY);

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

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

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

7) сведения о статистических таблицах модели, включающие для каждой таблицы ее имя или номер (TABLE), среднее значение (MEAN) и среднеквадратическое отклонение (STD.DEV.) табулируемой величины, границы частотных интервалов (RANGE), частоты (FREQUENCY) и накопленные ча-

стоты в процентах (CUM.%) попадания наблюдений в эти интервалы;

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

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

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

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

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

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

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

Разумеется, сведения об объектах того или иного типа появляются в отчете только в том случае, если в модели присутствует хотя бы один объект данного типа. Ниже приведен отчет о прогоне модели рассмотренного ранеепримера.

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

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

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

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

TABLE	MEAN	STD.DEV.	RETRY	RANGE		FREQUENCY	CUM.%
WTIME	5.59	25.23	0				
				-	50	144	96.00
				50 -	100	3	98.00
				100 -	150	1	98.67
				150 -	200	2	100.00
TTIME	69.48	70.88	0				
				-	100	117	78.00
				100 -	200	23	93.33
				200 -	300	8	98.67
				300 -	400	2	100.00

6 Современные пакеты моделирования на основе GPSS

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

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

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

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

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

Так как GPSS World™ достаточно новый пакет, системы быстрого ввода моделей для него еще не разработаны, а сам интерпретатор, хотя и работает под управлением ОС Windows, не использует графические возможности системы и ввод модели, как и у предшественника (GPSS/PC) производиться с помощью текстового редактора. Кроме того, при произведении экспериментов результаты выводятся в виде текстовых документов (отчеты, журнал прогона), что крайне неудобно, так, например, для того чтобы перенести результаты моделирования в табличный процессор (например, MS Excel) необходимо вручную вводить в таблицы результаты всех экспериментов.

Для решения проблем автоматизации ввода, проведения экспериментов и сбора результатов предназначен, представляемый **комплекс "ВиРОМ 2.0"**

6.1 Object

Новый продукт, Object GPSS содержит много новых функций и процедур, расширяющих возможности разработчика модели. Общее число блоков в системе доведено до 74. Система содержит 11 типов объектов и 73 процедуры для управления моделью. Общее число функций достигает 123.

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

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

Функции и параметры модели – типизированы, и могут быть следующих базовых типов: целый, вещественный, строковый и логический. Каждый тип строго контролируется, а преобразование типов, если необходимо, должно выполняться явным образом. В системе можно использовать многомерные массивы любых объектов, что намного повышает возможности разработчика моделей. В системе можно связать наборы X - параметров с любыми объектами или даже создавать и уничтожать наборы X-параметров в динамике.

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

Система Object GPSS, ориентирована на дискретно- событийное моделирование систем массового обслуживания. Представление жизни модели как движения во времени заявок, перемещающихся в модели и обслуживающихся в устройствах очень естественно для многих задач имитационного моделирования. Система Object GPSS, достаточно легка для изучения. Студенты после короткого времени обучения могут создавать достаточно сложные модели. Автоматический сбор статистики - это огромная помощь для начинающих. Для многих реальных систем, моделирование на Object GPSS, выполняется гораздо легче, чем другими методами. Компактность текста модели и возможность использования графического интерфейса позволяют ускорить создание прототипов моделей. При этом на каждую из них можно получить быстрый отклик после улучшений проведенных пользователем.

Object GPSS – это высоко интегрированная компьютерная среда моделирования общего назначения, разработанная для

профессионалов моделирования. Это - мощный инструмент моделирования, покрывающий, дискретно-событийное моделирование, с чрезвычайно высоким уровнем взаимодействия и визуализации. При использовании Object GPSS, возможно предсказать поведение весьма сложных реальных систем.

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

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

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

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

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

Однако для сложных систем, нужно знакомства со всем тем, что может предложить Object GPSS.

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

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

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

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

6.2 Комплекс «ВиРОМ»

Полное наименование программной разработки: «Комплекс ввода и расширенной обработки моделей» (Коротко “Комплекс ВиРОМ”). Представляемый программный комплекс предназначен для визуализации процесса построения и отладки имитационных моделей систем, автоматизации процесса исследования поведения модели (путем автоматического проведения экспериментов), автоматизации сбора данных о результатах мо-

делирования в ходе проведения экспериментов. Комплекс предназначен для работы с интерпретатором GPSS World™ .

Приведем некоторые недостатки системы GPSS World и пути их решения средствами комплекса ВиРОМ (таблица 1).

Таблица 1

Недостатки системы GPSS World и пути их решения

Недостаток или проблема	Решение
Текстовое представление модели	С помощью приложения GPSS Constructor проектируемая модель представляется в виде блок-схемы, что является более наглядными удобным при проектировании и реализации.
Сложность формирования не тривиальных, не стандартных законов распределения	Использование приложения Function's Builder позволяет ввести закон распределения (ЗР) графически в виде дифференциальной функции ЗР, приложение автоматически интегрирует функцию ЗР и представляет в виде используемом в GPSS.
Представление результатов моделирования (и/или экспериментов) в виде текстовых документов	Приложение Model Multi Runner предоставляет пользователю средства, с помощью которых при проведении многократных экспериментов их результаты накапливаются в табличном виде, что более удобно при статистической обработке данных на ЭВМ (например, с помощью MS Excel).
Сложность организации экспериментов над моделью	Используя средства приложения Model Multi Runner, пользователь получает возможность написать сценарий эксперимента путем декомпозиции последнего на простые операции, не прибегая к средствам языка PLUS, который встроен в GPSS World.

Как видно из названия ВиРОМ является составным программным средством, следовательно, наиболее правильным описанием назначения комплекса будет описание назначений его приложений.

GPSS Constructor - приложение предназначено для визуального построения моделей систем в виде блок схемы эквивалентной коду модели на языке ИМ GPSS. Также приложение позволяет пользователю создавать и редактировать предмодельные описания и Plus-процедуры используемые в модели.*Model Multi Runner* - позволяет быстро произвести множество прогонов модели, изменяя параметры блоков модели по заданному сценарию, собирая данные о прогоне модели из отчетов в таблицы с возможностью последующего сохранения в текстовый файл либо экспорта в Microsoft Excel, для дальнейшей статистической обработки.

Functions Builder - переводит функцию плотности распределения заданную графически, в обратную, интегрируя её, и представляет её в виде задающих пар чисел используемом в GPSS.

Model Report Master - позволяет пользователю собрать данные о модели и ее работе, используя данные экспортируемые из других приложений комплекса в ходе разработки, отладки и исследования поведения модели, и представить их в виде HTML – документов. Последние предполагают их использование в качестве рабочих отчетов на промежуточных этапах разработки и исследования модели.

Данный комплекс предназначен для использования в области системного анализа и рассчитан на использования специалистами в области проектирования АСУ либо системными аналитиками.

Из всех приложений комплекса с интерпретатором GPSS Word взаимодействуют только GPSS Constructor и Model Multi Runner.

Проблема передачи модели для прогона в интерпретатор GPSS World заключается в том, что формат, в котором интерпретатор сохраняет модели и соответственно из которого их загружает неизвестен. Хотя, при просмотре файлов в текстовом редакторе удалось установить, что в конце файла сохраненного в формате GPSS World записан текст сохраненного документа (журнала прогона, отчета, модели и т.д.) в формате RTF (Rich Text Format). Это позволяет отсечь эту часть файла и загрузить ее в приложение комплекса. Но проблема обратной передачи (из приложений в интерпретатор) этим не решается.

Для решения проблемы передачи модели из приложений в интерпретатор используется метод эмуляции потока данных от пользователя, то есть имитируются нажатия клавиш на клавиатуре пользователем. Таким образом, после загрузки приложения и передачи кода модели в буфер обмена операционной системы, мы с легкостью можем передать текст кода модели в окно интерпретатора, сохранить модель и запустить прогон модели. Для того чтобы выполнить все выше описанное необходимо лишь знать последовательность нажимаемых клавиш на клавиатуре.

Преимуществом данного метода работы с интерпретатором является независимость от внутреннего формата представления данных в интерпретаторе (с изменением версий формат может меняться), так как все действия производятся стандартными средствами операционной системы.

Недостатком же этого метода является проблема синхронизации работы интерпретатора и приложения вызвавшего его. Но эта проблема решается с помощью отслеживания тайм-аутов в работе интерпретатора.

ЛАБОРАТОРНАЯ РАБОТА № 1
ИЗУЧЕНИЕ СПОСОБОВ ЗАДАНИЯ СЛУЧАЙНОЙ ЗАГРУЗКИ ПРИ МОДЕЛИРОВАНИИ ВЫЧИСЛИТЕЛЬНЫХ СТРУКТУР

1. Общие указания по выполнению лабораторной работы

1.1. Цель работы

получение практических навыков и изучение способов задания случайной нагрузки при моделировании вычислительных структур (ВС) средствами языка GPSS;

приобретение знаний о возможностях языка GPSS по моделированию случайных процессов; способах задания функций случайного аргумента; методах случайного альтернативного ветвления транзактов в модели;

получение навыков описания операторов случайной генерации тран- закто в, описания функций случайного аргумента методов случайного альтернативного ветвления в модели; трансляции моделей, составленных на языке GPSS и формирование файла результатов; интерпретации результатов моделирования.

1.2. Используемое оборудование и программное обеспечение

Интерпретатор GPSS WORLD, персональный компьютер, совместимый с IBM PC с объемом оперативной памяти не менее 512 Мб, операционная система Windows 7 и выше.

1.3. План выполнения лабораторной работы

изучить загрузку интерпретатора GPSSR/PC;

составить на языке GPSS модель по заданию преподавателя;

оттранслировать описание модели;

устранить ошибки, если они есть и получить файл выходной статистики;

провести анализ результатов.

2. Домашние задания и методические указания по их

2.1. Задание первое

По методическому руководству к лабораторной работе и литературным источникам, например [1,2,4], ознакомиться со способами задания случайной нагрузки при моделировании ВС средствами языка GPSS.

2.2. Методические указания по выполнению первого задания

2.2.1. Имитационное моделирование

Имитационное моделирование проводится с целью исследования ВС путем построения и исследования модели ВС. Так как реальные условия функционирования ВС на этапе моделирования, как правило, неизвестны, то моделирование проводится при случайных условиях или, иными словами, при случайной нагрузке на ресурсы ВС. При этом для получения достоверных статистических результатов моделирования необходимо обеспечить, чтобы случайная нагрузка была не меньше реальной.

При случайных параметрах модели выходные характеристики модели также становятся случайными. К числу таких характеристик относятся время пребывания транзактов в модели, промежутки времени между моментами прохождения транзактами определенных точек модели, время пребывания транзактов в очередях к приборам и многоканальным устройствам (МКУ), длины очередей и другие.

2.2.2. Аппарат моделирования случайной нагрузки языка GPSS

Случайная нагрузка при моделировании ВС создается путем задания:

1) случайных моментов времени поступления транзактов в модель через блок GENERATE;

2) случайной длительности обслуживания транзактов в приборах и многоканальных устройствах;

3) Случайных альтернативных переходов транзактов между блоками.

2.2.3. Генерация транзактов

Блок GENERATE является источником потока сообщений в модели. В данном блоке производится подготовка сообщений, и запуск их в модель через интервалы времени, заданные пользователем. Кроме задания правильной временной последовательности, пользователь может в блоке GENERATE задать некоторую информацию об атрибутах сообщений.

Блок GENERATE имеет следующий формат записи:

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

В поле A указывается время, которое определяет интервал между моментами генерации сообщений блоком GENERATE. Операнд A может быть именем, положительным целым числом или непосредственно СЧА. Нельзя использовать в качестве операнда параметры сообщения.

В поле B задается модификатор, который изменяет значения интервала генерации сообщений по сравнению с интервалом, указанным в поле A. Операнд B может быть именем, положительным целым числом или непосредственно СЧА. Нельзя использовать в качестве операнда параметры сообщения.

Может быть два типа модификаторов: модификатор-интервал и модификатор-функция.

С помощью модификатора-интервала задается равномерный закон распределения времени между генерацией сообщений. При вычислении разницы значений, заданных в полях A и B, получается нижняя граница интервала, а при вычислении суммы — верхняя граница. После генерации очередного сообщения выбирается число из полученного интервала, и это будет значение времени, через которое следующее сообщение выйдет из блока GENERATE.

Например:

```
GENERATE 25,10
```

В этом случае генерация сообщений производится по равномерному закону из интервала — (15,35).

Более сложные распределения могут быть представлены при использовании модификатора-функции, под действием которого вычисленное значение аргумента поля A умножается на значение функции, заданной в поле B. От значения функции целая часть не берется; отбрасывание дробной части производится только после умножения его на среднее значение.

Например:

```
GENERATE FN$Norm
```

Здесь необходимо пояснить, что обращение к функциям и переменным происходит с указанием типа идентификатора (в данном случае это FN, означающий, что Norm — это функция и разделитель $). Это вызвано тем, что области имён функций, переменных, таблиц и прочих элементов не пересекаются. Иначе говоря, в программе могут сосуществовать функция Fun и переменная Fun. Это частое явление, например подобная система реализована в очень популярном языке PERL. В классических структурных языках подобная практика считается вредной.

Если первый из вычисленных интервалов между моментами генерации сообщений равен 0, то этот интервал принимается равным 1. Если поля A и B пустые, что указывает на нулевой интервал между моментами генерации собщений, то блок GENERATE будет генерировать сообщения до тех пор, пока не использует все сообщения, которые могут быть активными в какой-то определенный момент времени. Чтобы предупредить это, следует либо задать предел генерации (поле D), либо за блоком GENERATE должен следовать блок, который вызывает блокирующее условие.

В поле C задается начальная задержка. Начальная задержка относится к моменту формирования первого сообщения в блоке GENERATE как при первом просчете модели, так и по-

сле выполнения операции CLEAR. Начальная задержка — это момент времени, в который первое сгенерированное сообщение должно выйти из блока GENERATE; поля A и B на задержку сообщения влияния не имеют. Начальная задержка может быть меньше, равна или больше среднего времени, заданного в поле A. Операнд C может быть именем, положительным целым числом или непосредственно СЧА. Нельзя использовать в качестве операнда параметры сообщения.

В поле D задается предел генерации. Эта величина представляет собой максимальное число сообщений, которое будет создано в блоке GENERATE. Операнд D может быть именем, положительным целым числом или непосредственно СЧА. Нельзя использовать в качестве операнда параметры сообщения. Если поле D пусто, блок генерирует неограниченное число сообщений. Предел генерации инициализируется повторно операцией CLEAR.

Поле E определяет приоритет сообщений. Операнд E может быть именем, положительным целым числом или непосредственно СЧА. Нельзя использовать в качестве операнда параметры сообщения. Если поле E не задано, приоритет по умолчанию равен 0.

В начальный момент времени в каждом блоке GENERATE производится подготовка к выходу одного сообщения. На этой стадии модель еще полностью не инициализирована для выполнения. По этой причине все СЧА, описанные в блоке GENERATE, должны быть уже определены. В модели блоку GENERATE должны предшествовать операторы описания INITIAL, FUNCTION и VARIABLE. Это делается для того, чтобы СЧА в блоке GENERATE, который ссылается на них, давали желаемые результаты.

Когда сообщение покидает блок GENERATE, счетчик общего числа прошедших через блок сообщений (Nj) увеличивается на единицу. Так как последующее сообщение не генерируется до тех пор, пока предыдущее сообщение не покидает

блок GENERATE, то содержимое счетчика текущего числа находящихся в блоке сообщений (Wj) никогда не превышает 1.

При использовании блока GENERATE необходимо помнить, что *сообщение не должно входить в блок GENERATE*. Если сообщение пытается это сделать, возникает ошибка выполнения.

2.2.4. Функции в GPSS

Формат описания функций

```
NAME FUNCTION A, B, C
```

Имя функции должно записываться в поле метки (NAME) оператора описания FUNCTION. Поле A оператора FUNCTION должно содержать аргумент (независимую переменную) функции. Аргументом может быть любой из стандартных числовых атрибутов, за исключением матрицы ячеек; в качестве аргумента функции может быть использовано и значение любой другой функции. Если в качестве аргумента функции используется случайное число RNj, то значениями аргумента будут числа, равномерно распределенные в интервале [0,1]. Следует отметить, что во всех других случаях использование случайных чисел RNj дает значение в диапазоне [0,999].

Запись в поле B определяет тип и число точек функции (число пар значений X[i] и Y[i]). В табл. 2 перечислены типы функции и приведена мнемоника каждого типа.

Таблица 2 – Мнемоника типов функции

Тип функции	Мнемоника
Непрерывная числовая	C
Дискретная числовая	D
Табличная числовая	L
Дискретная атрибутивная	E
Табличная атрибутивная	M

За каждым оператором описания FUNCTION должны следовать операторы для задания координат (X[i] и Y[i]) функции. Не допускается использование комментариев между оператором описания FUNCTION и операторами, задающими значения функции. Для задания координат можно использовать нецелые числа, например:

```
RLGEX FUNCTION RN1,C5
0,0/.33,.45/.40,1.60/.70,2.75/1.00,3.90
```

При написании операторов, задающих значения координат функции, необходимо придерживаться следующих правил:

1. Запись должна начинаться в позиции 1;

2.Значения координат x[i] и y[i] одной точки функции разделяются запятой;

3.Наборы координат разделяются знаком (/);

4. Координаты x[i] и y[i], относящиеся к одной точке, должны задаваться одним оператором;

5. Каждое последующее значение x[i] должно быть больше предыдущего;

6. Значения y[i] не могут быть матрицами ячеек;

7. Каждая функция должна иметь, по крайней мере, две описанные точки.

Примеры:

```
ABC FUNCTION P3,D4
0,5/1,1002/3,20/4,25
FUN3 FUNCTION RN7,C4
0,0/5,12/.68,15/1.0,20
ACT FUNCTION Q$ALINE,C16
0,0/5,1/10,2/20,3/30,4/40,5/50,6/.../100,11
110,12/120,13/130,14/140,15
```

Непрерывные числовые функции

Когда значение аргумента непрерывной числовой функции попадает в интервал между двумя заданными значениями

(X[i],X[i+1]), программа производит линейную интерполяцию для определения значения функции FN, находящегося в интервале между (Y[i],Y[i+1]). Значения Y[i] хранятся в памяти в виде чисел с плавающей точкой. Интерполяция производится при помощи арифметики с плавающей точкой с нецелыми числами. Однако от конечного результата интерполяции всегда берется целая часть, и значение функции FN всегда является целым числом, за исключением следующих случаев, когда функция используется в качестве:

а) модификатора в блоках ADVANCE, GENERATE, ASSIGN;

б) стандартного числового атрибута в операторе описания FVARIABLE;

в) аргумента другой функции;

Дискретные числовые функции

Дискретные числовые функции Dn задают одно и то же значение функции FN = Y[i] для всех значений аргумента X[i-1] < X ≤ X[i]. Интерполяция не производится, значение функции берется равным значению в правом конце интервала. Как и для непрерывных функций, значения функции задаются следующим образом:

1. FN=Y[1] — для всех значений аргумента J X[1];
2. FN=Y[n] — для всех значений аргумента i X[n].

Нецелые значения функций приводятся к целым путем выделения целой части.

Использование значений функций

Значения функций используются в программах на GPSS/PC в шести основных случаях:

1) как аргументы полей блоков;
2) как аргумент другой функции;
3) как значение Y[i] атрибутивной функции;
4) как аргумент таблицы;
5) как операнд арифметической или булевской переменной;

6) как среднее значение и/или модификатор в блоках GENERATE и ADVANCE или как модификатор в блоке ASSIGN.

При использовании в качестве аргумента блока значение функции может означать следующее:

номер объекта;

номер объекта для логического атрибута (блок GATE);

номер n=1,2,...255 параметра сообщения (блоки ASSIGN, INDEX, LOOP, MARK и SPLIT);

значение стандартного числового атрибута FN[j].

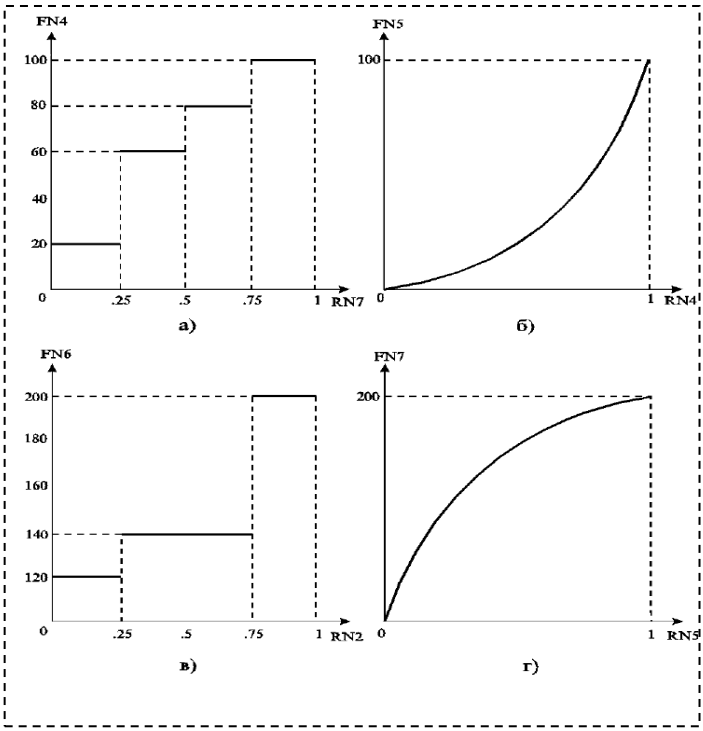

Рис.2. Примеры дискретных (а, в) и непрерывных (б, г) функций

2.2.5. Переменные

Типы переменных

В GPSS/PC имеются три типа переменных:
арифметические переменные (целые);
арифметические переменные с "плавающей точкой";
булевские переменные.

