

# Краткое описание языка GPSS

Язык **GPSS** (General Purpose Simulation System), ориентированный на процессы, разработан еще в 1961 г., но продолжает широко использоваться. Язык реализован в ряде программ **имитационного моделирования**, так, версия программы GPSS/PC в среде Windows создана в 2000 г.

Модель (программа) на языке GPSS представляет собой последовательность операторов (их называют блоками), отображающих события, происходящие в СМО при перемещениях транзактов. Поскольку в интерпретаторах GPSS реализуется **событийный метод** и в СМО может быть одновременно много транзактов, то интерпретатор будет попеременно исполнять разные фрагменты программы, имитируя продвижения транзактов в текущий момент времени до их задержки в некоторых устройствах или очередях.

Операторы (блоки) GPSS имеют следующий формат:

**<метка><имя\_оператора><поле\_операндов> [<комментарий>]**

Метка может занимать позиции, начиная со второй, имя оператора — с восьмой, поле операндов — с девятнадцатой, комментарий обязательно отделяется от поля операндов пробелом.

Поле операндов может быть пусто, иметь один или более операндов, обозначаемых ниже при описании блоков символами **A, B, C,...** Операндами могут быть идентификаторы устройств, накопителей, служебные слова и **стандартные числовые атрибуты** (СЧА). К СЧА относятся величины, часто встречающиеся в разных задачах. Это, например, такие операнды, как **S** — объем занятой памяти в накопителе, **F** — состояние устройства, **Q** — текущая длина очереди, **P** — параметр транзакта (каждый транзакт может иметь не более **I** параметров, где **I** зависит от интерпретатора), **V** — целочисленная переменная (вещественная и булева переменные обозначаются **FV** и **BV** соответственно), **X** — хранимая переменная (переменная, для которой автоматически подсчитывается статистика), **K** — константа, **AC1** — текущее время, **FN** — функция, **RN** — случайная величина, **RN1** — случайная величина, равномерно распределенная в диапазоне [0, 1] и др. При этом ссылки на СЧА записываются в виде **<СЧА>\$<идентификатор>**. Например, **Q\$ORD** означает очередь ORD или **FN\$COS** — ссылка на функцию COS.

Рассмотрим наиболее часто встречающиеся операторы, сопровождая знакомство с ними простыми примерами моделей.

Источники заявок обычно описываются блоком

**GENERATE A,B,C,D,E**

Здесь **A** и **B** служат для задания интервалов между появлениеми заявок, при этом можно использовать один из следующих вариантов:

- интервал — равномерно распределенная в диапазоне [A—B, A+B] случайная величина;
- интервал — значение функции, указанной в B, умноженной на A;

**С** — задержка в выработке первого транзакта; **D** — число вырабатываемых источником заявок; **E** — приоритет заявок. Если **D** пусто, то число вырабатываемых транзактов неограничено. Например:

```
GENERATE 6, FN$EXP, ,15
```

Этот оператор описывает источник, который вырабатывает 15 транзактов с интервалами, равными произведению числа 6 и значения функции EXP;

```
GENERATE 36,12
```

Здесь число транзактов неограничено, интервалы между транзактами — случайные числа в диапазоне [24, 48].

Функции, на которые имеются ссылки в операторах, должны быть описаны с помощью блока следующего типа:

```
M FUNCTION A,B
```

За ним следует строка, начинающаяся с первой позиции :

```
x1,y1/x2,y2/x3,/.../xn,yn
```

Здесь метка **M** — идентификатор функции, **A** — аргумент функции, **B** — тип функции, **X<sub>i</sub>** и **Y<sub>i</sub>** — координаты узловых точек функции, заданной таблично. Например:

```
EXP FUNCTION RN1,C12  
0,0/.2,.22/.4,.51/.5,.6/.6,.92/.7,1.2/.8,1.61/.9,2.3/.95,3/  
.99,4.6/.999,6.9/1,1000
```

Это описание непрерывной (С) функции EXP, заданной таблично 12-ю узловыми точками, аргументом является случайная величина (RN1), равномерно распределенная в диапазоне [0, 1]; или:

```
BBB FUNCTION *4,D6  
1,2/2,5/3,11/4,20/5,18/6,12/7,9
```

Дискретная (D) функция BBB задана 6-ю узловыми точками, аргумент — четвертый параметр транзакта, возбудившего обращение к функции BBB.

Здесь аргумент задан с использованием косвенной адресации, признаком которой является символ \*, т.е. запись \*4 означает, что аргументом является величина, указанная в 4-м параметре транзакта, вызвавшего функцию (в данном примере можно было бы использовать равноценную запись \*p4). В общем случае косвенная адресация выполняется путем записи операнда в виде СЧА\*СЧА, например: S\*X2, что означает емкость памяти, занятая в накопителе, именем которого является значение переменной X2, или Q\*p5 — длина очереди с именем, записанным в параметре 5 транзакта.

Транзакты могут порождаться и оператором размножения:

```
SPLIT A,B,C
```

Новые транзакты порождаются, когда в данный блок входит некоторый транзакт. При этом создается семейство транзактов, включающее основной (вошедший в блок) транзакт и А его копий. Основной транзакт переходит в следующий по порядку блок, а его копии переходят в блок с меткой В. Для различия транзактов параметр С основного транзакта увеличивается на 1, а транзактов-копий — на 2, 3, 4,... и т.д.

Обратное действие — сборка транзактов выполняется операторами:

**ASSEMBLE**    A

**GATHER**    A

Согласно оператору ASSEMBLE первый из вошедших в блок транзактов выйдет из него только после того, как в этот блок придут еще A-1 транзактов того же семейства. Второй оператор отличается от предыдущего тем, что из блока выходят все A транзактов.

Операторы занятия транзактом и освобождения от обслуживания устройства A:

**SEIZE**    A

**RELEASE**    A

Задержка в движении транзакта по СМО описывается оператором:

**ADVANCE**    A, B

A и B имеют тот же смысл, что и в операторе **GENERATE**.

### Пример 1

Обслуживание транзакта в устройстве WST продолжительностью  $\alpha$  единиц времени, где  $\alpha$  — равномерно распределенная в диапазоне [7,11] случайная величина, описывается следующим фрагментом программы

**SEIZE**    WST

**ADVANCE**    9,2

**RELEASE**    WST

Аналогично описывается занятие транзактом памяти в накопителе:

**ENTER**    A, B

Здесь помимо имени накопителя (A) указывается объем занимаемой памяти (B).

Освобождение B ячеек памяти в накопителе A выполняется оператором:

**LEAVE**    A, B

Для накопителей в модели нужно задавать общий объем памяти, что делается в следующем описании накопителя:

M    **STORAGE**    A

Здесь: M — имя накопителя, A — объем его памяти.

Если транзакт приходит на вход занятого устройства или на вход накопителя с недостаточным объемом свободной памяти, то транзакт задерживается в очереди к этому устройству или накопителю. Следение за состоянием устройств и очередей выполняет интерпретатор. Но если в модели требуется ссылаться на длину очереди или собирать статистику по ее длине, то требуется явное указание этой очереди в модели. Делается это с помощью операторов входа в очередь и выхода из очереди:

**QUEUE**    A, B

**DEPART**    A, B

Согласно этим операторам очередь A увеличивается и уменьшается на B единиц соответственно, если B=1, то поле B можно оставить пустым.

Движение транзактов выполняется по маршруту, заданному последовательностью операторов в модели. Если требуется изменение

естественного порядка, то используется оператор перехода. Оператор условного перехода имеет вид:

**TEST XX A,B,C**

В соответствии с ним переход к оператору, помеченному меткой C, происходит, если не выполняется условие A XX B, где XX ∈ {**E**, **NE**, **L**, **LE**, **G**, **GE**} ; **E** — равно; **NE** — неравно; **L** — меньше; **LE** — меньше или равно; **G** — больше; **GE** — больше или равно (XX всегда размещается в позициях 13 и 14).

### Пример 2

Приходящие пользователи ожидают обслуживания, если длина очереди не более 4, иначе от обслуживания отказываются. Соответствующий фрагмент программы

```
TEST LE Q$STR,4,LBL
    QUEUE      STR
    SEIZE      POINT
    DEPART    STR
    ADVANCE   50,16
RELEASE    POINT
...
LBL      TERMINATE 1
```

В примере 2 использован оператор выхода транзактов из СМО:

**TERMINATE A**

Согласно этому оператору из итогового счетчика вычитается число A. С помощью итогового счетчика задается длительность моделирования. В начале исполнения программы в счетчик заносится число, указанное в операнде A оператора

**START A,B,C**

Моделирование прекращается, когда содержимое счетчика будет равно или меньше нуля. Операнд C — шаг вывода статистики на печать. Если B=0 и C=0, то выполняется только стандартная печать по окончании моделирования. В стандартную печать входят собранные за время моделирования статистические данные по основным параметрам модели: средние и максимальные значения длин очередей, объемов занимаемой памяти в накопителях, времени занятого состояния устройств и др. От печати можно отказаться, указав B=NP.

### Пример 3

Общая структура программы на GPSS имеет вид

**SIMULATE**

<описания, в том числе функций, накопителей, массивов и т.п.>

<операторы, моделирующие движение транзактов>

**START A,B,C**

**END.**

Оператор безусловного перехода записывается следующим образом:

**TRANSFER ,B**

Здесь В — метка оператора, к которому следует переход.

Используется ряд других разновидностей оператора **TRANSFER**.

Например:

**TRANSFER P,B,C**

Переход происходит к оператору с меткой, равной сумме значения параметра В транзакта и числа С.

**TRANSFER FN,B,C**

То же, но вместо параметра транзакта слагаемым является значение функции В.

**TRANSFER PICK,B,C**

Это оператор равновероятного перехода к операторам, метки которых находятся в интервале [В,С].

Важное место в СМО занимает переход по вероятности:

**TRANSFER A,B,C**

Здесь А — вероятность перехода к оператору с меткой С, переход к оператору с меткой В будет происходить с вероятностью 1—А.

Оператор перехода в циклических процедурах выглядит так:

**LOOP A,B**

Здесь А — номер параметра транзакта, в котором содержится число повторений (витков) цикла, В — метка оператора, с которого начинается повторяющаяся часть.

#### Пример 4

Заказы, поступающие в СМО в случайные моменты времени в диапазоне [20,40], выполняет сначала бригада WGR1, затем параллельно работают бригады WGR2 и WGR3, каждая над своей частью заказа. Заданы экспоненциальные законы для времен выполнения работ бригадами WGR1, WGR2 и WGR3 с интенсивностями 0,05, 0,1 и 0,125 соответственно. Моделирование нужно выполнить на временном отрезке, соответствующем выполнению 1000 заказов.

Программа:

**SIMULATE**

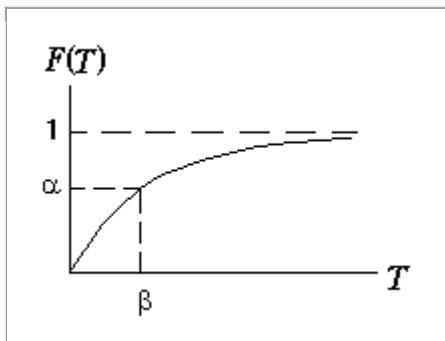
```
EXP FUNCTION RN1,C12
0,0/.2,.22/.4,.51/.5,.6/.6,.92/.7,1.2/.8,1.61/.9,2.3/.95,3/.99,4
.6/.999,6.9/1,1000

GENERATE 30,10
SEIZE WGR1
ADVANCE 20,FN$EXP
RELEASE WGR1
SPLIT 1,MET1
SEIZE WGR2
ADVANCE 10,FN$EXP
RELEASE WGR2
TRANSFER ,MET2
MET1 SEIZE WGR3
```

```

ADVANCE     8 , FN$EXP
RELEASE      WGR3
MET2  ASSEMBLE   2
TERMINATE    1
START        1000
END .

```



**Рис. 1. Функция экспоненциального закона распределения**

В этом примере использован экспоненциальный закон распределения с плотностью

$$P(t) = \lambda \exp(-\lambda t),$$

где  $\lambda$  — интенсивность. Функция распределения экспоненциального закона

$$F(T) = \int_0^T P(t) dt = 1 - \exp(-\lambda T).$$

Из рис. 1 ясно, что поскольку искомыми являются значения  $\beta$  случайной величины  $T$ , то, задавая значение  $\alpha$ , как равномерно распределенной в диапазоне  $[0, 1]$  случайной величины, по формуле

$$\beta = \frac{1}{\lambda} \ln \frac{1}{1-\alpha} \quad (1)$$

находим искомое значение. Именно в соответствии с (1) в операторах **ADVANCE** множителями были значения  $\frac{1}{\lambda}$ .

Приведем еще несколько операторов языка GPSS.

Оператор изменения параметров транзактов:

**ASSIGN**      **A, B**

Здесь A — номер параметра транзакта, B — присваиваемое ему значение.

В следующих операторах параметр A увеличивается (уменьшается) на значение B:

**ASSIGN**      **A+, B**

**ASSIGN**      **A-, B**

Оператор фиксации времени события

**MARK**      **A**

его выполнение заключается в запоминании значения текущего модельного времени в параметре вошедшего в оператор MARK транзакта, в поле А указывается номер параметра.

### Пример 5

На вход производственной линии поступают и проходят обработку на станке TOOL1 детали типов X и Y. Далее детали типа X обрабатываются на станке TOOL2, а детали типа Y — на станке TOOL3. Интервал моделирования соответствует обработке 600 деталей (ниже у операторов **GENERATE** и **ADVANCE** значения operandов не конкретизированы):

**SIMULATE**

```
    GENERATE    A,B
    ASSIGN      1,LBL4
    TRANSFER   ,LBL1
    GENERATE    A,B
    ASSIGN      1,LBL2
LBL1  SEIZE      TOOL1
    ADVANCE    A,B
    RELEASE    TOOL1
    TRANSFER   P,1
LBL4  SEIZE      TOOL2
    ADVANCE    A,B
    RELEASE    TOOL2
LBL3  TERMINATE  1
LBL2  SEIZE      TOOL3
    ADVANCE    A,B
    RELEASE    TOOL3
    TRANSFER   ,LBL3
START   600
END .
```

Расширение возможностей управления движением транзактов достигается благодаря операторам, реализующим механизм семафора:

**LOGIC X A**

**GATE XX A,B**

Первый оператор при  $X = S$  устанавливает переключатель А в единичное состояние, при  $X = R$  сбрасывает его в нулевое состояние, при  $X = I$  инвертирует значение состояния. Второй оператор при  $XX = LR$  и значении переключателя, указанного в А, равном 1, или при  $XX = LS$  и состоянии переключателя 0 передает транзакт оператору с меткой В (или задерживает его в блоке **GATE**, если поле В пусто), а при других сочетаниях XX и значений переключателя — направляет к следующему оператору.

Вычислительный оператор присваивает переменной с номером М значение арифметического выражения А:

**M VARIABLE A**

Например в следующем операторе переменной номер 3 присваивается разность числа 216 и объема занятой памяти в накопителе MEM2:

**XINIT VARIABLE K216-S\$MEM2**

Знаки арифметических операций сложения, вычитания, умножения, деления +, -, #, / соответственно. В случае логических выражений имя оператора должно быть BVARIABLE, а знаками операций дизъюнкции и конъюнкции являются + и #. Если операции выполняются над числами типа real, то имя оператора FVARIABLE.

Следующий оператор присваивает хранимой переменной, указанной в А. значение, записанное в В:

**SAVEVALUE A,B**

Прерывание обслуживания заявки в устройстве А происходит при входе некоторой другой заявки в блок:

**PREEMT A**

а возобновление прерванного обслуживания заявки — при входе в блок:

**RETURN A**

Оператор синхронизации, реализующий механизм randevu, имеет, например, вид:

**LBL MATCH NUMB**

Приходящий в него транзакт задерживается до тех пор, пока в некоторой другой части модели в сопряженный оператор не войдет транзакт того же семейства. Сопряженный оператор выглядит так:

**NUMB MATCH LBL**

Часто сведения о некоторых величинах, характеризующих моделируемый процесс, удобно представлять в виде гистограмм. Задание гистограммы выполняют в разделе описаний с помощью оператора:

**M TABLE A,B,C,D**

Здесь М — имя гистограммы; А — табулируемая величина; В — верхняя граница левого интервала гистограммы; С — ширина интервалов; D — число интервалов. Формирование гистограммы происходит с помощью оператора:

**TABULATE A**

Выполнение этого оператора увеличивает на единицу число попаданий в  $i$ -й интервал гистограммы, имя которой указано в А. При этом  $i$ -й интервал соответствует текущему значению переменной, являющейся аргументом для гистограммы.

### Пример 6

Требуется разработать модель процессов возникновения и устранения неисправностей в некоторой технической системе, состоящей из множества однотипных блоков; в запасе имеется один исправный блок; известны статистические данные об интенсивностях возникновения отказов и длительностях таких операций, как поиск неисправностей, замена и ремонт отказавшего блока. Поиск и замену отказавшего блока производит бригада TEAM1, а ремонт замененного блока — бригада TEAM2.

**SIMULATE**

**GENERATE A,B моделируется возникновение отказов**

```

SEIZE      TEAM1
    ADVANCE   A, В поиске исправности
ENTER      MEM, 1 получение запасного блока из резерва
ADVANCE    A, Взамен блока
    RELEASE   TEAM1
    SEIZE     TEAM2
    ADVANCE   A, Времонт
    LEAVE     MEM, 1 восстановление резерва
    RELEASE   TEAM2
    TERMINATE 1
START      1000
END .

```

### Пример 7

Требуется разработать модель сборки изделия из 30 деталей типа А1 и 16 деталей типа А2, поступающих на сборочный участок от независимых экспоненциальных источников с интенсивностями  $\lambda$ , равными 0,1 и 0,04 мин<sup>-1</sup> соответственно. Длительность сборочной операции находится в пределах [12,18] мин. Промоделировать выпуск 600 изделий. Табулировать наполнение входного бункера с деталями типа А2 перед началом сборки.

#### SIMULATE

```

MEM1  STORAGE      30
MEM2  STORAGE      16
TAB   TABLE        MEM2, 32, 16, 6
EXP   FUNCTION     RN1, C12
0,0/.2,.22/.4,.51/.5,.6/.6,.92/.7,1.2/.8,1.61/
.9,2.3/.95,3/.99,4.6/.999,6.9/1,1000
    GENERATE    10, FN$EXP
    ENTER       MEM1, 1
    TRANSFER    , MMM
    GENERATE    25, FN$EXP
    ENTER       MEM2, 1
MMM   TEST GE     S$MEM1, 30, LLL
    TEST GE     S$MEM2, 16, LLL
    TABULATE    TAB
    SEIZE       MONT
    ADVANCE     15, 3
    RELEASE     MONT
    TERMINATE   1
LLL   TERMINATE
START     600

```

**END .**

**Список литературы**

- 1. Томашевский В., Жданова Е. Имитационное моделирование в среде GPSS. — М.: Бестселлер, 2003.**
- 2. GPSS. — <http://www.compmode.ru/394/>**

# ITteach.RU

... Учиться, Учиться, и еще раз Учиться...

## Программирование в GPSS - Функции

### Оглавление

- [Программирование в GPSS](#)
- [Прагматический аспект](#)
- [Классы объектов языка GPSS](#)
- [Интерпретации транзактов](#)
- [Интерпретаций устройств](#)
- [Элементы реальных систем](#)
- [Очереди и таблицы](#)
- [Блоки](#)
- [Модельное время](#)
- [Общегороднические средства GPSS](#)
- [Стандартные числовые атрибуты](#)
- [Арифметические переменные](#)
- [Логические \(булевы\) переменные](#)
- [Функции](#)
- [Ячейки и матрицы ячеек](#)
- [Задание начальных значений ячеек и матриц](#)
- [Объектно-ориентированные средства GPSS](#)
- [Создание и уничтожение транзактов](#)
- [Задержка транзактов в блоках ADVANCE](#)
- [Операции занятия и освобождения устройств](#)
- [Операции захвата и освобождения устройств](#)
- [Операции блокирования и разблокирования устройств](#)
- [Описание памяти и работа с ними](#)
- [Операции с ключами](#)
- [Блок GATE](#)
- [Синхронизация транзактов](#)
- [Изменение параметров транзактов](#)
- [Управление маршрутами транзактов в модели](#)
- [Сбор статистики с помощью очередей](#)
- [Сбор статистики с помощью таблиц](#)
- [Стандартные числовые атрибуты](#)
- [Атрибуты транзактов](#)
- [Атрибуты блоков](#)
- [Системные атрибуты](#)
- [Атрибуты оборудования](#)
- [Статистические атрибуты](#)
- [Атрибуты ячеек SAVEVALUE](#)
- [Функции и переменные](#)
- [Атрибуты списков пользователя](#)
- [Атрибуты группы](#)
- [Системные атрибуты](#)
- [Блоки COUNT и SELECT](#)
- [Все страницы](#)

### Статьи раздела

- [Модель транспортного цеха](#)
- [GPSS World](#)
- [Имитационное моделирование в среде GPSS](#)
- [Скачать GPSS](#)
- [Общее описание](#)

[все статьи раздела](#)
[Скачать GPSS](#)

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

Пусть необходимо, чтобы значение FN\$KL1 вычислялось в модели через S\$MEM в соответствии с графиком, изображенным на приведенных ниже рисунках.

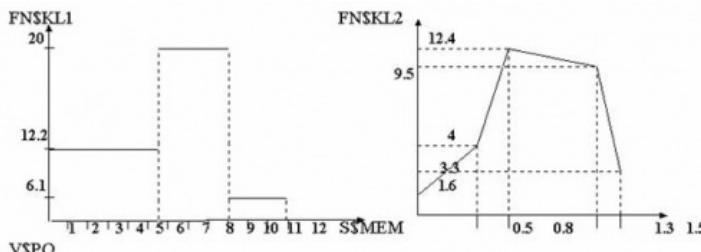


Рис. \*1

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

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

Первая строка описания функции должна содержать метку функции в поле метки, слово **FUNCTION** в поле операции и два операнда в поле операндов. Первый operand является обозначением того СЧА, который выбран в качестве аргумента функции. В данном примере это **S\$MEM** — количество занятых единиц памяти **MEM**. Второй operand состоит из буквы **D**, обозначающей, что данная функция дискретная, и из числа точек, которыми задается график функции. Координаты точек, задающих график, записываются в последующих строках описания функции. Из примера видно, что каждая точка задается парой координат, разделенных запятой, причем вначале указывается абсцисса точки, потом ордината. Координаты разных точек разделяются косой чертой. Функция может задаваться любым числом точек. Запись координат точек начинается с 1-й позиции строки и может быть продолжена на последующих строках.

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

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

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

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

? 12.2 , при - ? < S\$MEM < 5.5,

FN\$KL1 = ? 20, при 5.5 < S\$MEM < 9,

? 6.1 , при 9 < S\$MEM < ? .

График функции FN\$KL2, изображенный на рис.\*1, получен путем соединения нескольких точек ломаной линией. Функция, задаваемая таким графиком, называется непрерывной. Функция FN\$KL2 в данном случае описывается строками:

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

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

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

Важным частным случаем рассмотренных видов функций являются функции с аргументом RNj. Если в строке FUNCTION в поле A записан аргумент RNj , то подразумевается, что в действительности используется не сам СЧА RNj , а полученное с его помощью псевдослучайное число, заключенное в диапазоне от 0 до 0,999999 включительно, т.е. RNj берется в масштабе 1:1000. Когда аргументом функции является случайное число, то функция также будет принимать случайные значения. Функция от аргумента RNj используется для моделирования случайных величин, имеющих требуемые законы распределения вероятностей. Например, дискретная функция, описанная строками:

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

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

Действительно, график функции FN\$DIS имеет вид, показанный на рис.\*4. Поэтому функция FN\$DIS примет при ее очередном вычислении значение 1, если RN1 попадет в интервал от 0 до 0,1. Вероятность этого равна 0,1. Значение 5 для FN10 будет получено с вероятностью попадания RN1 в интервал от 0,1 до 0,3. Эта вероятность равна 0,3 - 0,1 = 0,2. Аналогично определяются вероятности значений 7 и 8.

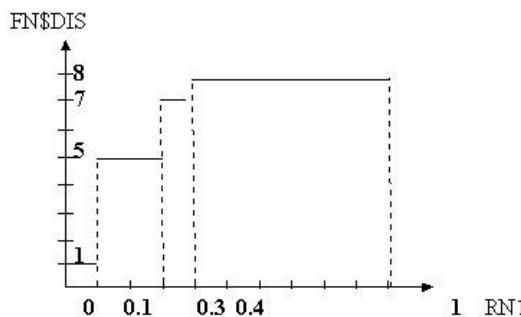


Рис. \*4

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

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

```
EXP FUNCTION RN1,C24
0,0/1,.104/.2,.222/.3,.355/.4,.509/.5,.69
.6,.915/.7,1.2/.75,1.38/.8,1.6/.84,1.83/.88,2.12
.9,2.3/.92,2.52/.94,2.81/.95,2.99/.96,3.2/.97,3.5
.98,3.9/.99,4.6/.995,5.3/.998,7/.9997,8.
```

В GPSS World содержится операция вычисления натурального логарифма, то можно задавать преобразование RN1 в экспоненциальную случайную величину без такого громоздкого набора чисел, как в приведенном примере функции FN\$EXP. Аналитически такое преобразование выражается формулой:

y = - M?ln (z),

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

M – произвольный положительный числовой множитель,

y – функция от z, (т.е. тоже случайная величина). Распределение случайной величины y будет

экспоненциальным с математическим ожиданием M.

В GPSS World содержится стандартная функция задания экспоненциального распределения, например

Generate (Exponential(1,0,65))

генерирует тракзакты по экспоненциальному закону распределения, в среднем 65 тракзактов.

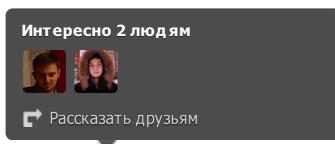
С другой стороны применение в GPSS функций FUNCTION позволяет значительно ускорить вычисления. Это ускорение обеспечивается за счет того, что числовая информация о функции FUNCTION представляется в памяти ЭВМ в виде таблицы, в которой для возможных значений аргумента уже хранятся готовые, заранее заданные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

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

<< Назад - Вперед >>

Читайте также:

- [Модель транспортного цеха](#)
- [GPSS World](#)
- [Имитационное моделирование в среде GPSS](#)
- [Скачать GPSS](#)



# **Моделирование**

# **Содержание**

Моделирование.....	1
Содержание.....	2
Моделирование.....	4
Не Марковские случайные процессы, сводящиеся к Марковским. ....	30
Способы получения псевдослучайных чисел. ....	34
Преимущества и недостатки типов генерации случайных чисел. ....	35
Простейшие алгоритмы генерации последовательности псевдослучайных чисел.....	36
Распределение Пуассона. ....	39
Распределение Эрланга. ....	39
Нормальное (Гауссово) распределение. ....	39
Методика построения программной модели ВС. ....	41
Моделирование работы источника информации (ИИ). ....	42
Моделирование работы Обслуживающего Аппарата.....	43
Моделирование работы абонентов. ....	44
Моделирование работы буферной памяти. ....	44
Разработка программы для сбора статистики. ....	45
Управляющая программа имитационной модели. ....	46
Моделирование систем и языки моделирования.....	50
Классификация языков имитационного моделирования. ....	51
Формальное описание динамики моделируемого объекта. ....	51
Задачи построения модели. ....	52
Языки, ориентированные на события. ....	52
Языки, ориентированные на процессы.....	53
Сравнение универсальных и специализированных языков программирования при моделировании: .....	54
Основные концепции языка РДО (Ресурсы, действия, операции). ....	55
Представление сложной дискретной системы в РДО методе. ....	58
AnyLogic™ .....	62
Открытая архитектура. ....	63
Уровни моделирования. ....	63
Язык General Purpose System Simulation (GPSS) .....	64
Классификация блоков GPSS.....	67

Управление процессом моделирования. ....	68
Задержки транзактов по заданному времени. ....	71
Группа блоков создания и уничтожения транзактов. ....	72
Изменения параметров транзакта. ....	73
Группа блоков, создания копий транзактов. ....	73
Группа блоков синхронизации движения транзактов. ....	74
Блоки, определяющие аппаратную категорию. ....	77
Блоки, изменяющие маршруты транзактов. ....	80
Блоки, относящиеся к статистической категории .....	83
Определение функции в GPSS .....	86
Моделирование вероятностных функций распределения GPSS World .....	88
Классификация систем массового обслуживания .....	88
Метод формализации для сложных дискретных систем и структур .....	93

[1.09.2006][Лекция 1]

## Моделирование

Лектор: Рудаков Игорь Владимирович

Литература:

- 1) Бусленко «Моделирование сложных систем» (последн. редакция)
- 2) Советов, Яковлев «Моделирование систем»
- 3) Шрайбер «Моделирование на JPSS»
- 4) Марков «моделирование информационно-вычислительных процессов»
- 5) Методички Рудакова и Курова
- 6) Наренков, Федорук «Моделирование сложных дискретных систем на базе...» (методичка) 1999 г.

Два преимущества моделирования:

- 1) дешевизна;
- 2) фантастика

## Философские аспекты моделирования.

**Методологическая основа моделирования** – это диалектический метод познания (придумал Гегель).

Все то, на что направлена человеческая деятельность над **объектом**.

Выработка методологии направлена на упорядочивание, получение и обработку информации об объекте, которые существуют вне нашего сознания и взаимодействуют между собой и внешней средой.

[4.09.2006] [Лекция 2]

В научных исследованиях большую роль играет понятие *гипотеза*, т.е. определенные предсказания, основывающиеся на небольшом количестве опытных данных, наблюдениях, догадках. Быстрая и полная проверка выдвигаемых гипотез может быть проведена в ходе специально поставленного эксперимента. При формировании и проверке правильности гипотезы большое значение в качестве метода суждения имеет *аналогия*.

>> Методика - совокупность методов.

**Аналогией** называется суждение о каком-либо частном сходстве двух объектов. Причем такое сходство может быть существенным или несущественным.

*Существенные* сходства (различия) условны и относительны - зависит от уровня абстрагирования и в общем случае определяется конечной целью проводимого исследования.

Современная научная гипотеза создается, как правило, по аналогии с проведенными на практике положениями.

Аналогию и гипотезу связывает эксперимент.

*Гипотезы и аналогии, отражающие реальный объективно существующий мир, должны обладать наглядностью или сводиться к удобным для исследования логическим схемам, упрощающим рассуждения и логические построения, или позволяющим проводить эксперименты, уточняющие природу явления, называются моделями.*

*Т.е. модель - это объект-заместитель объекта оригинала, обеспечивающий изучение некоторых свойств оригинала.*

*Замещение одного объекта другим с целью получения информации по важнейшим свойствам объекта оригинала с помощью объекта-модели называется моделированием.*

*Гносеологическая роль теории моделирования, т.е. её значение в теории познания, заключается в том, что изучение моделей, выступает в роли относительно самостоятельного квазиобъекта, позволяет получить при исследовании некоторые данные о самом объекте.*

>> Данные отображают характеристики, свойства объекта

Причем по отношении к моделям исследователь является экспериментатором, только эксперимент проводится не с объектом, а с моделью.

Любой эксперимент может иметь существенное значение в конкретной области науки и техники, только при его специальной обработке и обращении.

Единичный эксперимент не может быть решающим для подтверждения гипотезы или теории.

В основе моделирования лежит теория подобия, которая утверждает, что абсолютное подобие имеет место только при замене одного объекта другим точно таким же.

Подобия:

1. Полное
2. Неполное
3. Приближенное

## **Классификация видов моделирования.**

Будем давать классификации в зависимости от характера изучаемых процессов в системе

## Виды моделирования



**Детерминированное моделирование** отображает детерминированные процессы, т.е. такие процессы в которых отсутствуют всякие случайные величины и даже случайные процессы.

**Стохастическое моделирование** - отображают вероятностные процессы и события.

**Статическое моделирование** служит для описаний поведения объекта в какой-либо момент времени.

**Динамическое моделирование** отражает поведение объекта во времени.

**Дискретное моделирование** служит для отображения объекта в определенный момент времени.

**Непрерывное моделирование** позволяет отображать непрерывный процесс в системе.

Под **математическим моделированием** будем понимать процесс установления данному реальному объекту некоторого математического объекта, называемого **математической моделью**, и исследование этой модели позволяет получить характеристики реального объекта.

Вид математической модели зависит как от природы реального объекта, так от целей моделирования. Любая математическая модель, как и всякая другая, описывает реальный объект лишь с определенной степенью приближения.

>> В аналитике главное, что мы можем описать модель формулами.

**Аналитическое моделирование** – математическая формализация, изменение свойств объекта во времени.

Для аналитического моделирования характерно то, что процессы функционирования элементов системы записываются в виде некоторых функциональных соотношений (алгебраические, интегро-дифференцированные, конечно разностные) и логических условий.

Аналитическая модель может быть исследована 3-мя способами:

1. аналитическим способом – стремятся получить в общем виде зависимость от исходных характеристик;
2. численным способом – когда нельзя решить в общем виде, то получаем результаты для конкретных начальных данных;
3. качественный способ – не имея решения управления в общем виде, мы можем найти некоторые свойства решения;

>> Имитационное моделирование хуже аналитического. Последнее – самое лучшее.

При **имитационном моделировании** реализующий модель алгоритм воспроизводит процесс функционирования системы во времени, причем имитируются элементарные явления, составляющие процесс с сохранением их логической структуры и последовательности протекания во времени, что позволяет по исходным данным получить сведения о состоянии процесса в определенные моменты времени, дающие возможность оценить характеристики системы.

Основным преимуществом имитационного моделирования по сравнению с аналитическим, является возможность решения более сложных задач. Имитационные модели позволяют достаточно просто учитывать такие факторы, как наличие дискретных и непрерывных элементов, нелинейные характеристики системы и её элементов, многочисленные случайные воздействия.

Когда результаты, получаются при воспроизведении на имитационной модели процесса функционирования системы, являются реализациями случайных величин и функций, тогда для нахождения характеристик процесса, требуется его многократное воспроизведение с последующей статической обработкой.

**Комбинированное** моделирование при анализе системы позволяет определить достоинства его компонентов. Обычно проводят декомпозицию процесса функционирования объекта на составляющие подпроцессы.

>> Чем больше аналитики, тем ближе результат.

[11.09.2006][Лекция 3]

## Технические средства ЭВМ.

Это ЭВМ, которые мы используем при моделировании, т.е. компьютер, да и вообще любые вычислительные устройства.

Принципиально можно выделить 2 типа: цифровые и аналоговые. **Цифровая** техника является дискретной. Основной принцип – быстродействие (не догнать реальное время) слишком сложен механизм.

>>

*Процессор может выполнять:*

- 1) АЛУ
- 2) Управление

*Только сложение и сдвиг  
У ОП быстродействие должно быть сопоставимо с ЦП.*

<<

Аналоговая быстрее цифры. Скорость аналоговых ограничивается скоростью передвижения электрона в цепи.

Недостаток: низкая точность, т.к. всё представляется сигналами.

У цифровой недостаток: медленная.

При моделировании компьютера являются наиболее конструктивным способом для решения большинства инженерных задач 2 основные пути использования:

1. Как средство расчета по полученным аналитическим моделям.
2. Как средство имитационного моделирования.

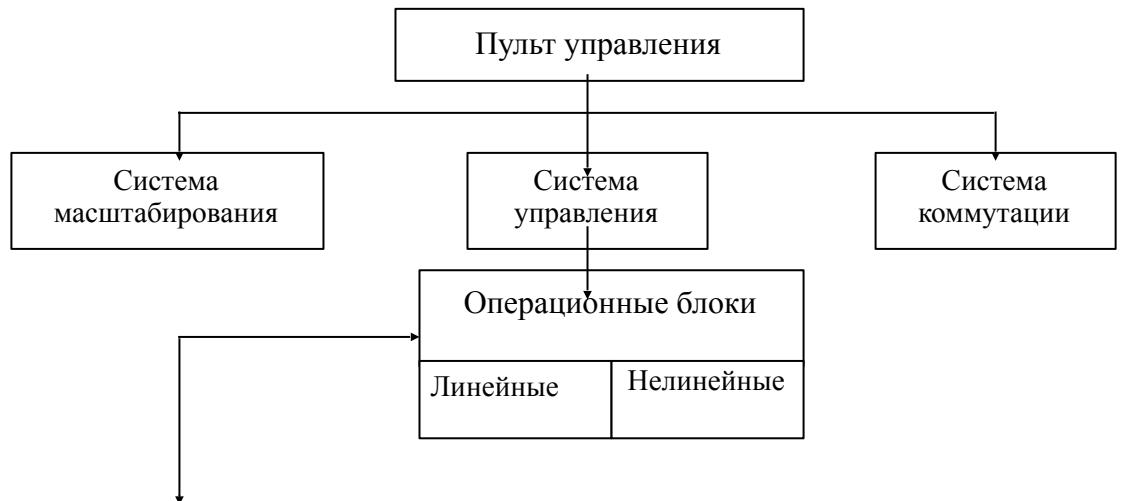
(!) Вспомнить про цифровые компьютеры.

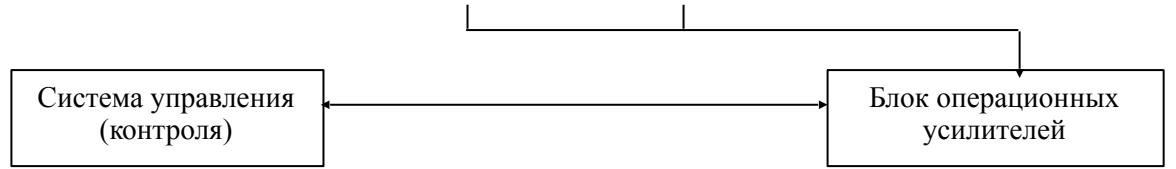
В отличие от дискретной техники в аналоговых компьютерах заложен принцип моделирования, а не счета. В качестве модели определенной задачи используются электронные цепи. В каждой переменной величине задачи ставится в соответствие переменная величина электронной цепи. При этом основа построения такой модели является *изоморфизм* (подобие) исследуемой задачи и соответствующей ей электронной модели. В большинстве случаев при определении критерия подобия используются специальные приемы масштабирования, соответствующих значений параметров модели и переменных задач. Согласно своим вычислительным возможностям аналоговые машины наиболее приспособлены для исследования объектов, динамика которых описывается обыкновенными и частными производными в ОДУ и алгебраических уравнениях. => АВМ можно отнести к специальным машинам.

Под АВМ (анalogовые ВМ) будем понимать совокупность электрических элементов организованных в систему, позволяющую изоморфно моделировать динамику изучаемого объекта. Функциональные блоки АВМ должны реализовывать весь комплекс арифметико-логических операций.

АВМ делятся по мощности (степень дифференциальных уравнений):

- малые ( $n < 10$ ),
- средние ( $10 \leq n \leq 20$ ),
- большие аналоговые комплексы ( $n > 20$ )

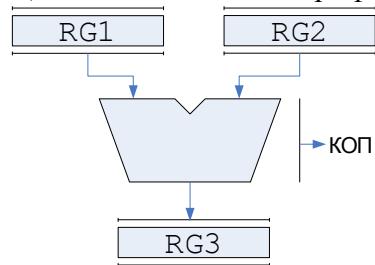




Под гибридной ВМ будем понимать широкий класс вычислительных систем, использующие как аналоговую, так и дискретную формы представления и обработки информации.

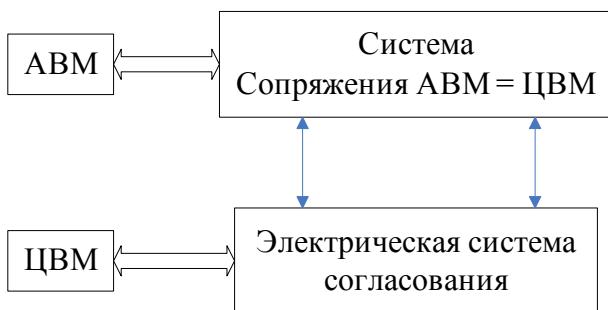
Подклассы гибридных ВМ:

1. АВМ, использующие цифровые методы численного анализа
2. АВМ, программируемые с помощью ЦВМ (цифровой).
3. АВМ с цифровым управлением и логикой
4. АВМ с цифровыми элементами (например, память, цифровые вольтметры и прочие ВМ)
5. ЦВМ с аналоговыми арифметическими устройствами



6. ЦВМ, допускающие программирование аналогового типа (программировать можем дифференциальные анализаторы)

Гибридная ВМ:



Применение гибридной вычислительной техники:

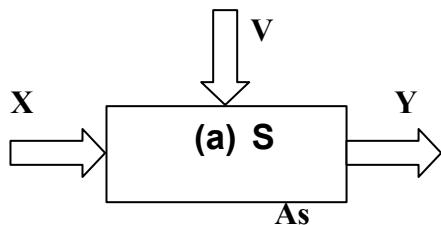
1. моделирование дискретных систем и случайных процессов;
2. решение задач оптимизации
3. исследование в области управления подвижными объектами
4. моделирование системы "человек - компьютер"

Сравнительная характеристика аналоговой и цифровой техники:

Тип информации	Непрерывный	Дискретный
Изменение значений	Величиной напряжения	Числовым значением
Базовые операции	Арифметические операции и интегрирование	Арифметические операции
Принцип вычисления	Высокопараллельный	Последовательно-параллельный
Режим реального времени	Без ограничений	Ограничен
Динамическое изменение решаемой задачи	Посредством системы коммутации	В диалоговом режиме
Требования к пользователю	Профессиональные знания, методика моделирования	Знание основ ПО ЭВМ
Уровень формализации задачи	Ограничен моделью решаемой задачи	Высокий
Способность к решению логических задач	Ограничена	Высокая
Точность вычисления	Ограничена ( $10^{-4}$ )	Ограничена разрядностью ( $10^{-40}$ )
Диапазон представления чисел	$1 \dots 10^{-4}$	Зависит от разрядности
Класс решаемых задач	Алгебраические и дифф. уравнения.	Любые
Специальные функции	Ограниченный набор	Неограниченный набор
Уровень миниатюризации	Ограничен	Высокий
Сфера применения	Ограничена	Практически любая
Пользовательский интерфейс	Низкий уровень	Высочайший уровень

### Основные понятия теории моделирования.

Пусть задана сложная дискретная система S.



Модель объекта моделирования можно представить в виде множества величин, определяющих процесс функционирования реальной системы S и образующие в общем случае следующие подмножества:

Множество входных параметров

Множество внутренних параметров

Внешнее воздействие

Множество выходных параметров

В общем случае эти переменные ( $x_i, h_j, v_k$ ) являются элементами непересекающихся подмножеств и содержат как детерминированные, так и стохастические составляющие.

При моделировании функционирования сложной системы S, входные воздействия X, воздействия внешней среды M и внутренние параметры системы являются независимыми (экзогенными) характеристиками (или переменными), которые в векторной форме имеют следующий вид:

$$\overrightarrow{x(t)} = \{x_1(t), x_2(t), \dots, x_{nx}(t)\}$$

$$\overrightarrow{h(t)} = \{h_1(t), h_2(t), \dots, h_{nh}(t)\}$$

$$\overrightarrow{v(t)} = \{v_1(t), v_2(t), \dots, v_{nv}(t)\}$$

А выходные характеристики системы являются зависимыми (эндогенными) переменными и в векторной форме имеют следующий вид:

$$\overrightarrow{y(t)} = \{y_1(t), y_2(t), \dots, y_{ny}(t)\}$$

Процесс функционирования системы S описывается по времени некоторым оператором  $F_s$ , который в общем случае преобразует независимые переменные в соответствии со следующим соотношением:

$$\overrightarrow{y(t)} = F_s(\vec{x}, \vec{h}, \vec{v}, t) \quad (1)$$

Эта зависимость (1) называется **законом функционирования сложной** системы S. В общем случае он может быть задан в виде функции, функционала, логических условий, в алгоритмическом или табличном виде и т.д. Вероятно важным является понятие алгоритма функционирования системы, под которым подразумевается метод получения выходных характеристик  $y(t)$  с учетом входных воздействий  $x(t)$ , воздействий внешней среды  $v(t)$  и соответствующих внутренних параметров системы  $h(t)$ . Очевидно, что один и тот же закон функционирования  $F_s$  может быть реализован различными способами, т.е. с помощью множества различных алгоритмов функционирования. Соотношение (1) может быть получено и через свойства системы в конкретные моменты времени, называемыми состояниями, которые характеризуют вектор состояний:

$$z_k \in Z, k = \overline{1, n_z}$$

$$\overrightarrow{z_m(t)} = \{z_1(t), z_2(t), \dots, z_k(t)\}$$

Если рассматривать процесс функционирования системы как последовательную смену состояний  $z_1(t), z_2(t), \dots, z_k(t)$ , то они могут быть интерпретированы как координаты точки в k-мерном пространстве (фазовом пространстве), причем каждой реализации процесса будет соответствовать некоторая траектория – совокупность всех возможных состояний  $\overrightarrow{z(t)}$ . Совокупность всех возможных состояний  $\overrightarrow{z(t)}$  называется **пространством состояния объекта**.

Состоянием системы в момент времени  $t_0 \leq t \leq T^*$  полностью определяется начальными условиями  $\overrightarrow{z^0(t)} = \{z_1^0(t), z_2^0(t), \dots, z_{n_z}^0(t)\}$ , где  $Z^0$  – состояние системы в момент времени  $t_0$ , входными воздействиями, внутренними параметрами и воздействиями внешней среды, которые имели место за промежуток времени  $(t - t_0)$ . Определим их с помощью двух векторных уравнений:

$$\begin{cases} \overrightarrow{z(t)} = \Phi(\overrightarrow{z^0}, \vec{x}, \vec{v}, \vec{h}, t) \\ \overrightarrow{y(t)} = F(\vec{z}, t) \end{cases}$$

В общем случае время в модели может быть непрерывным в интервале  $t_0 \leq t \leq T^*$ , а может быть и дискретным, т.е. квантованным на отрезке  $\Delta t$ :  $T^* = m\Delta t$ , где  $m$  - число интервалов дискретизации.

## Типовые математические схемы.

В практике моделирования на первоначальных этапах формализации объектов используют так называемые **типовыe математические схемы**, к которым относят такие хорошо проработанные (разработанные) математические объекты, как дифференциальные алгебраические уравнения, конечные вероятностные автоматы и т.д.

процесс функционирования системы	типовая математическая схема	обозначение
Н е п р е р ы в н о - детерминированный подход	стандартные ДУ	D-схема
Д и с к р е т н о - детерминированный подход	конечные автоматы	F-схема
Дискретно-стохастический подход	вероятностные автоматы	P-схема
Непрерывно-стохастический подход	система массового обслуживания	Q-схема
О б о б щ е н н ы е (универсальный)	агрегативная система	A-схема

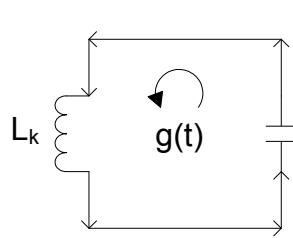
Объект или процесс функционирования сводится к конкретному классу задач.

P, Q - схемы относятся к P-net, CP-net

У A - схем основная операция - это декомпозиция.

### Непрерывно-детерминированные модели. ( D-схемы [dinamic] )

$$\text{ДУ: } y' = f(y, t)$$



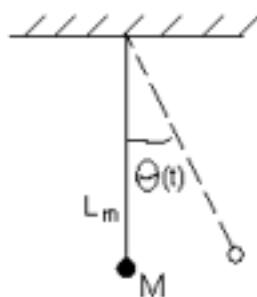
$$L_k \frac{d^2 q(t)}{dt^2} + \frac{q(t)}{C_k} = 0$$

- описание процессов в электрическом

колебательном контуре на базе ДУ.

q(t) - заряд конденсатора в момент времени t

$$T = 2\pi \sqrt{L_k C_k} \quad \text{- период колебаний}$$



$$M_M l_M^2 \frac{d^2 \theta(t)}{dt^2} + m_M g l_M = 0$$

$$T_n = 2\pi \sqrt{\frac{l_M}{g}}$$

- период

$$h_2 = L_k = M_M l_M^2$$

$$h_1 = 0$$

$$h_0 = \frac{1}{C_k} = M_M g l_M$$

$$h_2 \frac{d^2 z(t)}{dt^2} + h_1 \frac{dz(t)}{dt} + h_0 z(t) = 0$$

$h_1, h_2, h_3$  - внутренние параметры системы

$z(t)$  - состояние системы в момент времени  $t$

Если рассматриваемая система взаимодействует с внешней средой, то появляется входное воздействие  $x(t)$  и непрерывно детерминированная система имеет вид:

$$h_2 \frac{d^2 z(t)}{dt^2} + h_1 \frac{dz(t)}{dt} + h_0 z(t) = x(t)$$

При этом состояние системы  $S$  в данном случае можно рассматривать как выходную характеристику, т.е. полагать, что  $y = z$ . Следовательно, использование D - схем позволяет формализовать процесс функционирования непрерывно детерминированных систем и оценить их характеристики применяя аналитический или имитационный подход, реализованный в виде соответствующего языка моделирования непрерывных систем или использования аналоговой или гибридных средств вычислительной техники.

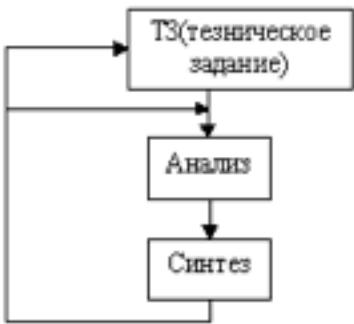
## [18.09.2006][Лекция 5]

### Формализация и алгоритмизация процесса функционирования сложных систем.

*Сущность компьютерного моделирования сложной системы* состоит в проведении на компьютере эксперимента с моделью, которая в нашем случае представляет собой некоторый программный комплекс, описывающий формально или алгоритмически поведения элементов системы в процессе её функционирования, т.е. в их взаимодействии друг с другом и внешней средой.

Сформулируем основные требования, предъявляемые к модели.

1. **Полнота модели** – должна предоставлять пользователю возможность получения необходимого набора характеристик, оценок системы, с требуемой точностью и достоверностью.
2. **Гибкость модели** – должна давать возможность воспроизведения различных ситуаций при варьировании структуры, алгоритмов и параметров системы. Причем структура должна быть блочной, т.е. допускать возможность замены, добавления, исключение некоторой части без переделки всей модели.
3. **Компьютерная реализация модели** должна соответствовать имеющимся техническим ресурсам.



Процесс моделирования, включая разработку и компьютерную реализацию модели является итерационным. Этот итерационный процесс продолжается до тех пор, пока не будет получена модель, которую можно считать адекватной в рамках решения поставленной задачи при исследовании или проектировании системы.

### **Основные этапы моделирования больших систем**

1. Построение концептуальной (описательной) модели системы и её формализация;
2. Алгоритмизация модели и её машинная реализация;
3. Получение и интерпретация результатов моделирования;

На первом этапе формулируется модель и строится её формальная схема, т.е. основным назначением данного этапа является переход от содержательного описания объекта к его математической модели. Этот этап наиболее ответственный и наименее формализованный.

Последовательность действий:

1. Проведение границы между системой и внешней средой.
2. Исследование моделируемого объекта с точки зрения выделения основных составляющих функционирования системы (по отношению к поставленной цели)  
 >>  
*Что является основным критерием выявления основных составляющих функциональной системы? – Цель!*  
 >>
3. Переход от содержательного описания модели к формализованному описанию свойств функционирования модели, т.е. к её **концептуальной модели**. Это сводится к исключению из рассмотрения некоторых второстепенных элементов. Полагают, что они не указывают существенного явления на ход процесса исследуемой модели.
4. Оставшиеся элементы модели группируются в блоки:
  - Блоки I-ой группы представляют собой имитатор событий внешних воздействий.
  - Блоки II-ой группы являются собственно моделью процесса функционирования.
  - Блоки III-ой группы являются вспомогательными и служат для реализации блоков I и II группы. Так же эти блоки обеспечивают корректность ввода данных, приемлемость результатов и т.д.

5. Процесс функционирования системы, так разбивается на подпроцессы, чтобы построение модели подпроцесса было элементарно и не вызывало особых трудностей.

(Должно реализовываться подбором типовых математических схем)

На втором этапе моделирования – этапе алгоритмизации и компьютерной реализации, математическая модель сформированная на первом этапе воплощается в конкретную программную модель.

Исходный материал – блочная логическая схема.

Последовательность действий:

1. Разработка схемы моделирующего алгоритма.
2. Разработка схемы программы.
3. Выбор технических средств для реализации программной модели.
4. Процесс программирования и отладки.
5. Проверка достоверности программы на тестовых примерах.
6. Составление технической документации.

На 3-м этапе (получение и интерпретация результатов) компьютер используется для проведения рабочих расчетов по готовой программе. Результаты этих расчетов позволяют проанализировать и сделать выводы о характеристиках процессов функционирования исследуемой схемы.

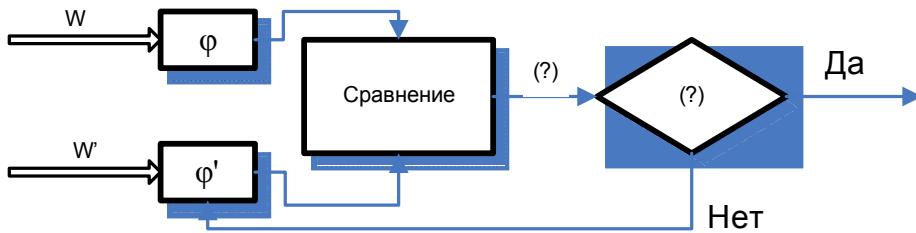
Последовательность действий:

1. Планирование машинного эксперимента с моделью. Составление плана проведения эксперимента с указанием комбинаций, переменных и параметров для которых должен проводится эксперимент.  
*Главная задача* – дать максимальный объем информации об объекте моделирования при минимальных затратах машинного времени.
2. Проведение собственных расчетов (контрольная калибровка модели).
3. Статистическая обработка результатов расчетов и представление результатов в наглядной форме.
4. Интерпретация результатов моделирования. Подведение итогов.
5. Составление технической документации.

Различают *стратегическое* и *практическое* планирование:

- При стратегическом планировании ставится задача построения оптимального плана эксперимента для достижения данной цели поставленной перед моделированием (оптимизация структуры алгоритмов и параметров системы).
- Тактическое планирование преследует частные цели оптимальной реализации каждого конкретного эксперимента из множества необходимых заданных при стратегическом планировании.

Схема итеративной калибровки модели.



### Три основных класса ошибок:

1. Ошибки формализации. Как правило возникают, когда модель недостаточно подробна определена.
2. Ошибки решения. Некорректный или слишком упрощенный метод построения модели.
3. Ошибки задания параметров системы.

Проверка адекватности модели некоторой системы заключается в анализе её соразмерностей, а так же равнозначности системы.

Адекватность часто нарушается из-за идеализации внешних условий и режимов функционирования, пренебрежением некоторых случайных факторов. Простейшей мерой адекватности может служить отклонение некоторой характеристики Y-оригинала от Y-модели ( $\Delta y = |y_{\text{ориг}} - y_{\text{мод}}|$ ). Считают что модель адекватна с системой, если вероятность того что отклонение  $\Delta y$  не превышает предельной величины дельта, больше допустимой вероятности.

Однако, фактическое использование данного критерия невозможно, т.к. для проектируемых или модернизируемых систем, отсутствует информация по выходным характеристикам объекта. Система отслеживается не по одной, а по множеству характеристик.

Замечание: Характеристики могут быть случайными величинами и функциями.

Замечание: Отсутствует возможность априорного точного задания предельных отклонений и допустимых вероятностей.

На практике оценка адекватности обычно проводится путем экспертного анализа. Разумности результатов моделирования.

Виды проверок:

- Проверка моделей элементов
- Проверка моделей внешних воздействий
- Проверка концептуальной модели
- Проверка формализованной и математической модели
- Проверка способов измерения и вычисления выходных характеристик.
- Проверка программной модели

Если по результатам проверки выделяется недопустимое рассогласование модели и системы, возникает необходимость в её корректировки или изменения.

Выделяют следующие типы изменений:

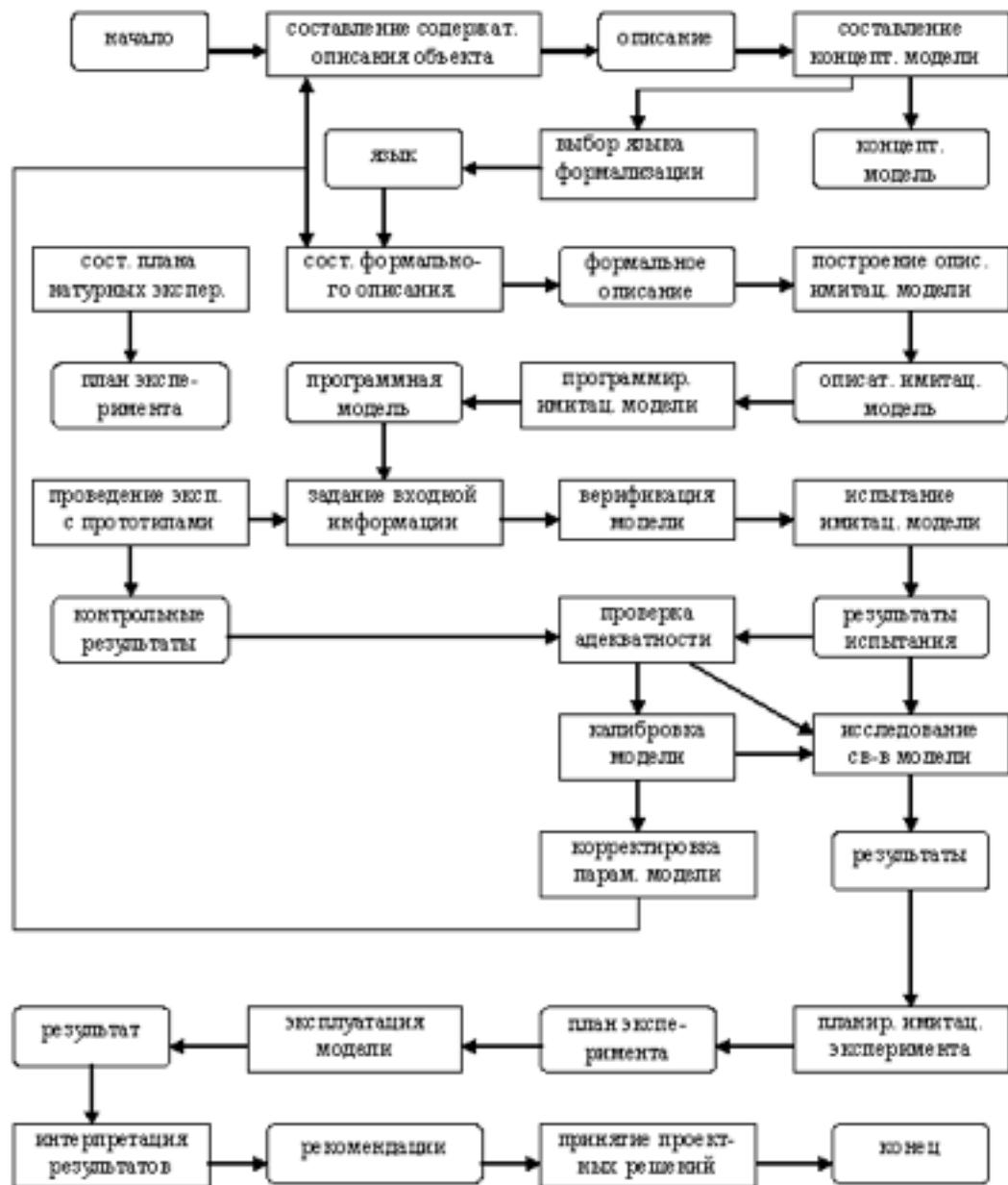
1. Глобальные. Возникают в случае методических ошибок в концептуальной или математической модели (проще всё начать сначала, чем исправлять).

2. Локальные. Связаны с уточнением некоторых параметров и алгоритмов. Заменить компоненты модели на более точные.
3. Параметрические изменения некоторых специальных характеристик называемых **калибровочными**. Как правило, эти характеристики мы задаем сами.

Завершается этот этап определением **области пригодности модели**. Под областью пригодности модели понимается множество условий при соблюдении которых, точность результатов моделирования находится в допустимых пределах.

## [25.09.06][Лекция 6]

### Схема взаимосвязи технологических этапов моделирования.



Всё что связано с процессом – это прямоугольник.  
С результатом – овал.

## **Основные понятия теории планирования эксперимента.**

Следующим этапом после создания математической модели и её программной реализации является постановка вычислительного эксперимента. В теории планирования эксперимента исследуемый объект рассматривается как черный ящик, имеющий входы Х и выходы Y. Переменные Х принято называть факторами. Факторы в эксперименте могут быть – качественными и количественными.

Качественные факторы можно квантифицировать или прописать им числовые обозначения, тем самым перейти к количественным значениям. В дальнейшем будем полагать, что все факторы являются количественными, представленными непрерывными величинами.

Переменным Х можно сопоставить геометрическое понятие факторного пространства, т.е. пространства, координатные оси которого соответствуют значениям факторов. Совокупность конкретных значений всех факторов образует точку в многомерном факторном пространстве.

Примеры факторов: интенсивность потока запросов в базе данных, скорость передачи данных по каналу, объем запоминающего устройства и т.д.

Но, к сожалению не все так хорошо. На объект воздействуют возмущающие факторы. Они являются случайными и не поддаются управлению.

Область планирования задается интервалами возможного изменения фактора:  
 $X_{\min} \leq X_i \leq X_{\max}, i = \overline{1, k}$ , k - количество факторов.

**Нормализация факторов** – преобразование натуральных значений факторов в безразмерные кодированные величины. Переход i-ого значения задается следующей

$$X_i = \frac{X_i - X_{i0}}{\Delta X_i}$$

формулой:  $X_i$  - переход безразличного фактора  $X_i$

$X_i$  - натуральное значение фактора

$X_{i0}$  - натуральное значение основного уровня фактора, соответствующее нулю в безразмерной шкале

$\Delta X_i$  - интервал варьирования

Совокупность основных уровней всех факторов представляет собой точку в пространстве параметров, называемой центральной точкой плана или центром эксперимента.

С геометрической точки зрения нормализация факторов равносильна линейному преобразованию пространства факторов, при котором проводятся две операции.

- 1) Перенос начала координат в точку соответствующую значениям основных уровней факторов
- 2) Сжатие/растяжение пространства в направлении координатных осей

**Активный эксперимент** включает: систему воздействий, при которых воспроизводится функционирование объекта и регистрация отклика объекта.

План эксперимента задаёт совокупность данных, определяющих количество, условия и порядок реализации опытов.

Опыт составляет элементарную часть эксперимента и предусматривает воспроизведение исследуемого явления в конкретных условиях с последующей регистрацией результатов.

В условиях случайности при одних и тех же условиях проводятся параллельные или повторные опыты, для получения статистически устойчивых результатов.

Опыт U предполагает задание конкретных значений фактора X:  $U = X_{1u}, X_{2u}, \dots, X_{ku}$ . Совокупность значений факторов во всех N точках плана эксперимента образует матрицу плана:

$$\Delta = \begin{pmatrix} X_{11} & \boxed{\text{X}} & X_{k1} \\ \boxed{\text{X}} & \boxed{\text{X}} & \boxed{\text{X}} \\ X_{z1} & \boxed{\text{X}} & X_{zz} \end{pmatrix}$$

Строки матрицы соответствуют опытам, столбцы - фактором.

Элемент матрицы  $X_{ij}$  задает значение i-ого фактора в j-ом опыте.

Реализовав испытание N факторного пространства, определенным фактором эксперимента, получим вектор наблюдений, имеющий следующий вид:

$$y = \begin{pmatrix} y_1 \\ y_2 \\ \boxed{\text{X}} \\ y_n \end{pmatrix}, \text{ где текущий } y_i \text{ соответствует i-ой точке плана}$$

Зависимость отклика от факторов носит название **функции отклика**, а геометрическое представление функции отклика - **поверхностью отклика**. Функция отклика рассматривается как показатель качества или эффективности объекта. Этот показатель является функцией от параметров, в качестве которых выступают факторы.

$$M(y) = \beta f(x);$$

$\beta$  – вектор неизвестных параметров модели:

$$\beta = (\beta_0, \beta_1, \dots, \beta_h), h+1$$

$f(x)$  - вектор заданных базисных функций.

$M(y)$  - математическое ожидание функции отклика

Такое представление функции отклика соответствует линейной по параметрам модели регрессионного анализа, т.е. функция отклика есть линейная комбинация базисных функций от фактора.

Из-за влияния на результаты эксперимента случайных воздействий, истинное значение коэффициентов можно определить только приближенно. Оценку вектора  $\beta$  находят по результатам экспериментов. В ходе, которых мы получаем значение  $y_u$  при заданных значениях факторов  $X_u$ . Эти оценки обычно оцениваются с помощью метода наименьших квадратов. Если не принимать специальных мер, то оценки коэффициентов  $\beta$  станут взаимозависимыми и полученное выражение для функции отклика можно рассматривать как интерполяционную формулу, что затрудняет её физическую интерпретацию и последующие расчеты.

### Получение независимых результатов.

Для этого нужно формировать специальным образом матрицу плана. И эти величины будут характеризовать вклад каждого фактора в значение функции отклика.

Основная задача - определение основных формул функций отклика  $y'$ .

В большинстве случаев вид этой функции, получаемой из теоретических соображений, является сложным для практического применения.

Функции принято обозначать в некотором универсальном виде - в виде полинома.

Тогда системой базисной функции является совокупность степенных функций с целыми неотрицательными значениями показателя степени:

$$y' = \beta_0 + \beta_1 X_1 + \dots + \beta_k X_k + \beta_{12} X_1 X_2 + \beta_{13} X_1 X_3 + \dots + \beta_{kk} X_k^2 + \varepsilon$$

$\varepsilon$  - случайная величина, характеризующая ошибку опыта.

Такая функция отклика линейна относительно неизвестных коэффициентов и будет полностью определена, если задана степень полинома и коэффициенты. Степень полинома обычно задается исследователем априорно. На практике широкое распространение имеют полиномы первого и второго порядка. Коэффициенты полинома принято называть эффектами факторов. К большинству сложных систем применим принцип Павето, согласно которому 20% факторов определяют свойства системы на 80%. Поэтому первоначальной задачей при исследовании имитационной модели является отсеивание несущественных факторов, позволяющие упростить вычисления функции отклика. Одним из методов решения этой задачи является метод дисперсионного анализа.

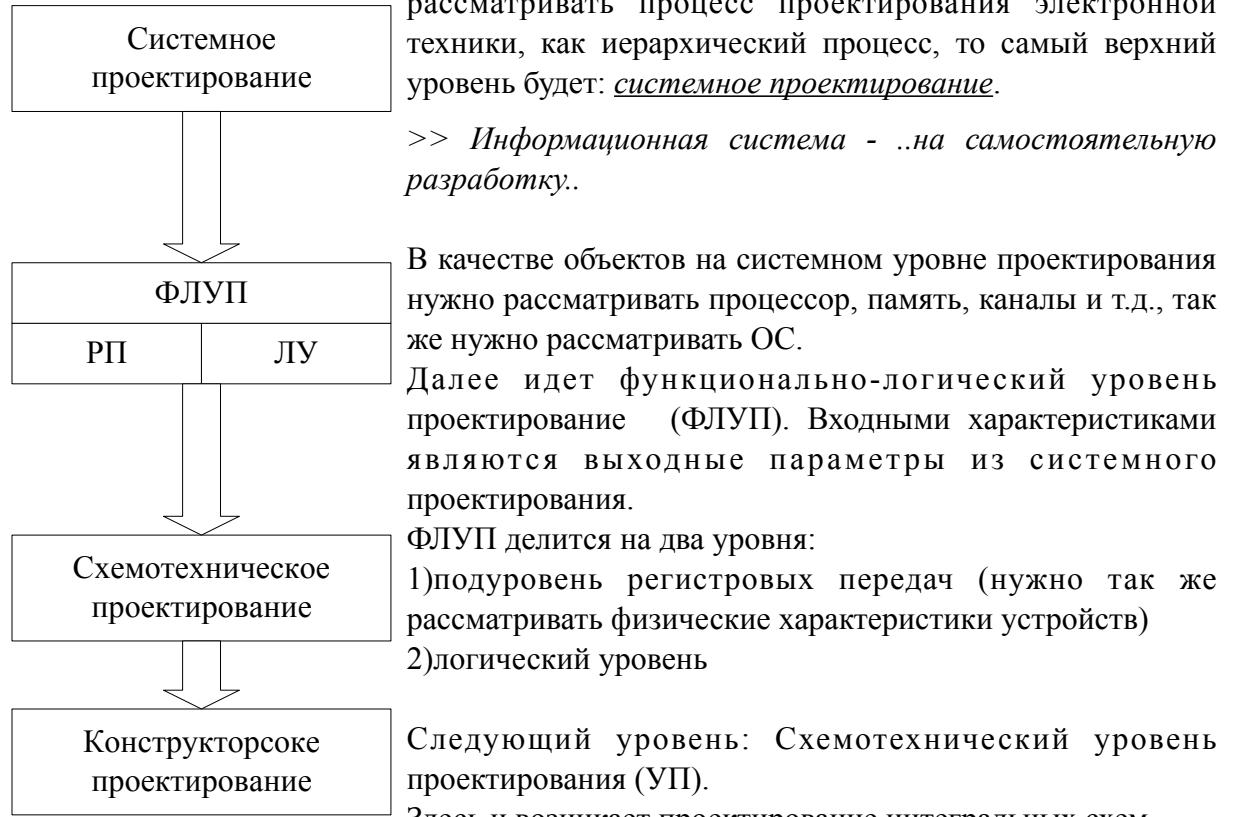
### **Виды планирования эксперимента.**

Можно выделить **стратегическое** и **тактическое** планирование эксперимента (...!) Тактическое планирование: дать понятие и пример.

[29.09.06][Лекция 7]

## **Вычислительная система, как объект моделирования.**

В теории проектирования ВТ принято выделять уровни проектирования. Если рассматривать процесс проектирования электронной техники, как иерархический процесс, то самый верхний уровень будет: системное проектирование.



