ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

УПРАВЛЕНИЕ ДИСТАНЦИОННОГО ОБУЧЕНИЯ И ПОВЫШЕНИЯ КВАЛИФИКАЦИИ

Кафедра «Вычислительные системы и информационная безопасность»

Методические указания к контрольной работе по дисциплине

«Теория информационных процессов и систем»

Автор: Чуйкова Е.Н

Ростов-на-Дону, 2013

Оглавление

ВВЕДЕНИЕ

Целью комплекса контрольных работ является освоение принципов моделирования дискретной системы в программной среде GPSS, получение навыков проведения имитационного эксперимента и анализа результатов моделирования.

1. ПОНЯТИЕ ИМИТАЦИОННОЙ МОДЕЛИ

Модель – описание системы. Моделирование систем выполняется с целью их исследования. При этом эксперименты проводятся не с самой системой, а с представляющей её моделью.

Процесс моделирования начинается с определения цели разработки модели, на основе которой затем устанавливаются границы системы и необходимый уровень детализации моделируемых процессов. В описание системы также должны быть включены критерии эффективности функционирования системы и оцениваемые альтернативные решения, которые могут рассматриваться как часть модели или как ее входы. Оценки же альтернативных решений по заданным критериям эффективности рассматриваются как выходы модели. Обычно оценка альтернатив требует внесения изменений в описание системы и перестройки модели. Поэтому процесс построения модели является итеративным. На основе полученных оценок альтернатив вырабатываются рекомендации по улучшению моделируемой системы.

Имитационной моделью называется логико-математическое описание системы, которое может быть исследовано в ходе проведения экспериментов на ЭВМ. После разработки имитационной модели с ней проводятся машинные эксперименты, которые позволяют сделать выводы о поведении системы. Таким образом, имитационные модели могут использоваться для проектирования, анализа и оценки функционирования систем. Ключевым моментом при имитационном моделировании является выделение и описание состояний системы. Система характеризуется набором переменных, каждая комбинация значений которых описывает ее конкретное состояние. Путем изменения значений переменных можно имитировать переход системы из одного состояния в другое. Имитационное моделирование – это представление поведения системы посредством продвижения ее от одного состояния к другому в соответствии с определенными правилами. Изменение состояния системы представляет собой событие. Моделирование состоит в динамическом отображении изменений состояния си-

стемы с течением времени путем обработки упорядоченных во времени событий.

В качестве примера моделируемой системы можно рассмотреть кассира в банке, обслуживающего клиентов. Клиенты прибывают в банк в случайные моменты времени, ожидают обслуживания в очереди, если кассир занят, обслуживаются и покидают банк. В этой системе кассир выступает в качестве «обслуживающего прибора», а клиенты – в качестве заявок на обслуживание. Можно построить имитационную модель описанной системы, провести эксперименты с этой моделью на ЭВМ (т.е. проимитировать ее функционирование в течение некоторого интервала времени) и вычислить характеристики функционирования системы, такие как коэффициент загрузки прибора (кассира), средняя длина очереди заявок к прибору, среднее время пребывания заявки в очереди и т.п. Эти характеристики представляют собой результаты имитационного моделирования, которые затем анализируются и по результатам анализа вырабатываются рекомендации по улучшению работы системы. Например, если коэффициент загрузки прибора высок (близок к 100%) и время пребывания заявок в очереди большое, то можно сделать вывод, что один прибор не справляется с нагрузкой, и следует добавить в систему еще один обслуживающий прибор. Наоборот, если загрузка прибора не более 50%, то система недогружена, следовательно, не эффективна, и возможно ее следует вообще ликвидировать.

2.ПРИНЦИПЫ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ В ПРОГРАММНОЙ СРЕДЕ GPSS

2. 1. Основы организации и функционирования GPSS

Имитационная модель сложной дискретной системы представляется в виде описания ее элементов и логических правил их взаимодействия. Структура моделируемой системы описывается конечным множеством абстрактных элементов, называемых объектами GPSS. Взаимодействие элементов системы описывается ограниченным набором логических операций.

Процесс функционирования систем состоит в последовательном наступлении событий, и язык моделирования должен иметь средства для их реализации. В GPSS для этой цели используются специальные динамические объекты – транзакты, перемещение которых между элементами модели приводит к возникновению событий.

В состав программного комплекса GPSS входят программы, описывающие функционирование выделенного конечного набора объектов, и специальная программа-интерпретатор, которая выполняет следующие функции:

- обеспечение заданных программистом маршрутов движения транзактов;
- планирование событий, происходящих в модели, и выполнение их в нарастающей временной последовательности;
- регистрацию статистической информации о функционировании модели;
- продвижение модельного времени в процессе моделирования системы.

Объекты GPSS подразделяются на типы и категории. Соответствие между категориями объектов и представляющими их типами приведено в табл.1.