Арифметические переменные (целые

Арифметические переменные представляют собой арифметические сочетания значений различных СЧА, в том числе и других арифметических переменных. Переменная задается одним оператором, называемым оператором описания переменной (оператор VARIABLE), в нем содержится задаваемое арифметическое выражение.

Например, следующий оператор описания VARIABLE определяет арифметическую переменную TEMP:

```
TEMP VARIABLE Q$Queue1+39#FN$Norm
```

При любом обращении к переменной TEMP (употребляется обозначение V$TEMP) её значение вычисляется как текущая длина очереди Queue1 плюс константа 39, умноженная на значение функции Norm.

В арифметических переменных используются пять арифметических операций (табл. 3).

Таблица 3 – Арифметические операции

Знак	Операция
+	Алгебраическое сложение
-	Алгебраическое вычитание
#	Алгебраическое умножение
/	Алгебраическое деление
@	Деление по модулю
^	Возведение в степень
\	Деление нацело

В выражении переменной может быть задано произвольное число перечисленных выше операций в различных сочета-

ниях. Знак результата вычислений определяется по обычным алгебраически правилам. Перед выполнением любой арифметической операции определяется значение каждого элемента и выделяется его целая часть.

Любые стандартные числовые атрибуты, а также библиотечные функции и другие арифметические переменные, могут быть использованы в выражении арифметической переменной. *Запрещается* только *использование самой вычисляемой переменной*. В противном случае происходит ошибка выполнения. Нельзя использовать непосредственно переменные со знаком, т.к. знаки в этом случае рассматриваются как знаки арифметических операций. *Выражение переменной может содержать не более пяти пар скобок*. В выражении, содержащем скобки, прежде всего вычисляется значение группы элементов, ограниченных самой правой из левых скобок.

Формат операторов описания арифметических переменных

Оператор описания арифметической переменной содержит следующие три поля:

1) поле метки, содержащее имя переменной, которое будет использоваться при обращении к этой переменной;

2) поле операции, содержит слово VARIABLE;

3) поле операндов, содержит выражение, в котором содержатся стандартные числовые атрибуты, используемые для вычисления значения переменной.

Пробелы между символами недопустимы. Первый пробел записи считается концом выражения.

Арифметические переменные с плавающей точкой аналогичны рассмотренным выше арифметическим переменным, за исключением того, что все операции над операндами выражений переменных с плавающей точкой выполняются без преобразования операндов и промежуточных результатов к целому виду.

Лишь окончательный результат вычисления преобразуется к целому числу.

Формат операторов описания арифметических переменных с плавающей точкой

Формат операторов описания арифметических переменных с плавающей точкой идентичен описанному выше формату операторов описания арифметических переменных, за исключением того, что в поле операции записывается слово FVARIABLE. Правила написания операторов те же, что и для арифметических переменных. *Арифметическая переменная и переменная с плавающей точкой не могут иметь одинаковые номера*. Если они имеют одинаковые номера, то при вычислении используется более позднее из двух описаний.

Для переменных с плавающей точкой недопустима операция деления по модулю. Только при описании переменных с плавающей точкой допускается применение дробных констант. Стандартный числовой атрибут V используется для обращения, как к арифметическим переменным, так и к переменным с плавающей точкой.

2.2.6. Реализация случайных задержек в блоках модели ВС

Для реализации случайных задержек в схеме используется оператор ADVANCE.

```
ADVANCE A,[B]
```

Блок ADVANCE задерживает продвижение сообщения на заданный период времени. В поле А задается среднее время пребывания сообщения в блоке ADVANCE. Содержимое поля А может быть именем, любым целым числом, в том числе и 0, СЧА или СЧА*«параметр». Если время задержки равно 0, сообщение помещается в список текущих событий перед сообщениями с таким же приоритетом. Сообщения с положительным временем задержки помещаются в список будущих событий. В

поле B указывается способ модификации среднего значения, заданного в поле A. Операнд B может быть именем, положительным целым числом, СЧА или СЧА*«параметр».

Типы модификаторов те же, что и у блока GENERATE:

Интервал изменения среднего времени задержки может быть задан константой, значение которой не должно превосходить среднего времени задержки, вычисленного для данного сообщения. Эта константа определяет интервал, в котором времена задержки распределены равномерно. Все времена задержки выражаются целыми числами. Любое из (2B+1) целых чисел, заключенных в интервале [A-B, A+B], будет выбираться с вероятностью 1/(2B+1).

Дробные значения времени задержки не допускаются, поскольку условное время интерпретатора принимает только целые значения. Константа, определяющая интервал времени задержки, не должна превосходить среднего времени задержки, в противном случае может быть получено отрицательное время задержки в блоке ADVANCE. Отрицательное значение задержки всегда считается ошибкой.

Если в поле B записан модификатор-функция, то вычисленное значение атрибута, заданного в поле A, умножается на значение функции, заданной в поле B. Результат округляется до целого значения и используется как время задержки.

2.2.7. Реализация случайных альтернативных переходов между блоками

Случайные альтернативные переходы транзактов между блоками реализуются блоком TRANSFER, работающем в режиме статистической передачи.

```
TRANSFER [A],[B],[C],[D]
```

Поле A задает режим выбора следующего блока, к которому должно перейти сообщение.

Существуют следующие режимы работы блока TRANSFER: безусловный (пробел); статистический (.); BOTH;

ALL; PICK; функция (FN); параметр (P); подпрограмма (SBR); SIM.

Кроме того, операнд A может быть дробным числом, именем, положительным целым числом, СЧА или СЧА*«параметр».

Поля B и C задают возможные значения номеров следующих блоков или их положение. Использование значений описано при рассмотрении определенных режимов выбора. Операнды могут быть именем, положительным целым числом, СЧА или СЧА* «параметр». Если поле B пусто, Ассемблер записывает в нем номер блока, следующего за блоком TRANSFER.

Если операнд A пропущен, то блок TRANSFER работает в безусловном режиме. Входящее в блок TRANSFER сообщение переходит к блоку, указанному в поле B. Если сообщение в этот блок войти не может, попытка направить сообщение к какому-либо другому блоку не производится.

Когда операнд A не является зарезервированным словом, блок TRANSFER работает в статистическом режиме выбора. Значение аргумента, записанного после точки (.) в поле A, рассматривается как трехзначное число, показывающее (в частях от тысячи), какой процент входящих в блок сообщений следует направить к блоку, указанному в поле C. Остальные сообщения направляются к блоку, указанному в поле B, или к следующему по номеру блоку, если операнд B пропущен. Для каждого сообщения выбирается один из двух возможных вариантов; после того как выбор сделан, второй вариант для этого сообщения не рассматривается. Числовое значение может быть задано при помощи любого стандартного числового атрибута. Если вычисленное значение аргумента меньше или равно нулю, будет происходить безусловная передача сообщений к блоку, указанному в поле B. Если же значение аргумента больше или равно 1000, то будет происходить безусловная передача сообщений к блоку, указанному в поле C.

Пример.

```
PNT TRANSFER .709, MARK2, MARK1
```

Из общего числа сообщений, входящих в блок PNT, в среднем .709 будут пытаться войти в блок MARK2. Остальные .209 будут пытаться войти в блок MARK1.

```
PNT TRANSFER .P1, MARK2, MARK1
```

Трехзначное число, записанное в параметре 1 сообщений, входящих в блок PNT, интерпретируется как вероятность (в частях от тысячи) того, что сообщение попытается войти в блок MARK2. В остальных случаях сообщение попытается войти в блок MARK1.

2.2.8. Табуляция статистической информации

Блок TABLE

Для сбора информации о работе схемы используется табулирование состояния блоков. Оператор описания таблицы TABLE имеет следующий формат:

```
NAME TABLE A,B,C,D
```

Метка NAME определяет имя таблицы. В поле A задается аргумент таблицы — элемент данных, чье частотное распределение будет табулироваться. Операнд может быть именем, целым, СЧА или СЧА*«параметр». В поле B задается верхний предел первого интервала. Операнд может целым или именем. В поле C задается ширина частотного интервала — разница между верхней и нижней границей каждого частотного класса. Операнд может быть положительным целым. В поле D задается число частотных интервалов. Это число не может превышать 8191. Операнд может быть положительным целым.

Для сбора элементов данных сообщение должно войти в блок TABULATE с тем же именем таблицы, что определено в блоке TABLE. Когда сообщение входит в блок TABULATE, оценивается аргумент таблицы (операнд A в операторе TABLE). Если он меньше или равен операнду B в операторе TABLE, то выбирается первый частотный класс таблицы. Если аргумент таблицы не подходит для этого класса, то класс выбирается пу-

тем деления значения аргумента на операнд С оператора TABLE. Нижняя граница частотного класса включается в предыдущий класс. Если таблицы не достаточно для размещения этого значения, то выбирается последний частотный интервал. Затем выбирается целое из частотного класса и счетчик увеличивается на величину, определяемую операндом В оператора TABULATE. По умолчанию увеличение происходит на 1. В конце работы оператора TABULATE изменяются значения среднего и стандартного отклонения аргумента таблицы.

Таблица может быть переопределена или переинициализирована другим оператором TABLE, с той же самой меткой, что и первая. Стандартные числовые атрибуты, связанные с описываемым оператором, следующие:

TB — среднее значение аргумента;

TC — число входов в таблицу;

TD — стандартное отклонение.

Блок, связанный с оператором TABLE — TABULATE.

TABULATE A,[B]

Блок TABULATE табулирует текущее значение заданного аргумента. Способ табуляции зависит от режима работы таблицы, который определяется оператором описания таблицы TABLE. В поле А задается номер или имя таблицы, в которую табулируется значение аргумента. Операнд должен быть именем, положительным целым, СЧА или СЧА*«параметр». Таблица должна быть определена оператором описания TABLE. В поле В задается число единиц, которые должны быть занесены в тот частотный интервал, куда попало значение аргумента. Если поле В пусто, эта величина полагается равной единице. Операнд В может быть именем, положительным целым, СЧА или СЧА*«параметр».

Когда сообщение входит в блок TABULATE, то для нахождения таблицы используется операнд А. Если такой таблицы нет, то возникает ошибка выполнения. Таблица должна

быть определена оператором TABLE. Таблица изменяется в соответствии с операндами оператора TABLE.

Стандартные числовые атрибуты, связанные с описываемым оператором, следующие:

TB — среднее значение аргумента;

TC — число входов в таблицу;

TD — стандартное отклонение.

2.3. Задание второе

По методическому руководству к лабораторной работе и литературным источникам, например [2,5], ознакомиться с порядком работы и командами системы GPSSR/PC.

2.4. Вопросы к домашнему заданию

1. Чем вызвана необходимость применения случайных величин при проведении имитационного моделирования?

2. Каким образом может быть задана случайная нагрузка при моделировании ВС?

3. Какие существуют способы задания операндов при применении операторов ADVANCE и GENERATE?

4. Как задать дискретную и непрерывную функции случайного аргумента?

5. Для каких целей используется табулирование случайных величин?

3. Лабораторное задание и методические указания по его выполнению

Получить у преподавателя вариант задания из таблицы 3. В которой используются следующие обозначения:

Э; M=170 – экспоненциальное распределение со средним значением, равным 170;

Н; M=200, D=30 – нормальное распределение со средним значением, равным 200 и среднеквадратическим отклонением, равным 30;

FNj – распределение в соответствии с функцией FNj;

P; 10...30 – равномерное распределение в интервале от 10 до 30.В крайней правой колонке дана частота перехода транзактов в блок с меткой LAB1. Для используемых функций составить описание операторов FUNCTION. Функции FN5, FN7 (рис. 1 б,г) закодировать 11 точками.

Используя приведенный текст базовой модели, составить программу конкретной модели на языке GPSS, задавая составленное описание функций и выбирая соответствующие операнды блоков GENERATE, ADVANCE, TRANSFER, а также операторов FVARIABLE и TABLE. Затем осуществить прогон модели на компьютере. Получить и проанализировать листинг. При необходимости скорректировать операнды TABLE и выполнить повторный прогон. По счетчикам числа входов в блоки определить процент транзактов, попавших в блоки с именами MARK1 и MARK2, и сопоставить его с операндом A блока TRANSFER.

Таблица 4 – Варианты заданий на лабораторную работу

Ва-ри-ант	Интерва-лы между сообщениями	Задержки				Частота перех. (MET1)
		1	2	3	4	
1	Э; M=110	P;	FN4	P;43..5	FN5	.170
2	H;M=150, D=20	FN5	P;17..2	P;31..4	FN4	.635
3	P;100..200	Э;M=5	FN5	FN4	P;38..4	.290
4	P;100..121	FN4	Э;M=4	FN5	P;	.700
5	FN6	H;M=	P;	P;	FN5	.587
6	FN6	P;26..3	FN4	H;M= D=4	P;26..3	.908
7	Э; M=150	FN5	P;60..8	FN4	P;52..6	.460
8	H; M=120 D=15	P;23..3	FN4	FN5	P;28..4	.821
9	P;105..135	FN4	Э;	P;64..7	FN5	.376
10	P;105..136	P;44..4	FN5	FN5	Э;M=4	.230
11	FN6	FN5	H;M= D=3	P;15..2	P;15..2	.800
12	FN7	P;15..2	FN4	P;	H;M=3 D=5	.390

4. Образец выполнения лабораторной работы

```
 10      simulate
 20 Fun1function    RN1,C24
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,
.915/.7,1.2/.75,1.38/.8,1.6/.84,1.83/.88,2.12/
.9,2.3/.92,2.52/.94,2.81/.95,2.99/.96,3.2/.97,
3.5/.98,3.9/.99,4.6/.995,5.3/.998,6.2/.999,7/.
9998,8
 30 Fun2function    RN4,D4
 .25,20/.5,60/.75,80/1,100
 40 Fun3function    RN5,C11
 0,0/.3,10/.48,20/.6,30/.71,40/.78,50/.84,60/.
 89,70/.93,80/.96,90/1,100
 50 tab1table       M1,20,10,12
 70 Normfvariable   FN$Fun1#15+20
 90      generate    V$Norm
 100     advance     25,2
 110     advance     FN$Fun2
 120     advance     FN$Fun3
 130     advance     34,6
 140     tabulate    tab1
 150     transfer    .777,mark1,mark2
 160     mark1  terminate 1
 170     mark2  terminate 1
 180     start 1000
```

Базовая модель функционирует следующим образом. Транзакты генерируются блоком GENERATE, операнды A и B которого определяют распределение интервалов между моментами поступления транзактов. Далее транзакты последовательно проходят четыре блока ADVANCE, задерживаясь в каждом бло-

ке на время, задаваемое его операндами A и B. Затем производится табулирование общего времени задержки транзактов с помощью СЧА M1 (M1 - время между моментом входа транзакта в модель и моментом использования данного СЧА). Транзакты удаляются из модели через один из двух блоков TERMINATE, определяемый случайным образом с помощью блока TRANSFER, работающего в режиме статистической передачи. Объем выборки исследуемой случайной величины (число значений, попадаемых в таблицу) задается операндом A управляющего оператора START.

Проведём построчный анализ текста программы:

Строка 10 оператор simulate [A] инициирует процесс имитации в течении A минут. Без параметров — бесконечность.

Строка 20 определение непрерывной функции, по 24-м точкам. Функция описывает нормальное распределение

Строка 50 определение статистической таблицы. Табулируется СЧА M1, равная времени прохождения транзакта от момента «рождения» до момента прохождения точки табулирования (транзитное время). Табулируются 12 интервалов начиная с 10 с шагом 10.

Строка 70 определение переменной Norm, описывающей нормальное распределение с дисперсией 15 и математическим ожиданием равным 20.

Строка 90 генерация транзактов с паузами, определяемыми значениями переменной Norm

Строка 100 задержка на равномерно распределённое случайное количество единиц времени со средним значением 25 и максимальным разбросом 2.

Строка 120 задержка на случайное количество единиц времени, описываемое распределением Fun2.

Строка 140 табулирование транзитного времени.

Строка 150 ветвление; с вероятностью 0.777 транзакт попадёт в блок mark2 и с обратной в блок mark1

Строка 160 удаление одного транзакта из модели.Строка 180 инициализация счётчика транзактов. При моделировании будет сгенерировано 1000 сообщений (транзактов).

В результате выполнения программы будет получен файл отчёта. Разберём его.

```
GPSS/PC Report file LR1.(V2,#37349)03-10-1999
21:03:42 page 1

START_TIME END_TIME BLOCKS FACILITIES STORAGES
FREE_MEMORY
    0        34529     8          0          0
355728

LINE LOC  BLOCK_TYPE ENTRY_COUNT CURRENT_COUNT
RETRY
90    1      GENERATE      1004         0
             0
```

Заголовок содержит следующую информацию (слева направо):

Время начала моделирования	0
Время окончания моделирования	34529
Количество блоков в модели	8
Количество устройств	0
Количество модулей памяти	0
Объём свободной памяти	355728

```
100    2      ADVANCE       1004          1
              0
110    3      ADVANCE       1003          1
              0
120    4      ADVANCE        1002         2
              0
130    5      ADVANCE       1000          0
              0
140    6      TABULATE      1000          0
              0
150    7      TRANSFER      1000          0
              0
160    MARK1  TERMINATE     228           0
              0
170    MARK2  TERMINATE     772           0
              0
```

Первая колонка — номер строки программы. Вторая — метка, если есть, иначе номер строки после компиляции. В четвёртой строке фигурируют названия блоков, соответствующие строкам. Колонка entry_count показывает количество вхождений транзактов в данную строку. Current_count — количество вхождений в текущий момент. Retry — количество возвратов.

```
TABLE   MEAN   STD.DEV.   RETRY   RANGE   FREQUENCY
          CUM.%
TAB1   154.16 39.55          0
                             70  - 80      24
          2.40
                             80  - 90      50
          7.40
                             90  - 100     44
          11.80
                            100  - 110     31
          14.90
                            110  - 120     51
          20.00
```

```
                          120  -         800
         100.0
XACT_GROUP          GROUP_SIZE           RETRY
POSITION                  0              0
```

Таблица TAB1.

Поле в mean фигурирует среднее значение параметра M1: 154.16. Иначе говоря, среднее транзитное время равно 154.16 единиц.

Поле STD.DEV. — standart deviation — взвешенное среднеквадратическое отклонение. --- (привести формулу)

Столбцы retry range и frequency содержат информацию о диапазоне и количестве попаданий в него. Обратите внимание на то, что не все значения попали **в** диапазон измерений таблицы. Об этом говорит последняя строчка, означающая интервал [120,---бесконечность].

Столбец CUM% содержит накопительный процент.

5. Указания по оформлению отчета и контрольные вопросы по выполненной работе

5.1. Отчет должен содержать

1.Описание шагов составления имитационной модели.

2. Перечень основных команд системы GPSSR/PC, используемых для редактирования и моделирования исходного описания.

3.Исходный текст модели и результаты моделирования.

4.Основные выводы по результатам моделирования.

5.2. Контрольные вопросы к лабораторной работе

1. Что понимается под «случайной нагрузкой»?

2. Какие существуют способы задания случайной нагрузки?

3. Какие существуют распределения случайных величин?

ЛАБОРАТОРНАЯ РАБОТА № 2
МОДЕЛИРОВАНИЕ КОНВЕЙЕРНЫХ ВЫЧИСЛИТЕЛЬНЫХ СТРУКТУР

1. Общие указания по выполнению лабораторной работы

1.1. Цель работы

Приобретение навыков построения и исследования имитационных моделей конвейерных вычислительных структур на языке GPSS

1.2. Используемое оборудование и программное обеспечение

Интерпретатор GPSS WORLD, персональный компьютер, совместимый с IBM PC с объемом оперативной памяти не менее 512 Мб, операционная система Windows 7 и выше.

1.3. План выполнения лабораторной работы

В результате выполнения лабораторной работы должны быть приобретены знания возможностей языка GPSS по моделированию конвейерных вычислительных структур; способов представления обслуживающих приборов и многоканальных устройств на языке GPSS; методов организации очередей к приборам и многоканальным устройствам.

В процессе выполнения лабораторной работы необходимо овладеть навыками описания операторов обслуживающих приборов и многоканальных устройств на языке GPSS; трансляции моделей, составленных на языке GPSS и формирование файла результатов; интерпретации результатов моделирования.

При выполнении лабораторной работы необходимо изучить операторы задания обслуживающих приборов и многоканальных устройств на языке GPSS; изучить операторы описания очередей к приборам; составить на языке GPSS модель по заданию преподавателя; оттранслировать описание модели, устра-

нить ошибки, если они есть, и получить файл выходной статистики; провести анализ результатов.