Далее: Конструкторский УП. Здесь рассматриваются вопросы о теплообмене, охлаждении и т.д.

*Вопрос: можно ли начать проектирование этой схемы снизу вверх?*

*Ответ:*

### **Моделирование на системном уровне.**

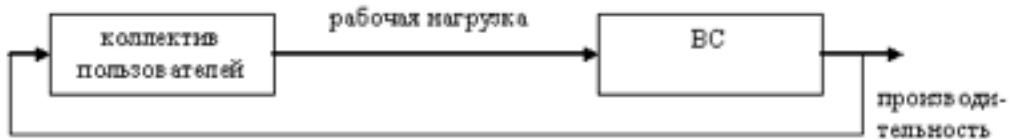
При моделировании новых и модернизации существующих вычислительных систем и сетей необходимо предварительно оценивать эффективность их функционирования с учетом различных вариантов структурной организации. Эти варианты могут отличаться составом и характеристиками устройств. Структурой межмодульных связей, режимами работы и алгоритмами управления. Для оценок таких структур используют модели вычислительных систем. Под вычислительной системой будем понимать комплект аппаратных (процессор, память, ву) и программных средств ( ОС ), которые в совокупности выполняют определенные рабочие функции.

>> ОС – множество согласованных управляющих программ [опр: на примитивном уровне]

**Коллектив пользователей** – это сообщество таких людей, которые используют вычислительную систему для удовлетворения своих нужд по обработке информации.

Входные сигналы (программы, команды, данные), которые создаются коллективом пользователей, называются **рабочей нагрузкой**.

*Схема вычислительной установки:*



**Индекс производительности** (ИП) – описатель, который используется для представления производительности системы. Различают:

- Качественные ИП. Тип процессора – RISC/CISC, мощность системы команд.
- Количественные ИП. **Пропускная способность** – объем информации обрабатываемый в единицу времени. **Время ответа (реакции)** – время между предъявлением системе входных данных и появлением соответствующей выходной информации. **Коэффициент использования оборудования** – отношение времени использования указанной части системы в течение заданного интервала времени к длительности этого интервала.

**Концептуальная модель** включает в себя сведения о выходных и конструктивных параметрах системы, её структуре, особенности работы каждого ресурса (элемента системы), характере взаимодействия между ресурсами. Как правило, включается постановка прикладной задачи, определяющей цели моделирования исходной системы, а так же исходные данные для исследования системы.

**Формализованная схема** представляет собой, как правило, некоторую сложную систему массового обслуживания.

Основные задачи, которые необходимо решить:

- 1) Определение принципов организации вычислительной системы;
- 2) Выбор архитектуры, уточнение функции и их разделение на подфункции, реализуемое аппаратным или программным способом;
- 3) Разработка структурной схемы, т.е. определение состава устройств и способов их взаимодействия;
- 4) Определение требований к выходным параметров устройств и формирование технического задания для разработки отдельных устройств.

## Непрерывно стохастические модели (Q-схемы)

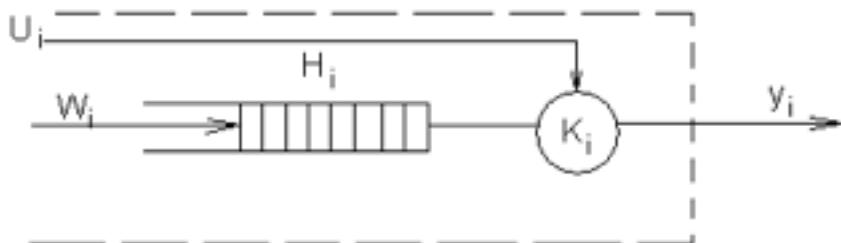
Особенность непрерывно стохастической модели будем рассматривать на примере систем массового обслуживания (СМО) в качестве типовых математических моделей. При этом используемая система формализуется как некая система обслуживания. Характерным для таких объектов является случайное появление требований (заявок) на обслуживание и завершение обслуживания в случайные моменты времени. Т.е. характер функционирования устройств носит стохастический порядок.

**Основные понятия теории массового обслуживания.**

В любом элементарном акте обслуживания можно выделить две основные составляющие:

- 1) Ожидание обслуживания
- 2) Собственно, обслуживание

Некоторые виды обслуживания некоторого оборудования:



ОА – обслуживающий аппарат

К – канал

Прибор обслуживания ( $i$ -ый) состоит из:

- накопителя заявок, в котором может одновременно находится  $L_i = \{0, L_i^n\}$ , где  $L_i^n$  – емкость  $i$ -ого накопителя
- канала обслуживания заявок.

**Потоком событий** называется последовательность событий происходящих одно за другим в какие-то случайные моменты времени.

Поток событий называется **однородным**, если он характеризуется только моментами поступления этих событий (вызывающие моменты) и задается временной последовательностью:  $t_i = \{0, t_1, \dots, t_n\}$ ,  $t_{i-1} \leq t_i \leq t_{i+1}$

Поток называется **неоднородным**, если он задается следующей совокупностью  $\{t_n, f_n\}$ , где  $t_n$  – вызывающий моменты,  $f_n$  – набор признаков события( наличие приоритета, принадлежность к тому или иному типу заявки).

Если интервал времени между сообщениями независимыми между собой являются случайными величинами, то такой поток называется потоком с **ограниченным** последействием.

Поток событий называется **ординарным**, если вероятность того, что на малый интервал времени  $\Delta t$  примыкающий к моменту времени  $t$  попадет более одного события, пренебрежительно мала по сравнению с вероятностью того что на этот же интервал  $\Delta t$  попадает ровно одно событие.

Поток называется **стационарным**, если вероятность появления того или иного числа событий на некотором интервале времени зависит лишь от длины интервала и не зависит от того, где на оси времени взят этот участок.

Для ординарного потока среднее число сообщений наступивших на участке  $\Delta t$  примыкающих к некоторому моменту времени  $t$  будет равно  $P_{>1}(t, \Delta t) + P_1(t, \Delta t) \sim P_1(t, \Delta t)$ .

Тогда среднее число сообщений наступивших на участке времени  $\Delta t$  составит:

$$\lim_{t \rightarrow 0} \frac{P_1(t, \Delta t)}{\Delta t} = \lambda(t)$$

*- интенсивность ординарного потока.*

Для стационарного потока – его интенсивность не зависит от времени и представляет собой постоянное значение равное среднему числу событий наступающих в единицу времени.

**Поток заявок** ( $^{(0)}_i$ ), т.е. интервалы времени между моментами появления заявок на входе канала (это подмножество неуправляемых переменных)

**Поток обслуживания** ( $^{(U_i)}$ ) - т.е. интервалы времени между началом и окончанием обслуживанием заявок, принадлежащим подмножеству управляемых заявок.

Заявки обслуженные каналом или заявки покинувшие прибор необслуженными, образуют выходной поток. Процесс функционирования  $i$ -ого прибора можно представить как процесс изменения состояний его элементов во времени.

Переход в новое состояние для  $i$ -ого прибора означает изменение количества заявок, которые находятся в накопителе или канале:  $Z_i = (Z_i^n, Z_i^k)$

Где  $Z_i^n$  – состояние накопителя, если он = 0, то накопитель пуст (нет заявок), если количество заявок совпадает с емкостью накопителя, то накопитель полон;  $Z_i^k$  – состояние канала (0 – свободен или 1 – занят).

В практике моделирования элементарные Q-схемы обычно объединяют, при этом, если каналы различных приборов обслуживания соединены параллельно, то имеет место **многоканальное обслуживание**. А если последовательно – **многофазное обслуживание**. Таким образом для задания Q-схемы необходимо использовать оператор сопряжения R, отражающий взаимосвязь элементов структуры. Различаются разомкнутые и замкнутые Q-схемы.

Разомкнутые – выходной поток заявок не может поступить к какому либо элементу, т.е. отсутствует обратная связь

Замкнутые – есть обратная связь.

## [2.10.2006][Лекция 8]

Собственными внутренними параметрами Q-схемы будут являться:

- количество фаз
- количество каналов в каждой фазе
- количество накопителей каждой фазы
- ёмкость накопителя.

В зависимости от ёмкости накопителя в теории массового обслуживания применяют следующую терминологию: если ёмкость равна нулю (т.е. накопитель отсутствует, а есть только канал), то система с потерями. Если ёмкость стремится к бесконечности, то система с ожиданием, т.е. очередь заявок неограничена.

**Система смешанного типа.**

Для задания Q-схемы так же необходимо описать алгоритм её функционирования, который определяет набор правил поведения заявок в системе в различных ситуациях. Неоднородность заявок, отражающая процессы в той или иной реальной системе, учитывается с помощью введения классов приоритетов.

Весь набор возможных алгоритмов поведения заявок в Q-схеме можно представить в виде оператора:

$$Q = (W, U, R, H, Z, A)$$

Где  $W$  - подмножество входных потоков;

$U$  - подмножество потока обслуживания;

$R$  - оператор сопряжения элементов структуры;

$H$  - подмножество собственных параметров;

$Z$  - множество состояний системы;

$A$  - оператор алгоритмов поведения и обслуживания заявок;

Для получения соотношений связывающих характеристики, которые определяют функционирование Q-схемы, вводят некоторые допущения относительно входных потоков, функций распределения, длительности обслуживания запросов, дисциплин обслуживания.

Для математического описания функционирования устройств, процесс функционирования которого развивается в случайном порядке, могут быть применены математические модели для описания так называемых **Марковских случайных процессов**.

Случайный процесс называется Марковским, если он обладает следующим свойством – для каждого момента времени  $t_0$  вероятность любого состояния системы в будущем (т.е. в какой-то момент времени  $t > t_0$ ) зависит только от состояния системы в настоящем и не зависит от того, когда и каким образом система пришла в это состояние. Иначе, в Марковском случайном процессе будущее его развитие зависит только от его настоящего состояния и не зависит от исторического процесса.

/\* реально таких систем, конечно, не существует. Но существуют механизмы, которые позволяют свести к этим процессам.\*/

Для Марковских процессов обычно составляют уравнения Колмогорова.

В общем виде уравнения Колмогорова выглядят следующим образом:

$$F = (p'(t), p(t), \lambda) = 0,$$

где  $\lambda$  - вектор, определяющий некоторый набор коэффициентов присущих системе

Для стационарного соотношения:

$$\Phi = (p(t), \lambda) = 0,$$

что дает возможность для стационарной зависимости получить

$$p = p(\lambda).$$

А затем связать выходные характеристики через набор коэффициентов соответствующих системе:

$$Y = Y(p(\lambda))$$

Последнее соотношение представляет собой зависимость выходных параметров от некоторых внутренних параметров модели, и имеют название **базисной модели**.

В результате всего нам нужно найти:

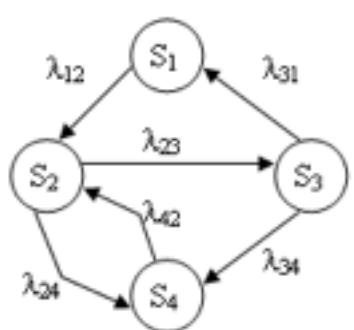
$$\lambda = \lambda(X, U, H)$$

- которая будет называться **интерфейсной моделью**.

Следовательно, математическая модель системы строится как совокупность базисной и интерфейсной модели, что позволяет использовать одни и те же базисные модели, для различных задач проектирования осуществляя настройку на соответствующую задачу посредством изменения только интерфейсной модели. Для Q-схем математическая модель должна обеспечивать вычисление времени реакции и определения производительности системы.

Пример: пусть есть некоторая система  $S$ , имеющая конечный набор состояний (будем рассматривать для 4 состояний).

Получаем ориентированный граф:



$\lambda_{i,j}$  - плотности вероятностей для множества состояний.

Найдем вероятность  $p_1(t)$ , т.е. вероятность того что в момент  $t$  система будет находиться в состоянии  $S_1$ .  
Придадим  $t$  малое приращение  $\Delta t$  и найдем, что в момент времени  $t + \Delta t$  система будет находиться в состоянии  $S_1$ .

Это может быть реализовано двумя способами:

1. В момент  $t$  система  $S$  уже была в состоянии  $S_1$  и за время  $\Delta t$  не вышла из него.
2. В момент  $t$  система была в состоянии  $S_3$  и за время  $\Delta t$  перешла из него в состояние  $S_1$ .

Вероятность первого способа найдем как произведение вероятности  $p_1(t)$  на условную вероятность того, что будучи в состоянии  $S_1$  система за время  $\Delta t$  не перейдет из него в состояние  $S_2$ . Это условная вероятность с точностью до бесконечно малых величин высших порядков будет равна:

$$p_1(t)(1 - \lambda_{1,2}\Delta t)$$

Аналогично вероятность второго способа равна вероятности того что в следующий момент  $t$  была в состоянии  $S_3$  умноженную на условную вероятность перехода в состояния  $S_1$ , т.е.:

$$p_3(t)\lambda_{3,1}\Delta t$$

$$p_1(t + \Delta t) = p_1(t)(1 - \lambda_{1,2}\Delta t) + p_3(t)\lambda_{3,1}\Delta t$$

$$\lim_{\Delta t \rightarrow \infty} \frac{p_1(t + \Delta t) - p_1(t)}{\Delta t} = -\lambda_{1,2}p_1(t) + \lambda_{3,2}p_3(t)$$

$$\Rightarrow \frac{dp_1(t)}{dt} = -\lambda_{1,2}p_1(t) + \lambda_{3,2}p_3(t)$$



Мы вывели уравнение колмагорова для первого состояния.

Выведем далее для 2, 3 и 4 состояний.

$$p_2(t + \Delta t) = p_1(t)\lambda_{1,2}\Delta t + p_4(t)\lambda_{4,1}\Delta t + p_2(t)(1 - \lambda_{2,3}\Delta t) + p_2(t)(1 - \lambda_{2,4}\Delta t)$$

$$p'_2(t) = p_1(t)\lambda_{1,2} + p_4(t)\lambda_{4,1} - p_2(t)(\lambda_{2,3} + \lambda_{2,4})$$

$$p_3(t + \Delta t) = p_2(t)\lambda_{2,3}\Delta t + p_3(t)(1 - \lambda_{3,1}\Delta t) + p_3(t)(1 - \lambda_{3,4}\Delta t)$$

$$p'_3(t) = p_2(t)\lambda_{2,3} - p_3(t)(\lambda_{3,1} + \lambda_{3,4})$$

$$p'_4(t) = p_3(t)\lambda_{3,4} + p_2(t)\lambda_{2,4} - p_4(t)\lambda_{4,2}$$

Интегрирование данной системы дает искомые вероятности системы как ф-ции времени. Начальные условия берутся в зависимости от того какого было начальное состояние системы. Например, если в момент времени  $t = 0$ , система находилась в состоянии  $S_1$ , то начальное условие будет  $t_1$ .

Кроме того, необходимо добавлять условие нормировки (сумма вероятностей = 1).

Уравнение Колмогорова строится по следующему правилу: в левой части каждого уравнения стоит производная вероятности состояния, а правая часть содержит столько членов сколько стрелок связано с данным состоянием. Если стрелка направлена из состояния, то соответствующий член имеет знак "-", в состояние - "+". Каждый член равен произведению плотности вероятности перехода (интенсивности) соответствующий данной стрелке, умноженной на вероятность того состояния, из которого исходит стрелка.

### **Лабораторная работа №1.**

Определить среднее относительное время прибывания системы в предельном стационарном состоянии. Интенсивности переходов из состояния в состояние задаются в виде матрицы размером  $\leq 10$ .

Отчет: название, цель, теоретическая часть и расчеты.

Рассмотрим многоканальную систему массового обслуживания с отказами.

Будем нумеровать состояние системы по числу занятых каналов. Т.е. по числу заявок в системе.

Обзовем состояния:

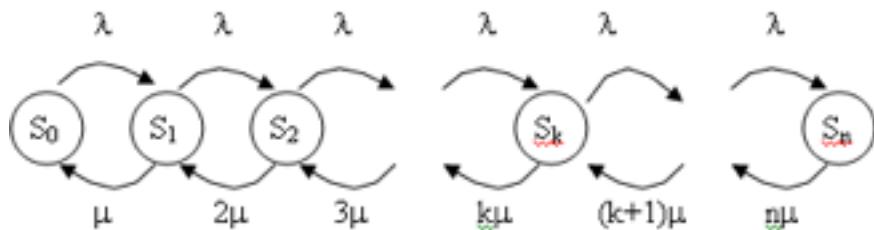
$S_0$  - все каналы свободны

$S_1$  - занят один канал, остальные свободны

$S_k$  - занято  $k$  каналов, остальные свободны

$S_n$  - заняты все n каналов

Граф состояний:



Разметим граф, т.е. расставим интенсивности соответствующих событий.

По стрелкам с лева на право система переводит один и тот же поток с интенсивностью  $\lambda$ .

Определим интенсивность потоков событий, переводящих систему справа налево.

Пусть система находится в  $S_1$ . Тогда, когда закончится обслуживание заявки, занимающей этот канал, система перейдет в  $S_0 \Rightarrow$  поток, переводящий систему в другое состояние, будет иметь интенсивность перехода  $\mu$ . Если занято 2 канала, а не один, то интенсивность перехода составит  $2\mu$ .

Уравнения Колмогорова:

$$\begin{cases} p_0' = -p_0\lambda + p_1\mu \\ p_1' = -p_1\lambda - p_1\mu + p_0\lambda + p_22\mu \\ p_2' = -p_2\lambda - p_22\mu + p_1\lambda + p_33\mu \\ p_k' = -p_k\lambda - p_kk\mu + p_{k-1}\lambda + p_{k+1}(k+1)\mu \\ p_n' = p_{n-1}\lambda - p_nn\mu \end{cases}$$

Предельные вероятности состояний  $p_0$  и  $p_n$  характеризуют установившийся режим работы системы массового обслуживания при  $t \rightarrow \infty$ .

$$p_0 = \frac{1}{1 + \frac{\lambda / \mu}{1!} + \frac{(\lambda / \mu)^2}{2!} + \dots + \frac{(\lambda / \mu)^n}{n!}}$$

$$p_k = \frac{(\lambda / \mu)^k}{k!} p_0$$

$\lambda / \mu = \rho$  - среднее число заявок, приходящих в систему за среднее время обслуживания одной заявки.

$$p_0 = \left[ 1 + \frac{\rho}{1!} + \frac{\rho^2}{2!} + \dots + \frac{\rho^n}{n!} \right]^{-1}$$

$$p_k = \frac{\rho^k}{k!} p_0$$

Зная все вероятности состояний  $p_0, \dots, p_n$ , можно найти характеристики СМО:

- *вероятность отказа* – вероятность того, что все  $n$  каналов заняты  

$$p_{OTK} = p_n = \frac{\rho^n}{n!} p_0$$
- *относительная пропускная способность* – вероятность того, что заявка будет принята к обслуживанию  

$$q = 1 - p_n$$
- среднее число заявок, обслуженных в единицу времени  

$$A = \lambda q$$

Полученные соотношения могут рассматриваться как базисная модель оценки характеристик производительности системы. Входящий в эту модель параметр

$$\frac{1}{\mu} = \frac{n_{np}}{V_{np}}$$

, является усредненной характеристикой пользователя. Параметр  $\mu$  является функцией технических характеристик компьютера и решаемых задач.

Эта связь может быть установлена с помощью соотношений, называемых интерфейсной моделью. Если время ввода/вывода информации по каждой задачи мало по сравнению со временем решения задачи, то логично принять, что время решения равно  $1 / \mu$  и равно отношению среднего числа операций, выполненных процессором при решении одной задачи к среднему быстродействию процессора.

$$\frac{1}{\mu} = \frac{n_{np}}{V_{np}}$$



[9.10.2006][Лекция 9]

Самостоятельно: Метод вложенных цепей Маркова

Требования к отчету: название, цель, краткие теоретические сведения (писать то что не знаешь), пример, текст программы.

### ***Не Марковские случайные процессы, сводящиеся к Марковским.***

Реальные процессы весьма часто обладают последействием и поэтому не являются Марковским. Иногда при исследовании таких процессов удается воспользоваться методами, разработанными для Марковских цепей. Наиболее распространенными являются:

1. Метод разложения случайного процесса на фазы (метод псевдо состояний)
2. Метод вложенных цепей

### **Метод псевдо состояний.**

Сущность метода заключается в том, что состояние системы, потоки переходов из которых являются немарковскими, заменяются эквивалентной группой фиктивных состояний, потом переходы, из которых уже являются Марковскими.

Условие статистической эквивалентности реального и фиктивного состояния могут в каждом конкретном случае выбираться по-разному. Очень часто может использоваться

следующее:  $\int_{t_1}^{t_2} (\lambda_i(\tau) - \lambda(\tau)) d\tau$ , где  $\lambda_i$  - эквивалентная интенсивность перехода в  $i$ -ой группе переходов, заменяющей реальный переход, обладающий интенсивностью  $\lambda_i(\tau)$ . За счет расширения числа состояний системы некоторые процессы удается точно свести к Марковским. Созданная таким образом система статистически эквивалентна или близка к реальной системе, и она подвергается обычному исследованию с помощью аппарата Марковских цепей.

К числу процессов, которые введением фиктивных состояний можно точно свести к Марковским относятся процессы под воздействием потоков Эрланга. В случае потока Эрланга  $k$ -ого порядка интервал времени между соседними событиями представляет собой сумму  $k$  независимых случайных интервалов, распределенных по показательному закону. Поэтому с введением потока Эрланга  $k$ -го порядка к Пуассоновскому осуществляется введением к псевдо состояний. Интенсивности переходов между псевдо состояниями равны соответствующему параметру потока Эрланга. Полученный таким образом эквивалентный случайный процесс является Марковским, т.к. интервалы времени нахождения его в различных состояниях подчиняются показательному закону.

Пример. Устройство  $S$  выходит из строя с интенсивностью  $\lambda$ , причем поток отказов Пуассоновский. После отказа устройство восстанавливается. Время восстановления распределено по закону Эрланга 3 порядка с функцией плотности  $f_2(t) = 0.5\mu(\mu t)^2 e^{(-\mu t)}$ .

Найти предельные вероятности возможных состояний системы.

*Решение.*

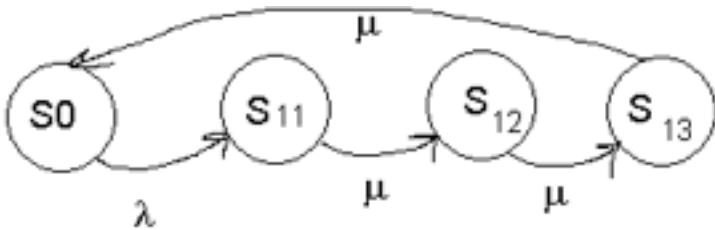
Пусть система может принимать 2 возможных состояния:

$S_0$  - устройство исправно;

$S_1$  - устройство отказалось и восстанавливается

Переход из  $S_0$  в  $S_1$  осуществляется под воздействием пуассоновского потока, а из  $S_1$  в  $S_0$  - потока Эрланга.

Представим случайное время восстановления в виде суммы 3x случайных временных интервалов, распределенных по показательному закону с интенсивностью  $\mu$ .



$$\begin{cases} p'_0 = 0 = -\lambda p_0 + \mu p_{13} \\ p'_{11} = 0 = -\mu p_{11} + \lambda p_0 \\ p'_{12} = 0 = -\mu p_{12} + \mu p_{11} \\ p'_{13} = 0 = -\mu p_{13} + \mu p_{12} \\ p_0 + p_1 = 1 \\ p_1 = p_{11} + p_{12} + p_{13} \end{cases}$$

$$p_{13} = \frac{\lambda}{\mu} p_0; p_{11} = \frac{\lambda}{\mu} p_0$$

$$p_{12} = p_{11} = \frac{\lambda}{\mu} p_0$$

$$p_1 = \frac{3\lambda}{\mu} p_0$$

$$p_0 + \frac{3\lambda}{\mu} p_0 = 1 \Rightarrow p_0 = \frac{\mu}{\mu + 3\lambda}$$

$$p_1 = \frac{3\lambda}{\mu + 3\lambda}$$

$$P_0 = \frac{\mu}{\mu + 3\lambda}, \quad P_1 = \frac{3}{3 + \frac{\mu}{\lambda}}$$

Ответ:

### Метод вложенных цепей Маркова.

Вложенные цепи Маркова образуются следующим образом. В исходном случайному процессе выбираются такие случайные процессы, в которых характеристики образуют Марковскую цепь. Моменты времени обычно являются случайными и зависят от свойств исходного процесса. Затем обычными методами теории Марковских цепей исследуются процессы только в эти характерные моменты. Случайный процесс называется **полумарковским** (с конечным или счетным множеством состояний) если заданы переходы состояний из одного состояния в другое и распределение времени пребывания процессов в каждом состоянии. Например, в виде функции распределения или функции плотности распределения.

<остальное самостоятельно>

### Метод статистических испытаний. Метод Монте-Карло.

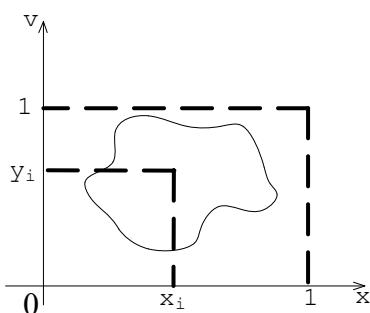
В СМО поток заявок редко бывает Пуассоновским и еще реже наблюдается распределенный закон.

Для произвольных потоков событий переводящих систему из состояния в состояние. Аналитические решения получены только для отдельных частных случаев. Когда построение аналитической модели является по той или иной причине трудно осуществимым ставится метод статистических испытаний.

\*Когда этот метод нашел реальное применение: с развитием компьютеров.

**Идея метода:** вместо того чтобы описывать случайные явления с помощью аналитической зависимости производится т.н. «розыгрыш», т.е. моделирование «случайного» явления с помощью некоторой процедуры дающей «случайный» результат. Проведя такой розыгрыш достаточно большое количество раз, получаем **статистический материал**, т.е. множество реализаций случайного явления. Дальше эти результаты могут быть обработаны статистическими методами математической статистики.

Метод Монте-Карло был предложен в 1948 году Фон-Нейманом как метод численного решения некоторых математических задач.



Введем в некотором единичном квадрате случайную величину. Задача стоит в определении её площади.

**Суть метода:**

1. Вводим в некотором единичном квадрате любую поверхность  $S$ .
2. Любым способом получаем 2 числа  $x_i, y_i$ , подчиняющиеся равномерному закону распределения случайной величины на интервале  $[0, 1]$ .
3. Полагаем, что одно число определяет координату  $x$ , второе – координату  $y$
4. Анализируем принадлежность точки  $(x, y)$  фигуре. Если принадлежит, то увеличиваем значение счетчика на 1.
5. Повторяем  $n$  раз процедуру генерации 2х случайных чисел с заданным законом распределения и проверку принадлежности точки поверхности  $S$ .
6. Определяем площадь фигуры как количество попавших точек, к количеству сгенерированных.

$$\varepsilon \leq \sqrt{\frac{1}{n}}$$

Фон-Нейман доказал, что погрешность

**Преимущество** метода статистических испытаний: в его универсальности, которая обуславливает его возможность статистического исследования объекта, причем всестороннего. Но для реализации этого исследования необходимы довольно полные статистические сведения о параметрах элемента.

**Недостаток:**

Большой объем требующихся вычислений, равный количеству обращений к модели. Поэтому вопрос выбора величины  $n$  имеет важнейшее значение. Уменьшая  $n$  – повышаем экономичность расчетов, но одновременно ухудшаем их точность.

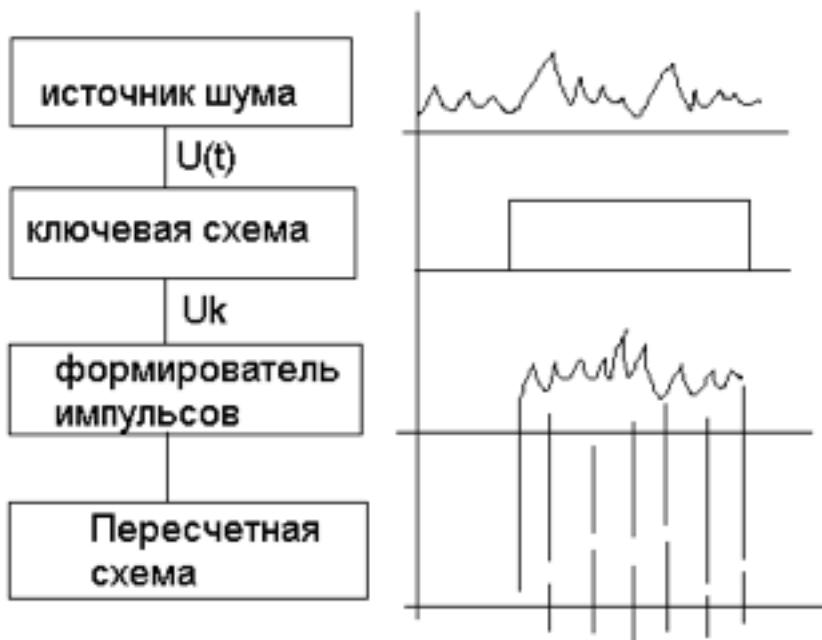
### **Способы получения псевдослучайных чисел.**

При имитационном моделировании системы одним из основных вопросов является стохастических воздействий. Для этого метода характерно большое число операций со случайными числами или величинами и зависимость результатов от качества последовательностей случайных чисел. На практике используется 3 основных способа:

1. **Аппаратный** (физический)
2. **Табличный** (файловый)
3. **Алгоритмический** (программный)

#### **Аппаратный способ.**

Аппаратный представляет из себя шум.



Случайные числа вырабатываются случайной электронной приставкой (генератор случайных чисел), служащей, как правило, в качестве одного из внешних устройств. Реализация этого способа обычно не требует дополнительных вычислений, а необходима только операция обращения к ВУ. В качестве физического эффекта, лежащего в основе таких генераторов чаще всего используют шумы в электронных приборах.

Т.е. необходимо: источник шума, ключевая схема, формирователь импульсов, пересчетная (см. рис.)

### **Табличная схема.**

Случайные числа оформляются в виде таблицы и помещаются во внешнюю или оперативную память.

## **Алгоритмический способ.**

Способ основан на формировании случайных чисел с помощью специальных алгоритмов.

## **Преимущества и недостатки типов генерации случайных чисел.**

Маша ела кашу.

Способ	Достоинства	Недостатки
Аппаратный	1. Запас чисел неограничен 2. Расходуется мало операций 3. Не занимается место в оперативной памяти.	1. Требуется периодическая проверка на случайность 2. Нельзя воспроизводить последовательности 3. Используются специальные устройства. Надо стабилизировать
Табличный	1. Требуется однократная проверка 2. Можно воспроизводить последовательности	1. Запас чисел ограничен 2. Занимает место в оперативной памяти и требуется время на обращение к памяти
Алгоритмический	1. Однократная проверка 2. Можно многократно воспроизводить последовательности чисел 3. Относительно малое место в оперативной памяти 4. Не используются внешние устройства	1. Запас чисел последовательности ограничен её периодом 2. Требуются затраты машинного времени

---

## **Лабораторная работа №2.**

Сравнить эти два способа. Причем сравниваем их по критерию случайности. Т.е. придумать свой критерий случайности (можно конечно и в книжке посмотреть, но лучше самому)..  
количественная оценка..

---

К пятнице:

1. Алгоритм Марселя Зейнмана (?). (целочисленная арифметика) (3-я группа)

2. Функция, увеличивающая период последовательности стандартного генератора rand.
3. Способ Ленмара (?)

### **[13.10.2006][Лекция 10]**

В настоящем времени с помощью рекуррентных математических уравнений реализовано несколько алгоритмов генерирования псевдослучайных чисел. **Псевдослучайными** эти числа называются потому, что фактически они, даже пройдя все статистические испытания на случайность и равномерность распределения, остаются полностью **детерминированными**. Т.е. если каждый цикл работы генератора начинается с одними и теми же с исходными данными (константами и начальными условиями и значениями), то на выходе мы получаем одни и те же последовательности.

*Сфера применения.*

Генератор случайных чисел используется в программных приложениях связанных с конструированием ядерных реакторов, радиолокационного оповещения и обнаружения, поисков полезных ископаемых, многоканальные связи и т.д.

## ***Простейшие алгоритмы генерации последовательности псевдослучайных чисел***

Одним из первых способов получения последовательности псевдослучайных чисел было выделение дробной части многочлена первой степени:  $y_n = Ent(an + b)$

Если  $n$  пробегает значения натурального ряда чисел, то поведение  $y_n$  выглядит весьма хаотично. Физик Якобит доказал, что при рациональном коэффициенте  $a$  множество у конечно, а при иррациональном – бесконечно и всюду плотно в интервале  $[0, 1]$ . Для многочленов больших степеней такую задачу решил Герман Вей, т.е. он предложил критерий равномерности распределения любой функции от натурального ряда чисел. Называется это **эргодичностью** и заключается в том, что среднее по реализациям псевдослучайных чисел равно среднему по всему их множеству с вероятностью 1. Эти результаты далеки от практики, поэтому она используется только для действительных чисел, что затрудняет практическую её реализацию. Попытки замены настоящего иррационального числа его приближением на компьютере привели к тому, что полученные последовательности оканчиваются циклом с коротким периодом.

### **1. 1946 год, Фон Нейман.**

Каждое последующее число образуется возведением предыдущего в квадрат и отбрасыванием цифр. Способ с точки зрения случайности оказался нестабильным.

### **2. Лимер**

$$g_{n+1} = g_n k + c \bmod m$$

Для подбора коэффициентов  $k$ ,  $c$ ,  $m$  потрачены десятки лет. Подбор почти иррациональных  $k$  ничего не дает. Установили, что при  $c = 0$  и  $m = 2^n$

наибольший период достигается при нечетном начальном числе и при  $k = 3 + 8i$ ,  
 $k = 5 + 8i$ .

### 3. Форсайд

В 1977 году показал, что тройки последовательности чисел лежат на 15 параллельных плоскостях.

От отчаяния используют 2 и даже 3 разных генератора, смешивая их значения. Если бы разные генераторы не зависели, то сумма их последовательностей обладала дисперсией равной сумме дисперсий. Иначе случайность рядов возрастет при суммировании. Сейчас в системах программирования обычно используют конгруэнтные генераторы по алгоритму, предложенному национальному бюро стандартов США, который имеет длину  $2^{24}$ .

Генерация случайных чисел по алгоритму Зеймана.

$\{1, 1, 2, 3, 5, 8, 13, 21, \dots\}$

$\text{mod } 10$

$\{1, 1, 2, 3, 5, 8, 3, 1, \dots\}$

Переименовываем с помощью какого-либо ГСЧ; пусть всё так и осталось:

$\{1, 1, 2, 3, 5, 8, 3, 1, \dots\}$

С начала  $CF = 1$ .

$a_i = a_{i-1} - a_{i-2} + CF;$

$CF = (a_i \geq 10);$

*Пример:*

randomize(231)

x = rnd();

randomize(231)

y = rnd();

// x != y

x = rnd(-231)

y = rnd(-231)

// x = y

(Серега рассказывает про то как можно пытаться смешивать генерацию случайных чисел)

---

### Лабораторная работа №3.

Суть: изучить 2 стандартных распределения по всем свойствам распределения Ф-ции распределения, плотность распределения, мат. ожидание, дисперсия, ...

Равномерное распределение изучают все.

По списку с периодом 4 изучают

- 1.
2. экспоненциальное
3. нормальное распределение (Гауссовское)
4. k – распределение Эрланга

Построить графики:

1. Теоретического распределения (функция и плотность распределения)
  2. Экспериментального по «своему» закону распределения (ф-ция и плотность)
- 

### [16.10.2006][Лекция 11]

Программа генерации случайных чисел на Фортране для машин ES (~IBM 360)

```
SUBROUTINE RANDUM( IX, IY, RN)           // была придумана для 32 разрядной
машины
    IY = IX * 1220703125
    IF (IY) 3,4,4      // if ( IY < 0) then
3   IY = IY + 2147483647 + 1
4   RN = IY
    RN = RN * 0.4656613E-9
    IX = IY
    RETURN
END
```

```
// обращение к данной процедуре
CALL RANDUM(IX, IY, YFL)
```

IX – число, которое при первом обращении должно содержать нечетное целое число, состоящее менее чем из 9 цифр

IY - полученное случайное число, используемое при последующих обращениях к программе

YFL - полученное равномерно распределенное в интервале [0, 1] случайное число

Для имитации **равномерного распределения в интервале от [a, b]** используется обратное преобразование функции плотности вероятности:

$$\frac{x-a}{b-a} = R$$

$$x = a + (b - a)R$$

где R – равномерно распределенное псевдослучайное число на [0, 1].

В основе построения программы, генерирующей случайные числа с законом распределения отличным от равномерного лежит **метод преобразования последовательности случайных чисел с равномерным законом распределения в последовательность случайных чисел с заданным законом**.

$$F(t) = \int_{-\infty}^t f(x)dx = R \quad (1)$$

Метод основан на утверждении, что случайная величина  $x$ , принимающая значения, равные корню уравнения (1) имеет плотность распределения  $f(x)$ .  $R$  - равномерная случайная величина от 0 до 1.

Значение случайной величины распределенной по показательному закону исходя из (1) может быть вычислено следующим образом:

$$1 - e^{-\lambda x} = R$$

$$x = \left(-\frac{1}{\lambda}\right) \ln(1 - R)$$

### ***Распределение Пуассона.***

Распределение Пуассона относится к числу дискретных, т.е. таких при которых переменная может принимать лишь целочисленные значения, включая норму с мат. ожиданием и дисперсией равной  $\lambda > 0$ .

Для генерации Пуассоновских переменных можно использовать метод точек, в основе которого лежит генерируемое случайное значение  $R_i$ , равномерно распределенное на  $[0, 1]$ , до тех пор, пока не станет справедливым

$$\prod_{i=0}^x R_i \geq e^{-\lambda} > \prod_{i=0}^{x+1} R_i$$

При получении случайной величины, функция распределения которой не позволяет найти решение уравнения (1) в явной форме можно произвести кусочно-линейную аппроксимацию, а затем вычислять приближенное значение корня. Кроме того, при получении случайных величин часто используют те или иные свойства распределения.

### ***Распределение Эрланга.***

Распределение Эрланга характеризуется двумя параметрами:  $\lambda$  и  $k$ . Поэтому при вычислении случайной величины в соответствии с данным законом воспользуемся тем, что поток Эрланга может быть получен прореживанием потока Пуассона  $k$  раз. Поэтому достаточно получить  $k$  значений случайной величины распределенной по показательному закону и усреднить их.

$$x = \frac{1}{k} \left( \sum_{i=1}^k \left( -\frac{1}{\lambda} \ln(1 - R_i) \right) \right) = -\frac{1}{k\lambda} \sum_{i=1}^k \ln(1 - R_i)$$

### ***Нормальное (Гауссово) распределение.***

Нормально распределенная случайная величина может быть получена как сумма большого числа случайных величин распределенных по одному и тому же закону и с одними и теми же параметрами.

Случайная величина  $X$  имеющая нормальное распределение с математическим ожиданием  $M_x$  и среднеквадратичным отклонением  $\sigma_x$  может быть получена по следующей формуле:

$$x = \sigma_x \cdot \sqrt{\frac{12}{N}} \cdot \left( \sum_{i=1}^N R_i - \frac{N}{2} \right) + M_x$$

Для сокращения вычислений по нормальному закону распределения на практике часто принимают  $N = 12$ . Что в дает довольно точные результаты.

### **Процедура генерирования псевдослучайных чисел (равномерный и нормальный законы распределения):**

```

var n, i:integer;
    x,R:double;
    Const m34: double = 28395423107.0;
        m35: double = 34359738368.0;
        m36: double = 68719476736.0;
        m37: double = 137438953472.0;

function Rand(n:integer):double;
var S, W: double;
    i: integer;
begin
    if n = 0 then
    begin
        x := m34; Rand := 0; exit;
    end;
    S := -2.5;
    for i := 1 to 5 do
    begin
        x := 5.0 * x;
        if x > m37 then x := x - m37;
        if x > m36 then x := x - m36;
        if x > m35 then x := x - m35;
        w := x / m35;

        if n = 1 then
        begin
            Rand := w; exit
        end;
        S := S + w;
    end;

    S := S * 1.54919;
    Rand := ( sqr(S) - 3.0 ) * S * 0.01 + S;
end;

begin
    R := Rand(0);
    for i := 1 to 200 do
        writeln( Rand(2):18:10)
end.

```

При  $n = 0$  происходит параметрическая настройка или т.н. «установка».

При  $n = 1$  будем получать равномерно распределенную случайную величину.

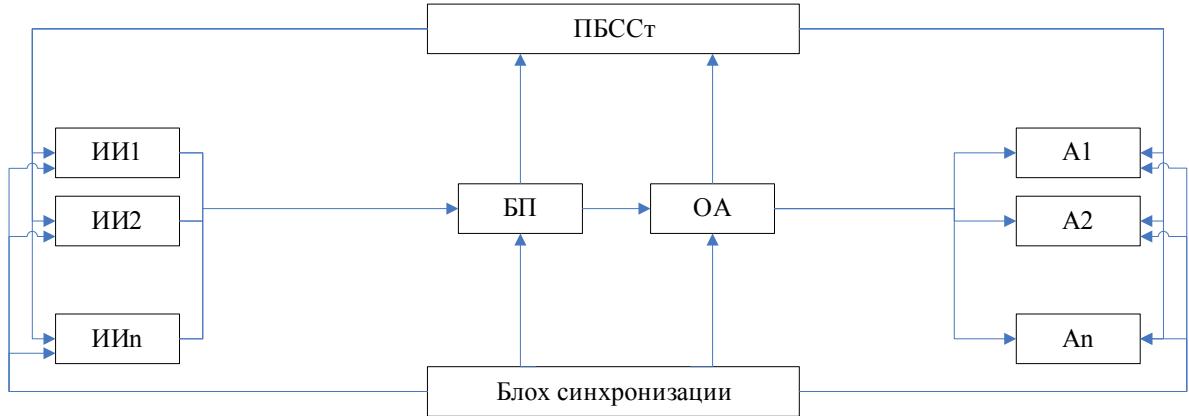
При  $n = 2$  будет гауссово (нормальное) распределение.

[«всем, пожалуйста, поиграться с этим алгоритмом и построить график» (с) by Рудаков]

## Методика построения программной модели ВС.

Для разработки программной модели исходная система должна быть представлена как *стохастическая система массового обслуживания*. Это можно объяснить следующим: информация от внешней среды поступает в случайные моменты времени, длительность обработки различных типов информации может быть в общем случае различна. Т.о. внешняя среда является *генератором сообщений*. А комплекс вычислительных устройств (ВС) – *обслуживающими устройствами*.

### Обобщенная структурная схема ВС.

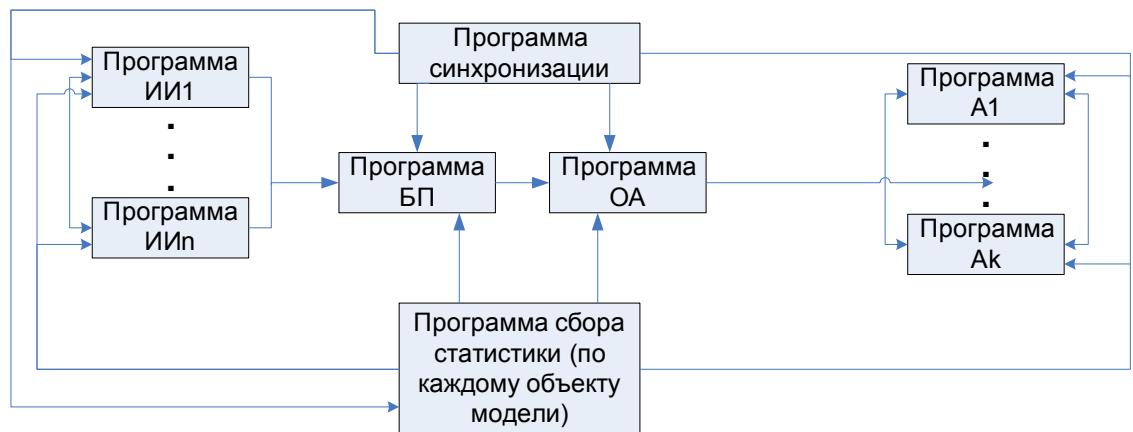


**ИИ** – источники информации – выдают на вход буферной памяти (БП) независимые друг от друга сообщения. Закон появления сообщений – произвольный, но задан на перед.

В **БП** (буферной памяти) сообщения записываются «в навал» и выбираются по одному в обслуживающий аппарат (ОА) по принципу FIFO/LIFO. Длительность обработки одного сообщения в **ОА** в общем случае так же может быть случайной, но закон обработки сообщений должен быть задан. Т.к. быстродействие ОА ограничено то на входе системы в БП возможно сложение данных ожидающих обработки.

**А** – абоненты.

Программная модель из этой системы создается следующим образом:

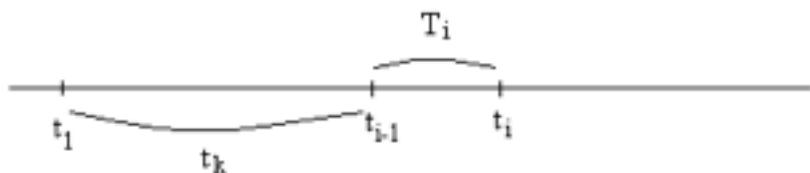


Должна быть обязательно программа сбора статистики (**ПБССт** – программный блок сбора статистики). Причем статистику программа должна собирать по каждому из объектов модели. Так же должна быть программа, которая позволит "оживить" систему

– это **программа синхронизации (блок синхронизации)**, которая покажет когда и в какое время будут активизированы те или иные фрагменты модели.

### Моделирование работы источника информации (ИИ).

Поток сообщений обычно имитируется моментами времени, отображающими появление очередного сообщения в потоке.



$$t_i = \sum_{k=1}^{i-1} T_k + T_i$$

где  $T_i$  – интервал времени между появлением  $i$ -го и  $(i-1)$ -го сообщения.

Программа – имитатор выработки таких интервалов:

- 1) Обратиться к генератору равномерно распределенных случайных величин на  $[a,b]$
- 2)  $T_i$  – по заданному закону
- 3) К текущему времени +  $T_i$

```
// процедура равномерного распределения псевдослучайных чисел на интервале
[a,b]
// U - равном. распр. на [0, 1]
// x = a + (b - a)U
double get_time (int i)
{
    double S = 0;
    srand(seek);
    if ( i > 1 ) S += get_time(i - 1);
    S += a + (b - a)get_u();
    return S;
}
```

или

```
double get_time (int i)
{
    double S = 0;
    srand(seek);
    for (int i = 0; i < .. ; i++)
        S += a + (b - a)rand();
    return S;
}
```

Выражения для вычисления времени с различным распределением:

Вид распределения	Выражение
-------------------	-----------

равномерное на $[a,b]$	
нормальное	, $n \sim= 12$
экспоненциальное	
Эрланга	

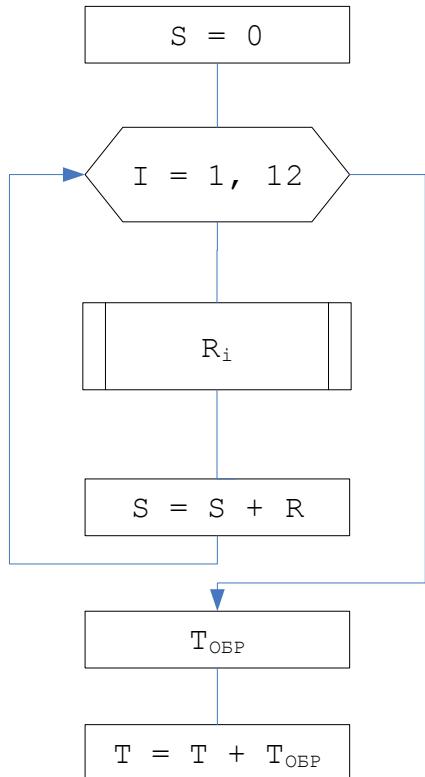
## [23.10.2006][Лекция 12]

### Моделирование работы Обслуживающего Аппарата.

Программа-имитатор работы ОА представляет собой комплекс, вырабатывающий случайные отрезки времени, соответствующие длительностям обслуживания требований. Например, если требования от источника обрабатываются в ОА по нормальному закону с параметрами  $M_x$  и  $\sigma_x$ , то длительность обработки  $i$ -ого требования:

$$T_{обp} = M_x + (\sum_{i=1}^{12} R_i - 6) \cdot \sigma_x$$

### Схема алгоритма имитатора.



$R_i$  – случайное число с равномерным законом распределения

$T_{обp}$  – время обработки очередного сообщения

$T$  – время освобождения ОА

ХМ – Мат ожидание для заданного закона обратки

DX – СКО (средне квадратичное отклонение) для заданного закона обратки

$$T_{OBP} = XM + (S - 6) * DX$$

### Моделирование работы абонентов.

Абонент может рассматриваться как Обслуживающий Аппарат, поток информации, который поступает от процессора.

Для имитации работы абонентов необходимо составить программу выработки длительности обслуживания требования. Кроме того, абонент сам может быть источником заявок на те или иные ресурсы вычислительной системы. Эти заявки могут моделироваться с помощью генератора сообщений, распределенными по заданному закону. Таким образом, абонент либо имитируется как ОА, либо как генератор.

### Моделирование работы буферной памяти.

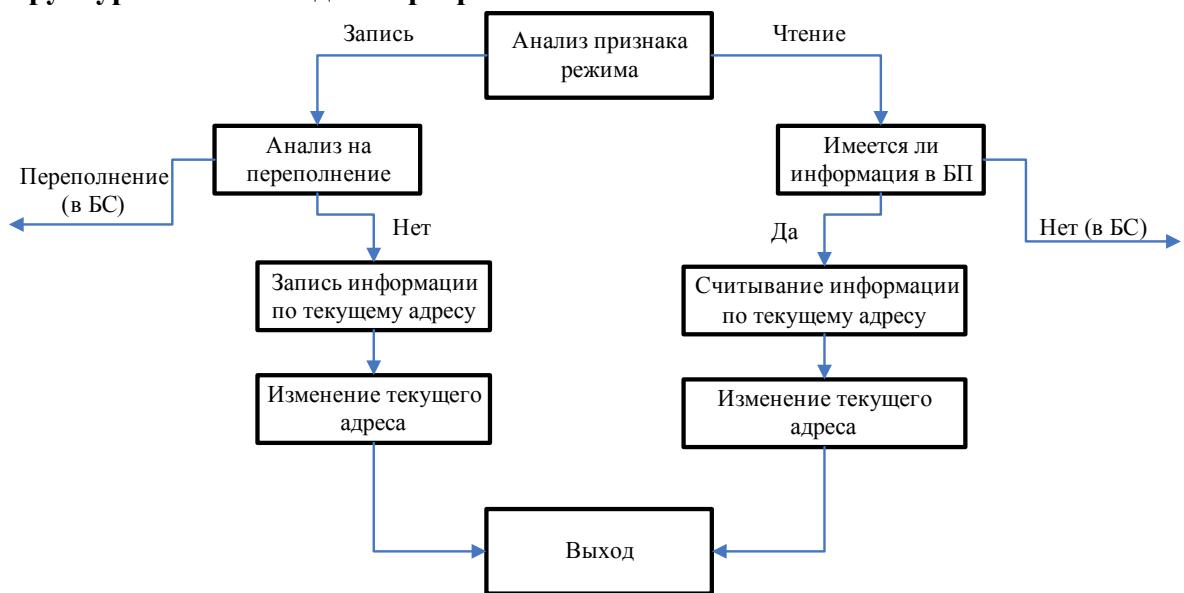
**Память** - относится к электромеханическому устройству, включающему в себя: среду для запоминания, устройство управления, (информация находится по адресу) база + смещение + [индекс].

**Свойства памяти:** предназначена для хранения, чтения и записи информации.

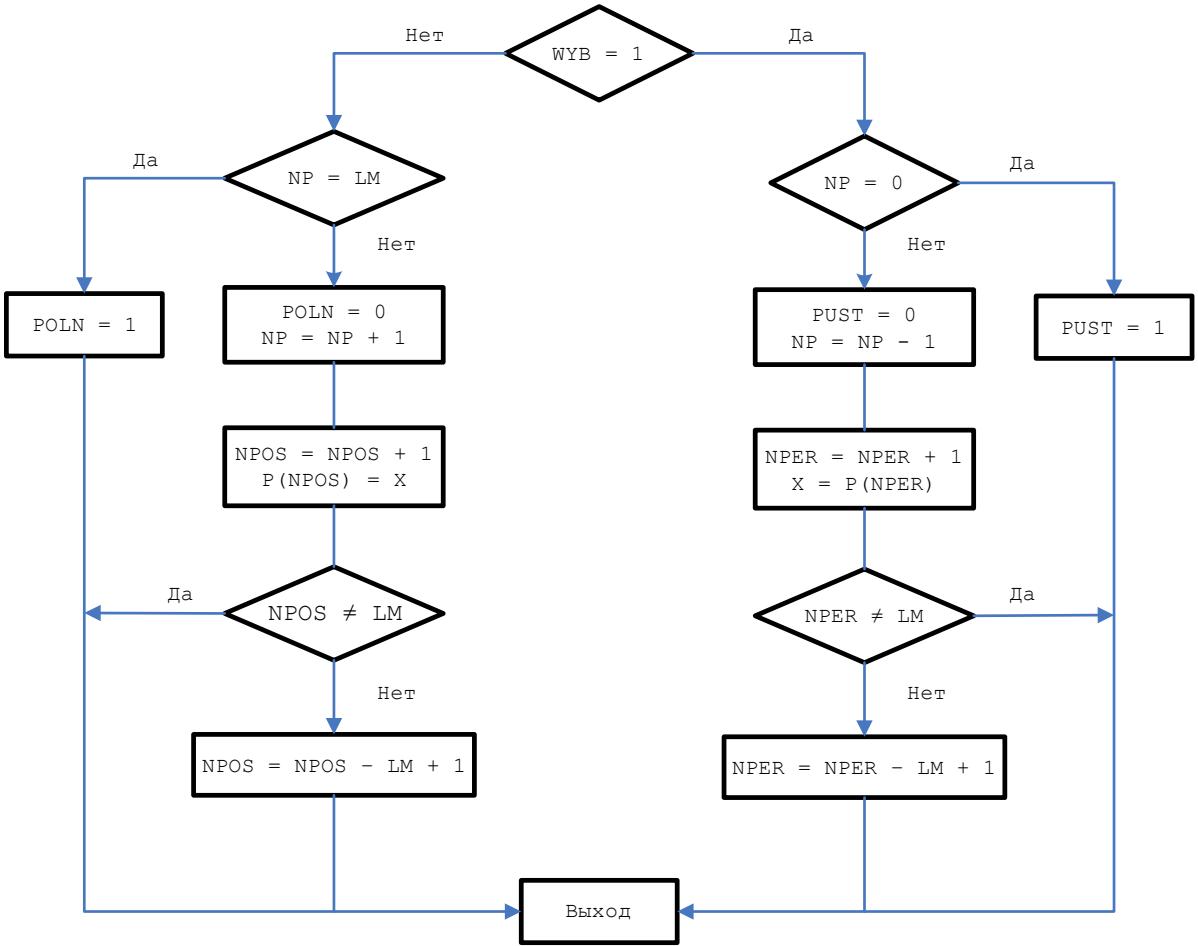
В блок статистики: ошибки записи, ошибки чтения.

Блок буферной памяти должен производить запись и считывание чисел, выдавать сигналы переполнения и отсутствия данных в любой момент времени располагать сведениями о количестве требований (заявок) в блоке. Сама запоминающая среда в простейшем случае имитируется одномерным массивом, размер которого определяет ёмкость памяти. Каждый элемент этого массива может быть либо свободен и в этом случае мы считаем, что он равен 0, либо «занят», в этом случае в качестве эквивалента требования ему присваивается значение времени появления требования.

### Структурная схема модели программной памяти:



Алгоритм реализации работы буферной памяти:



$P$	массив сообщений	$LM$	объем буферной памяти
$WYB$	признак обращения к буф. памяти $= 1$ – режим выборки сообщений $= 0$ – режим записи	$NPOS$	номер последнего сообщения, поступившего в память
$NP$	число сообщений в памяти	$NPER$	номер первого сообщения в памяти
$POLN$	признак переполнения памяти $= 1$ – нет свободных ячеек	$PUST$	признак отсутствия сообщений $= 1$ – в памяти нет сообщений
$NPOS$	$= NPOS + 1$ , если $NPOS < LM$ $= NPOS - LM + 1$ , иначе	$NPER$	$= NPER - 1$ , если $NPER < 1$ $= NPER - LM + 1$ , иначе
$X$	ячейка для сообщения		

### Разработка программы для сбора статистики.

Задача блока статистики заключается в накоплении численных значений необходимых для вычисления статистических оценок, заданных параметров работы моделируемой системы. При моделировании простейшей модели СМО, как правило, оценивают среднее время ожидания в очереди. Для каждого сообщения время ожидания в очереди

равно разности между моментами времени когда оно было выбрано на обработку обслуживающим аппаратом и моментом времени когда оно пришло в систему от источника информации.

Суммируя количество сообщений в блоке памяти через небольшие промежутки времени и разделив полученную сумму на число суммирований, получим среднее значение длины очереди.

Коэффициент загрузки обслуживающего аппарата (ОА) определяется как отношение времени работы ОА, к общему времени моделирования.

Чтобы определить вероятность потери сообщений в системе, нужно разделить кол-во потерянных сообщений на сумму потерянных и обработанных сообщений в системе.

### **Управляющая программа имитационной модели.**

Если программа-имитатор работы источника или буферной памяти обслуживающего аппарата имитируют работу отдельных устройств, то **управляющая программа** имитирует алгоритм взаимодействия отдельных устройств системы.

Управляющая программа реализуется в основном по двум принципам:

1. **Принцип  $\Delta t$**
2. **Событийный принцип**

#### ***Принцип $\Delta t$ .***

Принцип  $\Delta t$  заключается в последовательном анализе состояний всех блоков в момент  $t + \Delta t$  по заданному состоянию блоков в момент  $t$ . При этом новое состояние блоков определяется в соответствии с их алгоритмическим описанием с учетом действующих случайных факторов, задаваемых распределениями вероятности. В результате такого анализа принимается решение о том, какие общесистемные события должны имитироваться программной моделью на данный момент времени.

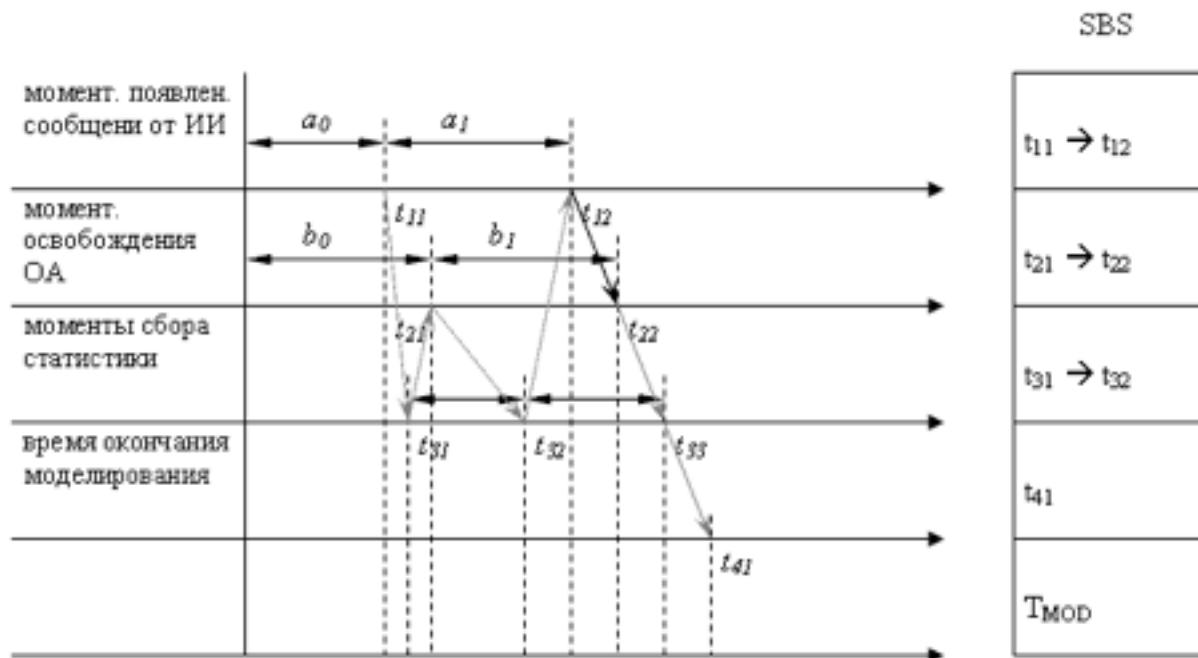
Основной недостаток этого принципа: значительные затраты машинного времени на реализацию моделирования системы. А при недостаточно малом  $\Delta t$  появляется опасность пропуска отдельных событий в системе, что исключает возможность получения адекватных результатов при моделировании.

**Достоинство:** равномерная протяжка времени.

#### ***Событийный принцип.***

Характерное свойство систем обработки информации то, что состояние отдельных устройств изменяются в дискретные моменты времени, совпадающие с моментами времени поступления сообщений в систему, временем поступления окончания задачи, времени поступления аварийных сигналов и т.д. Поэтому моделирование и продвижение времени в системе удобно проводить, используя **событийный принцип**, при котором состояние всех блоков имитационной модели анализируется лишь в момент появления какого-либо события. Момент поступления следующего события определяется минимальным значением из списка будущих событий, представляющего собой совокупность моментов ближайшего изменения состояния каждого из блоков системы.

**Схема событийного принципа:**



Первая ось: момент появления сообщений

Вторая ось: момент освобождения обслуживающего аппарата

Третья ось: момент сбора статистики (здесь абсолютно равные интервалы, мы сами определяем, когда собирать статистику)

Четвертая ось: время окончания моделирования

Пятая ось: текущее время

$t_{11}, t_{12}$  – моменты появления сообщений на выходе генератора (источника информации)

$b_1$  – интервал времени обслуживания первого сообщения

$t_{3n}$  – момент сбора статистики

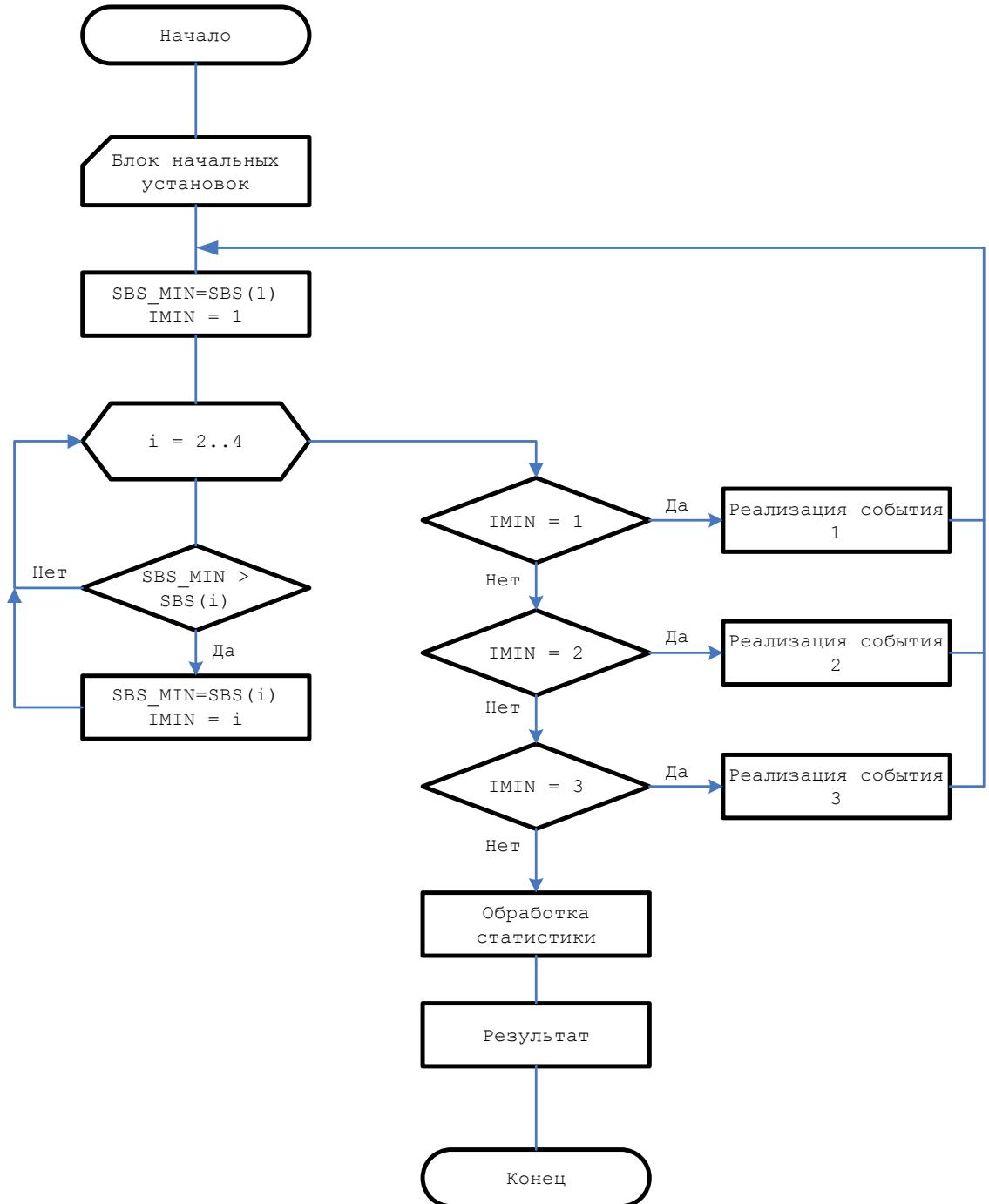
$t_{41}$  – момент окончания моделирования

SBS – список будущих событий.

**Методика реализации событийной модели.**

1. Для всех активных блоков (блоки, порождающие события) заводят свой элемент в одномерном массиве – в списке будущих событий (СБС).
2. В качестве подготовительной операции в СБС заносят время ближайшего события от любого активного блока. Активизируя программный имитатор источника событий вырабатывают псевдослучайную величину  $a_0$ , определяющую момент появления первого сообщения  $t_{11}$  от источника информации и эту величину заносят в СБС. Активизируя программу-имитатор, ОА вырабатывает псевдослучайную величину  $b_0$ , определяющую момент времени  $t_{21}$ , которую также заносят в SBS. В момент времени  $t_{31}$  (момент первого сбора статистики) определяется равным стандартному шагу сбору статистики

$t_{CTAT}$  и заносится так же в СБС. В этот же список заносим время окончания моделирования  $t_{41}$ . На этом подготовительный этап заканчивается и далее протяжка времени осуществляется по следующему алгоритму:



Алгоритм:

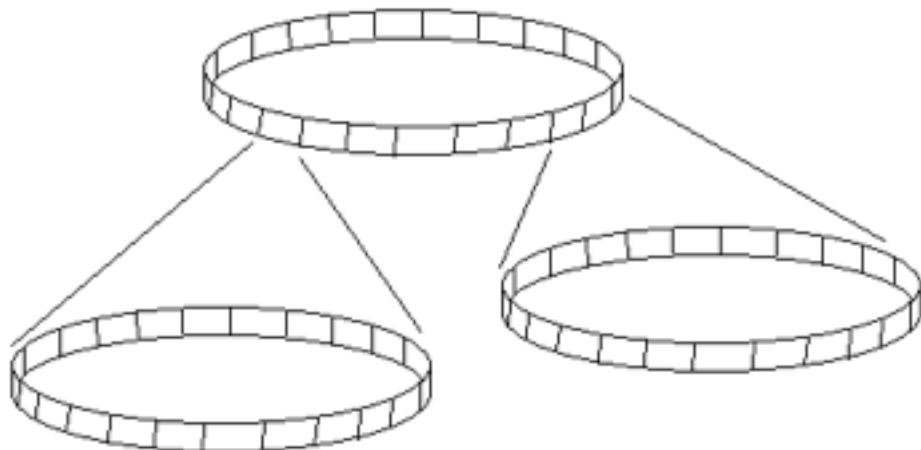
1. В SBS определяется минимальное числовое значение и его номер.
2. Реализуется событие, порождаемое блоком с соответствующим номером, т.е. модельное время =  $t_{II}$ . Далее реализуется событие с номером 1, связанное с появлением нового сообщения в ИИ. Реализация этого события заключается в том, что само сообщение записывается в память, а с помощью имитатора ИИ, вырабатывается момент появления следующего события  $t_{I2}$ . Это время помещается в соответствующую ячейку SBS вместо  $t_{II}$ .

3. Затем вновь организуется поиск минимального элемента в SBS. Для данного примера реализуется событие 3, после чего выражение момента времени  $t_{32}$  – новое время сбора статистики. Так до тех пор, пока минимальное время не станет равным  $t_{41}$ .

[27.10.2006][Лекция 13]

### *Комбинированный метод.*

Два приведенных метода являются универсальными алгоритмами протяжки модального времени. Причем для некоторых предметных областей один принцип может работать быстро и без потерь, а другой будет работать неэффективно. Выбор метода необходимо производить исходя из распределения событий по времени. В реальных системах распределение событий, как правило, *неоднородно*. События, как бы группируются по времени. Образование таких групп связано с наступлением какого-то «значимого» события, которое начинает определенную последовательность действий с соответствующими событиями, имеющими высокую плотность на следующем временном интервале. Такой интервал называется ***пиковым***. А распределение событий ***квази-синхронным***. Примером может являться – *цифровая сеть*, в которой синхронизирующие сигналы переключают большое количество триггеров. Для сложных дискретных систем, в которых присутствуют *квазисинхронное* распределение событий, был разработан алгоритм с название **Delft**. Особенностью данного метода является автоматическая адаптация к распределению событий. Метод реализуется таким образом, что на пиковых интервалах он приближается к *методу  $\Delta t$* , а вне пиковых к *событийному*. В основе лежит использование иерархической структуры циркулярных списков.