Таблица 1

Категории и типы объектов GPSS

Категории GPSS	Типы объектов GPSS
Динамическая	Транзакты
Операционная	Блоки
Аппаратная	Устройства, памяти, ключи
Вычислительная	Переменные, функции
Статистическая	Очереди
Запоминающая	Ячейки

Объекты динамической категории – транзакты – являются средством реализации событий в моделируемой системе. Продвигаясь по фиксированной структуре, представляющей собой совокупность объектов других категорий, транзакты производят в модели определенные действия.

Объекты операционной категории – блоки – задают логику функционирования модели и определяют пути движения транзактов между объектами аппаратной категории.

Объекты аппаратной категории – это абстрактные элементы (устройства, памяти, логические ключи), из которых составляется структура реальной системы. Воздействуя на эти объекты, транзакты могут изменять их состояние и влиять на движение других транзактов.

Объекты вычислительной категории используются для описания таких ситуаций, когда связи между компонентами моделируемой системы выражаются в виде аналитических или логических соотношений.

Объекты статистической и запоминающей категорий используются для сбора информации о поведении системы.

Каждому типу объекта соответствуют атрибуты, описывающие его состояние в данный момент времени. Значения атрибутов могут быть арифметическими или логическими. Атрибуты, которые доступны программисту, называются стандартными числовыми атрибутами (СЧА) и стандартными логическими атрибутами (СЛА). СЧА и СЛА основных типов объектов GPSS приведены соответственно в табл. 2 и 3.

Таблица 2

Стандартные числовые атрибуты объектов GPSS

Тип объекта	СЧА	Описание
Транзакты	PR Pj MPj ML$1	Приоритет транзакта (0…127) Значение j-го параметра транзакта Время с момента входа транзакта в блок MARK, сохраняемое в параметре Pj Время пребывания транзакта в модели
Памяти (много-канальные устройства)	Sj Rj SRj SMj SAj SCj STj	Ёмкость памяти j Свободный объём памяти j Коэффициент использования памяти j Максимальное заполнение памяти j Среднее заполнение памяти j Число входов в память j Среднее время пребывания транзакта в памяти j
Устройства	Fj FRj FCj FTj	Состояние устройства j: 0 - если устрой-ство свободно;1 - если устройство занято Коэффициент использования устройства j Число входов в устройство j Среднее время использования устройства j одним транзактом
Очереди	Qj QMj QAj QCj QTj QXj	Текущая длина очереди j Максимальная длина очереди j Средняя длина очереди j Число входов в очередь j Среднее время пребывания в очереди j, включая нулевые входы Среднее время пребывания в очереди j, без нулевых входов
Ячейки	XHj	Значение ячейки j
Блоки	N W	Количество входов в блок Текущее содержимое
Переменные	Vj	Значение переменной j
Генераторы	RNj	Значение генератора j
Функции	FNj	Значение функции с номером j

Таблица 3

Стандартные логические атрибуты объектов GPSS

Тип объекта	СЛА	Описание
Памяти (многоканальные устройства)	SEj SNEj SFj SNFj	Память j пуста Память j не пуста Память j заполнена Память j не заполнена
Устройства	FUj NUj FIj FNIj	Устройство j занято Устройство j свободно Устройство j прервано Устройство j не прервано
Логические ключи	ISj	Значение ключа j

Модель на языке GPSS представляет собой последовательность блоков, определяющую логическую структуру моделируемой системы. Динамические элементы – транзакты – продвигаются в процессе имитации последовательно от блока к блоку. Каждый блок можно рассматривать как точку, в которой происходит обращение к некоторой подпрограмме. Движение транзакта в модели продолжается до тех пор, пока не произойдет одно из следующих событий:

- транзакт входит в блок, функцией которого является задержка транзакта на некоторое определенное моделью время;
- транзакт входит в блок, функцией которого является удаление транзакта из модели;
- транзакт пытается войти в следующий блок, но блок отказывается его принять. В этом случае транзакт остается в текущем блоке. Позднее он будет повторять свою попытку войти в следующий блок. Когда условия в модели изменятся, одна из таких попыток окажется успешной. После этого транзакт продолжит свое перемещение в модели.

Содержательное значение транзактов определяет разработчик модели. Он устанавливает некоторую аналогию между транзактами и реальными элементами моделируемой им системы. Например, при моделировании магазина транзактами выступают покупатели, при моделировании банка – клиенты.

2.2. Структура программы на языке GPSS

Программа на GPSS конструируется из блоков, каждый из которых выполняет определенные действия в момент входа в него транзактов. Кроме блоков модель может содержать карты описания используемых в модели объектов и системные карты, необходимые для прогона модели.

Каждый блок имеет следующий формат:

[МЕТКА] ОПЕРАЦИЯ A, B, C, D, E, F, G [; КОММЕНТАРИЙ]

Метка должна помещаться с первой позиции и содержать до 5 алфавитно–цифровых символов, первый из которых должен быть буквой. Если в первой позиции находится символ "*", то вся строка считается комментарием.