2. Домашние задания и методические указания по их выполнению

2.1. Задание первое

По методическому руководству к лабораторной работе и литературным источникам, например [1,4,6], ознакомиться с вычислительными системами, имеющими конвейерную организацию.

2.2. Методические указания по выполнению первого задания

Имеется вычислительная система с конвейерной структурой, состоящей из трех процессоров П1, П2, П3 (рис. 3). На вход системы поступает поток сообщений, которые последовательно обрабатываются в каждом из процессоров. На входе каждого процессора имеется буфер (Б1, Б2, Б3), необходимый для приема сообщений и предотвращения их потери при пиковых нагрузках. Сообщения в буфере образуют очередь и обрабатываются соответствующим процессором в порядке их поступления. Обработанные сообщения образуют выходной поток сообщений

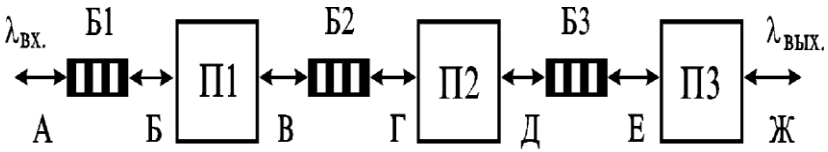

Рис. 3. Вычислительная система с конвейерной структурой

При исследовании конвейерных вычислительных структур возникают задачи определения общего времени обработки сообщений и выполнения некоторых этапов обработки, интен-

сивности выходного потока сообщений и потоков в определенных точках системы (Б, В, Г, Д, Е, Ж), коэффициентов загрузки процессоров и необходимых объемов буферов.

2.3. Задание второе

По методическому руководству к лабораторной работе и литературным источникам, например [2,3,5], ознакомиться с назначением многоканальных устройств и обслуживающих приборов, а также методами организации очередей в моделях, реализованных на языке GPSS.

2.4. Методические указания по выполнению второго задания

Каждый процессор системы можно моделировать прибором, которому соответствует пара блоков: SEIZE A занять прибор с именем A и RELEASE A — освободить прибор с именем A. Имя прибора может быть числовым или символическим. Собственно процесс обслуживания транзактов прибором моделируется блоком ADVANCE, операнды которого определяют время обработки. Таким образом, каждый из процессоров может быть описан тремя блоками:

```
SEIZE PROC; ЗАНЯТЬ ПРИБОР
ADVANCE A,B;     ЗАДЕРЖАТЬ ТРАНЗАКТ
RELEASE PROC;    ОСВОБОДИТЬ ПРИБОР
```

Необходимо помнить, что в приборе одновременно может обслуживаться только один транзакт. Если транзакт пытается занять прибор (т.е. войти в блок SEIZE), но прибор занят обслуживанием другого транзакта, то вновь прибывший транзакт встает в очередь к прибору. В очереди может одновременно находиться множество транзактов. Когда прибор освобождается, то его занимает транзакт, «стоящий» в очереди первым. В общем случае факт наличия или отсутствия очереди определяется

соотношением интенсивностей входного потока транзактов и потока обслуженных транзактов, а также распределением этих потоков. Для присвоения очереди имени (например, для того, чтобы использовать в модели какой-либо СЧА очереди), а также для организации распечатки стандартной статистики по очередям (средняя и максимальная длина очереди, среднее время пребывания транзактов в очереди и т.д.) используют блоки QUEUE и DEPART.

Блок QUEUE

Формат записи блока QUEUE:

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

Блок QUEUE увеличивает длину очереди. В поле А задается номер или имя очереди, к длине которой добавляются единицы. Операнд может быть именем, положительным целым, СЧА или СЧА*«параметр». Поле В определяет число единиц, на которое увеличивается текущая длина очереди. Если поле В пусто, то прибавляется единица. Операнд может быть именем, положительным целым, СЧА или СЧА*«параметр». Когда сообщение входит в блок QUEUE, то ищется очередь с именем, определенным операндом А. Если необходимо, очередь создается. Поскольку к очереди добавляются единицы, а не сами сообщения, не составляется список членов очереди. Сообщения в этот же момент условного времени пытаются перейти к следующему блоку.

Поскольку очередь обычно используется для измерения времени ожидания, за блоком QUEUE обычно следуют такие блоки как, SEIZE, PREEMPT или ENTER, которые могут задержать сообщение. К таким блокам относятся также блоки GATE и TEST, работающие в режиме условного входа.

Одно и то же сообщение может одновременно увеличить длину нескольких очередей, т.е. сообщение может войти в не-

сколько блоков QUEUE перед тем, как войти в соответствующие блоки DEPART.

Стандартные числовые атрибуты, связанные с описываемым оператором, следующие:

Q — текущая длина очереди;
QA — средняя длина очереди;
QC — общее число входов в очередь;
QM — максимальная длина очереди;
QT — среднее время ожидания в очереди;
QX — среднее время ожидания в очереди для сообщений с ненулевым временем задержки;
QZ — число входов в очередь с нулевым временем задержки.

При входе сообщения в блок QUEUE текущая длина очереди j увеличивается на величину, указанную в поле B. Новое значение длины очереди Qj сравнивается с максимальным значением QMj. Если новое значение больше, то это значение становится новым максимальным значением длины очереди. Общее число входов в очередь также увеличивается на величину, указанную в поле B. Поэтому число сообщений, входящих в блок QUEUE, будет всегда меньше или равно общему числу входов (QCj) в блок QUEUE.

Для того, чтобы обеспечить правильный сбор статистики в случае, когда сообщение одновременно входит в несколько очередей, а также в случае, когда сообщение входит в блок ADVANCE, будучи при этом членом одной или нескольких очередей, фиксируются номера очередей, куда поступает сообщение. При выходе из очереди в блоке DEPART сообщение не обязательно должно уменьшать длину очереди на ту же величину, на которую оно увеличило ее при входе в блок QUEUE. Но в конечном счете число входов в очередь должно быть равно числу выходов из очереди.

Блок DEPART

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

Блок DEPART служит для уменьшения длины очереди. В поле A задается номер или имя очереди, длину которой нужно уменьшить. Операнд может быть именем, положительным целым, СЧА или СЧА*«параметр». В поле B задается число единиц, на которое уменьшается длина очереди. Это число не должно превышать текущую длину очереди. Если поле B пусто, длина очереди уменьшается на единицу. Операнд может быть именем, положительным целым, СЧА или СЧА*«параметр».

Всякий раз, когда сообщение входит в блок DEPART, текущая длина Qj очереди, номер j которой указан в поле A, уменьшается на число единиц, заданное в поле B. Общее число входов в очередь при входе сообщения в блок DEPART не изменяется. Если длина очереди становится отрицательной, то возникает ошибка выполнения.

При использовании данной пары блоков рассмотренное выше описание процессора дополняется следующим образом:

```
QUEUE QQQ1; ВСТАТЬ В ОЧЕРЕДЬ
SEIZE PROC; ЗАНЯТЬ ПРИБОР DEPART
DEPART QQQ1; ПОКИНУТЬ ОЧЕРЕДЬ
ADVANCE A,B; ЗАДЕРЖАТЬ ТРАНЗАКТ
RELEASE PROC; ОСВОБОДИТЬ ПРИБОР
```

Отметим, что очередь в данном случае выполняет функции входного буфера и может его моделировать. При этом о необходимом объеме буфера можно судить по максимальной длине очереди, которая наблюдалась за время моделирования.

Буфер может быть также промоделирован многоканальным устройством (МКУ), которому соответствуют блоки ENTER — войти в МКУ и LEAVE — покинуть МКУ. МКУ представляет собой несколько параллельно работающих прибо-

ров (каналов), число которых (емкость МКУ) определяется оператором STORAGE. Например, оператор

```
BUF STORAGE 10
```

задает емкость МКУ с именем BUF, равную 10. Для задания бесконечной емкости МКУ оператор STORAGE используется без операндов.

Блок ENTER

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

В поле А указывается номер МКУ или его имя.

Поле В содержит число занимаемых единиц МКУ. По умолчанию 1.

Процессор с входным буфером в виде МКУ может быть описан следующей последовательностью блоков:

```
ENTER BUF; ВОЙТИ В МКУ
SEIZE PROC; ЗАНЯТЬ ПРИБОР
LEAVE BUF; ПОКИНУТЬ МКУ
ADVANCE A,B; ЗАДЕРЖАТЬ ТРАНЗАКТ
RELEASE PROC; ОСВОБОДИТЬ ПРИБОР
```

Транзакт может войти в МКУ, если хотя бы один канал свободен. В конце моделирования распечатывается стандартная статистика по МКУ, включающая среднее и максимальное число занимавшихся каналов, коэффициент загрузки МКУ и т.д.

Блок LEAVE

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

Освободить В единиц из устройства А.

Исследование конвейерной системы

При исследовании модели конвейерной вычислительной системы часто возникает необходимость собирать следующую статистику:

1. Статистика о распределении интервалов между моментами входа транзактов в модель и моментами прибытия транзактов в некоторую точку N модели (резидентное время транзактов). Блок TABULATE должен быть помещен в точку N модели, а аргументом таблицы должен являться СЧА M1.

2. Статистика о распределении интервалов между моментами прохождения транзактами точек S и T модели (транзитное время транзактов). В точку S модели помещается блок MARK j, а в точку T - блок TABULATE, причем аргументом таблицы должен быть СЧА MPj.

3. Статистика о распределении времени пребывания транзактов в очереди. Блок TABULATE помещается непосредственно после блока DEPART, относящегося к данной очереди. В поле операции оператора определения таблицы записывается ключевое слово TABLE (QTABLE — режим использования таблицы). Аргументом таблицы должно являться имя очереди.

2.5. Вопросы к домашнему заданию

1. В чем заключается особенность конвейерных вычислительных систем?

2. Каковы достоинства и недостатки конвейерных вычислительных систем?

3. В чем заключается различие между прибором и МКУ?

4. При каких условиях на входе прибора или МКУ образуется очередь?

5.Чем определяется факт наличия очереди?

3. Лабораторное задание и методические указания по его выполнению

1. Ознакомиться с описанием работы и получить у преподавателя вариант задания. В табл.3 указаны точки модели, в которых необходимо контролировать перечисленные характеристики, обозначенные следующим образом: X1 — резидентное время транзактов; X2 — интервалы между прибытиями транзактов; X3 — время пребывания транзактов в очереди; X4 — транзитное время транзактов. Буфер необходимо моделировать одним из двух указанных способов: Y1 — многоканальным устройством, Y2 — очередью.

2. Составить программу модели на языке GPSS.

3. Выполнить прогон модели на ПЭВМ.

Таблица 5 – Вариант заданий

Вариант	Выходная характеристика	Вариант	Выходная характеристика
1	Ж, X1, У1	7	Д-Е, X3, У1
2	Д, X1, У2	8	В-Г, X3, У2
3	В, X1,У1	9	А-В, X3,У1
4	Ж, X2, У2	10	В-Д, X4,У2
5	Д, X2, У1	11	В-Ж, X4, У1
6	В, X2,У2	12	Г-Е, X4, У2

4. Получить и проанализировать листинг. При необходимости скорректировать операнды операторов TABLE и QTABLE и выполнить повторный прогон.

5. Определить коэффициенты загрузки процессоров и необходимый объем буферов. Сделать вывод относительно режима работы моделируемой системы.

6. Оформить отчет.

4. Указания по оформлению отчета и контрольные вопросы по выполненной работе

4.1. Отчет должен содержать

1. Описание шагов составления имитационной модели.
2. Блок-схему имитационной модели.
3. Исходный текст модели и результаты моделирования.
4. Основные выводы по результатам моделирования.

4.2. Контрольные вопросы к лабораторной работе

1. Каково назначение блоков QUEUE и DEPART?
2. Что включает в себя стандартная статистика по приборам, МКУ и очередям? Как организовать ее вывод?
3. Как собрать статистику о резидентном и транзитном времени транзактов?

ЛАБОРАТОРНАЯ РАБОТА № 3
МОДЕЛИРОВАНИЕ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛИТЕЛЬНЫХ СТРУКТУР С ПРИОРИТЕТНОЙ ДИСЦИПЛИНОЙ ОБСЛУЖИВАНИЯ ЗАЯВОК

1. Общие указания по выполнению лабораторной работы

1.1. Цель работы

- приобретение навыков построения и исследования имитационных моделей параллельных вычислительных структур с приоритетной дисциплиной обслуживания заявок на языке GPSS;

- приобретение знания: возможностей языка GPSS по моделированию параллельных вычислительных структур; особенностей языка GPSS по присвоению приоритетов обслуживаемым заявкам; способов захвата обслуживающих приборов транзактами с высоким приоритетом; дополнительных методов организации случайного альтернативного ветвления транзактов в модели;

- овладение навыками: описания операторов изменения приоритета транзактов; использования оператора захвата обслуживающих устройств транзактами; дополнительного применения операторов альтернативного ветвления; интерпретации результатов моделирования.

1.2. Используемое оборудование

Интерпретатор GPSSWORLD, персональный компьютер, совместимый с IBM PC с объемом оперативной памяти не менее 512 Мб, операционная система Windows 7 и выше.

1.3. План выполнения лабораторной работы

При выполнении лабораторной работы необходимо изучить операторы захвата и освобождения обслуживающих приборов в языке GPSS; изучить оператор изменения приоритета транзактов; составить на языке GPSS модель по заданию препо-

давателя; оттранслировать описание модели, устранить ошибки, если они есть, и получить файл выходной статистики; провести анализ результатов.

2. Домашние задания и методические указания по их выполнению

2.1. Задание первое

По методическому руководству к лабораторной работе и литературным источникам, например [1,4,6], ознакомиться с вычислительными системами, имеющими параллельную организацию с приоритетной дисциплиной обслуживания заявок.

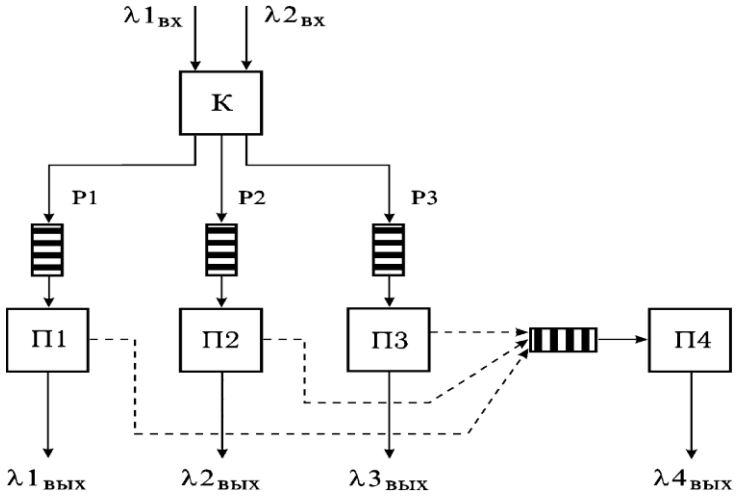

Рис. 4. Вычислительная система параллельного типа

2.2. Методические указания по выполнению первого задания

Имеется вычислительная система параллельного типа, процессоры П1, П2, П3 которой функционируют параллельно и независимо друг от друга (рисунок 3).

На вход системы поступают два потока сообщений А1 и А2. Каждое сообщение содержит признак, указывающий, какой из трех процессоров — П1, П2 или П3 должен его обрабатывать. Признаки являются случайными величинами, характеризующимися вероятностями соответственно Р1, Р2, Р3. Распределительное устройство К анализирует признаки и распределяет сообщения по процессорам. Сообщения потока А1 имеют более высокий приоритет и могут прервать обработку сообщений потока А2 для собственного обслуживания. Дообслуживание прерванного сообщения может происходить либо в том же процессоре, в котором началась его обработка, по окончании прерывания, либо в специальном процессоре П4 сразу же при возникновении прерывания. На входе каждого процессора имеется буфер емкостью S. Если сообщение поступает при заполненном буфере, то оно теряется. Необходимо построить GPSS-модель данной системы и определить такие характеристики, как время пребывания заявок в системе, интенсивности входных потоков, среднее время дообслуживания прерванных заявок, а также процент потерянных из-за ограниченного объема буфера заявок.

2.3. Задание второе

По методическому руководству к лабораторной работе и литературным источникам, например [2,3,5], ознакомиться с назначением операторов захвата/возврата обслуживающих приборов, а также методами изменения приоритетов обслуживания транзактов в моделях, реализованных на языке GPSS.

2.4. Методические указания по выполнению второго задания

Уровень приоритета транзактов может быть задан с помощью операнда E блока GENERATE или путем использования специального блока PRIORITY A, присваивающего приоритету транзакта значение операнда A. Для реализации прерывания обслуживания транзактов в модели необходимо вместо блоков SEIZE и RELEASE использовать соответственно блоки

```
PREEMPT A,B,C,D,E
```

и

```
RETURN A
```

Захватить прибор и возвратить прибор. Здесь A — имя захватываемого (возвращаемого) прибора; B — если данный операнд не используется, то захват возникает в том случае, когда обслуживаемый транзакт сам не является захватчиком; если в поле данного операнда записана аббревиатура PR, то захват возникает в том случае, когда возможный захватчик имеет приоритет выше, чем приоритет обслуживаемого транзакта; C — имя блока, в который будет послан прерванный транзакт; D — номер параметра прерванного транзакта, в который помещается значение времени, оставшееся транзакту до окончания обслуживания в приборе; E — если операнд не используется, то транзакт сохраняет право автоматического восстановления в приборе; если в поле данного операнда записана аббревиатура RE, то транзакт теряет такое право. Операнды B, C, D, E — необязательные. Операнды D и E используются только при условии использования операнда C. Каждый из параллельно используемых процессоров (П1,П2,П3) может быть описан отдельной тройкой блоков: SEIZE-ADVANCE-RELEASE, или PREEMPT-ADVANCE-RETURN. Для сокращения текста модели можно воспользоваться способом косвенного задания номера прибора, например:

```
SEIZE *1;   ЗАНЯТЬ ПРИБОР
```

```
ADVANCE;     ЗАДЕРЖАТЬ ТРАНЗАКТ
RELEASE*1;   ОСВОБОДИТЬ ПРИБОР
... ...
PREEMPT*1;   ЗАХВАТИТЬ ПРИБОР
ADVANCE;     ЗАДЕРЖАТЬ ТРАНЗАКТ
RETURN*1;    ВОЗВРАТИТЬ ПРИБОР
```

В данном примере номер занимаемого (захватываемого) и освобождаемого (возвращаемого) прибора определяется значением 1-го параметра транзакта, входящего в соответствующий блок. При этом количество параллельно работающих приборов, описанных одной тройкой блоков, определятся числом значений указанного параметра транзакта. Для реализации такого способа задания номера прибора необходимо использовать блок

```
ASSIGN A,B,
```

осуществляющий присвоение значения B параметру с номером A транзакта. Для осуществления разветвлений в модели помимо рассмотренного ранее блока TRANSFER, работающего в режиме статистической передачи, могут использоваться следующие средства.

1. Блок TRANSFER в режиме безусловной передачи:

```
TRANSFER ,B
```

где B — имя блока, в который безусловным образом передается транзакт. Операнд A не используется. На месте операнда B может стоять СЧА функции, или ПЗ должен его обрабатывать. Признаки являются случайными величинами,

TRANSFER ,FN1.

При этом функция может быть определена так:

```
1 FUNCTION P1,D3
1,MET1/2,MET2/3,MET3
```

В данном примере при P1=1 осуществляется переход в блок с именем MET1, при P1=2 — с именем MET2, при P1=3 — с именем MET3.

2. Блок TRANSFER в режиме BOTH:

```
TRANSFER A,B,C
```

где A — слово BOTH, B и C — имена блоков. Если операнд B отсутствует, то предполагается следующий по порядку блок. Транзакт переходит в тот блок, который способен принять его первым. Попытка перехода начинается с блока B. Блок TRANSFER в данном режиме обычно непосредственно предшествует блокам SEIZE, PREEMPT, ENTER, а также другим блокам, работающим в режиме отказа.

3. Блок TEST в режиме условной передачи:

```
TEST X A,B,C
```

где A,B — имена соответственно 1-го и 2-го СЧА;

X — оператор отношения (G: A>B; GE: A >= B; E: A=B; NE: A <> B; LE: A <= B; L: A<B); C — имя блока, в который переходит транзакт при невыполнении проверяемого условия. Для построения модели можно воспользоваться следующим методом. Транзакты поступают в модель через два блока GENERATE, определяющих уровень их приоритета. Далее два потока объединяются в один. С помощью двух блоков TRANSFER, работающих в режиме статистической передачи, производится расщепление общего потока на три потока с заданной вероятностью для присвоения соответствующих значений первому параметру транзактов с помощью оператора ASSIGN. Далее вновь производится объединение трех потоков в один, который направляется в группу блоков, имитирующих параллельную работу трех приборов при косвенном задании их номеров. Буфер имитируется МКУ с косвенным заданием номера. Перед входом в МКУ должна производиться проверка его занятости. Время обслуживания транзактов приборами может

задаваться тремя блоками ADVANCE, переход в которые производится по функции. Для выделения транзактов, направляемых в блок TABULATE, производится проверка значения их первого параметра.