Список уровня 1 содержит  $n_1$  элементов и описывает планируемое событие в пиковых интервалах. Число  $n_1$  представляет собой разбиение пикового интервала на более мелкие участки, с каждым из которых связан список событий произошедших за этот интервал. Списки второго уровня и выше являются масштабирующими списками,

количество элементов которого равно константному значению  $n_2$ , которое характеризует коэффициент масштабирования временных интервалов.

Собственно алгоритм протяжки времени заключается в последовательном поиске непустых элементов в самом верхнем циркулярном списке с большим шагом и дальнейшим спуском на нижние уровни (иерархические), вследствие чего уменьшается шаг протяжки модельного времени.

---

### **Лабораторная работа №4.**

Программная имитация i-го прибора.

Генератор, очередь и ОА. Закон генерации заявок выбирается равномерный (параметры настраиваются и варьируются). Закон в ОА из 3 лабораторной работы.

Определить оптимальную длину очереди, т.е. ту длину, при которой ни одно сообщение необработанным не исчезает. Т.е. нет отказа.

---

## **Моделирование систем и языки моделирования.**

Алгоритмические языки при моделировании систем служат вспомогательным аппаратом в разработке машинной реализации и анализа характеристик моделей. *Основная задача* – это выбор языка.

Каждый язык имеет свою систему абстракций, лежащую в основе формализации функционирования сложных систем.

Для программирования модели могут использоваться следующие языки:

1. Универсальные алгоритмические языки высокого уровня.
2. Специализированные языки моделирования: языки, реализующие событийный подход, подход сканирования активностей, языки, реализующие процессно-ориентированный подход.
3. Проблемно-ориентированные языки и системы моделирования.

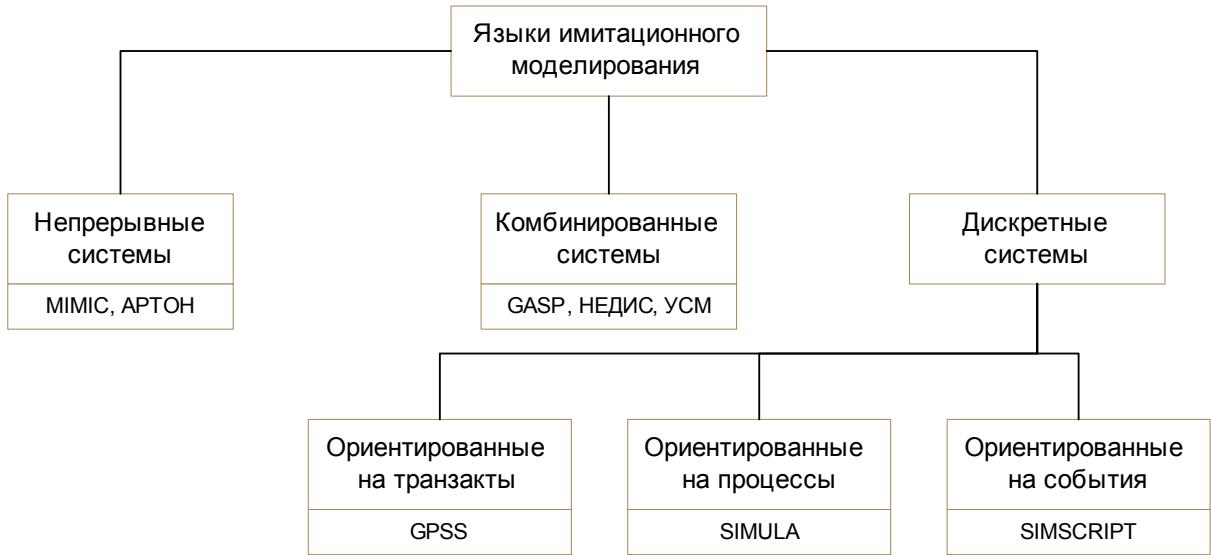
Качество языков моделирования характеризуется:

1. Удобство описания процесса функционирования;
2. Удобство ввода исходных данных, варьирования структуры, алгоритмов работы и параметров модели;
3. Эффективностью анализа и вывода результатов моделирования;
4. Простотой отладки и контроля работы моделирующей программы;
5. Доступностью восприятия и использования языка.

В большинстве своем языки моделирования определяют поведение систем во времени с помощью модифицированного событийного алгоритма. Как правило, он включает в себя список текущих и будущих событий.

## **Классификация языков имитационного моделирования.**

Основа классификации – принцип формирования системного времени.



Непрерывное представление систем сводится к представлению дифференциальных уравнений, с помощью которых устанавливают связь между входной и выходной функциями. Если выходные переменные модели принимают дискретные значения, то уравнения являются разностными.

GASP – комбинированный, в основе лежит язык FORTRAN. Предполагается, что в системе могут наступать события двух типов:

- события, зависящие от состояния
- события, зависящие от времени.

Состояние системы описывается набором переменных, причем некоторые из них меняются непрерывно. При таком подходе пользователь должен составлять процедуры, описывающие условия наступления событий. Законы изменения непрерывных переменных, правила перехода от одного состояния к другому, т.е. реализуется классический принцип ДУ.

Группы языков моделирования, ориентированные на дискретное время, используется при построении именно имитационных моделей, но при этом используются разные способы описания динамического поведения исследуемого объекта.

## **Формальное описание динамики моделируемого объекта.**

Будем считать, что любая работа в системе совершается путем выполнения **активностей**. Т.е. **активность** является наименьшей единицей работы и её рассматривают как единый дискретный шаг. Следовательно, активность является, единственным динамическим объектом, указывающим на совершение единицы работ. **Процесс** – это логически связанный набор активностей.

Пример: активность установки головки жесткого диска, активность передачи информации с жесткого диска.

Активности проявляются в результате свершения событий. **События** – это мгновенное изменение состояния некоторого объекта системы. Рассмотренные объекты (активности, процессы, события) являются конструктивными элементами для динамического описания поведения системы. На их основе строятся языки моделирования системы. В то время, когда динамическое поведение системы формируется в результате выполнения большого числа взаимодействующих процессов, сами эти процессы образуют относительно небольшое число классов. Чтобы описать поведение системы, достаточно описать поведение каждого класса процессов и задать значение атрибутов для конкретных процессов.

### **Задачи построения модели.**

Построение модели состоит из решения двух основных задач:

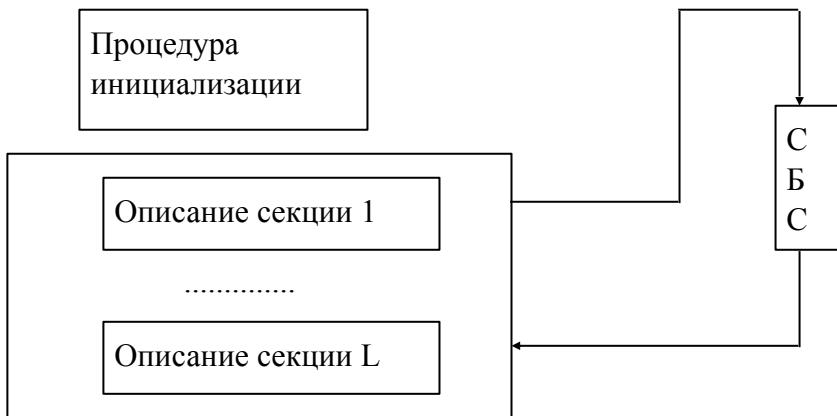
1. Первая задача сводится к тому, чтобы описать правила, описывающие виды процессов, происходящих в системе.
2. Вторая задача заключается в том, чтобы описать правила задания атрибутов или задать правила генерации этих значений. При этом система определяется на конкретном уровне детализации в терминах множества описаний процессов, каждый из которых в свою очередь включает множество правил и условий возбуждений активности. Такое описание системы может быть детализировано на более подробном или более иерархическом уровне представления с помощью декомпозиции процессов (в идеальном случае в активности). Это обеспечивает многоуровневое исследование системы.

Т.к. система в общем случае служит для описания временнОго поведения, то язык моделирования дискретных систем должен обладать средствами, отображающими время. В реальной системе совместно выполняются несколько активностей, принадлежащим как связанным, так и не связанным процессам. Имитация их действий должна быть строго последовательной. Таким образом, модель системы можно рассматривать как модель описаний, активностей, событий или процессов. Отсюда и деление языков моделирования.

### **Языки, ориентированные на события.**

Моделирующая программа организованна в совокупности в виде секций событий (процедуры отражающие события). Процедура состоит из набора операций, которые выполняются после выполнения какой-либо активности. Синхронизация происходит с помощью списка будущих событий.

Структуру рассмотрим на примере языка SIMSCRIPT.



### ***Языки, ориентированные на процессы.***

Моделирующая программа организуется в виде набора описаний процесса. Каждый из которых описывает один класс. Описание процесса функционирования устанавливает атрибуты и активности всех процессов. Синхронизация операций во времени реализуются так же с помощью списка будущих событий, который содержит точку возобновления конкретного процесса (точка прерывания).

На примере языка SIMULA:



### ***Результаты экспертных оценок сравнения различных языков при моделировании большого класса систем.***

Критерии:

1. Возможность языка. Выше всех находится SIMULA > SIMSCRIPT > GPSS -> C -> PASCAL
2. Простота применения: GPSS -> SIMSCRIPT -> SIMULA -> C -> PASCAL
3. Предпочтение пользователей: GPSS -> SIMSCRIPT -> SIMULA -> PASCAL -> C

GPSS (Лабу которую уже выдали, реализована на этом языке):

```

GEN 3,1
QUEUE 1
SEIZE 1
DEPART 1
ADVANCE 3, FN$XPDIS
RELEASE 1
START 100

// говорит здесь и ошибка есть

```

### [30.10.2006][Лекция 14]

При использовании универсальных алгоритмических языков программист имеет практически неограниченные возможности по созданию имитационной модели, наилучшим образом используя ресурсы системы, особенности ОС и т.д. Но в то же самое время требует больших трудозатрат, работы программистов высокой квалификации, взаимодействие специалистов разных профилей (программисты, эксперты предметной области и т.д.). В результате модель получается узко направленной и, как правило, использование этой модели в других разработках невозможно.

При использовании языков имитационного программирования снижается гибкость и универсальность. Модель создается в несколько раз быстрее и не требует присутствие системы от программистов. Модели обладают свойством концептуальной выразительности. Используются специальные термины языка в области, исследуемой задачи.

**Алгоритм сканирования активности** включает в себя цикл по всем действиям активностей, дальше срабатывает проверка выполнения действия. И если условие выполняется, то выполняется действие, изменяющее состояние модели. Иначе цикл по всем действиям заканчивается.

### *Сравнение универсальных и специализированных языков программирования при моделировании:*

Преимущества	Недостатки
Универсальные	
Минимум ограничений на выходной формат	Значительное время, затрачиваемое на программирование
Широкое распространение	Значительное время, затрачиваемое на отладку
Специализированные	
Меньше затраты времени на программирование	Необходимость точно придерживаться ограничений на форматы данных

Более эффективные методы выявления ошибок	Меньшая гибкость модели
Краткость, точность понятий, характеризующих имитируемые конструкции	
Возможность заранее строить стандартные блоки, которые могут использоваться в любой имитационной модели	
Автоматическое формирование определенных типов данных, необходимых именно в процессе имитационного моделирования	
Удобство накопления и представления выходной информации	
Эффективное использование ресурсов	

### ***Основные концепции языка РДО (Ресурсы, действия, операции).***

Язык придуман в МГТУ.

Причина создания РДО:

- Универсальность имитационного моделирования относительно класса исследуемых систем и процессов.
- Легкости модификации моделей
- Моделирование сложных систем управления совместно с управляемым объектом (включая использования имитационного моделирования в режиме реального времени)

Язык РДО является реализацией т.н. интеллектуального подхода к имитационному моделированию. При этом подходе стараемся отойти от жесткого алгоритмического подхода в процессе принятия решения при моделировании для удобства автоматизации той части процесса, где используются знания человека, и сделать процесс моделирования максимально гибким по способам представления информации об объекте. В основе языка РДО лежит продукционная система, которая состоит из трех элементов: классов и отношений, правил, управляющей структуры.

**Классы и отношения** трактуются как база данных, содержащая декларативные знания. Процедуры представляют собой набор модифицированных продукционных правил: ЕСЛИ – ТО ДЕЙСТВИЕ.

**Управляющая структура** – это интерпретатор правил, управляющий выборкой правил.  
**Условие** – это проверка состояния базы данных.

**Действие** – изменяет некоторым образом базу данных.

### **Достоинства** системы основанной на продукциях:

- Простота создания и понимания, отдельных правил
- Легкость модификации

### **Недостатки:**

- Неясность взаимных отношений правил
- Сложность оценки целостного образа знаний
- Низкая эффективность обработки

Для имитационного моделирования **основным недостатком системы** продукции является отсутствие времени, поэтому такие производственные правила применимы для анализа статических объектов, т.е. правила описывают причинно следственные связи и не описывают динамику процесса. Поэтому при интеллектуальном моделировании, т.е. для моделирования динамики системы, на основе знаний о протекающих процессах в РДО используют модифицированное производственное правило.

Как оно выглядит:

*ЕСЛИ <УСЛОВИЕ>, ТО <СОБЫТИЕ 1> ... ЖДАТЬ (отведенный интервал) ... ТО <СОБЫТИЕ 2>*

СОБЫТИЕ 2 наступает через некоторый интервал времени.

Если в течение указанного времени происходит нерегулярное событие, затрагивающее релевантное данному событию ресурсы, то событие 2 может не наступить или наступить в другой момент времени.

В РДО сложная дискретная система представляется в виде множества взаимодействующих между собой *ресурсов*.

*Ресурс* – это элемент сложной системы, внутренней структурой которого можно пренебречь, в то время как наличие и свойства его, как целого, важны и существенны для описания.

Каждый ресурс модели должен быть описан и должен иметь свое уникальное имя.

Ресурсы могут быть двух типов:

1. *Постоянные*. Они всегда присутствуют в системе.
2. *Временные*. Поступают в систему и покидают её в процессе функционирования. Причем они могут быть и результатом работы, т.е. здесь наличие обратной связи.

Все ресурсы системы образуют некоторое множество:  $R(t) = \{r_i: i = 1, \dots, N(t)\}$

Где  $r_i$  –  $i$ -ый ресурс сложной дискретной системы

$N(t)$  – число ресурсов в текущий момент времени.

Каждый ресурс описывается множеством параметров, которые могут быть следующих типов:

1. *Описательные*, представляющие факты внутренне присущие каждому ресурсу.
2. *Указывающие*, используемые для дачи имени или обозначения ресурса, проще говоря, идентификаторы.
3. *Вспомогательные*, используемые для связи различных ресурсов, накопления статистики, графического вывода при имитации и т.д.

Состояние ресурса в момента времени  $t$ :  $C_i(t) = \{C_{ij}(t): j = 1 \dots M_i\}$

Где  $C_{ij}$  – значение  $j$ -ого параметра  $i$ -ого ресурса

$M_i$  – число параметров  $i$ -ого ресурса

Состояние всей системы является совокупностью состояний всех ресурсов.

Ресурсы, принадлежащие к одному типу, наследуют общие свойства этого типа. Отношения наследования может использоваться как для отображения общности ресурсов, так и для идентификации ассоциативных связей.

Ресурсы взаимодействуют друг с другом в соответствии с определенным алгоритмом. Каждое действие связано с изменением состояния системы, которое связано с конкретным событием. *Все события должны быть определены и зафиксированы в модели.* Они могут быть *внешними и внутренними* по отношению к системе.

В РДО событие происходит в счетные моменты времени, которые фиксируются в модели с помощью независимой переменной. Эта переменная изменяется дискретно и служит базой для определения различий в наблюдении одного и того же свойства. Все события делятся на *регулярные и нерегулярные*.

**Регулярные** – это события, вызываемые штатным функционированием ресурсов. Они выражают логику взаимодействия ресурсов между собой.

**Нерегулярные события** происходят либо при нештатной работе (поломка или отказ) ресурса, либо из-за внешних по отношению к системе причин, т.е. в систему пришел новый временный ресурс.

В отличие от регулярных событий, нерегулярные носят стихийский характер.

Ресурсы в процессе функционирования выполняют определенные действия. С каждым действием связано два события:

1. Время начала
2. Время окончания

**Действие** представляет собой целенаправленное мероприятие, выполняемое под управлением некоторой подсистемы и направленное на достижение определенной цели. Поэтому действие планируется и может находиться в следующих состояниях:

- Запланировано
- Начато
- Окончено
- Прервано по какой-либо причине

Действие с нулевой длительностью представляет собой *событие*.

Процесс функционирования сложной дискретной системы строится как времененная последовательность действий и нерегулярных событий. Действие может начинаться только в том случае, если значение параметров его релевантных ресурсов удовлетворяет необходимому условию.

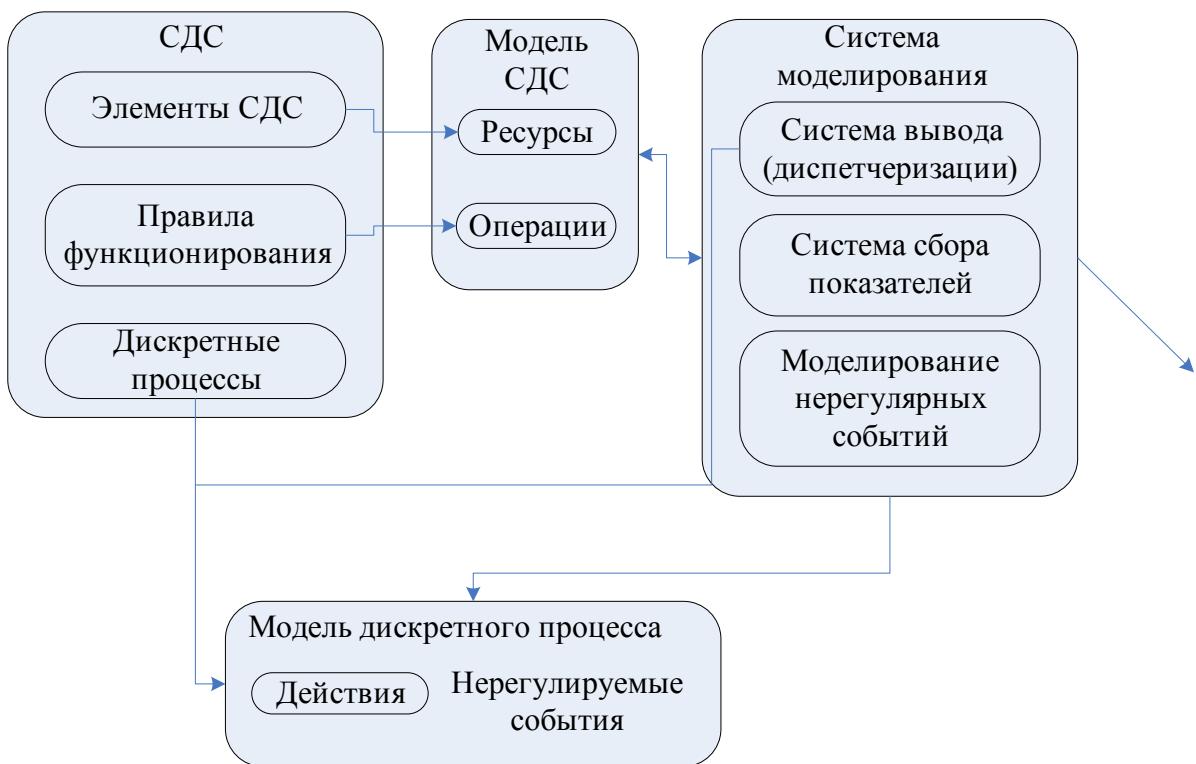
В языке используется понятие **виртуального действия**, которое не привязано ни к началу, ни к окончанию, а привязано только к необходимому условию начала действия.

Множество виртуальных действий может быть разбито на небольшое число подмножеств, имеющих одинаковую природу, т.е. одинаковую логику взаимодействия ресурсов и различаются лишь конкретно участвующими в них ресурсами. Для формального описания логики однотипных виртуальных действий используются понятия *операций*.

**Операция** – это фактически процедура, у которой формальные параметры условия выполнения и алгоритмы, а в качестве фактических параметров выступают подмножества ресурсов. При задании фактических ресурсов элементами соответствующих подмножеств однотипных ресурсов из операции получается виртуальное действие. Т.е. операция отражает логику взаимодействия ресурсов. Всякий раз, когда состояние соответствует условию срабатывания виртуального действия, происходит действие описывающее соответствующей операции с различными временами начала и окончания.

Следовательно, *операция* описывает, как происходит действие или виртуальное действие и с какими множествами релевантных ресурсов. Т.е. что может произойти в сложной системе при определенных условиях. А *действие* – что произошло, происходит, произойдет и в какое время.

### Представление сложной дискретной системы в РДО методе.



СДС – сложная дискретная система

Основные моменты этого метода:

Все элементы сложной дискретной системы представлены как ресурсы.

Основные положения РДО-метода формируются следующим образом:

1. Каждый ресурс определенного типа описывается одними и теми же параметрами.

2. Состояние ресурса определяется вектором значений всех его параметров. Состояние системы в целом – совокупностью всех его параметров.
3. Процесс протекающий в сложной системе описывается как последовательность целенаправленных действий и нерегулярных событий, изменяющих состояние ресурсов. Действия ограничены во времени событиями начала и конца.
4. Нерегулярные события описывают изменение состояния сложной системы не предсказуемые в рамках производственной модели системы. Моменты наступления нерегулярных событий случайны.
5. Действия описываются операциями, которые представляют собой модифицированные правила, учитывающие временные параметры. Операция описывает предусловия, которым может удовлетворять состояние, участвующих в операциях ресурсов. И правила изменения состояния ресурсов в начале и конце соответствующего действия.
6. Множество ресурсов и множество операций образуют модель сложной дискретной системы.

---

### **Лабораторная работа №5.**

Создать концептуальную модель функционирования метро «Бауманская» в час пик.  
Структура включает в себя:

Равномерное прибытие поездов, как в противофазе, так и вместе.

3 эскалатора (Супербабушка может включить два верх, один вниз и наоборот)

Выходные двери от 4 до 1.

Вся статистика: сколько времени стоишь в очереди и т.п.

Определить, куда лучше втискиваться в очередь, т.е. где лучше всего находиться...

---

### **[10.11.2006][Лекция 15]**

Модель сложной дискретной системы в РДО представляет собой динамическую производственную систему. *Базой данных* этой производственной системы является *множество ресурсов, базой знаний – множество операций*. Параметрическая настройка в конкретной системе заключается в формализованном описании ресурсов и операций на внутреннем языке и введении в соответствующую базу знаний и базу данных.

Структура РДО имитатора:



ABC – аппарат введения событий

Основными элементами РДО имитатора является: **динамическая производственная система и аппарат событий**. Действия имитируются самой системой вывода, а нерегулярные события имитируются специальным блоком.

При имитации состояний системы изменяются в соответствии с описанием нерегулируемого события, либо действия, которое началось и завершилось. После любого изменения состояния, т.е. при каждом событии вызывается система вывода. Она просматривает в базе знаний все операции и проверяет у предусловия, могут ли они начаться. При нахождении таких операций инициируются события начала соответствующих действий. Продукционная система (БД, БЗ и Система Вывода, т.е. принятия решений), система имитаций нерегулярных событий и аппарат ведения событий совместно осуществляют процесс построения модели. На основании результатов анализа имитации, вычисляются различные показатели функционирования модели. Система трассировки вводит подробную информацию о событиях в специальный файл, по содержимому которого принимается решение о проведении математического эксперимента.

Система анимации позволяет отображать на экране поведение системы во время моделирования.

РДО имитатор создавался как средство создания имитационных моделей систем планирования, игр и тренажеров. Кроме того, на языке РДО могут быть реализованы экспертные системы, а так же гибридные системы включающие в себя экспертную составляющую, имитирующую модель и алгоритмы оптимизации.

В языке используются следующие понятия:

1. **Модель** – совокупность объектов РДО языка, определяющих реальный объект, собираемый в процессе имитации показателей, кадры анимации, различные графические элементы, результаты трассировки.
2. **Прогон** – единая неделимая точка имитационного эксперимента. Характеризуется совокупностью объектов представляющих собой исходные данные и результаты.

3. **Проект** – один или более прогонов, объединенных какой-либо общей целью.
4. **Объект** – совокупность информации пред назначенной для определенных целей и имеющей смысл для имитационной программы. Состав объектов обусловлен РДО методом, определяющим парадигму представления сложной дискретной системы на языке РДО. Описание объекта в зависимости от типа разделяются по разным модулям.

**Объекты исходных данных:**

- типы ресурсов,
  - образцы операций,
  - операции,
  - точки принятия решений,
  - константы,
  - функции,
  - последовательности,
  - кадры анимации,
  - требуемая статистика,
  - результаты трассировки.
5. **Комментарий прогона** – произвольный текст, предназначенный для хранения сопроводительной информации прогона.
  6. **Комментарий проекта** – текстовая информация, характеризующая проект.

Подготовка исходных данных моделирования и анализ результатов производится с помощью интегрированной среды РДО, которая имеет следующие возможности:

- a. Реализует графический объектно-ориентированный интерфейс
- b. Позволяет структурировать информацию по имитационным исследованиям
- c. Автоматически поддерживает целостность информации
- d. Режимы работы эксперта и пользователя

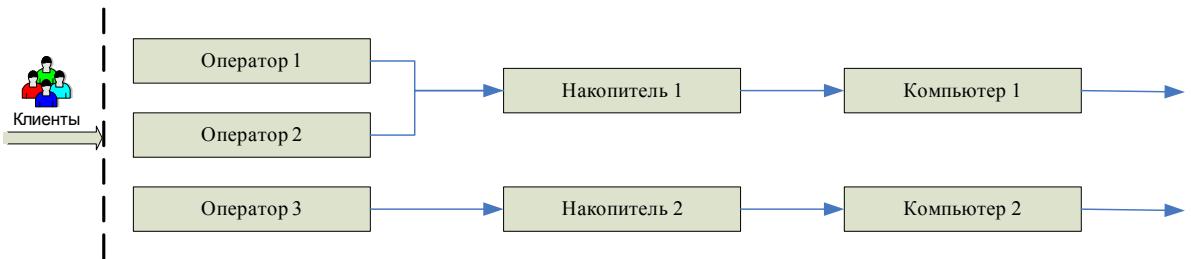
В рамках одной среды объединены как средства описания исходных данных, так и средства моделирования и анализа результатов. Предусмотрена возможность комментирования отдельных имитационных исследований и прогонов.

---

**Лабораторная работа №6.**

В информационный центр приходят клиенты через интервал времени  $10 \pm 2$  минуты. Если все три имеющихся оператора заняты, клиенту отказывают в обслуживании. Операторы имеют разную производительность и могут обеспечивать обслуживание среднего запроса пользователя за  $20 \pm 5$ ;  $40 \pm 10$ ;  $40 \pm 20$ . Клиенты стремятся занять свободного оператора с максимальной производительностью. Полученные запросы сдаются в накопитель. Откуда выбираются на обработку. На первый компьютер запросы от 1 и 2-ого операторов, на второй – запросы от 3-его. Время обработки запросов первым и 2-м компьютером равны соответственно 15 и 30 мин. Промоделировать процесс обработки 300 запросов.

Необходимо для этого создать концептуальную модель в терминах СМО, определить Эндогенные и Экзогенные переменные и уравнения модели. За единицу системного времени выбрать 0,01 минуты.



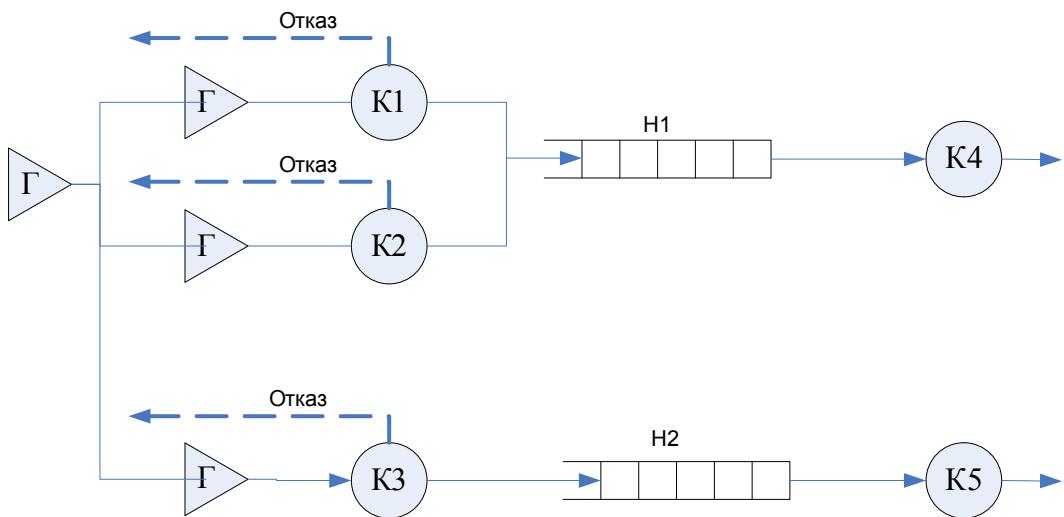
В процессе взаимодействия клиентов с информационным центром возможно:

- 1) Режим нормального обслуживания, т.е. клиент выбирает одного из свободных операторов, отдавая предпочтение тому у которого меньше номер.
- 2) Режим отказа в обслуживании клиента, когда все операторы заняты

Переменные и уравнения имитационной модели.

Эндогенные переменные: время обработки задания  $i$ -ым оператором, время решения этого задания  $j$ -ым компьютером.

Экзогенные переменные: число обслуженных клиентов и число клиентов получивших отказ.



$$P_{omk} = \frac{C_{omk}}{C_{omk} + C_{обсл}}$$


---

### *AnyLogic™*

AnyLogic™ имитационного моделирования которой позволяет моделировать при помощи визуальных компонент как стандартных, так и разработанных пользователем. Программировать иерархические структуры на разных уровнях абстракции.

Создавать интерактивные 2 и 3D анимации визуально отображающие результаты работы модели в реальном времени.

Увеличить жизненный цикл модели.

Использовать средства анализа и оптимизации непосредственно из среды разработки модели.

Достаточно просто интегрировать модель открытой архитектуры с офисными и корпоративными программными продуктами (Электронные таблицы, БД и БЗ, CRM и т.д.)

### **Открытая архитектура.**

Модель может динамически читать и сохранять данные в электронных таблицах, БД, системах планирования корпоративных ресурсов, управление взаимоотношением с клиентами.

Моделирование:

- отображение результатов
- библиотеки численных методов
- базы данных
- анализ параметров
- оптимизация
- анализ результатов

**AnyLogic** позволяет строить как стохастические так и детерминированные модели. Поддерживает 35 стандартных распределений, можно создавать и свои.

С помощью СтатФит можно построить аналитическое распределение.

В систему входят средства сбора и анализа статистики в работающей модели. С моделью могут быть проведены различные эксперименты, в том числе и метод Монте-Карло.

Анализ чувствительности, анализ рисков, оптимизация, а так же эксперименты по сценарию пользователя.

Сочетания эвристики, нейронные сети и математическую оптимизацию, встроенный в систему оптимизатор позволяет находить значения дискретных и непрерывных параметров модели, соответствующие максимуму и минимуму целевой функции. В условиях неопределенности и при наличии ограничений.

Модуль настраивается и запускается прямо из среды разработки моделей. Есть возможность применения пользовательских методов оптимизации, которые вносятся в модель через Java API.

С помощью технологии визуализации модели создается интерактивная анимация связывая графические объекты. Как и модель, анимация имеет иерархическую структуру, которая может динамически изменяться.

### **Уровни моделирования.**

Разработчиками заявлено применение системы от микромодели физического уровня, где важную роль играет такие параметры как размеры, расстояния, скорости, времена, до макро моделей стратегического уровня, на которых рассматривается глобальная динамика обратных связей. Оцениваются стратегические решения.

В системе выделяют 3 основных уровня:

- 1. Стратегический**
- 2. Операционный**

### 3. Физический

Данная система позволяет:

1. Предсказать эффективность действий по продвижению продукта в условиях конкретного рынка.
2. Выбрать оптимальную стратегию компании в конкурентной борьбе.
3. Исследовать колебания спроса или внутренних задержек на функционирование цепочки поставок и определить оптимальный «портфель» заказа или проекта с учетом их взаимосвязи.
4. Сравнить сценарии развития урбанизированной территории и предсказать экологические последствия.

Система поддерживает все элементы динамики: *накопители, потоки, обратные связи, задержки, вспомогательные переменные, табличные функции, решение различных уравнений*. Протяжка модельного времени определяется по дискретно событийному уровню при помощи диаграмм состояний и диаграмм процессов. Связывая её с системно-динамической частью.

---

Изучить самостоятельно! - Сети массового обслуживания. Открытые, замкнутые, комбинированные.

---

### [13.11.2006][Лекция 16]

---

В общем, отчеты по лабораторным работам писать в электронном виде.

---

### Язык *General Purpose System Simulation (GPSS)*

Язык GPSS – общечелевая система моделирования. Как и любой язык программирования, она содержит словари и грамматику, с помощью которых разрабатываются имитационные модели сложных дискретных систем версий 1, 2, V, GPSS /PC, GPSS WORLD.

Позволяет:

- Многозадачность
- Использование виртуальной памяти
- Интерактивность
- Графический интерфейс пользователя.
- Визуализация процесса моделирования.

Основное применение (то, что заявлено в качестве рекламы):

- Транспорт (самая известная модель: эксплуатация парка самолетов в авиационно-технической транспортной компании)
- Сетевые технологии. Исследование распределенной региональной сети передачи данных.

- Промышленность. Имитация автоматизированного металлургического производства.
- Финансовые и медицинские аспекты.

Система GPSS построена в предположении, что моделью сложной дискретной системы является описание её элементов и логических правил их взаимодействия в процессе функционирования моделируемой системы. Для определенного класса моделируемых систем можно выделить конечный набор абстрактных элементов, называемых «**объекты**», причем набор логических правил так же ограничен и может быть описан небольшим числом стандартных операций. Объекты языка подразделяются на 7 категорий и 14 типов.

Категория	Типы
Динамическая	Транзакция
Операционная	Блоки
Аппаратная	Устройства памяти, ключи
Вычислительная	Переменные, арифметические, логические, функции
Статистическая	Очереди, таблицы
Запоминающая	Ячейки, матрицы ячеек
Группирующие	Списки, группы

Основой системы являются программы, описывающие функционирование выделенного конечного набора объектов и специальная программа-диспетчер, которая выполняет следующие функции:

- Обеспечивает, заданные программистом, маршруты
- Продвижение динамических объектов, называемых *транзактами* (*заявками* или *сообщениями*).
- Планирование событий происходящих в модели, путем регистрации времени наступления события и реализацию этих событий в нарастающей временной последовательности
- Регистрация статистической информации.

**Динамическими объектами** являются транзакты, которые представляют собой единицы исследуемых потоков и производят ряд определенных действий, продвигаясь по фиксированной структуре, представляющей собой совокупность объектов других категорий.

**Операционный объект.** Блоки задают логику функционирования системы и определяют маршрут движения транзактов между **объектами аппаратной категории**. Это абстрактные элементы, на которые может быть декомпозирована структура реальной системы. Воздействуя на эти объекты, транзакты могут изменять их состояния и оказывать влияние на движение других объектов.

**Вычислительный объект.** Служит для описания таких операций в процессе моделирования, когда связи между элементами моделируемой системы наиболее просто выражаются в виде математических соотношений.

К **статистическим объектам** относятся очереди и таблицы, служащие для оценок влияющих характеристик.

В процессе моделирования системы одни объекты взаимодействуют с другими, в результате чего происходит изменение атрибутов и преобразование их значений. Такие преобразования называются «**события**». Транзакты моделируют прохождение по системе соответствующих единиц исследуемого потока. Такое движение может быть разбито на цепь элементарных событий, происходящих в определенные моменты времени. Основной задачей симулятора является выявления моментов наступления этих событий, расположенных в их правильной временной последовательности и выполнения определенных действий при наступлении определенных событий. Все отрезки времени описываются целыми числами. Поэтому перед составлением модели необходимо провести временное масштабирование для всех характеристик модели связанных со временем. Каждому объекту соответствуют атрибуты, определяющие его состояние в данный момент времени. Значения атрибутов могут быть **арифметическими** или **логическими**. Большая часть атрибутов недоступна для пользователей. Атрибуты, которые доступны (или атрибуты, которые нужно адресовать) называются **стандартными числовыми** или **стандартными логическими атрибутами** (СЧА или СЛА).

Практически все изменения состояний происходят в результате ввода транзакции в блок и выполнения блоком своих функций.

С блоками непосредственно связаны операционные блоки, изменяющие процесс моделирования, блоки вывода и печати промежуточных результатов, команды, управляющие процессом моделирования и редактированием результатов.

**Транзакты** представляют собой описание динамических процессов в реальных системах. Они могут описывать как реальные физические объекты, так и нефизические (например, канальная программа). Транзакты можно генерировать и уничтожать в процессе моделирования. Основным атрибутом любого транзакта является число параметров. Изменяется это число параметров может от 0 до 1020. Параметры обозначаются как  $P_x$ : номер параметра x + тип параметра. Может быть:

- слово – W
- полуслово – H
- байт – B
- плавающая точка – L

Важными атрибутами любого транзакта является **уровень приоритета** PR. Изменяются от 0 до 100000. Когда два транзакта соперничают за что-то, первым обрабатывается тот, у которого приоритет выше. Если приоритеты одинаковы, то сначала обрабатывается тот, у которого время ожидания обработки больше.

В одном задании может выполняться как один, так и несколько прогонов модели. При этом текущим значением абсолютного времени  $A_1$  модели будет называться **суммарное время по всем реализованным прогонам**, а текущим значением относительного

времени модели –  $C_1$  – *системное время в пределах одного прогона*. Время в течение которого транзакт обрабатывается в процессе моделирования обозначается  $M_1$  и называется *транзактным временем*. Оно отсчитывается:

1. С момента относительного времени
2. С момента прохода транзакта через специальный блок MAP, до текущего момента относительного времени

Параметрическое транзактное время определяется вычитанием из текущего относительного момента времени значения n-ого параметра транзакта с типом X.

### **Классификация блоков GPSS.**

У каждого блока имеется два стандартных числовых атрибута:

- $W_n$  – *счетчик входов в блок* или *ожидающий счетчик*, который содержит в себе номер текущего транзакта, находящегося в блоке
- $N_n$  – *общий счетчик транзактов*, поступивших в блок с начального момента моделирования или с момента обнуления.

Оба счетчика меняют свое содержимое автоматически.

1. **Блоки, осуществляющие модификацию атрибутов транзактов.**  
Временная задержка ADVANCE  
Генерация и уничтожение транзактов GENERATE TERMINATE SPLIT ASSEMBLE  
Синхронизация движения нескольких транзактов MATCH GATHER  
Изменение параметров транзакции ASSIGN INDEX MARK  
Изменение приоритетов PRIORITY
2. **Блоки, изменяющие последовательность передвижения транзактов, т.е. блоки передачи управления.**  
TRANSFER, LOOP, TEST, GATE
3. **Блоки, связывающие с группирующей категорией.**  
JOIN REMOVE EXEMINE SCAN ALTER
4. **Блоки, сохраняющие значения для дальнейшего использования.**  
SAVEVALUE  
ASAVEVALUE
5. **Блоки, организующие использование объектов аппаратной категории.**  
Устройства:  
SEIZE RELEASE (парные команды)  
PREEMP RETURN – тоже самое, но с приоритетной обработкой  
FAVAIL FUNAVAIL – доступность устройства  
Памяти:  
ENTER LEAVE  
SAVAIL SUNAVAIL  
Ключи:  
LOGIC
6. **Блоки, обеспечивающие получение статистической информации.**  
QUEUE DEPART

TABULATE TABLE – статистические таблицы

7. Специальные блоки.

HELP TRACE UNTRACE PRINT и еще куча

8. Блоки для организации цепей.

LINK UNLINK

9. Вспомогательные блоки.

REPORT – создание стандартного отчета

LOAD SAVE и т.д.

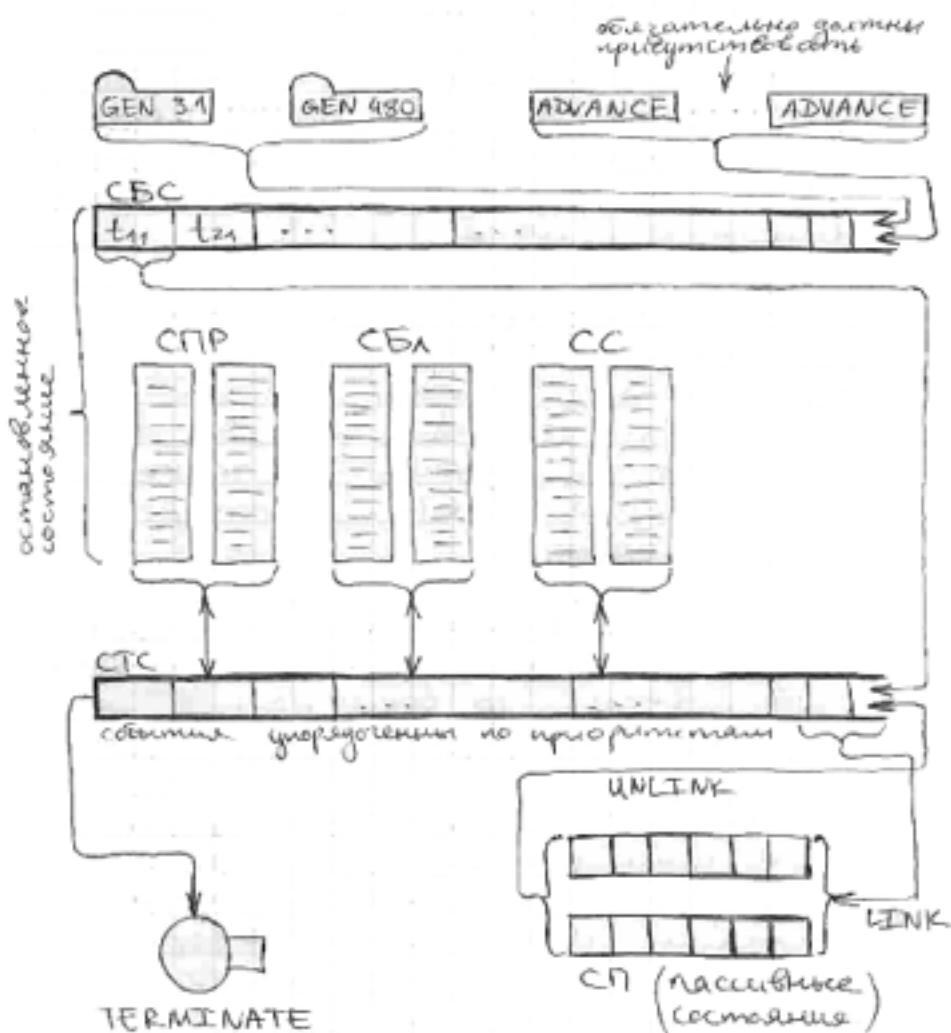
Каждый блок определяется с помощью отдельной команды.

В общем случае: сначала идет нумерация (как в Basic'е), затем обязательно поле метки, затем поле операции, поле operandов, затем, если необходимо, комментарий (через ; - точку с запятой). Т.е.:

<Нумерация><Оператор><Метка><Операнды><Комментарии>

### Управление процессом моделирования.

В системе GPSS интерпретатор (программа управления моделированием) поддерживает сложные структуры организации списков:



С целью сокращения затрат времени при просмотре списков система GPSS ведет 2 основных списка событий.

1. **Список текущих событий.** СТС, куда входят все события, запланированные на момент текущего времени не зависимо от того условные они или безусловные. Программа управления моделированием просматривает в первую очередь этот список и пытается переместить по модели те транзакты, для которых выполнены условия. Если в этом списке таких транзактов нет, то интерпретатор обращается к другому списку СПС перенося все события, которые запланированы на ближайший момент времени (время только модальное) из списка СПС и повторяет его просмотр. Такой перенос определяется так же в случае совпадения текущего времени моделирования со временем первого события в списке будущего события. В СПС транзакты размещены в порядке уменьшения приоритета. Транзакты с одинаковыми приоритетами размещаются в соответствии с последовательностью поступления в список. Каждый транзакт в списке может находиться или в активном состоянии и рассматривается интерпретатором в данный момент времени или в состоянии задержки.

#### [20.11.2006][Лекция 17]

В начальный момент (при выполнении оператора управляющего START, который начинает фазу интерпретации модели) управляющая программа обращается ко всем блокам GENERATE модели. Каждый из этих блоков планирует момент появления транзактов и заносит их в СБС (список будущих событий). После чего управляющая программа обращается к списку текущих событий (СТС) и выбирает из него все транзакты запланированные на ближайшие моменты времени и переносит их СТС. После чего пытается продвинуть первый транзакт этого списка по блокам модели. Если перемещение транзакта было задержано по какой либо причине, не связанной с блоком ADVANCE, то он остается в СТС и управляющая программа пробует перемещать этот транзакт далее по блокам. Если же транзакт вошел в блок ADVANCE, то планируется его выход из этого блока и этот транзакт переносится в СБС.

СТС и СБС можно просмотреть, если использовать команду EVENTS или же в окне списков. Для эффективной процедуры просмотра важен порядок транзактов, движение которых заблокировано. Движение транзактов может быть заблокировано т.к. ожидают какие-нибудь ресурсы. Простейшим решением является пересмотр всех ожидающих транзактов для каждого нового модельного значения времени и выбор тех, у которых снято условие блокировки. Если исследуемая система перегружена, то данный способ с точки зрения затрат компьютерного времени не подходит, т.к. каждый транзакт просматривается многократно до того как выйдет из состояния блокировки. Если причина перевода транзакта в состояние блокирования – это состояние определенного ресурса системы, то более лучшим является способ обработки, по которому заблокированный по этой причине транзакты вообще не просматриваются до тех пор, пока не изменится состояние ресурса. Реализация такого способа предполагает регистрацию для каждой единицы ресурсов транзактов, движение которых

заблокировано ввиду состояния именно этого ресурса. Если транзакты находятся в активном состоянии, то процедура просмотра пытается переместить их к следующим блокам. Если же перемещение таких транзактов блокируется какими-либо ресурсами, то вхождение транзакта в блок невозможно и он переводится в состояние задержки. Такие транзакты не просматриваются и размещаются в списке задержек. Если при просмотре текущего активного транзакта произошло изменение ресурса, то просмотр начинается с начала. И опять обслуживаются все транзакты из СТС, которые находятся в активном состоянии.

**Список блокировок** – это список транзактов, которые ожидают изменения состояния ресурса. Существует 6 видов связанных с устройствами, 7 видов связанных с многоканальными устройствами и 2 вида связанных с ключами.

С устройствами используются списки для занятых и незанятых, доступных и недоступных устройств и устройств работающих без прерывания и с прерыванием.

С многоканальными устройствами используются списки для заполненного, незаполненного, пустого, непустого, доступного и недоступного устройства и транзактов, которые могут войти в это устройство.

С логическими ключами связаны списки для включенных и выключенных ключей.

**Список прерываний** содержит прерванные во время обслуживания транзакты, а так же транзакты, вызвавшие прерывания. Этот список используется для организации и обслуживания одноканальных устройств по абсолютным приоритетам, что позволяет организовать приоритетные дисциплины обслуживания транзактов.

**Список синхронизации** содержит транзакты, которые на данный момент времени сравнивают. Этот список работает с транзактами, полученными с помощью блока SPLIT, который создает транзакты копии, принадлежащие одному семейству или ансамблю.

Синхронизацию транзакта одного семейства выполняют следующие блоки:

- MATCH – синхронизирует движение транзакта с другим блоком
- ASAMBLE – собирает все копии транзактов и выдает один начальный транзакт
- GATHER – собирает заданное количество транзактов и задерживает их до тех пор пока не соберется необходимое количество копий транзакта.

Блок SPLIT можно использовать многократно.

Остановленные процессы находятся в СБС, списках синхронизации и списках блокировок.

**Список пользователя** содержит транзакты, выведенные пользователем из СТС с помощью блока LINK и помещенные в список пользователя как временно неактивные. При работе симулятора они недоступны ему до тех пор, пока не будут возвращены пользователем в СТС с помощью блока UNLINK.

Моделирование заканчивается тогда когда **счетчик завершений** (стандартный числовой атрибут), инициализированный оператором START будет равен 1. Или когда в списках СТС и СБС не будет ни одного транзакта.

Блоки, связанные с динамической категорией.

Следующие группы:

1. Задержка транзактов по заданному времени.
2. Создание и уничтожение транзактов.
3. Изменение параметров транзактов ( $1020 = 4 * 255$ )
4. Создание копий транзактов
5. Синхронизация движения транзактов

### **Задержки транзактов по заданному времени.**

ADVANCE A,B

Блок задает среднее время выполнения операций в моделируемой системе, а так же разброс времени относительно среднего. Задержка – целое число.

Для задания времени пребывания в блоке ADVANCE пользователь указывает среднее время в поле А, а модификатор в поле В. Если поле задержки постоянно, то поле В может быть пустым. А если нулевое, то и поле А может отсутствовать.

Модификаторы могут быть двух типов:

1. Модификатор «**интервал**», используется, когда время задержки транзакта распределено равномерно в некотором заданном интервале.  
Например: ADVANCE 5,2 (т.е. интервал от 3 до 7)
2. Модификатор «**функция**», когда интервал отличается от равномерного и приходится с помощью этого блока находить данное время. Указываем среднюю величину, а дальше функцию, на значение которой должна быть умножена данная величина.

Например: ADVANCE 3, FN\$XPDIS

**Параметры транзактов** – свойства транзактов, определяемые пользователем, т.е. набор стандартных числовых атрибутов (СЧА), которые принадлежат транзакту.

Параметры, по сути, являются локальными переменными, которые доступны только этому транзакту.

В процессе перемещения транзакта по модели его параметры могут задаваться и модифицироваться в соответствии с логикой работы модели.

Особенности параметров.

1. Задаются:  
P <номер>  
P \$<имя>  
где Р – стандартный числовой атрибут транзакта определяющий его групповое имя.
2. Номера или имена конкретных членов множества параметров задаются с помощью целых чисел или символьных имен.
3. При входе транзакта в модель начальные значения параметров равны нулю. Значения всех параметров транзактов и их изменение определяет сам пользователь. Причем эти значения могут быть любыми числами, в том числе и отрицательными.
4. Транзакт может обращаться только к своим параметрам. Если необходим доступ к параметрам других транзактов, то это можно сделать с помощью ячеек сохраняемых величин или использовать группы транзактов.

5. Параметры можно использовать в качестве operandов блоков или в качестве аргументов.
6. Параметры также позволяют организовать косвенную адресацию.  
  <самостоятельно>

## **Группа блоков создания и уничтожения транзактов.**

### **Блок GENERATE A,B,C,D,E**

Функцией данного блока является создание транзактов входящих в систему.

В поле А задается среднее время между поступлением отдельных транзактов. Как и в блоке ADVANCE, это поле может быть модифицировано с помощью модификатора находящегося в поле В (также интервал или функция). В поле может быть записан NULL. Если при вычислении времени появления в системе 1-ого транзакта, оно получилось равным 0, то симулятор полагает его равным 1.

Задаваемый модификатором интервал не должен превосходить среднего, записанного в поле А.

Интервал между транзактами, т.е. время появления следующего транзакта вычисляется только после того, как генерируемый транзакт покидает блок GENERATE. Поэтому если после блока GENERATE стоит блок, который может по какой либо причине задержать транзакт, то время генерации следующего транзакта будет вычислено после снятия блокирующего устройства, т.е. когда сгенерированный транзакт пройдет следующий за блоком GENERATE блок. Поэтому средний интервал между транзактами будет больше чем среднее значение заданное в поле А. Что приводит к ошибке. Избежать её можно поместив после блока GENERATE блок, не задерживающий транзакт.

В поле С записывается начальная задержка. Заданное в этом поле число без модификации определяет интервал времени до создания данным блоком первого транзакта. По отношению к А оно может быть любым.

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

В поле Е задается приоритет присваиваемый генерируемому транзакту. Если поле пусто, то нулевой приоритет.

Поля F – I: максимальное число параметров каждого типа.

*Пример.*

```
GENERATE 10, 3, 100, 16, 5, 5PB, 20PH, 3PL, 4PW
```

Каждый транзакт имеет по 5 параметра формата «байт», 20 формата «..», 3 формата «..», 4 формата «..» (?)

### **Блок TERMINATE A**

Удаляет транзакты из системы. Он используется для обозначения окончания пути транзакта.

Поле А указывает изменяет ли этот блок содержимое счетчика завершения в момент поступления транзакта и, если изменяет, то на сколько единиц.

## **Изменения параметров транзакта.**

### **Блок ASSIGN A, B, C, D**

Является основным средством для задания параметров транзактов.

В поле А указывается какой параметр поступившего транзакта должен быть изменен. Следующий непосредственно за номером транзакта символ указывает что нужно сделать с записанным в поле В целым числом. Прибавить (+), вычесть (-) или заменить этим числом.

Если в поле С указано значение, то оно интерпретируется как номер функции. Определяется значение этой функции, а результат используется для модификации целого числа, указанного в поле В. Произведение помещается в параметр, указанный в поле А.

В поле D задается тип изменяемого параметра.

*Пример.* ASSIGN 1, 4 (присвоили первому параметру 4).

### **LOOP A, [B]**

Циклы можно организовывать с помощью параметров. Блок LOOP управляет количеством повторных прохождений транзактов определенной последовательности блоков модели.

А – параметр транзакта, используемый для организации цикла. Оно может быть именем, числом, стандартным числовым атрибутом.

В – метка (имя) блока начального блока цикла.

Когда транзакт входит в блок LOOP транзакт указанный в операнде А уменьшается на 1. А затем проверяется его значение на равенство нулю. Если нулю не равно, то транзакт переходит в блок указанный в операнде В, если равен, то в следующий блок.

### **[24.11.2006][Лекция 18]**

*Задача:* построить программу модели процесса прохождения 70 деталей, поступающих с интервалом времени 12+2; и обработка происходит 1 рабочим по 5-ти последовательно идущим операциям, времена которых также распределены равномерно в интервале 2 +1 ед. времени.

```
GENERATE    12,2
ASSIGN      2,5          // P2=5
SEIZE       1
WAIT        ADVANCE     2,1
              LOOP       2, WAIT
              RELEASE     1
              TERMINATE  1
              START      70
```

## **Группа блоков, создания копий транзактов.**

### **SPLIT A,B,C**

В отличие от блока GENERATE данный блок не создает самостоятельных транзактов, а лишь генерирует заданное число копий входящего в него транзакта. Число копий задается в поле А.

После прохождения блока SPLIT исходный транзакт направляется в следующий блок. А все копии пересылаются по адресу, указанному в поле В.

Исходное сообщение и копии являются равноправными и могут снова проходить через любое количество блоков SPLIT. Все транзакты полученные копированием, а так же копии копий принадлежат к одному **ансамблю**. И далее к этому ансамблю можно применять специальные операции или блоки, осуществляющие обработку ансамблей транзактов. Получаемый ансамбль транзактов может быть пронумерован. Для этого в поле С записывается номер параметра транзакта, в котором будет произведена нумерация. Если в исходном транзакте значение этого параметра было равно по величине k, то после нумерации исходный транзакт получит номер k+1, первая копия k+2 и т.д.

Копии, полученные в блоке SPLIT, могут иметь число и типы параметров отличные от исходного.

### **Группа блоков синхронизации движения транзактов.**

#### **ASSEMBLE A**

Блок ASSEMBLE для объединения определенного числа транзактов, являющихся членами одного ансамбля. Число определяемых ансамблей указывается в поле А. Транзакты, принадлежащие одному ансамблю, будут задерживаться в блоке ASSEMBLE до тех пор, пока не поступит заданное число транзактов этого ансамбля.

В результате на выходе блока появляется один первый транзакт ансамбля, а остальные транзакты уничтожаются. В одном блоке ASSEMBLE могут накапливаться транзакты разных ансамблей. Транзакты одного ансамбля могут накапливаться в разных блоках ASSEMBLE. Если число собираемых транзактов задается с помощью косвенной адресации, то для его установления используется параметр первого пришедшего транзакта.

**Задача.** Построить модель прохождения 100 деталей, поступление которых подчиняется равномерному закону в интервале 8+2 ед. времени. И обработка производится параллельно двумя рабочими, каждый из которых выполняет свою операцию независимо друг от друга со временем 5+3.

Определить коэффициенты занятости этих рабочих.

Распараллеливание – split.

Передать (transfer) и собрать (assemble).

```
GENERATE 8,2
SPLIT 1, Lwrk2          // копию второго рабочего

// начало обработки детали
SEIZE      Wrk1
ADVANCE    5,3
RELEASE    Wrk1
TRANSFER   ,LJoin

LWrk2 SEIZE      Wrk2
        ADVANCE   5,3
```

```

RELEASE      Wrk2

LJoin ASSEMBLE    2
TERMINATE     1

START 100

```

Изменить программу так, чтобы можно было написать START 1

\*сделать всем на gpss.

~=

```

GENERATE 8,2,,100
SPLIT 1, LWrk2           // копию второго рабочего

// начало обработки детали
SEIZE      Wrk1
ADVANCE   5,3
RELEASE   Wrk1
TRANSFER, LJoin

LWrk2 SEIZE Wrk2
ADVANCE 5,3
RELEASE Wrk2

LJoin ASSEMBLE    2
TERMINATE     1

START 1

```

Разница будет в том, что первый закончит свою работу кода 100 уничтожится. А второй когда 100 создастся.

## GATHER A

Действие блока GATHER аналогично действию ASSEMBLE. Отличие в том, что после накопления в блоке числа транзактов указанного в поле А – они все передаются в следующий блок. Этот блок позволяет синхронизировать движение транзактов одного ансамбля по одному пути.

*Задача.* Необходимо моделировать 80 деталей. Каждая деталь является подшибником (поступают обоймы и шарики) с интервалом времени 25+-4 ед. времени. На контроль обоймы затрачивается 4+-1 ед. времени. Контроль шариков производится последовательно со времени 2+-1 ед. времени на шарик. Операция сборки требует одновременного поступления обоймы и всех шариков и производится со временем 4+-2.

```

GENERATE    25,4
SPLIT       8,Sharik
SEIZE       OboimaControl
ADVANCE    4,1
RELEASE    OboimaControl
TRANSFER   FINAL

Sharik     SEIZE     SharikControl
            ADVANCE   2,1
            RELEASE   SharikControl

            GATHER    8

FINAL      ASSEMBLE 9

```

```

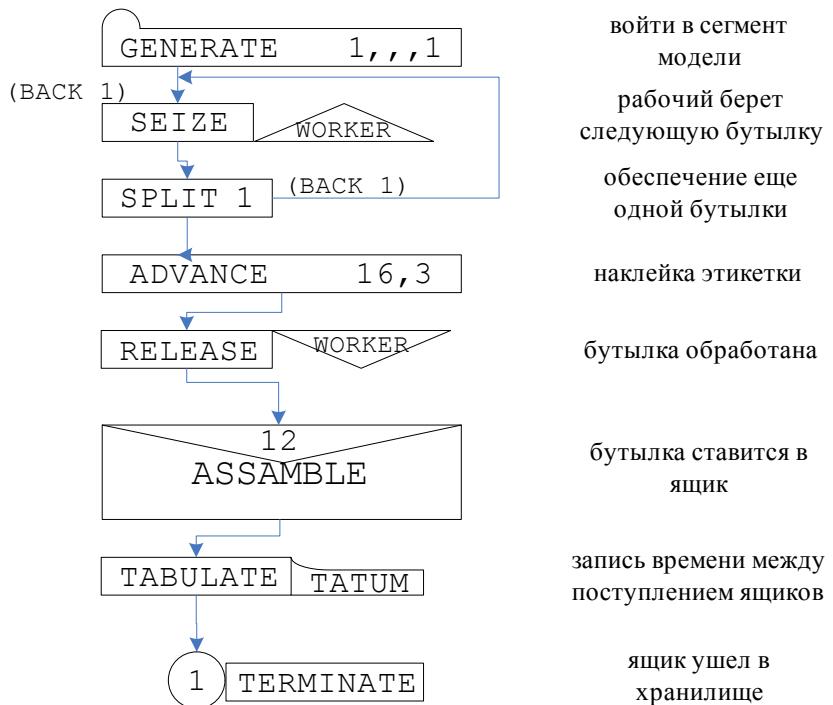
SEIZE      Sborka
ADVANCE    4,2
RELEASE    Sborka

TERMINATE  1
START      80

```

Если вместо GATHER поставить ASSEMBLE. Что будет?

*Задача.* Рассмотрим часть производственного процесса на небольшом винном заводе в Калифорнии. Объем продукции не оправдывает приобретение машины для автоматической наклейки этикеток на бутылке. Этую операцию проделывает один рабочий вручную. Ему требуется 16+3 секунды. В каждый ящик умещается 12 бутылок. Оценить интервалы времени между перемещениями полных ящиков в хранилище.



Оператор SPLIT обеспечивает неограниченный источник бутылок и осуществляет необходимое условие принадлежности всех входящих транзактов одному ансамблю.

## MATCH A

Блок MATCH A предназначен для синхронизации продвижения двух транзактов одного ансамбля, движущихся по разным путям. Для синхронизации необходимо 2 блока MATCH A, находящиеся в соответствующих местах блок-диаграммы и называемые сопряженные.

В поле A каждого блока MATCH A указывается метка сопряженного ему блока. При подходе этого транзакта к этому блоку, проверяется наличие в сопряженном ему блоке транзакта из того же ансамбля. Если в обоих блоках имеются транзакты одного ансамбля, то они одновременно пропускаются через блоки MATCH A. Иначе поступающий транзакт будет ожидать поступления транзакта того же ансамбля в сопряженный блок MATCH A. После чего они оба будут пропущены в следующие за блоками MATCH A блоки.

## [27.11.2006][Лекция 19]

*Задача.* Построить программу моделирования для исследования обработки 500 деталей. Детали поступают с интервалом времени 300+-50 ед. Обработку производят 2 рабочих по 2-м операциям. После первой операции выполняемой 1-ым рабочим со временем 70+-20ед. и вторым 60+-30 производится операция сверки (время её выполнения = 0). После сверки выполняется вторая операция первым рабочим со временем 20+-10 и вторым 30+-20. Затем 3-ий рабочий производит сборку изделия из этих деталей со временем 50+-20. Все процессы подчиняются равномерному закону.

---

При сдаче лабораторных написанных на GPSS - знать следующую «теоретическую часть» «Методика построения дискретной модели в среде GPSS и оценка результатов моделирования». Нужно написать как строить программу. Т.е. как запустить, какие окошки открыть, почему пишем, например, START 1.

---

Решение (условие задачи выше)

```
// поступление деталей
GENERATE    300,50

        SPLIT      1,Worker2

Worker1   SEIZE      1
          ADVANCE    70,20
Check1    MATCH     Check2      ; сверка
          ADVANCE    20,10
          RELEASE    1
          TRANSFER   ,Worker3

Worker2   SEIZE      2
          ADVANCE    60,30
Check2    MATCH     Check1      ; сверка
          ADVANCE    30,20
          RELEASE    2

Worker3   ASSEMBLE   2
          SEIZE      3
          ADVANCE    50,20
          RELEASE    3
          TERMINATE  1
          START      500
```

### Блоки, определяющие аппаратную категорию.

#### PREEMPT A,B,C,D,E

Фиксирует использование устройства на более высоком уровне, чем блок SEIZE, а также приостанавливает обслуживание транзакта захватившего устройство ранее и предоставляет возможность прерванному транзакту захватить устройство после того, как закончится обслуживание прервавшего транзакта. Если при реализации данного блока оказывается, что одно прерывание уже произошло (устройство обслуживает прерывание), то данный блок не может выполниться, и транзакт задерживается до тех пор, пока не освободится устройство. Затем обслуживается новый прерывающий транзакт (а не прерванный).

*Исключение:* когда блок PREEMPT работает в режиме приоритетов (т.е. в поле В стоит мнемоническое значение PR), он подразумевает разрешение прерывания в зависимости от приоритетности транзактов.

Для последующей обработки прерванных транзактов существуют следующие возможности.

В поле С может быть описан какой-либо блок, на который будет передан прерванный транзакт. При этом прерванный транзакт будет претендовать на тот блок, который указан в поле А. Если прерванный транзакт находится в поле ADVANCE, то вычисляется остаток времени от момента прерывания до момента выхода из блока ADVANCE, и полученное значение помещается в параметр описанный в поле D. Если в поле Е данного блока стоит мнемоническое обозначение RE, то блок будет проводить обычные операции за исключением того, что прерванный транзакт больше не участвует в конфликте из-за захвата устройства.

### **RETURN A**

Говорит об окончании прерывания. При входе в блок, задержки возникнуть не может, но закончить прерывание может только тот транзакт, который перед этим прошел блок PREEMPT, относящийся именно к данному устройству. Прерывание заканчивается в момент входа транзакта в блок RETURN.

Время, в течение которого транзакт находится в прерванном состоянии, не фиксируется.

На прерывания имеются следующие ограничения: нельзя производить прерывание транзакта, захватившего или прервавшего обслуживание других транзактов.

Имя\_устройства, задержка, конец\_прерывания.

### **FUNAVAIL A, B–H**

Выполняет операции переводящие устройства в состояния *недоступности*. Недоступность устройства предупреждает прерывания или занятие устройства последующими сообщениями. При этом возможно задание специальных режимов работы для данного блока обеспечивающих окончание обслуживания последнего транзакта, передачу его другому блоку до обслуживания транзакта после окончания периода недоступности устройства.

В поле А – номер или диапазон номеров, переводимых в состояние недоступности  
Поля В–Н – для задания специальных режимов.

### **FAVAIL A**

Делает доступным устройство с указанным номером или диапазоном в поле А. Отменяет все режимы, заданные блоком FUNAVAIL для данного устройства.

*Пример.*

FUNAVAIL	1-15
ADVANCE	30
FUNAVAIL	1-10
ADVANCE	15
FANAVAIL	11-15

Устройства 1-15 становятся доступными через 30 ед. времени, 11-15 через 45.

Элементы исследуемой системы предназначенные для хранения или обработки нескольких транзактов называются **памятью**. Для того чтобы описать память используют команду **STORAGE**. А изменение состояния памяти производится операторами **ENTER**, **LEAVE**, **SUNAVAIL**, **SAVAIL**.

### **ENTER A,B**

Поле **A** – интерпретируется как номер памяти.

Поле **B** указывает число единиц памяти, занимаемых транзактом при входе в блок.

При выходе транзактов из блока ENTER никаких изменений в содержимом памяти не происходит. Если поле **B** пусто, то число единиц памяти полагают = 1. Если в памяти нет достаточного числа свободных единиц, чтобы удовлетворить запрос транзакта, то этот транзакт не может быть обслужен оператором ENTER. А если для последующего это число единиц достаточно, то он входит в память раньше первого.

### **LEAVE A,B**

Поле **A** определяет имя памяти.

Поле **B** — число единиц, которые надлежит освободить при входе транзакта в блок.

Не всегда освобождается такое же число единиц памяти, какое было занято. Транзакт, освобождающий память, не обязательно должен был ее занимать. Однако необходимо, чтобы в сумме освобождалось столько единиц памяти, сколько было занято. Освобождать можно 0 единиц.

При реализации блока LEAVE задержка не возникает.

*Пример.*

```
ENTER      1,1
SEIZE      2
LEAVE      1,*2
// * - косвенная адресация (т.е. освободим ровно столько сколько
содержалось в 2.)
```

Реализуется занятие единицы памяти *памяти №1*, а затем происходит освобождение числа ед. памяти равное содержимому параметра 2.

### **SUNAVAIL A**

Переводит накопитель в состояние недоступности, при котором транзакты не могут войти в накопитель. Уменьшение содержимого накопителя в этот период может происходить путем прохождения транзакта через блок LEAVE.

Номер или диапазон номеров накопителей, переводимых в состояние недоступности, записывается в поле **A**.

### **SAVAIL A**

Оператор **SAVAIL** переводит заданный накопитель из состояния недоступности в состояние доступности. Если данный накопитель уже доступен, то оператор **SAVAIL** никаких действий не выполняет.

Номер или номера накопителей, переводимых в состояние доступности, записываются в поле **A**.

*Пример.*

SUNAVAIL 2-5 - делаются недоступными накопители с именами 2, 3, 4, 5

SAVAIL 2-5 - делаются доступными накопители с именами 2, 3, 4, 5.

**Логические ключи** предназначены для описания элементов моделируемой системы, которые могут находиться только в двух состояниях. Статистика о работе ключей не собирается. Логические ключи не имеют СЧА (стандартных числовых атрибутов). Но зато они имеют два логических атрибута, принимающих 0 при не выполнении и 1 при выполнении следующих условий:

LR ключ в состоянии 0.

LS ключ в состоянии 1.

В начале моделирования ключи могут быть установлены в состояние 1 с помощью команды INITIAL. А изменение состояния в процессе моделирования производится блоком LOGIC, который используется для установки логических ключей, состояние которых может быть запрошено в любом другом месте модели.

При входе в блок LOGIC задержки не возникает.

Состояние логического объекта, указанного в поле А изменяется одним из 3х способов:  
LOGIC S – установлен

R – сброшен

I – инвертирован

Вид изменения определяется соответствующим мнемоническим обозначением, идущим сразу за блоком LOGIC.

*Пример:*

```
LOGICS      4          // установить ключ 4
LOGICR      65         // сбросить ключ 65
LOGICI      4          // инвертировать ключ 4
```

## **Блоки, изменяющие маршруты транзактов.**

### **GATE О А, В**

Этот блок используется для определения состояния объектов устройств без изменения собственно их состояний и работает в двух режимах:

- 1) Отказа или условного входа. При работе в этом режиме блок не пропускает транзакты, если соответствующий объект не находится в требуемом состоянии.
- 2) Перехода или безусловного входа.

Поле А определяет номер объекта аппаратной категории (устройства, памяти или ключа).

Если в поле В указано наименование или номер блока, то вместо отказа блок GATE будет посыпать транзакт на указанный адрес. Следовательно, если поле В пусто, то блок работает в режиме отказа, - нет – в режиме перехода.

Существуют специальные логические атрибуты, описывающие состояние устройств, памяти, ключей и условий синхронизации. Мнемонические обозначения проверяемого условия записываются непосредственно после GATE.