В поле операции записывается имя блока. Это поле может начинаться со второй позиции и должно быть отделено от метки хотя бы одним пробелом.

Аргументы отделяются от операции не менее, чем одним пробелом. Их разделяют запятыми и именуют в порядке расположения: A, B, C,… Точка с запятой означает начало комментария в текущей строке.

Имена используемых в модели объектов могут быть цифровыми и символическими. Если используются символические имена объектов, то их необходимо определить с помощью карты **EQU**, которая имеет следующий формат:

name EQU num, type ,

где **name** – символическое имя объекта; **num** – номер объекта, задаваемый целым положительным числом; **type –** мнемоническое обозначение типа объекта: **F –** приборы; **L –** логические ключи; **S –** памяти; **V –** переменные; **XH –** ячейки; **Q –** очереди; **Z –** функции.

Системные карты используются для управления процессом прогона модели. Если разработчик намерен выполнить прогон модели, то в начале программы должна стоять системная карта **SIMULATE**. При отсутствии карты **SIMULATE** интерпретатором GPSS осуществляется проверка синтаксических ошибок в программе, но прогон модели не выполняется.

Окончание процесса моделирования происходит при обнулении системной переменной GPSS, называемой счетчиком завершения. Начальное значение счетчика завершения устанавливается картой **START**, а уменьшение его значения происходит

при входе транзактов в блоки **TERMINATE**.

Карта **START** имеет следующий формат:

START A ,

где **A** – начальное значение счётчика завершения.

Для определения конца программы используется системная карта **END**. Когда интерпретатор GPSS доходит до этой карты, управление передается программе отображения результатов моделирования.

В GPSS для получения случайных величин используется 9 генераторов случайных чисел. Генераторы порождают случайные числа, равномерно распределенные в диапазоне [0, 1]. Значение генератора содержится в стандартном числовом атрибуте **RN**.

Структура программы на языке GPSS имеет следующий вид:

SIMULATE
<карты описания объектов>
<блоки GPSS >
START N
END

Карты описания объектов и основные блоки языка GPSS рассматриваются в следующих разделах.

2.3. Основные блоки GPSS

2.3. 1. Блоки создания, уничтожения и задержки транзактов

Для ввода транзактов в модель используется блок **GENERATE**, который имеет 6 параметров и записывается в следующем виде:

GENERATE A, B, C, D, E, F

После выхода из блока **GENERATE** транзакты направляются в следующий за ним блок.

Параметры блока **GENERATE** имеют следующее назначение:

A – среднее значение периода поступления транзактов;

B – если в поле **B** указано число, то интервал между поступлением транзактов в модель является случайным числом, равномерно распределенным в диапазоне от (**A – B**) до (**A + B**). Если параметр **B** является функцией, то этот интервал определяется произведением параметра **A** на значение функции, заданной параметром **B**.

C – время появления первого транзакта;

D – общее число транзактов, которые войдут в модель через данный блок **GENERATE**;

E – приоритет генерируемых транзактов (0 – низший, 127 – высший);

F – число параметров транзактов.

По умолчанию принимается: **A** = 1, **B** = 0, **C** = 0, **D** – бесконечность, **E** = 0, **F** = 12.

Поскольку блок **GENERATE** является источником транзактов, недопустимо направлять в него транзакт.

Пример: **GENERATE 25, 5, 50 ;** Генерация транзактов с интервалом 20 …30 единиц времени, первый из которых поступает в момент 50.

Удаление транзактов из модели происходит с помощью блока **TERMINATE**, имеющего следующий формат:

TERMINATE A

При входе в блок **TERMINATE** транзакт удаляется из модели. Если параметр **A** не задан, то счетчик завершения не изменяется. В противном случае его значение уменьшается на величину, равную значению параметра **A**.

Рассмотрим пример, в котором блок **TERMINATE** и карта **START** используются для управления процессом моделирования.

Пусть в качестве единицы времени выбрана 1 минута и необходимо промоделировать поведение системы в течение 8 часов. Это можно сделать, включив в программу следующий фрагмент:

```
GENERATE   480
TERMINATE   1
```

При этом во всех прочих блоках **TERMINATE** в модели операнд **A** не указывается, так что завершение моделирования, определяемое счетчиком завершения, не будет зависеть от других блоков **TERMINATE**. В карте **START** в качестве операнда **A** указывается **1** (начальное значение счетчика завершения). Таким образом, в момент модельного времени 480 транзакт выйдет из указанного выше блока **GENERATE**, сразу перейдет в следующий блок **TERMINATE**. Так как операнд этого блока содержит единицу, то из счетчика завершения вычитается 1. Это уменьшает значение счетчика от 1 до 0. В результате интерпретатор прекращает моделирование.