Для сбора сведений о работе системы могут понадобиться следующие СЧА:

STj – среднее время пребывания сообщений в МУ j ;

SCj – общее число входов в МУ j;

FTj – среднее время использования устройства одним сообщением;

FCj – общее число входов в устройство j.

2.5. Вопросы к домашнему заданию

1. В чем заключается особенность параллельных вычислительных систем?

2. Каковы достоинства и недостатки параллельных вычислительных систем?

3. Чем характерен режим захвата приборов и МКУ и в каких случаях он необходим?

4. При каких условиях используется режим работы с отказами приборов и МКУ?

3. Лабораторное задание и методические указания по его выполнению

1. Ознакомиться с описанием работы и получить у преподавателя вариант задания. В табл. 4 использованы следующие обозначения: P1,P2,P3 — вероятность обслуживания транзактов соответствующим прибором; S - емкость буфера на входах процессоров; П - дообслуживание после прерывания в том же приборе, в котором началось обслуживание; Д - дообслуживание в процессоре П4.

В столбце «Выходные характеристики» буквами А...Л обозначены характеристики, которые необходимо определить: А - время дообслуживания прерванных транзактов; Б - время пре-

бывания в системе заявок, обслуженных процессором П4; В — время пребывания в системе всех заявок; Г — время пребывания в системе заявок, обслуженных процессорами П1 и П2; Д — время пребывания в системе заявок, обслуженных процессорами П1 и П3; Е — время пребывания в системе заявок, обслуженных процессорами П2 и П3; Ж — интервалы между сообщениями на выходе П1; З — интервалы между сообщениями на выходе П2; И — интервалы между сообщениями на выходе П3; К — процент потерянных заявок по каждому из процессоров; Л — общий процент потерянных заявок.

2. Составить программу на языке GPSS.

3. Выполнить прогон модели на ПЭВМ.

4. Получить и проанализировать листинг. Для характеристик А...И определить среднее значение, среднеквадратическое отклонение и их распределение. Характеристики К, Л контролировать по счетчикам числа входов транзактов в блоки.

5. Определить коэффициенты загрузки процессоров. Сделать вывод относительно режима работы моделируемой системы.

6. Оформить отчет.

Таблица 6 – Варианты заданий

Вариант	P1	P2	P3	S	Дообслуживание	Вых. характеристики
1	0.5	0.3	0.2	5	П	К,В,Ж
2	0.6	0.2	0.2	4	Д	К,А,Б
3	0.4	0.2	0.4	3	П	К,Г,И
4	0.3	0.2	0.5	2	Д	К,А,Г
5	0.2	0.5	0.3	5	П	К,Д,З
6	0.7	0.2	0.1	4	Д	К,А,Д
7	0.5	0.2	0.3	3	П	Л,Е,Ж
8	0.6	0.1	0.3	2	Д	Л,А,Б
9	0.4	0.4	0.2	5	П	Л,В,З
10	0.3	0.5	0.2	4	Д	Л,А,Е
11	0.2	0.3	0.5	3	П	Л,Г,И
12	0.7	0.1	0.2	2	Д	Л,А,Б

4. Указания по оформлению отчета и контрольные вопросы по выполненной работе

4.1. Отчет должен содержать

1. Описание шагов составления имитационной модели.
2. Блок-схему имитационной модели.
3. Исходный текст модели и результаты моделирования.
4. Основные выводы по результатам моделирования.

4.2. Контрольные вопросы к лабораторной работе.

1. Чем отличается захват прибора от занятия прибора?
2. Как моделируются параллельно работающие приборы?
3. С помощью каких блоков можно организовать разветвления в моделях?
4. Как моделируется буфер ограниченного объема?

ЛАБОРАТОРНАЯ РАБОТА № 4
МОДЕЛИРОВАНИЕ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛИТЕЛЬНЫХ СТРУКТУР С ОБЩИМИ РЕСУРСАМИ

1. Общие указания по выполнению лабораторной работы

1.1. Цель работы

- приобретение навыков построения и исследования GPSS-моделей параллельных вычислительных структур с общими ресурсами и изучение методов предотвращения тупиковых ситуаций, возникающих при функционировании таких систем.

1.2. Используемое оборудование

Интерпретатор GPSS WORLD, персональный компьютер, совместимый с IBM PC с объемом оперативной памяти не менее 512 Мб, операционная система Windows 7 и выше.

1.3. При выполнении лабораторной работы необходимо

- в результате выполнения лабораторной работы должны быть приобретены знания: возможностей языка GPSS по моделированию общих ресурсов, используемых вычислительными структурами; способов преодоления тупиковых ситуаций, возникающих при использовании общих ресурсов в моделируемой системе; особенностей языка GPSS по использованию сохраняемых величин и булевых переменных; дополнительных методов организации случайного альтернативного ветвления транзактов в модели с использованием логических ключей.

- в процессе выполнения лабораторной работы необходимо овладеть навыками: описания общих ресурсов в моделируемых вычислительных системах; использования операторов описания логических ключей и переменных; применения сохраняемых величин в моделях.

- при выполнении лабораторной работы необходимо изучить операторы описания и использования логических ключей,

переменных и сохраняемых величин в языке GPSS; составить на языке GPSS модель по заданию преподавателя; оттранслировать описание модели, устранить ошибки, если они есть, и получить файл выходной статистики; провести анализ результатов.

2. Домашние задания и методические указания по их выполнению

2.1. Задание первое

По методическому руководству к лабораторной работе и литературным источникам, например [1,4,6], ознакомиться с вычислительными системами, имеющими параллельную организацию с общими ресурсами.

2.2. Методические указания по выполнению первого задания

Имеется многомашинная вычислительная система, состоящая из нескольких ЭВМ (ЭВМ1 — ЭВМn), взаимодействующих между собой через системную общую шину ОШ (рис. 5).

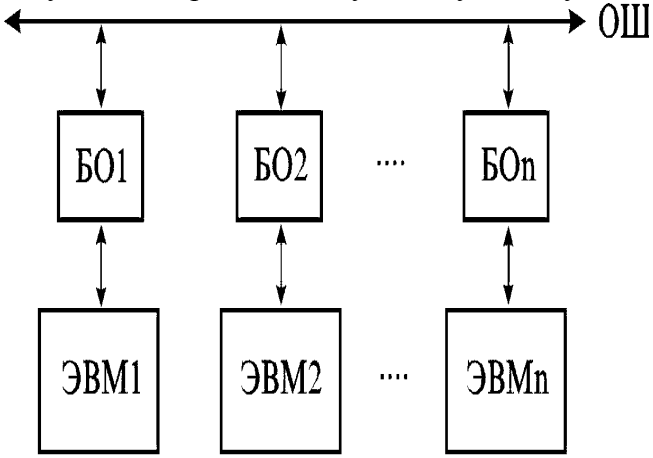

Рис. 5. Многомашинная вычислительная система

В процессе работы ЭВМ обмениваются сообщениями. Каждая ЭВМ имеет свой блок обмена (БО), предназначенный для управления передачей и приема сообщений. При передаче сообщения, например, от ЭВМ1 для ЭВМ2, процесс, управляющий передачей, поочередно занимает ресурсы БО1, ОШ, БО2, собственно передает сообщение и освобождает занятые им ресурсы в обратном порядке. При наличии встречной передачи (от ЭВМ2 для ЭВМ1) могут возникнуть следующие ситуации: а) для передачи от ЭВМ1 к ЭВМ2 заняты БО1 и ОШ, а для передачи от ЭВМ2 к ЭВМ1 — БО2; б) для передачи от ЭВМ1 к ЭВМ2 занят БО1, а для передачи от ЭВМ2 к ЭВМ1 — БО2 и ОШ.

При наличии одной из таких ситуаций продолжение встречных передач невозможно, так как процессы, управляющие передачами, заблокировали друг друга. В таких случаях говорят, что в системе возникла тупиковая ситуация. Причиной тупика, как правило, является то, что несколько общих ресурсов занимаются двумя или более процессами поочередно и в обратном порядке. Для предотвращения тупиковых ситуаций необходимо введение специальных мер, моделированию которых и посвящена данная лабораторная работа.

2.3. Задание второе

По методическому руководству к лабораторной работе и литературным источникам, например [2,3,5], ознакомиться с назначением и способами применения сохраняемых величин и булевых переменных, используемых в языке GPSS.

2.4. Методические указания по выполнению второго задания

Базовая модель процесса обмена сообщениями между двумя ЭВМ рассмотренной системы приведена ниже. Модель состоит из трех сегментов. Сегменты 1 и 2 имитируют встречные передачи между ЭВМ1 и ЭВМ2. Блоки GENERATE имитируют потоки сообщений, поступающих от ЭВМ к соответству-

ющему БО и предназначенных для передачи другой ЭВМ. Каждому блоку SEIZE, моделирующему занятие определенного ресурса, предшествует блок ADVANCE, наличие которого отражает тот факт, что занятие ресурса требует времени. Завершение моделирования, реализуемое сегментом 3, может иметь место в следующих случаях:

1) если текущее содержимое одного из блоков с именем MET1, MET2, MET3 (т.е. блоков, непосредственно предшествующих блокам SEIZE сегмента становится равным 30;

2) если суммарное число транзактов, вошедших в блоки TERMINATE с именами MET4 и MET5, становится равным 500.

Базовая модель процесса обмена сообщениями между двумя ЭВМ

```
          SIMULATE
          ...
VAR1 VARIABLE N$MET4+N$MET5
BVAR1 BVARIABLE
(W$MET1'GE'30)+(W$MET2'GE'30)+
(W$MET3'GE'30)+(V$VAR1'GE'500)
* СЕГМЕНТ МОДЕЛИ 1 (ПЕРЕДАЧА ОТ ЭВМ1 К ЭВМ2)
          GENERATE 200,20,1
MET1      ADVANCE  5,3
          SEIZE  BUF1; ЗАНЯТЬ БО1
MET2      ADVANCE 5,3
          SEIZE BUS; ЗАНЯТЬ ОШ
MET3      ADVANCE 5,3
          SEIZE BUF2; ЗАНЯТЬ БО2
          ADVANCE 30,10; ПЕРЕДАТЬ СООБЩЕНИЕ
          RELEASE BUF2; ОСВОБОДИТЬ БО2
          RELEASE BUS; ОСВОБОДИТЬ ОШ
          RELEASE BUF1; ОСВОБОДИТЬ БО1
MET4      TERMINATE
```

```
* СЕГМЕНТ МОДЕЛИ 2 (ПЕРЕДАЧА ОТ ЭВМ2 К ЭВМ1)
        GENERATE 200,30,2
        ADVANCE  5,3
        SEIZE BUF2; ЗАНЯТЬ БО2
        ADVANCE 5,3
        SEIZE BUS; ЗАНЯТЬ ОШ
        ADVANCE 5,3
        SEIZE BUF1; ЗАНЯТЬ БО1
        ADVANCE 30,10; ПЕРЕДАТЬ СООБЩЕНИЕ
        RELEASE BUF1; ОСВОБОДИТЬ БО1
        RELEASE BUS; ОСВОБОДИТЬ ОШ
        RELEASE BUF2; ОСВОБОДИТЬ БО2
MET5    TERMINATE
* СЕГМЕНТ МОДЕЛИ 3 (ЗАВЕРШЕНИЕ МОДЕЛИРОВАНИЯ)
        GENERATE,,,1
        TEST E BV$BVAR1,1
        TERMINATE 1
* УПРАВЛЯЮЩИЕ ОПЕРАТОРЫ
```

Первый случай отражает тупиковое состояние модели, так как при тупике в сегментах 1 и 2 имеется по одному блоку SEIZE в которые транзакты принципиально не могут войти. При этом транзакты накапливаются в блоках ADVANCE, непосредственно предшествующих указанным блокам SEIZE. Второй случай соответствует нормальному завершению моделирования после выполнения передачи 500 сообщений. О причине окончания моделирования можно судить по значениям счетчиков блоков, которые распечатываются в качестве стандартной статистики. Следует отметить, что рассмотренный способ установления факта наличия тупика можно использовать лишь в том случае, когда у программиста есть уверенность, что большая длина очереди к прибору не вызвана чрезмерной интенсивностью входного потока. При прогоне базовой модели неминуемо возникает

тупиковая ситуация, которая заключается в том, что ни один транзакт не сможет войти в блок SEIZE BUF2 сегмента 1 и в блок SEIZE BUS сегмента 2, причем приборы BUF2 и BUS никогда не будут освобождены.

Для предотвращения тупиков могут быть использованы следующие методы.

1. Занятие общих ресурсов начинается в тот момент, когда все они свободны, причем все ресурсы занимаются одновременно. Для реализации данного метода в модели могут быть использованы следующие средства языка GPSS:

а) блок TRANSFER, работающий в режиме SIM:

```
TRANSFER SIM,B,C
```

здесь B — имя блока, куда переходит транзакт при выполнении всех проверяемых условий; C — имя блока, в который переходит транзакт при невыполнении хотя бы одного из условий. Операнды B и C необязательные. Проверка условий осуществляется с помощью ряда блоков GATE или TEST, работающих в режиме отказа и непосредственно предшествующих блоку TRANSFER. Число таких блоков определяется числом проверяемых условий. Например, блок GATE NU1 проверяет в режиме отказа незанятость прибора 1;

б) булевы переменные, определяемые следующим образом:

Name BVARIABLE ЛОГИЧСКОЕ_ВЫРАЖЕНИЕ

Значение булевой переменной (0 или 1), определяемое логическим выражением, может зависеть от состояния приборов. Для проверки значения булевой переменной в данном случае должен использоваться блок TEST в режиме отказа, т.е. с условным оператором ‘E’.

2. Каждый из процессоров, претендующих на занятие общих ресурсов, проверяет состояние так называемого «семафора». Если семафор включен, то процесс ожидает до тех пор, пока он не выключится. Если же семафор выключен, то один из процессов включает его и последовательно занимает ресурсы.

По окончании использования ресурсов процесс выключает семафор. Семафор может быть реализован в виде логического ключа (ЛК) либо в виде сохраняемой величины (СВ). Состояние ЛК может быть изменено с помощью блока

LOGIC X A,

где A — имя ЛК; X — вспомогательный оператор, задающий действие, которое необходимо выполнить над ключом (S — установить, R — сбросить, I — инвертировать). Проверка состояния ЛК осуществляется блоком

GATE X A,B,

который в данном случае должен работать в режиме отказа, где A — имя ЛК; X — вспомогательный оператор, определяющий проверяемое условие (LS — ключ установлен, LR — ключ сброшен). В поле B задается номер блока, к которому переходит сообщение, если логический оператор, указанный во вспомогательном поле операции X имеет значение "ложь". Если значение логического оператора "истина", сообщение переходит к следующему по номеру блоку. Если поле B пусто, блок работает в режиме условного входа, если заполнено — в режиме безусловного входа.

Для изменения значения сохраняемой величины служит блок

SAVEVALUE A,B,

где A — имя СВ; B — значение, присваиваемое СВ. Ссылка на СВ — Xj или X$ИМЯ. Для проверки значения СВ в данном случае может использоваться блок TEST в режиме отказа (оператор ‘E’).

3. Занимается первый из общих ресурсов. Перед занятием других ресурсов производится проверка их доступности. Если очередной ресурс доступен, то он занимается, если не доступен, то все уже занятые данным процессом ресурсы освобождается и процедура занятия ресурсов начинается снова. Для проверки доступности ресурсов может быть использован блок GATE NU A,B, работающий в условном режиме, где B — имя блока, в ко-

торый переходит транзакт в случае занятости прибора А. Для этой же цели может использоваться блок TRANSFER в режиме BOTH.

2.5. Вопросы к домашнему заданию

1. В чем заключается особенность параллельных вычислительных систем с общими ресурсами?

2. Каковы причины возникновения тупиковых ситуаций при использовании общих ресурсов?

3. Объясните понятие «семафор», используемое при построении имитационных моделей и в каких случаях они используются?

4. В каких случаях используются сохраняемые величины и логические переменные?

3. Лабораторное задание и методические указания по его выполнению

1. Выполнить прогон базовой модели. Определить причину окончания моделирования.

2. По номеру варианта выбрать задание из табл. 5. Каждая строка задания имеет вид С1, С2, С3, где С1=1,2,3 — способ предотвращения тупиков; С2=А,Б,В,Г,Д,Е — определяет средства языка GPSS, которые необходимо использовать при построении модели (А — блок TRANSFER в режиме SIM; Б — булевы переменные; В — ЛК; Г — СВ; Д — блок GATE в условном режиме; Е — блок TRANSFER в режиме BOTH); С3=Ж,И — определяет статистику, которую необходимо собрать в результате моделирования (Ж — процент сообщений, передача которых была задержана из-за занятости общих ресурсов; И — среднее время задержки передачи сообщений).

3. С использованием текста базовой модели и с учетом конкретного задания составить GPSS-модель процесса обмена сообщениями двух ЭВМ через общие ресурсы. Имена MET1,

MET2, MET3 присвоить блокам, непосредственно предшествующим блокам SEIZE сегмента 1.

Таблица 7 – Варианты заданий

Вариант	Задание	Вариант	Задание	Вариант	Задание
1	1,а,ж	5	2,г,ж	9	3,е,и
2	2,в,ж	6	3,е,ж	10	1,а,и
3	3,д,ж	7	1,б,и	11	2,в,и
4	1,б,ж	8	2,в,г	12	3,д,и

4. Выполнить прогон построенной GPSS-модели на ПЭВМ. Получить листинг. Определить причину окончания моделирования.

5. Определить требуемые статистические характеристики.

6. Оформить отчет.

4. Указания по оформлению отчета и контрольные вопросы по выполненной работе

4.1. Отчет должен содержать

1. Описание шагов составления имитационной модели.
2. Блок-схему имитационной модели.
3. Исходный текст модели и результаты моделирования.
4. Основные выводы по результатам моделирования.

4.2. Контрольные вопросы к лабораторной работе

1. Какая ситуация при функционировании вычислительной системы с общими ресурсами называется тупиковой? При каких условиях возникают тупики?

2. Какие существуют методы предотвращения тупиков?

3.Какие средства языка GPSS можно использовать для моделирования вычислительных структур с общими ресурсами?

ЛАБОРАТОРНАЯ РАБОТА № 5
ИЗУЧЕНИЕ МЕТОДОВ И СРЕДСТВ GPSS WORLD ДЛЯ РАБОТЫ СО СТАТИСТИЧЕСКИМИ ДАННЫМИ

1. Общие указания по выполнению лабораторной работы

1.1. Цель работы

получение практических навыков и изучение способов обработки статистических данных средствами языка GPSS;

изучение использования статистического метода, известного как дисперсионный анализ, при проведении моделирования в GPSS WORLD;

практическое применение изученных методов и средств GPSS WORLD для анализа данных по результатам моделирования.

1.2. Используемое оборудование и программное обеспечение

Интерпретатор GPSS WORLD, персональный компьютер, совместимый с IBM PC с объемом оперативной памяти не менее 512 Мб, операционная система Windows 7 и выше.

1.3. План выполнения лабораторной работы

Изучить теоретические основы выполнения лабораторной работы;

составить на языке GPSS модель по заданию преподавателя;

оттранслировать описание модели;

устранить ошибки, если они есть и получить файл выходной статистики;

провести анализ результатов моделирования;

провести анализ и обработку статистических данных.

2. Домашние задания и методические указания по их

2.1. Задание

По методическому руководству к лабораторной работе и литературным источникам, например [1,2,4], ознакомиться с методами и средствами обработки статистических данных в пакете моделирования GPSS WORLD.

2.2. Методические указания по выполнению задания

GPSS World имеет широкий инструментарий сбора статистики. Для несложных симуляций статистика стандартного отчёта и статистика библиотеки ANOVA удовлетворит большинство пользователей. В этих случаях GPSS World собирает и отчитывается о статистике автоматически.

Если требуется использование более детальной статистики, используйте переменные объекты и процедуры PLUS. В этих случаях вся сила математических функций библиотеки в вашем распоряжении. Статистические таблицы определяются в выражениях TABLE или QTABLE. Собственно запись информации вызывается одним или несколькими блоками имитации TABULATE. Каждый раз, когда вводится команда TABULATE, данная величина регистрируется в таблице. Команды QTABLE основываются на объектах очереди и представляют собой легкий путь для сбора статистики. В этом случае табуляции возникают путем входа в блок DEPART вместо блоков TABULATE.

Обычно, необходимо производить моделирование таким образом, чтобы значения управления были взяты от пользовательских переменных, и при окончании моделирования результаты находились в пользовательских переменных. Это происходит из-за особенностей работы созданных автоматическими генераторами экспериментов PLUS. Если вам все же придется использовать их, необходимо будет учесть эти ограничения.

Для всестороннего анализа можно накопить результаты моделирования при использовании команд Open, Close, Read, Write, Seek в форме блоков GPSS или вызовов процедуры. Статистика моделирований приводит к созданию базы данных, которая может использоваться для анализа. Для анализа данных в матрице результата, доступна встроенная многоканальная процедура ANOVA.