Состояние устройства описывается следующими условиями:

- FNU — устройство не используется, свободно;
- FU — устройство используется, занято (обслуживает захвативший транзакт или прерывание);

- **FNI** — устройство работает без прерывания (свободно или обслуживает захвативший его транзакт);
- **FI** — устройство обслуживает прерывание;
- **FV** — устройство доступно;
- **FNV** — устройство недоступно.

Могут быть две мнемонические записи , которые позволяют проверить условие синхронизации:

M – выполнение условия

NM – невыполнение условия

*Пример.*

GATE SF, 16

Состояние памяти описывается условиями:

- **SE** - память пуста;
- **SNE** — память не пуста;
- **SF** — память заполнена;
- **SNF** - память не заполнена;
- **SV** - память доступна;
- **SNV** — память недоступна.

Состояние ключа описывается двумя условиями:

- **LR** — логический ключ в состоянии «выключен»;
- **LS** — логический ключ в состоянии «включен».

*Пример:*

GATESF 167 — блокировать транзакт до тех пор, пока память 167 не будет заполнена.

GATELS265 — блокировать транзакт до тех пор, пока ключ 265 не установлен

GATEFU 19 — блокировать транзакт до тех пор, пока устройство 19 не освободится

GATEFI34, ALTR — если устройство 34 прервано, то перейти к ALTR

## TEST О А,В,С

Описывает условие, которое проверяет при входе в него транзакта и определяет направление его дальнейшего движения в зависимости от условия, которое записывается в виде алгебраического соотношения двух аргументов.

При выполнении соотношения транзакт пропускается в следующий за блоком TEST блок. В случае невыполнения транзакт направляется в блок, метка которого указана в поле С. Если поле С пусто, то транзакт блокируется данным блоком до выполнения соотношения.

Проверяемое соотношение записывается сразу за блоком TEST, при этом используются классические мнемонические обозначения операции отношения.

Соотношение рассматривается между первым и вторым компонентами записываемых в полях А и В. Аргументы должны принадлежать к стандартным числовым атрибутам. Условие указывается сразу за именем оператора. Символы условий:

- **G** - больше,
- **L** - меньше,
- **E** - равно,
- **NE** - неравно,
- **LE** - меньше или равно,
- **GE** - больше или равно.

В случае не выполнения условия транзакт направляется в оператор, метка которого указана в поле **C**. Если поле C пусто, то транзакт при выполнении условия не сможет войти в блок TEST и управляющая программа в каждый момент модельного времени будет проверять, не изменилось ли блокирующее условие. Такой режим является нежелательным вследствие больших затрат машинного времени на многократную проверку блокирующего условия.

#### *Пример.*

TESTE V7,256,LAB – переход по условию (условная передача управления) : перехода нет, если переменная V7 = 256, иначе переход к оператору с номером LAB .

TESTL S1,10 – если число транзактов в памяти S1<10, то выполнять следующий оператор. Иначе остановить движение транзакта.

TESTG C1,120 – если системное время больше 120 единиц, то выполнять следующий оператор. Иначе остановить движение транзакта.

TESTE P1,2,MET1 – перехода нет, если переменная первый параметр транзакта равен 2, иначе переход к оператору с номером MET1 .

## **TRANSFER A,B,C,D**

Этот блок обычно используется для того, чтобы передать в него транзакты не следующие по номеру за ним. Передача может быть выполнена – логически, статистически, условно и безусловно.

Вид передачи определяется мнемоническим изображением указанным в поле А. Если безусловно, то указывается один следующий блок.

Поле В определяет первый или единственный из следующих блоков.

Поле С определяет следующий блок и интерпретируется с режимом работы блока TRANSFER.

- Если поле А пусто, то все транзакты приходящие на этот блок будут переданы на блок, определяемый в поле В.
- Если в поле А стоит BOTH, то каждый транзакт поступающий в этот блок проверяет два пути. Сначала проверяется блок указанный в поле В и если транзакт не может войти, то он пытается войти в блок указанный в блоке С, а если он не может войти и туда, то вынужден опять постоянно проверять эти условия. И происходит задержка в блоке TRANSFER.
- Если в поле А стоит ALL, то транзакты входящие в блок могут опрашивать много путей. Поле В – в этом случае определяет первый определяемый блок, поле С – последний, поле D – индексную константу, которая предоставляет пользователю возможность пользователю опрашивать определенные блоки, находящиеся между первым и последним.
- Если стоит SIM, то выбирается один из возможных путей.

### Статистический режим выбора

Если в поле А блока TRANSFER записана десятичная дробь, то производится случайный выбор между блоками В и С. Вероятность перехода в блок С задается эта дробь.

#### *Пример.*

```
TRANSFER 0.607,Work1,Work      // транзакт с вероятностью 0.607 пойдет на второго рабочего, а с вероятностью 1-0.607 перейдет к оператору с меткой Work1.
```

```
TRANSFER PICK,STK7,STK21      // равновероятный переход к операторам с номерами STK7, STK7+1, STK7+2, . . . , STK21.
```

```
TRANSFER FN,AAA,5            // переход к оператору, метка которого равна сумме значения функции AAA и числа 5.
```

```
TRANSFER .P5,,МЕТ           // трехзначное число, записанное в параметре 5 транзакта, интерпретируется как вероятность (в долях от тысячи) того, что транзакт будет передаваться на метку МЕТ, а в остальных случаях - следующему оператору.
```

```
TRANSFER P,4,41              // переход к оператору, метка которого равна сумме значения параметра 4 транзакта и числа 41.
```

```
TRANSFER SBR,PRC,7           // переход к оператору PRC с записью метки данного оператора в параметр 7 транзакта.
```

### [4.12.2006][Лекция 20]

---

Экзамен 12-ого в 9.00

---

консультация 11.01 ~532Л

---

## **Блоки, относящиеся к статистической категории**

Используются два типа объектов:

- очереди
- таблицы

### **QUEUE A,B**

Этот блок аналогичен блоку ENTER и осуществляет сбор статистики об очереди. Номер очереди в которую заносится транзакт задается в поле А. При записи нового транзакта в очередь определяется длина интервала времени, в течение которого длина очереди оставалась неизменной.

При входе транзакта в данный блок текущая длина очереди увеличивается на число единиц, указанное в поле В. Затем происходит сравнение с максимальной длиной очереди, достигнутой до этого момента времени. Если оно больше старого значения, то оно его заменяет. Кроме того, счетчик общего числа единиц прошедших через очередь увеличивается на тоже число единиц.

### **DEPART A,B**

Аналогичен блоку LEAVE.

Поле А интерпретируется как номер очереди.

Поле В задает количество единиц на которое уменьшается длина очереди.

Моделирующая программа вычисляет длину интервала времени в течение которого транзакт находился в очереди и если длина получается равной нулю, то указанное в поле В число единиц добавляется к счетчику, регистрирующему число транзактов прошедших через блок без задержки.

### **QTABLE A,B,C,D**

С помощью этой команды можно заносить в таблицу время пребывания транзакта в очередь.

А – номер очереди

В – начальное значение

С – шаг таблицы

Д – количество шагов

*Сбор статистики:*

QUEUE	Queue1
SEIZE	1
DEPART	Queue1
ADVANCE	10
RELEASE	1
QTABLE	Q1,0,5,100

### **TABULATE A,B**

Используется для создания таблиц нескольких типов.

Для занесения информации в таблицы с помощью специального блока TABULATE необходимо с помощью QTABLE или TABLE задавать характеристики таблицы.

При входе транзакта в блок TABULATE моделирующая программа записывает в соответствующую таблицу статистическую информацию.

Поле А определяет номер этой таблицы. В поле В заносится число единиц, добавляемых к числу наблюдений того интервала, в который попадает при данном обращении аргумент. Если В == 0, то полагается В == 1.

Предусмотрено несколько режимов табулирования, которые указываются в поле А. Знак минус за величиной, указанной в поле А, указывает на то, что в таблицу заносится не само значение, а разность между значением этой величины и последним значением, занесенным в таблицу. Такой режим называется разностным.

Если в поле А стоит мнемоническое обозначение RT, то при в ходе в блок TABULATE, который связан с таблицей, именно таким образом автоматическое обращение к классу частот не производится. Вместо этого число единиц заданное в блоке TABULATE добавляется к счетчику числа входов. Поэтому при описании блока в поле D должен быть определен временной интервал.

Если в поле А стоит мнемоническое обозначение IA, то моделирующая программа определяет время, прошедшее с момента последнего обращения к этой таблице. И такая таблица представляет собой распределение промежутков времени между моментами поступления транзактов в данную точку программы.

**Задача.** Простейшая телефонная система имеет две линии связи. Звонки, которые приходят извне, поступают каждые 100+-60 секунд. Когда линия занята, абонент набирает номер повторно через 5+-1 минуты. Требуется осуществить табулирование распределения времени, которое необходимо каждому абоненту, чтобы установить связь и произвести разговор. Сколько времени понадобится, для реализации 200 разговоров. Продолжительность разговора 3+-1 минуты.

200	SETS	STORAGE	2
210	TRANSIT	TABLE	M1,100,100,20
220		GENERATE	100,60
230	AGAIN	GATE	SNF SETS, OCCUPIED
240		ENTER	SETS
250		ADVANCE	100,50
260		LEAVE	SETS
270		TABULATE	TRANSIT
280		TERMINATE	1
290	OCCUPIED	ADVANCE	300,60
300		TRANSFER	, AGAIN

Комментарии:

200: память с именем SETS с общей емкостью 2 ед. берется для имитирования двух телефонных линий.

210: определяется таблица TRANSIT. Когда транзакт попадает в блок TABULATE, то его время прибывания в модель записывается в СЧА M1, т.е. длительность времени, отсчитанного с первого звонка абонента до тех пор пока абонент не закончит разговор.

220: транзакт, который имитирует вызов, создается каждые 100+-60 секунд.

230: блок GATE пересыпает блоку с меткой OCCUPIED, когда все линии заняты. Такая ситуация возможна, когда память заполнена и абонент должен ждать, прежде чем звонить повторно.

240: если память не занята, либо не занято только одно место, то транзакт проходит через блок ENTER, занимая тем самым место в памяти. Если все места в памяти заняты, то GATE не пропускает дальше транзакт. Каждый транзакт, пришедший в блок LEAVE, имитирует вызов, который был успешно осуществлен.

250: транзакт входит в блок ADVANCE, где задерживается на продолжительность разговора.

260: когда транзакт входит в блок LEAVE, он освобождает одно место в памяти с именем SETS, т.е. происходит имитация вновь свободившейся линии.

270: TABULATE добавляет длительность проведенного разговора к гистограмме.

280: выводит транзакт из системы, после того, как разговор завершен.

290: транзакт переходит в блок ADVANCE с меткой OCCUPIED, когда он пытался и не сумел занять в памяти SETS, т.е. происходит имитация абонента, который должен подождать, прежде чем заново начать набирать номер.

300: блок TRANSFER посылает каждый транзакт в блок GATE помеченный как AGAIN. Там транзакт снова пытается занять место в памяти. Т.е. абонент пытается перезвонить.

## **Определение функции в GPSS**

Она относится к управляющим операторам.

Формат: имя\_функции FUNCTION A,B

A – либо генератор случайных чисел (ГСЧ), либо СЧА.

B – тип функции. (D – дискретная, C – непрерывная, L – табличная (числовая), E – дискретная атрибутивная, M – табличная атрибутивная).

Дискретная функция (D) представляет собой кусочно непрерывную функцию, состоящую из горизонтальных ступенек.

Непрерывная функция (C) представляет кусочно-непрерывную, состоящую из соединенных между собой прямых отрезков. Получается ломаная линия.

Чтобы задать D-функцию необходимо задать координаты крайних точек горизонтальных отрезков. Для C-функции необходимо задать координаты всех точек, которые являются концами отрезков.

Действия необходимые для определения функций.

1. Присвоить функции имя. Имя либо словесное либо символьное.
2. Задать аргумент функции. Аргументами могут быть:
  - a. Ссылка на генератор случайных чисел, используемый для розыгрыша в соответствии с распределением заданной функции.
  - b. СЧА
  - c. Ссылка на любую другую функцию.
3. Задать тип функции и число крайних точек функции.
4. Задать значение аргумента и соответствующее значение функции.

За каждым оператором описания функции следуют операторы описания точек функции, т.е. значения точек x и y. Это операторы описания координат точек функции. Пишутся через запятую.

Особенности оператора описания:

1. Основной единицей информации оператора описания координат функций является пара координат i-ой точки ( $x_i, y_i$ ).
2. Значение координат одной точки отделяются друг от друга знаком «,». Последовательные наборы координат отделяются знаком «».
3. Все строки должны начинаться с первой позиции.
4. Необходимо соблюдать соотношение:  $x_1 < x_2 < \dots < x_n$ .

---

Самостоятельно:

Написать следующие типы функций в GPSS World.

1. Моделирование Пуассоновского потока.
2. Моделирование Гипер-экспоненциального распределения.
3. К-распределение Эрланга.
4. ? – распределение
5. Распределение Вейбулла.
6. Нормальный закон распределения
7. Беттономиальное, логистическое, лог-лаплассова, лог-нормальное. Обратно-гаусово и остальные...

---

См. в документе «Распределения.doc»

## [8.12.2006][Лекция 21]

Моделирование Пуассоновского потока:

$$P_k(t) = \frac{e^{-\lambda t} (\lambda t)^k}{k!}$$

$t = 2$

GENERATE 2#100, FN\$XPDIS //100 – потому, что округление до целого приводит к нарушению ордиарности потока: весь начальный отрезок получается на нуле.

Интервалы поступления заявок пуассоновского потока распределены по экспоненциальному закону. Согласно методу обратной функции можно получить ряд чисел, которые имеют экспоненциальное распределение, если для какого-то ряда случайных чисел  $\sim R(0,1)$  преобразовать эти числа в соответствии с функцией, обратной к экспоненциальной. Т.е. мы получаем:  $t \cdot \ln(\dots)$ , где  $t$  – разыгранный интервал времени.

Разработчиками GPSS была выполнена аппроксимация этой функции при  $\lambda = 1$  и функция  $e^{f(x)}$  была заменена 23 отрезками, которые преобразовали значение генератора в  $\log$  от этого значения.

Пуассоновский входящий поток с интенсивностью  $\lambda \neq 1$ , моделируется с помощью блока GENERATE следующим образом:

1. В качестве операнда А используют среднее значение интервала времени  $t = \frac{1}{\lambda}$ , где  $\lambda$  – интенсивность пуассоновского потока
2. В качестве операнда В используют СЧА, а именно значение функции XPDIS.

Если необходимо моделировать задержку со средним значением 3, то выполняем масштабирование и т.д.: ADVANCE 300, FN\$XPDIS

**Задача:** Необходимо решить какое число мест на стоянке для автомобилей, ожидающих мойки следует предусмотреть, чтобы их грузить по максимуму. Поток автомобилей является Пуассоновским со значением среднего интервала равным 5 минутам. Время мойки автомобиля распределено экспоненциально со значением среднего 4 минуты. Если клиенты подъезжают и не застают свободного места, то они уезжают. Исследовать систему при использовании 1, 2 и 3 мест на стоянке. Моделировать работу в течение 8 часового рабочего дня.

```

Park      STORAGE      1
          GENERATE    300, FN$XPDIS
          TRANSFER    BOTH,,Bye_bye

          ENTER        Park           ;заехали на стоянку
          SEIZE        Wash          ;заняли мойку
          LEAVE        Park           ;выехали со стоянки
          ADVANCE     240, FN$XPDIS   ; экспоненциальный закон
          RELEASE      Wash

Bye_bye    TERMINATE

```

```
GENERATE    28800
TERMINATE   1
```

## Моделирование вероятностных функций распределения GPSS World

В библиотеку процедур включено 24 вероятностных распределения. При вызове вероятностного распределения требуется определить 4 аргумента:

1. Stream – может быть выражением и определяет, как правило, номер генератора случайных чисел. При моделировании генератора случайных чисел создаются по мере необходимости и их явное определение необязательно.

Большинство вероятностных распределений имеет собственные параметры, которые называются Locate, Scale, Shape.

2. Locate – используется после построения примененного распределения и прибавляется к нему. Это позволяет горизонтально перемещать функцию распределения по оси X.
3. Scale – меняет масштаб функции распределения.
4. Shape – меняет форму.

*Задача.* Сгенерировать поток транзактов экспоненциального распределения с параметром  $\lambda = 0.25$  и использовать первый генератор случайных чисел.

```
GENERATE (Exponentilal(1,0, (1/0.25))
```

---

Остальные самостоятельно

---

## Классификация систем массового обслуживания

Признаки классификации:

- 1) закон распределения входного потока заявок
- 2) числа обслуживающих приборов
- 3) закон распределения времени обслуживания в обслуживающих приборах
- 4) число мест в очереди
- 5) дисциплина обслуживания

Для обозначения СМО принята система кодирования A|B|C|D|E, где

- А – закон распределения интервалов времени между поступлениями заявок.  
Наиболее часто используемое обозначение:
  - М - экспоненциальное
  - Е – эрлангово
  - Н – гипер-экспоненциальное
  - (?) – гамма
  - Д – детерминированное
  - Г – произвольное
- В – закон распределения времени обслуживания в приборах. Приняты те же обозначения, как и для интервалов между появлениеми заявок.
- С – число обслуживающих приборов. Для одноканальной – 1, для многоканальной –  $l$ .

- D – число мест в очереди.  
n или  $\infty$  – конечно  
<опущено> – неограниченно
- E – дисциплина обслуживания. Наиболее часто используются следующие варианты дисциплины обслуживания: FIFO (может опускаться), LIFO, RANDOM.

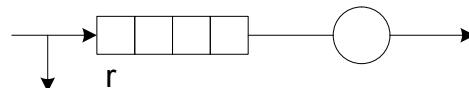
*Пример:* M/M/1 – СМО с одним ОА, бесконечной очередью, экспоненциальными законами распределения времени между поступлениями заявок и времени обслуживания, дисциплина обслуживания FIFO.

E/H/L/r/LIFO

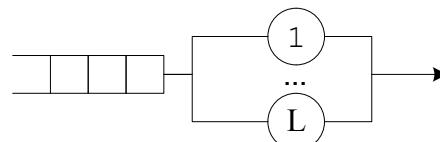
**G/G/1** - Одноканальные системы с ожиданием:



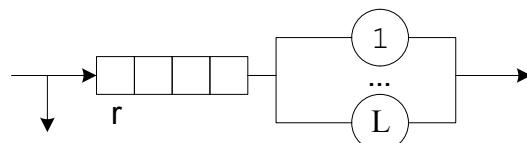
**G/G/1/r** - Одноканальная система с потерями:



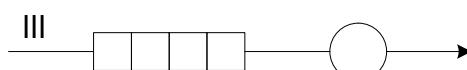
**G/G/l** - Многоканальная система с ожиданием



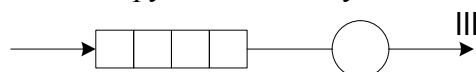
**G/G/l/r** – Многоканальная система с потерями



**Gr/G/1** – одноканальная система с групповым поступлением заявок:



**G/Gr/1** – одноканальная система с групповым обслуживанием



Для моделирования вычислительных систем и сетей наиболее часто используются следующие типы СМО:

1. Одноканальное СМО с ожиданием. Представляет собой один обслуживающий прибор с бесконечной очередью. Является наиболее распространенной при исследовании СДС. Формализует функционирование практически любого числа узла вычислительной сети.

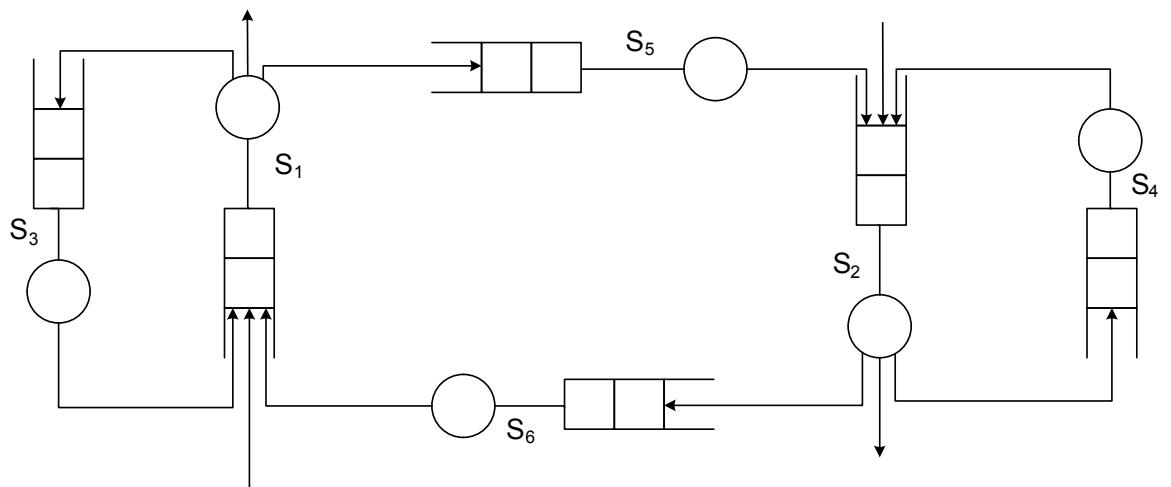
2. Одноканальная СМО с потерями. Один обслуживающий прибор с конечным числом мест в очереди. Используется при моделировании каналов передачи в вычислительных сетях.
3. Многоканальные СМО с ожиданием. Представляют собой несколько параллельно работающих обслуживающих приборов с общей параллельной очередью. Используется при моделировании групп абонентских терминалов, работающих в диалоговом режиме.
4. Многоканальные СМО с потерями. Наиболее часто используются при моделировании работы каналов.
5. Одноканальные СМО с групповым поступлением заявок. Также как и одноканальная СМО с групповым обслуживанием заявки используются для моделирования центров коммутации.

Вычислительные сети в целом могут быть исследованы с помощью сетей массового обслуживания.

Различают сети:

- 1) **Открытые.** Сеть массового обслуживания, состоящая из  $m$  узлов, причем хотя бы в один из узлов сети поступает извне входящий поток заявок и обязательно имеется сток заявок из сети.

Для открытых сетей характерно то, что интенсивность поступления заявок в сеть не зависит от состояния сети, т.е. от числа уже поступивших. Такие сети используются как правило, для исследования функционирования вычислительной сети, работающей в неоперативном режиме.



$S_1, S_2$  – моделируют работу узлов коммутации

$S_3, S_4$  – моделируют работу серверов

$S_5, S_6$  – моделируют работу межузловых каналов

В сети циркулируют два потока заявок. Каждая заявка поступает на вход соответствующего узла коммутации, где определяется место её обработки. Затем заявка передается на «свой» сервер или по каналу связи на соседний сервер, где заявки обрабатываются. После чего возвращается к источнику и покидает сеть.

- 2) **Замкнутые** – называются сети МО с множеством узлов без источника и стока, в которой циркулирует постоянное число заявок.

Замкнутые сети МО используются для моделирования таких вычислительных систем, источниками информации для которых служат абонентские терминалы, работающие в диалоговом режиме. В этом случае каждая группа абонентских терминалов представляется в виде многоканальной системы МО с ожиданием и включается в состав устройств сети.

Различают простой и сложный режим диалога.

- При простом: абоненты не производят никаких действий кроме посылки заданий в вычислительную сеть и обдумывания полученного ответа.

Схема (самостоятельно): группы абонентов, каналы связи с абонентами, узлы коммутации, серверы и каналы межузловой связи.

Абоненты с терминалов посылают запросы, которые по каналам связи поступают на узлы коммутации. А оттуда на обработку на свой или соседний сервер.

- При сложном режиме диалога работа абонентов представляется в виде совокупности операций некоего процесса, называемого *технологическим*. Каждая операция технологического процесса, моделируется соответствующей системой массового обслуживания. Часть операций предусматривает обращение к вычислительной системе, а часть может и не обращаться.

- 3) **Смешанные** – называются сети МО в которой циркулирует несколько различных типов заявок (трафик). Причем относительно одних типов заявок сеть замкнута, а относительно других открыта. С помощью смешанных сетей МО моделируют такие вычислительные сети часть которых работает в диалоговом режиме, а часть в неоперативном. Причем для диалоговых абонентов также различают простой и сложной режим работы.

Так же смешанными сетями МО моделируются вычислительные системы в которых сервер дополнительно загружается задачами, решаемыми на фоне работы сети.

## [11.12.2006][Лекция 22]

**Задача.** Для изготовления детали последовательно выполняется 3 операции, за каждой из которых следует 2 минуты контроля. После первой операции контроль не проходят 20% деталей, после второй и третьей контроль не проходит 15 и 5% соответственно. 60% деталей не прошедших контроль идут в брак. Остающиеся 40% нуждаются в повторном выполнении операции, после которой они не прошли контроль. Изготовление новой детали начинается в среднем через каждые 30 минут, распределенных экспоненциально.

Время выполнения первой операции задается таблицей.

Частота	0.05	0.13	.16	.22	.19	.15
---------	------	------	-----	-----	-----	-----

Время выполнения операции в минуту	10	14	21	32	38	45
------------------------------------	----	----	----	----	----	----

Вторая операция выполняется за  $15+6$  минут. Третья операция за время распределенное нормально при среднем 24 минуты и стандартом отклонения 4 минуты. Необходимо исследовать процесс при прохождении 100 единиц продукции.

Определить затраты времени и число забракованных деталей.

```

; установка генератора случайных чисел
RMULT 93 211
; создаем таблицу
TRANSIT TABLE M1,100,100,20
XPDIS FUNCTION RN1,C24
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915/.7,1.2/.75,1.38/.8,1
.6/.84,1.83/.88,2.12/.9,2.3/.92,2.52/.94,2.81/.95,2.99/.96,3.2/.97,3.5
.98,3.9/.99,4.6/.995,5.3/.998,6.2/.999,7/.9998,8

SNORM FUNCTION RN1,C25
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

THIRD FVARIABLE 24+4#FN$SNORM

PROCESS FUNCTION RN1,D7
0,0/.05,10/.18,14/.34,24/.56,32/.85,38/1.0,45

STAGE1      GENERATE 30,FN$XPDIS      ;30 умноженное на значение функции
           ASSIGN   1,FN$PROCESS
           SEIZE   Mashin1
           ADVANCE P1
           RELEASE Mashin1
           ADVANCE 2
           TRANSFER .200,,REVOKE1

STAGE2      SEIZE Mashin2
           ADVANCE 15,5
           RELEASE Mashin2
           ADVANCE 2
           TRANSFER .150,,REVOKE2

STAGE3      SEIZE Mashin3
           ADVANCE V$THIRD
           RELEASE Mashin3
           ADVANCE 2
           TRANSFER .05,,REVOKE3
           TABULATE TRANSIT

           TERMINATE 1

REVOKE1     TRANSFER .400,,STAGE1
           TERMINATE

REVOKE2     TRANSFER .400,,STAGE2
           TERMINATE

```

REVOKE3	TRANSFER .400,, STAGE3
	TERMINATE
START	100

Модель организована в несколько сегментов. После того, как определены таблицы, функции, переменные, идут 3 сегмента модели, каждый из которых отображает соответствующую операцию. Каждый транзакт представляет собой деталь на определенной стадии обработки. Единицы времени – минуты. Каждая операция имеет определенную вероятность того, что деталь после её реализации не пройдет контроль и в этом случае транзакт пересыпается либо обратно в блок с меткой REVOKE1, REVOKE2, REVOKE3 с вероятностью 40%, либо с вероятностью 60% попадает в брак.

## **Метод формализации для сложных дискретных систем и структур**

**Сложная система** (СС) – это система, обладающая, по крайней мере, одним из следующих признаков:

1. Запускает разбиение на подсистемы, изучение каждой из которых при исследовании с учетом влияния других подсистем в рамках поставленной задачи имеет содержательный характер.
2. Функционирует в условиях существенной неопределенности и воздействие среды на неё обуславливает случайный характер изменения её параметров или структуры.
3. Осуществляет целенаправленный выбор своего поведения.

Процесс проектирования сложных систем характеризуется:

- 1) высокой размерностью решаемых задач
- 2) наличием большого числа различных вариантов
- 3) необходимостью учета разнообразных факторов
- 4) в основе проектирования лежит блочно-иерархический подход; его сущность в уменьшении сложности решаемой проектной задачи за счет выделения ряда уровней абстрагирования, которые различаются степенью детализации представления об объекте.

**Дискретная система** – система, в которой состояния изменяются мгновенно во времени.

Среди методов выделяют:

1. Графовые методы (основные):
  - 1.1. Графы состояний
  - 1.2. Управляющие графы (ориентированный граф со взвешенными вершинами и ребрами)
  - 1.3. Граф-схемы алгоритмов (ориентированный граф)
  - 1.4. Графы Петри (сети Петри)(ориентированный граф)(самое сложное средство формализации) хорошо применимы при синтезе программ и очень хорошо используется при параллельном программировании.

2. Методы теории автоматов – используются в основном для описания последовательных процессов при формализации структур управления в виде функционально зависимых состояний.

Если нужно описывать динамику, то это сложно и нужно описывать несколько функционирующих систем.

// Формализация библиотек – хорошо применимы иерархические цветные сети.

# Билет 1

## 1. Философские аспекты (понятие эксперимента)

**Методическая основа моделирования** - это диалектический метод познания.

Научно техническое развитие в любой области обычно идёт по следующему пути:

- 1) Наблюдение и эксперимент
- 2) Теоретическое исследование
- 3) Организация производственного процесса

В научных исследованиях большую роль играет понятие «гипотеза», т.e определенные предсказания, основывающиеся на небольшом количестве опытных данных, наблюдениях или догадок.

Быстрая и полная проверка гипотезы может быть проведена в ходе специально поставленного эксперимента. Мы должны связать гипотезу и эксперимент.

**Аналогией** называется суждение о каком-либо частном сходстве.

Гипотеза и аналогия, отражающей реальный мир, должны обладать наглядностью или сводиться к удобным для исследования логическим схемам. Такие логические схемы, упрощающие рассуждения и логические построения, или позволяющие проводить эксперимент, уточняющий природу явления, называются моделями.

**Модель** - объект, заместитель объекта оригинала, обеспечивающий изучение некоторых свойств оригинала. Замещение одного объекта другим с целью получения информации о важнейших свойствах оригинала с помощью объекта модели называется **моделированием..**

## 2. Моделирование СМО с поступлением заявок по пуассоновскому закону и с обслуживающим прибором по экспоненциальному закону на языке GPSS

```
QUE GENERATE (Uniform(1,5,10)),,1000 ; генерируется 1000 заявок равномерным распределением [5, 10]
QUEUE STAND

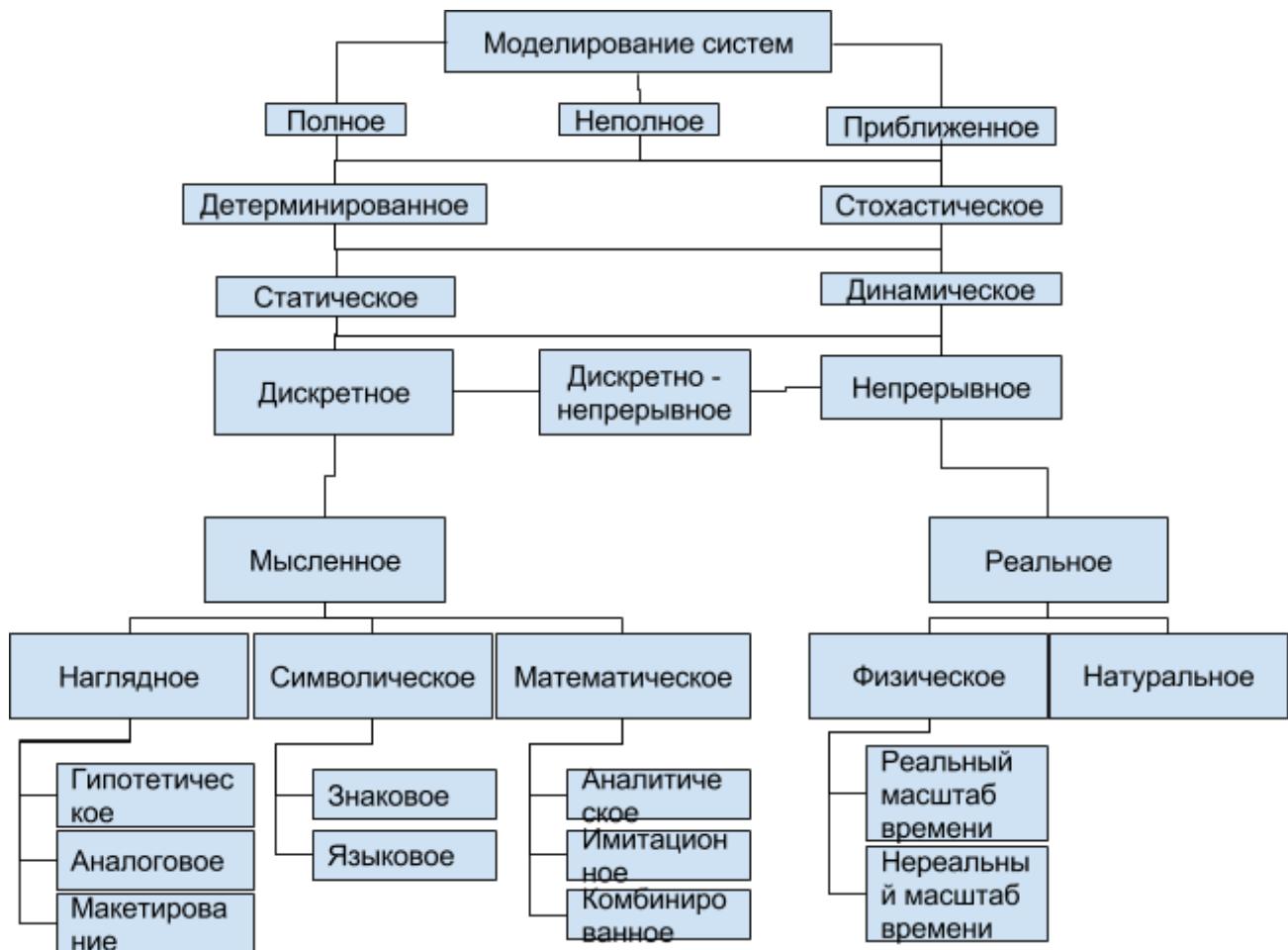
;-----Захват обработчика и заявки из очереди
SEIZE PROCUNIT
DEPART STAND
ADVANCE (Exponential(1,0,0.1)),, ; обработка по экспоненциальному распределению, бета = 1/lmb = 1/5 = 0.2
RELEASE PROCUNIT
TRANSFER .2,,QUE ; с вероятностью .2 заявка вернется к метке QUE

TERMINATE 1

RESET
START 1000
```

## Билет 2

### 1. Виды моделирования



По глубине моделирования методы делятся на две группы:

- Материальное - воспроизведение геометрических, физических и других свойств объекта
- Идеальное

В зависимости от типа носителя и сигнатуры:

1. Детерминированное - отображает процессы, предполагает отсутствие случайных воздействий
2. Стохастическое - тоже самое, но с/в учитываются
3. Статическое
4. Динамическое - исследование объекта во времени

## 2. Управляющие блоки

START A,B,C,D \* A – начальное значение счётчика завершения. \* В – может быть написано NP, признак подавления формирования стандартного отчёта. \* С – не используется. \* D – включение в отчёт списков текущих и будущих событий, если 1.

SIMULATE A \* A – ограничение реального времени в минутах.

RMULT – устанавливает значения ГСЧ.

RESET – сбрасывает всю статистическую информацию, накопленную в процессе прогона модели. При этом состояние аппаратных, динамических и запоминающих объектов, а также ГСЧ сохраняется.

CLEAR – модель приводится к состоянию до первого прогона (сбрасывается всё)

END – завершает сеанс работы с GPSS и возвращает управление в ОС.

## Билет 3

### 1. Технические средства математического моделирования

Принципы фон Неймана

1. Принцип однородности памяти: принципиальное отличие архитектуры "фон Неймана" (принстонской) от "Гарвардской". Команды и данные хранятся в одной и той же памяти и внешне в памяти неразличимы. Распознать их можно только по способу использования; то есть одно и то же значение в ячейке памяти может использоваться и как данные, и как команда, и как адрес в зависимости лишь от способа обращения к нему
2. Принцип адресности: структурно память состоит из пронумерованных ячеек, каждая из которых доступна процессору в любое время
3. Принцип программного управления: все вычисления, предусмотренные алгоритмом задачи, должны быть представлены в виде набора последовательно выполняемых команд, множество которых определяют возможности системы
4. Принцип двоичного кодирования: вся информация кодируется двоичными цифрами 0 и 1. Каждый тип информации имеет свой формат данных.

Виды вычислительных машин

Различают два вида: аналоговые и цифровые. Цифровая точна, но дискретна, а аналоговая быстра, но имеет узкий спектр применения. Аналоговые машины в качестве модели работают с величинами электрических цепей, которые идут в соответствии с переменными заданной модели. Преимущественно АВМ применяются для исследования объектов, динамику которых можно описать

дифференциальными уравнениями или ДУ в частных производных. Под АВМ стоит понимать совокупность электрических элементов, позволяющих провести изоморфное моделирование изучаемого объекта. По степени способности решения ИДУ различают малые, средние, большие АВМ.

Как особенный вид выделяются гибридные ВМ, позволяющие в разной степени применять обе формы представления и обработки информации. К таким машинам относятся:

- АВМ с цифровыми методами численного анализа, программируемые с помощью ЦВМ
- АВМ с цифровым управлением и логикой
- АВМ с цифровыми элементами
- ЦВМ с аналоговым арифметическим устройством
- ЦВМ с дифференциальным анализатором

Технические средства -- компьютеры -- используются в двух своих аспектах:

1. как средства расчета по полученным аналитическим моделям (ЦВМ, АВМ)
2. как средства имитационного моделирования

**Гибридная вычислительная машина** – вычислительная система, использующая как аналоговую, так и дискретную форму представления сигнала.

Гибридная ВМ:

Примеры гибридных ВМ:



- АВМ с цифровыми элементами (вольтметры, запоминающие устройства, но не решающие элементы)
- ЦВМ с аналоговыми арифметическими устройствами

## 2. Построение выражений в языке GPSS

GPSS World поддерживает широкое использование выражений. Они могут использоваться в PLUS-процедурах или в операторах GPSS (если заключены в скобки). Это означает, что высокий уровень мощности вычислений может быть достигнут уже в operandах блоков и команд. Выражения могут выполнять простые вычисления, вызывая процедуры, которые выполняют операции математического характера или операции над строками, выборку возможных распределений или осуществляют реализацию пользовательских алгоритмов, включая файлы ввода-вывода.

## Билет 4

### 1. Основные понятия в теории моделирования (и определение имитационной модели)

**Система** — множество элементов, находящихся в отношениях и связях между собой. Часть системы, представление о которой не целесообразно подвергать дальнейшему членению при моделированию — **элемент**.

**Сложная система** — система, характеризующаяся большим числом элементов и большим числом взаимосвязей. Именно взаимосвязями определяется сложность системы.

Свойства: членность, многоаспектность, целенаправленность, иерархичность.

**Подсистема** — часть системы (подмножество элементов и их взаимосвязи), которая имеет свойства системы.

**Надсистема** — система, по отношению к которой рассматриваемая система является подсистемой.

**Структура** — это отображение совокупности элементов системы и их взаимосвязей. Понятие структуры отличается от понятия системы тем, что при описании структуры принимают во внимание лишь типы элементов и связей без конкретизации значений их параметров.

**Параметр** — величина, выражающая свойство системы или ее части, или влияющая на систему через внешнюю среду.

При **имитационном** моделировании реализующий модель алгоритм воспроизводит процесс функционирования системы во времени, причем имитируются элементарные явления, составляющие процесс с сохранением их логической структуры и последовательности протекания во времени, что позволяет по исходным данным получить сведения о состоянии процесса в определенные моменты времени, дающие возможность оценить характеристики системы.

## 2. Графические возможности gpss (графики СЧА и гистограммы по таблицам)

Статистические таблицы используются для получения частотных распределений определенных аргументов, которыми могут быть некоторые СЧА (например, времени задержки транзакта в модели в целом или в отдельных ее частях; длин очередей; содержимого памяти и т. д.).

M TABLE A,B,C,D

A- табулир величина

B-верхняя граница левого интервала

C-ширина интервала

D-число интервалов

информация о гистограмме накапливается посредством попадания транзакта в блок:

TABULATE A

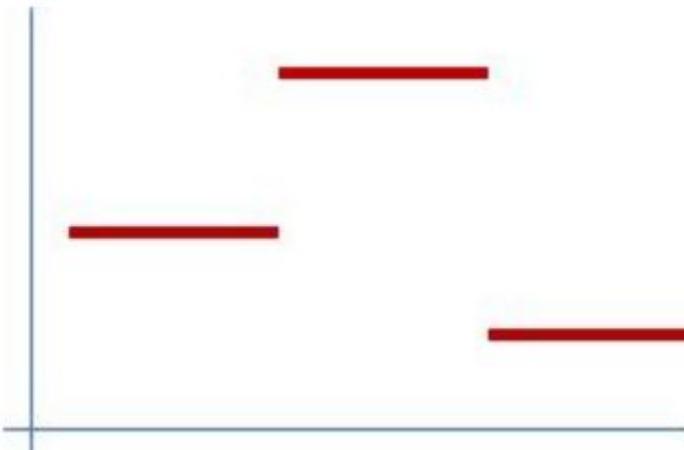
где A - имя гистограммы.

Функция – стандартный числовой атрибут, название и численная зависимость, обозначаемые в виде FN\$name. Описывается пользователем в виде численной зависимости от другого СЧА. Два основных типа – дискретные и непрерывные.

Пример: хранилище S с занятой памятью SMemory. Дискретная функция:

Func1 FUNCTION S\$Mem, D3 ;дискретная по трем точкам, ещё C, M, L, ..., S – стандартный числовой атрибут

5,12 / 9,20 / 12,6 ;дробная часть различается точкой



до 5 будет идти на уровне 12, до 9 на 20, до 12 на 6 (кусочная). Функция неявно продлевается влево и вправо если пользователь введёт соответствующий аргумент.

На граничных участках берётся «всё ещё» значение (в 5 – 12).

В случае непрерывной – не набор горизонтальных отрезков, а ломаная.

4.5, 12.1 / 5.5, 10.5 / 7, 14 / 9, 15.3



Если задается много точек, то новая точка начинается со следующей строки, причем знак слэша в конце не ставится. Во всех незаданных участках (влево и вправо) функция продлевается горизонтально.

Для обращения к функции: FN\$Func1

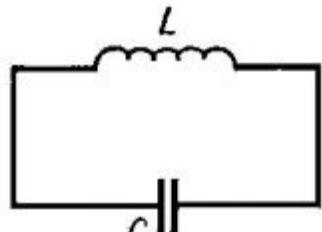
## Билет 5

### 1. D-схема

Процесс, описываемый этой схемой, имеет непрерывно-детерминированный подход и ТМС основана на дифференциальных уравнениях.

В качестве примера возьмем колебательный контур и маятник:  
колебательный контур с емкостью  $C_k$  и индукцией  $L_k$ , или маятник массой  $M$  и длиной  $l$ .

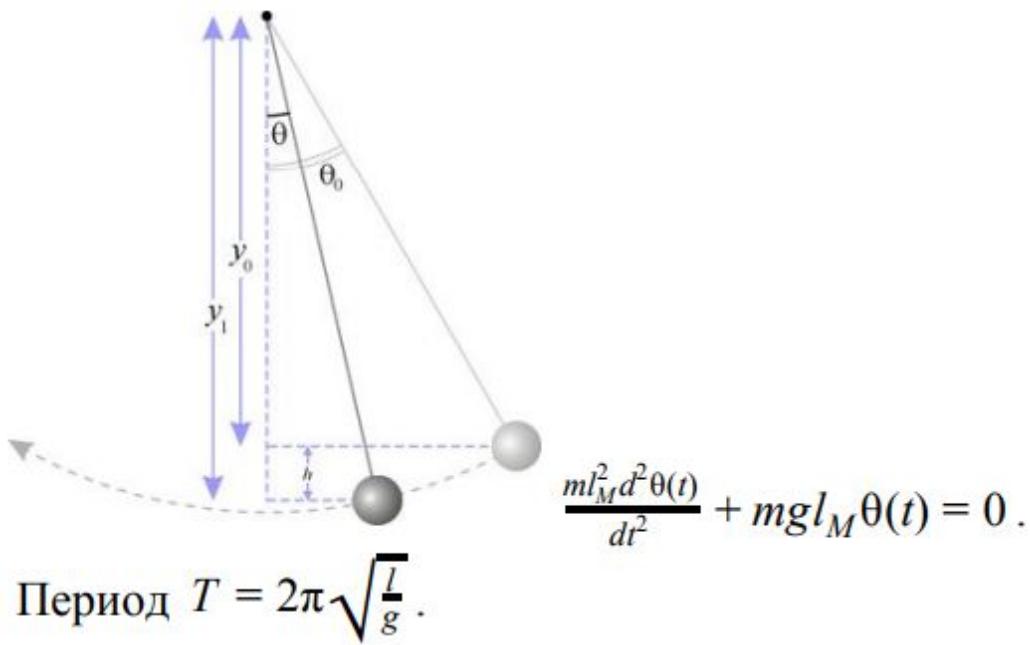
В контуре:



$$\frac{L_k d^2 q(t)}{dt^2} + \frac{q(t)}{C_k} = 0 .$$

$q(t)$  - заряд на конденсаторе. Период  $T_k = 2 \pi \sqrt{L_k C_k}$

В маятнике:



Суть объектов разная, но процессы описываются обычновенными ДУ. Введем понятия  $h_2 = L_k = ml_M$ ,  $h_1 = 0$ ,  $h_0 = 1/C_k = mgl$ . Если теперь положить состояние  $z(t)=q(t)=\theta(t)$ , то можно записать общий вид уравнения, описывающего функционирование данных объектов:

$$\frac{h_2 d^2 z(t)}{dt^2} + \frac{h_1 dz(t)}{dt} + h_0 z(t) = 0$$

Ноль символизирует собой отсутствие внешних воздействий X (включающих в себя заодно и V).

Использование D-схем позволяет формализовать процесс функционирования непрерывно-детерминированных систем и оценить их характеристики, применяя аналитический (иногда имитационный) подход, реализованный в виде соответствующего языка моделирования непрерывных систем с использованием аналоговых или гибридных средств вычислительной техники.

## 2. Интерактивные графические возможности в GPSS

Статистические таблицы используются для получения частотных распределений определенных аргументов, которыми могут быть некоторые СЧА (например, времени задержки транзакта в модели в целом или в отдельных ее частях; длин очередей; содержимого памяти и т. д.).

M TABLE A,B,C,D

A- табулир величина

B-верхняя граница левого интервала

C-ширина интервала

D-число интервалов

информация о гистограмме накапливается посредством попадания транзакта в блок:

TABULATE A

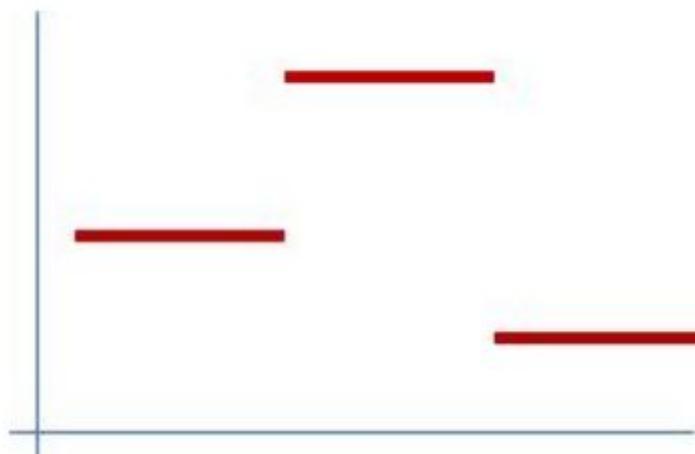
где A - имя гистограммы.

Функция – стандартный числовой атрибут, название и численная зависимость, обозначаемые в виде FN\$name. Описывается пользователем в виде численной зависимости от другого СЧА. Два основных типа – дискретные и непрерывные.

Пример: хранилище S с занятой памятью SMemory. Дискретная функция:

Func1 FUNCTION S\$Mem, D3 ;дискретная по трем точкам, ещё C, M, L, ..., S – стандартный числовой атрибут

5,12 / 9,20 / 12,6 ;дробная часть различается точкой



до 5 будет идти на уровне 12, до 9 на 20, до 12 на 6 (кусочная). Функция неявно продлевается влево и вправо если пользователь введёт соответствующий аргумент. На граничных участках берётся «всё ещё» значение (в 5 – 12).

В случае непрерывной – не набор горизонтальных отрезков, а ломаная.

4.5, 12.1 / 5.5, 10.5 / 7, 14 / 9, 15.3



Если задается много точек, то новая точка начинается со следующей строки, причем знак слэша в конце не ставится. Во всех незаданных участках (влево и вправо) функция продлевается горизонтально.

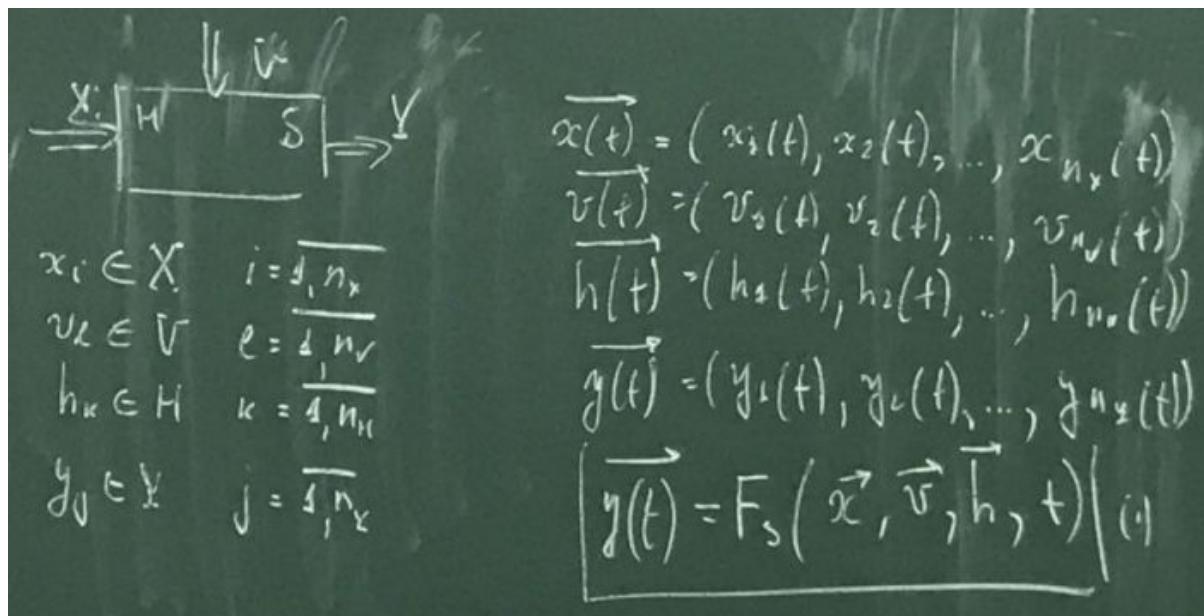
Для обращения к функции: FN\$Func1

# Билет 6

## 1. Типовые математические схемы для сложных систем

Модель объекта моделирования можно представить в виде множества величин, описывающих процесс функционирования реальной системы и образующих следующие подмножества:

2. совокупность входных воздействий  $x$
3. совокупность воздействий внешней среды  $v$
4. совокупность внутренних собственных параметров  $h$
5. совокупность выходных характеристик системы  $y$  (под схемкой множества)



В общем случае эти множества не пересекаются и содержат параметры стохастического и детерминированного характера.

При моделировании функционирования системы  $x$ ,  $v$  и  $h$  являются независимыми (экзогенными) переменными, которые в векторной форме могут быть записаны в следующей форме, а выходные характеристики являются эндогенными ( зависимыми ). Процесс функционирования описывается в итоге оператором  $F_s$ , который преобразует независимые переменные в зависимые. (справа сверху от схемы)

Выражение 1 описывает закон функционирования системы, в общем случае закон задается в виде функции, функционала, логических условий, в алгоритмическом или табличном виде.

Алгоритм -- это метод получения  $y$  с учетом  $x, v, h$ . Состояние -- свойства системы в конкретный момент времени, характеризуются некоторым вектором.

Если рассматривать процесс функционирования системы как последовательную смену состояний, то состояния могут быть интерпретированы как координаты точки в  $k$ -мерном фазовом пространстве. Причем, каждой реализации процесса будет соответствовать некоторая фазовая траектория.

Совокупность всех возможных состояний называется пространством состояний процесса моделирования.

В общем случае, время может быть непрерывным или дискретным (квантованным на интервале  $[t_0, t]$ ).

Под математической моделью реальной системы будем понимать конечное множество переменных  $x, v, h$  вместе с математическими связями между ними и характеристиками  $y(t)$ .

Практика моделирования: на первоначальных этапах формализации объектов используются т.н. математические типовые схемы, к которым относятся:

Процесс функционирования	Типовая математическая схема	Обозначения
непрерывно-детерминированный подход	дифференциальные уравнения	D
дискретно-детерминированный	конечные автоматы	F
дискретно-стохастический процесс	вероятностные автоматы	P
непрерывно-стохастический	системы массового обслуживания	Q
универсальный подход	агрегативные системы	A

## 2. Методика отладки в GPSS

Для пошагового выполнения модели с целью ее отладки можно воспользоваться командой STEP (выполнить шаг). Операнд в поле А команды задает количество входов активного транзакта в блоки, которое производится при каждом выполнении команды. Обычно этот операнд равен 1, и каждое выполнение команды STEP приводит к продвижению активного транзакта к следующему блоку. Отладку с использованием команды STEP удобно проводить, находясь в окне блоков. Для продолжения моделирования после прерывания следует ввести в командную строку команду CONTINUE (продолжить). Подробнее ниже. GPSS имеет в своем составе развитые средства отладки ИМ, доступ к которым осуществляется из пункта главного меню «Window → Simulation Window». GPSS реализует пошаговую отладку модели с одновременным отображением процесса перемещения транзактов между блоками ИМ в окне «BLOCK ENTITIES». Для этого в главном меню необходимо выбрать пункт «Window → Simulation Window → Block Window». Для управления процессом моделирования в панели инструментов окна «BLOCK ENTITIES» предусмотрены кнопки «Continue», «Halt» и «Step».

## Билет 7

### 1. Основные требования при разработке модели и аппаратной реализации

Суть машинного моделирования состоит в проведении на компьютере эксперимента с моделью, которая представляет собой, как правило, программный комплекс, описывающий формальным или алгоритмически поведение элементов системы в процессе ее функционирования.

**Функционирование элементов** — поведение в их взаимодействии друг с другом и внешней средой.

Основные требования к модели

1. Полнота модели. Должна предоставлять пользователю возможность получения необходимого набора характеристик, оценок системы с требуемой точностью и достоверностью.
2. Гибкость модели. Должна давать возможность воспроизведения различных ситуаций при варьировании структуры, алгоритмов и параметров модели. Структура должна быть блочной.
3. Компьютерная реализация модели должна соответствовать имеющимся тех средствам.

## 2. Визуальные средства отображения всякого о модели в процессе работы программы в gpss (или что-то похожее)

Для наблюдения за процессом моделирования и действием на него команд на этапе тестирования и верификации используются делять графических окон. Окна подразделяются по типам объектов:

- блоки
- выражения
- устройства
- логические ключи
- матрица
- график
- очереди
- ячейки
- памяти
- таблоицы

Окно блоки показывает выход транзактов в блоки. Оно позволяет с помощью мыши или клавиатуры устанавливать и удалять контрольные точки и визуально отслеживать передвижение транзактов.

Окно выражения предназначено для наблюдения за изменениями любого количества Plus-выражений.

С помощью окна график одновременно можно наблюдать любое количество многоцветных графиков.

Окно таблица представляет собой динамическую гистограмму, полезную для наблюдения за сбором данных, поиска выбросов и оценки сходимости к порождающему вероятностному распределению.

Окна блоки, устройства, логические ключ, очереди, ячейки, памяти имеют подробный обзор и общий обзор. Открываются они всегда с подробным образом.

## Билет 8

### 1. Основные этапы моделирования сложных систем

1. Построение концептуальной модели и её формализация
2. Алгоритмизация и компьютерная реализация модели.
3. Получение и интерпретация результатов моделирования.

Первый этап -- формирование модели и построение формальной схемы.

Переход от содержательного описания к математической модели.

1. Проведение границы между системой и внешней средой
2. Выделение основных составляющих функционирования системы
3. Исключение из рассмотрения некоторых второстепенных элементов

4. Группировка блоков в группы: имитаторов событий внешних воздействий, модели процесса функционирования и вспомогательных блоков
5. Разбитие ПФ на подпроцессы до достижения элементарности построения модели

Второй этап -- этап алгоритмизации и компьютерной реализации.

1. Разработка схемы алгоритма
2. Разработка схемы программы
3. Выбор технических средств для реализации
4. Процесс программирования и отладки
5. Тестирование достоверности
6. Составление ТД

Третий этап -- этап проведения рабочих расчетов по готовой программе.

1. Планирование эксперимента с моделью
  2. Проведение собственных расчетов (контрольная калибровка модели)
  3. Статистическая обработка результатов расчетов
  4. Интерпретация результатов моделирования
  5. Составление ТД
- ## 2. Способ моделирования СМО в GPSS

```

QUE GENERATE (Uniform(1,5,10)),,1000 ; генерируется 1000 заявок равномерным распределением [5, 10]
QUEUE STAND

-----Захват обработчика и заявки из очереди
SEIZE PROCUNIT
DEPART STAND
ADVANCE (Exponential(1,0,0.1)),, ; обработка по экспоненциальному распределению, бета = 1/lmb = 1/5 = 0.2
RELEASE PROCUNIT
TRANSFER .2,,QUE ; с вероятностью .2 заявка вернется к метке QUE

TERMINATE 1

RESET
START 1000

```

## Билет 10

### 1. Структурный уровень проектирования

Выделяют следующие уровни проектирования:

1. Структурный. В качестве типовых матсхем - системы массового обслуживания, А-схемы, сети Петри, вероятностные автоматы
2. Функционально-логический уровень проектирования

- a. подуровень регистровых передач. Элемент моделирования - регистры, сумматоры; всё разделяется на комбинационные и последовательные схемы.
  - b. логический уровень. Моделируем работы ЛЭ, когда они синтезированы на уровне одной кремниевой подложки. Используются цифровые автоматы, Ф-схемы
3. Конструкторский
- a. ОДУ, ДУ в частных производных. Характерен тем, что решаются в основном задачи отвода тепла, вентиляции, оптимального размещения интегральных схем (оптимизация длины проводников) либо вопросы технологичности многослойной печатной платы.

### **Моделирование на системном уровне**

При моделировании новой или модернизации действующей ВС и сетей необходимо предварительно оценить эффективность их функционирования с учетом различных вариантов структурной организации. Эти варианты могут отличаться составом и характеристиками модулей (моделей устройств), структурой межмодульных связей, режимами работы, алгоритмами управления и т.д. Именно в таких случаях используются модели ВС.

**Вычислительное Средство** – комплекс аппаратных и программных средств, которые в совокупности выполняют определенные обработочные функции.

**Операционная Система** – набор ручных и автоматических процедур, которые позволяют группе людей эффективно использовать вычислительную установку.

**Коллектив пользователей** – сообщество таких людей, которые используют ВС для удовлетворения своих нужд по обработке информации.

Входные сигналы (программы, данные, команды), которые создаются коллективом пользователей, называются **рабочей нагрузкой**.



**Индекс производительности** – описатель, который используется для представления производительности системы. Различают количественные и качественные индексы производительности.

Качественные:

- легкость использования системы
- мощность системы команд

Количественные:

- пропускная способность – объем информации, обрабатываемый в единицу времени.
- время ответа (реакции) – время между предъявлением системе входных данных и появлением соответствующей выходной информации.
- коэффициент использования оборудования – отношение времени использования указанной части оборудования в течение заданного интервала времени к длительности этого интервала.

**Концептуальная модель** ВС включает сведение о выходных и конструктивных параметрах системы, её структуре, особенностях работы каждого элемента и ресурса, постановка прикладных задач, определение цели моделирования.

Основные задачи:

2. Определение принципов организации ВС.
3. Выбор архитектуры, уточнение функций ВС и их разделение на подфункции, реализация аппаратным и программным путем.
4. Разработка структурной схемы – определение состава устройств и способов их взаимодействий.
5. Определение требований к выходным параметрам устройств и формирования технического задания на разработку устройств для функционально-логического уровня проектирования.

## 6. Table, tabulate

NAME TABLE A,B,C,D

- NAME - Entity Label for this entity. Required. The field must be Name. The length of a Table name is limited to 32 characters.
- A - Table argument. Optional. The data item whose frequency distribution is to be tabulated. The operand must be Name, Number, String, ParenthesizedExpression, or SNA. Ignored by ANOVA, but must be specified when used by TABULATE Blocks.
- B - Upper limit of first frequency class. The maximum argument which causes the first frequency class to be updated. Required. The operand must be Number or String.
- C - Size of frequency classes. The difference between the upper limit and lower limit of each frequency class. Required. The operand must be Number or String.
- D - Number of frequency classes. Required. The operand must be PosInteger.

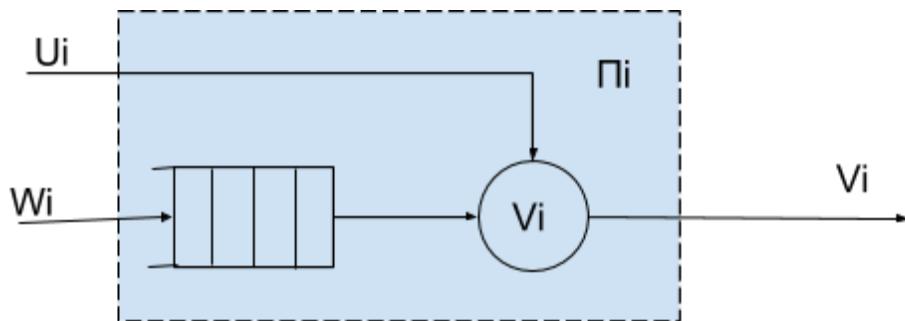
TABULATE Sales

- When a Transaction enters this TABULATE Block, the Table Entity named Sales is found. Sales must have been defined in a TABLE Command. Then the statistics associated with the table are updated with no weighting.

## Билет 11

### 1. Q-схемы

Q-схема реализует непрерывно-стохастический подход, используется для моделирования систем массового обслуживания.



Самым главным понятием является понятие потока заявок.

Для СМО характерно случайное появление заявок на обслуживание и случайное же распределение времени на обработку заявки.

В любом акте обслуживания можно выделить две составляющие: ожидание обслуживания и процесс обслуживания.

Поток событий -- последовательность событий, происходящих одно за другим в какие-то случайные моменты времени.

Поток событий однороден, если он характеризуется только моментами поступлений этих событий и задается некоторой последовательностью  $t: \{t_n\} = t_0 < t_1 < t_2 < \dots < t_n$ . Поток неоднороден, если он задается еще и набором признаков события  $f: \{t_n, f_n\}$ .

Если интервалы между сообщениями случайны и независимы, то мы имеем поток с ограниченным последействием.

Поток событий ординарен, если вероятность того, что на момент времени  $t$  придется 2 и более события, пренебрежительно мала по сравнению с вероятностью иного исхода.

ПС стационарен, если вероятность появления того или иного числа событий на интервале времени зависит лишь от длины этого интервала, а не от того, где этот самый интервал был взят.

Для стационарного потока интенсивность потока равна среднему числу событий в единицу времени.

Для ординарного потока число сообщений на участке  $dt$ , примыкающему к моменту времени  $t$ , будет равно  $P(>1)[t, dt] + P(1)[t, dt] \sim P(1)[t, dt]$ . Соответственно, предел при  $t \rightarrow 0$  от  $P(1)/dt$  даёт нам  $\lambda(t)$  -- интенсивность ординарного потока.

Процесс функционирования прибора можно представить как изменение состояний его элементов во времени. Таким образом, вектор состояний может

быть либо вектором накопителя, либо вектором канала:  $Z_i = (Z_i^N, Z_i^K)$ . ...  
Что касается канала — то он находится в двух состояниях — канал свободен и канал занят.

В практике моделирования элементарной Q-схемы объединяются, при этом, если каналы обслуживания различных приборов соединены параллельно, то имеет место многоканальное обслуживание. А если последовательно, то многофазное. Следовательно, для задания QS необходимо использовать оператор сопряжения R, отражающий взаимосвязь элементов структуры между собой. Различают разомкнутые и замкнутые Q-схемы.

Параметрами для Q-схемы будут:

- количество фаз
- количество каналов в каждой фазе
- количество накопителей в каждой фазе
- емкость накопителя (если она равна нулю, то система с потерями, иначе — система с ожиданием.)

Еще кое что:

- необходимо описать алгоритм её функционирования, который определяет набор правил поведения заявок в системе.
- неоднородность заявок, отражающая состояние системы в то или иное время учитывается с помощью введения классов приоритетов.

Весь набор алгоритмов поведения заявок в Q-схеме можно передать в виде оператора Q:

$$Q = (U, W, H, Z, R, A)$$

где

2. W - входные потоки
3. U - поток обслуживания
4. R - оператор сопряжения элементов
5. H - собственные параметры
6. Z - состояния системы
7. A - оператор алгоритмов поведения и обслуживания заявок.

## 8. Блоки SEIZE и RELEASE

Транзакты в GPSS перемещаются от блока к блоку. Если в какой-то момент транзакт занимает одноканальное устройство (ОКУ), то он должен войти в соответствующий блок, описывающий ОКУ.

Для SEIZE:

1. Если ОКУ уже используют, то транзакт встать в очередь и должен ждать, пока он освободится.
2. Если ОКУ не используется, то транзакт войдёт в блок, а статус ОКУ изменится на «занят».

RELEASE соответственно переводит статус ОКУ в состояние «свободен». В качестве параметров обоим блокам передаётся имя ОКУ.

## Билет 13

### 1. марковские случайные процессы. Многоканальная СМО с отказами

Все уравнения Колмогорова соответствуют следующим правилам:

1. В левой части находится производная вероятности состояния по времени, а в правой части столько членов, сколько и стрелок, связанных с этим состоянием.
2. В правой части для исходящих стрелок используется знак минус, а для входящих -- плюс.
3. Каждый член равен произведению плотности вероятности перехода интенсивности, умноженную на вероятность того состояния, из которого выходит стрелка.

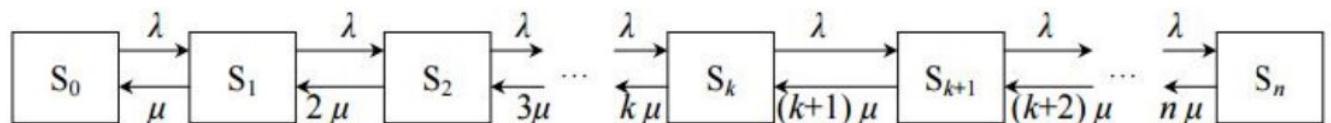
Состояние системы будем интегрировать по числу занятых каналов.

$S_0$  – все каналы свободны.

$S_1$  - занят один канал, остальные свободны.

$S_k$ - занято  $k$  каналов, остальные свободны.

$S_n$  - все  $S_n$  каналов заняты.



Проставим интенсивности:

1. стрелка слева направо переводит поток заявок с интенсивностью  $\lambda$
2. стрелка справа налево -- с интенсивностью  $\mu^i$ , где  $i$  -- номер текущего состояния.

По уравнениям Колмогорова получаем:

$$p'_0 = -\lambda * p_0 + \mu * p_1$$

$$p'_1 = -\lambda * p_1 - \mu * p_1 + \lambda * p_0 + 2 * \mu * p_2$$

$$p'_k = -\lambda * p_k - k * \mu * p_k + \lambda * p_{k-1} + (k+1) * \mu * p_{k+1}$$

$$p'_n = -n * \mu * p_n + \lambda * p_{n-1}$$

Предельные вероятности состояний  $p_0, \dots, p_n$  характеризуют установившийся режим работы СМО. Следовательно, все левые части у уравнений равны 0.

- Предельные вероятности состояний  $p_0, \dots, p_n$  характеризуют установившийся режим работы СМО. Это значит, что левые части всех уравнений равны 0.

$$p_0 = \frac{1}{1 + \frac{(\frac{\lambda}{\mu})}{1!} + \dots + \frac{(\frac{\lambda}{\mu})^n}{n!}}$$
$$p_k = \frac{(\frac{\lambda}{\mu})^k}{k!} * p_0$$
$$\frac{\lambda}{\mu} = \rho$$

- Зная все вероятности  $p_0, \dots, p_n$  можно найти характеристики СМО:

1) Вероятность отказов:  $p_{отк} = p_n = \frac{\rho^n}{n!} * p_0$

2) Вероятность принятия:  $q = 1 - p_{отк}$

3) Среднее число обработанных заявок:  $= \lambda * q = \lambda(1 - p_{отк})$

4) Производительность:  $\lambda = \frac{1}{\text{время обработки}}$

5) Если время входа мало по сравнению с временем обработки

$$\text{время обработки} = \frac{1}{\mu}, \mu - \text{технические характеристики компьютера}$$

## 2. Блоки синхронизации транзактов

MATCH A -- одноместная очередь, один приходит, а тот что перед ним был -- выходит.

ASSEMBLE A -- число A транзактов объединяется в ансамбль, и в результате после набора числа A транзактов выходит первый его транзакт, а остальные уничтожаются.