Блок **ADVANCE** выполняет задержку транзакта и имеет 2 параметра:

ADVANCE A, B

Если параметр **B** является числом, то при входе в блок **ADVANCE** транзакт задерживается в нем на время, равномерно распределенное в интервале от (**A – B**) до (**A + B**). Если поле **B** является функцией, то время задержки равно **A · B**.

Пример: **ADVANCE 20, 5 ;** Задержка транзакта в блоке на 15 …25 единиц времени.

2.3.2. Блоки изменения СЧА транзактов

Для изменения значений параметров транзактов используется блок **ASSIGN**, который имеет следующий формат:

ASSIGN A, B

В поле **A** указывается номер изменяемого параметра транзакта. В поле **B** – новое значение параметра транзакта. Если за полем **A** следует знак "+" или "–", то значение поля **B** соответственно прибавляется или вычитается из значения параметра транзакта, номер которого указан в поле **A**. В противном случае в параметр, номер которого указан в поле **A**, записывается значение, указанное в поле **B**.

Блок MARK осуществляет запись значения таймера абсолютного модельного времени в параметр транзакта, номер которого указан в поле A:

MARK A

Данный блок используется для определения времени прохождения транзактом определённого фрагмента модели. Для этого в начальной точке фрагмента необходимо записать блок MARK. Тогда при входе транзакта в начальную точку фрагмента время входа будет записано в параметр транзакта, номер которого указан в поле A блока MARK. При достижении транзактом конечной точки фрагмента время прохождения фрагмента будет определяться значением СЧА транзакта MPj, где j – равно значению поля A блока MARK.

2.3.3. Блоки, изменяющие маршрут движения транзактов

В эту группу входят блоки **TEST**, **GATE**, **TRANSFER**.

Блок **TEST** имеет следующий формат:

TEST_op A, B, C

Блок **TEST** выполняет сравнение аргументов, записанных в полях **A** и **B** в соответствии со значением внутреннего операнда **«op»**. Если условие сравнения выполняется, то транзакт проходит в следующий за блоком **TEST** блок. В случае невыполнения условия сравнения транзакт направляется в блок, метка ко-

торого указана в поле **C**. Если поле **C** не задано, то транзакт задерживается в блоке **TEST** до тех пор, пока условие сравнения не будет выполнено. Внутренний операнд **«op»** может обозначать следующие условия сравнения: **E** – равно; **NE** – не равно; **L** – меньше; **LE** – меньше или равно; **G** – больше; **GE** – больше или равно.

Пример: **TEST_E P5, 7 , A3 ;** Если пятый параметр транзакта равен 7, транзакт проходит в следующий за блоком TEST блок. В противном случае транзакт перейдёт в блок, имеющий метку A3.

Блок **GATE** имеет следующий формат:

GATE_op A, B

Блок **GATE** определяет состояние устройств, памятей или логических ключей и сравнивает определённое состояние объекта со значением внутреннего операнда **«op»**. В поле **A** указывается имя объекта, в поле **B** – метка альтернативного блока. Если состояние объекта совпадает с состоянием, определяемым значением внутреннего операнда, транзакт проходит в следующий за блоком **GATE** блок. В случае, когда состояние объекта отличается от состояния, определяемого значением внутреннего операнда **«op»**, транзакт следует в блок, метка которого указана в поле **B**. Если поле **B** пусто, то транзакт задерживается в блоке **GATE** до совпадения состояния проверяемого объекта со значением внутреннего операнда **«op»**.

Возможные значения внутреннего операнда **«op»** определяются наборами **СЛА**, соответствующими устройствам, памятям, логическим ключам: **U** – устройство занято; **NU** – устройство не занято; **I** – устройство прервано; **NI** – устройство не прервано; **SF** – память заполнена; **SNF** – память не заполнена; **SE** – память пуста; **SNE** – память не пуста; **LR** – ключ выключен; **LS** – ключ включен.

Пример: **GATE_SF 2 ;** Если память 2 заполнена, то транзакт проходит в следующий за блоком GATE блок, в противном случае транзакт задерживается в блоке GATE до момента заполнения памяти 2.

Блок **TRANSFER** имеет следующий формат:

TRANSFER A, B, C

После входа в блок **TRANSFER** транзакт направляется в блок, определяемый в соответствии с режимом передачи значением поля **A**. Поле **A** может содержать следующие значения:

1) пробел – транзакт передаётся в блок, метка которого указывается в поле **B**;

2) " **.** " – статистический режим; в поле **A** указывается вещественное число в интервале от 0 до 1, определяющее вероятность перехода в блок **C**, дополнение этого числа до единицы определяет вероятность перехода в блок **B**;
3) **BOTH** – транзакт последовательно пытается войти в блок, метка которого указана в поле **B**, затем в блок, метка которого указана в поле **C**, до тех пор, пока один из этих блоков станет доступным для транзакта;

Примеры: **TRANSFER , BACK ;** Транзакт передаётся в блок с меткой BACK.