Если Вы плохо знакомы с моделированием, то Вы заметите, что, когда Вы повторяете моделирование на той же самой сессии, Вы можете получить различные результаты из-за случайных эффектов. Такие эффекты нужно отличfть от реальных эффектов, вызванных в соответствии с новыми проектами в ваших моделируемых системах. Процедура библиотеки ANOVA предоставляет Вам простое и разумное решение для того, чтобы установить, корректны ли ваши результаты или содержат вызванные отклонениями помехи.

Чтобы измерить количество случайного шума, Вы должны повторить моделирования, изменяя только определители случайных чисел. Эти повторные прогоны называют ответами. Они важны для измерения статистического шума, то есть стандартной ошибки в вашем эксперименте. В самой простой схеме каждый такой моделируемый проект должен повторяться несколько раз. Тогда процедура библиотеки ANOVA будет использовать данные, для того чтобы обнаружить присутствие эффектов, в значительной степени влияющих на изменение факторов эксперимента.

Процедура ANOVA предусматривает анализ данных нажатием одной кнопки. Она вычисляет доверительные интервалы и дисперсионный анализ результатов моделирования. Обычно, в процессе эксперимента, заполняется специальный глобальный объект матрицы, который называется матрицей результата с результатами моделирований, и затем передаете это как аргумент процедуре ANOVA.

При использовании статистического метода два варианта событий должны быть различны, а посторонние события не должны происходить при моделировании. Если происходит одно действие в некотором неизвестном месте моделирования, то это действие не выполняется в следующий раз. Следовательно, Вы не должны вообще использовать интерактивные методы управления моделированием, которые будут использоваться для статистики в Вашем заключительном отчете.

Возвратимся к примеру парикмахера, который не может остановить прибывающих заказчиков. Как администратор парикмахерской, Вы хотите определить то, какой эффект будет при более интенсивной работе парикмахера. Вы не хотите нанимать другого парикмахера. Вы хотите увидеть будет ли значительно уменьшено время ожидания заказчика, если парикмахер будет делать стрижку в среднем 5 минут.

Задача такова: парикмахер 1 против парикмахера 2. Парикмахеры отличаются только по среднему временю стрижки. Наши эксперименты состоят из двух уровней обработки: 6.8 минуты составляет в среднем время стрижки первого парикмахера, и 5 минут среднее время стрижки второго парикмахера.

Мы выполним 3 моделирования на каждом уровне. Вообще, Вы можете выполнять и больше.

Если наши результаты находятся в арифметической форме, среднее значение нескольких элементов можно определить, используя команду ANOVA, чтобы сделать статистический анализ. Мы можем найти критерий, используя среднее время ожидания заказчиков в полную меру.

Используя процедуру ANOVA можно сформировать Матрицу Результатов специального формата. Начнем.

Так как мы имеем 1 обработку с двумя уровнями, мы нуждаемся в 2 размерностях нашей Матрице Результатов. Последняя размерность Матрицы Ре-зультатов будет равна трем на каждом уровне обработки. Следовательно матрица 2 x 3.

Запустите Сеанс GPSS World.

ВЫБЕРИТЕ Файл / Открыть

В диалоговом окне

ВЫБОР Anova

ВЫБОР Открыть

Вы увидеть в Окне Модели.

```
**********************************************************
* Моделирование Парикмахерской
* Время находится в минутах
**********************************************************
 GENERATE    5,1.7;   Создайте следующего заказчика.
 QUEUE       Barber;  Начинается время очереди.
 SEIZE       Barber;    Ждут парикмахера.
 DEPART      Barber;   Конечное время очереди.
 ADVANCE     Cut_Time; Стрижка идет несколько мин.
 RELEASE     Barber;   Парикмахер свободен.
 TEST G      TG1,1,Doout; В последний раз XN пишет в Anova
данные.
 TERMINATE   1;     Заказчик уезжает.
Doout   OPEN        ("BARBER.RST"),,Prob; Откройте Anova базу
данных.
 SEEK        10000;     Поместите указатель в конце файла.
 WRITE       (Polycatenate(QT$Barber," ",Treatment," ")),,Prob,On
 CLOSE       Errorcode,,Prob ;Close,send to Prob if error.ошибку.
 TERMINATE   1
Prob    TERMINATE   1;   Завершите, если есть ошибка Ввода -
вывода.
```

Обратите внимание, что основной экспериментальный параметр в Блоке ADVANCE представляется Переменной Пользователя - Cut_Time. Это позволит изменять время, которое берется за полную стрижку. При втором наборе, мы используем те же самые произвольные номера и измени только быстродействие парикмахера, то есть моделирование выполняется при более быстрой работе парикмахера.

Перед стартом моделирования давайте рассмотрим Включенный файл, который используется для выполнения многократного моделирования.

```
RESULTS MATRIX ,2,3 ; Set up for 3 replicates of two levels
    Cut_Time EQU 6.8
    Treatment EQU 1
    RMULT 411
    Start 100,NP
    MSAVEVALUE RESULTS,1,1,QT$Barber
    Clear Off
    RMULT 421
    Start 100,NP
    MSAVEVALUE RESULTS,1,2,QT$Barber
    Clear Off
    RMULT 431
    Start 100,NP
    MSAVEVALUE RESULTS,1,3,QT$Barber
    Clear Off
    Cut_Time EQU 5
    Treatment EQU 2
    RMULT 411
    Start 100,NP
    MSAVEVALUE RESULTS,2,1,QT$Barber
    Clear Off
    RMULT 421
    Start 100,NP
    MSAVEVALUE RESULTS,2,2,QT$Barber
    Clear Off
    RMULT 431
    Start 100,NP
    MSAVEVALUE RESULTS,2,3,QT$Barber
```

Рассмотрим, что же будет, когда Вы Транслируете Включенный файл (Include-file). Сначала он создает Глобаль-

ную Матрицу, именованную Результатами (RESULTS), которая будет использоваться для ввода числовых значений Дисперсионного анализа. Матрица имеет две размерности, обеспечивая 2 уровня обработки с 3 попытками.

Далее, чтобы установить среднее время стрижки, равное 6.8 минутам, используется Команда EQU. Затем, после трехкратного выполнения моделирования и определения соответствующих данных, время установлено 5 мин, учитывая более быструю работу второго парикмахера в отличии от первого. Вы увидите, что установлено значение обработки (Treatment) перед каждым рядом выполнения. Это дифференцирует значения выполнения 6.8 мин Cut_Time (Treatment 1) и выполнение 5 мин Cut_Time (Treatment 2).

Обратите внимание, чтобы избежать корреляции между одним моделированием и следующим, мы очищаем (CLEAR OFF) статистику между ними. Параметр OFF необходим для избежания сбрасывания Матрицы Результатов. В этом простом примере мы не сделали скидку для пустой парикмахерской в начале каждого моделирования. На занятиях моделирования Вы убедились, что эффект запуска незначительный. Это можно сделать, игнорируя переходную точку запуска и используя команду RESET некоторое время. В GPSS World команда PLOT полезна для идентификации переходных точек, которые Вы можете исключить. Итак делаем вывод, что значение времени ожидания (waittime) для каждого из выполненных моделей начинается при полном зале. Сконцентрируемся на более простом примере.

В заключение, сделаем две дополнительные вещи. Выполним моделирование с опцией NP в операнде START. Это подавляет Стандартный Отчет. Отчеты могут также управляться с помощью Параметров настройки Объекта Моделирования. Также установим указатели произвольных генераторов номеров в Командном файле. Те же самые указатели используются во втором ряде, выполняясь так, чтобы единственное

различие в модели шло во времени, за которое делается стрижка. Это уменьшает разницу в результатах.

Создайте моделирование.

ВЫБЕРИТЕ Command / Create Simulation

Это создаст Объект Моделирования и откроет вид Журнала. Затем, используйте Включенный файл (Include-file), чтобы выполнить моделирование. Помните, GPSS World будет искать Включенный файл в той же самой папке, где находится объект модели. Создайте Команду для Включенного файла:

ВЫБЕРИТЕ Command / Custom

НАПЕЧАТАЙТЕ INCLUDE "ctlanova.txt"

ВЫБОР OK

Моделирование выполняется автоматически. Вы увидите, что моделирование выполнилось 6 раз за несколько секунд. Рассмотрим Матрицу Результатов, которую Вы формировали для Дисперсионного анализа.

ВЫБЕРИТЕ Window / Simulation Window / Matrix Window

Когда появится Диалог Раздела Пересечения Окна Матрицы,

ВЫБОР OK

Это открывает Окно Матрицы на Матрице Результатов (Results). Если все ячейки в Матрице невидимы, разверните окно. Вы увидите результаты каждого индивидуального моделирования (рис. 6).

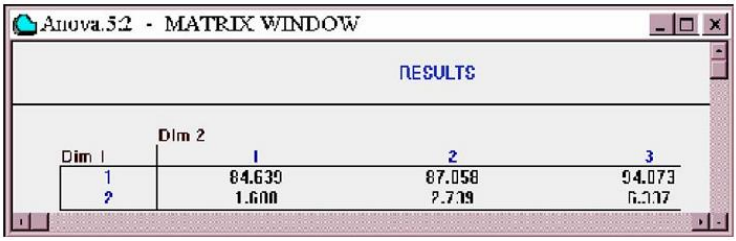

Рис. 6. Матрица Результатов (Results Matrix)

Теперь проанализируем.

ВЫБЕРИТЕ Command / Custom
НАПЕЧАТАЙТЕ SHOW ANOVA(Results,2,1)
И ВЫБОР ОК

ANOVA Процедура пишет ANOVA Таблицу в Окне Журнала и возвращает значение F анализа (рис. 7).

Теперь проанализируем наши результаты. Первый вопрос – есть ли эффект от более быстрой работы парикмахера.

Рассмотрите линию для фактора А в ANOVA Таблице в Окне Журнала. Мы видим, что критическое значение F для этого эксперимента в уровне 95 %, является 7.71. Так как наше расчетное значение F намного большее, делаем вывод, что эффект более быстрого парикмахера статистически значителен.

Иногда возможно подвергнуть исследованию реальные эффекты, увеличивая продолжительность моделирования и (или) номера копий. Статистическое значение более быстрого парикмахера подавляющее.

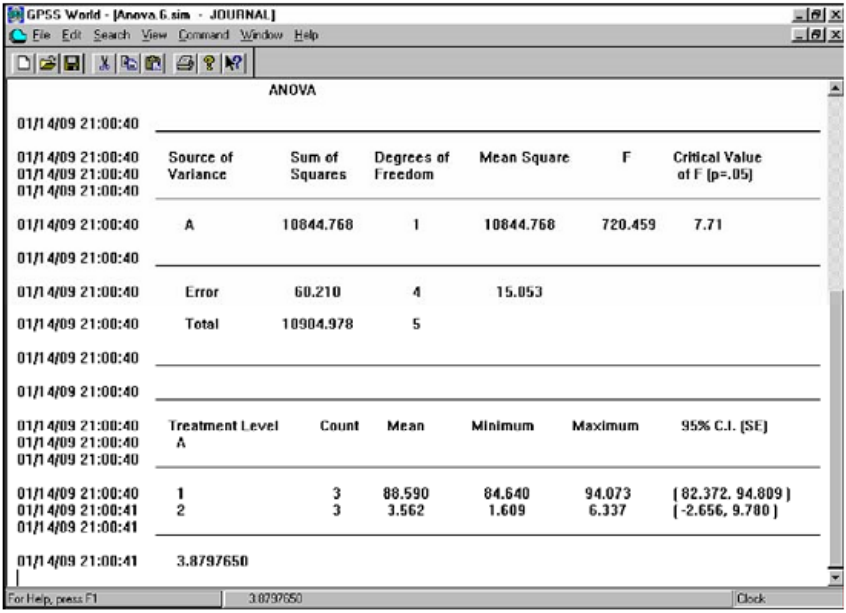

Рис. 7. ANOVA Таблица в Окне Журнала

Теперь рассмотрите последнюю линию в Журнале. Увидим, что возвращенное ANOVA Библиотечной Процедурой, 3.8797650. Это - Стандартная

Ошибка анализа и доступно для использования Процедурами, которые вызывают ANOVA подпрограмму.

В GPSS World ANOVA свойство обеспечивает очень простой первый уровень статистического анализа. Однако, Вы можете сделать и больше. В этом случае Вы можете создавать текстовый файл результатов для ввода к отдельной статистической программе анализа. Это - относительно простое использование Блоков GPSS OPEN, CLOSE, READ, WRITE и SEEK, а также находить Вереницу, форматирующую подпрограммы в ПЛЮС Библиотеки (PLUS Library).

Мощная GPSS World Процедура ЭКСПЕРИМЕНТА может использоваться, если нужно автоматически направлять и

анализировать прогресс эксперимента, включающего выполнение большого моделирования.

2.3 Вопросы к домашнему заданию

1. В каком случае возможно использование процедуры ANOVA?
2. Команды для всестороннего анализа результатов моделирования.
3. Как происходит трансляция включенного файла Include-file?
4. Алгоритм создания моделирования.

3 Лабораторное задание и методические указания по его выполнению

На основе теоретических сведений и разобранного примера, представленных в пункте 2.2 провести усовершенствование модели из предыдущей лабораторной работы и провести обработку и анализ полученных статистических данных.

4 Контрольные вопросы

1. Как создаются и заполняются статистические таблицы в GPSS?
2. В чем заключается предназначение переменных объектов и процедуры PLUS?
3. Как происходит сбор и обработка статистики библиотеки ANOVA?
4. Для чего предназначена опция NP в операнде START?
5. как осуществляется анализ данных в матрице результата?

ЛАБОРАТОРНАЯ РАБОТА № 6
ПРОВЕДЕНИЕ ЭКСПЕРЕМЕНТОВ В GPSS WORLD И ИХ ИСПОЛЬЗОВАНИИ. ОПТИМИЗАЦИЯ РЕЗУЛЬТАТОВ ЭКСПЕРИМЕНТОВ.

1. Общие указания по выполнению лабораторной работы

1.1. Цель работы

получение теоретических сведений и практических навыков проведения экспериментов и оптимизации их результатов средствами языка GPSS;

знакомство с типами и видами экспериментов в среде GPSS WORLD;

практическое применение изученных методов и средств GPSS WORLD для анализа данных по результатам моделирования.

1.2. Используемое оборудование и программное обеспечение

Интерпретатор GPSS WORLD, персональный компьютер, совместимый с IBM PC с объемом оперативной памяти не менее 512 Мб, операционная система Windows 7 и выше.

1.3. План выполнения лабораторной работы

Изучить теоретические основы выполнения лабораторной работы;

составить на языке GPSS модель по заданию преподавателя;

оттранслировать описание модели;

устранить ошибки, если они есть и получить файл выходной статистики;

провести анализ результатов моделирования;

провести анализ и обработку статистических данных, полученных в ходе проведения экспериментов;

провести оптимизацию результатов эксперимента.

2 Домашнее задание и методические указания по их выполнению

2.1. Задание

По методическому руководству к лабораторной работе и литературным источникам, например [1,2,4], ознакомиться с методами и средствами проведения экспериментов при моделировании в GPSS WORLD, проведении их оптимизации.

2.2 Методические указания к выполнению задания

Экспериментальная фаза проекта моделирования основана на существовании полностью разработанной, проверенной системы моделирования в среде GPSS, которая отображает влияние каждого из факторов, эффекты которых должны быть измерены. Только в этом случае вы можете продолжать экспериментальную фазу вашего проекта. Концептуально, среда GPSS поддерживает три различных подхода к экспериментированию: эксперименты отбора, пользовательские эксперименты и эксперименты оптимизации. Есть особенности, которые отличают каждый из них.

Эксперименты, сгенерированные GPSS, экспериментируют с неподвижным числом уровней обработки для каждого фактора. Когда Вы создаете собственные эксперименты, Вы не имеете этого ограничения. Вы ограничены 6 факторами, каждый из которых может иметь любое число уровней обработки. Ваша задача состоит в определении эксперимента, заполнении матрицы результата результатом каждого прогона и передаче ее процедуре библиотеки ANOVA.

Эксперимент отбора обычно используется, чтобы идентифицировать самые важные факторы, затрагивающие моделируемую систему. Эта информация является критической для

того, чтобы направить остальную часть исследования самым эффективным способом. Результаты отбора определяют, какие факторы не эффективны и должны получить низкий приоритет относительно дальнейшего исследования. Кроме того, чувствительность результатов эксперимента к одному или нескольким скрытым факторам даёт также может иметь значение. Среда GPSS содержит автоматический генератор эксперимента, который может создать эксперименты отбора. Для его использования необходимо заполнить параметры в окне меню редактирования основного окна. Это генерирует PLUS- код, вставляемый в объект модели. Этот процесс также может загрузить функциональную клавишу соответствующей командой CONDUCT для моделирования. После этого, вам остаётся создать объект модели (Ctrl+Alt+S), запустить эксперимент (обычно F11), и затем проанализировать результаты.

Оптимизация и количественное предсказание поведения системы - часто первичные цели проекта моделирования. Оба из них непосредственно поддерживаются в среде GPSS. Поверхность отклика – это уравнение, которое предсказывает результаты моделирования. Часто желательно установить поверхность отклика для того, чтобы сократить результаты, обеспечивая прогнозирующую методологию, определяя количество чувствительности результатов к числовым входам и другим факторам, и определяя оптимальные уровни обработки. GPSS обеспечивает основу для разнообразных методов нахождения поверхности откликов. PLUS эксперимент, который вычисляет поверхность отклика, Результирующий эксперимент может использовать метод наискорейшего подъёма и метод местного исследования для нахождения оптимальную значение. В случае успеха, выводится математическое описание наилучшей пригодной поверхности отклика и предсказанных оптимальных условий.

Дисперсионный анализ - инструмент, созданный сэром Рональдом Фишером, который является в состоянии извлечь

большую часть информации, доступной в ряде замеров. Мы определяем количество изменений наблюдений от полного среднего числа, и затем ломаем это на кусочки, каждый из которых имеет отдельную причину. Если какой-нибудь экспериментальный фактор не может быть найден, чтобы вызвать изменчивость в измерении, мы говорим, что это не имеет существенного эффекта на эксперимент. С другой стороны, если фактор действительно вызывает изменчивость, мы сравниваем количество этого с оценкой свойственной изменчивости наблюдения, стандартной ошибки. Мы делаем это, чтобы исключить очевидные эффекты, которые являются ни чем иным, как случайным колебанием. Наш стандарт сравнения то, что изменение из любого источника должно быть намного большим чем стандартная ошибка, чтобы считаться существенным эффектом. Тест F, названный по имени Фишера, используется с этой целью. Мы используем тест F, как критерий, в соответствии с которым мы объявляем, что эффекты экспериментальных факторов и их взаимодействий статистически существенные. Неявный в использовании ANOVA – это совокупность математических моделей, для описания изменяющихся компонентов в наблюдениях. Мы назовем это "статистической моделью".

Теперь мы обратимся к определенным особенностям GPSS, который может использоваться в любой из фаз экспериментирования проекта моделирования. Сначала, мы полагаем, что особенности предусмотрели анализ пользовательских экспериментов. GPSS предусматривает два вида автоматически сгенерированных экспериментов, которые проектированы для Вас, и пользовательских экспериментов, которые спроектированы Вами.

GPSS World может создать эксперименты фильтрации или оптимизации. Их можно использовать для исключения нерелевантных факторов или для нахождения лучшей комбинации обработки. Процесс вставляет PLUS-эксперимент в объект

модели и по желанию назначает функциональной клавише F11 соответствующую команду CONDUCT. Вся процедура занимает 4 шага:

1. Заполните диалог (меню Edit) и щелкнитеOK.
2. Отредактируйте процедуру щапуска и нажмите OK.
3. Оттранслируйте модель (Ctrl+Alt+S).
4. Нажмите функциональную клавишу.

Вот и всё. Эксперимент запустится и запишет статус в журнал. После завершения будет также создан полный отчёт. В случае эксперимента оптимизации в отчет будет включено математическое описание поверхности отклика вокруг оптимума. Анализ отчета входит в ваши задачи.

Оба типа сгенерированных PLUS-эксперментов содержат включение процедуры запуска PLUS. Проследите за тем, чтобы эта процедура корректно задала условия запуска.

Процедура запуска вызывается каждый раз при начале новой имитации. Вы также можете настроить генератор экспериментов на создание шаблонной процедуры запуска. Даже в этом случае её можно прспособить под свои требования. Технология этого описана в следующем разделе.

Процедура запуска – это гибкая связь между произведенным экспериментом и имитацией пользователя. И эксперименты отбора, и эксперименты оптимизации используют их. Произведенные эксперименты неоднократно вызывают процедуру запуска, по заданным в диалоге создания пользовательским именам, чтобы выполнить каждый запуск в эксперименте. Каждый эксперимент должен вызвать вашу процедуру запуска или процедуру запуска GPSS World. Процедура запуска используется, чтобы установить период измерения в течение моделирования, которое должно быть соблюдено. Это настраивает каждый запуск в эксперименте согласно специфическим требованиям вашего моделирования и вызывается один раз для каждого запуска в эксперименте. Процедуры, которыми управляют PLUS процедуры вызывают экспери-

менты и поэтому могут содержать запросы DoCommand. Обычно процедура запуска содержит команды установки для моделирования, типа команд RMULT, чтобы управлять генераторами случайных чисел, и START и RELOAD команды, чтобы настроить период измерения запуска.