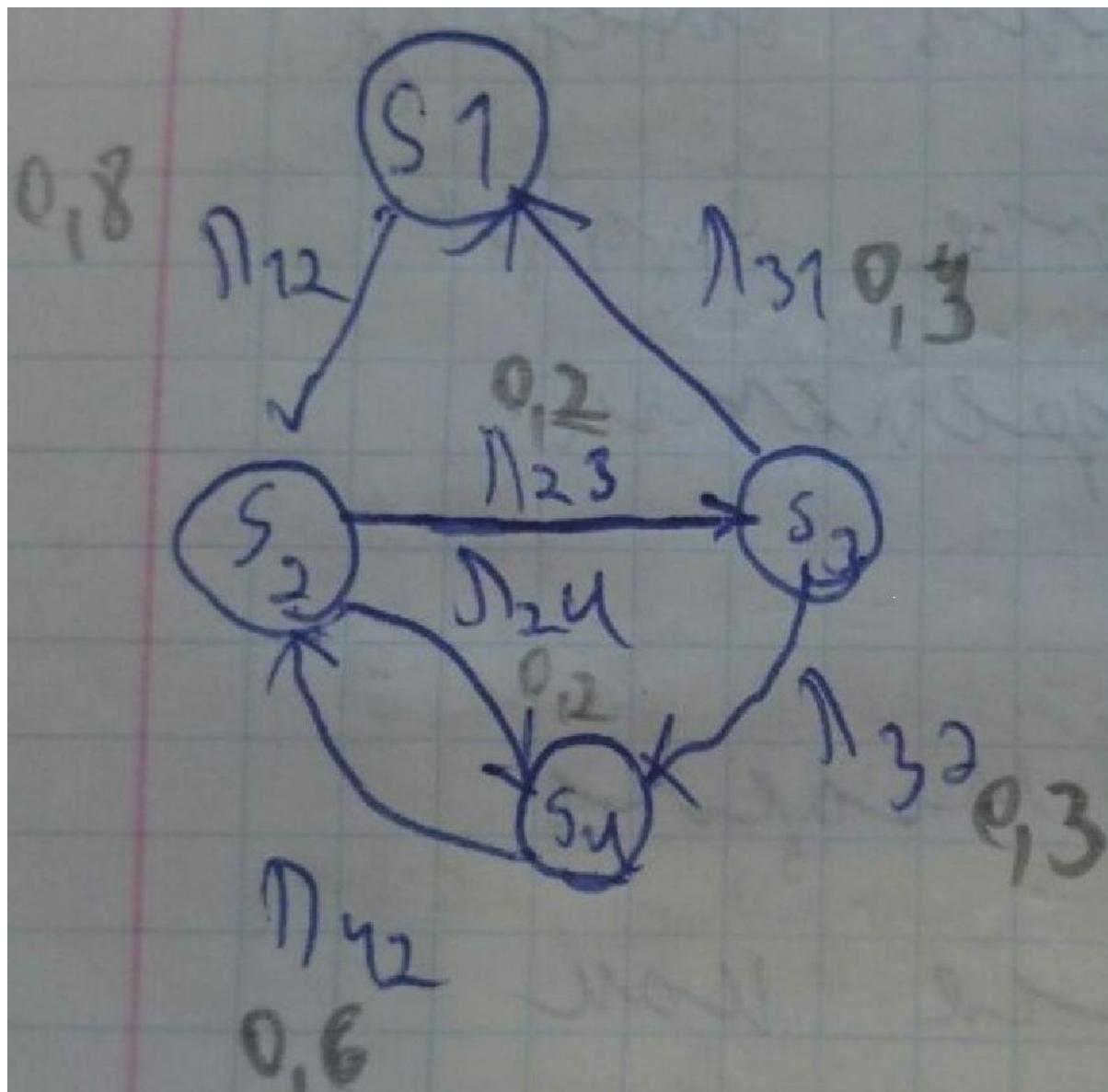
GATHER A -- аналогично, только без уничтожения транзактов

# Билет 14

## 1. Марковские процессы. Уравнения Колмогорова.

Все уравнения Колмогорова соответствуют следующим правилам:

1. В левой части находится производная вероятности состояния по времени, а в правой части столько членов, сколько и стрелок, связанных с этим состоянием.
2. В правой части для исходящих стрелок используется знак минус, а для входящих -- плюс.
3. Каждый член равен произведению плотности вероятности перехода интенсивности, умноженную на вероятность того состояния, из которого выходит стрелка.



Пусть  $\lambda$  -- плотность вероятности. Найдём вероятность нахождения системы в состоянии  $S_1 = P_1(t)$ .

Придадим  $t$  малое приращение  $dt$  и найдем вероятность того, что в момент времени  $t+dt$  система будет находиться в состоянии  $S_1$ .

Вероятность того, что система не вышла из  $S_1$ :  $p_1 \times (1 - \lambda_{12} \times dt)$  Вероятность того, что система перешла из  $S_3$  в  $S_1$ :  $p_3(t) \times \lambda_{31} \times dt$

Тогда в итоге имеем: [вероятность первого] + [вероятность второго] =  $p_1(t+dt)$

- Первое состояние:

$$\lim_{\Delta t \rightarrow 0} \frac{p_1(t + \Delta t) - p_1(t)}{\Delta t} = -\lambda_{12} * p_1(t) + \lambda_{31} * p_3(t)$$

$$p'_1(t) = -\lambda_{12} * p_1(t) + \lambda_{31} * p_3(t)$$

- Второе состояние:

$$p_2(t + \Delta t) = p_2(t) * (1 - \lambda_{24} * \Delta t - \lambda_{23} * \Delta t) + p_1(t) * \lambda_{12} * \Delta t + p_4(t) * \lambda_{42} * \Delta t$$

$$\lim_{\Delta t \rightarrow 0} \frac{p_2(t + \Delta t) - p_2(t)}{\Delta t} = -\lambda_{23} * p_2(t) - \lambda_{24} * p_2(t) + \lambda_{12} * p_1(t) + \lambda_{42} * p_4(t)$$

$$p'_2(t) = -\lambda_{23} * p_2(t) - \lambda_{24} * p_2(t) + \lambda_{12} * p_1(t) + \lambda_{42} * p_4(t)$$

(Аналогично, для третьего и четвертого)

Интегрируя пределы, можно получить исковые вероятности как функции от времени. Начальные условия берутся в зависимости от начального состояния системы. Учитывается условие нормировки.

## 2. Блоки аппаратной категории GPSS: PREEMPT и RETURN.

PREEMPT A B C D E -- фиксирует использование устройства на более высоком уровне чем SEIZE, приостанавливает обслуживание транзакта, захватившего устройство ранее, дает возможность прерванному транзакту захватить устройство после того, как закончится обслуживание прервавшего транзакта. Если при выполнении PREEMPT выяснилось, что устройство обслуживает прерывание, то блок не выполняется и транзакт задерживается до тех пор, пока не освободится устройство.

RETURN A -- сбрасывает владение над устройством и идет дальше.

## Билет 15

### 1. Способы получения последовательности случайных чисел

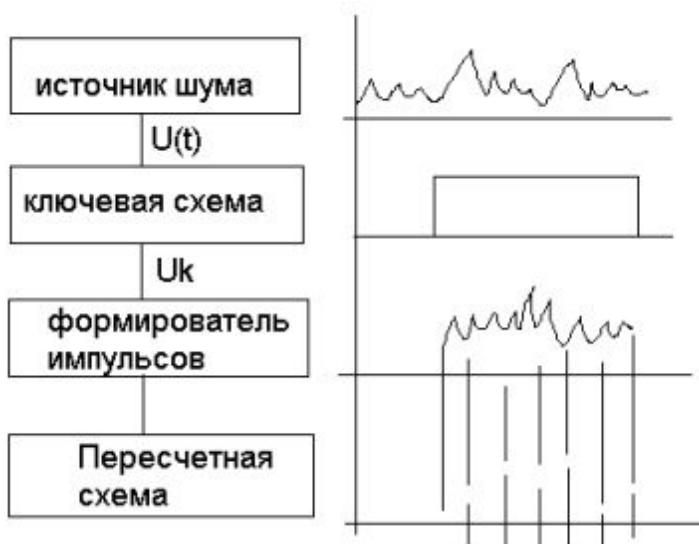
При имитационном моделировании системы одним из основных вопросов является стохастических воздействий. Для этого метода характерно большое число операций со случайными числами или величинами и зависимость результатов от качества последовательностей случайных чисел. На практике используется 3 основных способа:

1. Аппаратный (физический)
2. Табличный (файловый)
3. Алгоритмический (программный)

**Аппаратный способ.** Аппаратный представляет из себя шум.

Случайные числа вырабатываются случайной электронной приставкой (генератор случайных чисел), служащей, как правило, в качестве одного из внешних устройств. Реализация этого способа обычно не требует дополнительных вычислений, а необходима только операция обращения к ВУ. В качестве физического эффекта, лежащего в основе таких генераторов чаще всего используют шумы в электронных приборах.

Т.е. необходимо: источник шума, ключевая схема, формирователь импульсов, пересчетная (см. рис.)



**Табличная схема.** Случайные числа оформляются в виде таблицы и помещаются во внешнюю или оперативную память.

**Алгоритмический способ.** Способ основан на формировании случайных чисел с помощью специальных алгоритмов.

## 2. Блоки изменения параметров транзактов в GPSS

ASSIGN A,B,C

- номер параметра
- значение для добавления или замены
- номер модификатора-функции

ALTER -- изменяет приоритет или атрибуты сообщений данной группы.

MARK A - изменяет время рождения транзакта

## Билет 17

### 1. Моделирование потоков событий

Самым главным понятием является понятие потока заявок.

Для СМО характерно случайное появление заявок на обслуживание и случайное же распределение времени на обработку заявки.

В любом акте обслуживания можно выделить две составляющие: ожидание обслуживания и процесс обслуживания.

Поток событий -- последовательность событий, происходящих одно за другим в какие-то случайные моменты времени.

Поток событий однороден, если он характеризуется только моментами поступлений этих событий и задается некоторой последовательностью  $t$ :  $\{t_n\} = t_0 < t_1 < t_2 < \dots < t_n$ . Поток неоднороден, если он задается еще и набором признаков события  $f$ :  $\{t_n, f_n\}$ .

Если интервалы между сообщениями случайны и независимы, то мы имеем поток с ограниченным последействием.

Поток событий ординарен, если вероятность того, что на момент времени  $t$  придется 2 и более события, пренебрежительно мала по сравнению с вероятностью иного исхода.

ПС стационарен, если вероятность появления того или иного числа событий на интервале времени зависит лишь от длины этого интервала, а не от того, где этот самый интервал был взят.

Для стационарного потока интенсивность потока равна среднему числу событий в единицу времени.

Для ординарного потока число сообщений на участке  $dt$ , примыкающему к моменту времени  $t$ , будет равно  $P(>1)[t, dt] + P(1)[t, dt] \sim P(1)[t, dt]$ . Соответственно, предел при  $t \rightarrow 0$  от  $P(1)/dt$  даёт нам  $\lambda(t)$  -- интенсивность ординарного потока.

Процесс функционирования прибора можно представить как изменение состояний его элементов во времени. Таким образом, вектор состояний может быть либо вектором накопителя, либо вектором канала:  $Z_i = (Z_i^N, Z_i^K)$ . ... Что касается канала — то он находится в двух состояниях — канал свободен и канал занят.

## 2. Блоки SPLIT/ASSEMBLE

ASSEMBLE A -- число A транзактов объединяется в ансамбль, и в результате после набора числа A транзактов выходит первый его транзакт, а остальные уничтожаются.

SPLIT A,B,C,D -- создат А копий транзакта

- входящих в блок В
- записываемых в параметры С
- с числом параметров D

## Билет 23

### 1. Событийный принцип протяжки модельного времени

**Событийный принцип.** Характерное свойство моделируемых систем обработки информации – состояние отдельных устройств изменяется в дискретные моменты времени, совпадающие с: моментами времени поступления сообщений в систему; временем окончания решения задач; временем возникновение аварийных сигналов и т.п. Моделирование и продвижение текущего времени в системе удобно проводить с использованием событийного метода. Состояние всех блоков имитационной модели анализируется лишь в момент появления какого-либо события.

Моменты наступления следующих событий определяются минимальным значением из списка будущих событий. Список представляет собой совокупность моментов ближайшего изменения состояния каждого из блоков системы.

Достоинство: не пропустим ни одного события

Недостаток: при большом количестве событий (сложная система) список необходимо просматривать постоянно (можно держать отсортированным).

### 2. Классификация блоков GPSS

Блоки, используются для описания функций и управляют движением транзактов. У каждого блока имеется 2 стандартных числовых атрибута(СЧА):

Wn–счетчик входа в блок. Как правило номер транзакта, входящего в данный блок.

Nn–Общий счетчик транзактов, поступивших в блок с начального момента моделирования.

Практически все изменения состояния модели возникают в результате поступления транзактов в соответствующие блоки. После выполнения блока транзакт либо движется к следующему блоку, либо задерживается, либо уничтожается. Блоки измененият атрибуты транзакта.

1. Блоки осуществляющие модификацию атрибутов транзакта
  - Временная задержка ADVANCE
  - Генерация и уничтожение транзактов GENERATE TERMINATE SPLIT ASSEMBLE
  - Синхронизация движения нескольких транзактов MATCH GATHER

- Изменение параметров транзакции ASSIGN INDEX MARK
  - Изменение приоритета PRIORITY
2. Блоки, изменяющие последовательность передвижения транзактов, т.е. блоки передачи управления.

TRANSFER, LOOP, TEST, GATE

1. Блоки, связанные с группирующей категорией.

JOIN, REMOVE, EXEMINE, SCAN, ALTER

1. Блоки, сохраняющие значения для дальнейшего использования.

SAVEVALUE, MSAVEVALUE

1. Блоки, организующие использование объектов аппаратного устройства.

SEIZE - RELEASE, PREEMPT - RETURN, F\S AVAIL – UNAVAIL, ENTER – LEAVE, LOGIC

1. Блоки, обеспечивающие получение статистических результатов. QUEUE, DEPART, TABULATE, TABLE
2. Специальные блоки

HELP, TRACE, UNTRACE, PRINT, ...?

1. Блоки для организации цепей

LINK, UNLINK

9. Вспомогательные блоки

LOAD, SAVE

Каждый блок определяется с помощью отдельной команды. В общем случае: сначала идет нумерация, затем обязательно поле метки, затем поле операции, поле operandов, затем, если необходимо, комментарий (через ;). Т.е.:  
<Нумерация><Метка><Оператор><Операнды><Комментарии>

# Билет 24

## 1. Классификация языков имитационного моделирования



Непрерывное моделирование сводится к представлению ДУ, с помощью которой устанавливается связь между входной и выходной функциями.

GASP -- комбинированный, на основе Fortran. Может описать события, зависящие от времени и от состояния.

Последняя группа используется в основном для имитационного моделирования, при этом используются разные способы описания динамического поведения системы.

## 2. Язык GPSS. Построение и основные принципы

General Purpose Simulation System - язык общечелевой системы моделирования.

Возможности:

- Многозадачность
- Использование вирт. памяти
- Интерактивность
- GUI
- Визуализация процесса моделирования

Для GPSS моделью сложной дискретной системы является описание её элементов и логических правил их взаимодействия в процессе функционирования. Можно выделить конечный набор абстрактных элементов "объектов", причем набор логических правил также ограничен и может быть описан небольшим числом операций.

В процессе моделирования объекты взаимодействуют друг с другом путем изменения атрибутов и преобразования их значений - в результате "событий".

Транзакты моделируют прохождение по системе соответствующих единиц потока. Такое движение может быть разбито на цепь элементарных событий, происходящих в определенные моменты времени.

Основная задача симулятора - выявление моментов наступления этих событий и выполнение определенных действий при их наступлении.

## Билет 25

### 1. Алгоритмический способ получения последовательности псевдослучайных чисел

Алгоритмы:

1. Выделение значения дробной части многочлена первой степени ( $Y_n = A * n + B$ )
2. фон Нейман. Каждое последующее число образуется возведением предыдущего в квадрат и отбрасыванием начальной и конечной цифр
3. Лимер  $g_{n+1} = (K * g_n + C) \bmod M$ . Наибольший период: при  $k = 3+8i$  или  $5+8i$  (комплексные числа)
4. Метод Фибоначчи на основе одноименных чисел и натуральной арифметике

Достоинства

- Однократная проверка
- Многократное воспроизведение последовательности
- Места в RAM не занимает
- Не требуется внешнее устройство

Недостатки

1. Запас чисел ограничен периодом
2. Затраты машинного времени

### 2. Язык GPSS. Блоки. Объекты языка

Объекты языка подразделяются на 7 категорий и 14 типов:

Категория	Тип
Динамическая	Транзакт
Операционная	Блоки
Аппаратная	Устройства памяти, ключи, (?) одноканальные устройства
Вычислительная	Переменные, арифметические\логические функции, (?) генераторы случайных чисел
Статистическая	Очереди, таблицы
Запоминающие	Ячейки, матрицы ячеек
Группирующие	Списки, группы

Динамические объекты - транзакты (сообщения), которые представляют собой единицы исследуемых потоков и производят ряд определенных действий, продвигаясь

по фиксированной структуре, представляющей собой совокупность объектов других категорий.

Операционные объекты задают логику функционирования системы и определяют маршрут движения транзактов между объектами аппаратной категории.

Аппаратные объекты – абстрактные элементы, на которых может быть декомпозировано оборудование реальной системы. Воздействуя на эти объекты, транзакты могут изменять их состояния и влиять на движение других транзактов.

Вычислительная категория – служит для описания таких ситуаций в процессе моделирования, когда связи между компонентами моделируемой системы наиболее просто и компактно выражаются в виде математических соотношений.

Статистический объект - очередь\таблица, служащая для оценки поведения системы.

Объекты запоминающей категории обеспечивают обращения к сохраняемым значениям. Ячейки сохраняемых величин и матрицы ячеек сохраняемых величин используются для сохранения некоторой числовой информации. Любой активный транзакт может произвести запись информации в эти объекты. Впоследствии записанную в эти объекты информацию может считать любой транзакт.

Группирующая категория состоит из трёх типов объектов: числовых групп, групп транзактов и списков.

## Билет 26

### 1. Типы СМО для моделирования сложных систем

Признаки классификации:

1. закон распределения входного потока заявок
2. числа обслуживающих приборов
3. закон распределения времени обслуживания в обслуживающих приборах
4. число мест в очереди
5. дисциплина обслуживания

Для обозначения СМО принята система кодирования A|B|C|D|E, где

- А–закон распределения интервалов времени между поступлениями заявок.

Наиболее часто используемое обозначение:

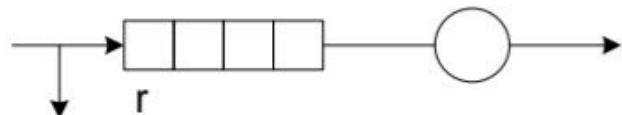
- М-экспоненциальное
  - Е–эрлангово
  - Н–гипер-экспоненциальное
  - (?)–гамма
  - D–детерминированное
  - G–произвольное
- В–закон распределения времени обслуживания в приборах. Приняты те же обозначения, как и для интервалов между появлением заявок.

- С – число обслуживающих приборов. Для одноканальной – 1, для многоканальной – l.
  - D – число мест в очереди.
- n или r – конечно  
<опущено> – неограниченно
- Е – дисциплина обслуживания. Наиболее часто используются следующие варианты дисциплины обслуживания: FIFO (может опускаться), LIFO, RANDOM
- Пример: M/M/1 – СМО с одним ОА, бесконечной очередью, экспоненциальными законами распределения времени между поступлениями заявок и времени обслуживания, дисциплина обслуживания FIFO.  
G/G/1 - Одноканальные системы с ожиданием:

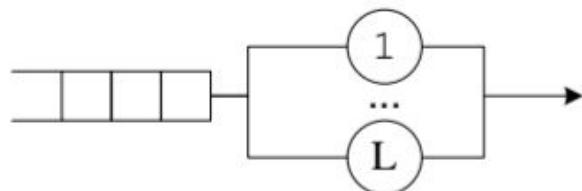
### **G/G/1** - Одноканальные системы с ожиданием:



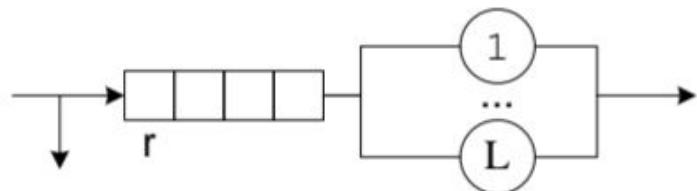
### **G/G/1/r** - Одноканальная система с потерями:



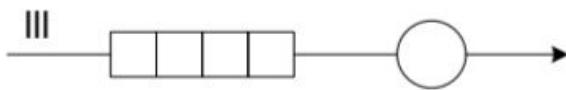
### **G/G/l** - Многоканальная система с ожиданием



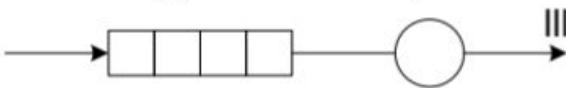
### **G/G/l/r** – Многоканальная система с потерями



**Gr/G/1** – одноканальная система с групповым поступлением заявок:



**G/Gr/1** – одноканальная система с групповым обслуживанием



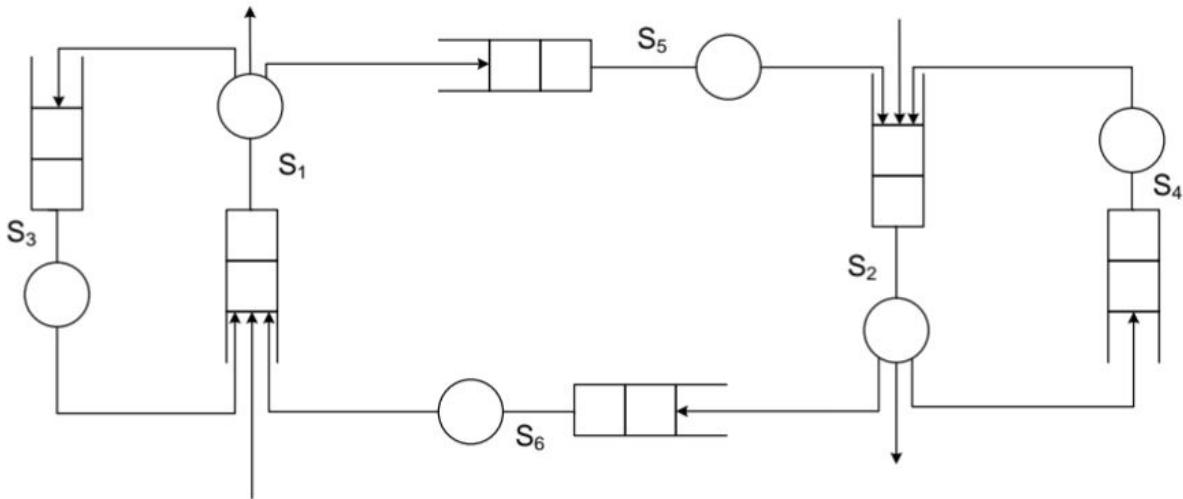
Для моделирования вычислительных систем и сетей наиболее часто используются следующие типы СМО:

1. Одноканальное СМО с ожиданием. Представляет собой один обслуживающий прибор с бесконечной очередью. Является наиболее распространенной при исследовании СДС. Формализует функционирование практически любого числа узла вычислительной сети.
2. Одноканальная СМО с потерями. Один обслуживающий прибор с конечным числом мест в очереди. Используется при моделировании каналов передачи в вычислительных сетях.
3. Многоканальные СМО с ожиданием. Представляют собой несколько параллельно работающих обслуживающих приборов с общей параллельной очередью. Используется при моделировании групп абонентских терминалов, работающих в диалоговом режиме.
4. Многоканальные СМО с потерями. Наиболее часто используются при моделировании работы каналов.
5. Одноканальные СМО с групповым поступлением заявок. Так же как и одноканальная СМО с групповым обслуживанием заявки используются для моделирования центров коммутации.

Вычислительные сети в целом могут быть исследованы с помощью сетей массового обслуживания.

Различают сети:

- 1) **Открытые.** Сеть массового обслуживания, состоящая из  $m$  узлов, причем хотя бы в один из узлов сети поступает извне входящий поток заявок и обязательно имеется сток заявок из сети. Для открытых сетей характерно то, что интенсивность поступления заявок в сеть не зависит от состояния сети, т.е. от числа уже поступивших. Такие сети используются как правило, для исследования функционирования вычислительной сети, работающей в неоперативном режиме



S<sub>1</sub>, S<sub>2</sub> – моделируют работу узлов коммутации

S<sub>3</sub>, S<sub>4</sub> – моделируют работу серверов

S<sub>5</sub>, S<sub>6</sub> – моделируют работу межузловых каналов

В сети циркулируют два потока заявок. Каждая заявка поступает на вход соответствующего узла коммутации, где определяется место её обработки. Затем заявка передается на «свой» сервер или по каналу связи на соседний сервер, где заявки обрабатываются. После чего возвращается к источнику и покидает сеть.

- 2) **Замкнутые** – называются сети МО с множеством узлов без источника и стока, в которой циркулирует постоянное число заявок.

Замкнутые сети МО используются для моделирования таких вычислительных систем, источниками информации для которых служат абонентские терминалы, работающие в диалоговом режиме. В этом случае каждая группа абонентских терминалов представляется в виде многоканальной системы МО с ожиданием и включается в состав устройств сети.

Различают простой и сложный режим диалога.

- При простом: абоненты не производят никаких действий кроме посылки заданий в вычислительную сеть и обдумывания полученного ответа.
- Схема (самостоятельно): группы абонентов, каналы связи с абонентами, узлы коммутации, серверы и каналы межузловой связи.

Абоненты с терминалов посыпают запросы, которые по каналам связи поступают на узлы коммутации. А оттуда на обработку на свой или соседний сервер.

- При сложном режиме диалога работа абонентов представляется в виде совокупности операций некоего процесса, называемого **технологическим**. Каждая операция технологического процесса, моделируется соответствующей системой массового обслуживания. Часть операций предусматривает обращение к вычислительной системе, а часть может и не обращаться.

- 3) Смешанные – называются сети МО в которой циркулирует несколько различных типов заявок (трафик). Причем относительно одних типов заявок сеть замкнута, а относительно других открыта. С помощью смешанных сетей МО моделируют такие вычислительные сети часть которых работает в диалоговом режиме, а часть в неоперативном. Причем для диалоговых

абонентов также различают простой и сложной режим работы. Так же смешанными сетями МО моделируются вычислительные системы в которых сервер дополнительно загружается задачами, решаемыми на фоне работы сети.

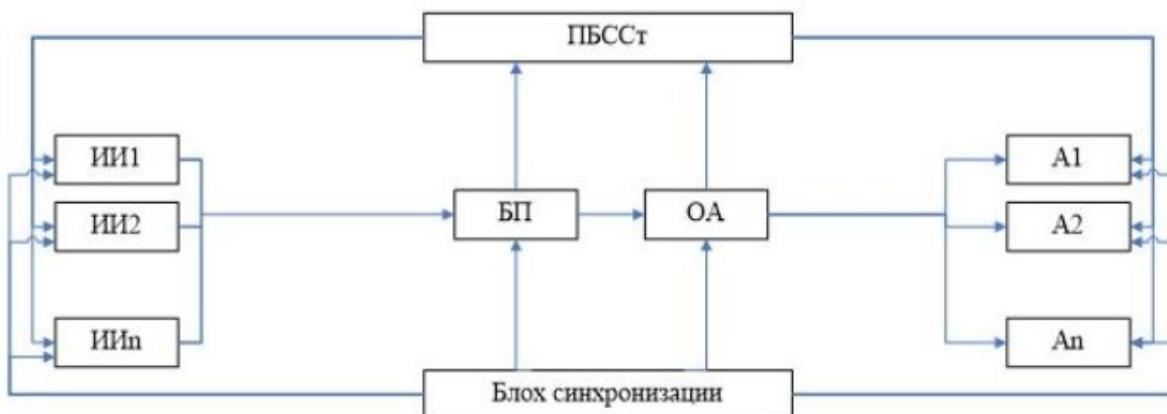
## 2. Отладка в GPSS

Для пошагового выполнения модели с целью ее отладки можно воспользоваться командой STEP (выполнить шаг). Операнд в поле А команды задает количество входов активного транзакта в блоки, которое производится при каждом выполнении команды. Обычно этот операнд равен 1, и каждое выполнение команды STEP приводит к продвижению активного транзакта к следующему блоку. Отладку с использованием команды STEP удобно проводить, находясь в окне блоков. Для продолжения моделирования после прерывания следует ввести в командную строку команду CONTINUE (продолжить). GPSS имеет в своем составе развитые средства отладки ИМ, доступ к которым осуществляется из пункта главного меню «Window → Simulation Window». GPSS реализует пошаговую отладку модели с одновременным отображением процесса перемещения транзактов между блоками ИМ в окне «BLOCK ENTITIES». Для этого в главном меню необходимо выбрать пункт «Window → Simulation Window → Block Window». Для управления процессом моделирования в панели инструментов окна «BLOCK ENTITIES» предусмотрены кнопки «Continue», «Halt» и «Step».

# Билет X1

## 1. Методика построения программной модели

Для разработки программной модели исходная система должна быть представлена как **стохастическая СМО**, так как информация от внешней среды поступает в случайные моменты времени, и в общем случае длительность обработки различных типов информации может быть различна. То есть, внешняя среда является **генератором сообщений**, а комплекс вычислительных устройств -- ОУ.



ИИ -- источники информации. БП -- буферная память, FIFO/LIFO-очередь сообщений. Вследствие различности времени обработки и появления сообщения возможно сложение данных, ожидающих обработки. А -- абоненты. ПБССт -- программный блок

сбора статистики. Блок синхронизации -- блок, отвечающий за управлением активацией тех или иных фрагментов системы.

## 2. Язык GPSS. Блоки создания и уничтожения транзактов

GENERATE A,B,C,D,E

- время между поступлением транзактов
- модификатор интервала а
- начальная задержка
- число порождаемых транзактов
- приоритет транзакта

TERMINATE A удаляет транзакты из системы

## Билет X2

### 1. Проверка адекватности и корректности модели

Проверка адекватности модели некоторой системы заключается в анализе её соразмерностей, а также равнозначности системы.

Адекватность часто нарушается из-за идеализации внешних условий и режимов функционирования, пренебрежением случайными факторами. Простейшей мерой адекватности является отклонение некой характеристики оригинала от модели.

Считают, что модель адекватна с системой, если вероятность того, что отклонение не превышает определенного значения, выше допустимой нормы.

Однако на деле нельзя использовать такой подход, так как отсутствует информация по выходным характеристикам объекта. Нужно проводить экспертный анализ разумности результатов моделирования.

Если по результатам проверки выделяется недопустимое несогласование модели и системы, возникает необходимость во внесении изменений.

#### Типы изменений

1. Глобальные -- методические ошибки в концепте или матем. модели.
2. Локальные -- уточнение алгоритмов
3. Параметрические (калибровочные)

## 2. Язык GPSS. QUEUE, RELEASE.

### QUEUE A,B

Этот блок осуществляет сбор статистики об очереди. Номер очереди в которую заносится транзакт задается в поле А. При записи нового транзакта в очередь определяется длина интервала времени, в течение которого длина очереди оставалась неизменной.

При входе транзакта в данный блок текущая длина очереди увеличивается на число единиц, указанное в поле В. Затем происходит сравнение с максимальной длиной очереди, достигнутой до этого момента времени. Если оно больше старого значения, то оно его заменяет. Кроме того, счетчик общего числа единиц прошедших через очередь увеличивается на тоже число единиц.

### RELEASE A

В GPSS элементами, которые требуют обслуживания, являются транзакты. Они перемещаются в модели от блока к блоку. Если в какой-то момент активности транзакт занимает ОКУ, то для этого он входит (или пытается войти) в соответствующий блок, описывающий это ОКУ. Блок должен обладать следующими свойствами:

- если ОКУ уже используют, транзакт не может войти в блок и должен ждать в очереди;
- если ОКУ не используют, транзакт может войти в блок, статус ОКУ изменяется на "занято".

Блок, обладающий этими свойствами, является блоком SEIZE (занято). Вход транзакта

в блок SEIZE моделирует занятие ОКУ. После обслуживания вход того же транзакта в другой блок моделирует освобождение ОКУ. Назначением этого блока является изменение состояния ранее занятого ОКУ с "занято" в "незанято". Этим блоком является блок RELEASE (освободить).

ДОП.ВОПРОС: ADVANCE с распределением Пуассона и  $\lambda = 3$   
ADVANCE A,B

Блок задает среднее время выполнения операций в моделируемой системе, а так же разброс времени относительно среднего. Задержка – целое число. Для задания времени пребывания в блоке ADVANCE пользователь указывает среднее время в поле А, а модификатор в поле В. Если поле задержки постоянно, то поле В может быть пустым. А если нулевое, то и поле А может отсутствовать. Модификаторы могут быть двух типов:

1. Модификатор «интервал», используется, когда время задержки транзакта распределено равномерно в некотором заданном интервале. Например:  
ADVANCE 5,2 (т.е. интервал от 3 до 7)
2. Модификатор «функция», когда интервал отличается от равномерного и приходится с помощью этого блока находить данное время. Указываем среднюю величину, а дальше функцию, на значение которой должна быть умножена данная величина.

## Билет X3

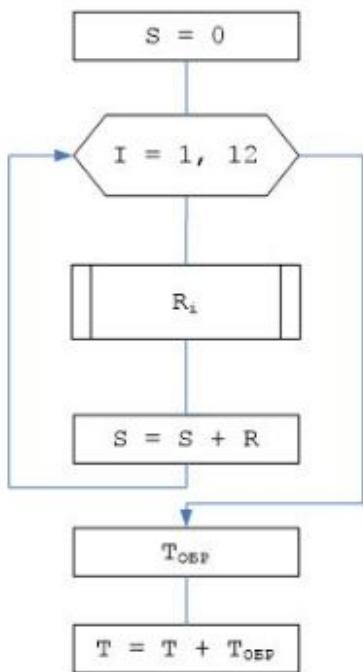
### 1. Моделирование работы обслуживающего аппарата

Программа-имитатор ОА - это комплекс, вырабатывающий случайные отрезки времени, соответствующие длительностям обслуживания требований. Например, если

требования от источника обрабатываются в ОА по нормальному закону с параметрами M и delta, то длительность обработки i-требования будет:

$$T_{обр} = M_x + \left( \sum_{i=1}^{12} R_i - 6 \right) \cdot \sigma_x$$

### Схема алгоритма имитатора.



где

1.  $R_i$  - случайное число с равномерным законом распределения
2.  $T_{обр}$  - время обработки очередного сообщения
3.  $T$  - время освобождения ОА

## 2. Язык GPSS. Объекты запоминающей категории, ячейки.

Объекты запоминающей категории обеспечивают обращения к сохраняемым значениям. Ячейки сохраняемых величин и матрицы ячеек сохраняемых величин используются для сохранения некоторой числовой информации. Любой активный транзакт может произвести запись информации в эти объекты. Впоследствии записанную в эти объекты информацию может считать любой транзакт. Матрицы могут иметь до шести измерений.

Ячейки используются для записи и хранения в процессе моделирования текущих значений СЧА.

Занесение информации в ячейку производится блоком **SAVEVALUE**, имеющим формат

**SAVEVALUE A,B,C**

где А – имя ячейки (может сопровождаться в конце знаком плюс + или минус -), В – присваиваемое значение, С – тип ячейки. Если после А стоит знак + или -, то значение поля В прибавляется или вычитается из текущего содержимого ячейки А. Если знак не указан, то значение поля В присваивается ячейке А. Поле С определяет

тип ячейки и может принимать значения: XH – полусловная, XF – полнословная, XL – с плавающей точкой. При отсутствии поля С по умолчанию принимается полнословная ячейка.

Начальное значение ячейки по умолчанию равно нулю. Для изменения начального значения применяется оператор инициализации  
INITIAL Ячейка1, Значение1,..., Ячейка K, ЗначениеK

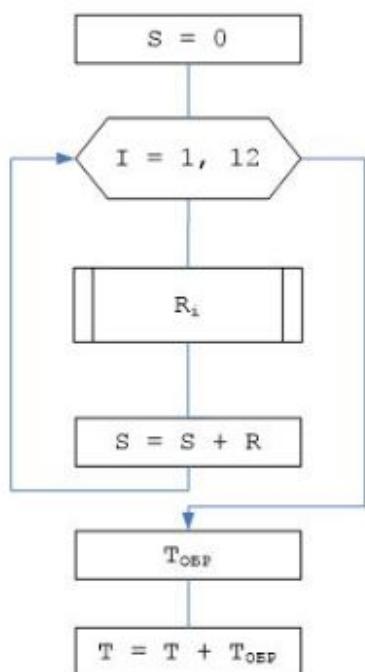
## Билет X4

### 1. Моделирование работы абонентов

Абонент - частный случай ОА, с той лишь разницей, что он может как обрабатывать заявки, так и вырабатывать их. Для моделирования работы абонентов необходимо вырабатывать длительности обслуживания требований. А генерация заявок может имитироваться с помощью генератора сообщений по наперед заданному закону.

$$T_{обр} = M_x + \left( \sum_{i=1}^{12} R_i - 6 \right) \cdot \sigma_x$$

**Схема алгоритма имитатора.**



где

1.  $R_i$  - случайное число с равномерным законом распределения
2.  $T_{обр}$  - время обработки очередного сообщения
3.  $T$  - время освобождения ОА

### 2. Язык GPSS. Ячейки, матрицы ячеек

Ячейки используются для записи и хранения в процессе моделирования текущих значений СЧА.

Занесение информации в ячейку производится блоком SAVEVALUE, имеющим формат

SAVEVALUE A,B,C

где А – имя ячейки (может сопровождаться в конце знаком плюс + или минус -), В – присваиваемое значение, С – тип ячейки. Если после А стоит знак + или -, то значение поля В прибавляется или вычитается из текущего содержимого ячейки А. Если знак не указан, то значение поля В присваивается ячейке А. Поле С определяет тип ячейки и может принимать значения: XН – полусловная, XF – полнословная, XL – с плавающей точкой. При отсутствии поля С по умолчанию принимается полнословная ячейка.

Начальное значение ячейки по умолчанию равно нулю. Для изменения начального значения применяется оператор инициализации

INITIAL Ячейка1, Значение1,..., Ячейка K, ЗначениеK

Наряду с ячейками в моделях можно использовать матрицы ячеек. В отличие от простых ячеек матрицы перед использованием должны быть описаны. Для описания матрицы применяется строка описания матрицы. В поле метки этой строки записывается имя описываемой матрицы, в поле операции - слово MATRIX, в поле operandов - параметры матрицы: в поле А записывают любое слово или оставляют поле пустым, в поле В указывают число строк матрицы, в поле С - число столбцов.

NAME MATRIX A,B,C

В начальный момент любая матрица содержит только нулевые значения. Обращение MX\$NAME(A,B).

Запись значения в ячейку матрицы:

MSAVEVALUE NAME,B,C,v\$ABC

## Может пригодиться

### delta-t принцип протяжки времени

Принцип dt – заключается в последовательном анализе состояний всех блоков в момент t+dt по заданному состоянию блоков в момент t. При этом новое состояние определяется в соответствии с их алгоритмическим описанием с учётом действующих случайных факторов, задаваемыми распределениями вероятностей. В результате этого анализа принимается решение о том, какие общесистемные события должны имитироваться программной моделью на данный момент времени. Основной недостаток принципа: значительные затраты вычислительных ресурсов на реализацию модели, а при недостаточно малом dt появляется опасность пропуска отдельных событий в системе, что приводит к искажению результатов моделирования. Характерное свойство системы обработки информации – то, что состояние отдельных устройств изменяется в дискретные моменты времени, совпадающие с моментов вступления сообщений в систему.

### Метод вложенных цепей Маркова.

Вложенные цепи Маркова образуются следующим образом. В исходном случайному процессе выбираются такие случайные процессы, в которых характеристики образуют Марковскую цепь. Моменты времени обычно являются случайными и зависят от свойств исходного процесса. Затем обычными методами теории Марковских цепей исследуются процессы только в эти характерные моменты. Случайный процесс называется **полумарковским** (с конечным или счетным множеством состояний) если

заданы переходы состояний из одного состояния в другое и распределение времени пребывания процессов в каждом состоянии. Например, в виде функции распределения или функции плотности распределения.

## Метод статистических испытаний. Метод Монте-Карло.

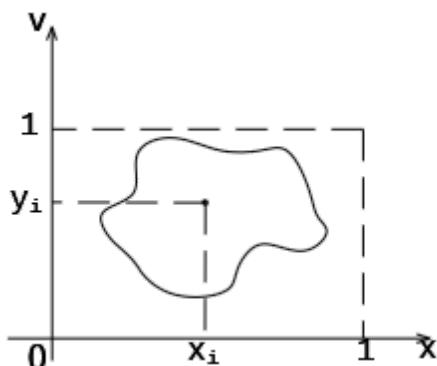
В СМО поток заявок редко бывает Пуассоновским и еще реже наблюдается распределенный закон.

Для произвольных потоков событий переводящих систему из состояния в состояние. Аналитические решения получены только для отдельных частных случаев. Когда построение аналитической модели является по той или иной причине трудно осуществимым ставится метод статистических испытаний.

\*Когда этот метод нашел реальное применение: с развитием компьютеров.

**Идея метода:** вместо того чтобы описывать случайные явления с помощью аналитической зависимости производится т.н. «розыгрыш», т.е. моделирование «случайного» явления с помощью некоторой процедуры дающей «случайный» результат. Проведя такой розыгрыш достаточно большое количество раз, получаем **статистический материал**, т.е. множество реализаций случайного явления. Дальше эти результаты могут быть обработаны статистическими методами математической статистики.

Метод Монте-Карло был предложен в 1948 году Фон-Нейманом как метод численного решения некоторых математических задач.



Введем в некотором единичном квадрате случайную величину. Задача стоит в определении её площади.

**Суть метода:**

1. Вводим в некотором единичном квадрате любую поверхность  $S$ .
2. Любым способом получаем 2 числа  $x_i, y_i$ , подчиняющиеся равномерному закону распределения случайной величины на интервале  $[0, 1]$ .
3. Полагаем, что одно число определяет координату  $x$ , второе – координату  $y$ .
4. Анализируем принадлежность точки  $(x, y)$  фигуре. Если принадлежит, то увеличиваем значение счетчика на 1.
5. Повторяем  $n$  раз процедуру генерации 2х случайных чисел с заданным законом распределения и проверку принадлежности точки поверхности  $S$ .
6. Определяем площадь фигуры как количество попавших точек, к количеству сгенерированных.

Фон-Нейман доказал, что погрешность  $\varepsilon \leq \sqrt{\frac{1}{n}}$ .

**Преимущество** метода статистических испытаний: в его универсальности, которая обуславливает его возможность статистического исследования объекта, причем

всестороннего. Но для реализации этого исследования необходимы довольно полные статистические сведения о параметрах элемента.

*Недостаток:*

Большой объем требующихся вычислений, равный количеству обращений к модели. Поэтому вопрос выбора величины  $n$  имеет важнейшее значение. Уменьшая  $n$  – повышаем экономичность расчетов, но одновременно ухудшаем их точность.





## **Моделирование. 03.09.04.**

### **Философские аспекты моделирования.**

**Объектом** называется всё то, на что направлена человеческая деятельность.

В научном исследовании большую роль играет понятие **гипотезы** – определенное предсказание, основанное на небольшом количестве опытных данных, наблюдениях, догадках. Быстрая проверка гипотезы может быть проведена в ходе специально поставленных экспериментов.

При формировании и проверке правильности гипотезы в качестве метода суждения используется *аналогия*. **Аналогией** называется суждение о каком либо частном сходстве двух объектов.

Современные научные гипотезы создаются как правило по аналогии проверенным на практике положениям. Таким образом, аналогия связывает гипотезу с экспериментом.

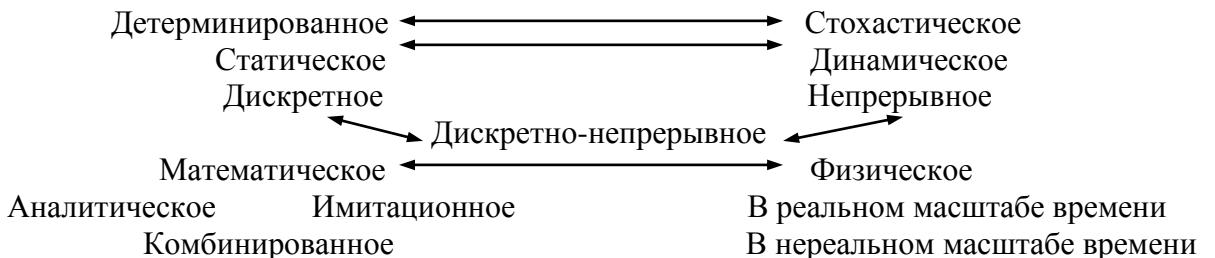
Гипотезы и аналогии, отражающие реальный объективно-существующий мир, должны обладать наглядностью или сводиться к удобным для исследования логическим схемам. Такие логические схемы, упрощающие рассуждения и позволяющие проводить эксперименты, уточняющие природу явлений, называются *моделями*.

**Модель** – объект - заместитель объекта оригинала, обеспечивающий изучение некоторых свойств оригинала.

Замещение одного объекта другим с целью получения информации о важнейших свойствах объекта-оригинала с помощью объекта-модели называется **моделированием**.

### **Классификация видов моделирования.**

В зависимости от характера изучаемых процессов в некоторой сложной системе все виды моделирования можно разделить.



- *Детерминированное* моделирование отображает детерминированные процессы, т.е. такие, в которых отсутствуют всякие случайные воздействия.
- *Стохастическое* моделирование отображает случайные, вероятностные процессы и события.
- *Статическое* служит для описания сложной системы в конкретный момент времени.
- *Динамическое* отражает поведение системы во времени.

- *Дискретное* моделирование используется для описания процессов, происходящих в дискретные моменты времени.
- *Непрерывное* используется для описание непрерывных во времени процессов.
- *Дискретно-непрерывное* используется для тех случаев, когда хотят отразить наличие как дискретных, так и непрерывных процессов в системе.
- Под *математическим моделированием* будем понимать процесс установления данному реальному объекту некоторого математического объекта, называемого *математической моделью* и исследование этой модели, позволяющее получить характеристики реального объекта. Любая математическая модель, как и всякая другая, описывает реальный объект лишь с некоторой степенью приближения.
- Для *аналитического моделирования* характерным является то, что процессы функционирования элементов системы записываются в виде некоторых функциональных соотношений (алгебраических, интегрально-дифференциальных, конечно-разностных и т.д.) или логических условий.

Аналитические модели могут быть исследованы тремя способами:

1. *Аналитическим*. Получение в общем виде зависимости выходных характеристик от исходных.
  2. *Численным*. Нельзя решить сложные уравнения в общем виде. Результаты получают для конкретных начальных данных.
  3. *Качественным*. Нет возможности получения конкретных решений, но можно выделить некоторые свойства объектов или решений уравнений, например, оценить устойчивость решения.
- При *имитационном моделировании* алгоритм, реализующий модель, воспроизводит процесс функционирования системы во времени. Имитируются элементарные явления, составляющие процесс, с сохранением логической структуры объекта и последовательности протекания процесса во времени. Это позволяет по исходным данным получить сведения о состоянии процесса в определенные моменты времени. Преимуществом имитационного моделирования является возможность решения более сложных задач.  
Имитационные модели позволяют достаточно просто учитывать такие факторы, как наличие дискретных и непрерывных элементов, нелинейные характеристики системы, многочисленные случайные воздействия. Когда результаты, полученные имитационной моделью, являются реализацией случайных величин и функций, то для нахождения характеристик процесса функциональной системы необходимо его многократное воспроизведение с последующей статистической обработкой.
  - *Комбинированное моделирования* при анализе сложных систем позволяет объединить достоинства отдельных методов. В нем проводят декомпозицию процесса функционирования сложной системы на подпроцессы и для тех, где можно используют аналитические модели, где нельзя – имитационное моделирование.

## **Технические средства математического моделирования.**

### **Цифровая техника**

Цифровая техника является дискретной. Основная проблема – быстродействие (не догнать реальное время) слишком сложен механизм.

## **Аналоговая техника.**

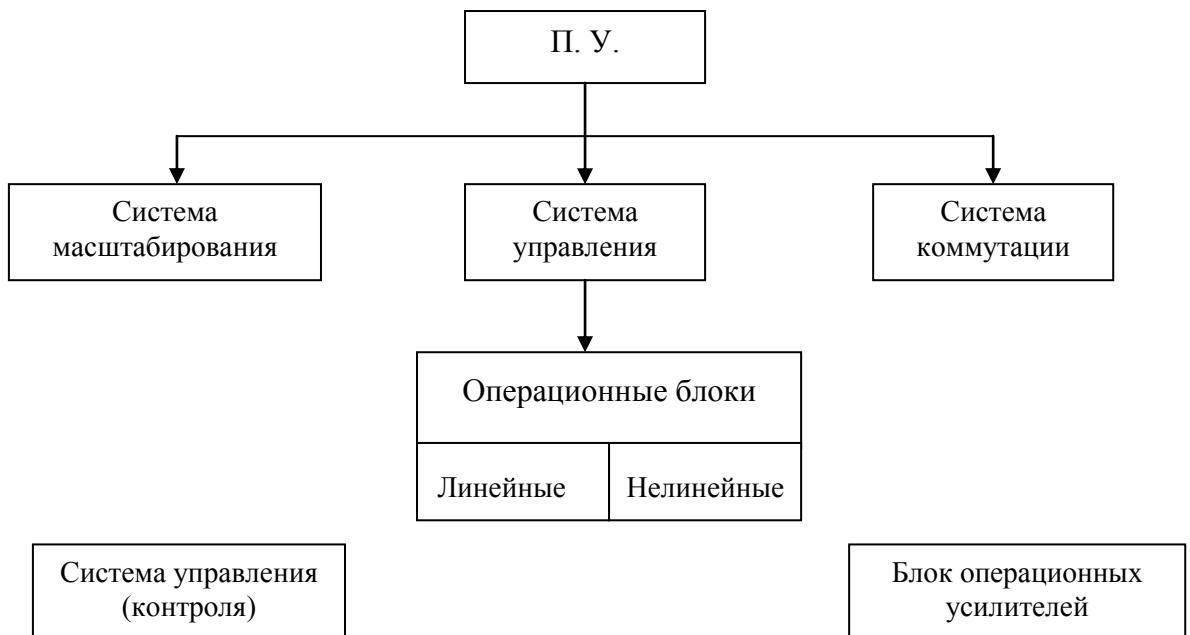
В отличие от дискретной техники в основе аналоговой лежит принцип моделирования, а не счета. При использовании в качестве модели некоторой задачи электронных цепей, каждой переменной величине ставится в соответствие определенную переменную величину электрической цепи. При этом основой построения такой модели является *изоморфизм* - подобие исследуемой задачи и соответствующей электрической модели. При определении критерия подобия используют специальные приемы масштабирования, соответствующие заданным параметрам.

Согласно своим вычислительным возможностям АВМ наиболее приспособлены для исследования объектов, динамика которых описывается обыкновенными дифференциальными уравнениями и уравнениями в частных производных, реже - алгебраическими, следовательно, АВМ можно отнести к классу специальных машин.

В общем случае под АВМ понимаем совокупность электрических элементов, организованных в систему, позволяющих изоморфно моделировать динамику изучаемого объекта. Функциональные блоки АВМ должны реализовывать весь комплекс арифметико-логических операций.

АВМ делятся по мощности (степень дифференциальных уравнений):

- малые ( $n \leq 10$ )
- средние ( $10 \leq n \leq 20$ )
- большие ( $n \geq 20$ )



## **Гибридные ВМ**

Широкий класс ВС, использующий как аналоговый, так и дискретный метод представления и обработки информации.

Подклассы гибридных ВМ:

1. АВМ с цифровыми методами численного анализа

2. АВМ, программируемые с помощью ЦВМ
3. АВМ с цифровым управлением и логикой
4. АВМ с цифровыми элементами (цифровые вольтметры, память)
5. ЦВМ с аналоговыми арифметическими устройствами
6. ЦВМ, допускающие программирование аналогового типа.

ЦВМ

ЦАП АЦП

схема  
согласования

АВМ

В АВМ накладывают и складывают сигналы.

АВМ  $\Leftrightarrow$  система сопряжения АВМ – ЦВМ.

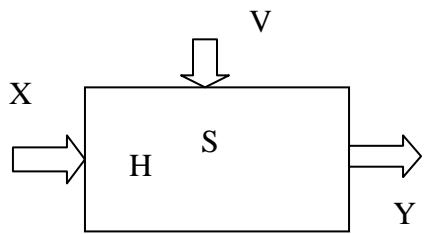
ЦВМ  $\Leftrightarrow$  электрическая система согласования

#### **Моделирование. 10.09.04.**

Сравнительная характеристика аналоговых и цифровых вычислительных машин.

Показатель	Аналоговые	Цифровые
Тип информации	Непрерывный	Дискретный
Изменение значений	Величиной напряжения	Числовым значением
Базовые операции	Арифметические интегрирование	Арифметические
Принцип вычисления	Высоко параллельный	Последовательно-параллельное.
Режим реального времени	Без ограничений	Ограничены возможности
Динамическое изменение решаемой задачи	Посредством системы коммутации	В диалоговом режиме
Профессиональные основные требования к пользователю	Проф. Знания плюс методика	Знания основ алгоритмизации
Уровень формализации задач	Ограничен моделью рассматриваемой задачи	Высокий
Способность решения логических задач	Ограниченный	Высокое
Точность вычислений	$\Phi_i \leq 10^{-4}$	$\Phi_i \sim 10^{-40}$
Диапазон представления чисел	$1 - 10^{-4}$	$10^{-40} - 10^{+40}$
Класс решаемых задач	Алгебраические и дифференциальные уравнения	Любые
Специализированные функции	Ограниченный набор	Широкий класс
Уровень миниатюризации	Ограничен	Высокая степень интеграции
Область применения	Ограниченная	Широкий спектр
Пользовательский интерфейс	Низкий уровень	Высокий

#### **Основные понятия теории моделирования.**



В общем случае модели объектного моделирования можно представить себе в виде множества величин описывающих процесс функционирования реальной системы и образующих следующие подмножество:

$x_i \in X, i = \overline{1, n_x}$	$\vec{x(t)} = \{x_1(t), x_2(t), \dots, x_{n_x}(t)\}$	Множество входных параметров
$h_j \in H, j = \overline{1, n_h}$	$\vec{h(t)} = \{h_1(t), h_2(t), \dots, h_{n_h}(t)\}$	Множество внутренних параметров
$v_k \in V, k = \overline{1, n_v}$	$\vec{v(t)} = \{v_1(t), v_2(t), \dots, v_{n_v}(t)\}$	Внешнее воздействие
$y_m \in Y, m = \overline{1, n_y}$	$\vec{y(t)} = \{y_1(t), y_2(t), \dots, y_{n_y}(t)\}$	Множество выходных параметров

В общем случае  $x$   $v$   $h$   $y$  являются элементами непересекающихся подмножеств и содержат как стохастические, так и детерминированные составляющие. При моделировании функционирования сложной системы  $S$  входные воздействия  $X$ , воздействия внешней среды  $V$ , внутренние параметры  $H$  являются независимыми (экзогенными) переменными, которые в векторном виде можно определить следующим образом.

$$\vec{x(t)} = (x_1(t), x_2(t), \dots, x_n(t))$$

$$\vec{v(t)} = (v_1(t), v_2(t), \dots, v_n(t))$$

$$\vec{h(t)} = (h_1(t), h_2(t), \dots, h_n(t))$$

А выходные характеристики являются зависимыми (эндогенными) переменными.

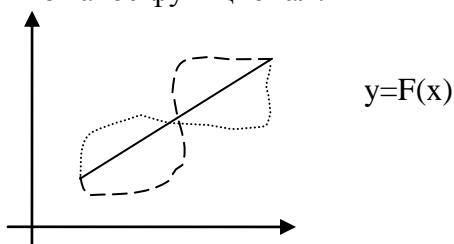
$$\vec{y(t)} = (y_1(t), y_2(t), \dots, y_n(t))$$

Процесс функционирования системы описывается во времени некоторым оператором  $F_s$ . Который в общем случае преобразует независимые переменные в зависимые.

$$\vec{y(t)} = F_s(\vec{x}, \vec{v}, \vec{h}, t) \quad (1)$$

Зависимость (1) называется **законом функционирования сложных систем**. В общем виде он может быть задан в виде функции, функционала, логических условий, в алгоритмическом или табличном виде.

Что такое функционал?



Множество путей соединяющих точек – функционал.

### **Алгоритм функционирования.**

Под которым будем понимать метод получения выходных характеристик с учетом входных воздействий, воздействий внешней среды и соответствующих объекту внутренних параметров. Один и тот же закон функционирования может быть реализован различными способами, т.е. с помощью множества различных алгоритмов функционирования.

Соотношение (1) может быть получено через состояния системы, т.е. через свойства системы в определенные моменты времени, которые так же характеризуются вектором состояний

$$\overline{z(t)} = (z_1(t), z_2(t) \dots z_n(t))$$

Если рассматривать функционирование системы как последовательную смену состояний, то они могут быть интерпретированы как координаты точки в n-мерном фазовом пространстве, причем каждые реализации процесса будут соответствовать некоторая фазовая траектория. Совокупность всех возможных состояний системы называется пространством состояний.

состояние системы S в момент времени t полностью определяется начальным условием  $z_0$  в векторной форме

$$t_0 \leq t \leq T$$

$$\overrightarrow{z^0} = (z_1^0, z_2^0, \dots)$$

$$z_1^0 = z_1(t_0)$$

$$z_2^0 = z_2(t_0)$$

...

входными воздействиями, внутренними параметрами, и воздействиями внешней среды, которые имели место за интервал времени  $(t - t_0)$ . Это можно определить следующими уравнениями:

$$\overrightarrow{z(t)} = \phi(\overrightarrow{z^0}, \vec{x}, \vec{h}, \vec{v}, t)$$

$$\overrightarrow{y(t)} = F(\overrightarrow{z}, t)$$

$$\overrightarrow{y(t)} = F(\phi(\overrightarrow{z^0}, \vec{x}, \vec{h}, \vec{v}, t))$$

В общем случае время в модели может быть непрерывным во всем временном диапазоне, а может быть и дискретным, т.е. квантованным на заданном интервале.

$$F = m \cdot \Delta t$$

Где m – число интервалов дискретизации.

Таким образом, под математической моделью реальной системы понимают конечное множество переменных x, v, h вместе с математическими связями между ними и характеристиками y(t).

### **Типовые математические схемы.**

В практике моделирования на первоначальных этапах формализации объектов используют так называемые типовые математические схемы, к которым относят хорошо разработанные математические объекты.

Процесс функционирования системы	Типовая математическая схема	Обозначение
Непрерывно-детерминированный подход	Дифференциальные уравнения	D
Дискретно-детерминированный подход	Конечные автоматы	F
Дискретно-стохастический подход	Вероятностные автоматы	P
Непрерывно-стохастический подход	Система массового обслуживания (P-Net)	Q
Обобщенный универсальный подход	Агрегативные системы	A

---

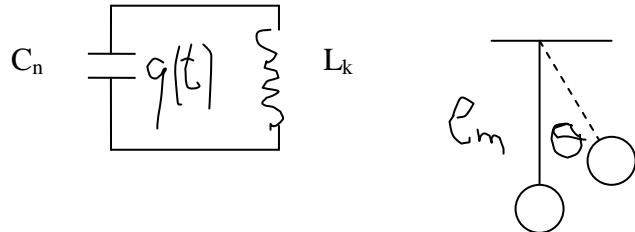
### Моделирование. 17.09.04.

Примеры:

Схема с конденсатором (рис. 1)

Маятник (рис. 2.)

Полная модель (с внутр. и внешними воздействиями): Рис. 3



$$y' = f(y^* t)$$

$$L_k \frac{d^2 q(t)}{dt^2} + \frac{q(t)}{C_k} = 0$$

$$T = 2\pi\sqrt{hl}$$

$$m_M l_M^2 \frac{d^2 \theta(t)}{dt^2} + m_M g l_M \theta = 0$$

$$h_2 = L_k = m_M l_M^2$$

$$h_1 = 0$$

$$h_0 = \frac{1}{C_k} = m_M g l_M$$

$$Z(t) = q(t) = \theta(t)$$

$$h_2 \frac{d^2 z(t)}{dt^2} + h_1 \frac{dz(t)}{dt} + h_0 z(t) = 0 - \text{универсальное уравнение}$$

$$h_2 \frac{d^2 z(t)}{dt^2} + h_1 \frac{dz(t)}{dt} + h_0 z(t) = X(t)$$

## **Формализация и алгоритмизация процесса функционирования сложных систем**

Сущность компьютерного моделирования состоит в проведении эксперимента с моделью, которая обычно представляет собой некоторый программный комплекс, описывающий формально или алгоритмически поведение элементов в системе в процессе ее функционирования, т.е. во взаимодействии друг с другом и с внешней средой.

*Основные требования*, предъявляемые к модели, отображающей функционирование некоторой системы:

1. *полнота модели* - должна предоставлять пользователю возможность получения необходимого набора характеристик, оценок системы, с требуемой точностью и достоверностью.
2. *гибкость модели* – должна давать возможность воспроизведения различных ситуаций при варьировании структуры, алгоритмов и параметров модели. При чем структура должна быть блочной – допускать возможность замены, добавления, исключения некоторых частей без переделки всей модели.
3. *машинная реализация модели* – должна соответствовать имеющимся ресурсам. Ресурсы - технические, экономические.  
Пример: автомобиль, критерий – мин. расход топлива. В результате получаем...  
инвалидную коляску из Германии ☺

Процесс моделирования включает разработку и машинную реализацию модели, является итерационным. Этот итерационный процесс продолжается до тех пор, пока не будет получена модель, которую можно считать адекватной в рамках решения поставленной задачи (исследования и проектирования).

### **Основные этапы моделирования больших систем:**

1. Построение концептуальной (описательной) модели системы и ее формализация
2. Алгоритмизация модели и ее компьютерная реализация.
3. Получение и интерпретация результатов моделирования

#### **1 этап**

Формулируется модель и строится ее формальная схема.

Основное назначение – переход от содержательного описания объекта к его математической модели. Исходный материал – описание.

##### **Последовательность действий**

1. Проведение границы между системой и внешней средой
2. Исследование объекта с точки зрения основных составляющих процесса
3. Переход от содержательного описания системы к формализованному описанию свойств процесса функционирования – непосредственно к концептуальной модели. Переход от описания к модели сводится к исключению из рассмотрения некоторых второстепенных элементов описания. Причем полагается, что они не оказывают существенного влияния на ход процесса.

То, что осталось, разбивается на группы:

Блоки 1 группы – имитатор воздействия внешней среды.

Блоки 2 группы – модель процесса функционирования.

Блоки 3 группы – вспомогательные, служат для машинной реализации 1 и 2 групп.

4. Процесс функционирования системы так разбивается на подпроцессы, чтобы построение модели отдельных процессов было элементарно и не вызывало особых затруднений.

## **2 этап**

Математическая модель реализуется в конкретную программу. Основной этап – блочная логическая схема.

### *Последовательность действий*

1. Разработка схемы моделирующего механизма
2. Разработка схемы программы.
3. Выбор технических средств для реализации компьютерной программы
4. Программирование, отладка
5. Проверка достоверности программы на тестовых примерах.
6. Составление технической документации – логические схемы, схемы программы, текст, спецификация, иллюстрации и т.д.

## **3 этап**

Компьютер используется для проведения рабочих расчетов по готовой программе. Результаты этих расчетов позволяют проанализировать и сделать выводы по характеристикам процесса функционирования исследуемой системы.

### *Последовательность действий*

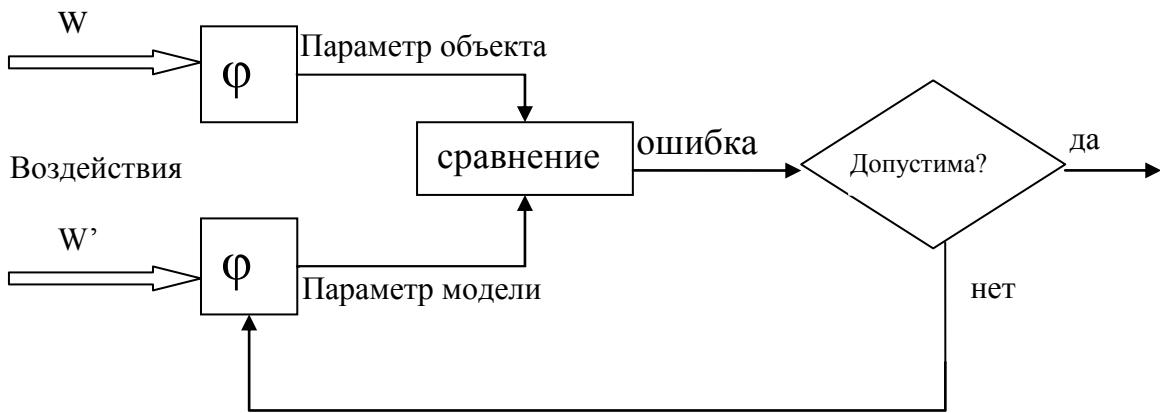
1. Планирования машинного эксперимента (активный/пассивный)  
Составление плана проведения эксперимента, с указанием переменных и параметров, для которых должен проводиться эксперимент.  
Главная задача – дать максимальный объем информации об объекте при минимальных затратах машинного времени.
2. Проведений рабочих расчетов (контрольная калибровка модели).
3. Статистическая обработка результатов и их интерпретация.
4. Составления отчетов

Различают стратегическое и тактическое планирования машинного эксперимента.

При *стратегическом планировании* ставится задача построения оптимального плана эксперимента для достижения цели, поставленной перед моделированием (оптимизация структуры, алгоритмов и параметров).

*Тактическое планирование* преследует частные цели оптимальной реализации каждого конкретного эксперимента из множества необходимых – оно задается при стратегическом планировании.

### **Схема интерактивной структуры калибровки**



3 основных класса ошибок:

1. ошибки формализации – неполная, недостаточная модель
2. ошибки решения – некорректный или слишком упрощенный метод построения модели.
3. ошибки задания параметров модели

### Проверка адекватности и корректировка модели

Проверка адекватности модели в некоторой системы заключается в анализе ее соразмерности, а также равнозначности. Адекватность нарушается из-за идеализации внешних условий и режима функционирования, пренебрежением некоторых случайных факторов.

Простейшая мера адекватности –

$$\Delta y = |y_0 - y_m|$$

$$\delta = |y_0 - y_m| / y_0$$

Считают, что модель адекватна с системой, если вероятность того, что отклонение дельта( $y$ ) не превышает предельной величины дельта, больше допустимой вероятности  $P_d$  Рис. 6.

Практически использовать невозможно, т.к.

1. для проектирования или модернизирования системы отсутствует информация о объекте
2. система оценивается не по одной, а по множеству характеристик. Они могут быть случайными величинами.
3. Отсутствует возможность априорного точного задания предельных отклонений дельта и допустимых вероятностей.

На практике оценка адекватности производится путем экспертного анализа *разумности результатов моделирования*. Выделяют следующие виды проверок:

1. Проверка модели элементов
2. Проверка модели внешних воздействий
3. Проверка концептуальной модели – ошибки в постановке задач.
4. Проверка формализованной и математической модели
5. Проверка способов вычисления выходных характеристик
6. Проверка программной модели.

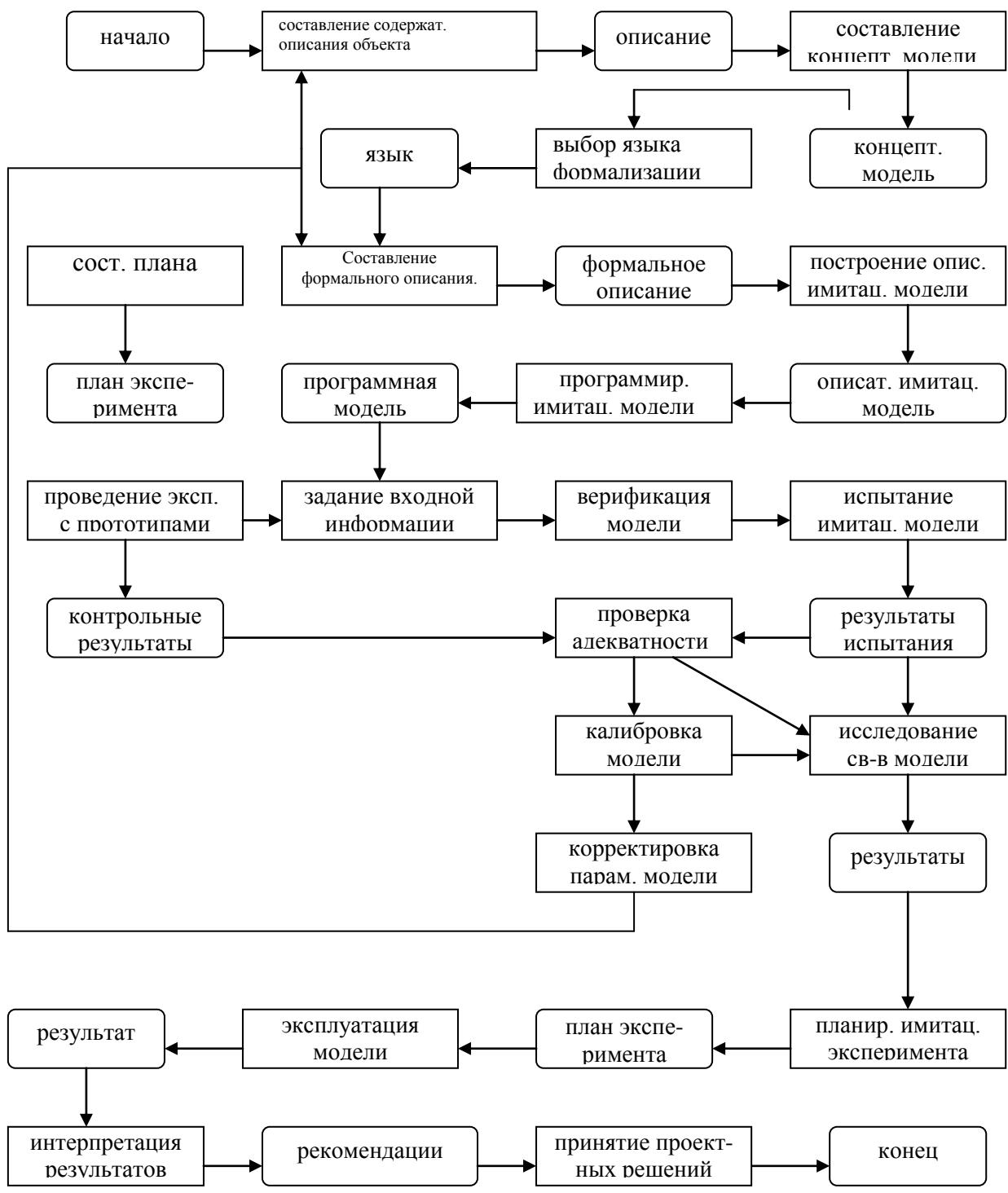
Если по результатам проверки адекватности модели появляются недопустимые рассогласования, то возникает необходимость в калибровке или корректировке.