TRANSFER . 7, FAC1, FAC2 ; После входа в блок TRANSFER транзакт с вероятностью 0.7 будет передан в блок с меткой FAC2 и с вероятностью 0.3 – в блок с меткой FAC1.

TRANSFER BOTH , A1 , A2 ; Транзакт последовательно пытается войти в блоки с метками A1 и A2 до тех пор, пока один из этих блоков не станет доступным для транзакта.

2.3.4. Блоки, описывающие работу устройств

Устройства используются для моделирования объектов, которые в данный момент времени могут обслуживать одно требование. При появлении нового требования в процессе обслуживания оно должно либо ожидать своей очереди обслуживания, либо прервать протекающее обслуживание до его завершения. В GPSS элементами, которые требуют обслуживания, являются транзакты, поэтому появление требований занятия устройств связано с движением транзактов по модели. При моделировании работы устройств в случае, когда новое требование ожидает окончания обслуживания предыдущего, используются блоки **SEIZE** (занять устройство) и **RELEASE** (освободить устройство). Блок **SEIZE** имеет следующий формат:

SEIZE A

При входе в блок **SEIZE** транзакт пытается занять устройство, номер которого указан в поле **A**. Если данное устройство занято или прервано, то транзакт задерживается перед блоком **SEIZE** до освобождения устройства.

Блок **RELEASE** имеет следующий формат:

RELEASE A

При входе в блок **RELEASE** освобождается устройство, номер которого указан в поле **A**. Устройство становится доступным для других транзактов.

При моделировании работы устройств, когда новое требование пытается прервать обслуживание предыдущего, используются блоки **PREEMPT** (захватить устройство) и **RETURN** (вернуть устройство ранее прерванному требованию).

Блок **PREEMPT** имеет следующий формат:

PREEMPT A

При входе в блок **PREEMPT** транзакт прерывает работу устройства, номер которого указан в поле **A**, и получает данное устройство в своё использование, если оно не было прервано другим транзактом. Если предыдущий транзакт захватил устройство через блок **PREEMPT**, данный транзакт блокируется до момента освобождения устройства предыдущим транзактом.

Блок **RETURN** имеет следующий формат:

RETURN A

При входе транзакта в блок **RETURN** снимается прерывание с устройства, которое было прервано этим же транзактом при его входе в блок **PREEMPT**. Номер устройства, с которого снимается прерывание, указывается в поле **A**. Снятие прерывания должно быть осуществлено тем же транзактом, который вызвал прерывание. Если устройство было занято до прерывания другим транзактом, то прерванный транзакт после снятия прерывания вновь занимает данное устройство.

После окончания моделирования выдается стандартная статистика по использованию устройств.

2.3.5. Блоки, описывающие работу многоканальных устройств (памятей)

Памяти используются для моделирования однородных параллельно работающих устройств. Память характеризуется ёмкостью, определяющей количество каналов параллельного обслуживания поступающих требований, возникновение которых связано с движением транзактов в модели.

Памяти, также как и одноканальные устройства, занимаются и используются транзактами, но в отличие от одноканальных устройств при появлении нового требования в процессе обслуживания оно всегда ожидает своей очереди обслуживания и не может прервать протекающие в памяти процессы обслуживания до их завершения.

При использовании в модели памятей их необходимо предварительно описать при помощи карты описания памяти **STORAGE**. Карта **STORAGE** может быть записана в следующих двух формах:

1) **<имя> STORAGE <ёмкость памяти>** ,

где **<имя>** - имя или номер памяти, записываемое обязательно с первой позиции при наборе текста модели.

Пример: **3 STORAGE 12 ;** описание памяти с номером 3 ёмкостью 12 единиц.

2) **STORAGE <номер памяти >, <ёмкость >**

Пример: **STORAGE 1, 5 ;** описание памяти с номером 1 ёмкостью 5 единиц.

При моделировании работы памятей используются блоки **ENTER** (войти в память) и **LEAVE** (выйти из памяти).

Блок **ENTER** имеет следующий формат:

ENTER A, B

При входе в блок **ENTER** транзакт пытается занять в памяти, имя которой указано в поле **A**, количество каналов, заданных в поле **B**. Поле **B** не является обязательным и по умолчанию принимает значение 1. Если количество свободных каналов в памяти достаточно (т.е. больше или равно значению поля **B**), то транзакт входит в блок **ENTER**, уменьшая при этом количество свободных каналов памяти на величину, указанную в поле **B**. В противном случае транзакт задерживается перед входом в блок **ENTER** до тех пор, пока количество свободных каналов в памяти станет достаточным для входа в неё данного транзакта.

Блок **LEAVE** имеет следующий формат:

LEAVE A, B

При входе в блок **LEAVE** в памяти, имя которой указывается в поле **A**, освобождается количество каналов, равное значению поля **B**. Поле **B** не является обязательным и по умолчанию принимает значение 1.

После окончания моделирования выдается стандартная статистика по использованию памятей.