Когда эксперимент генерируется автоматически, генератор по желанию создаёт шаблонную процедуру запуска, которая может быть изменена как необходимо. В качестве примера приведена процедура запуска по умолчанию, созданную генератором экспериментов:

```
PROCEDURE RunProc(Run_Number) BEGIN
    DoCommand("CLEAR OFF"); /* Must use OFF to preserve results. */
    /* EXPAND THIS RMULT IF YOU HAVE MORE RNGs. */
    /* All Random Number Streams must have new seeds. */
    TEMPORARY CommandString;
    /* Evaluate before passing to DoCommand. */
    CommandString = Catenate("RMULT ",Run_Number#111);
    /* DoCommand compiles the string in Global Context. */
    DoCommand(CommandString);
    /* SET UP YOUR OWN RUN CONDITIONS. */
    DoCommand("START 100,NP"); /* Get past the Startup Period. */
    DoCommand("RESET"); /* Begin the Measurement Period. */
    DoCommand("START 1000,NP"); /* Run the Simulation. */
END;
```

Если процедура запуска создаётся в GPSS World, вы можете редактировать её под свои нужды. Процедуру можно редактировать как до помещения в объект модели, так и после этого.

Фильтрующие эксперименты используются, чтобы искать факторы, которые должны быть исследованы более тщательно или те, которые подозреваются в несущественности. Однако, комбинаторная природа измерения главных эффек-

тов и всех возможных взаимодействий часто делает полный набор из запусков, названных полным факторным планом экспериментов, слишком длинным и/или дорогим для запуска. Чтобы сделать процедуру более управляемой, мы ограничиваем число уровней обработки двумя факторами, и ищем пути управления только частью возможных запусков, не жертвуя самой важной информацией. Генератор фильтрующего эксперимента GPSS World делает большинство работы за Вас. Вы должны решить, какую информацию Вы можете опустить, чтобы уменьшить число запусков в эксперименте. Обычно это были бы высокоуровневые взаимодействия фактора, которые являются обычно незначащими и могут игнорироваться. Даже если Вы считаете иначе, помните, что цель эксперимента отбора состоит в том, чтобы идентифицировать важные факторы, которые будут подробнее изучены позже. Когда Вы требуете, чтобы GPSS уменьшил полный факторный план, показывающий на экране эксперимент исходя из набора запусков, некоторые эффекты смешиваются с другими, и Вы не можете отличить их в дисперсионном анализе. GPSS World разделит набор всех возможных множеств эффектов наподмножества называемые альтернативными группами. Только объединенный результат всех эффектов в группе показателен в дисперсионном анализе. Это означает, что, если фактор А и фактор

В находится в альтернативной группе, Вы потеряете возможность определить, какой из них является ответственным за наблюдаемый эффект. Возможно еще худшим событием является наличие двух сильных эффектов в той же самой альтернативной группе, которые отменяют друг друга, таким образом скрывая фактические эффекты каждого фактора.

Однако, обычно возможно устроить альтернативные группы так, чтобы каждый главный эффект и самые важные взаимодействия с 2 путями находились в собственных альтернативных группах, только с более высокими порядками

взаимодействия. GPSS World предоставляет альтернативные группы по умолчанию, которые изолируют главные эффекты, если это возможно. Это облегчит для Вас изменение альтернативных групп, если настройки по умолчанию не подходят. В любом случае, когда Вы запускаете эксперимент отбора в GPSS World, Вы должны всегда исследовать альтернативные группы прежде, чем Вы начинаете запуски. Если Вы не отделили самые важные эффекты в отличные группы, интерпретация результатов будет трудна.

Генерация PLUS-эксперимента отбора

Первый шаг в создании эксперимента отбора - открыть диалог. Для этого щелкните "Insert Experiment" в меню «Edit» главного окна, и нажмите "эксперимент отбора ...". Для этого Вы должны иметь открытый активный объект модели. Теперь заполните поля и нажмите OK, когда Вы готовы. Рисунок 8 показывает диалоговое окно эксперимента отбора

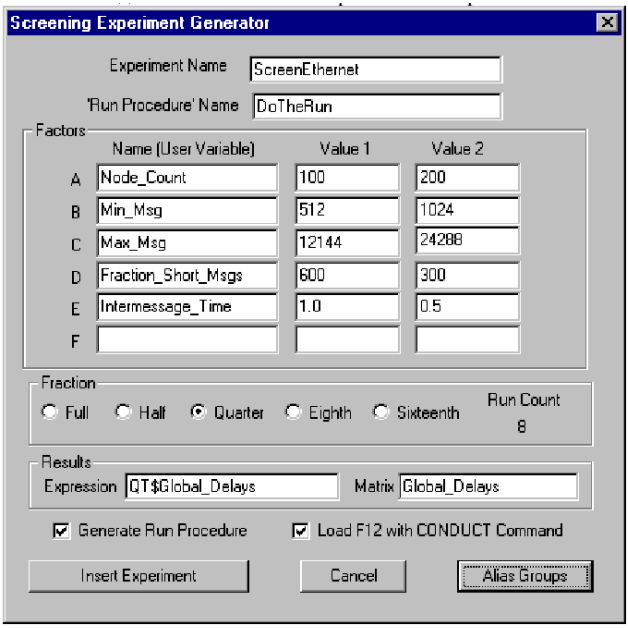

Рис. 8. Генератор эксперимента отбора

Чтобы определить производимый PLUS-эксперимент, мы просто заполняем поля диалога. "Название эксперимента" и «области 'Name'» - процедуры, которыми управляют, используются как названия ПРОЦЕДУРЫ в произведенных экспериментах. ЭКСПЕРИМЕНТ переименовывает процедуру запуска неоднократно. Факторы эксперимента прибывают следом. Каждое название фактора - фактически название пользовательской переменной и должно подчиняться правилам GPSS World. То есть именование должно начаться с алфавитного символа, и это не должно совпадать с ключевым словом, SNA, или классом SNA. Начиная с экспериментов отбора в GPSS World введены полные или дробные факторные эксперименты 2 КБ. Есть два уровня обработки, которые будут определены для каждого фактора. Вы должны определить названия и два уровня обработки для между 1 и 6 факторами, включительно. Факторы должны быть определены, последовательно начинаясь с первого фактора. Выбор уровней обработки является критическим. Вы можете хотеть сделать некоторое предварительное экспериментирование так, чтобы выборы, которые Вы делаете, были хорошие. Выберите уровни, которые должны быть далеко друг от друга, чтобы выявить изменения поведения и пробовать избежать уровней, где эффекты скрыты другими факторами. Группа "Дробь" является следующей. Это позволяет нам определять, какой дробью полного эксперимента 2 КБ нужно управлять. Счет запуска, который закончится, определен направо. Выберите меньшую дробь, чтобы уменьшить счет запуска. "Выражение результата" требуется. Вы должны определить выражение, которое будет оценено как метрическое из моделирования. Затем мы имеем два флажка, которые позволяют нам выбирать дополнительные варианты. Вы только должны нажать функциональную клавишу F11, чтобы начать эксперимент.

Улучшенная модель эксперимента.

Из диалога «альтернативные группы» мы можем видеть, что этот экспериментальный дизайн изолирует все главные факторы, не путает их со взаимодействиями, и это даже изолирует взаимодействия с 2 путями первых немногих главных эффектов. Бросьте более близкий взгляд на эффекты, используемые как "генераторы" в диалоге «альтернативные группы». Мы можем изменить разделение эффектов в различный набор альтернативных групп, просто используя различные взаимодействия высокого уровня сгенерированного набора. Чтобы сделать это, измените генераторы и нажмите на кнопку GENERATE.

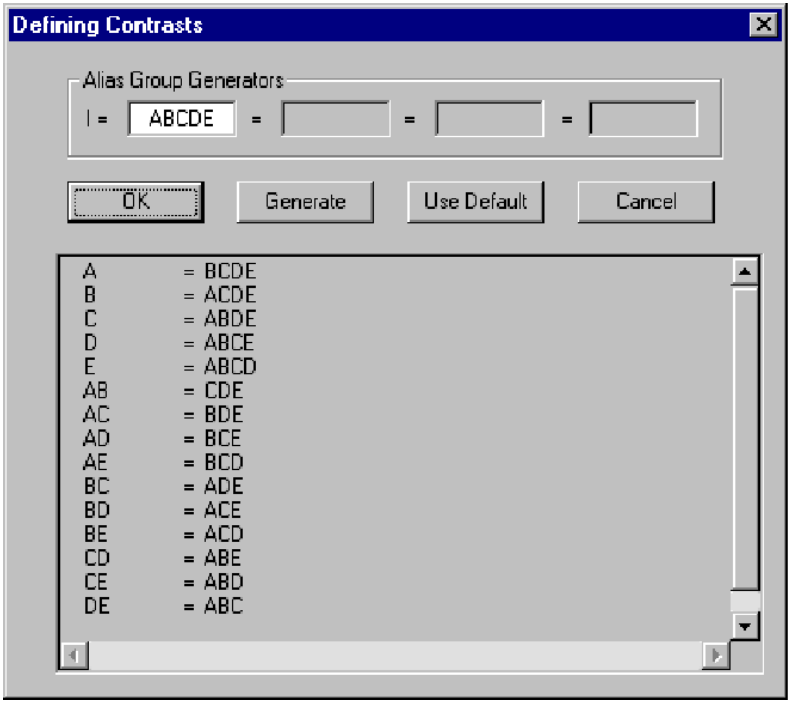

Рис. 9. Диалоговое окно псевдонима групп.

Прежде, чем Вы делаете это, есть несколько вещей, которые необходимо знать о генераторах. Сначала, Вы

должны знать то, что когда GPSS World создает альтернативу, это сначала расширяет набор генераторов, формируя весь возможный модуль на 2продукта и добавляя их к набору генераторов, теперь названных "Расширенным Набором" генераторов.

Например, если ABCD и BCDE будут генераторами, то их продукт, выполнение приложения, будет в «Расширенном Наборе» генераторов (ABCDE * BCDE = ABBCCDDE = (BB) (машинописная копия) (DD) E = выполнение приложения). Отметьте, что любой эффект в «Расширенном Наборе» генераторов не будет появляться в сообщении эффектов.

Это означает, что Вы не должны выбирать генераторы, которые приводят к важному эффекту, происходящему в «Расширенном Наборе» генераторов. Вы можете возвратиться к генераторам по умолчанию, предложенным GPSS World, нажимая на Кнопку «Use defaults». Нажмите Enter, чтобы создать новый список альтернативных групп.

На следующий шаг должен определиться PLUS-кодекс, включающий эксперимент в ваш объект модели. Когда Вы пошли на компромисс между числом запусков и совмещением эффектов, щелкните кнопку OK в диалоге альтернативных групп и кнопку «Вставка эксперимента» в диалоге генератора эксперимента отбора. Если Вы потребовали, чтобы GPSS World произвел процедуру запуска, откроется соответствующий диалог. Детали об опрелении процедуры запуска находятся в предыдущей секции. В любом случае, Вы можете редактировать произведенный эксперимент непосредственно в вашем окне модели.

Сгенерированный эксперимент

```
**********************************************
*
* ScreenEthernet
* Fractional Factorial Screening Experiment
```

```
*
**********************************************
 Global_Delays MATRIX ,2,2,2,2,2
INITIAL Global_Delays,UNSPECIFIED
 EXPERIMENT ScreenEthernet() BEGIN
      /* Run 1 */
      Node_Count = 100;
      Min_Msg = 512;
      Max_Msg = 12144;
      Fraction_Short_Msgs = 600;
      Intermessage_Time = 1.0;
      IF                                          (StringCom-
pare(DataType(Global_Delays[1,1,1,1,1]),"UNSPECIFIED")'E'0)
      THEN BEGIN
           /* Run Procedure Call */
           DoTheRun(1);
           Global_Delays[1,1,1,1,1] = QT$Global_Delays;
       END;
      /********* Runs 2-15 would go here *********/
      /* Run 16 */
      Node_Count = 200;
      Min_Msg = 1024;
      Max_Msg = 24288;
      Fraction_Short_Msgs = 300;
      Intermessage_Time = 1.0;
      IF                                          (StringCom-
pare(DataType(Global_Delays[2,2,2,2,1]),"UNSPECIFIED")'E'0)
      THEN BEGIN
           /* Run Procedure Call */
           DoTheRun(16);
           Global_Delays[2,2,2,2,1] = QT$Global_Delays;
      END;
      /* Aliased Effects in Fractional Factorial Experiment */
      EFFECTS(Global_Delays,"I=ABCDE");
 END;
 **********************************************
 * Run Procedure *
 **********************************************
```

```
PROCEDURE DoTheRun(Run_Number) BEGIN
    DoCommand("CLEAR OFF"); /* Must use OFF to preserve
results. */
    /* EXPAND THIS RMULT IF YOU HAVE MORE RNGs. */
    /* All Random Number Streams must have new seeds. */
    TEMPORARY CommandString;
    /* Evaluate before passing to DoCommand. */
    CommandString = Catenate("RMULT ",Run_Number#111);
    /* DoCommand compiles the string in Global Context. */
    DoCommand(CommandString);
    /* SET UP YOUR OWN RUN CONDITIONS. */
    DoCommand("START 100,NP"); /* Get past the Startup Pe-
riod. */
    DoCommand("RESET"); /* Begin the Measurement Period.
*/
    DoCommand("START 1000,NP"); /* Run the Simulation. */

END;
**********************************************
```

Запуск эксперимента

Команда CONDUCT используется в GPSS World для начала эксперимента. Синтаксис схож вызовом процедуры. Однако, есть способ лучше. В диалоге создания эксперимента можно настроить GPSS World на использование клавиши F11 с соответствующей командой CONDUCT. Это можно увидеть в настройках объекта модели. (см. Edit / Settings / Function Keys).

Чтобы начать выполнение эксперимента, необходимо сделать две вещи: во-первых, необходимо оттранслировать модель, создав объект имитации. Для этого щелкните

Command / Create Simulation или нажмите Ctrl+Alt+S.

Затем нажмите F11. Это вводит команду CONDUCT. эксперимент начинает работу, сообщая статус и выводя информацию в окно журнала созданного объекта имитации. На ри-

сунке 10 показан результат выполнения эксперимента по отбору.

Alias Group	Effect	Sum of Squares	Degrees of Freedom	F - for Only Main Effects	Critical Value of F (p=.05)
A = BCD	0.030	0.002	1	0.000	10.13
B = ACD	-0.058	0.007	1	0.000	10.13
AB = CD	-0.178	0.063	1		
C = ABD	0.063	0.008	1	0.000	10.13
AC = BD	-0.014	0.000	1		
AD = BC	-0.109	0.024	1		
D = ABC	0.177	0.062	1	0.000	10.13
Grand Mean	11.456				
Total		1050.158	7		
Error		1050.079	3		

01/16/01 13:45:33 Experiment Ended.

Рис. 10. Представление генератора экспериментов

Отчет об эксперименте показывает силу каждого эффекта и, если возможно вычисляет статистическую величину F для групп переменных с главными эффектами в качестве участников. Так Вы можете запланировать будущие действия, игнорируя незначащие факторы и более полно исследуя важные.

Оптимизация и количественный прогноз поведения системы часто - главные цели проекта моделирования. Оба из них непосредственно поддерживаются языком GPSS World's Optimizing Experiment Generator.

Методология поверхностной характеристики - набор статистических методик для эмпирической разработки модели и образов исследования. Хотя чаще всего используют оценку оптимальных уровней обработки, это также может предоставить важную информацию для общих прогнозов

объема Выпуска, ожидаемого при непроверенных условиях. GPSS поддерживает несколько важных методологий поверхностной характеристики, в известном смысле, предназначенный для того, чтобы сделать их удобными. Анализ моделирования устанавливает начальные условия в нескольких диалоговых окнах, и тогда GPSS автоматически создает отклик поверхностного эксперимента, который ищет оптимальное значение. Идеально, Вы можете запустить эксперимент и позволить ему работать автоматически до тех пор, пока он не найдет оптимальное значение Выбранного показателя. Когда Вы заполняете диалоги в windows, Вы, в действительности, определяете блок (фактически "гиперблок") названный локальной экспериментальной областью, которая блуждает по набору допустимых факторных уровней. Если Вы сделаете этот блок слишком маленьким, то эксперимент будет выполняться больше, чем оптимально необходимо. Если Вы сделаете его слишком большим, то оценки оптимальных уровней обработки будут менее точными, чем они могли быть.

GPSS World применяет несколько различных методологий, поскольку он исследует поверхностные характеристики объекта моделирования GPSS WOULD. Первичное понятие - понятие локальной экспериментальной области, которая определена Вами в диалоге установки. Чтобы определить его, каждый из 5 факторов, которые будут получены, связаны с двумя значениями, определяющими степень локальной экспериментальной области. Тогда, это - целая область, которая перемещает факторное пространство. Цель, которая не всегда достижима, состоит в том, чтобы содержать оптимум в пределах локальной области, осуществить подтверждение выполнения, и затем сообщить об оптимальных условиях и математической модели, которая использовалась.

GPSS использует несколько главных экспериментов различной сложности, чтобы получить необходимую инфор-

мацию. Поскольку результаты доступны, они должны соответствовать или линейной модели или второй модели (включая двухсторонние взаимодействия) данных. Это начинается при использовании метода Steepest Ascent, поскольку он оставляет свою отправную точку. Директория определена градиентом первой модели, которая соответствует данным. Если происходят сбои теста для первой модели, используется вторая.

Как только градиент вычислен, центр локальной экспериментальной области перемещает его до тех пор, пока он не попадает или в сортировку с убывающим шагом или с пределом, утверждаемым в локальной экспериментальной области, как определено в генераторе оптимизированного эксперимента. Если диапазон уровней обработки будет слишком узким, то выполнится много ненужных шагов, чтобы достигнуть градиента.

Каждый шаг перемещения требует моделирования в центре локальной экспериментальной области, и о его результатах сообщают в журнале. Если, при остановке движения, локальная экспериментальная область не содержит оптимальных значений, эксперимент попытается перейти в новую директорию. Это действие может быть подавлено перенаправлением границ в генераторе эксперимента оптимизации.

Рисунок 11 показывает диалог обычно используемого генератора оптимизированного эксперимента. Так же, как с генератором представления эксперимента к нему обращаются через меню правка главного окна Windows.

Чтобы определить PLUS эксперимент, который генерируется, мы просто заполняем в поля диалога. "Название Эксперимента" и ‘Запуск процедуры’ Имя – это поля, использующиеся как имена процедур в сгенерированных экспериментах. Эксперимент неоднократно вызывает выполненную процедуру.

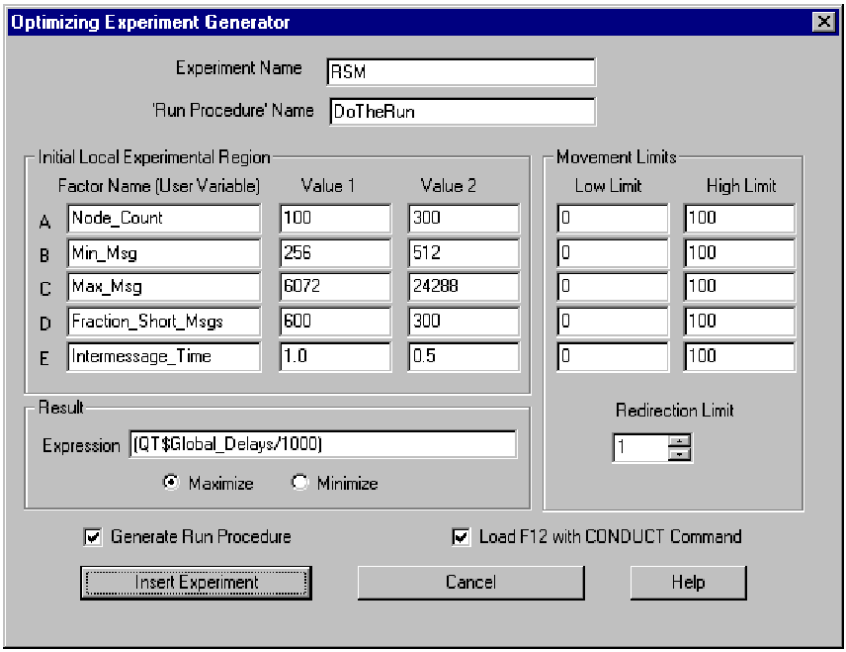

Рис. 11. Генератор оптимизированного эксперимента

Затем появляются факторы эксперимента. Каждое факторное название - фактически название пользовательской переменной и должно удовлетворять условию соглашениям об именах GPSS WOULD. Имена должны начаться с алфавитного символа, и это не должны совпадать с ключевым словом, протоколом SNA, или классом протокола SNA. Два уровня обработки каждого фактора определяют локальную экспериментальную область, которая смещает поиск оптимальных условий. Вы должны определить имена и два уровня обработки для диапазона между 1 и 5 факторами, включительно. Факторы должны быть определены, последовательно начиная с А фактора. Выбор уровней обработки крайне важен.