Выделяют следующие виды изменений.

1. Глобальные – возникают в случае обнаружения методических ошибок концептуальной или методической модели.  
Исправления сводится к исправлению модели
2. Локальные, связанные с уточнение параметров и алгоритмов. Эти изменения выполняются путем замены модели на эквивалентные, но более точные.
3. Параметрические изменения некоторых специальных калибровочных параметров.  
Следует заранее выявить эти «влияющие» параметры и предусмотреть простые способы их варьирования.

Завершается определением и фиксацией области пригодности модели, под которой будем понимать множество условий, при соблюдении которых точность моделирования остается в рамках допустимой.

#### **Схема взаимосвязей технологических этапов моделирования**



# Вычислительная система как объект моделирования

В практике проектирования выявляют уровни проектирования:

1. Системный уровень проектирования.  
В качестве элементов – процессор, внешние устройства
  2. ФЛУП – функционально логический уровень проектирования
    - РП – регистровые передачи
    - ЛУ – логический уровень
  3. Схемотехнический – вероятностный конечный автомат.
  4. Конструкторский – рассматривается конструкция системы. Важнейшие вопросы – охлаждение, надежность, размещение элементов на платах и т.д.

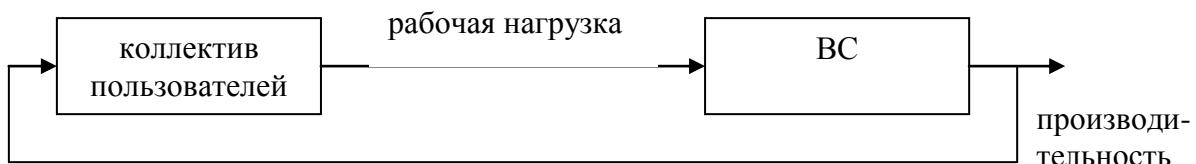
Моделирование на системном уровне – под вычислительной системой будем понимать комплекс аппаратных и программных сред, которые в совокупности выполняют определенные рабочие функции.

ОС – набор ручных и автоматических процедур, которые позволяют группе людей эффективно использовать вычислительную установку.

Коллектив пользователей – сообщество людей, которые используют систему для удовлетворения своих нужд по обработке информации. Входные сигналы – программы, данные, команды, которые создаются коллективом пользователей, называются рабочей нагрузкой.

Термин установка – комплекс из определенной среды.

### **Схема вычислительной установки.**



*Индекс производительности* – некоторый описатель, который используется для представления производительности системы. Различают количественные и качественные индексы производительности.

Легкость использования системы, мощность системы команд.

#### *Основные количественные индексы:*

1. Пропускная способность – объем информации, обрабатываемой в единицу времени
2. Время реакции – время между предъявлением системе входных данных и выявлением соответствующей выходной информации
3. Коэффициент использования оборудования – отношение времени использования указанной части оборудования в течении заданного интервала времени, длительности этого интервала.

#### *Концептуальная модель включает в себя:*

сведения о выходных и конструктивных параметрах системы, ее структуре, особенности работы каждого элемента – ресурса, характере взаимодействия между ресурсами. Сюда же относится постановка прикладной задачи – цель моделирования, а также цель моделирования. Формализация процесса функционирования такой системы отображается например СМО – системах массового обслуживания.

#### *Основные задачи, решаемые при моделировании вычислительных систем:*

1. Определение принципов организации вычислительных систем.
2. Выбор архитектуры, уточнение функций вычислительной системы и их разделение на подфункции, реализуемые аппаратно или программным способом.
3. Разработка структурной схемы – определение состава устройств, и способов их взаимодействия.
4. Определение требований к выходным параметрам устройств

## 5. Формирование ТЗ на разработку отдельных устройств.

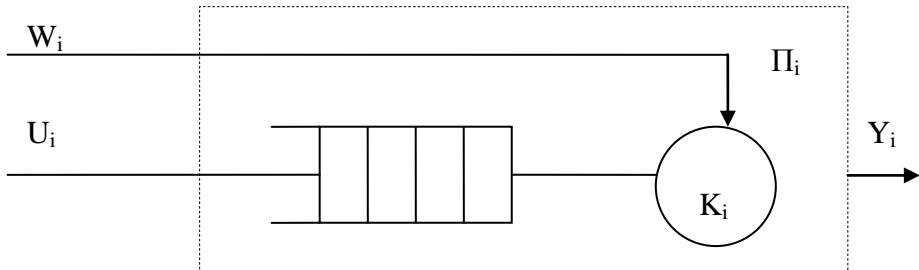
### Непрерывно стохастические модели (Q-схемы)

В этом случае используемая система формализуется как некоторая система обслуживания. Характерным для таких объектов является случайное появление заявок (требований) на обслуживание и завершение обслуживания в случайные моменты времени.

В любом элементарном акте обслуживания можно выделить 2 основные составляющие

1. ожидание обслуживания
2. собственно обслуживание

Можно изобразить в виде некоторого прибора обслуживания.



Поток событий – последовательность событий, происходящих одно за другим.

*Однородный* - только моментами поступления этих событий. Задается временной последовательностью

*Неоднородный* – если он задается не только множеством вызывающих моментов, но и набором признаков-событий (наличие приоритета).

Если интервалы времени между сообщениями независимы между собой и являются СВ, то такой поток – поток с ограниченным последействием.

Поток событий – *ординарный*, если вероятность того, что на малый интервал времени  $dt$ , примыкающий к моменту времени  $t$  попадает больше одного события пренебрежительно мала по сравнению с вероятностью того, что на этот же интервал попадает ровно одно событие.

Поток – *стационарный*, если вероятность появления того или иного числа событий на интервале времени зависит лишь от длины этого участка и не зависит от того, где на оси времени взят этот участок.

*Пример*

Для ординарного потока среднее число сообщений, поступающих за интервал  $dt$ , примыкающее к моменту времени  $t$

Условие  
рис. 10.

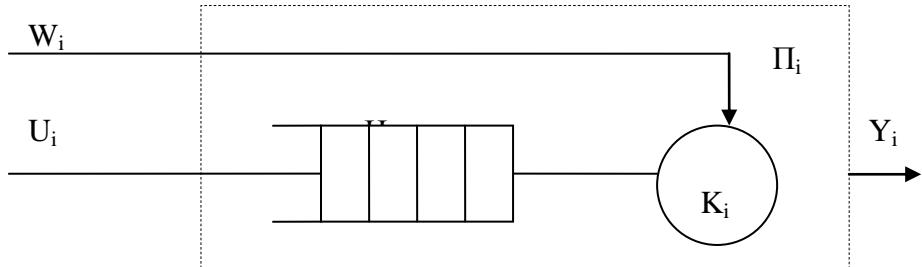
Для стационарного потока его интенсивность не зависит от времени и представляет собой постоянное значение равное среднему числу событий, наступающих в единицу времени.

**Моделирование. 24.09.04.**

Базовый – прибор обслуживания.

В любом элементарном акте обслуживания можно выделить 2 основные составляющие:

1. собственно, обслуживание
2. ожидание обслуживания заявки



$K$  – канал,  $H$  – накопитель,  $\Pi$  – прибор обслуживания

В  $i$ -ом приборе обслуживания имеем:

- $w_i$  – поток заявок т.е. *интервалы времени* между моментами появления заявок (вызывающие моменты) на входе канала  $K_i$ .
- $u_i$  – поток обслуживания – интервалы времени между началом и окончанием обслуживания заявок.

$W_i$  – интервалы времени между началом и концом времени обслуживания.

Заявки, обслуженные каналом, и заявки, покинувшие  $i$ -ый прибор не обслуженным, образуют *выходной поток*  $Y_i$ , т.е. интервалы времени между выходами заявок из системы.

Процесс функционирования  $i$ -ого прибора обслуживания можно представить как изменение состояний его элементов во времени. Переход в новое состояние для  $i$ -ого прибора означает изменение количества заявок, которые в нем находятся. Они либо в накопителе, либо в канале.

$$\vec{Z}_i(t) = (Z_i^H, Z_i^K, t)$$

$$Z_i^H = \begin{cases} 0 - \text{накопитель\_свободен} \\ 1 - \text{накопитель\_занят} \end{cases} \quad Z_i^K = \begin{cases} 0 - \text{канал\_свободен} \\ 1 - \text{канал\_занят} \end{cases}$$

$Z_i$  – состояние накопителя в дискретные моменты времени.

В практике моделирования, элементарные схемы обычно объединяются, при этом если каналы соединены параллельно, то имеет место *многоканальное обслуживание*; если последовательно – *многофазное обслуживание*.

Таким образом, для формализации схемы функционирования Q-схемы необходимо использовать *оператор сопряжения* –  $R$ , отражающий взаимосвязь элементов структуры между собой.

Различают разомкнутые и замкнутые Q-схемы.

В *разомкнутых* выходной поток обслуживаемых заявок не может поступить снова на какой-либо элемент структуры, т.е. отсутствует обратная связь.

В *замкнутых* системах имеется обратная связь.

*Собственные параметры* Q-схемы:

- количество фаз

- количество каналов в каждой фазе
- Количество накопителей в каждой фазе
- Емкость каждого накопителя

$$R, Q = \begin{cases} L_\phi \\ L_{Kj}, j = \overline{1, L_\phi} \\ L_H \\ L_i^H \end{cases}$$

В зависимости от емкости накопителя, в теории массового обслуживания исп. след. терминологию:

$$L_i^H = \begin{cases} 0 - \text{система с потерями} \\ \rightarrow \infty - \text{бесконечная емкость, очередь заявок не ограничена} \end{cases}$$

$H$  – вся система параметров

$$H = (L_\phi, L_{Kj}, L_H, L_i^H)$$

Для задания Q-схемы также необходимо описать ее алгоритм, который определяет набор правил поведения заявок в различных ситуациях. Неоднородность заявок, отражающая реальный процесс в той или иной мере с введением понятий классов и приоритетов.

Различают – *статические, динамические, относительные, абсолютные* приоритеты.  
При работе главная задача – минимально загрузить процессор.

Весь набор возможных алгоритмов поведения заявок можно представить в виде некоторого оператора алгоритма – А поведения заявок. Таким образом Q-схема, описывающая процесс функционирования системы массового обслуживания любой сложности, однозначно задается в виде:

$$Q = (W, U, R, H, Z, A)$$

$W$  – подмножество входящих потоков

$U$  – подмножество потоков обслуживания

$R$  – оператор сопряжения элементов в системе

$H$  – подмножество собственных параметров

$Z$  – множество со стоянкой системы

$A$  – оператор алгоритмов обслуживания заявок

Возможности использования характеристик с использованием аналитических методов, разработанных в теории массового обслуживания, является узкоограниченными по сравнению с требованиями практического исследования таких систем. Следовательно необходимо использование имитационных методов.

## Эксперимент

Задача моделирования – исследование объекта оригинала по методу черного ящика.  
Факторы – критерии системы, от которых зависит поведение системы.

В процессе моделирования мы задаем значения входных факторов, получаем значения выходных. Однако могут еще присутствовать случайные воздействие, которые мы можем не учесть.

Имитационная модель – сложная модель, вероятностный характер, все характеристики имеют вероятностные значения. На вход – определенная комбинация. То, что получается на выходе, по закону функционирования, может не соответствовать. Берем отдельный выходной параметр и наблюдаем при серии экспериментов, как он меняется.

Цель планирования – обеспечение минимального числа затрат. Как проводить, сколько испытаний, интервалы времени. Минимум затрат, максимум информации об объекте. На основе исследования модели получаем *поверхность отклика*.

рис.1

Закон распределения выходного параметра.

Планирование разделяют на *тактическое и стратегическое*.

*Тактическое планирование* – условия проведения единичного эксперимента: чаще всего определение объема выборки, обеспечивающего заданную статистическую погрешность целых числовых характеристик, распределение функции отклика.

*Стратегическое планирование*.

Если решается задача синтеза, задача определения параметров, обеспечивающих требуемое качество функционирования исследуемой системы, необходимо знание зависимости характеристик от параметров. Эти задачи с помощью тактического пл. Как правило существует прямая и обратная задачи.

*Прямая задача* – проводится серия экспериментов и по их результатам необходимо определить функциональную зависимость, связывающую характеристики исследуемой системы с параметрами. При этом обычно задаются классом исследуемые зависимости.

Рис. 2.

Минимизировать количество точек.

*Обратная задача* – необходимо так планировать эксперимент, чтобы при минимальном числе опытов, построить экспериментальную функцию отклика (реакции), связывающую отклик (реакцию) с системой параметров или с множеством входных независимых факторов  $Y_i$ , которые можно варьировать при постановке эксперимента.  $X_i$  – переменные,  $Y_i$  – вектор отклика системы. Рис. 3

Разложение в ряд Тейлора в окрестности точки  $X_0$ .

Наряду с управляемыми факторами, на систему могут воздействовать неуправляемые и не контролируемые факторы, в результате чего  $Y$  – случайная величина, поэтому по результатам эксперимента мы можем построить только приближенную зависимость вида

Рис. 4.

$Y$  – математическое ожидание.

Могут служить математическими оценками теоретических коэффициентов ряда Тейлора. Таким образом наша задача – определение выборочных коэффициентов регрессии по результатам опытов в  $M$  точках пространства.

Делаем 2 допущения:

1. результаты наблюдений отклика системы в различных точках факторного пространства рассматриваются как независимые случайные величины с одинаковыми дисперсиями.
2. погрешности задания варьируемых факторов существенно меньше погрешности измерения  $Y$ .

Рис.5.

### **Моделирование. 01.10.04.**

(Эксперимент - продолжение)

Рис. 1.

Необходимо организовать  $m$  экспериментов, в котором каждому  $F_j$  назначается...  
каждому  $j$ -ому набору эффективных переменных  $B_j$  соответствует единичный эксперимент с номером  $G$ , в котором проводится  $n$  наблюдений отклика системы. После усреднения определяется математическое ожидание отклика  $Y_j$ .

Наша задача – построение плана такого эксперимента – выбор параметра  $F_j$ . Построим матрицу.

Рис.2.

Для нахождения наименьшего решения системы уравнений, относительно коэф.  $B_j$  необходимо и достаточно, чтобы определитель матрицы  $C$  не равен 0. Это условие выполняется в случае линейной независимости столбцов матрицы  $M$ .

Если  $C$  – диагональная матрица, то решение упрощается. Рис.3.

Условие диагональности  $C$  – попарная ортогональность столбцов матрицы  $F$ . Для получения независимых оценок, коэффициентов уравнения регрессии надо так спланировать эксперимент, т.е. построить такую матрицу планирования, чтобы выполнялось условие линейной независимости и ортогональности матриц.

### **Тактическое планирование**

Связано с вопросом эффективности. Прежде связано с :

1. определение начальных условий в той мере, в которой они влияют на эксперимент.
2. сокращение размеров выборки при уменьшении размеров дисперсии

....

Для получения данных, связывающих характеристики, описывающие функционирование  $q$ -схемы вводят некоторые допущения относительно входных потоков, функции распределения, длительностей обслуживаний запросов и дисциплин обслуживания.

Для таких систем в качестве типовой модели будем рассматривать теорию Марковских процессов. Случайный процесс, протекающий в некоторой системе  $S$  называется

*Марковским процессом*, если он обладает следующим свойством: для каждого момента времени  $t_0$  вероятность любого состояния системы в будущем (при  $t > t_0$ ), зависит только от ее состояния в настоящем ( $t = t_0$ ), и не зависит от того, когда и каким образом система пришла в это состояние. Т.е. в таком процессе будущее развитие зависит только от настоящего состояния и не зависит от истории процессов. Для Марковского процесса разработаны уравнения Колмогорова:

$$F = (P'(t), P(t), \Lambda)$$

$$\Lambda = \{f_1, f_2, \dots, f_z\}.$$

$\Lambda$  – набор некоторых коэффициентов.

Для стационарного распределения соотношение имеет следующий вид:  
 $\Phi = (P(t), \Lambda) = 0$ .

Отсюда выводим:  $P = P(\lambda)$

Выходная функция  $Y = Y(P(\lambda))$  – Данное соотношение представляет собой зависимость выходных параметров от некоторых внутренних параметров. Данное соотношение – *базисная модель*.

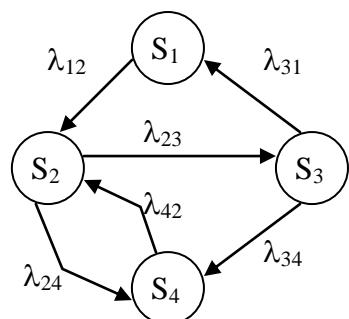
Необходимо осуществить связь между внутренними параметрами модели, входными параметрами, внешним воздействием.

$\Lambda = \Lambda(X, V, H)$  – *интерфейсная модель*.

Математическая модель Q-системы строится как совокупность базисной и интерфейсной модели. Это позволяет использовать одни и те же базисные модели при решении задачи проектирования, осуществляя настройку на соответствующую задачу посредством изменения интерфейсной модели. Параметрическая настройка на конкретную задачу.

Математическая модель должна обеспечить вычисление времени реакции на запрос и определить производительность системы.

### Методика вывода уравнений Колмогорова.



Система описана 4 состояниями и переходами из одного состояния в другое. Как они переходят? Есть такое понятие как интенсивность состояния.

$\Lambda$  – плотность вероятности для множества состояний.

Найдем вероятности  $P_1(t)$ , т.е. вероятность того, что в момент времени  $t$  система будет находиться в состоянии  $S_1$ . Придадим  $t$  малое приращение  $dt$ , и найдем вероятность того, что в момент  $(t+dt)$  система будет находиться в состоянии  $S_1$ .

Это событие может произойти двумя способами:

1. В м-т  $t$  система уже находилась в состоянии  $S_1$  и за время  $dt$  не вышла из него  
 $P_1(t)(1-\lambda_{12}dt)$

2. Система находилась в состоянии S3.

$$P_3(t)\lambda_{31}dt$$

$$P_1(t)*(1-\lambda_{12}*\Delta t).$$

$$P_3(t)*\lambda_{31}*\Delta t.$$

$$P_1(t+\Delta t) = P_1(t)*(1-\lambda_{12}*\Delta t) + P_3(t)*\lambda_{31}*\Delta t.$$

Раскроем скобки в правой части, P1 в левую часть и все поделить на dt.

$$\lim_{\Delta t \rightarrow 0} \frac{p_1(t + \Delta t) - p_1(t)}{\Delta t} = -p_1(t)\lambda_{12} + p_3(t)\lambda_{31}$$

$$p_1'(t) = -p_1(t)\lambda_{12} + p_3(t)\lambda_{31} \text{ - уравнение Колмогорова для состояния S}_1.$$

Рассмотрим состояние S2. P2(t) – вероятность того, что система в S2.:

1. В момент t была в S2 и за dt ни перешла ни в S3, ни в S4
2. Система была в состоянии S1 и за время dt пришла к состоянию S2
3. В момент t – S4, и за время dt пришла в состояние S2.

$$p_2(t + \Delta t) = p_2(t)(1 - \lambda_{24}\Delta t - \lambda_{23}\Delta t) + p_1(t)\lambda_{12}\Delta t + p_4(t)\lambda_{42}\Delta t$$

$$\lim_{\Delta t \rightarrow 0} \frac{p_2(t + \Delta t) - p_2(t)}{\Delta t} = -p_2(t)\lambda_{24} - p_2(t)\lambda_{23} + p_1(t)\lambda_{12} + p_4(t)\lambda_{42}$$

$$p_2'(t) = -p_2(t)\lambda_{24} - p_2(t)\lambda_{23} + p_1(t)\lambda_{12} + p_4(t)\lambda_{42}$$

3. Найдем вероятность того, что система находится в состоянии S3:

$$p_3'(t) = -p_3(t)\lambda_{31} - p_3(t)\lambda_{34} + p_2(t)\lambda_{23}$$

4. Найдем вероятность того, что система находится в состоянии S4:

$$p_4'(t) = -p_4(t)\lambda_{42} + p_2(t)\lambda_{24} + p_3(t)\lambda_{34}$$

В результате получаем систему уравнений Колмогорова:

$$\begin{cases} p_1' = -p_1\lambda_{12} + p_3\lambda_{31} \\ p_2' = -p_2\lambda_{24} - p_2\lambda_{23} + p_1\lambda_{12} + p_4\lambda_{42} \\ p_3' = -p_3\lambda_{31} - p_3\lambda_{34} + p_2\lambda_{23} \\ p_4' = -p_4\lambda_{42} + p_2\lambda_{24} + p_3\lambda_{34} \end{cases}$$

Интегрирование этой системы дает искомые вероятности состояний как функции времени. Начальные условия берутся в зависимости от того, какого было начальное состояние системы. Если в момент времени t система в S1, то начальное условие при t = 0: P1=1, P2=0, P3=0, P4=0 – *условие нормировки*, т.е. сумма всех вероятностей равна 1. Из структуры полученных уравнений выводим правила построения уравнения Колмогорова.

1. В левой части каждого уравнения стоит произвольное вероятностное состояние, а правая часть содержит столько частей, сколько стрелок связано с данным состоянием.
2. Если стрелка направлена из состояния, то соответствующий элемент имеет “-”
3. Каждый член уравнения равен произведения плотности вероятности перехода (*интенсивность*), соответствующей данной стрелке, умноженной на вероятность того состояния, из которого приходит стрелка.

Рассмотрим систему массового обслуживания с отказами.

Будем формировать состояние системы по числу занятых каналов (по числу заявок).

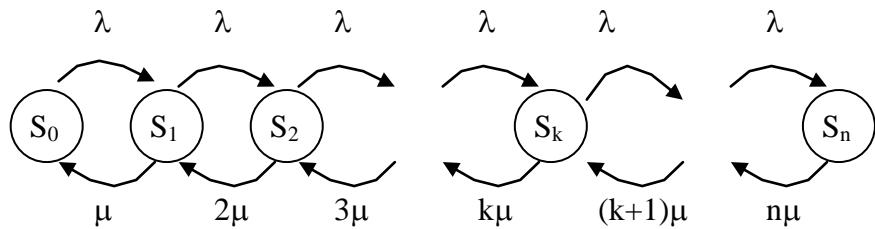
Обозначим

$S_0$  – все каналы свободны,

$S_1$  – один канал занят, остальные свободны

$S_k$  – занято  $k$  каналов, остальные свободны

$S_n$  - заняты все каналы.



Разметим граф, приставим у стрелок интенсивности соответствующих потоков событий.  
По стрелкам слева направо.

Пусть система в состоянии  $S_1$ , как только закончится обслуживание этой заявки, система перейдет в  $S_0$ . Интенсивность из  $S_2$  в  $S_1$  –  $\mu$

Предельные вероятности состояний  $P_0 P_n$  характеризуют устоявшейся режим работы:

$$\begin{cases} p_0' = -p_0\lambda + p_1\mu \\ p_1' = -p_1\lambda - p_1\mu + p_0\lambda + p_22\mu \\ p_2' = -p_2\lambda - p_22\mu + p_1\lambda + p_33\mu \\ p_k' = -p_k\lambda - p_kk\mu + p_{k-1}\lambda + p_{k+1}(k+1)\mu \\ p_n' = p_{n-1}\lambda - p_nn\mu \end{cases}$$

$$p_0 = \frac{1}{1 + \frac{\lambda/\mu}{1!} + \frac{(\lambda/\mu)^2}{2!} + \dots + \frac{(\lambda/\mu)^n}{n!}}$$

•  
•  
•

$$p_k = \frac{(\lambda/\mu)^k}{k!} p_0$$

$\lambda/\mu = \rho$  - среднее число заявок, приходящих в систему за среднее время обслуживания одной заявки.

$$p_0 = \left[ 1 + \frac{\rho}{1!} + \frac{\rho^2}{2!} + \dots + \frac{\rho^n}{n!} \right]^{-1}$$

$$p_k = \frac{\rho^k}{k!} p_0$$

- *Вероятность отказа - вероятность отказа* – вероятность того, что все  $n$  каналов заняты

$$p_{OTK} = p_n = \frac{\rho^n}{n!} p_0$$

- *относительная пропускная способность  $Q$*  – вероятность того, что заявка будет принята к обслуживанию

$$q = 1 - p_n$$

- среднее число заявок, обслуженных в единицу времени

$$A = \lambda q$$

Полученные соотношения могут рассматриваться как *базисная модель оценки характеристик производительности* системы. Входящий в эту модель параметр  $\lambda = 1 / t_{\text{обработки}}$ , является усредненной характеристикой пользователей. Параметр  $\mu$  является функцией технических характеристик компьютера (вычислительных средств) и решаемых задач. Эта связь должна быть установлена с помощью интерфейсной модели.

В простейшем случае, если время ввода-вывода информации мало по сравнению со временем решения этой задачи, то можно принять, что время решения равно  $1 / \mu = N_{\text{пр}} / V_{\text{пр}}$ .

$N_{\text{пр}}$  – среднее число операций, выполняемых процессором при решении одной задачи  
 $V_{\text{пр}}$  – среднее быстродействие процессора, измеряется в операциях в секунду.

Задание на Лабу:

определить среднее время нахождения системы в состояниях при установленном режиме работы. Должна строиться матрица (max 10x10). Количество состояний вводится пользователем: 2 – матрица 2x2. Потом вводятся все состояния:

	S1	S2
S1	2	1.5
S2	0	3

При вычислении – расчет времен пребывания в каждом расстоянии.

На практике далеко не все случайные процессы являются Марковскими или близкими к ним. В СМО поток заявок не всегда Пуассоновский, ещё реже наблюдается показательное или близкое к нему распределение времени обслуживания.

Для произвольных же потоков событий, переводящих систему из состояния в состояние, аналитические решения получены только для отдельных частных случаев. Когда

построение аналитической модели по той или иной причине трудно выполнимо, применяют метод статистических испытаний (*Монте-Карло*).

**Идея метода:** вместо того, чтобы описывать случайное явление с помощью аналитической зависимости производится «розыгрыш», т.е. происходит моделирование случайного явления с помощью некоторой процедуры, дающей случайный результат. Произведя такой розыгрыш  $n$  раз, получаем статистический материал, т.е. множество реализаций случайного явления, которое потом можно обработать обычными методами математической статистики. Метод Монте-Карло предложил Фон-Нейман в 1948 году, как метод численного решения некоторых математических задач.

Рис 11.

Суть:

1. В единичном квадрате – любая поверхность  $S$ .
2. Любым способом получаем 2 числа:  $X_i, Y_i$ , подчиняющихся равномерному закону распределения на интервале от 0 до 1.
3. Считаем, что одно число – координаты точки по  $X$ , другое – по  $Y$ . Анализируем, принадлежит ли точка поверхности. Если да, то счетчик на 1.
4. Процедура генерации 2 случайных чисел и проверки принадлежности точки поверхности повторяется  $N$  раз
5. Площадь - как отношение количества сгенерированных к количеству попавших точек.

$$\text{Погрешность} - \varepsilon \leq \sqrt{\frac{1}{n}}$$

*Преимущества* метода статистических испытаний в его универсальности, обуславливающей возможность всестороннего статистического исследования объекта, однако, для реализации этой возможности нужны довольно полные статистические сведения о параметрах элементов.

К недостаткам относится большой объем требуемых вычислений, равный количеству обращений к модели. Поэтому вопрос выбора величины  $n$  имеет важнейшее значение. Уменьшая  $n$  – повышаем экономичность расчетов, но одновременно ухудшаем их точность.

### **Способы получения последовательностей случайных чисел.**

На практике – три основных способа:

1. *аппаратный* – случайные числа вырабатываются специальной электронной приставкой, генератором, служащей как правило в качестве одного из внешних устройств компьютера. Реализация данного способа не требует дополнительных вычислительных операций по выработке случайных чисел, а необходима только операция обращения к внешнему устройству. В качестве физического эффекта, лежащего в основе таких генераторов, чаще всего используют шумы в электронных приборах.

рис 12.

2. *табличный* – случайные числа оформляются в виде таблице и помещаются в оперативную или внешнюю память.

3. алгоритмический – основан на формировании случайных чисел с помощью специальных алгоритмов.

выбрать порядка 1000.

выбрать критерий, чтобы определить, какой лучший – табличный или алгоритмический.

---

#### Моделирование. 15.10.04.

#### Простейшие алгоритмы генерации последовательности псевдослучайных чисел.

Наиболее часто используются – криптография, статистика, поиск объектов из космоса (нефть, метеориты).

Одним из первых способов получения последовательности псевдослучайных чисел было выделение значения дробной части у многочлена первой степени  
 $Y_n = Ent(an+b)$ .

Если  $n$  пробегает по ряду натуральных чисел, то поведение  $Y_n$  выглядит весьма хаотично. Карл Якоби доказал, что при рациональном  $a$ , множество значений  $Y_n$  конечно, а при иррациональном – бесконечно и всюду плотно в интервале  $[0,1]$ .

*Критерий равномерности распределения любой функции от натурального ряда чисел.* Это свойство эргадичности – среднее по реализации псевдослучайных чисел равно среднему по всему их множеству с вероятностью 1. Такие результаты далеки от практики получения последовательностей псевдослучайных чисел, т.к. теорема Якоби относится к действительным числам, которые не могут быть использованы при вычислениях, потому что иррациональные действительные числа требуют для своей записи бесконечного числа знаков.

Попытки замены настоящего иррационального числа его приближением на компьютере опасна, т.к. полученные последовательности оканчиваются циклом с коротким периодом.

Способы генерации:

1. 1946 год. Фон Нейман. Каждое последующее случайное число образуется возведением предыдущего в квадрат и отбрасыванием цифр с обоих концов. Не выдержал проверки
2. Лемер.  $g_{n+1} = kg_n + (c \bmod m)$   
Для подбора коэффициентов потрачены десятки лет. Подбор почти иррациональных почти ничего не дает.  
Выбрав закон генерации .... при просчете с плавающей точкой 4 байта, получают псевдослучайные числа, обязательно заканчивающиеся циклами с периодом длиной всего лишь 1225 или 147, в зависимости от начального заполнения.
3. Разумнее в целых числах. Установлено, что при  $c=0$  и  $m=2^n$ , наибольший период достигается при  $k=3+8i$  или  $k=5+8i$  и при нечетном начальном числе.  
По данному алгоритму в 1972 IBM на 360 разработала программу. А в 1977 году Форсайд показал, что тройки чисел такой последовательности лежат на 15 параллельных плоскостях.

Лучше использовать несколько генераторов и смешивать их значения. Если разные генераторы независимы, то сумма их последовательностей обладает дисперсией равной сумме дисперсий.

В системах программирования обычно используют конгруэнтные генераторы, по алгоритму предложенному в США.

4. Метод целочисленной арифметики. Работу написал Зейнеман. Использовали Фибоначчи, брали в своем алгоритме только последнюю цифру числа и получали последовательности.

RANDOMIZE 231

Y=RND

RANDOMIZE 231

X=RND

Х не равно Y, т.к. устанавливаются только 2 байта.

Для увеличения периода последовательности, используют

```
FUNCTION Rand(x, y)
x = RND(-x)
y = RND(-y)
IF y = 0 THEN y = RND(-y)
RAND = (x+y) mod 1
END FUNCTION
```

Период функции  $2^{24}$  ( $2^{41} - 1$ ).

Но свойства этого ряда не будут такими же, как у X, Y.

При создании с помощью встроенного генератора случайных объектов, имеющих число состояний больше, чем у генератора, его приходится использовать несколько раз, переустанавливая по заранее заданному ключу.

```
FOR I = 1 TO 5
X = RND(-gamma(i))
FOR J= 0 TO 32
    SWAP map(j), map(32xRND)
    NEXT J
NEXT I
```

Здесь происходит перестановка 33-х элементов массива map, которая может быть сделана  $2^{11}$  способами. При длине генератора  $2^{24}$  надо запустить не менее 5 раз, чтобы реализовать все варианты перестановки.

Для получения значения случайной величины из последовательности случайных чисел с заданным законом распределения, обычно используют одно или несколько значений равномерно распределенных случайных чисел. Псевдослучайные равномерно распределенные числа получаются в компьютере программным способом с помощью некоторого рекуррентного соотношения. Это означает, что каждое последующее число образуется из предыдущего (или из группы предыдущих) путем реализации некоторого алгоритма, состоящего из арифметических и логических операций.

Процедура на языке Фортран, предназначенная для последовательности равномерно распределенных случайных числа на интервале [0,1] с помощью мультипликативного конгруэнтного метода для 32 разрядного компьютера.

```

SUBROUTINE RANDUM(IX, IY, RN)
IY = IX*1220703125
IF (IY) 3,4,4
IY = IY+2147483647+1 // метод 3
RN = IX // метод 4
RN = RN * 0.4656613E-9
IX = IY
RETURN
END

```

IX – число, которое при первом обращении должно содержать нечетное целое число, состоящее менее чем из 9 цифр.

```

Function Rand(n:integer):double
Var S,N:double, i:integer;
Begin
If n = 0 then
Begin
X = m34;
Rand := 0;
Exit;
End;
S:=-2.5;
For i:=1 to 5 do
Begin
X:=5.0*x;
If x>=m34 then x :=x-m37;
If x>=m36 then x:= x - m36;
If x >=m35 then x:= x- 35;
W := x/ m35;
If n = 1 then
Begin
Rand :=W; exit;
End
S := S +W;
End
S:=S*1.54919;
Rand:=(sqr(S)-3.0)*S*0.01+S
End;

Begin
R:= Rand(0);
For I = 0 to 200 do
Begin
Writeln (Rand(2):12:10);
Readln;
End;
End;
0 – равномерное распределение
1 – гауссово.

```

Для имитации нормального распределения на интервале [a,b] используются обратное преобразование функции плотности.

$$\frac{x-a}{b-a} = R \text{ отсюда } x = A + (B - A)R, R <= 1$$

В основе построения программы, генерирующей случайные числа, с законом отличном от равномерного, лежит метод преобразования последовательности случайных чисел с равномерным законом распределения в последовательность случайных чисел с заданным законом распределения. Метод основан на теореме, утверждающей, что некоторая случайная величина X: рис.2 имеет плотность распределения  $F(x)$ , где R – случайная величина, равномерно распределенная в интервале [0,1]. Значение случайной величины, распределенной по показательному закону, может быть вычислено по

$$1 - e^{-\lambda x} = R$$

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

Распределение Пуассона.

Относится к числу дискретных, т.е. таких, при которых переменная может принимать лишь целое значение с математическим ожиданием и дисперсией равными lambda. Для генерируемых пуассоновых переменных, использую метод Точера, в основе которого лежит генерируемая случайная переменная  $R_i$ , равномерно распределенная до тех пор, пока не станет справедливо следующее соотношение.

$$\prod_{i=0}^x R_i \geq e^{-\lambda} > \prod_{i=0}^{x+1} R_i$$

При получении случайной величины, функция распределения которой не позволяет найти решения уравнения в явной форме, можно произвести кусочно-линейную аппроксимацию и затем вычислить приближенные значения корня. Кроме того, при получении случайной величины часто используют те или иные свойства распределения.

Распределение Эрланга – 2 параметра: lamda, k. При вычислении случайной величины, воспользуемся тем, что поток Эрланга может быть получен прореживанием k раз. Или же потока Пуассона. Т.е. достаточно получить k значений случайной величины, распределенной по показателям и усреднить их.

$$x = \frac{1}{k} \left( \sum_{i=1}^k \left( -\frac{1}{\lambda} \ln(1 - R_i) \right) \right) = -\frac{1}{k\lambda} \sum_{i=1}^k \ln(1 - R_i)$$

Нормально распределенная случайная величина может быть получена как сумма большого числа случайных величин, распределенных по одному и тому же закону и одними и теми же параметрами.

Случайная величина X имеющая нормальное распределение с математическим ожиданием  $M_x$  и среднеквадратичным отклонением  $\sigma_x$  может быть получена по следующей формуле:

$$x = \sigma_x \cdot \sqrt{\frac{12}{N}} \cdot \left( \sum_{i=1}^N R_i - \frac{N}{2} \right) + M_x$$

Для сокращения вычислений на практике принимают N=12, что дает хорошее приближение.

## Лаба 4.

Классификация систем массового обслуживания.

СМО классифицируются по следующим принципам:

1. закон распределения заявок
2. числу обслуживающих приборов
3. закону распределения времени обслуживания обслуживающих приборов
4. числу мест в очереди
5. дисциплине обслуживания

Для краткости записи при обозначении любой системы массового обслуживания:

**A/B/C/D/E**

где на место буквы ставится характеристики.

А - закон распределения времени между поступлением заявок. Наиболее часто используются следующие:

М – экспоненциальные

Е - эрландовские

Н – гиперэкспоненциальные

Г - гамма-распределение

Д - детерминированное распределение

Р – произвольное

В – Закон распределения времени обслуживания приборов СМО.  
приняты такие же обозначения

С – число обслуживающих приборов.

для одноканальных систем – 1

для многоканальных в общем случае – 1

Д – число мест в очереди

если число мест в очереди неограничено, то оно может опускаться. Для конечного числа мест в очереди – R/N.

Е - Дисциплина обслуживания. По умолчанию LIFO – в этом случае поле может опускаться.

Примеры:

М/М/1 – СМО с одним прибором, бесконечной очередью, экспоненциальным законом распределения, интервалом времени между поступлением заявок и временем обслуживания, дисциплина обслуживания FIFO.

Е/Н/l/r/LIFO - СМО с несколькими обслуживающими приборами, конечной очередью, законом распределения Эрланга интервалов между поступлением заявок, гиперэкспоненциальным законом распределения времени обслуживания заявок в приборах и дисциплиной обслуживания LIFO.

**G | G | 1**

СМО с несколькими приборами, бесконечной очередью, произвольными законами распределения времени между поступлением заявок и времени обслуживания, FIFO.

Для моделирования вычислительной системы наиболее часто используются следующие типы СМО:

**1. Одноканальная СМО с ожиданием.**

- один обслуживающий прибор с бесконечной очередью
- структура является наиболее распространенной при моделировании
- с той или иной степенью приближения с ее помощью можно моделировать практически любой узел вычислительной системы или ЛВС

**2. Одноканальная СМО с потерями.**

- один обслуживающий прибор с конечным числом мест в очереди
- если число заявок превышает число мест в очереди, то лишние заявки теряются
- используется при моделировании каналов передач данных в ВС и ЛВС

**3. Многоканальная СМО с ожиданием.**

- несколько параллельно работающих обслуживающих приборов с общей бесконечной очередью
- используется при моделировании групп абонентских терминалов, работающих в диалоговом режиме

**4. Многоканальная СМО с потерями.**

- несколько параллельно работающих обслуживающих приборов с общей очередью, число мест в которой ограничено
- часто используется, как и (2), при моделировании каналов связи

**5. Одноканальная СМО с групповым поступлением заявок**

- один обслуживающий прибор с бесконечной очередью
- перед обслуживанием заявки группируются в пакеты по определенном признаку или правилам
- используется для моделирования центров коммутации

**6. Одноканальная СМО с групповым обслуживанием заявок**

- один обслуживающий прибор с бесконечной очередью
- заявки обслуживаются пакетами, составленными по определенному правилу
- используется для моделирования центров коммутации

Наименование	Обозначение	Схема
Одноканальная с ожиданием	$G / G / 1$	
Одноканальная с потерями	$G / G / 1 / r$	
Многоканальная с ожиданием	$G / G / l$	

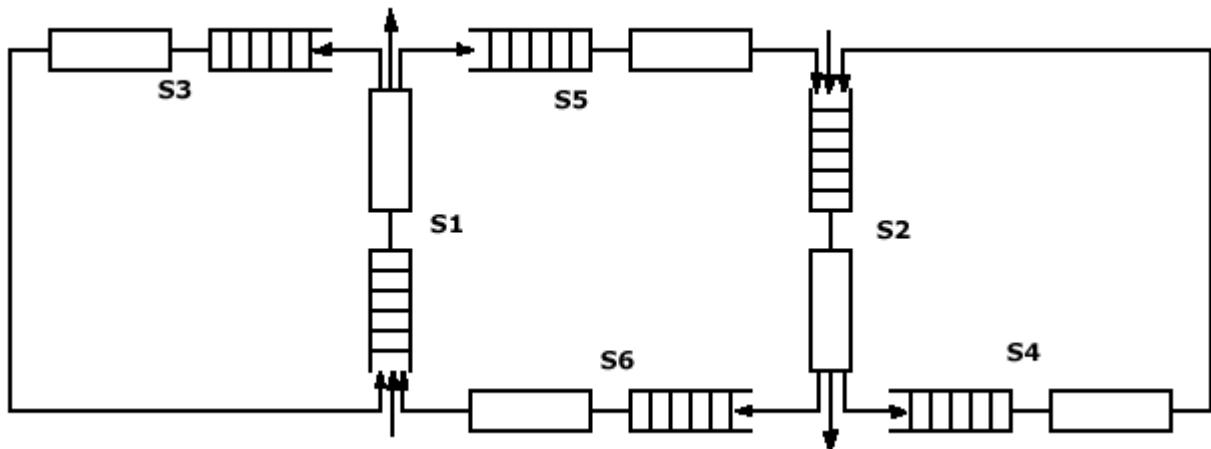
Многоканальная с потерями	$G / G / l / r$	
Одноканальная с групповым поступлением заявок	$Gr / G / 1$ $R$	
Одноканальная с групповым обслуживанием заявок	$G / Gr / 1$ $R$	

Вычислительные сети в целом могут быть представлены в виде СМО. Различают следующие типы сетей:

1. **Открытые**
2. **Замкнутые**
3. **Смешанные**

*Открытой* называется СМО, состоящая из m узлов, причем хотя бы в один из узлов сети поступают извне входной поток заявок и обязательное имеется хотя бы один сток заявок из сети.

Для открытых сетей характерно то, что интенсивность поступления заявок не зависит от состояния системы, т.е. от числа заявок уже поступивших в сеть. Такие сети как правило используются для моделируемых ВС, работающих в неоперативном режиме.



S1, S2 – моделируют работу узлов коммутации

S3, S4 – работу сервера

S5, S6 – работу межузловых каналов

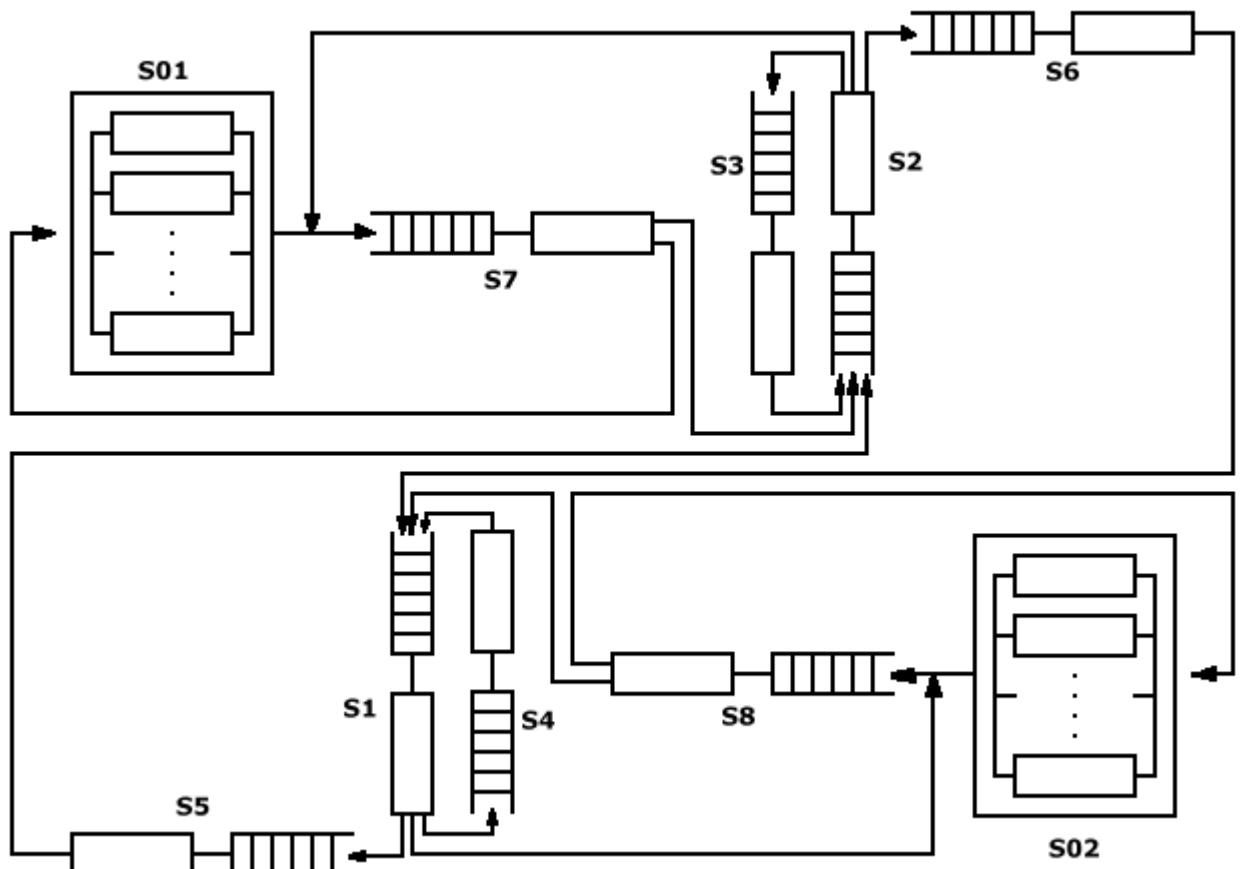
В сети циркулируют 2 потока заявок:

Каждая заявка поступает на вход соответствующего узла коммутации, где определяется место обработки. Затем заявка передается на «свой» сервер или по каналу связи на соседний, где обрабатывается, после чего возвращается к источнику и покидает сеть.

*Замкнутой* называется СМО с множеством узлов m без источника и стока, в которой циркулирует постоянное число заявок.

Замкнутые СМО используются для моделирования таких ВС, источниками информации для которых служит абонентский терминал, работающий в диалоговом режиме. В этом случае каждая группа абонентских терминалов представляется в виде многоканальной СМО с ожиданием и включается в состав устройств сети.

Простой режим работы диалоговых абонентов: абоненты не производят никаких действий, кроме посылки заданий в вычислительную систему и обслуживание полученного ответа.



**S01, S02** – группа абонентских терминалов

**S7, S8** – каналы связи с абонентами

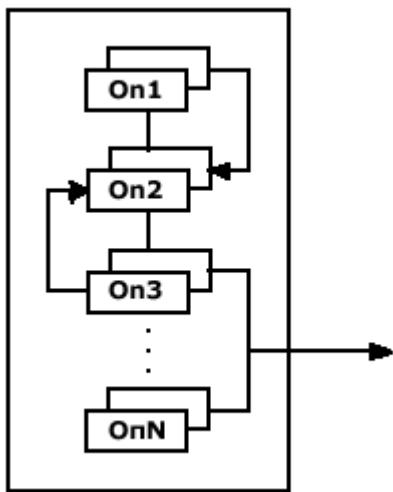
**S1, S2** – узлы коммутации

**S3, S4** – серверы

**S5, S6** – каналы межузловой связи

Абоненты с терминалов посылают запросы, которые по каналам связи поступают на узлы коммутатора, а оттуда на обработку на свой или соседний сервер.

При сложном режиме диалога работа абонентов представляется в виде совокупности операций некоего процесса, называемого *технологическим*. Каждая операция технологического процесса моделируется СМО. Часть операций предусматривает обращение в ВС. Причем, часть операций может предусматривать обращение к ВС, а другая часть может замыкаться сама на себя.



*Смешанной* называется СМО, в которой циркулируют несколько различных типов заявок (трафика). Причем относительно одних типов заявок сеть замкнута, а относительно других – открыта.

С помощью смешанных сетей моделируются такие ВС, часть абонентов которых работает в диалоговом, а часть в неоперативном режиме. Для диалоговых абонентов также различают простой и сложный режим работы. Часто смешанные СМО моделируют ВС, в которых сервер дополнительно загружается задачей, решаемыми на фоне работы самой сети.

---

#### Моделирование. 22.10.04.

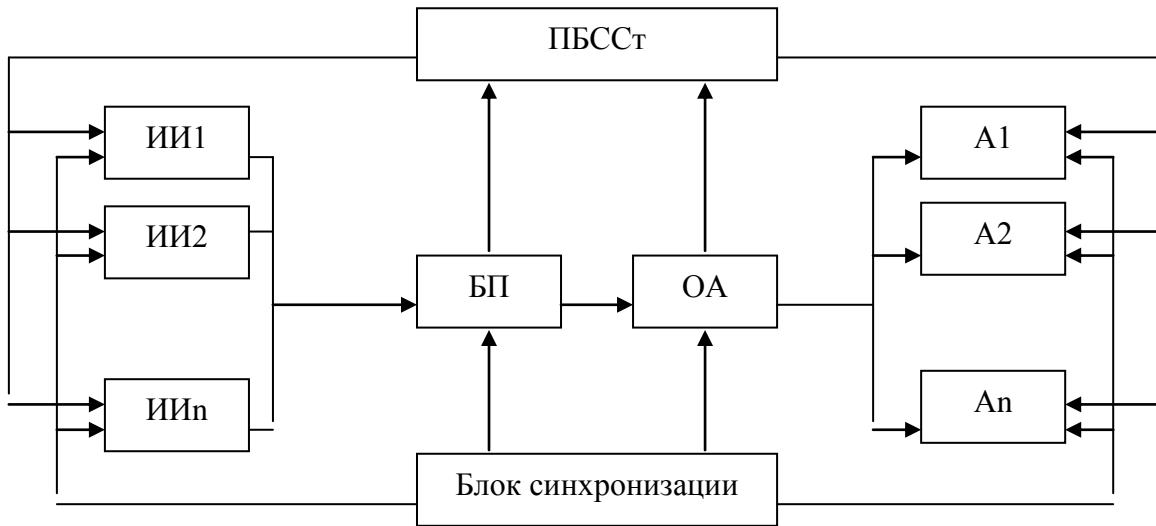
##### Методика построения программной модели.

Для разработки программной модели исходная система должна быть представлена как стохастическая. Это решается несколькими факторами – информация поступает от внешних признаком. Длительность обработки различных признаков информации может быть в общем случае различной. Все эти воздействия могут отображаться каким-то генератором сообщений. А весь комплекс вычислительных средств в виде обслуживающих устройств.

рис. 1.

Источники информации подают на вход буферной памяти независимо друг от друга поток сообщений. Закон появления этих сообщений произволен, но обязательно задан наперед. В буферной памяти сообщения как правило записываются при поступлении, и выбираются по одному в обслуживающий аппарат, как правило по принципу FIFO. Длительность обработки одного сообщения в обслуживающем аппарате в общем случае может быть случайной. Но закон обработки должен быть задан заранее.

Быстродействие обслуживающего аппарата ограничено, поэтому на входе как в обслуживающий аппарат, как и в буферную память (если она конечна), в общем случае возможно сложение данных.



### **Моделирование потока сообщений.**

Поток сообщений обычно лимитируется моментами появления очередного сообщения в потоке.

Текущий момент времени появления очередного сообщения:

$$t_i = \sum_{k=1}^{i-1} T_k + T_i$$

где  $T_i$  – интервал времени между появлением  $i$ -го и  $(i-1)$ -го сообщения.

#### **Процедура.**

обращение к процедуре выражения случайного числа Rnd

$$T_i = -\frac{1}{\lambda} \ln(1 - R_i)$$

$$T = T + T_i$$

Вид распределения	Выражение
равномерное на $[a,b]$	$T_i = a + (b - a)R$
нормальное	$T_i = \sigma_x \sqrt{\frac{12}{n}} \left( \sum_{i=1}^n R_i - \frac{n}{2} \right) + M_x$
экспоненциальное	$T_i = -\frac{1}{\lambda} \ln(1 - R)$
Эрланга	$T_i = \frac{1}{k\lambda} \sum_{i=1}^k \ln(1 - R_i)$

### **Моделирование работы Обслуживающего Аппарата.**

Программа, имитирующая работу обслуживающего аппарата – набор процедур, вырабатывающих случайные отрезки времени, соответствующие длительностям обслуживания требований.

Например, если требования от источника обрабатываются в OA по нормальному закону с параметрами  $M_x$  и  $\sigma_x$ , то длительность обработки  $i$ -ого требования:

$$t_{iAD} = M_x + \left( \sum_{i=1}^{12} R_i - 6 \right) \cdot \sigma_x$$

### Схема алгоритма имитатора.

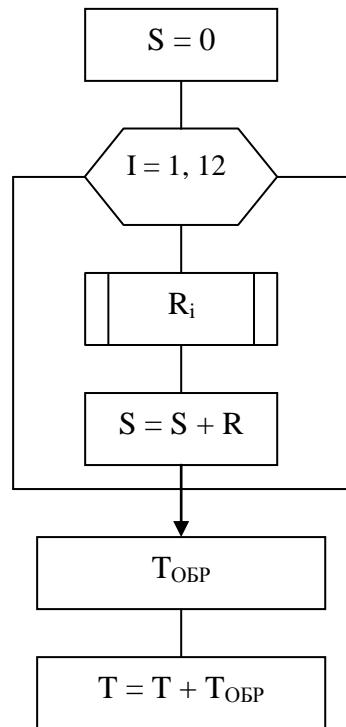
$R_i$  – случайное число с равномерным законом распределения  
 $T_{\text{ОБР}}$  – время обработки очередного сообщения

$T$  – время освобождения ОА

$XM$  – Мат ожидание для заданного закона обратки

$DX$  – СКО для заданного закона обратки

$$T_{iAD} = XM + (S - 6) * DX$$



### Моделирование работы абонента.

Абонента можно рассматривать как ОА, поток информации на который поступает от процессора. Для моделирования работы абонентов необходимо вырабатывать длительности обслуживания требований. Кроме того, абонент сам может быть источником заявок, претендую на те или иные ресурсы вычислительной системы. Эти заявки могут имитироваться с помощью генератора сообщений по наперед заданному закону. Таким образом, абонент либо имитируется как ОА, либо как генератор.

### Моделирование работы буферной памяти.

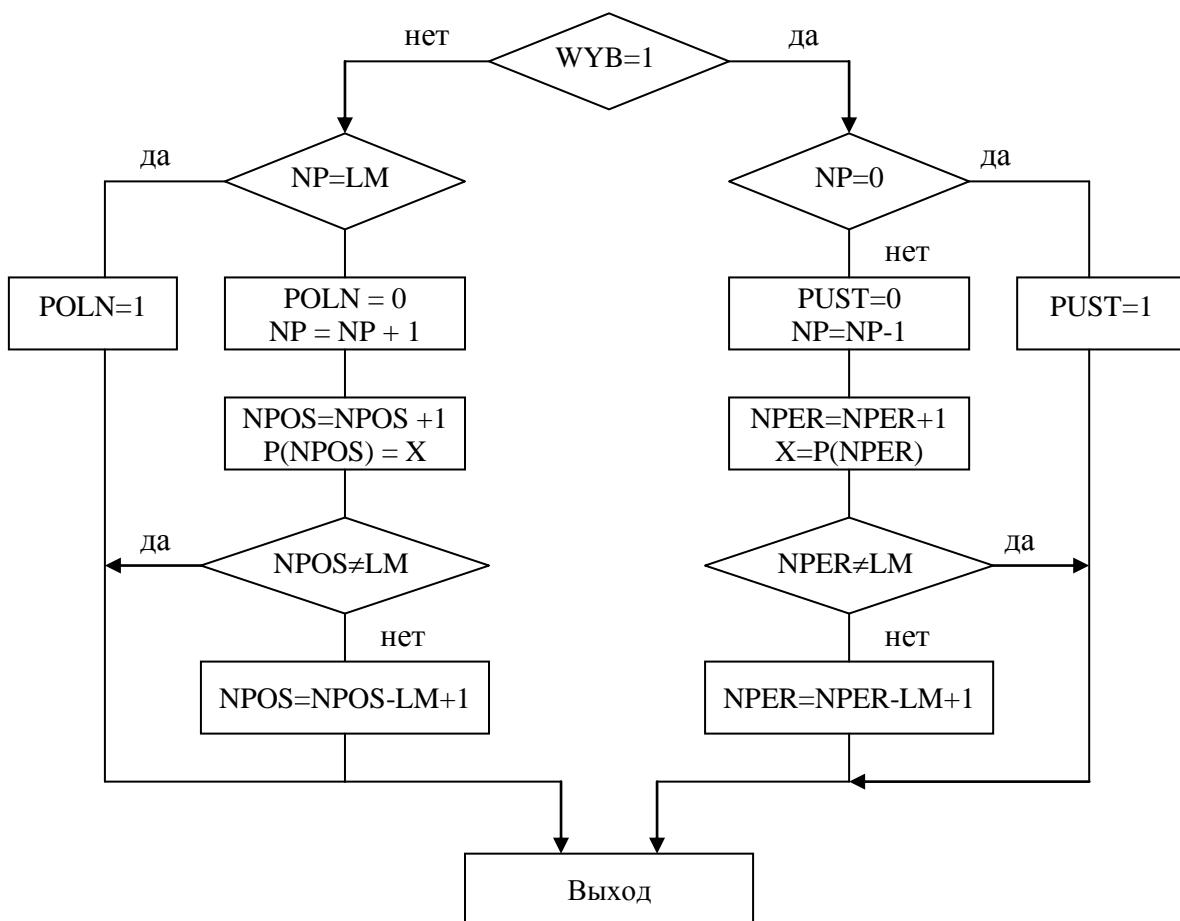
Блок буферной памяти должен производить запись и считывания числа, выдавать сигналы переполнения и отсутствия данных. В любой момент времени располагать сведениями о количестве требований в блоке. Сама запоминающая среда имитируется некоторым одномерным массивом, размер которого определяет размер БП. Каждый элемент этого массива может быть либо свободен, либо занят. В качестве эквивалента требования присваивается значение времени появление этого явления.

<ol style="list-style-type: none"> <li>1. Анализ признака режима. (запись)</li> <li>2. Анализ на переполнение есть → блок статистики нет → запись информации по текущему адресу, изменение текущего адреса на 1</li> </ol>	<p>Чтение</p> <ol style="list-style-type: none"> <li>1. Анализ наличия сообщения в БП нет → запись в БС есть → чтение по текущему адресу, уменьшение текущего адреса (на 1)</li> <li>2. выход</li> </ol>
--	--

Алгоритм:

Есть генератор, очередь, обслуживающий аппарат. Определяет оптимальную длину очереди. Оптимум – минимальная, при которой не теряется сообщение. Обслуживающий – по равномерному закону. Генерация – по своему закону. Параметры настраиваются.

Модель которая в простейшем (один генератор, одна очередь, один обслуживающий аппарат) приборе обслуживания ищет оптимальную длину очереди – длина очереди минимальная, при которой не теряются сообщения. У всех будет генераторы работают по своему закону. А обслуживающий аппарат по равномерному. Параметры настройки должны вводиться.



#### Моделирование. 29.10.04.

#### Программа сбора статистики.

Задача сбора статистики заключается в накоплении численных значений, необходимых для вычисления статистических оценок заданных параметров моделируемой системы.

При моделировании работы простейшей СМО обычно интерес представляет среднее время ожидания в очереди. Для каждого сообщения время ожидания в очереди равно разности между моментами времени, когда оно было выбрано на обработку ОА, и моментом времени, когда оно пришло в систему от источника информации.

Суммируя значение количества сообщений буферной памяти через небольшие промежутки времени и разделив полученную сумму на число суммирований, получим *среднее значение длины очереди в памяти*.

*Коэффициент загрузки обслуживающего аппарата* – отношение времени непосредственной работы ОА, к общему времени моделирования. Вероятность потери сообщения определяется как количество потерянных сообщений к общему числу.

### **Разработка управляющей программы.**

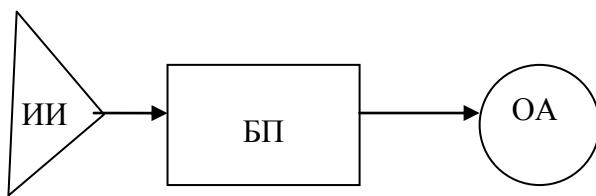
Если программа работы источника обслуживающего аппарата или памяти - имитирует работу отдельных устройств, то *управляющая программа* имитирует алгоритм взаимодействия элементов системы. Управляющая программа реализуется в основном по двум принципам:

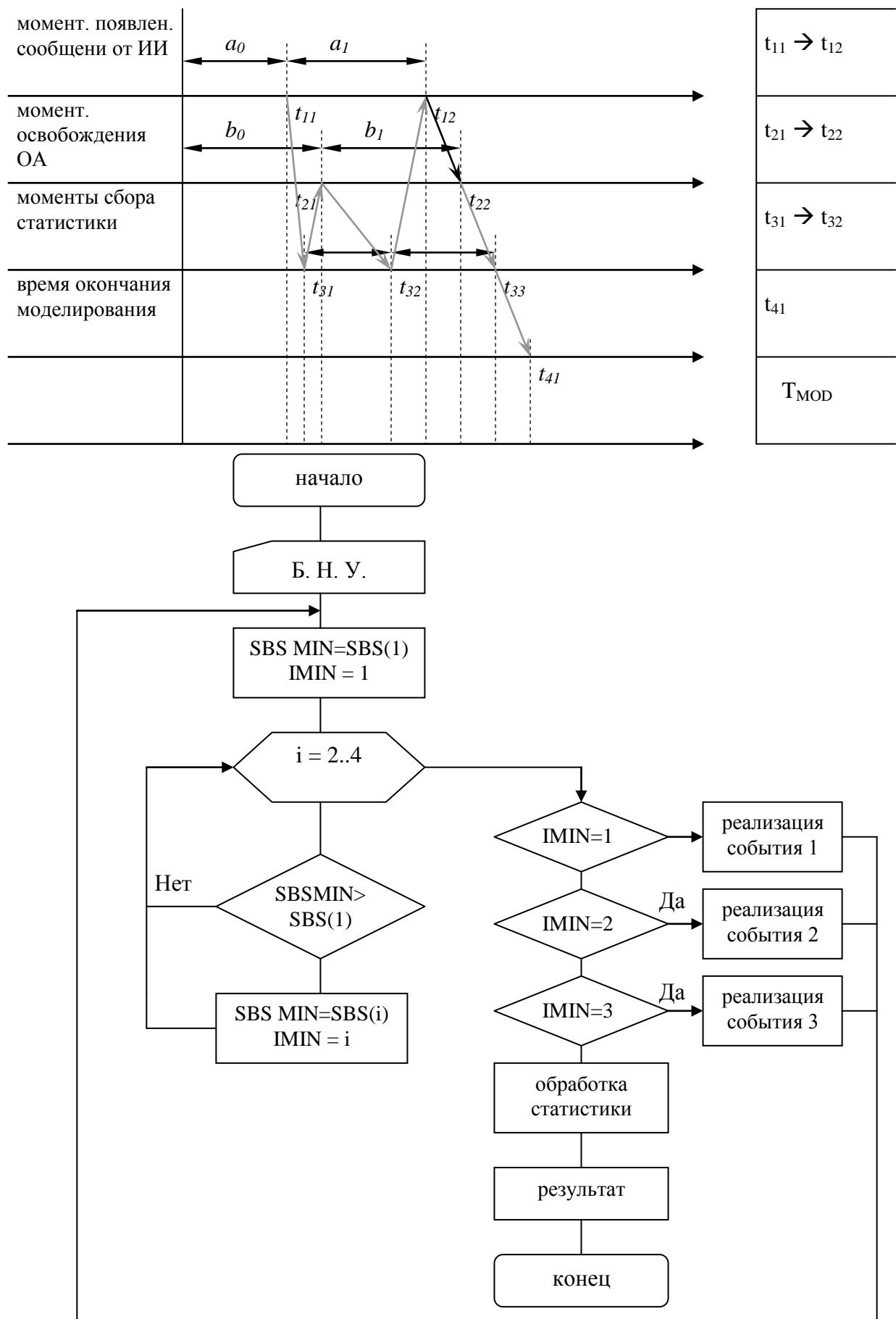
#### **1. принцип $\Delta t$**

Заключается в последовательном анализе состояний всех блоков в момент  $t+dt$  по заданному состоянию блоков в момент  $t$ . При этом новое состояние блоков определяется в соответствии с их алгоритмическим описанием, с учетом действующих случайных факторов, задаваемых распределением вероятности. В результате такого анализа принимается решение о том, какие общесистемные события должны имитироваться программной моделью на данный момент времени. Основной недостаток – значительные затраты машинного времени на реализацию моделирования. При недостаточно малом  $dt$  появляется опасность пропуска отдельных событий, что может привести к неверным результатам моделирования.

#### **2. событийный принцип**

Характерной свойство систем обработки информации заключается в том, что состояние отдельных устройств изменяется в дискретные моменты времени, совпадающие с моментами поступлений сообщений в систему или окончанием решения задач, или возникновения аварийных ситуаций. Поэтому, моделирование и продвижение текущего времени производят, используя событийный принцип. При его использовании, состояние всех блоков имитационной модели анализируется лишь в момент появления какого-либо события. В момент наступления следующего события определяется минимальным значением из списка будущих событий, представляющего собой совокупность моментов *ближайшего изменения состояния* каждого блока системы.





$t_{11}, t_{12}$  – моменты появления сообщений на выходе генератора (источника информации)

$b_1$  – интервал времени обслуживания первого сообщения

$t_{3n}$  – момент сбора статистики

$t_{41}$  – момент окончания моделирования

SBS – список будущих событий.

## Методика реализации событийной модели.

1. Для всех активных блоков (блоков, порождающих события) заводится свой элемент в одномерном массиве – списке будущих событий.
2. В качестве подготовительной операции список в будущих событий заносится время ближайшего события от любого активного блока. Активизируя программу-имитатор, ИИ вырабатывает псевдослучайную величину  $a_0$ , определяющую момент появления первого сообщения  $t_{11}$ . Эту величину заносят в список будущих событий. Активизируя программный имитатор ОА, вырабатывают псевдослучайную величину  $b_0$ , определяющую момент времени  $t_{21}$  – в список будущих событий. Момент времени  $t_{31}$  (1ый сбор статистики) определяется равным стандартному шагу сбора  $t_{STAT}$ , и заносится в SBS  
В SBS заносится  $t_{41}$  – время окончания моделирования.  
Подготовительная часть на этом закончена и начинается протяжка модельного времени.
3. В SBS определяется минимально числовое значение и его номер.
4. Реализуется событие, порождаемое блоком с соответствующим номером, т.е. модельное время =  $t_{11}$ . Далее реализуется событие с номером 1, связанное с появлением нового сообщения в ИИ. Реализация этого события заключается в том, что само сообщение записывается в память, а с помощью имитатора ИИ, вырабатывается момент появления следующего события  $t_{12}$ . Это время помещается в соответствующую ячейку SBS место  $t_{11}$ .  
Затем вновь организуется поиск минимального элемента в SBS. Для данного примера реализуется событие 3, после чего выражение момента времени  $t_{32}$  – новое время сбора статистики. Так до тех пор, пока минимально время не станет равным  $t_{41}$ .

Два описанных принципа являются универсальными алгоритмами протяжки модельного времени. Для некоторых предметных областей один принцип может работать быстро и без потерь событий, а другой будет работать при этом очень медленно. Выбор метода необходимо производить исходя из распределения событий во времени. В реальных системах распределение событий как правило неоднородно – события группируются по времени. Образование групп связано с наступлением какого-либо значимого события, которое инициирует определенную последовательность действий соответствующими событиями, имеющими высокую плотность на определенном интервале времени. Такой интервал – *пиковый*. А распределение событий – *квазисинхронными*. Пример – цифровая сеть, в которой синхронизирующие сигналы переключают большое количество триггеров.

Дельфт.

Данный алгоритм был специально разработан для сложных дискретных систем, в которых присутствует квазисинхронной распределение событий. Особенность данного метода – автоматическая адаптация к распределению событий. Метод реализуется таким образом, что на пиковых интервалах он приближается к метода ДельтаT, а вне них к событийному методу, с большим шагом. Алгоритм основан на использовании иерархической структуры циркулярных списков.

Рис.1.

Список уровня содержит N1 элементов и описывает планируемые события в пиковых интервалах. Число N1 представляет собой разбиение пикового интервала на более мелкие участки, с каждым из которых связан список событий, произошедших за этот интервал времени. Списки второго уровня и выше – являются масштабирующими списками, количество элементов которых равно константному значению N2, которое характеризует коэффициент масштабирования временных интервалов. Собственно, алгоритм протяжки модельного времени заключается в последовательном поиске не пустых элементов в самом верхнем циркулярном списке с большим шагом и дальнейшим поиском на более низкие уровни.

### **Языки имитационного моделирования.**

Могут использоваться следующие языки:

1. универсальные алгоритмические языки высокого уровня.
2. функциональные языки. процессно-ориентированные языки
3. проблемно-ориентированные языки и системы моделирования

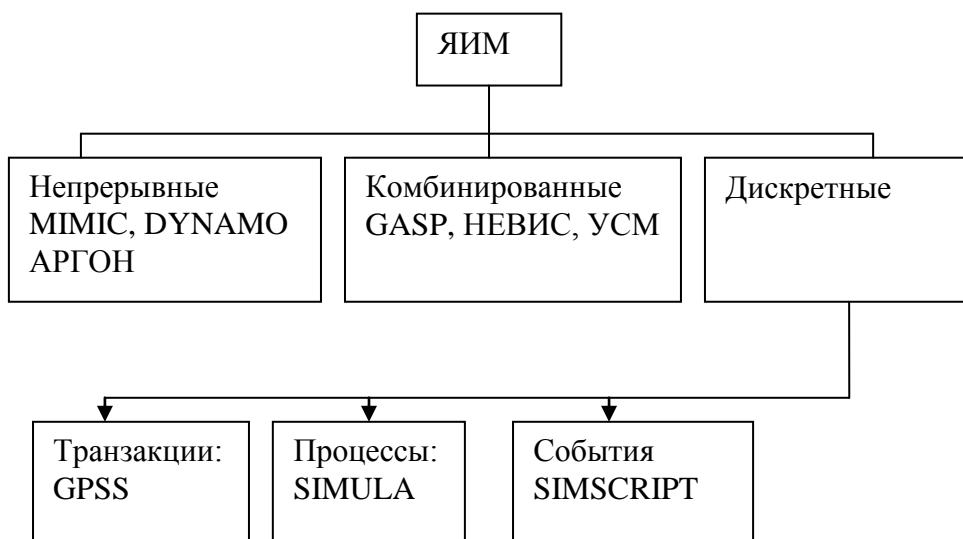
### **Основные методы построения языка РДО.**

Качество языков характеризуется:

1. удобство описания процесса функционирования
2. удобство ввода исходных данных, варьирование структуры, алгоритмов работы и параметров модели.
3. эффективность анализа и вывод результатов моделирования
4. простотой отладки
5. доступностью восприятия и использования языка

Все современные языки моделирования определяют поведение систем во времени, с помощью событийного алгоритма или его модификации.