2.3.6. Блоки, описывающие очереди

Транзакты в процессе движения могут задерживаться перед блоками, вход в которые в данных условиях невозможен. Примерами таких блоков являются **SEIZE** (если устройство занято), **ENTER** (если память заполнена), **GATE** и **TEST** (если указанные в этих блоках условия не выполняются и альтернативные блоки не указаны). В таких ситуациях перед вышеперечисленными блоками образуются очереди. Для сбора статистики об очередях используются блоки **QUEUE** и **DEPART**. Эти блоки сами по себе не создают очередь, а лишь являются средством ее регистрации. Транзакт одновременно может со- стоять в нескольких очередях.

Блок **QUEUE** регистрирует вход транзакта в очередь и имеет следующий формат:

QUEUE A, B ,

где **A** – номер очереди; **B** – число добавляемых единиц в очередь. По умолчанию **B** = 1. При входе транзакта в блок **QUEUE** текущая длина очереди получает приращение на величину **B**.

Блок **DEPART** регистрирует выход транзакта из очереди и имеет следующий формат:

DEPART A, B ,

где **A** – номер очереди; **B** – число удаляемых единиц из очереди. По умолчанию **B** = 1.

По окончании моделирования по очередям выдается стандартная статистика.

2.3.7. Определение и использование ячеек

Ячейки применяются для хранения значений констант, переменных, СЧА используемых в модели объектов. Для изменения значений ячеек используется блок **SAVEVALUE**, который имеет следующий формат:

SAVEVALUE A, B ,

где **A** – номер ячейки; **B** – значение аргумента.

Если за полем **A** не стоит знак «+» или «-», то значение поля **B** присваивается ячейке, номер которой задает аргумент **A**. Если за полем **A** стоит знак «+» или «-», то значение поля **B** соответственно прибавляется или вычитается из значения ячейки с номером **A**.

Примеры: **SAVEVALUE 1, 12 ;** Ячейке с номером 1 присваивается значение 12;

SAVEVALUE 3-, P1 ; Из ячейки с номером 3 вычитается значение первого параметра вошедшего в блок **SAVEVALUE** транзакта.

Перед началом моделирования ячейки имеют нулевые значения. В случаях, когда требуется установить ненулевые значения некоторых ячеек перед началом моделирования, необходимо использовать карту **INITIAL**, которая имеет следующий формат:

INITIAL Ячейка, Значение

2.3.8. Определение и использование функций

При определении моментов поступления и временных задержек транзактов часто требуется получение значений случайных величин, имеющих распределение, отличное от равномерно-

го. Для формирования таких распределений используются функции. В GPSS имеется два типа функций: непрерывные – *C* и дискретные – *D*. Функция задаётся упорядоченным набором из n пар точек (X_i, Y_i).

Для описания функции используется карта **FUNCTION**, которая имеет следующий формат:

<имя> FUNCTION A, B

где **<имя>** – номер или имя функции; **A** – номер генератора; **B** – тип функции и число используемых для её аппроксимации пар точек (X_i, Y_i).

За картой **FUNCTION** следует одна или несколько карт, содержащих значения пар точек (X_i, Y_i). Карты, содержащие точки (X_i, Y_i), имеют следующий вид:

$X1, Y1/X2, Y2/.../Xi, Yi/.../Xn, Yn$

Обязательно соблюдение порядка $X1 < X2 < ... Xi < ... < Xn$.

Поле **B** определяет тип функции и принимает значение *Cn* для непрерывных и *Dn* – для дискретных функций, где *n* – число пар точек (X_i, Y_i).

Вычисление значений функций осуществляется следующим образом. Из множества {$X1$, … Xi, … Xn} находится минимальное значение Xj, большее или равное значению аргумента функции. Если функция является дискретной, то значением функции будет величина Yj. Если функция является непрерывной, то выполняется линейная интерполяция для пары точек ($X(j-1)$, $Y(j-1)$) и (Xj, Yj). При этом значение функции для аргумента A будет определяться в соответствии со следующим выражением:

$F(A)= Y(j-1)+(A-X(j-1))*(Yj-Y(j-1))/(Xj-X(j-1))$

На рис. 1 представлены примеры непрерывной (а) и дискретной (б) функций, соответствующих одному и тому же набору точек $X1, Y1$; $X2, Y2$; $X3, Y3$; $X4, Y4$.

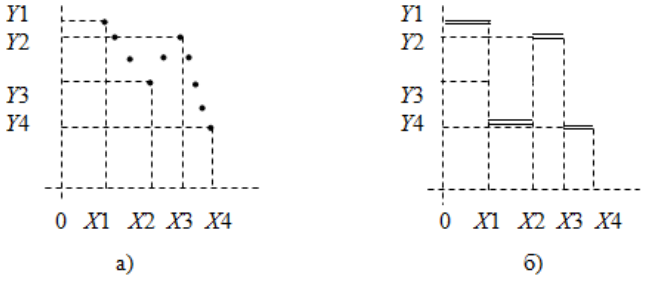

Рис. 1. Графическая интерпретация непрерывных (а) и дискретных (б) функций

Примеры описания функций:

1 FUNCTION 1, C5
0, 0/.2, 2/.5, 7/.8, 15/1, 20

Значения функции 1 распределены случайным образом от 0 до 20 в зависимости от значений генератора случайных чисел RN1.