Несколько важных обменов неявны в способе, которым Вы определяете локальную экспериментальную область. Во-первых, формируется отправная точка эксперимента. Вы

должны сделать ее как можно ближе к оптимальной. Во-вторых, минимизируя ее, как правило, ваша математическая модель наиболее пригодна. В-третьих, когда эксперимент делает один шаг к оптимуму, точка перемещается в смежную область. Поэтому, большая экспериментальная область сделает меньше шагов. Как Вы видите, результаты конфликтуют друг с другом. Вам решать к какому компромиссу прийти. В дальнейшем, не упускайте факт того, что легко изменить сгенерированный эксперимент после того, как он был вставлен в вашу модель. Стоит проверить сгенерированный PLUS код, чтобы увидеть способы улучшения эксперимента. Возможно, Вам захочется сделать некоторое предварительное экспериментирование так, чтобы выборы, которые Вы делаете, были хорошими.

Вы должны использовать эксперименты оптимизации. Раздел диалога для перемещения границ дает Вам возможность ограничить движение локальной экспериментальной Области. Если возможно Вы должны ввести границы для каждого из факторов так, чтобы время не было потрачено впустую на неосуществленные моделирования.

Перенаправление границ - другой способ ограничить поиск. Значения устанавливают границы для количества экспериментов по времени и могут передвигаться в новую директорию факторного пространства. Эти границы могут предотвратить бесконечное выполнение цикла.

Требуется "Выражение результата". Вы должны определить выражение, которое будет оценено как показатель моделирования.

В Итоге, у нас есть два переключателя, которые позволяют нам выбирать дополнительные опции. Во-первых, у вас может быть GPSS, которая производит шаблон выполнения процедуры, чтобы согласовать с данным сгенерированным экспериментом. Вы можете редактировать выполнение процедуры, чтобы удовлетворить Ваши потребности. Во-вторых,

у вас может быть GPSS, устанавливающая настройки, которые загружают соответствующую команду CONDUCT в установку нажатием клавиши F12. Если Вы выбрали это, после того, как создали объект моделирования, Вы только должны нажать F12, чтобы выполнить эксперимент.

Следующий шаг должен вставить PLUS код, включающий эксперимент в ваш шаблонный объект. Когда Вы заполнили все соответствующие поля в диалоге генератора оптимизации эксперимента, нажмите Insert Experiment, чтобы создать PLUS код и вставить его в Вашу модель. Если Вы запрашиваете GPSS для того, чтобы он сгенерировал выполнение процедуры, Вы видите диалоговое окно, которое позволяет нам персонализировать его. Подробности об индивидуализации выполненной процедуры находятся в начале этого раздела.

В любом случае, Вы, возможно, захотите отредактировать сгенерированный эксперимент непосредственно в окне вашей модели. Вы можете изменить выполнение процедуры там, как Вы пожелаете. Нажмите только OK, и все новые операторы GPSS включающие эксперимент оптимизации, помещены внизу вашей модели.

Выполнение эксперимента

Conduct - команда GPSS, которая используется, чтобы вызвать эксперимент. Параметр к команде Conduct подобна типичному вызову процедуры. Но у нас есть лучший путь. В диалоговом окне дерево построений мы также можем вызвать загрузку GPSS через клавишу F12 с соответствующей командой Conduct . Вы можете видеть это в параметрах настройки вашей модели (см. редактирование / параметры настройки / функциональные клавиши).

Чтобы запустить выполнение эксперимента, мы должны сделать две вещи. Сначала мы должны преобразовать модель, таким образом создавая объект моделирования.

Нажмите на команда/ создать моделирование или сочетание Ctrl+Alt+S.

Тогда мы нажимаем функциональную клавишу F12. Это вводит команду Conduct.

Эксперимент переходит в режим работы, сообщая текущее состояние и выводя его в окно журнала недавно созданного объекта моделирования. Рисунок 8 показывает результаты эксперимента оптимизации.

Когда все прошло успешно, эксперимент оптимизации сообщает результаты каждого запуска. Благодаря этому можно просмотреть результаты каждого подэксперимента и проследить движения всех локальных экспериментальных областей. Сгенерированный PLUS эксперимент пытается переместить локальную экспериментальную область так, чтобы это включить оптимальные условия. Эксперимент заканчивается, когда это происходит, или когда предел перенаправления или предел движения достигнуты.

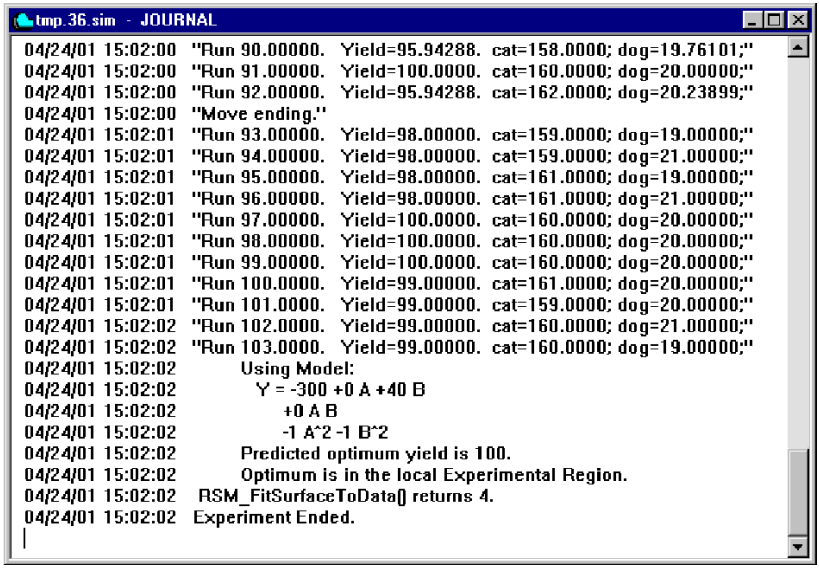

Рис.12. Результаты оптимизации эксперимента

Когда перемещение останавливается, о результатах пригодности тестирования и уравнения для соответствующих поверхностных характеристик также сообщаются в окне журнала. Тогда невыполненный PLUS эксперимент пытается проверить спрогнозированные оптимальные условия запуском моделирования.

2.3 Вопросы к домашнему заданию

1. На чем основана экспериментальная фаза проекта моделирования?
2. В чем заключается сущность эксперимента отбора?
3. Оптимизация и количественное предсказание поведения системы.
4. Алгоритм создания и обработки PLUS-эксперимента.
5. Генерация PLUS-эксперимента отбора.

3 Лабораторное задание и методические указания по его выполнению

На основе теоретических сведений и разобранных примеров, представленных в пункте 2.2, провести усовершенствование модели из предыдущей лабораторной работы и провести эксперименты фильтрации и оптимизации.

На основании полученных результатов моделирования провести их анализ, сделать вывод.

4 Контрольные вопросы

1. Что понимается под компьютерным экспериментом?
2. Каковы цели стратегического и тактического планирования эксперимента?
3. Какова цель и средства проведения дисперсионного анализа (отсеивающего эксперимента) в GPSS World?
4. В каком виде выводятся результаты дисперсионного

анализа? Дайте характеристику выводимых величин.

5. Как изучить группы смешивания с целью осуществления стратегического планирования эксперимента?

6. Какова цель и средства проведения оптимизирующего эксперимента в GPSS World?

7. Дайте характеристику метода поверхности отклика.

8. Как должны быть представлены в программе модели характеристики моделируемой системы, которые будут использоваться при проведении исследований с применением генераторов экспериментов?

ЛАБОРАТОРНАЯ РАБОТА № 7 - 9
СОЗДАНИЕ, АНАЛИЗ ВЫПОЛНЕНИЯ, И АНИМИРОВАНИЕ ИМИТАЦИОННЫХ МОДЕЛЕЙ В РАСШИРЕННОМ РЕДАКТОРЕ GPSS

1. Общие указания по выполнению лабораторной работы

1.1. Цель работы

- приобретение навыков построения и исследования GPSS-моделей в расширенном редакторе GPSS параллельных вычислительных структур с общими ресурсами и изучение методов предотвращения тупиковых ситуаций, возникающих при функционировании таких систем.

1.2. Используемое оборудование

Интерпретатор GPSS WORLD, Расширенный редактор GPSS World, персональный компьютер, совместимый с IBM PC с объемом оперативной памяти не менее 512 Мб, операционная система Windows 7 и выше.

1.3. При выполнении лабораторной работы необходимо

- в результате выполнения лабораторной работы должны быть приобретены знания: возможностей расширенного редактора GPSS World, по созданию, анализу выполнения и анимированнию имитационных моделей;
- при выполнении лабораторной работы необходимо изучить функционал графического редактора GPSS World; составить анимированные модели по заданию преподавателя;

2. Домашние задания и методические указания по их выполнению

2.1. Задание первое

По методическому руководству к лабораторной работе ознакомиться с графическим редактором GPSS World и его функционалом.

2.2. Методические указания по выполнению первого задания

Расширенный редактор – инструмент для разработки имитационных моделей и анализа результатов моделирования. Он создан, чтобы помочь разработчикам в области имитационного моделирования реализовывать сложные проекты с использованием современных средств ввода и отображения информации.

Компоненты редактора:

– упрощают написание и отладку модели на языке GPSS World;

– позволяют строить имитационные модели в виде иерархических схем и нарабатывать библиотеки повторно используемых элементов;

– помогают формировать одиночные и серии экспериментов и анализировать их результаты;

– делают возможным создание независимого exe-модуля для развертывания модели у заказчика.

В качестве моделирующего ядра используется язык GPSS World. Он имеет огромные возможности для построения имитационных моделей и удивительно прост в освоении. Язык GPSS World преподается во многих ВУЗах и является образовательным эталоном для изучения принципов, задач и возможностей имитационного моделирования. Расширенный редактор значительно расширяет возможности исследователя, по сравнению со стандартным редактором GPSS World.

Графический редактор моделей - редактор схем

С его помощью, разработчик способен формировать структурную схему модели, двигаясь от частного к общему (снизу-вверх) или сверху-вниз (от абстрактных понятий к деталям). Схема формируется из определяемых разработчиком типовых элементов (ТЭБов) двух видов:

Первый вид элементов («чёрный ящик») представляет конечный элемент схемы. Он содержит набор входов и выходов, для взаимодействия с другими элементами, а также модель GPSS World, которая определяет его поведение и состояние. ТЭБ можно настроить с помощью параметров. Входы и выходы ТЭБа ассоциируются с метками модели, а взаимодействие между ТЭБами выражается в движении транзактов между их входами и выходами. Т.к. ТЭБ представляет собой «чёрный ящик», разработчик имеет возможность указать тип транзактов, которые могут работать с определенным входом или выходом ТЭБа, чтобы согласовать логику работы множества элементов.

Второй вид элементов – сложный блок, состоящий из нескольких простых. Он также имеет входы и выходы. Но вместо модели содержит подсхему из взаимосвязанных ТЭБов более низкого уровня.

Комбинируя эти два вида элементов, разработчик получает возможность строить иерархические схемы с множеством уровней декомпозиции и с требуемым уровнем детализации.

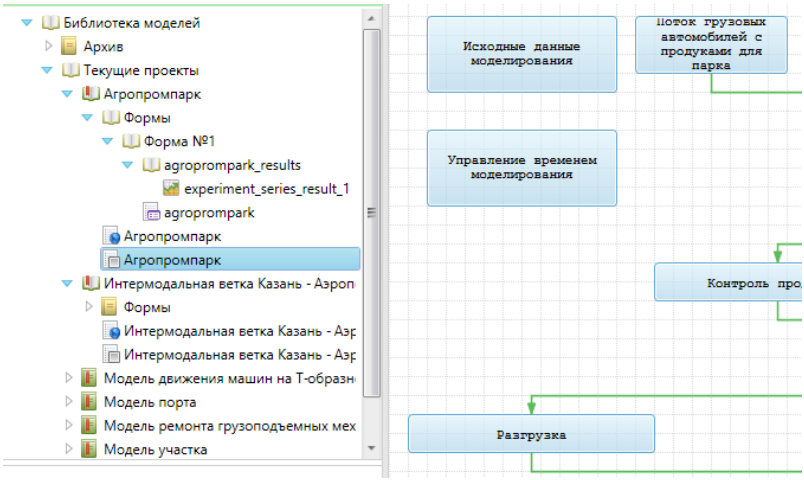

Рис.13. Окно расширенного редактора моделей

Текстовый редактор GPSS моделей

В состав расширенного редактора входит новый многофункциональный текстовый редактор GPSS моделей.

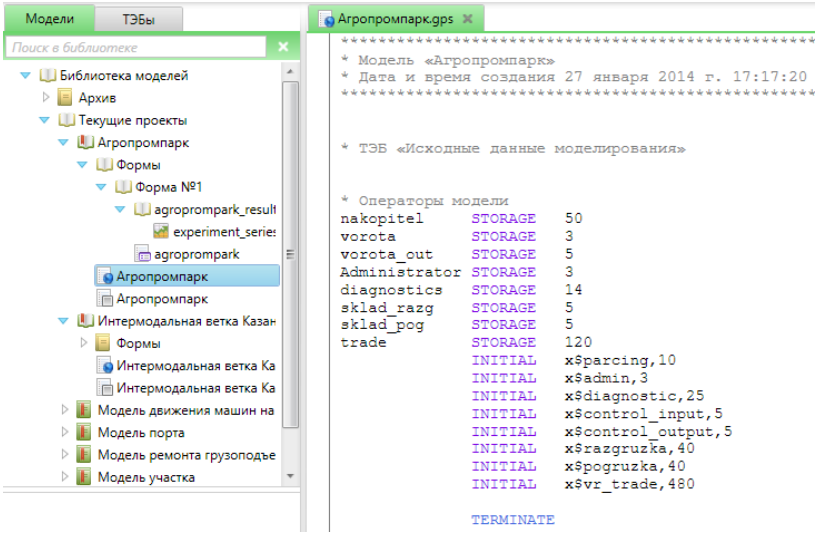

Рис.14. Окно расширенного текстового редактора моделей

Для повышения наглядности текста модели, наименования блоков и команд, ключевые слова PLUS, комментарии и строковые лексемы выделяются различными цветами. Команды и блоки, их аргументы и комментарии, а также PLUS процедуры автоматически выравниваются.

В редактор встроена система распознавания объектов модели. Ввод текста модели сопровождается контекстной подсказкой, которая отображает описание редактируемого контекста, и список подходящих значений, полученных при анализе модели. Система автоматического документирования позволяет разработчикам описывать любые объекты модели, PLUS процедуры и их аргументы с помощью комментариев. Это описание отображается в контекстной подсказке, и позволяет впоследствии автоматически обрабатывать и создавать интегрированную «сквозную» технологию присвоения имен на других этапах имитационного исследования.

Поддерживается группировка текста, позволяющая объединять и при ненадобности скрывать ненужные строки модели. Кроме того имеется поиск, быстрое подключение связанных файлов и система графического формирования закона распределения и автоматического формирования GPSS функции на её основе.

Редактор самостоятельно обнаруживает структурные ошибки в модели. Операторы с ошибками выделяются красной линией, а сами ошибки отображаются в подсказке.

Редактор позволяет отлаживать модель непосредственно в тексте. Для этой цели создан специальный модуль трассировки. Отладка производится по блокам модели. При этом может отслеживаться перемещение всех, или определённого транзакта, изменение ячеек и матриц в модели. Разработчик может найти нужный транзакт по его номеру, семейству или значению параметра. Также, можно ограничить часть блоков модели, чтобы отладка велась только внутри неё, а не по всей модели.

Средства мониторинга и проведения серий экспериментов

В состав расширенного редактора входит подсистема динамического мониторинга. Динамический мониторинг модели состоит в отслеживании, сохранении и отображении состояния любых стандартных числовых атрибутов (СЧА) модели через определённые интервалы времени. Пользователь, с помощью простых диалогов, задает, что в модели и в какие моменты времени необходимо отслеживать и сохранять. Модель исполняется, в процессе исполнения все значения СЧА накапливаются в базе данных результатов моделирования и в дальнейшем динамика изменения указанных СЧА может быть проанализирована пользователем. Мониторинг существенно повышает возможности пользователя при отладке модели и дает возможности для качественного анализа результатов моделирования.

В рамках имитационного исследования расширенный редактор позволяет проводить одиночные эксперименты, и серии экспериментов. Редактор обеспечивает множество возможностей для выделения факторов и показателей модели, организации последовательного автоматического исполнения серий, вывода результатов экспериментов и серий в удобном для анализа виде (таблицы, графики, анимация). Все результаты экспериментов сохраняются в базе данных результатов моделирования и доступны пользователю в любой момент исследования.

Анимация

Текущая версия редактора форм поддерживает 2D анимацию и позволяет создавать анимационные формы. Исследователь может сформулировать и графически описать сценарий анимации, используя карты, схемы и другие изображения в качестве подложки и строя возможные траектории движения объектов (транзактов) в сформированном пространстве. При этом подложка и дополнительные графические построения масштабируются и управляются пользователем. После завершения анимации, система формирует анимационный ролик, который

имеет встроенные средства управления просмотра, анализа и документирования.

Анимация позволяет системе выйти на принципиально новый уровень в презентабельности и глубине анализа результатов. Просматривая анимационные ролики, вы своими глазами видите, как "живет" система в ходе эксперимента, причем в виде, максимально приближенном к реальности. Сразу видны ошибки в логике функционирования модели, если вы сделали что-то не так. Очевидней становится и поиск "узких" мест, вы просто их увидите! Т.е. анимация обладает не только "генеральским" эффектом для демонстрации модели, но и является прекрасным методом подтверждения факта адекватности модели и ускоряет процесс исследования. И, наконец, анимация служит популяризации имитационного моделирования, привлечения внимания к нему.

2.3. Задание второе

По методическому руководству разобранному в пособии рассмотреть процесс создания модели в графическом редакторе GPSS World.

2.4. Методические указания по выполнению второго задания

В качестве примера, была выбрана конвейерная модель, из 2 лабораторной работы.

Рассмотрим оздание модели в расширенном редакторе GPSS, а также создание ее формы и анимированние процесса моделирования.

Опишем поэтапно процесс создания модели конвейерной структуры:

1) Необходимо, естественно, открыть расширенный редактор GPSS, и создать в левом меню, в папке текущие проекты, новый проект. Для примера создадим проект под названием

“Конвейерная модель”. Результаты создания проекта отображены на рисунке 15.

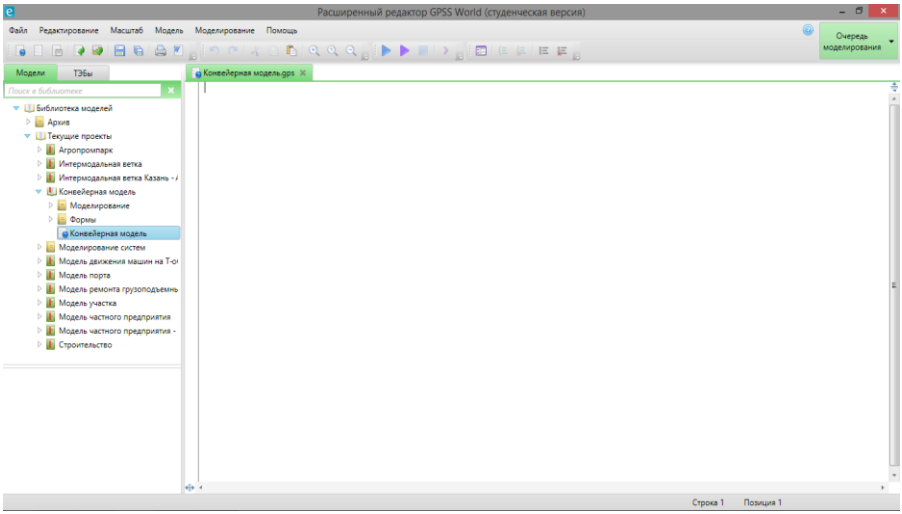

Рис.15. Вид окна расширенного редактора с созданным и открытым в нем проектом

2) После создания проекта модели открывается одноименный файл модели, в котором непосредственно происходит описание модели на языке GPSS. Модель конвейерной вычислительной структуры для примера возьмем следующую:

```
normalFunction  function  RN1,C24
        0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915/.7,1.2/.75,1.3
8/.8,1.6/.84,1.83/.88,2.12/.9,2.3/.92,2.52/.94,2.81/.95,2.99/.96,3.2/.97
,3.5/.98,3.9/.99,4.6/.995,5.3/.998,6.2/.999,7/.9998,8
        normDist  fvariable  FN$normalFunction#1+1
        table1    table   M1,20,10,12
        table2    table   M1,20,10,12

        generate  V$normDist
```

```
    queue    Queue1
    seize    Processor1
    depart    Queue1
    advance    1,1
    release    Processor1
    queue    Queue2
    seize    Processor2
    depart    Queue2
    advance    1,1
    release    Processor2
    tabulate    table1
    queue    Queue3
    seize    Processor3
    depart    Queue3
    advance    1,1
    release    Processor3
    tabulate    table2
terminate    1
start        1000
```

3) Далее необходимо проверить работоспособность модели, для этого необходимо запустить процесс моделирования. Для запуска процесса моделирования можно нажать либо клавишу F5, либо же нажать кнопку в виде треугольника, расположенной в панели инструментов редактора (на рисунке 3 обведена). Как и для обычного редактора результатом правильно построенной модели будет сформированный отчет, изображенный на рисунке 3.