Классификация языков моделирования по принципу формирования системного времени.



Непрерывное представление систем сводится к дифференциальным уравнениям. Если переменные модели принимают дискретные модели, то уравнение – разностное.

## GASP

События 2х типов:

1. события, зависящие от состояния
2. события, зависящие от времени

### **Формальное описание динамики моделируемого объекта.**

Будем считать, что любая работа в системе совершается путем выполнения активности. Активность является наименьшей единицей работы. Ее рассматривают как единый дискретный шаг. Она имеет свое время выполнение. Активность является единым динамическим объектом, указывающим на совершение единицы работы. Процесс – это логически связанный набор активностей. Активности проявляются в результате совершения событий. Событие – это мгновенное изменение состояния некоторого объекта системы. Рассмотренные объекты - активности, процессы и события, являются конструктивными элементами для динамического описания поведения системы. На их основе строятся языки моделирования таких систем. В то время, когда динамическое поведение системы формируется в результате большого числа взаимодействующих процессов, сами эти процессы образуют относительно небольшое число классов, чтобы описать поведение системы, достаточно описать поведение каждого класса процессов и задать значение атрибутов для конкретных классов.

Построение модели – 2 взаимные задачи:

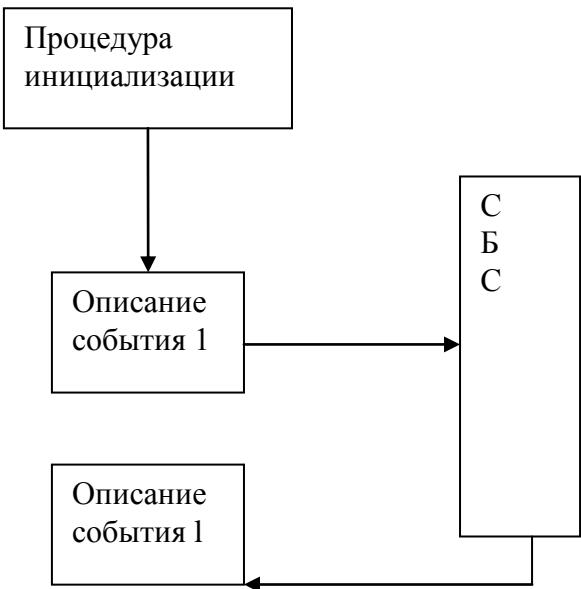
1. необходимо задать правило, определяющее виды процессов, происходящие в системе.
2. указать значение атрибутов таких процессов или задать правила генерации этих процессов. При этом, как правило, системы описываются на определенном множестве детализации, в терминах множества описания процессов, каждый из которых включает множество правил и условий возбуждения активностей. Такое описание системы может быть детализировано на более низкий иерархический уровень представления, с помощью декомпозиции процессов.

Для отображения временного поведения системы, язык моделирования дискретных систем должен обладать средствами отображения времени. В реальной системе совместно выполняется несколько активностей, принадлежащих как связанным и несвязанным процессам. Имитация их действие должна быть строго последовательной. Модель системы можно рассматривать как модель описаний активностей, событий или процессов.

### **Языки ориентированные на события.**

Моделирующая программа организована в виде секций, они включают в себя события. Процедура событий состоит из набора операций, который в общем случае выполняется после завершения какой-либо активности. Выполнение процедуры синхронизируется во времени списком будущих событий.

Рис 2.



## **Языки ориентированные на процессы**

Моделирующая программа в виде набора описаний процессов, каждый из которых описывает один класс процессов. Описание процесса функционирования устанавливает описание атрибутов всех процессов. Синхронизация операций во времени с помощью СБС, который содержит точку возбуждения конкретного процесса.

Описание объектов моделирования.

Инициализация

- Описание процесса a1
- Описание процесса a2
- ...
- Описание процесса an

Все это программы моделирования

Результаты экспертных оценок, сравнение различных языков при моделировании широкого класса систем:

1. Возможности языка
  - SIMULA
  - SIMSCRIPT
  - GPSS
  - C
  - PASCAL
2. Простота применения
  - GPSS
  - SIMSCRIPT
  - SIMULA
  - C
  - PASCAL

3. Предпочтение пользователей
- SIMSCRIPT  
GPSS  
SIMULA  
PASCAL  
C

Сравнение универсальных и специализированных языков программирования при моделировании.

	Преимущества	Недостатки
Универсальные	<ol style="list-style-type: none"> <li>1. Минимум ограничений на выходной формат</li> <li>2. Широкое распространение</li> </ol>	<ol style="list-style-type: none"> <li>1. значительное время затрачиваемое на программирование</li> <li>2. значительное время на отладку</li> </ol>
Специализированные	<ol style="list-style-type: none"> <li>1. меньше затрат времени на программирование</li> <li>2. более эффективные методы выявления ошибок</li> <li>3. краткость, точность понятий, характеризующих имитируемые процессы.</li> <li>4. возможность заранее строить стандартные блоки, которые могут использоваться в любой имитационной модели</li> <li>5. автоматическое формирование определенных типов данных, необходимых именно в процессе ИМ.</li> <li>6. удобство накопления и представления выходной информации.</li> <li>7. эффективное использование ресурсов.</li> </ol>	<ol style="list-style-type: none"> <li>1. необходимость точно придерживаться определенных ограничений на форматы данных</li> <li>2. меньшая гибкость модели</li> </ol>

### **РДО – ресурсы данные операции**

Причина создания – требования универсальности ИМ относительно классов моделирования систем и процессов, легкости модификации модели, моделирование сложных систем управления.

Язык РДО – реализация так называемого интеллектуального подхода к ИМ. Это сравнительно новый подход, отойти от жесткого алгоритмического подхода в процессе принятия решения, и сделать процесс принятия решения максимально гибким по способам .. сложной дискретной системы.

В основе – продукционная система, состоит из 3 элементов – класса и отношение, правил управляющей структуры. Классы и отношения трактуются как БД, содержащая декларативные знания. Процедура – набор модифицированных продукционных правил типа «Если, ... то...». Управляющая структура – интерпретатор правил, управляющий выборкой правил. Условие – проверка правила, действие – изменение.

Достоинство системы – простота создания и понимания отдельных правил, простота пополнения и модификации.

Недостатки – неясность взаимных отношений правил, сложность оценки целостного образа знаний, крайне низкая эффективность обработки.

Для ИМ основным недостатком системы продукции является отсутствие времени, т.е. такие правила применимы только при моделировании статических объектов.

В РДО используют модифицированное правило:

Если <условие> то <событие 1> ждать (временной интервал) то <событие 2>

В РДО сложная дискретная система представляется в виде множества взаимодействующих между собой ресурсов. Ресурс – это элемент сложной системы, внутренней структурой которой можно пренебречь, в то время как наличие и свойство его как целого важны и существенны для описания. Каждый ресурс модели должен быть описан множеством параметров, которые могут быть следующих типов:

1. описательные, представляющие факты, внутренне присущие каждому ресурсу.
2. указывающие, используемые для имени (id)
3. вспомогательные, используемые для связи различных ресурсов, накопления статистики, графического вывода при имитации.

---

#### **Моделирование. 05.11.04.**

##### Лаба 5

###### Метро Бауманская

1. концептуальная модель – графическая схема термина СМО.
2. выделены параметры: входная информация, ограничения на параметры входного потока, внутренняя структура (количество,...), выходная информация.
3. формализация процесса в виде математической модели – выбирается либо типовая математическая модель, либо своя, с обоснованием и доказательством – почему можно промоделировать станцию именно так
4. результаты – в графическом виде.
5. текстовый отчет должен содержать информацию с рекомендациями по повышению производительности, пропускной способности

#### **General Purpose System Simulation (GPSS)**

GPSS – общепринятая система моделирования, как и любой язык, он содержит словарь и грамматику, с помощью которых легко могут быть разработаны точные модели систем определенного типа. Существуют версии 1, 2, V, PC. Любой пакет в нем построен в предположении что моделью сложной дискретной системы является описание ее элементов и логических правил, взаимодействия в процессе функционирования. Для определения класса моделирования системы можно выделить конечный набор абстрактных элементов, называемых *объектами*. Набор логических правил также ограничен и может быть описан небольшим числом стандартных операций.

Объекты языка подразделяются на 7 категорий и 14 типов.

Категории	Типы
Динамическая	Транзакт
Операционная	Блоки
Аппаратная	Устройства, памяти, ключи
Вычислительная	Переменные, арифметические, логические, функции
Статистическая	Очереди, таблицы
Запоминающие	Ячейки, матрицы ячеек
Группирующие	Списки, группы

Для облегчения процесса построения модели разработан так называемый язык блок-диаграмм, который позволяет упростить переход от алгоритма, определяющего процесс функционирования, к модели.

Основой пакета является программа, описывающая функционирование выделенного ограниченного набора объектов, и специальная диспетчеризирующая программа, которая называется симулятор и выполняет следующие функции:

1. обеспечение заданных маршрутов продвижения динамических объектов
2. планирования событий, происходящих в модели путем регистрации времени каждого события и выполнение их в нарастающей временной последовательности
3. регистрация статистической информации по функционированию модели
4. продвижение модельного времени

*Динамическими* объектами являются *транзакты* (сообщения), которые представляют собой единицы исследуемых потоков и производят ряд определенных действий, приводящихся по фиксированной траектории, представляющих собой совокупность объектов других категорий.

*Операционные – блоки*, задающие логику работы системы и определяющие пути движения транзактов между объектами аппаратной категории.

*Аппаратные* объекты – абстрактные элементы, на которых может быть декомпозировано оборудование реальной системы. Воздействуя на эти объекты, транзакты могут изменять их состояния и влиять на движение других транзактов.

*Вычислительная* категория – для описания таких ситуаций в процессе моделирования, когда связи между компонентами моделируемой системы наиболее просто и компактно выражаются в виде математических соотношений.

*Функции* – используя их, пользователь может задавать непрерывную или дискретную последовательность между аргументом функции и ее значением. Функции задаются табличным способом с помощью оператора описания функции.

*Очереди* – в любой системе движение потоков транзактов может быть задержано из-за недоступности ресурсов, например, необходимое устройство уже занято. Как правило, это многоканальные устройства. В этом случае задержанные транзакты становятся в очередь. Учет этих очередей составляет одну из основных функций интерпретатора. Пользователь может специально определить точки модели, в которых необходимо собирать статистику об очередях, т.е. установить регистраторы очереди. В этом случае интерпретатор автоматически собирает статистику об очередях – длина, среднее время нахождения в очереди, и т.д. Вся эта информация является стандартным числовым атрибутом – СЧА. И

она доступна пользователю в процессе моделирования. Интерпретатором также автоматически поддерживается дисциплина обслуживания FIFO. Пользователь может получить стандартную статистическую информацию только о таких очередях. Если же есть необходимость организовать очередь из транзактов другой дисциплины обслуживания, то для этого используются списки пользователя. Эти списки также помогают осуществить синхронизацию движения разных транзактов по модели.

Объект *таблица* – для сбора статистики о случайных величинах. Таблица состоит из частотных классов, в которые заносится число попаданий конкретной величины (СЧА). Вычисляется математическое ожидание и среднеквадратичное отклонение.

В процессе моделирования системы одни объекты взаимодействуют с другими, в результате чего происходят изменения атрибутов и преобразование их значений. Такие преобразования – *события*.

Транзакты моделируют прохождение по системе соответствующих единиц исследуемого потока. Такое движение может быть разбито на ряд элементарных событий, происходящих в определенные моменты времени.

Основной задачей имитатора является распределение этих событий, расположение их в правильной временной последовательности и выполнение соответствующих действий при наступлении каждого события. Все отрезки времени описываются целыми числами.

При составлении модели необходимо провести временное масштабирование, для всех временных параметров.

### **Особенности модели на языке GPSS.**

Каждому объекту соответствуют атрибуты, описывающие объект в данный момент времени. Значения – арифметические или логические. Большая их часть недоступна пользователю. Атрибуты которые можно адресовать – стандартные логические, СЧА.

С блоками непосредственно связано – операционные блоки, изменяющие процесс моделирования, блоки вывода и печать промежуточных результатов, команды управляющие процессом моделирования, команды редактирования. Всем блокам – порядковые номера. Транзакты представляют собой описание динамических процессов реальной системы. Они могут определять реальные физические объекты и нефизические.

### *Канальные программы*

Транзакты можно генерировать и уничтожать. Основным атрибутом транзакта является его параметры. Число которых для каждого колеблется от 0 до 1020. Ри. I – номер, х – тип. Слово, полуслово, байт, плавающая точка L. Когда два транзакта соперничают при занятии какого-либо места, первым обрабатывается тот, у которого приоритет выше. Если они одинаковые, то сначала тот, у которого время ожидания обработки больше. В одном задании может выполняться как один так и несколько прогонов модели. При этом текущем значением абсолютного времени модели будет называться суммарное время по всем реализованным прогонам – A1, а текущем значением относительного времени – C1, системное время в пределах одного прогона.

### *Классификация блоков GPSS.*

Блоки, используются для описания функций и управляют движением транзактов. У каждого блока имеется 2 стандартных числовых атрибута.  
Wn – счетчик входа в блок. Как правило номер транзакта, входящего в данный блок.  
Nn – Общий счетчик транзактов, поступивших в блок с начального момента моделирования.

1. Блоки, осуществляющие модификацию атрибутов.  
Временная задержка – ADVANCE  
генерация и уничтожение – GENERATE TERMINATE SPLIT ASSEMBLE  
синхронизация движения нескольких транзактов – MATCH GATHER  
изменение параметров транзактов – ASSIGN INDEX MARK  
изменение приоритетов - PRIORITY
2. Блоки, изменяющие последовательность передвижения транзактов (блоки передачи управления).  
TRANSFER, LOOP, TEST, GATE
3. Блоки, связанные с группирующей категорией  
JOIN, REMOVE, EXAMINE, SCAN, ALTER
4. Блоки, сохраняющие значения  
SAVEVALUE, MSAVEVALUE
5. Блоки, организующие использование объектов аппаратного устройства.  
устройство – SEIZE - RELEASE  
PREEMPT - RETURN  
FAVAIL – FUNAWAIL  
памяти – ENTER – LEAVE  
SAVAIL – SUNAVAIL  
ключи – LOGIC
6. Блоки, обеспечивающие получение статистических результатов.  
QUEUE, DEPART  
TABULATE, TABLE
7. Специальные блоки
8. Блоки для организации цепей  
LINK, UNLINK
9. Вспомогательные блоки  
LOAD, SAVE

Каждый блок определяется с помощью отдельной команды. В общем случае сначала нумерация, потом поле метки, поле операции, поле operandов, если необходимо – комментарии

#### **Моделирование. 11.11.04.**

##### **Блоки, связанные с динамической категорией.**

Самостоятельно:  
управляющая команда START со всеми ситуациями.  
блок MATRIX – описание матрицы  
Самостоятельно функцию типа

Относится к транзактам, которые в процессе моделирования создаются, размножаются, собираются и уничтожаются. Каждому транзакту соответствует набор параметров. Параметры транзактов: это свойства транзакта, определяемые пользователем. Множество параметров – набор стандартных числовых атрибутов, которые принадлежат именно

этому данному транзакту. С точки зрения программной реализации, это локальные переменные, которые доступны именно этому транзакту. В процессе перемещения транзакта по модели, его параметры могут задаваться и модифицироваться.

Особенности параметров транзактов:

P<номер транзакта>

P22

P\$<имя>

P\$Color

Номера – с помощью целых чисел или символьных имен.

При входе транзактов в модель, начальные значения всех его параметров устанавливаются в 0. Значение параметров транзакта и их изменение определяется пользователем. Значения могут быть и отрицательные числа. Транзакт может обращаться только к своим параметрам. Если необходимо получить доступ к параметрам других транзактов, то понятие ячейки или через понятие групп транзактов.

Параметры можно использовать в качестве операндов блоков и в качестве аргументов функций. Параметры позволяют организовать косвенную адресацию.

С динамической категорией связаны следующие группы блоков:

1. *группа блоков задержки транзактов по заданному времени.* Из 4 типов событий, которые могут произойти с транзактом, простейшим является задержка транзакта в течении определенного времени. Задать это можно в блоке ADVANCE A,B. Он задает среднее время выполнения операции в моделируемой системе, а также разброс времени относительно среднего. Время может задаваться любым положительным числом, в том числе и 0. Если время равно 0, то транзакт в блоке ADVANCE не задерживается и передается в следующий блок. Для задания времени пребывания его в блоке ADVANCE, в поле А указывается среднее время, а модификатор указывается в поле В. Если время задержки постоянно, то поле В может быть пустым. Иначе, если оно 0 – то поле А пустым. Модификатор может быть 2х типов:
  1. *модификатор-интервал.* используется, когда время задержки транзакта распределено равномерно в некотором заданном диапазоне.  
ADVANCE 10,5
  2. *модификатор-функция.* Он используется, если время задержки транзакта распределено более сложно. При обращении к функции определяется некоторое число – ее значение. И время задержки транзакта в блоке определяется умножением этого числа на значение среднего. Если результат – не целое число, то берется целое.  
ADVANCE 500, FN\$XPDIS
2. *Группа блоков создания и уничтожения.* GENERATE, TERMINATE, SPLIT, ASSEMBLE.  
GENERATE – создание транзакта, входящего в систему.  
GENERATE A,B,C,D,E,F,G,H,I  
А – среднее время между поступлениями отдельных транзактов, как и в поле ADVANCE, оно может быть модифицировано – быть параметром-интервалом/функцией. Может быть 0. Если при вычислении времени появления

первого транзакта оно получилось равным 0, то программа-симулятор полагает его равным 1. Среднее время принимается равным 1, если поле В пусто, а в поле А – модификатор-функции. Задаваемый модификатор-интервал не должен превосходить среднего, записанного в поле А, чтобы не получились отрицательные интервалы между появлением транзактами. Интервал между транзактами, т.е. время появления следующего транзакта, вычисляется после того, как генерируемый транзакт покидает блок. Поэтому, если после блока GENERATE стоит блок, который по какой-либо причине может задержать сгенерированный транзакт, то время генерирования следующего транзакта будет вычислено после снятия блокирующего устройства. Следовательно, средний интервал между транзактами будет *больше* чем среднее значения поля А, что приводит к ошибке. С – записывается начальная задержка. Заданное в этом поле число (без модификации) определяет интервал времени до создания данного блока первого транзакта.

D – задает число транзактов, которое должно быть сгенерировано блоком GENERATE. Если оно пусто, то он генерирует неограниченное число транзактов. E – задается приоритет

F – I: резервируют для транзакта необходимое число типов параметров. Слово, пол слова, байт, плавающая точка.

GENERATE 10,3,100,16,5,5PB,20PH,3PL,4PF

GENERATE 10,2,1000,10,4

GEN 100, FN\$EXPON

TERMINATE A – удаляет транзакты. На блок-диаграммах для обозначения окончания. Поле указывает, изменяет ли этот блок содержимое счетчика TG1 в момент поступления транзакции. Если изменяет, то на что. Каждый раз, когда транзакт входит в этот блок, то значение счетчика меняется на это число (отнимается). Во всей модели только один блок TERMINATE, то по завершению моделирования через этот блок пройдет 500. Если поле не определено, то оно считается равным 0. И транзакты, проходящие через этот блок не уменьшают содержимое счетчика. Следовательно, в модели должен быть хотя бы один такой блок, у которого поле А не меньше 1.

3. *Группа блоков изменения параметров.* Интерпретация смысла параметров транзактов – произвольна.  
ASSIGN A,B,C – основное средство для задания значений параметров транзактов. А – какой параметр поступившего транзакта должен быть изменен. Следующий непосредственно за ним символ указывает, что надо сделать, с записанным в поле В целым числом. Варианты: прибавить, вычесть, заменить текущее значение этим числом.  
Если в поле С указано какое-либо значение, оно интерпретируется как номер функции, производится определение значения функции, а результат используется для модификации целого числа, указанного в поле В. Произведение помещается в параметр, указанный в поле А.

### Организация циклов.

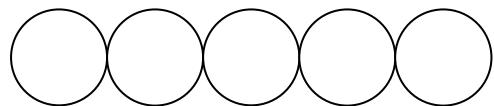
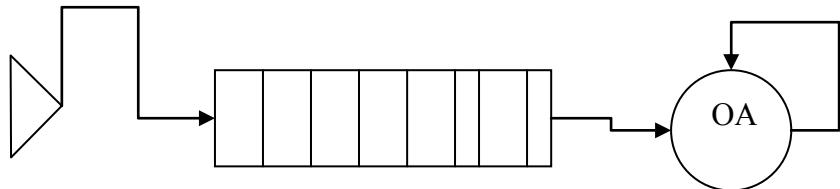
LOOP A [,B] – управляет количеством повторных прохождений транзактами определенной последовательности блоков модели.

A – параметр транзакта, используемый для организации цикла – переменная цикла. Он может быть числом, именем, стандартным числовым атрибутом.

B – метка (имя) начального блока цикла. Когда транзакт входит в блок цикла, параметр поля А уменьшается на 1, а затем его значение проверяется на равенство

0. Если оно не равно 0, то транзакт переходит в блок, указанный в операнде В.  
Иначе он проходит в следующий блок.

Задача: построить имитационную программу модели процесса прохождения 70 деталей, поступающих с интервалом 12+/-2, обрабатываемых одним рабочим по 5 последовательно идущим друг за другом операциям. 2+/-1 времена распределения. Загрузка рабочего?



```

10 GENERATE 12,2
20 ASSIGN 2,5
30 SEIZE WORKER
40 WAIT ADVANCE 2,1
50 LOOP 2,WAIT
60 RELEASE WORKER
70 TERMINATE 1
80 START 70
  
```

Лаба: на примере – отладка в GPSS.

*Группа блоков созданий копий транзактов.*

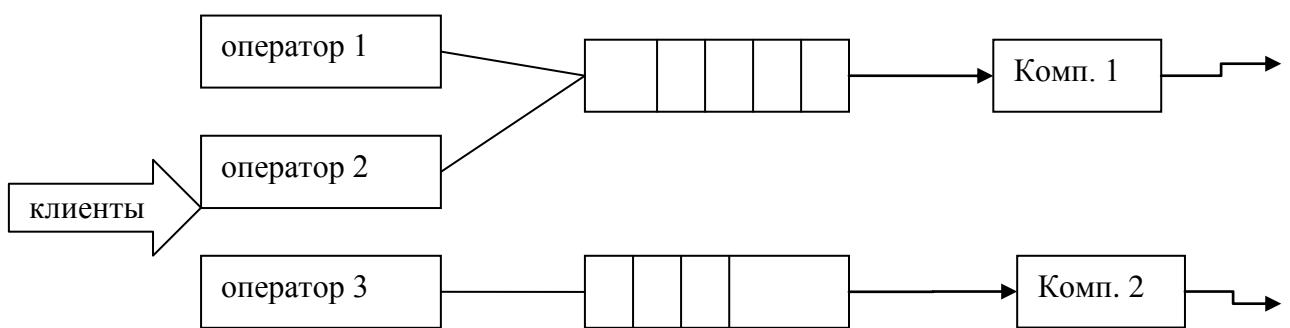
SPLIT A,B,C,D,E,F,G

Также как блок GENERATE, предназначен для создания транзактов, но в отличие от него не создает дополнительных транзактов, а лишь генерирует заданное число копий входящего в него транзактов. Получаемые копии идентичны. Число копий задается в поле А. После прохождения блока SPLIT, исходный транзакт направляется в следующий блока, а все копии пересыпаются по адресу, указанному в В. Если в поле А задано некоторое число, то  $i+1$ . Исходное сообщение и копия являются равноправными и могут проходить через любое количество блоков. Все полученные копирования транзакты, а также копии копий, принадлежат к одному ансамблю. К этому ансамблю можно применять специальные блоки, осуществляющие обработку ансамблей транзакта – MATCH, GATHER, ASSEMBLE. Получаемый ансамбль может быть пронумерован, с поле С записывается номер ансамбля, в котором и будет произведена нумерация. Если в исходном транзакте значение этого параметры было равно K, то после – K+1. Первая копия – K+2. Так как копии параметров одинаковы, необходимо использование индексов для указания типа параметров, который берется при объединении серии. Полученные копии могут иметь число и типы параметров, отличные от исходных. Эти поля можно задавать в любом порядке – в каждом поле указывается индекс параметра, для определения его типа.

Лаба:

Составить имитационную модель:

в информационный центр приходят клиенты, через интервал времени  $10+/-2$  мин. Если все 3 имеющихся оператора заняты, то клиенту отказывают в обслуживании. Операторы имеют разную производительность и могут обеспечивать обслуживание запроса пользователя за  $20+/-5$ ,  $40+/-10$ ,  $40+/-10$ . Клиенты стараются занять свободного оператора с максимальной производительностью. Полученные запросы собираются в накопитель, оттуда выбираются на компьютеры. На первый – от первого и второго. На второй – от третьего. Время обработки запроса на первом и втором компьютере =  $15/30$  минут соответственно. Промоделировать процесс обработки 300 запросов. Определить вероятность отказа.



1. режим нормального обслуживания – клиент выбирает одного из свободных операторов, отдавая предпочтение тому, у которого меньше номер
2. режим отказа в обслуживании клиента – когда все операторы заняты

Переменные уравнения имитационной модели

Эндогенные переменные.

$TR_j$  – время решения задачи на  $j$ -ом компьютере.

Экзогенные переменные

$No$  – число обслуженных клиентов.

Группа блоков синхронизации движения транзакции.

Блок ASSEMBLE A – используется для определения определенного числа транзактов, являющихся членами одного ансамбля. Транзакты, принадлежащие одному ансамблю, будут задерживаться в блоке ASSEMBLE до тех пор, пока не поступят заданное число транзактов этого ансамбля. В результате на выходе появляется один (первый) транзакт ансамбля, а остальные транзакты уничтожаются. В одном блоке могут накапливаться транзакты разных ансамблей. Транзакты одного ансамбля могут накапливаться. Если число разных ансамблей задается с косвенной адресацией, то для его установления используется параметр первого пришедшего транзакта.

Рассмотрим пример.

Tabulate – Запись времени между поступлениями в ящик.

### **Моделирование. 12.11.04.**

#### **Блоки, связанные с динамической категорией.**

Самостоятельно:

управляющая команда START со всеми ситуациями.

блок MATRIX – описание матрицы

Самостоятельно функцию типа

Относится к транзактам, которые в процессе моделирования создаются, размножаются, собираются и уничтожаются. Каждому транзакту соответствует набор параметров.

Параметры транзактов: это свойства транзакта, определяемые пользователем. Множество параметров – набор стандартных числовых атрибутов, которые принадлежат именно этому данному транзакту. С точки зрения программной реализации, это локальные переменные, которые доступны именно этому транзакту. В процессе перемещения транзакта по модели, его параметры могут задаваться и модифицироваться.

Особенности параметров транзактов:

P<номер транзакта>

P22

P\$<имя>

P\$Color

Номера – с помощью целых чисел или символьных имен.

При входе транзактов в модель, начальные значения всех его параметров устанавливаются в 0. Значение параметров транзакта и их изменение определяется пользователем. Значения могут быть и отрицательные числа. Транзакт может обращаться только к своим параметрам. Если необходимо получить доступ к параметрам других транзактов, то понятие ячейки или через понятие групп транзактов.

Параметры можно использовать в качестве операндов блоков и в качестве аргументов функций. Параметры позволяют организовать косвенную адресацию.

С динамической категорией связаны следующие группы блоков:

4. *группа блоков задержки транзактов по заданному времени.* Из 4 типов событий, которые могут произойти с транзактом, простейшим является задержка транзакта в течении определенного времени. Задать это можно в блоке ADVANCE A,B. Он задает среднее время выполнения операции в моделируемой системе, а также разброс времени относительно среднего. Время может задаваться любым положительным числом, в том числе и 0. Если время равно 0, то транзакт в блоке ADVANCE не задерживается и передается в следующий блок. Для задания времени пребывания его в блоке ADVANCE, в поле А указывается среднее время, а модификатор указывается в поле В. Если время задержки постоянно, то поле В может быть пустым. Иначе, если оно 0 – то поле А пустым. Модификатор может быть 2х типов:
  1. *модификатор-интервал.* используется, когда время задержки транзакта

распределено равномерно в некотором заданном диапазоне.

ADVANCE 10,5

2. *модификатор-функция*. Он используется, если время задержки транзакта распределено более сложно. При обращении к функции определяется некоторое число – ее значение. И время задержки транзакта в блоке определяется умножением этого числа на значение среднего. Если результат – не целое число, то берется целое.

ADVANCE 500,FN\$XPDIS

5. *Группа блоков создания и уничтожения*. GENERATE, TERMINATE, SPLIT, ASSEMBLE.

GENERATE – создание транзакта, входящего в систему.

GENERATE A,B,C,D,E,F,G,H,I

A – среднее время между поступлениями отдельных транзактов, как и в поле ADVANCE, оно может быть модифицировано – быть параметром-интервалом/функцией. Может быть 0. Если при вычислении времени появления первого транзакта оно получилось равным 0, то программа-симулятор полагает его равным 1. Среднее время принимается равным 1, если поле В пусто, а в поле А – модификатор-функции. Задаваемый модификатор-интервал не должен превосходить среднего, записанного в поле А, чтобы не получились отрицательные интервалы между появлениями транзактов. Интервал между транзактами, т.е. время появления следующего транзакта, вычисляется после того, как генерируемый транзакт покидает блок. Поэтому, если после блока GENERATE стоит блок, который по какой-либо причине может задержать сгенерированный транзакт, то время генерирования следующего транзакта будет вычислено после снятия блокирующего устройства. Следовательно, средний интервал между транзактами будет *больше* чем среднее значения поля А, что приводит к ошибке. С – записывается начальная задержка. Заданное в этом поле число (без модификации) определяет интервал времени до создания данного блока первого транзакта.

D – задает число транзактов, которое должно быть сгенерировано блоком GENERATE. Если оно пусто, то он генерирует неограниченное число транзактов.

E – задается приоритет

F – I: резервируют для транзакта необходимое число типов параметров. Слово, пол слова, байт, плавающая точка.

GENERATE 10,3,100,16,5,5PB,20PH,3PL,4PF

GENERATE 10,2,1000,10,4

GEN 100,FN\$EXPON

TERMINATE A – удаляет транзакты. На блок-диаграммах для обозначения окончания. Поле указывает, изменяет ли этот блок содержимое счетчика TG1 в момент поступления транзакции. Если изменяет, то на что. Каждый раз, когда транзакт входит в этот блок, то значение счетчика меняется на это число (отнимается). Во всей модели только один блок TERMINATE, то по завершению моделирования через этот блок пройдет 500. Если поле не определено, то оно считается равным 0. И транзакты, проходящие через этот блок не уменьшают содержимое счетчика. Следовательно, в модели должен быть хотя бы один такой блок, у которого поле А не меньше 1.

6. *Группа блоков изменения параметров*. Интерпретация смысла параметров транзактов – произвольна.

ASSIGN A,B,C – основное средство для задания значений параметров транзактов. А – какой параметр поступившего транзакта должен быть изменен. Следующий непосредственно за ним символ указывает, что надо сделать, с записанным в поле

В целым числом. Варианты: прибавить, вычесть, заменить текущее значение этим числом.

Если в поле С указано какое-либо значение, оно интерпретируется как номер функции, производится определение значения функции, а результат используется для модификации целого числа, указанного в поле В. Произведение помещается в параметр, указанный в поле А.

Организация циклов.

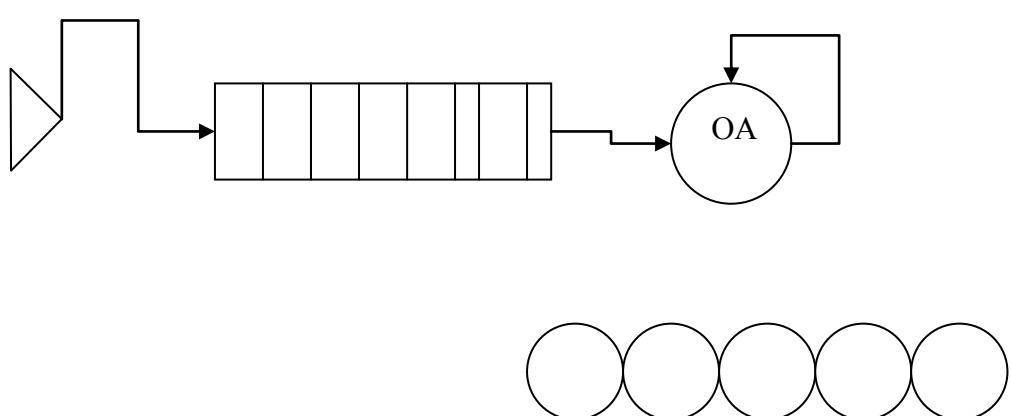
LOOP A [,B] – управляет количеством повторных прохождений транзактами определенной последовательности блоков модели.

А – параметр транзакта, используемый для организации цикла – переменная цикла. Он может быть числом, именем, стандартным числовым атрибутом.

В – метка (имя) начального блока цикла. Когда транзакт входит в блок цикла, параметр поля А уменьшается на 1, а затем его значение проверяется на равенство 0. Если оно не равно 0, то транзакт переходит в блок, указанный в операнде В.

Иначе он проходит в следующий блок.

Задача: построить имитационную программу модели процесса прохождения 70 деталей, поступающих с интервалом 12+/-2, обрабатываемых одним рабочим по 5 последовательно идущим друг за другом операциям. 2+/-1 времена распределения. Загрузка рабочего?



```
10 GENERATE 12,2  
20 ASSIGN 2,5  
30 SEIZE WORKER  
40 WAIT ADVANCE 2,1  
50 LOOP 2,WAIT  
60 RELEASE WORKER  
70 TERMINATE 1  
80 START 70
```

Лаба: на примере – отладка в GPSS.

*Группа блоков создания копий транзактов.*

SPLIT A,B,C,D,E,F,G

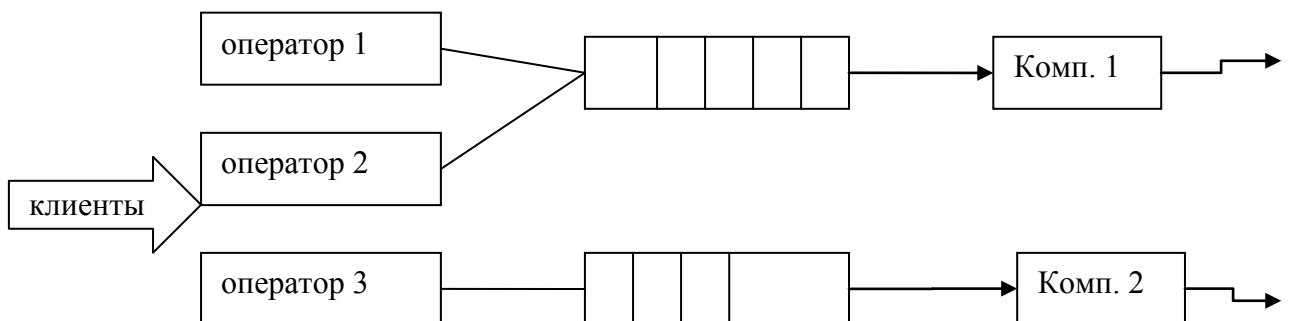
Также как блок GENERATE, предназначен для создания транзактов, но в отличие от него не создает дополнительных транзактов, а лишь генерирует заданное число копий

входящего в него транзактов. Получаемые копии идентичны. Число копий задается в поле А. После прохождения блока SPLIT, исходный транзакт направляется в следующий блока, а все копии пересыпаются по адресу, указанному в В. Если в поле А задано некоторое число, то  $i+1$ . Исходное сообщение и копия являются равноправными и могут проходить через любое количество блоков. Все полученные копии – транзакты, а также копии копий, принадлежат к одному ансамблю. К этому ансамблю можно применять специальные блоки, осуществляющие обработку ансамблей транзакта – MATCH, GATHER, ASSEMBLE. Получаемый ансамбль может быть пронумерован, с поле С записывается номер ансамбля, в котором и будет произведена нумерация. Если в исходном транзакте значение этого параметра было равно K, то после – K+1. Первая копия – K+2. Так как копии параметров одинаковы, необходимо использование индексов для указания типа параметров, который берется при объединении серии. Полученные копии могут иметь число и типы параметров, отличные от исходных. Эти поля можно задавать в любом порядке – в каждом поле указывается индекс параметра, для определения его типа.

Лаба:

Составить имитационную модель:

в информационный центр приходят клиенты, через интервал времени  $10 \pm 2$  мин. Если все 3 имеющихся оператора заняты, то клиенту отказывают в обслуживании. Операторы имеют разную производительность и могут обеспечивать обслуживание запроса пользователя за  $20 \pm 5$ ,  $40 \pm 10$ ,  $40 \pm 10$ . Клиенты стараются занять свободного оператора с максимальной производительностью. Полученные запросы собираются в накопитель, оттуда выбираются на компьютеры. На первый – от первого и второго. На второй – от третьего. Время обработки запроса на первом и втором компьютере = 15/30 минут соответственно. Промоделировать процесс обработки 300 запросов. Определить вероятность отказа.



3. режим нормального обслуживания – клиент выбирает одного из свободных операторов, отдавая предпочтение тому, у которого меньше номер
4. режим отказа в обслуживании клиента – когда все операторы заняты

Переменные уравнения имитационной модели  
Эндогенные переменные.

$TR_j$  – время решения задачи на  $j$ -ом компьютере.

Экзогенные переменные

No – число обслуженных клиентов.

Группа блоков синхронизации движения транзакции.

Блок ASSEMBLE A – используется для определения определенного числа транзактов, являющихся членами одного ансамбля. Транзакты, принадлежащие одному ансамблю, будут задерживаться в блоке ASSEMBLE до тех пор, пока не поступят заданное число транзактов этого ансамбля. В результате на выходе появляется один (первый) транзакт ансамбля, а остальные транзакты уничтожаются. В одном блоке могут накапливаться транзакты разных ансамблей. Транзакты одного ансамбля могут накапливаться. Если число разных ансамблей задается с косвенной адресацией, то для его установления используется параметр первого пришедшего транзакта.

Рассмотрим пример.

Tabulate – Запись времени между поступлениями в ящик.

#### **Моделирование. 19.11.04.**

Построить программу модели процесса прохождения 100 деталей, равномерный закон 8 +/-2. Обработка параллельно 2мя рабочими, каждый из которых выполняет свою операцию независимо со временем 5+/-3, также распределенным равномерно. Требуется определить коэффициент занятости.

```
10  GENERATE 8, 2    // приход деталей
20  SPLIT 1, THIS      // начало обработки детали
30  SEIZE 1            // первый рабочий
40  ADVANCE 5, 3
50  RELEASE 1
60  TRANSFER , THAT

70
THIS SEIZE 2          // второй рабочий
80  ADVANCE 5, 3
90  RELEASE 2
100
THAT ASSEMBLE 2        // окончание обработки
110  TERMINATE 1
      START 100
```

#### **самостоятельно – все окна графики**

Действие блока GATHER A аналогично ASSEMBLE, отличие в том, что после накопления в блоке числа транзактов указанного в поле A, они все передаются в следующий блок. Этот блок позволяет синхронизировать в режиме транзактов одного ансамбля при их движении по одному пути.

GATHER 3 – сначала все 3 придут, и потом они же последуют дальше.

задача: на производственный участок сборки подшипников поступают обоймы и шарики с интервалом времени 25 +/- 4. На контроль обоймы затрачивается 4 +/- 1. Контроль шариков производится последовательно со временем 2 +/- 1 на каждый шарик. Операция

сборки требует одновременного поступления обоймы и всех шариков и производится со временем 4 +/- 2.

```
10  GENERATE 25, 4 // поступление шариков и обойм
20  SPLIT 8, THAT // разделение обоймы и шариков
30  SEIZE 1
40  ADVANCE 4, 1
50  RELEASE 1
60  TRANSFER , FINAL
70
THAT SEIZE 2
80  ADVANCE 2, 1
90  RELEASE 2
100 GATHER 8
110 FINAL ASSEMBLE 9
120 SEIZE 3
130 ASSEMBLE 4, 2
140 RELEASE 3
150 TERMINATE 1
START 80
```

Блок MATCH предназначен для синхронизации, продвижения транзактов ансамбля, движущихся разными путями. Для синхронизации необходимо 2 блока MATCH. Они называются сопряженными. В поле А каждого блока указывается метка сопряженного ему блока. При подходе транзакта к блоку, проверяется наличие в сопряженном ему блоке транзакта из того же ансамбля. Если в обоих блоках имеются транзакты одного ансамбля, то они одновременно пропускаются через эти блоки. Если в сопряженном блоке нет ни одного транзакта данного ансамбля, то поступивший транзакт будет ожидать поступления транзакта в сопряженный ему блок. После чего оба транзакты будут пропущены в следующие за блоками MATCH.

Одна и та же пара одновременно синхронизирует любое число пар транзактов из разных ансамблей. Транзакты одного ансамбля также могут быть синхронизированы в любом числе блоков MATCH. Он может быть сопряжен сам себе. При этом его действие будет аналогично блоку ...

AAA1 MATCH BBB1

...

BBB1 MATCH AAA1

рис 1.

CCC1 MATCH CCC1

Здесь транзакт будет ждать прихода члена этого же ансамбля в этот же блок MATCH.

500 деталей, поступают 300 +/- 50. Обработка – двое рабочих, которые выполняют по 2 операции. После первой операции, выполняемой первым рабочим 70 +/- 20 и вторым 60 +/- 30 производится операция сверки, ее время выполнения - ... После сверки выполняется вторая операция первым рабочим со временем 20 +/- 10 и вторым 30 +/- 20. Затем третий рабочий производит сборку изделия из этих деталей со временем 50 +/- 20. Все процессы подчиняются равномерному распределению.

```

10      GENERATE 300, 50
20 MAN_A SEIZE 1
30      ADVANCE 70, 20
40      HERE MATCH THERE
50      ADVANCE 20, 10
60      RELEASE 1
70      TRANSFER , MAN_C
80 MAN_B SEIZE 2
90      ADVANCE 60, 30
100     THERE MATCH HERE
110     ADNVACE 30, 20
120     RELEASE 2
130 MAN_C ASSEMBLE
140     SEIZE 3
150     ADVANCE 50, 20
160     RELEASE 3
170     TERMINATE 1
          START 500

```

**найти ошибку**

***Блоки, описывающие аппаратную категорию.***

Устройство – аналог обслуживающего прибора СМО. В любой момент времени устройство может быть занято только одним транзактом. Состояние устройства меняют 6 блоков.

SEIZE  
RELEASE  
PREEMPT  
RETURN  
FAVAIL  
FUNAVAIL

В результате входа транзакта блок SEIZE, устройство, указанное в данном блоке будет занято. Оно остается занятым, пока тот же транзакт не пройдет соответствующий блок RELEASE. Если какой-либо транзакт занимает устройство, описанное в поле А блока SEIZE, то никакой другой транзакт не сможет войти в этот блок (и вообще не сможет захватить это устройство). Один транзакт может занять любое число устройств. Блок RELEASE служит для освобождения устройства, которое ранее было захвачено, проходившего через блок SEIZE транзакта. При выполнении этого блока, задержка возникнуть не может. Устройство освобождается в момент входа транзакта в блок RELEASE. Освобождение происходит только тем транзактом, которым устройство было занято. Если перед блоком SEIZE задерживаются несколько транзактом, то они обслуживаются в соответствии с режимом FIFO.

Блок PREEMPT фиксирует использование устройства на более высоком уровне, а также приостанавливает обслуживание транзакта, захватившего устройство ранее и дает возможность прерванному транзакту захватить устройство после того, как закончится обслуживание прервавшего транзакта. Если при реализации блока PREEMPT оказалось, что одно прерывание уже произошло, т.е. устройство обслуживает прерывание, то блок не

выполняется, и соответствующий транзакт задерживается до тех пор, пока не освободится устройство. Затем обслугивается новый прерывающий транзакт, а не прерванный.

Исключением из описанных правил является случай, когда этот блок работает в режиме приоритета, т.е. в поле В стоит PR. При этом действие данного блока предусмотрено случай разрешения прерывания на основании анализа результатов приоритетов. Для последующей обработки прерванного транзакта: в поле С может быть определен какой-либо блок, на который будет передан прерванный транзакт. Если прерванный транзакт находится в блоке ADVANCE, то вычисляется остаток времени до момента времени выхода транзакта. Полученное значение помещается в параметре, описанном в поле D. В этом случае прерванный транзакт пересыпается к блоку, указанному в С. Если в поле Е стоит обозначение RE, то будут производиться обычные операции, присущие данному блоку, за исключение того, что прерванный транзакт не участвует больше в конфликте из-за захвата.

#### **Моделирование. 26.11.04.**

Блок Return – сигнализирует об окончании прерывания. При входе в блок задержка возникнуть не может. закончить прерывание может только тот транзакт, который прошел предыдущий блока, относящийся именно к данному устройству. Прерывание заканчивается в момент входа в блок return. Время в течении которого транзакт находится в прерванном состоянии, не фиксируется.

Ограничение на прерывание – нельзя делать прерывание, захватившего или обрабатывающее другие транзакты более чем на 255 устройствах одновременно. Задержка прерванного транзакта начинается с момента первого прерывания, и оканчивается в моментов окончания последнего.

Блок TRANSFER – имеет 9 режимов. **3 остальных режима – самостоятельно.**

#### **Блок FUNAVAIL.**

Выполняет операции, переводящие устройство в состояние недоступности. Недоступность устройства предупреждает занятие или прерывание устройства последующими сообщениями. При этом возможно задание специальных режимов работы данного блока, обеспечивающих окончание обслуживания последнего транзакта, передачу его на обслуживание к другому блоку до обслуживания транзакта, после окончания его периода недоступности.

Номер или диапазон номеров, переводимых в состояние недоступности, записывается в поле А. Поля В-Н предназначены для задания специальных режимов.

#### **Блок FAVAIL.**

Делает доступными устройства, указанные как номер или диапазон номеров устройств в поле А. Данный блок отменяет все режимы, заданные блоком FUNAVAIL.

FUNAVAIL	1-15
ADVANCE	30
FAVAIL	1-10
ADVANCE	15
FAVAIL	11-15

Элементы системы, предназначенные для хранения транзактов, называются памятью.

Память - <имя> STORAGE <емкость>

Изменение состояния памяти производится следующими блоками:

ENTER – LEAVE  
SUNAVAIL – SAVAIL

### **Блок ENTER**

Поле А – номер памяти.

Поле В – число единиц памяти, занимаемых транзактом при входе в блок.

А при выходе транзакта, память не изменяется.

Если поле В – пустое, то число единиц памяти = 1. В этом поле может быть и 0. Если в памяти нет достаточно пустого места, то этот транзакт не может быть обслужен блоком ENTER. Если же для последующего транзакта, число единиц памяти достаточно, то он входит в память раньше первого.

### **Блок LEAVE**

Поле А – номер памяти.

Поле В – число единиц памяти, которые необходимо освободить при выходе транзакта в этот блок.

ENTER	1,1
SEIZE 2	
LEAVE	+3,1

### **Блок SUNAVAIL**

Переводит накопитель в состояние неготовности, при котором транзакты не могут войти в него. Уменьшение содержимого накопителя в этот период может происходить путем прохождения транзакта через блок LEAVE. Номер или диапазон номеров накопителей, переводимых в это состояние, записывается в поле А.

### **Блок SAVAIL**

Переводит данный накопитель из состояния недоступности в состояние доступности. Если данный накопитель уже доступен – то ничего.

### **Логические ключи**

Предназначены для описания моделируемой системы, элементы которой могут находиться только в двух состояниях. Статистика о работе ключей не собирается. Имеет только два логических атрибута, принимающих ноль – при не выполнении, и 1 – при выполнении условия:

Lr – ключ состояния  
Ls

В начале состояния ключ может быть установлен в состояние 1 с помощью команды INIT, а его изменения в процессе моделирования – LOGIC.

Для установки логических ключей, состояние которых может быть запрошено в любом месте модели. При входе в данный блок задержки не возникает. Состояние логического объекта, указанного в поле А, изменяется одним из трех способов:

1. ключ может быть установлен – S
2. сброшен – R
3. инвертирован – I

Вид изменения определяется соответствующим мнемоническим обозначением, помещаемым непосредственно за блоком.

### **Блоки, изменяющие маршруты транзактов.**

Поток транзактов обычно проходит в блоке последовательно.

#### **Блок GATE**

Используется для определения состояния объекта без изменения, без изменения их состояния. Работает в 2х режимах:

1. отказа или условного входа
2. перехода или безусловного входа.

При работе в режиме отказа, данный блок не пропускает транзакты, если соответствующий объект не находится в требуемом состоянии. Если же поставленное в блоке условие удовлетворяется, то блок разрешает вход транзакта. Если в поле В указано наименование (номер блока), то вместо отказа, блок будет пересыпать транзакты на указанный блок. Следовательно, если поле В пустое, блок работает в режиме отказа, иначе в режиме перехода. Существует 6 условий или логических атрибутов, описывающих состояние устройств, памяти, ключей, и условия синхронизации. Мнемонические обозначения проверяемого условия записываются непосредственно после блока GATE.

Поле А определяет номер объекта аппаратного категории.

Состояние устройства описывается следующими условиями.

FNU - устройство не используется, свободно

FU - устройство используется, занято

FNI - устройство работает без прерывания, свободно или обслуживает захвативший транзакт.

FI - устройство обслуживает прерывание

FNV - устройство недоступно

FV - устройство доступно

память:

SE - память пуста

SNE - память полная

SNF - память незаполнена

SNV - память недоступна

SV - память доступна

ключи:

LR - логический ключ в состоянии “выключен”

LS - логический ключ в состоянии “включен”

M - блок GATE выполняет проверку условия синхронизации в указанном месте программы

NM - блок GATE проверяет невыполнение условия синхронизации

Пример:

GATE SF 167 // Блокировать транзакт до тех пор, пока память с номером 167 не будет заполнена

GATE LS 260 // Блокировать транзакт до тех пор пока ключ 260 не установлен

GATE FU 20 // Блокировать транзакт до тех пор, пока 20 устройство не освободится

GATE FI 44, ALIR // если устройство прервано, то перейти по указанной меткой

работает в режиме отказа.

## Блок DEST

Описывает условие, которое проверяет при входе в него транзакта и определяет дальнейшее движение транзакта в зависимости от этого условия, которое записывается в виде алгебраического соотношения двух аргументов. При выполнении условия, транзакт пропускается в следующий блок. Иначе – транзакт направляется в блок, метка которого указана в поле С. Если поле С пусто, то транзакт блокируется блоков до выполнения соотношения. Проверяемое соотношение записывается за блоком операции DEST. Используются мнемонические обозначения:

L, LE, E, NE, G, GE

Соотношения между первым и вторым аргументами – поле А и В.

Аргументы должны принадлежать к стандартным числовым атрибутам. Если поле С не пусто, то транзакт всегда может войти в блок DEST, и в зависимости от соотношения аргументов, будет передан либо в следующий блок, либо в блок, указанный в поле С. Если поле С – пусто, то транзакт при невыполнении условия не сможет пройти блок DEST.

В каждый момент времени будет проверять, не изменилось ли блокирующее условие.

## Блок TRANSFER

Основное средство, позволяющее направить транзакты к любому блоку модели.

Существует 9 режимом его работы:

1. *безусловный режим выбора.* Операнд в поле А пропущен, поле В – блок, к которому должен перейти входящий в TRANSFER транзакт.
2. *статистический режим выбора.*

TRANSFER A, B, C

Если в поле А не находится зарезервированное слово, то он работает в этом режиме.

А – вероятность перехода транзакта к блоку, указанному в поле С, в долях тысячи – 0.333. (33 – 0.033)

В – блок, к которому перейдет транзакт с вероятностью 1-А. Если оно пусто, то

транзакт переходит с вероятностью 1-А к следующему блоку.

TRANSFER .33, , LBL1

3. *режим BOTH*

TRANSFER BOTH, B, C // транзакт сначала пытается перейти к блоку, указанному в поле B, если это не удается, переходит к блоку C. Если не получается перейти ни к одному из этих блоков, то он остается в блоке TRANSFER. При каждом просмотре списка текущих событий, будет повторяться попытка перехода в указанные блоки, до тех пор пока транзакт не сможет выйти из блока TRANSFER

4. *режим ALL* – транзакт пытается войти поочередно в блоки, номера которых входят в последовательность

TRANSFER ALL, B, C, D

N, N+M, N+2M, ..., K

N – номер блока, заданного в поле B,

K – номер блока, заданного в поле C

M – приращение, заданное в поле D

TRANSFER ALL, FIRST, LAST, 3

FIRST SEIZE 1

ASSIGN 12, K1, PB

TRANSFER , LAST+2

SEIZE 2

ASSIGN 12, K2, PB

TRANSFER , LAST+2

LAST SEIZE 3

Транзакт пытается войти в блок FIRST. Если он получает отказ, что определяется типом блока, то он пытается войти в блок FIRST+3. Если и в этом блоке отказ, то транзакт пытается войти в блок FIRST+6 (LAST). Если не смог войти ни в один из этих блоков, то остается в блоке TRANSFER и система переходит к обработке другого транзакта.

5. *режим PICK*

TRANSFER PICK, B, C

При работе в этом режиме, из последовательности блоков от блока указанного в поле B, до блока в поле C, случайным образом выбирается блок, к которому будет направлен транзакт. Вероятность выбора любого блока одинакова.

6. *режим функции*

TRANSFER FN, B, C

Вычисляется значение функции из поля B. К нему добавляется значение из поля C. Таким образом вычисляется номер блока, к которому должен перейти транзакт.

TRANSFER FN, 3, PH3

значение функции по номеру 3

7. *режим параметров*

TRANSFER P, B, C

Номер блока, к которому перейдет транзакт, вычисляется как сумма параметра поля B и значения поля C.

- TRANSFER P, 12, 37  
8. режим подпрограммы  
TRANSFER SBR, B, C

Транзакт будет пытаться перейти к блоку, указанному в поле В. Поле С – номер параметры транзакта, в который записывается номер данного блока TRANSFER.

- TRANSFER SBR, LBL1, 10  
9. режим SIM – не рассматривают

Задача:

на станции техобслуживания, которая состоит из бокса для ремонта, и бокса для обслуживания, каждые 25+/-10 минут поступает автомобиль. Из них 73% требуют ремонта, который продолжается 45+/-15 минут. 27% - 17+/-8 минут - обслуживание. Промоделировать процесс обслуживания.

```
10      GENERATE 25, 10
20      TRANSFER .730, , REPAIR
30      SEIZE 1
40      ADVANCE 17, 8
50      RELEASE 1

60 REPAIR  SEIZE 2
70      ADVANCE 45, 15
80      RELEASE 2
90      TERMINATE 1

START 50
```

### **Блоки, относящиеся к статистической категории.**

1. Очереди – вводятся в моделирующую программу для сбора статистической информации о процессе ожидания.  
Постановка транзакта в очередь – QUEUE  
Удаление его из очереди - DEPART
2. Таблицы – вводятся в программу для сбора статистических данных, ввод таблицы – TABLE. Регистрация статистических данных – при входе транзакта в блок TABULATE.

### **Блок QUEUE**

Аналогичен блоку ENTER, извещает программу, что данные точки программы нужно собирать статистику об очереди. Номер очереди, в которую пользователь хочет занести транзакт, задается в поле А. При записи нового транзакта в очередь, определяется длина интервала времени, в течении которого длина очереди оставалась неизменной. При входе транзакта в блок, текущая длина очереди увеличивается на число единиц, указанное в поле В. После чего программа симулятор сравнивает новую длину очереди с максимальной, достигнутой до сих пор. Кроме того, счетчик общего числа единиц, прошедших через очередь, увеличивается на тоже число единиц.

### **Блок DEPART**

Аналогичен блоку LEAVE. Поле А – номер очереди, поле В – число единиц, на которое уменьшается длина очереди. Программа симулятор вычисляет длину времени, в течении которого транзакт находился в очереди.

### Блок TABULATE

При входе транзакта в блок, он регистрируется в таблице. Поле А – номер таблицы. В каждом блоке может быть задано число единиц, добавляемых к таблице, которое при данном обращении попадает в блок – задается в поле В. Если оно пусто, то 1. Несколько специальных режимов табулирования. Они указываются в поле А описания таблицы. Знак “–“ говорит о том, что в таблицу заносится не само значение, а разность между текущим значением этой величины и последним значением, занесенным в таблицу. Этот режим называется разностным. Если в поле А стоит обозначение RT, то по приходу транзакта в блок TABULATE, который связан с таблицей, обращение к классам частот не производится. Вместо этого, число единиц, добавляется к счетчику числа входов в таблицу. Описание таблицы должно содержать в поле В временной интервал. Если он равен скажем 1000, то значение счетчиков будет заноситься в таблицу по истечению каждого 1000. После занесения, счетчик сбрасывается. Следовательно, при этом способе, определяется распределение числа заявок – поступающих за тысячу.

Если в поле А – IA, то при входе транзакта в блок, которые соответствует таблице, заполняемой таким способом, определяется время, прошедшее с момента последнего обращения к этой таблице. Полученное значение заносится в таблицу. Данный тип таблицы представляет собой распределение промежутков времени, между моментами поступления транзактов в данную точку программы.

```
QUEUE 10
SEIZE 1
DEPART 10
ADVANCE 150, 5
RELEASE 1
QTABLE 10, 0, 5, 100 // 0 – начальное значение, 5 – шаг таблицы, 100 – количество шагов
```

Задача:

Телефонная система – 2 линии связи. Звонки, которые извне, поступают каждые  $100 \pm 60$  секунд. Когда линия занята, абонент набирает номер повторно, после  $5 \pm 1$  минута. Требуется осуществить моделирование распределения времени по каждому абоненту, чтобы осуществить связь и провести разговор. Требуется определить, сколько времени понадобится для реализации 200 разговоров. Время разговора:  $3 \pm 1$  минута.

### Моделирование. 03.12.04.

200 SETS	STORAGE 2
210 TRANSIT	TABLE M1, 100, 100, 20
220	GENERATE 100, 60
230 AGAIN	GATE SNF SETS, OCCUPIED
240	ENTER SETS
250	ADVANCE 100, 10
260	LEAVE SETS
270	TABULATE TRANSIT

280	TERMINATE 1
290 OCCUPIED	ADVANCE 300, 60
300	TRANSFER , AGAIN

Память с именем SETS – емкость в 2 единицы, берется для имитирования 2х телефонных линий. Таблица 210 – TRANSIT, длительность времени, отсчитываемой с первого звонка абонента, до тех пора, пока абонент не закончит разговаривать.

220 – транзакт, который имитирует вызов с распределением 100/60.

230 – блок GATE посыпает транзакт к блоку OCCUPIED, когда все линии заняты. Это происходит, когда память SETS заполнена и абонент должен ожидать, прежде чем позвонить вторично.

240 – если память не занята, либо в ней занято одно место, то транзакт проходит через блок GATE в блок ENTER, заняв тем самым еще одно место памяти. Если всем места в памяти заняты, то он не пропускает транзакт. Каждый транзакт, приходящий в блок ENTER имитирует вызов, который был успешно осуществлен.

250 – задержка транзакта.

260 – когда транзакт входит в блок LEAVE, он освобождает одно место в памяти.

270 – блок TABULATE добавляет длительность проведенного разговора к гистограмме времени реализации разговоров таблицы TRANSIT.

280 – блок выводит транзакт из модели, после того, как разговор завершен.

290 – транзакт приходит в блок ADVANCE, когда он пытался и не сумел занять место в памяти. Это имитирует абонента, который должен подождать, прежде чем снова набрать номер.

300 – имитирует попытку повторного звонка.

Show AC1 – время окончания моделирования. Оно окажется большим, чем из стандартного отчета.

SHOW SR\$SETS – процент занятости памяти.

SHOW ST\$SETS – среднее время занятости телефонной линии, т.е. среднее время разговора.

MI 1, SR\$SETS

MI 2, ST\$SETS

Методика отладки программы.

1. краткое описание команд отладки
2. как использовать команды
3. в виде графа

## Управляющие операторы GPSS

BVARIABLE

CLEAR – очистка статистики и удаление транзакта

END – окончание работы

EQU – назначение номера ответу

FUNCTION

FVARIABLE – для плавающей точки

INITIAL – инициализация или модификация логических ключей ячеек или матриц.

MATRIX

QTABLE

RESET – обнуление накопленной статистической информации и очистка блоков

RMULT – назначение начальных значений генератором случайных чисел.

START  
STORAGE  
TABLE  
VARIABLE

Функции в GPSS.

Существует 5 типов:

1. непрерывные числовые значение (c)
2. дискретные числовые значения (d)
3. (l) – перечень числовых значений
4. (e) – дискретное значение атрибутов.
5. (m) – перечень значений атрибутов

<метка> <операция> A B  
FUNCTION

A – аргумент функции.

B – указывается тип функции и число точек для которых вычисляется функция

1 FUNCTION RN1, C4  
x1,y1/x2,y2/x3,y3/x4,y4

Все случайные числа получаются в результате расчета на основе 8ми основных чисел – исходные. Можно генерировать одни и те же последовательности – варьируется датчиками.

Задача:

Моделирование Пуассоновского распределения. Вероятность того, что ровно K требований придет за время t. Lambda – среднее время прихода. Свойства –

1. вероятность того, что поступление требования возникнет на некотором малом интервале времени, пропорционально длине интервала.
2. Придут два или более требования в течении малого интервала, вероятность мала.
3. Все интервалы независимы.

Теорией доказано, что когда интенсивность прихода распределена по закону Пуассона, соответствующие распределения имеют экспоненциальное.

Используется стандартное распределение языка GPSS: XPDIS.

XPDIS FUNCTION RN1, C24

.0, 0  
.1, .104  
.2, .222  
.3, .335  
.4, .509  
.5, .69  
.99, .97  
8.0

Необходимо решить, какое число мест на стоянке следует отвести для автомобилей, ожидающих мойка. Поток автомобилей – Пуассоновский, со значением среднего интервала = 5. Время мойки – распределено экспоненциально со значением ... Если не застают свободного места, они моют свой автомобиль в другом месте. Исследовать систему при использовании 1, 2, 3 мест на стоянке. Моделировать работу в течение 8ми часового рабочего дня.

Start – счетчик для завершения.

Моделирование.

Для получения независимых оценок коэффициентов уравнения регрессии надо так спланировать эксперимент, то есть построить такую матрицу планирования, чтобы выполнялось условие линейной независимости и ортогональности матрицы.

## Тактическое планирование

Тактическое планирование связано с вопросами эффективности и определением способа проведения испытаний, намеченных планом эксперимента. Тактическое планирование прежде всего связано с решением задач:

1. Определение начальных условий в той мере, в которой они влияют на достижение
2. Сокращение размеров выборки при уменьшении дисперсии решения.

Для получения соотношений, связывающих характеристики, описывающие функционирование q-схемы, вводят некоторые допущения относительно входных потоков, функций распределения, длительности обслуживания запросов и дисциплины обслуживания. Для таких систем в качестве типовой математической модели будем рассматривать теорию Марковских процессов.

Случайный процесс, протекающий в некоторой системе S, называется Марковским процессом, если он обладает следующими свойствами:

1. для каждого момента времени  $t_0$  вероятность любого состояния системы в будущем (при  $t > t_0$ ) зависит только от её состояния в настоящий момент ( $t = t_0$ ) и не зависит от того, когда и каким образом система пришла в это состояние.

Другими словами в Марковском случайном процессе будущее развитие зависит только от настоящего состояния и не зависит от предыстории процесса. Для Марковского процесса разработаны уравнения Колмогорова. В общем виде они представляются:

$F = (P'(t), P(t), \lambda)$ .  $\lambda$  – набор некоторых коэффициентов. В общем случае – просто вектор.

Для стационарного распределения соотношение имеет следующий вид:

$$\Phi = (P(t), \lambda) = 0$$

Отсюда мы можем вывести соотношение следующего вида:  $P = P(\lambda)$ , где  $P$  – вероятность. Можно написать выходную функцию  $Y$  в зависимости:  $Y = Y(P(\lambda))$

Данное соотношение называется базисной моделью. Для получения зависимостей нам необходимо осуществить связь между внутренними параметрами модели, ... и внутренними параметрами системы. То есть мы должны получить зависимость:

$\Lambda = \Lambda(X, V, H)$ , которая будет называться интерфейсной моделью.

Следовательно, математическая модель Q-системы строится как совокупность базисной и интерфейсной модели. Это позволяет использовать одни и те же базисные модели при решении задач проектирования, осуществляя настройку на соответствующую задачу посредством изменения интерфейсной модели. Это так называемая параметрическая настройка на конкретную задачу. Математическая модель должна обеспечить вычисление времени реакции на запрос и определить производительность.

## Методика вывода уравнений Колмогорова.

Система характеризуется четырьмя состояниями.

<Рисунок 1>

$\Lambda[i][j]$  - Плотность вероятности для множества состояний.

Найдем вероятность  $P_1(t)$ , т.е. вероятность того, что в момент времени  $t$  система будет находиться в состоянии  $S1$ . Придадим  $t$  малое приращение  $\Delta t$  и найдем вероятность того, что в момент  $t + \Delta t$  система будет находиться в состоянии  $S1$ .

Это событие может произойти двумя способами:

1. В момент  $t$  система уже находилась в состоянии  $S1$  и за время  $\Delta t$  не вышла из него.
2. В момент времени  $t$  система находилась в состоянии  $S3$  и за время  $\Delta t$  перешла в состояние  $S1$ .

Вероятность первого варианта:  $P_1(t)(1-\lambda_{12}\Delta t)$

Аналогично вероятность второго варианта:  $P_3(t) \lambda_{31}\Delta t$

Найдем вероятность того что система находится в состоянии  $S1$ . Она равна сумме:

$$P_1(t+\Delta t) = P_1(t)(1-\lambda_{12}\Delta t) + P_3(t) \lambda_{31}\Delta t$$

Раскроем скобки в правой части, перенес  $P_1$  в левую часть и разделим все на  $\Delta t$ :

$$[P_1(t+\Delta t) - P_1(t)] / \Delta t = -\lambda_{12}P_1(t) + \lambda_{31}P_3(t)$$

при  $\Delta t > 0$

$$P_1'(t) = -\lambda_{12}P_1(t) + \lambda_{31}P_3(t) \text{ – уравнение Колмогорова для } S1.$$

Рассмотрим второе состояние  $S2$  и найдем  $P_2(t + \Delta t)$ , то есть вероятность того, что система будет находиться в состоянии  $S2$ . Варианты:

1. В момент времени  $t$  система уже была в состоянии  $S2$  и за время  $\Delta t$  не перешла ни в  $S3$  ни в  $S4$ .
2. Система была в состоянии  $S1$  и за время  $\Delta t$  пришла в состояние  $S2$
3. Система была в состоянии  $S4$  и за время  $\Delta t$  пришла в состояние  $S2$

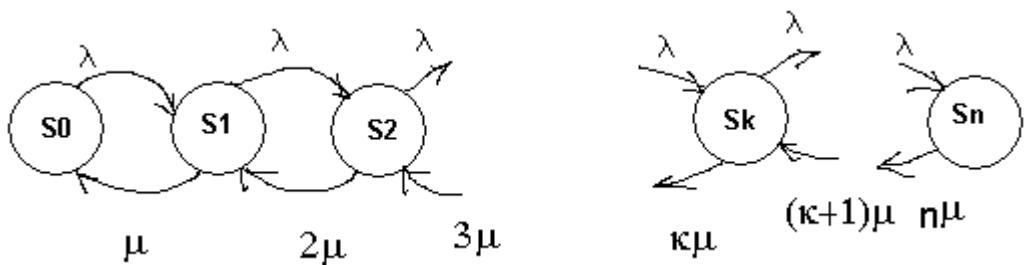
Аналогичные уравнения для всех состояний...

Интегрирование этой системы дает искомые вероятности состояний как функции времени. Начальные условия берутся в зависимости от того, каково было начальное состояние системы. Например, если в момент  $t = 0$  система находилась в состоянии  $S1$ , то начальные условия при  $t = 0$  будут:  $P_1 = 1, P_2 = P_3 = P_4 = 0$ . Отсюда следует, что при решении этой системы добавляется уравнение нормировки, то есть сумма всех вероятностей равна 1. Из структуры полученных уравнений выводим правила построения уравнений Колмогорова.

В левой части каждого уравнения стоит производная вероятности состояния, а правая часть содержит столько членов, сколько стрелок связано с данным состоянием. Если стрелка направлена из состояния, то соответствующий элемент имеет знак  $-$ , если в состояние, то  $+$ . Каждый член уравнения равен произведению плотности вероятности перехода (интенсивности), соответствующей данной стрелке, умноженной на вероятность того состояния, из которого исходит стрелка.

Рассмотрим многоканальную систему массового обслуживания с отказом. Будем нумеровать состояния системы по числу занятых каналов (или по числу заявок иными словами). Обозначим  $S0$  – все каналы свободны.  $S1$  – занят один канал, остальные свободны.  $Sk$  – занято  $k$  каналов, остальные свободны.  $Sn$  – заняты все  $n$  каналов.

Разметим граф, пропустив у стрелок интенсивности соответствующих потоков событий.



Пусть система в состоянии  $S_1$ , тогда как только закончится обслуживание заявки, занимающей канал, система перейдет в состояние  $S_0$ .