EXP EQU 1, Z
EXP FUNCTION 1, C12
0, 0/ .2, .22/ .4, .51/ .5, .69/ .7, 1.2/ .8, 1.61/.9, 2.3/. 95, 3./ .99, 4.6/.999,
.9/1, 10.0

Описано экспоненциальное распределение по 12 точкам. Имя функции *EXP* предварительно объявлено картой EQU.

2.3.9. Определение и использование переменных

Переменная является СЧА, определяемым пользователем. В GPSS имеется три типа переменных: целочисленные, булевы и действительные переменные. Ссылка на переменную обозначается Vj, где j – номер переменной. Для использования переменных в программе необходимо предварительно описать выражения, в соответствии с которыми вычисляются их значения. Переменные описывают картой **VARIABLE**, которая имеет следующий формат:

<номер переменной> VARIABLE <выражение>

В выражении используются стандартные числовые атрибуты, соединённые знаками арифметических и логических операций:

(+) – сложение арифметическое или логическое;
(–) – вычитание;
(*) – умножение;
(/) – деление;
(@) – деление по модулю;
(=) – равно;
(! или NE) – не равно;
(<) – меньше;
(>) – больше;
(GE) – больше или равно;
(LE) – меньше или равно;
(& или AND) – логическое И;
(| или OR) – логическое ИЛИ.

В логических операциях ненулевые значения рассматриваются как «ИСТИНА», при этом переменная принимает значение 0 (ЛОЖЬ) или 1 (ИСТИНА).

Примеры: **1 VARIABLE (2+P1)*FN3**

Значением переменной является произведение функций FN3 на сумму параметра 1 и константы 2.

3 variable x1/x2 ; определена переменная 3, значение которой равно отношению содержимого ячеек 1 и 2.

3. ИНТЕРФЕЙС ПРОГРАММНОЙ СИСТЕМЫ GPSS

Программный комплекс GPSS включает моделирующий файл GP.EXE, а также следующие файлы для работы в среде:

1.EXE – запускной файл среды;

GPSS.EXE – среда, написанная на TURBO VISION;

GPSS.CFG – конфигурация среды;

MYCTX.TXT – служебный файл среды для контекстной помощи;

PRIMER.GP – файл – подсказка с примерами моделирования;

DEMOHELP.HLP – файл с информацией по GPSS для HELPa;

MATHCAD.CFG – файл с полным именем файла mcad.exe;

TVDEMO.OPT – настройка среды по умолчанию.

Интегрированная среда GPSS включает многооконный редактор, встроенный калькулятор, встроенный файл помощи Help по GPSS, контекстную помощь по языку GPSS, возможность работы с математическим пакетом, возможность изменения цветов и других настроек.

Запуск интегрированной среды осуществляется с помощью файла 1.EXE. Результаты моделирования записываются в файл с расширением LST. Содержимое этого файла появляется в нижнем окне экрана по окончании процесса моделирования.

4. ПРИМЕР МОДЕЛИРОВАНИЯ С ПОМОЩЬЮ GPSS

Построим модель, имитирующую работу кассира в банке. Интервалы прихода клиентов в банк распределены равномерно в интервале 18 ± 6 минут. Время обслуживания кассиром клиента распределено равномерно в интервале 5 ± 3 минуты. Клиенты обслуживаются кассиром по одному в порядке их прихода. Модель банка должна обеспечить сбор статистических данных об очереди клиентов к кассиру. Необходимо промоделировать работу в течение 8 часов модельного времени.

Работу кассира в банке можно рассматривать как функционирование системы массового обслуживания с одним прибором и очередью. Динамическими элементами модели данной системы – транзактами – являются клиенты, обслуживаемые кассиром. Порядок блоков в модели соответствует порядку фаз, в которых оказывается клиент при движении в реальной системе. Клиенты приходят; если необходимо, ждут своей очереди; затем обслуживаются кассиром; наконец, клиенты уходят. В качестве единицы времени возьмем 1 минуту.

Текст программы на GPSS имеет следующий вид:

```
SIMULATE
GENERATE 18,6; Приход клиентов
QUEUE  1       ; Регистрация в очереди
SEIZE  1       ; Захват обслуживающего прибора (кассир)
DEPART  1      ; Уход из очереди
ADVANCE  5,3   ; Задержка на время обслуживания
RELEASE  1     ; Освобождение прибора (кассира)
TERMINATE      ; Уход клиента
GENERATE  480; Генерация транзакта в момент времени 480
TERMINATE  1  ; Завершение прогона модели
START  1       ; Задание начального значения счетчика
               ;  завершения
END
```

5. ОПИСАНИЕ КОНТРОЛЬНОЙ РАБОТЫ

Тема работы: применение GPSS для анализа характеристик системы.