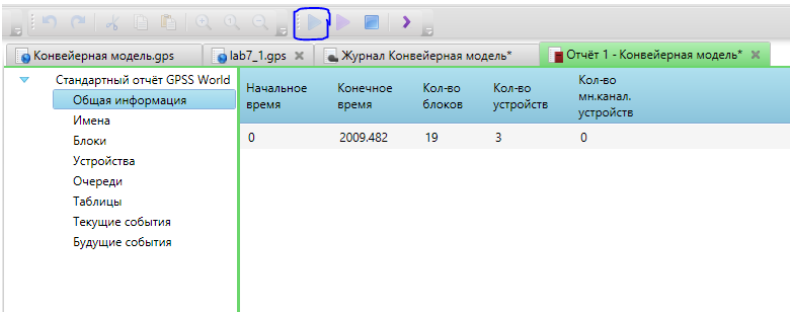

Рис.16. Вид отчета по завершении процесса моделирования

Удостоверившись, что процесс моделирования прошел успешно, то есть получив отчет, можно переходить к созданию формы модели. Расширенный редактор представляет довольно солидный функционал по созданию форм. Для примера рассмотрим создание анимации процесса моделирования.

3.1) Первоначально необходимо создать форму модели. Для этого необходимо в меню “Модель” выбрать пункт “Создать форму”, после чего необходимо ввести название для формы. После ввода названия и его подтверждения будет запущен редактор форм, вид которого представлен на рисунке 17.

3.2) Для создания анимации необходимо в меню “Формы” выбрать пункт “Анимационная форма”. Первым делом необходимо установить фон для модели, делается это с помощью кнопки “Изображение” (обведена на рисунке 5). Необходимо указать область, которую займет изображение и выбрать непосредственно изображение, с помощью кнопки импорт изображения (обведена на рисунке 18). Установим фоном для нашей модели рисунок 1. Результат данных действий отображен на рисунке 18.

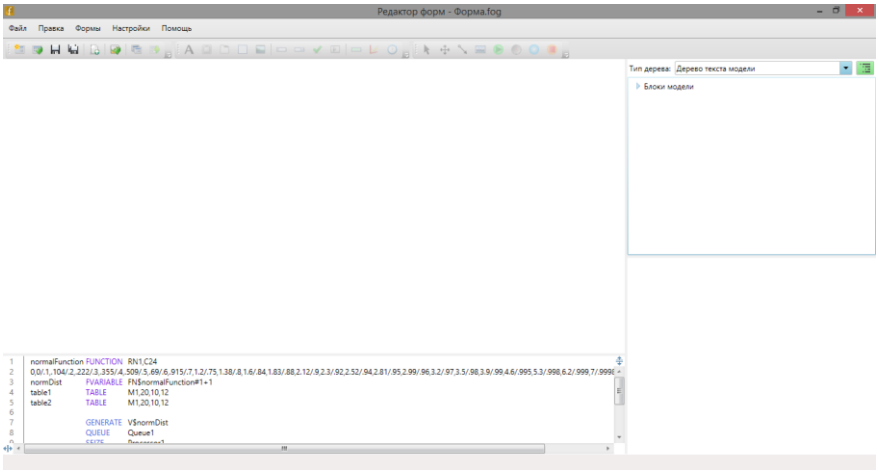

Рис.17. Вид редактора форм

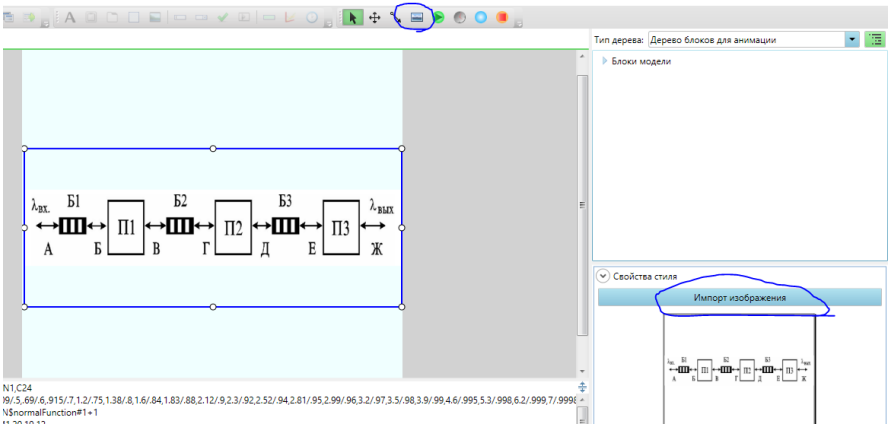

Рис.18. Установленный фон для анимации

3.3) Далее необходимо разместить на фоне все блоки модели, которые производят, какие-либо операции с транзактами, и создать связи между этими блоками. Для создания процесса анимации доступны блоки генерации транзактов (зеленый круг), задержи транзактов (круг с часовыми стрелками), блоки прибо-

ров (синий круг) и блоки вывода транзактов из системы (красный круг). Все эти блоки можно поместить на модель либо перетаскиванием из панели инструментов, либо из перечня блоков модели в списке, расположенном справа вверху. На рисунке 19 приведен пример модели с уже размещенными блоками и связями.

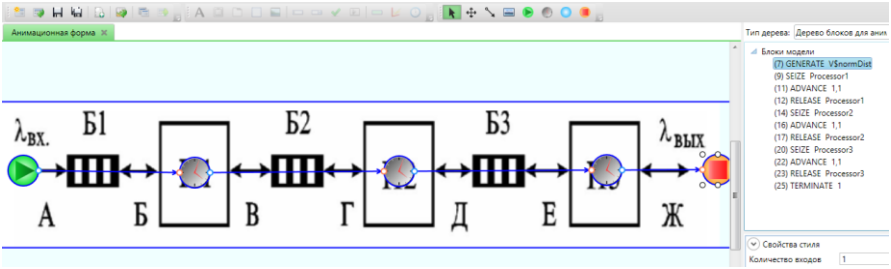

Рис. 19. Добавленные на форму блоки модели.

4. Следующим шагом будет являться создание связей между блоками. Для этого служит инструмент “Путь”, находится возле кнопки добавления изображения. Процесс создания пути выглядит следующим образом: вы нажимаете кнопку, нажимаете точку на блоке, из которой выходит транзакт (она всегда синяя), а затем нажимаете на точку, в которую блока в которую входит транзакт (оранжевая), создается связь. Все линии должны быть обязательно сплошными, если линия не сплошная необходимо нажать на нее, и выбрать в меню, которое находится справа внизу источник задержки. Если транзакт будет задержан каким-либо блоком задержки, в сторону которого ведет линия выберете пункт “Следующий блок ADVANCE”, иначе необходимо выбрать предыдущий блок или вовсе указать его самостоятельно. Вид модели, со всеми связями так же представлен на рисунке 19.

5. Далее требуется настроить форму таким образом, чтобы в процессе моделирования происходил сбор данных для создания анимации. Для этого в меню “Формы”, пункт “Настройка мониторинга”, необходимо поставить галочку на пункте “Собирать данные для анимации” (обведено на рисунке 20).

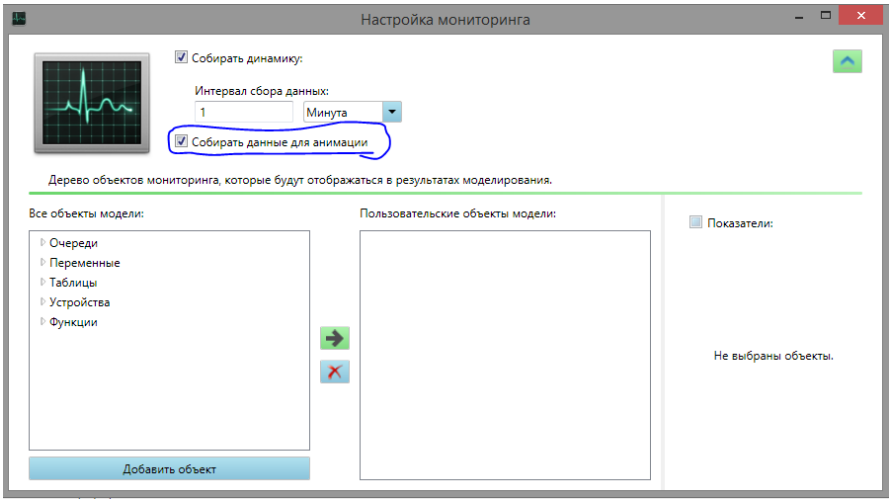

Рис. 20. Вид окна настройки мониторинга

6. После настройки можно переходить к проверке работоспособности формы и созданной анимации. Для запуска формы необходимо в меню “Формы”, необходимо выбрать пункт “Проверка формы” или нажать F5, будет запущена форма модели, вид которой представлен на рисунке 21.

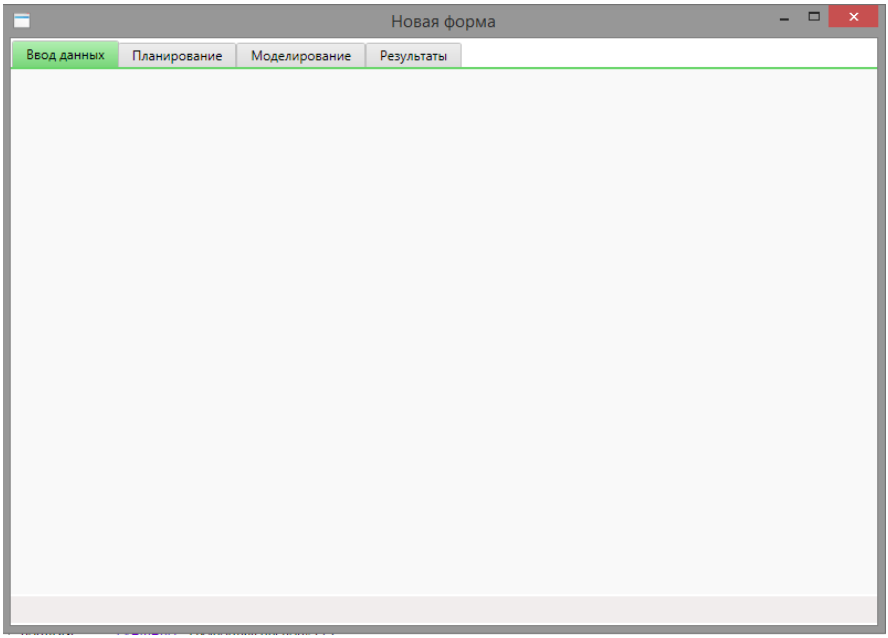

Рис. 21. Вид формы модели

7. Нас для начала будет интересовать вкладка “Моделирование”, на этой вкладке мы можем запускать процесс моделирования для нашей модели. Моделирование запускается с помощью кнопки “Начать моделирование”. Запустим процесс моделирования, и получим результаты (рисунок 22).

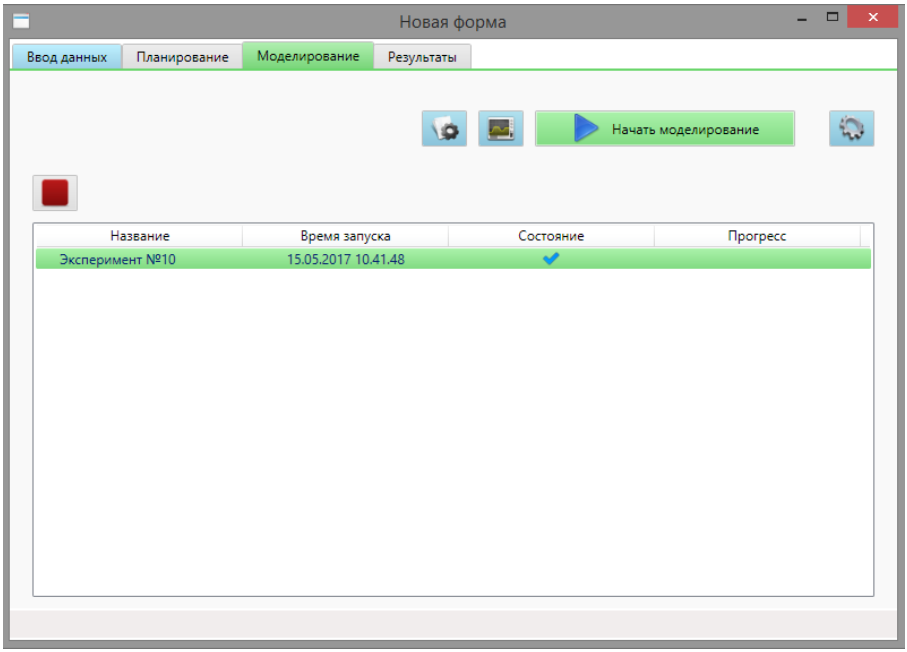

Рис.22. Вид вкладки “Моделирование” с отчетом об одном успешном эксперименте

8. Перейдем во вкладку “Результаты”. Как видно, данная вкладка предоставляет функционал для просмотра всех экспериментов, проведенных с моделью, а также всевозможные виды отчетности по ним (рисунок 23).

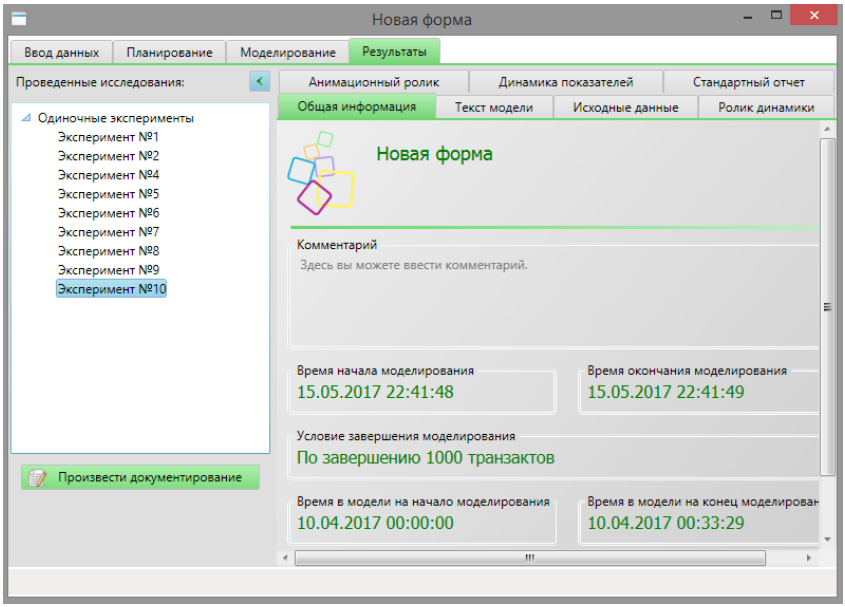

Рис. 23. Вид вкладки “Результаты”

9. Так как нашей целью было анимированние, перейдем во вкладку “Анимационный ролик”. На этой вкладке мы можем запустить анимацию процесса проведения эксперимента. Транзактами будут является красные треугольники, можно увидеть, как они задерживаются в блоках задержки, или как они создаются в блоках создания и удаляются в блоках вывода, таким образом можно выделить особенность проектирования процесса анимации – блоки модели необходимо ассоциировать с объектами фона, для того чтобы сам процесс анимации нес какой-то смысл. В нашем случае можно увидеть, как транзакты задерживаются в процессорах конвейерной структуры (рисунок 24, транзакт обведен).

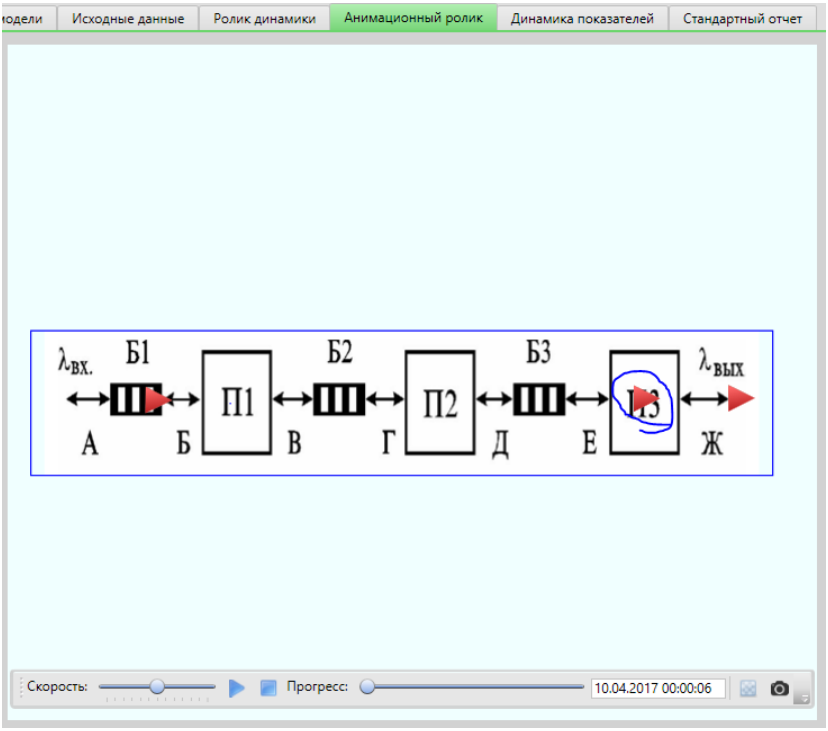

Рис. 24. Анимация эксперимента с моделью конвейерной структуры с выделенным транзактом, задержанным в третьем процессоре на обработку

10. Если в процессе проигрывания анимации четко видны перемещения транзактов, то процесс анимированния можно считать успешным. В противном случае необходимо удостовериться в правильности настроек мониторинга (пункт 5) и в размещении блоков модели и путях между ними.

3. Задание на лабораторные работы 7-9

Задание лабораторной работы № 7: Построить модель в графическом редакторе GPSS World, соответственно варианту задания, полученной во 2-ой лабораторной работе от преподавателя.

Задание лабораторной работы № 8: Построить модель в графическом редакторе GPSS World, соответственно заданию, полученному в 3-ей лабораторной работе от преподавателя.

Задание лабораторной работы № 9: Построить модель в графическом редакторе GPSS World, соответственно заданию, полученному в 4-ей лабораторной работе от преподавателя.

4. Оформить отчет.

Указания по оформлению отчета и контрольные вопросы по выполненной работе

4.1. Отчет должен содержать

1. Описание шагов составления имитационной модели.
2. Анимацию эксперимента с моделью.
3. Исходный текст модели и результаты моделирования.
4. Основные выводы по результатам моделирования.

ЗАКЛЮЧЕНИЕ

Язык GPSS создавался в результате раннего опыта, накопленного при разработке имитационных моделей. И в последующее время дальнейшая эволюция GPSS, все больше основывалась на широком использовании опыта, совмещая это с извлечением выгоды из практических приложений. Классы GPSS и связанные с ними методы. Наиболее важные классы объектов (транзакты) и другие классы объектов (например, устройства, памяти и логические ключи) и их свойства (блоки) отображаются в языке имитационного моделирования элементами, которые используются в реальных вычислительных сетях и в других дискретных системах. В действительности - GPSS приспосабливается так быстро и легко, так хорошо представляет реальные дискретные системы и сделан так вовремя в истории вычислительной техники и имитационного моделирования, что все это обусловило его долголетие.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

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

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

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

4 Советов Б.Я. Моделирование систем. Практикум: Учеб. пособие для вузов/ Б.Я. Советов, С.А. Яковлев.- 2-е изд., перераб. И доп.- М.: Высш. шк., 2003,-295 с.

5 Советов Б.Я., Яковлев С.А. Моделирование систем: Учеб. для вузов – 3-е изд., перераб. и доп.- М.: Высш. шк., 2001. – 343 с.: ил.

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

8 А.Г. Королёв. Моделирование систем средствами Object GPSS. Практический подход в примерах и задачах. Учебное пособие. Луганск: Изд-во Восточно-украинского нац. ун-та, 2005.- 307 с.

9 Каштанов Д.И. Документация на комплекс “ВиРОМ 2.0”. – Владивосток, 2002.

10 Воробьев Э.И. Моделирования на GPSS: учеб. пособие/ Э.И. Воробьев, О.Л. Щеглова. Воронеж: ВГТУ. 2006. - 99 с.

ОГЛАВЛЕНИЕ

Учебно-методическое издание

Воробьев Эдуард Игоревич
Тишуков Борис Николаевич

МОДЕЛИРОВАНИЕ ПРОЦЕССОВ И СИСТЕМ НА GPSS

В авторской редакции

Компьютерный набор Б.Н. Тишукова

Подписано к изданию

Объем данных 4,39 МБ

ФГБОУ ВО «Воронежский государственный технический университет»
394026 Воронеж, Московский просп., 14