Цель работы: изучение средств GPSS, используемых для анализа характеристик системы, получение навыков анализа функционирования систем.

Постановка задачи

Рассмотрим следующую модель системы массового обслуживания с отказами, описывающую работу системы, содержащей обслуживающее устройство с накопителем, в котором хранятся ожидающие обслуживания заявки:

```
        simulate
        storage  s1, 4          ; накопитель 1 имеет емкость 4
        generate  M, N
        gate_snf  1, otkaz      ; если накопитель не полон, то
        enter 1                 ; добавить в накопитель 1
        seize 2
        leave 1
        advance  A, B
        release 2
        terminate
otkaz   save 1+, 1              ; в ячейке 1 - счетчик отказов
        terminate
        generate  ,,25000
        terminate 1
        start 1
        end
```

Значения параметров *M*, *N*, *A*, *B*, используемых в данной программе, приведены в табл. 1.

Для выполнения анализа характеристик моделируемой системы в указанную модель необходимо добавить операторы, обеспечивающие при моделировании следующее (номера заданий, подлежащих выполнению по вариантам, указаны в таблице 1):

1) Определение числа обслуженных запросов.
2) Получение информации для построения графика изменения загрузки прибора.
3) Вычисление вероятности отказа в обслуживании.
4) Получение информации для построения графика изменения загрузки накопителя.

5) Вычисление максимального времени пребывания заявки в системе.
6) Вычисление вероятности того, что запросу не будет отказано в обслуживании.
7) Вычисление интенсивности поступления заявок λ – среднего числа заявок, поступающих в единицу времени.
8) Вычисление интенсивности обслуживания заявок μ – среднего числа заявок, которое может быть обслужено в единицу времени.
9) Вычисление загрузки системы ρ, которая определяется по формуле $\rho = \lambda/\mu$ или $\rho = \lambda \cdot \mathbf{u}$, где $\mathbf{u}$ – среднее время обслуживания одной заявки.
10) Вычисление минимального времени пребывания заявки в системе.
11) Вычисление минимального времени обслуживания заявки в системе.

Таблица 4

Исходные данные для моделирования

Номер варианта	Интервал прихода заявки $M \pm N$	Интервал обслуживания $A \pm B$	Номера заданий для выполнения
1	25 ± 5	18 ± 10	1, 2
2	25 ± 7	28 ± 9	3, 4
3	20 ± 5	25 ± 6	7, 11
4	20 ± 3	22 ± 7	4, 5
5	22 ± 5	22 ± 4	6, 7
6	20 ± 4	21 ± 5	5, 6
7	21 ± 4	28 ± 2	8, 11
8	25 ± 4	28 ± 5	2, 3
9	37 ± 3	31 ± 6	4, 8
10	15 ± 5	17 ± 7	6, 10
11	27 ± 3	35 ± 5	4, 9
12	23 ± 7	26 ± 5	2, 9
13	35 ± 7	29 ± 10	4, 7
14	15 ± 7	17 ± 5	3, 9
15	12 ± 2	15 ± 5	3, 11
16	21 ± 3	35 ± 5	5, 7
17	37 ± 3	28 ± 5	1, 5

Номер варианта	Интервал прихода заявки $M \pm N$	Интервал обслуживания $A \pm B$	Номера заданий для выполнения
18	42 ± 10	37 ± 2	6, 9
19	17 ± 5	16 ± 3	3, 10
20	25 ± 7	30 ± 5	1, 6
21	24 ± 5	26 ± 4	9, 10
22	25 ± 6	33 ± 5	5, 7
23	26 ± 2	28 ± 3	4, 11
24	27 ± 7	35 ± 5	2, 8
25	22 ± 3	23 ± 4	2, 10
26	18 ± 4	24 ± 5	1, 8
27	28 ± 7	26 ± 5	3, 7
28	23 ± 5	22 ± 4	8, 10
29	24 ± 8	17 ± 3	9, 11
30	11 ± 3	25 ± 3	1, 4

ЗАДАНИЕ

1. Изучите стандартные числовые атрибуты, используемые в GPSS, а также блоки **savevalue, test, gate, mark,** карты **variable, fstatistic, sstatistic**, используя соответствующие разделы настоящих методических указаний и справочную систему пакета GPSS.
2. Модифицируйте приведенную выше программу так, чтобы она позволяла получить требуемую в соответствии с заданием преподавателя информацию.
3. Выполните прогон модели, выбрав значения *M*, *N*, *A*, *B* из таблицы 1 согласно назначенному варианту.
4. Постройте указанные графики. Проанализируйте их и сделайте выводы о работе системы.
5. Оформите отчет по работе.