<тут система>

$\lambda\mu$

Предельные вероятности состояния  $P_0$ ,  $P_n$  характеризуют установившийся режим работы. Получим  $P_0 = 1 / [ 1 + (\lambda/\mu) / 1! + \dots + (\lambda/\mu)^n / n! ]$

$$P_k = (\lambda/\mu)^k / k! * P_0$$

$\lambda/\mu = \rho$  - среднее число заявок, приходящих в систему массового обслуживания за среднее время обслуживания одной заявки.

$$P_0 = [ 1 + P/1! + P^2/2! + \dots + P^n/n! ]^{-1}$$

$$P_k = P^k/k! * P_0$$

Зная все вероятности состояний, можно найти характеристики эффективности системы.

Вероятность отказа для данной системы. Отказ – все  $n$  каналов заняты.

$$P_{\text{отказ}} = P_n = P^n/n! * P_0$$

Относительная пропускная способность  $q$  – это вероятность того, что заявка будет принята к обслуживанию.

$$q = 1 - P_{\text{отказ}}$$

Среднее число заявок обрабатываемых в единицу времени:

$$A = \lambda * q = \lambda * (1 - P_n)$$

Полученные соотношения могут рассматриваться как базисная модель оценки производительности системы. Входящий в эту модель параметр  $\lambda$  равный 1 / (время работы) является усредненной характеристикой пользователей, а параметр  $\mu$  – функцией технических характеристик компьютера и решаемых задач. должны быть установлена с помощью интерфейсной модели. В простейшем случае, если время ввода/вывода информации мало по сравнению со временем решения этой задачи, то можно принять, что время решения  $t$  равно единице деленной на  $\mu$  ( $t = 1/\mu = n_{\text{пр}}/V_{\text{пр}}$ ). Где  $n_{\text{пр}}$  – среднее число операций, выполняемых процессором при решении одной задачи,  $V_{\text{пр}}$  – среднее быстродействие процессора (измеряется в операциях в секунду).

Лаба:

Определить среднее время нахождения системы в состояниях при установленном режиме работы. Количество состояний вводится пользователем. Клавиша Вычислить – расчет времен.

На практике далеко не все случайные процессы являются Марковскими или близкими к ним. В СМО поток заявок не всегда бывает пуассоновским. Еще реже наблюдается показательное или близкое к нему распределение времени обслуживания.

Для произвольных же потоков событий, переводящих систему из состояния в состояние, аналитические решения получены только для отдельных частных случаев. Когда построение аналитической модели по той и или иной причине трудновыполнимо, применяют метод статистических испытаний.

Идея метода Монте-Карло следующая: вместо того, чтобы описывать случайные явления с помощью аналитических зависимостей, производится «розыгрыш», моделирование случайного явления с помощью некоторой процедуры, дающей «случайный результат». Произведя такой розыгрыш большое количество раз, получаем статистический материал в множестве реализаций случайного события, который обрабатывается методами математической статистики.

1. Любым способом получаем два числа:  $x_i$  и  $y_i$ , подчиняющихся равномерному закону распределения на интервале  $[0;1]$
2. Считаем, что одно число определяет координаты точки по  $x$ , другое – по  $y$ .
3. Анализируем, принадлежит ли точка  $(x_i, y_i)$  поверхности. Если принадлежит, то увеличиваем счетчик.
4. Процедура генерации двух случайных чисел с заданным законом распределения и проверка принадлежности точки к поверхности повторяются  $n$  раз.
5. Площадь фигуры определяется как отношение количества сгенерированных точек к количеству попавших. Эта площадь и является случайной величиной.
6. Погрешность будет равна  $\sqrt{1/n}$ . Доказано фон Нейманом.

Преимущество метода статистических испытаний в его универсальности, которая обуславливает возможность практически полного статистического исследования объекта. Однако для реализации этой возможности нужны довольно полные статистические сведения о параметрах переменных.

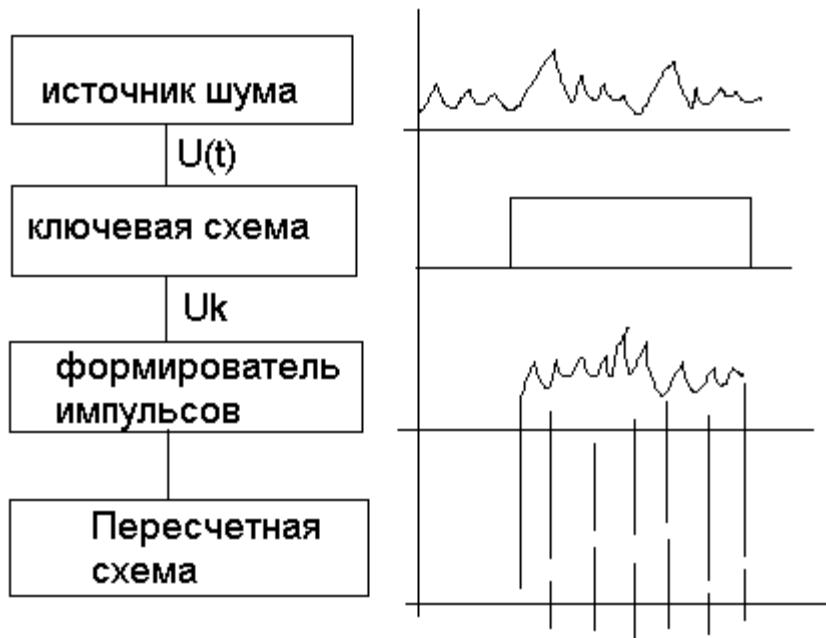
К недостаткам метода также относится большой объем требуемых вычислений, равный  $n$ . Отсюда и важность выбора величины  $n$ . Уменьшение количества испытаний повышает экономичность расчетов, но ухудшает их точность.

## **Способы получения последовательностей случайных чисел.**

На практике используются три основных способа генерации случайных чисел:

1. Аппаратный. Случайные числарабатываются специальной электронной приставкой – генератором случайных чисел – служащей, как правило, в качестве одного из внешних устройств. Реализация данного способа не требует дополнительных вычислительных операций по выработке случайных чисел, а необходима только операция обращения к внешнему устройству. В качестве физического эффекта, лежащего в основе таких генераторов, чаще всего

используют шумы в электронных



приборах.

2. Табличный способ. Случайные числа оформляются в виде таблицы и помещаются в оперативную или внешнюю память. Уже созданы хорошие таблицы.

## Лаба 2

Табличный способ и алгоритмический способ.

Найти таблицу.

Придумать критерий случайности этих чисел. По этому критерию сравнить числа одной разрядности полученные разными способами. Критерий количественный.

Хинт: посмотреть, что такое случайность.

3. Алгоритмический способ. Основан на формировании случайных чисел с помощью специальных алгоритмов.

Способ	Достоинства	Недостатки
Аппаратный	<ul style="list-style-type: none"> <li>1. Запас чисел неограничен</li> <li>2. Расходуется мало операций</li> <li>3. Не занимается место в оперативной памяти.</li> </ul>	<ul style="list-style-type: none"> <li>1. Требуется периодическая проверка на случайность</li> <li>2. Нельзя воспроизводить последовательности</li> <li>3. Используются специальные устройства. Надо стабилизировать</li> </ul>
Табличный	<ul style="list-style-type: none"> <li>1. Требуется однократная проверка</li> <li>2. Можно воспроизводить последовательности</li> </ul>	<ul style="list-style-type: none"> <li>1. Запас чисел ограничен</li> <li>2. Занимает место в оперативной памяти и требуется время на обращение к памяти</li> </ul>
Алгоритмический	<ul style="list-style-type: none"> <li>1. Однократная проверка</li> <li>2. Можно многократно воспроизводить последовательности чисел</li> </ul>	<ul style="list-style-type: none"> <li>1. Запас чисел последовательности ограничен её периодом</li> <li>2. Требуются затраты машинного времени</li> </ul>

	3. Относительно малое место в оперативной памяти 4. Не используются внешние устройства	
--	---	--

В настоящее время с помощью рекуррентных соотношений реализовано несколько алгоритмов генерации псевдослучайных чисел. Псевдослучайными они называются потому, что фактически они, даже пройдя все тесты на случайность и равномерность распределения, остаются полностью детерминированными. Это означает, что если каждый цикл работы генератора начинается с одними и теми же исходными данными (как правило, начальное значение и константы), то на выходе мы получаем одинаковые последовательности случайных чисел.

Распределение вероятности непрерывной случайной величины задается некоторой функцией  $F(x)$  равной вероятности того, что  $x$  меньше какого-то  $X$ .  $F(x) = P(x < X)$

Функция  $F(x)$  называется функцией распределения вероятностей случайной величины  $x$ . Если функция распределения вероятности обладает производной  $f(x)$ , то данная производная также определяет распределение случайной величины  $x$  и называется плотностью вероятностей величины  $x$ .

Свойства функции распределения:

- Числовые значения заключены в интервале от 0 до 1.  $0 \leq F(x) \leq 1$
- Если  $x_1 \leq x_2$ , то  $F(x_1) \leq F(x_2)$ . Т.е. функция распределения неубывающая.
- $F(x) \rightarrow 0$  при  $x \rightarrow -\infty$  и  $F(x) \rightarrow 1$  при  $x \rightarrow \infty$

Общие свойства плотности вероятности:

- Связь функции вероятности и плотности вероятности определяется формулой  $f(x) = dF(x) / dx$  или  $F(x) = \int_{-\infty}^x f(x)dx$
- Плотность вероятности неотрицательная функция
- $\int_{-\infty}^{\infty} f(x)dx = 1$

Рассмотрим некоторую случайную величину  $x$  из конечного промежутка  $[a,b]$  и  $f(x) = 1/(b-a)$ . Найти: функцию распределения, мат. ожидание и дисперсию.

### Лаба 3

Тип распределения	Характеристики распределения	Функция распределения	Плотность распределения	Мат. ожидание	Дисперсия
Равномерное на $[a,b]$					
Пуассоновское					
Экспоненциальное (показательное)					
Нормальное (Гауссово)					
k-распределение Эрланга					

- Равномерное (вывести функцию и плотность). Оцифровка осей.
- Выбирается распределение по списку в журнале

## **Немарковские случайные процессы, сводящиеся к Марковским.**

Реальные процессы очень часто обладают последействием и поэтому не являются Марковскими. Иногда при исследовании таких процессов удается воспользоваться методами, разработанными для Марковских цепей. Наиболее распространены:

1. Метод разложения случайного процесса на фазы (метод псевдосостояний).
2. Метод вложенных цепей.

### **Метод псевдосостояний:**

Сущность метода состоит в том, что состояния системы, потоки переходов из которых являются немарковскими, заменяются эквивалентной группой фиктивных состояний, потоки переходов из которых являются уже Марковскими. Условие статистической эквивалентности реального состояния и фиктивных в каждом конкретном случае подбирается по-своему. Например, очень часто используется критерий эквивалентности

$$\min \text{INT}(t_1, t_2) [\lambda_{i \text{ экв}}(\tau) - \lambda_i(\tau)] d\tau$$

где  $\lambda_{i \text{ экв}}$  – эквивалентная интенсивность перехода в  $i$ -ой группе переходов, заменяющей реальный переход, который обладает интенсивность  $\lambda_i$ . За счет расширения числа состояний системы некоторые процессы удается точно свести к Марковским. Созданная таким образом новая система. Такая система подвергается исследованию с помощью обычных приемов на базе цепей Маркова.

К числу процессов, которые введением фиктивных состояний можно точно свести к Марковским, относятся процессы, происходящие под воздействием потоков Эрланга. В случае потока Эрланга  $k$ -того порядка интервал времени между соседними событиями представляет собой  $k$  независимых случайных интервалов, распределенных по показательному закону. Поэтому сведение потока Эрланга  $k$ -того порядка к Пуассоновскому осуществляется введением  $k$  псевдосостояний. Интенсивности переходов между псевдосостояниями равны соответствующему параметру потока Эрланга. Полученный таким образом эквивалентный случайный процесс является Марковским, так как интервалы времени нахождения процесса в различных состояниях подчиняются показательному закону.

Устройство  $S$  выходит из строя с интенсивностью  $\lambda$ , причем поток отказов Пуассоновский. После отказа устройство восстанавливается. Время восстановления распределено по закону Эрланга третьего порядка. Функция плотности распределения  $f_2(t) = 0.5\mu (\mu t)^2 e^{-\mu t}$

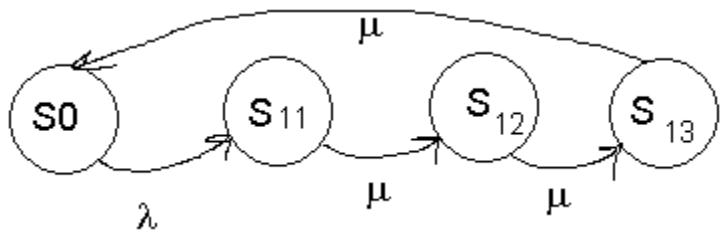
### **Найти предельные вероятности возможных состояний системы**

Пусть  $S$  может принимать два возможных состояния:

$S_0$  – устройство исправно

$S_1$  – устройство отказалось и восстанавливается

Переход  $S_0 \rightarrow S_1$  осуществляется под воздействием Пуассоновского потока, а переход  $S_1 \rightarrow S_0$  – потока Эрланга. Представим случайное время восстановления в виде суммы трех случайных времен, распределенных по показательному закону с интенсивностью  $\mu$ .



$$0 = -\lambda P_0 + \mu P_{13}$$

$$0 = -\mu P_{11} + \lambda P_0$$

$$0 = -\mu P_{12} + \mu P_{11}$$

$$0 = -\mu P_{13} + \mu P_{12}$$

$$P_0 + P_{11} + P_{12} + P_{13} = 1$$

$$P_0 = \mu / \lambda * P_{13}$$

$$P_1 =$$

## Метод вложенных цепей Маркова

В исходном случайном процессе выбираются такие случайные процессы, в которых значения процесса образуют Марковскую цепь. Моменты времени обычно являются случайными и зависят от свойств исходного процесса. Затем обычными методами теории Марковских цепей исследуются процессы в эти моменты времени. Частным случаем вложенных цепей Маркова является полумарковский случайный процесс.

Случайный процесс с конечным или счетным множеством состояний называется полумарковским, если заданы вероятности перехода системы из одного состояния в другое и распределение времени пребывания процесса в каждом состоянии, например в виде функции распределения или в виде функции плотности распределения.

## Простейшие алгоритмы генерации последовательностей псевдослучайных чисел.

Случайные числа связаны с задачами поиска каких-либо объектов и при наличии помех.

Одним из способов получения псевдослучайных чисел было выделение значения дробной части многочлена первой степени.  $y_n = t_n t(a_n + b)$ . Если  $t$  пробегает значения натурального ряда чисел, то поведение  $y_n$  выглядит весьма хаотичным. При рациональном коэффициенте  $a$  множество значений  $y_n$  конечно, а при иррациональном – бесконечно и всюду плотно в интервале от 0 до 1.

Критерий равномерности распределения любой функции от натурального ряда чисел: среднее по реализации псевдослучайных чисел равно среднему по всему множеству с вероятностью 1.

Но эти результаты далеки от практики получения последовательностей псевдослучайных чисел, потому что теоремы относятся к действительным числам  $x$  и  $y$ , которые не могут быть использованы при вычислении, потому что иррациональные действительные числа требуют для своей записи бесконечного числа знаков. Попытки замены настоящего иррационального числа его приближением для генерации псевдослучайных последовательностей опасны, так как полученные последовательности оканчиваются циклом с коротким периодом.

Способы генерации:

1. фон Нейман. Каждое последующее случайное число образуется возведением предыдущего в квадрат и отбрасыванием цифр с обоих концов.
2. Лемер.  $g_{n+1} = kg_n + c \bmod M$ . Для подбора коэффициентов  $k$ ,  $c$  и  $M$  потрачено много лет.
3. Разумнее вести вычисления в целых числах. Установлено, что при  $c = 0$  и  $M = 2^M$  наибольший период достигается при  $k$  равном  $3 + 8i$  и  $5 + 8i$  и нечетном начальном числе. Данным алгоритмом IBM создала стандартный `random`. Через пять лет некто Форсайт показал, что тройки чисел такой последовательности лежат на 15 параллельных плоскостях. От отчаяния используются два или даже три разных генератора, смешивая их значения. Если разные генераторы независимы, то сумма их последовательностей обладает дисперсией, равной сумме дисперсий. Другими словами случайность возрастает. Конгруэнтные генераторы по алгоритму .. bla-bla-bla.
4. Зейнеман. Метод целочисленной арифметики. Использовалась последовательность чисел Фибоначчи. Затем брали последнюю цифру числа.. Потому что оператор `randomize` переустанавливает не только случайное число из трех байт, а только из двух (????)

Для увеличения периода последовательности используют следующую функцию:

```
function Rand(x,y)
  x = RND(-x)
  y = RND(-y)
  if y = 0 THEN y = RND(-y)
  RAND = (x+y) mod 1
```

## **END FUNCTION**

Период такой функции  $2^{24} * (2^{41}-1)$ . Но, к сожалению, свойства у этого ряда будут такими же как у  $x$  и  $y$ .

При создании с помощью встроенного генератора случайных объектов, имеющих идентификаторов большее чем ..., его приходится использовать несколько раз, переустанавливая по заранее заданному ключу.

## **FOR I = 1 TO 5**

```
X = RND(-gamma(i))
```

## **FOR J = 0 TO 32**

```
SWAP map(j), map(32*RND)
```

## **NEXT J**

## **NEXT I**

Случайная подстановка 33 элементов массива `map`, которая может быть сделана примерно  $2^{118}$  способов и при длине периода генератора  $2^{24}$  его нужно запустить не менее 5 раз, чтобы реализовать все варианты перестановки.

Для получения значений случайной величины из последовательности случайных чисел, заданной законом распределения, обычно используют одно или несколько значений равномерно распределенных случайных чисел. Псевдослучайные равномерно распределенные случайные числа получаются в компьютере программным способом с помощью некоторого рекуррентного соотношения. Это означает, что каждое

последующее число образуется из предыдущего (или из группы предыдущих) путем реализации некоторого алгоритма, состоящего из арифметических и логических операций.

Процедура на языке Фортран, предназначенная для генерации конечной последовательности чисел равномерно распределенных на интервале от 0 до 1, с помощью мультиплекативного конгруэнтного метода для 32-ухразрядного компьютера.

## **SUBROUTINE RANDUM (IX,IY,RN)**

**IY = IX \* 1220703125**

**IF (IY) 3,4,4**

3        IY = IY + 2147483647 + 1  
4        RN = IX

**RN = RN \* 0.4656613E-9**

**IX = IY**

## **RETURN**

**END**

IX – число, которое при первом обращении должно содержать нечетное целое число, состоящее менее чем из 9 цифр.

IY – полученное случайное число, используемое при последующих обращениях к процедуре.

RN – число из интервале от 0 до 1.

```
var
  n, i : integer;
  x, r : double;
const
  M34 : double = 28395423107.0;
  M35 : double = 34359738368.0;
  M36 : double = 68719476736.0;
  M37 : double = 137438953472.0;

function Rand(n:integer):double;
  var S,W:double; i:integer;
begin
  if n = 0 then
  begin
    x := m34; Rand :=0; exit
  end;
  S:= -2.5;
  for i:=1 to 5 do
  begin
    x:= 5.0 * x;
    if x>=m37 then x:=x-m37;
    if x>=m36 then x:=x-m36;
    if x>=m35 then x:=x-m35;
    W:=X/m35;
    if n=1 then
    begin
```

```

Rand:=W; exit
end
S:=S+W;
end;
S:=S*1.54919;
Rand:=(sqr(s) - 3.0) * S + 0.01 + S
end;

begin
R:=Rand(0);
for i:=1 to 200 do
writeln(Rand(2):12:10); { 2 - Гауссово распределение, 1 -
нормальное }
readln;
end.

```

Для имитации равномерного распределения на интервале [a;b] используется обратное преобразование функции плотности:

$$(x-a)/(b-a) = R \quad X = A + (B-A)*R, R = [0;1]$$

В основе построения программы, генерирующей случайные числа с законом, отличным от равномерного, лежит метод преобразования последовательности случайных чисел с равномерным законом распределения в последовательность случайных чисел с заданным законом распределения. Метод основан на теореме, утверждающей, что некоторая случайная величина  $X$ , принимающая значения, равные корню уравнения (рис →), имеет плотность распределения  $f(x)$ , где  $R$  – случайная величина, равномерно распределенная в интервале  $[0;1]$ . Значение случайной величины, распределенной по показательному закону, может быть вычислено по данному уравнению  $1 - e^{-\lambda x} = R$ ,  $x = -1/\lambda * \ln(1-R)$

$$\int_{-\infty}^t f(x)dx = R$$

Распределение Пуассона. Относится к числу дискретных, т.е. таких, при которых переменная может принимать только целочисленное значение включая 0, с математическим ожиданием и дисперсией, равными  $\lambda$ . Используют метод точек, в основе которого лежит генерация случайной переменной  $R_i$ , равномерное распределенной в интервале  $[0;1]$ , до тех пор, пока не станет справедливым соотношение

При получении случайной величины, функция распределения которой не позволяет найти решение уравнения явным образом, можно произвести кусочно-линейную аппроксимацию и затем вычислить приближенное значение корня. Кроме того для получения случайной величины часто используют те или иные свойства распределения.

Распределение Эрланга. Определяется двумя параметрами:  $\lambda$  и  $k$ . При вычислении случайной величины по данному закону воспользуемся тем, что поток Эрланга может быть получен прореживанием потоков, распределенных по показательному закону,  $k$  раз. Поэтому достаточно получить  $k$  значений случайной величины, распределенной по показательному закону, и усреднить их. Формула 1

Нормально распределенная случайная величина может быть получена как сумма большого числа случайных величин, распределенных по одному и тому же закону и с одними и теми же параметрами.

Случайная величина  $X$ , имеющая нормальное распределение с математическим ожиданием  $M_x$  и среднеквадратичным отклонением, может быть получена по следующей формуле. Формула 2. Для сокращения вычислений на практике принимают  $n = 12$ , что дает хорошее приближение к нормальному закону.

Классификация систем массового обслуживания:

СМО классифицируются по:

1. Закону распределения входного потока заявок
2. Числу обслуживающих приборов
3. Закону распределения времени обслуживания в обслуживающих приборах
4. Числу мест в очереди
5. Дисциплине обслуживания

Для краткости записи при обозначении любой СМО принята следующая система кодирования: ABCDE

где на место буквы ставится соответствующие характеристики СМО.

- A- закон распределения интервала времени между поступлениями заявок. Наиболее часто используются следующие: экспоненциальное – m, Эрланга – e, гиперэкспоненциальное – h, гамма распределение – Г, детерминированное – D, произвольное – G.
- B- закон распределения времени обслуживания приборов СМО. Обозначения такие же как и А.
- C- число обслуживающих приборов. Для одноканальной системы – 1, для многоканальной в общем случае – a.
- D- число мест в очереди. Если число мест в очереди неограничено, то данное обозначение может опускаться. Для конечного числа мест в очереди обозначение – r или n.
- E- дисциплина обслуживания. При FIFO данное обозначение может отсутствовать

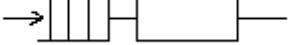
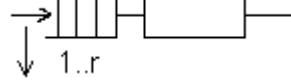
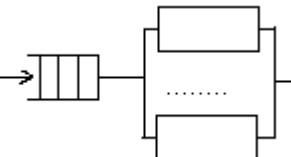
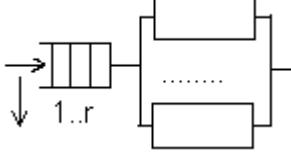
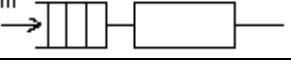
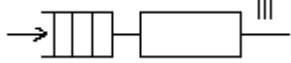
Пример: M/M/1 – СМО с одним обслуживающим прибором, бесконечной очередью, экспоненциальными законами распределения интервалов времени между поступлениями заявок и временем обслуживания. Дисциплина – FIFO

E/H/I/r/LIFO – конечная очередь, Эрланг, интервал времени между поступлениям заявок....

Для моделирования вычислительных систем и сетей наиболее часто используются СМО:

1. Одноканальные СМО с ожиданием. Представляют собой один обслуживающий прибор с бесконечной очередью. Структура является наиболее распространенной. С той или иной степенью приближения с её помощью можно моделировать практически любой узел системы или сети.
2. Одноканальные СМО с потерями. Представляет собой один обслуживающий прибор с конечной очередью. Если число заявок превышает число мест в очереди, то лишние заявки теряются. Используются при моделировании каналов передачи.
3. Многоканальные СМО с ожиданием. Представляют собой несколько параллельно работающих обслуживающих приборов с общей бесконечной очередью. Данный тип СМО часто используется при моделировании групп абонентских терминалов, работающих в диалоговом режиме.
4. Многоканальные СМО с потерями. Представляют собой несколько параллельно работающих обслуживающих приборов с общей очередью, число мест в которой ограничено. Эти СМО как и одноканальные с потерями часто используются при моделировании каналов.

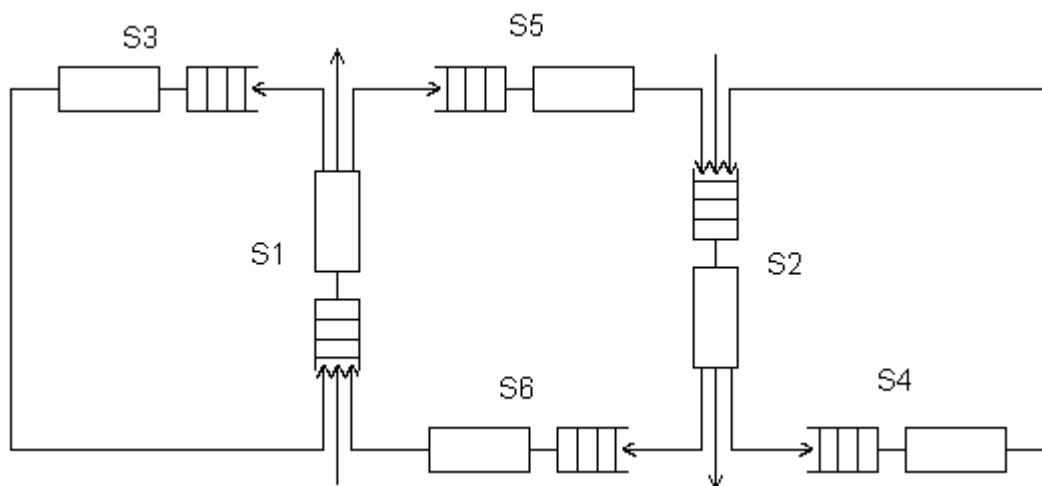
5. Одноканальные СМО с групповым поступлением заявок. Представляют собой один обслуживающий прибор с бесконечной очередью. Перед обслуживанием заявки группируются в пакеты по определенному правилу.
  6. Одноканальные СМО с групповым обслуживанием заявок, представляющие собой один прибор с бесконечной очередью. С групповым обслуживанием заявок.
- Последние два типа используются для моделирования центра коммутации.

Наименование	Обозначение	Схема
Одноканальные с ожиданием	G/G/1	
Одноканальная с потерями	G/G/1/r	
Многоканальная с ожиданием	G/G/l	
Многоканальная с потерями	G/G/l/r	
Одноканальная с групповым поступлением заявок	Gr/G/1 r	
Одноканальные с групповым обслуживанием заявок	G/Gr/1 r	

Вычислительные сети могут быть представлены в виде сети массового обслуживания. Различают: открытые, замкнутые, смешанные.

Открытой называется сеть массового обслуживания, состоящая из  $m$  узлов, причем хотя бы в один из узлов поступает извне входящий поток заявок.

Для открытых сетей характерно то, что интенсивность поступления заявок не зависит от состояния сети, т.е. от числа заявок уже поступивших в сеть. Такие сети как правило используются для моделирования вычислительных сетей, работающих в неоперативном режиме.



S1, S2 моделируют работу узлов коммутации,

S3,S4 - работу сервера

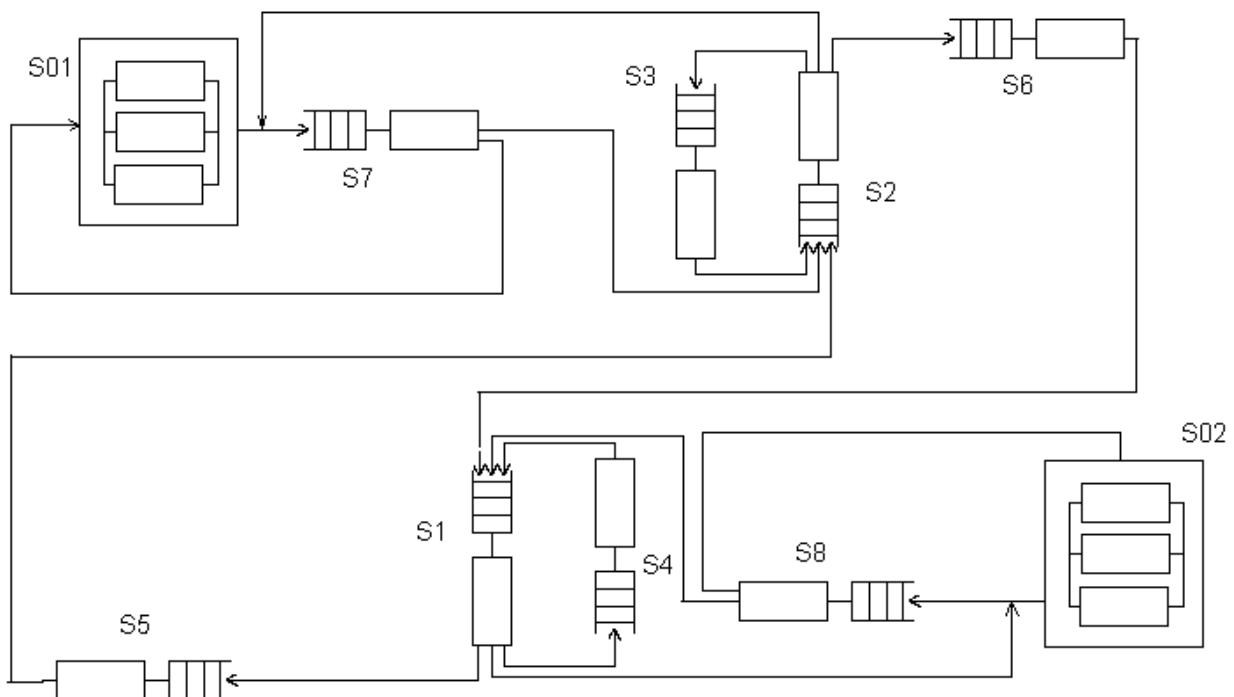
S5,S6 – работу межузловых каналов.

В сети циркулируют два потока заявок. Каждая заявка поступает на вход соответствующего узла коммутации, где определяется место её работы. Затем заявка передается на сервер или по каналу связи - на соседний, где обрабатывается, а после чего возвращается к источнику и покидает сеть.

Замкнутой называется сеть массового обслуживания с множеством узлов без источника и стока, в которой циркулирует постоянное число заявок. Замкнутые сети массового обслуживания используются для моделирования таких вычислительных сетей, источниками информации для которых служат абонентские терминалы, работающие в диалоговом режиме. В этом случае каждая группа абонентских терминалов представляется в виде многоканальной системы массового обслуживания с ожиданием и включается в состав устройств сети.

Простой режим работы диалоговых абонентов:

Абоненты не производят никаких действий, кроме посылки заданий в вычислительную систему и обслуживание полученного ответа.



S01,S02 – группы абонентских терминалов (две)

S7,S8 – каналы связи с абонентами

S1,S2 – узлы коммутации

S3,S4 – серверы

S5,S6 – каналы межузловой связи

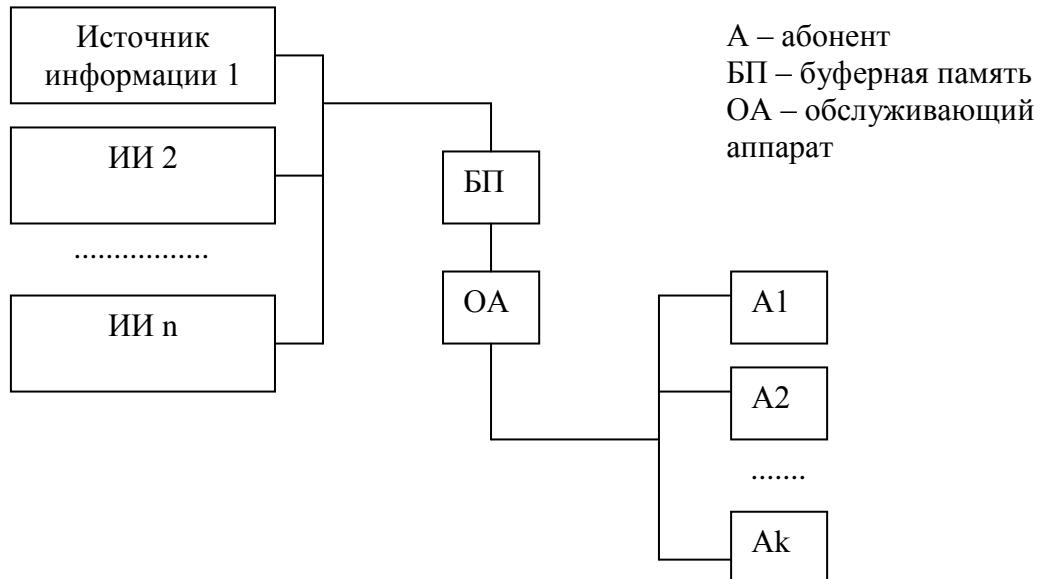
Абоненты с терминалов посыпают запросы, которые по каналам связи поступают на узлы коммутации, а оттуда на обработку в свой или соседний сервер.

При сложном режиме диалога работа абонентов представляется в виде совокупности операций некоего процесса, называемого технологическим. Каждая операция тех. процесса моделируется соответствующей СМО. Часть операций предусматривает обращение к вычислительной системе, а часть может и нет.

**Смешанной** называется сеть массового обслуживания, в которой циркулируют несколько различных типов заявок (трафик). Причем относительно одних типов заявок сеть замкнута, а относительно других – открыта. С помощью смешанных сетей массового обслуживания моделируются такие типы вычислительных сетей, часть абонентов которых работает в диалоговом, а часть – в неоперативном режиме. Для диалоговых абонентов также различают простой и сложный режим работы. Часто смешанные сети массового обслуживания моделируют вычислительные сети, в которых сервер дополнительно загружается задачами, решаемыми на фоне работы самой сети.

### Методика построения программной модели.

Для разработки программной модели исходная система должна быть представлена как стохастическая СМО. Это объясняется следующим фактором. Информация от внешней среды поступает в случайные моменты времени. Длительность обработки различных видов информации может быть в общем случае различной. Воздействия внешней среды могут отображаться каким-нибудь генератором сообщений, а весь комплекс вычислительных средств – в виде обслуживающего устройства.

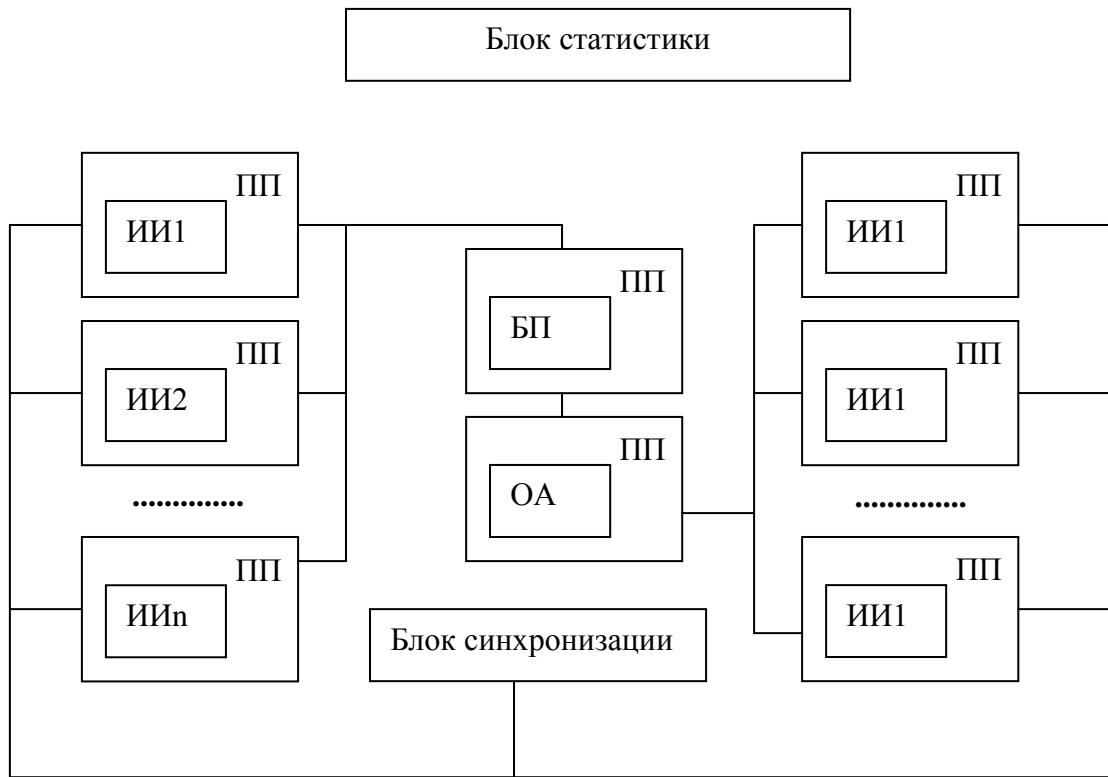


Источники информации подают на вход буферной памяти независимо друг от друга поток сообщений. Закон появления этих сообщений произволен, но обязательно задан. В буферной памяти сообщения как правило записываются навалом (как пришли) и выбираются по одному в обслуживающий аппарат (как правило по принципу FIFO). Длительность обработки одного сообщения в обслуживающем аппарате в общем случае также может быть случайной, но закон обработки должен быть задан.

Быстродействие обслуживающего аппарата ограничено, поэтому на входе как в обслуживающий аппарат так и в буферную память (если она конечная) в общем случае возможно сложение данных.

Для перехода к программной реализации необходимо каждый блок выделить в отдельную программу-модуль. Для запуска модели необходимо добавить блок синхронизации и блок статистики. Блок синхронизации осуществляет протяжку модельного времени, определяет какой блок должен быть активным в определенный момент времени. Блок статистики позволяет исследовать модель и обнаруживать её слабые места и пр. Фактически это

автоматизация планирования эксперимента на узком уровне. Таким образом резко возрастает роль человека.



### Моделирование потока сообщений

Поток сообщений обычно имитируется моментами появления очередного сообщения в

потоке. Текущий момент  $T_i = \sum_{k=1}^{i-1} T_k + t_i$  (сумма плюс интервал времени между появлением  $i$ -ого и  $(i-1)$ -ого сообщения.  $T = T + T_i$ .

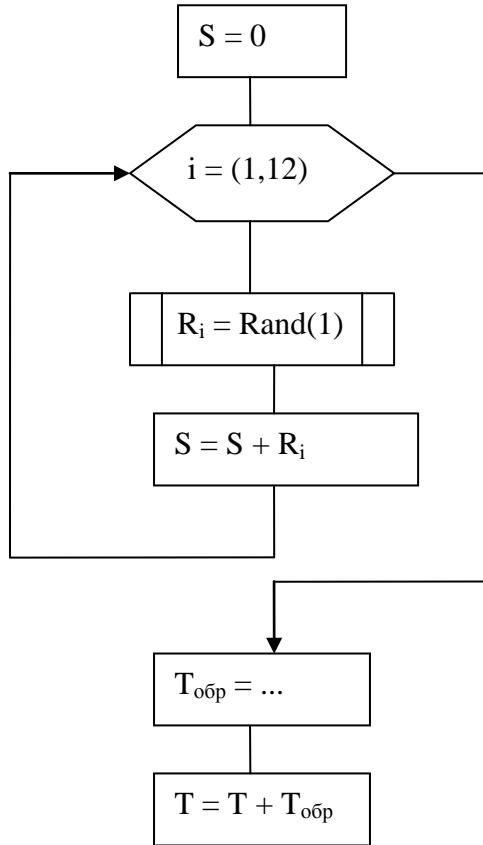
Типы распределения	Выражение для расчета времени
Равномерное на отрезке [A;B]	$T_i = A + (B - A) * R_i$
Экспоненциальное	$T_i = 1/\lambda \ln(1 - R_i)$
Нормальное (Гауссово)	$T_i = \sigma_x \sqrt{\frac{12}{n}} \left( \sum_{i=1}^n R_i - \frac{n}{2} \right) + m_x$
Распределение Эрланга	$T_i = 1/(k\lambda) \ln(1 - R_i)$

### Моделирование работы обслуживающего аппарата.

Программа, имитирующая работу обслуживающего аппарата – это набор процедур, вырабатывающих случайные отрезки времени, соответствующие длительностям обслуживания требования. Например,

$M_x, \sigma_x$ , нормальное

$$T_{обр} = M_x + \left( \sum_{i=1}^{12} R_i - 6 \right) \cdot \sigma_x$$



### **Моделирование работы абонентов**

Абонент может рассматриваться как обслуживающий аппарат, поток информации от которого передается в систему в случае наличия обратной связи или поток информации, на который поступает от процесса. Для моделирования работы абонента необходимо составить программу выработки длительности обслуживания требования. Кроме того, абонент сам может быть источником заявок на те или иные ресурсы вычислительной системы. Эти заявки имитируются с помощью генератора сообщений, которые распределены по заранее заданному закону распределения. Таким образом абоненты могут выступать либо как обслуживающий аппарат либо как генераторы.

### **Моделирование работы буферной памяти.**

Блок буферной памяти должен производить запись и считывание информации, выдавать сигналы переполнения и отсутствия данных, в любой момент располагать сведениями о количестве находящихся в нем требований. В простейшем случае сама запоминающая среда имитируется одномерным массивом, размер которого определяет объем буферной памяти. Каждый элемент этого массива может быть либо свободен, либо занят. В качестве эквивалента требования присваивается значение времени появления этого требования.

Анализ признака режима (запись или чтение)

запись - Анализ на переполнение

есть - Блок статистики

нет – запись информации о текущему адресу, изменение текущего адреса на 1

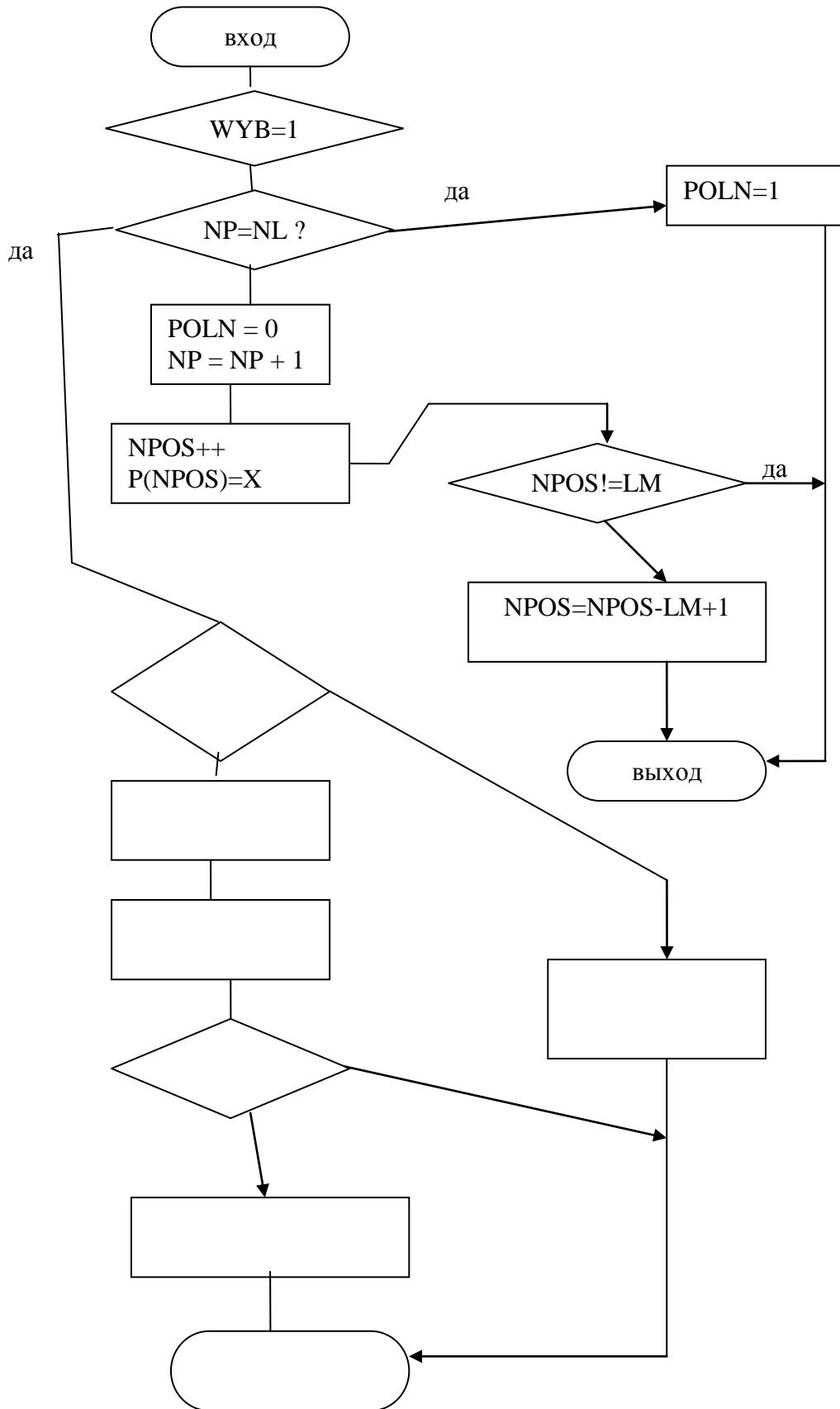
выход

чтение – Анализ наличия сообщений в буферной памяти

нет – Блок статистики

есть – чтение информации по текущему адресу, изменение текущего адреса на 1

ВЫХОД



## Лаба 4

Есть генератор, очередь, ОА, нет обратной связи. Определяем оптимальную длину очереди. Оптимум – минимальная длина очереди, при которой не теряются сообщения. Генератор работает по закону из варианта. ОА работает по равномерному закону. Параметры должны вводиться/выводиться в отдельном окне.

### Программа сбора статистики.

Задача блока статистики заключается в накоплении численных значений, необходимых для вычисления статистических оценок, заданных параметров моделируемой системы. При исследовании простейшей СМО интерес представляет среднее время ожидания очереди. Для каждого сообщения время ожидания в очереди равно разности между моментом времени, когда оно было выбрано на обработку в обслуживающий аппарат, и моментом времени, когда это сообщение пришло в систему от источника информации. Суммируя значение количества сообщений в буферной памяти через небольшие промежутки времени и разделив полученную сумму на число суммирований, получим среднее значение длины очереди. Коэффициент загрузки обслуживающего аппарата определяется как отношение времени непосредственно работы обслуживающего аппарата к общему времени моделирования. Вероятность потери сообщения определяется как количество потерянных сообщений к общему количеству сообщений (количество потерянных плюс количество обработанных).

### Разработка управляющей программы.

При создании любой модели это самый важный вопрос.

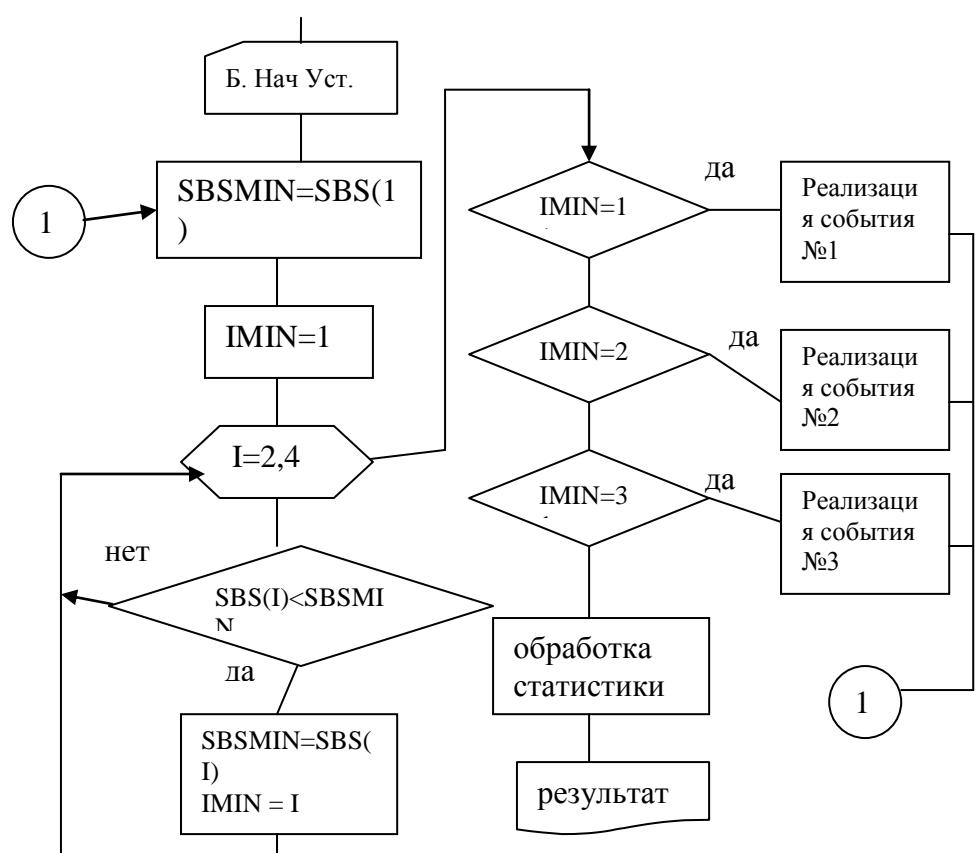
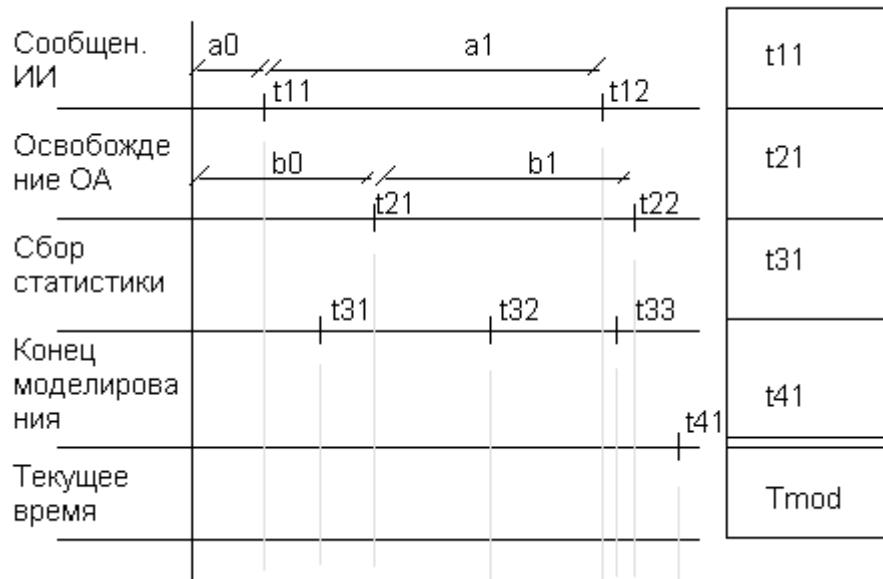
Принцип  $\Delta t$ : вся временная ось разбивается на равные отрезки, кратные  $\Delta t$ . Далее просматривается какие системные события происходят в эти промежутки. Недостаток данного алгоритма: если  $\Delta t$  выбрано слишком большим, то большая вероятность, что какие-то события будут пропущены, и этого даже не будет заметно. В случае малого  $\Delta t$  результатов придется ждать долго.

Поэтому стали привязывать время однозначно к событию. Такой принцип лишен недостатков «принципа  $\Delta t$ ». Недостаток: сложность реализации. Список событий в реальных условиях получается очень большим и поиск по нему медленный.

Если программы, отображающие работу источника информации, обслуживающего аппарата, буферной памяти, имитируют работу отдельных устройств, то управляющая программа имитирует программу взаимодействия этих отдельных устройств системы. Управляющая программа реализуется в основном по двум принципам:

1. **принцип  $\Delta t$ .** Заключается в последовательном анализе состояний всех блоков в момент  $t+\Delta t$  по заданному состоянию блоков в момент  $t$ . При этом новое состояние блоков определяется в соответствии с их алгоритмическим описанием с учетом действующих случайных факторов, задаваемых распределениями вероятности. В результате такого анализа принимается решение о том, какие общесистемные события должны имитироваться программной моделью на данный момент времени. Основной недостаток: значительные затраты машинного времени на реализацию моделирования, а при недостаточно малом  $\Delta t$  появляется опасность пропуска отдельных событий в системе, что может привести к неправильным результатам моделирования.
2. **событийный принцип.** Характерное свойство систем обработки информации заключается в том, что состояние отдельных устройств изменяется в дискретные моменты времени, совпадающие с моментами поступления сообщений в систему, окончанием решения задач или возникновением аварийных ситуаций. Поэтому

моделирование и продвижение текущего времени производят, используя событийный принцип. При использовании данного принципа состояние всех блоков анализируется лишь в момент появления какого-либо события. Момент наступления следующего события определяется минимальным значением из списка будущих событий, представляющего собой совокупность моментов ближайшего изменения состояний каждого из блоков системы.



t11, t12 – время появления событий....  
t21,t22 - Время обслуживания первого сообщения  
t31, t32, t33 - Моменты сбора статистики:  
t41 – момент окончания моделирования

SBS – список будущих событий.

Методика реализации событийного принципа:

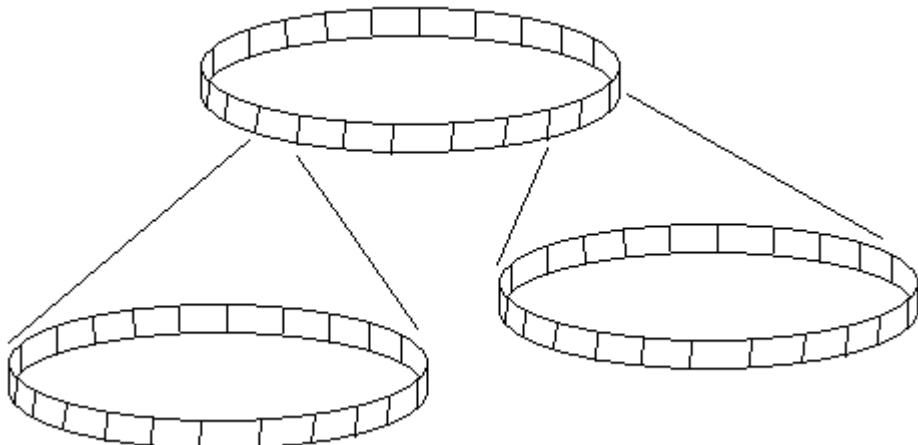
1. Для всех активных блоков (блоков, порождающих события) заводится свой элемент в одномерном массиве – списке будущих событий
2. В качестве подготовительной операции в список будущих событий заносится время ближайшего события от любого активного блока. Активизируя программный имитатор источника информации вырабатываем псевдослучайную величину  $a_0$ , определяющую момент появления первого сообщения. Эту величину заносят в список будущих событий. Активизируя программный имитатор обслуживающего аппарата вырабатывают псевдослучайную величину  $b_0$ , определяющую момент времени  $t_{21}$ , и заносят её в список будущих событий. Момент первого сбора статистики определяют равным стандартному шагу сбора статистики и его также заносят в список будущих событий как и время окончания моделирования. На этом подготовительная часть заканчивается и далее протяжка модельного времени реализуется самой программой по соответствующему алгоритму. В списке будущих событий определяется минимальное значение и его порядковый индекс, и реализуется событие, порождаемое блоком соответствующего номера. Реализация события (от источника информации) заключается в том, что само сообщение записывается в буферную память и с помощью имитатора источника информации вырабатывается момент появления следующего сообщения и это время помещается в соответствующую ячейку списка будущих событий вместо момента времени  $t_{11}$ . Затем вновь организуется поиск минимума и его номера. Реализуется событие номер 3, после чего вырабатывается момент времени  $t_{32}$  – новое время сбора статистики. Оно записывается на место времени  $t_{31}$  в список будущих событий и т.д. до тех пор, пока минимальным временем не окажется время окончания моделирования.

Два описанных выше принципа являются универсальными алгоритмами протяжки модельного времени. Для некоторых предметных областей один принцип может работать быстро и без потерь событий, а другой при этом может работать очень медленно. Выбор метода необходимо производить исходя из распределения событий во времени. В реальных системах распределение событий как правило неоднородно, события группируются по времени. Образование групп связано с наступлением какого-то «значимого» события, которое инициирует определенную последовательность действий с соответствующими событиями, имеющими высокую плотность на определенном интервале времени. Такой интервал называется пиковым, а распределение событий – квазисинхронным. Примером такой сложной системы может быть цифровая сеть, в которой синхронизирующие сигналы переключают большое количество триггеров.

### **Алгоритм Дельфта.**

Данный алгоритм был специально разработан для сложных дискретных систем, в которых присутствует квазисинхронное распределение событий. Особенностью данного метода является автоматическая адаптация к распределению событий. Метод реализуется таким образом, что на пиковых интервалах он приближается к методу  $\Delta t$ , а вне пиковых

интервалов – к событийному методу с большим шагом по времени. Алгоритм основан на использовании иерархической структуры циркулярных списков.



Список уровня 1 содержит  $n_1$  элементов и описывает планируемые события в пиковых интервалах. Число  $n_1$  представляет собой разбиение пикового интервала на более мелкие участки, с каждым из которых связан список событий, произошедших за этот интервал времени. Списки второго уровня и выше являются масштабирующими списками, количество элементов которых равно константному значению  $n_2$ , которое характеризует коэффициент масштабирования временных интервалов. Собственно алгоритм протяжки модельного времени заключается в последовательном поиске непустых элементов в самом верхнем циркулярном списке с большим шагом и дальнейшем спуске на более нижние уровни.

**Самостоятельная проработка** (Спасибо, Вано): Методы построения, основные блоки языка РДО.

**Самостоятельная проработка** (Спасибо, Ромео и дядя Дима ;): Симпас.

### **Языки имитационного моделирования.**

Для программирования модели могут использоваться следующие языки:

1. Универсальные алгоритмические языки высокого уровня.
2. Специализированные языки моделирования: языки, реализующие событийный подход, подход сканирования активностей, языки, реализующие процессно-ориентированный подход.
3. Проблемно-ориентированные языки и системы моделирования.

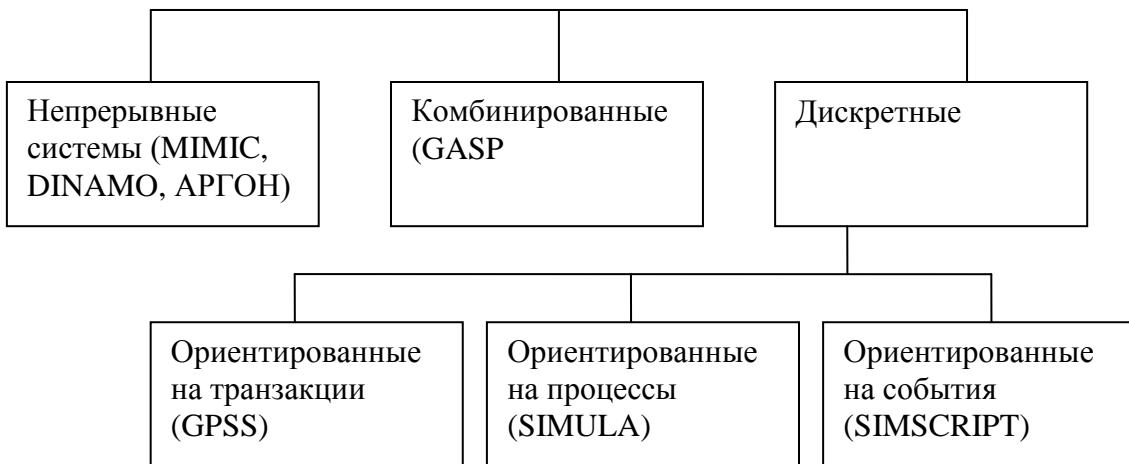
Качество языков моделирования характеризуется:

1. Удобством описания процесса функционирования.
2. Удобством ввода исходных данных. Варьированием структуры, алгоритмов работы и параметров модели.
3. Эффективностью анализа и вывода результатов моделирования.
4. Простотой отладки и работы
5. Доступностью восприятия и использования языка.

Все современные языки моделирования определяют поведение системы во времени с помощью событийного алгоритма или его модификации.

Классификация языков моделирования по принципу формирования системного времени:

## ЯИМ (Языки имитационных моделей)



Непрерывное представление систем сводится к составлению систем дифференциальных уравнений, с помощью которых устанавливают связь между входной и выходной функциям. Если переменные дискретные, то такой подход – разностный.

GASP – комбинированный, в основе лежит язык Fortran. Предполагается, что в системе могут наступать события двух типов:

- события, зависящие от состояния
- события, зависящие от времени.

### Формальное описание динамики моделируемого объекта.

Будем считать, что любая работа в системе совершается путем выполнения активности. Активность является наименьшей единицей работы. Её рассматривают как единый дискретный шаг, она имеет свое время выполнения. Таким образом, активность является единственным динамическим объектом, указывающим на совершение единицы работы. Процесс – это логически связанный набор активностей.

Активности появляются в результате свершения событий. Событие – это мгновенное изменение состояния некоторого объекта системы. Рассмотренные объекты, активности, процессы и события являются конструктивными элементами для динамического описания поведения системы. На их основе строятся языки моделирования. В то время когда динамическое поведение системы формируется в результате выполнения большого числа взаимодействующих процессов, сами эти процессы образуют относительно небольшое число классов. Чтобы описать поведение системы, достаточно описать поведение каждого класса процессов и задать значения атрибутов для конкретных классов.

Две задачи построения модели:

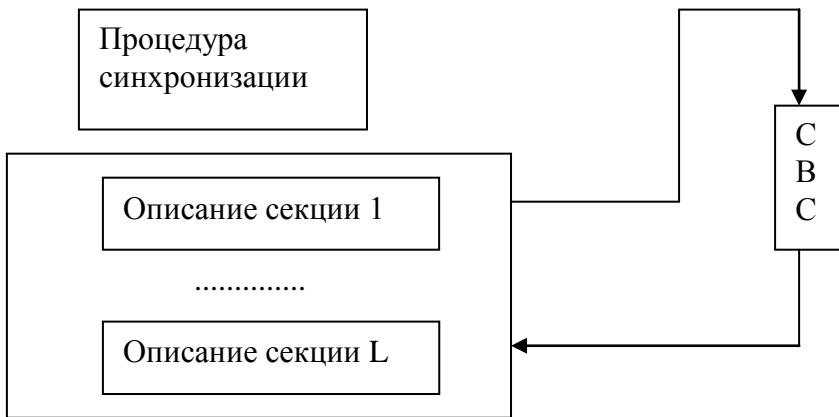
1. Необходимо задать правила, определяющие виды процессов, происходящих в системе.
2. Указать значения атрибутов или задать правила генерации этих атрибутов. При этом как правило системы описываются на определенном уровне детализации в терминах множества описаний процессов, каждый из которых включает множество правил и условий возбуждения активности.

Такое описание системы может быть детализировано на более низкий уровень представления с помощью декомпозиции процесса в активности. Так как модель служит для отображения временного поведения системы, то язык моделирования дискретных систем должен обладать средствами отображения времени. В реальной системе совместно выполняется несколько активностей, принадлежащих как связанным, так и не связанным

процессам. Имитация их действий должна быть строго последовательной. Таким образом, модель системы можно рассматривать как модель описаний, активностей, событий или процессов. Отсюда и деление языков моделирования.

#### Языки, ориентированные на события.

Моделирующая программа организована в виде совокупности секций. Секции включают в себя события (процедуры). Процедура событий состоит из набора операций, которые в общем случае выполняются после завершения какой-либо активности. Выполнение процедуры синхронизируется во времени списком будущих событий.

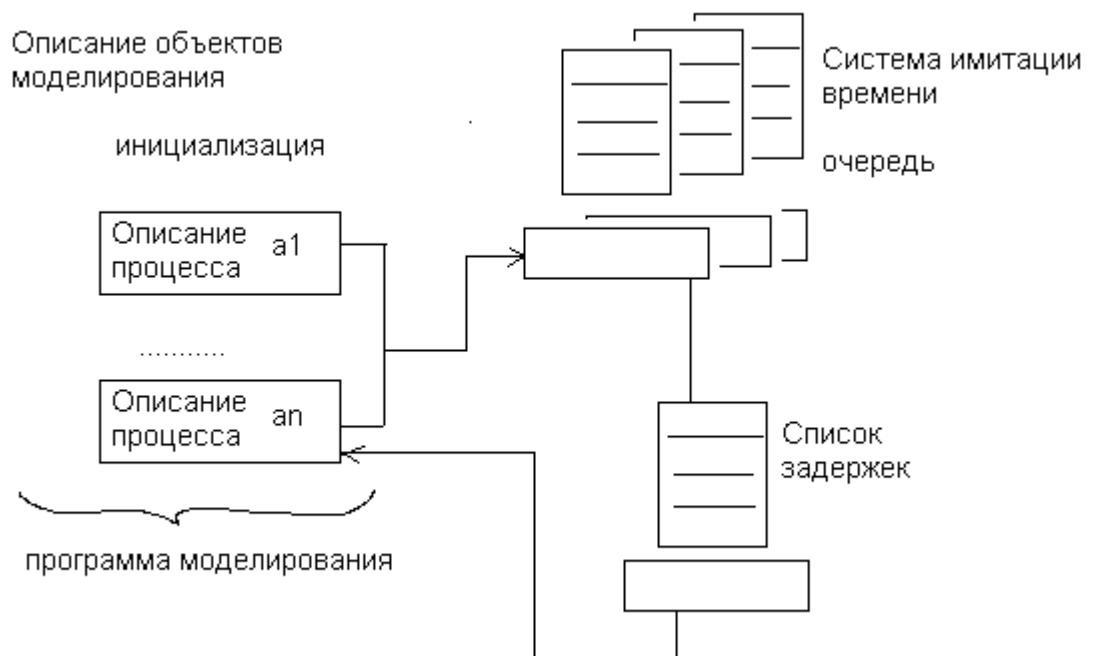



---

#### Языки, ориентированные на процессы

Моделирующая программа организуется в виде набора описаний процесса, каждый из которых описывает один класс. Описание процесса функционирования устанавливает атрибуты и активности всех процессов. Синхронизация описаний операций реализуется так же с помощью списка будущих событий, который содержит точку возобновления конкретного процесса или точку прерывания.

Структура программы на языке SIMULA:



Результаты экспертных оценок сравнения различных языков при моделировании широкого класса систем:

1. Возможности языка

SIMULA  
SIMSCRIPT  
GPSS  
C  
PASCAL

2. Простота применения:  
GPSS  
SIMSCRIPT  
SIMULA  
C  
PASCAL
3. Предпочтение пользователей:  
SIMSCRIPT  
GPSS  
SIMULA  
PASCAL  
C

Сравнение универсальных и специализированных языков программирования при моделировании:

Преимущества	Недостатки
Универсальные	
Минимум ограничений на выходной формат	Значительное время, затрачиваемое на программирование
Широкое распространение	Значительное время, затрачиваемое на отладку
Специализированные	
Меньше затраты времени на программирование	Необходимость точно придерживаться ограничений на форматы данных
Более эффективные методы выявления ошибок	Меньшая гибкость модели
Краткость, точность понятий, характеризующих имитируемые конструкции	
Возможность заранее строить стандартные блоки, которые могут использоваться в любой имитационной модели	
Автоматическое формирование определенных типов данных, необходимых именно в процессе имитационного моделирования	
Удобство накопления и представления выходной информации	
Эффективное использование ресурсов	

#### **РДО – Ресурсы, Действия, Операции**

Причинами создания данного языка были требования универсальности имитационного моделирования относительно классов моделируемых систем и процессов, легкости

модификации модели, моделирования сложных систем управления совместно с управляемым объектом.

Язык РДО является реализацией так называемого интеллектуального подхода к имитационному моделированию. Это сравнительно новый подход позволяет отойти от жесткого алгоритмического подхода в процессе принятия решений, и сделать процесс моделирования максимально гибким по способам представления информации о сложной дискретной системе. В основе языка лежит продукционная система, которая состоит из трех элементов: классов и отношений, правил, управляющей структуры. Классы и отношения трактуются как база данных, содержащая декларативные знания. Процедура представляет собой набор модифицированных продукционных правил типа Если – То. <...> Условие – проверка базы данных, а действие – это изменение базы (??). Достоинство системы, основанной на продукции, это простота создания и пополнения. Недостатки: неясность взаимных отношений, сложность оценки целостного образа знаний. Для имитационного моделирования основным недостатком систем продукции является отсутствие времени. Такие продукционные правила применимы только при моделировании статических объектов. Чтобы перейти к динамике в РДО используют модифицированное продукционное правило:

*Если Условие1 Ждать(Временной интервал) То Событие2*

В РДО сложная дискретная система представляется в виде множества взаимодействующих между собой ресурсов. Ресурс – это элемент сложной системы, внутренней структурой которого можно пренебречь, в том время как наличие и свойства его, как целого, важны и существенны для описания. Каждый ресурс в модели должен быть описан множеством параметров, которые могут быть следующих типов:

1. Описательные, представляющие факты внутренне присущие каждому ресурсу
2. Указывающие, используемые для дачи имени.
3. Вспомогательные, используемые для связи различных ресурсов, накопления статистики, графического вывода и имитации.

Требования:

Концептуальная модель – графическая схема в терминах СМО.

Должна быть выделена: входная информация, ограничения на параметры этого потока (входного), внутренняя структура, выходная информация.

Формализация процесса в виде математической модели.

Результаты отображаются в графическом виде.

Текстовый отчет должен содержать информацию о рекомендациях по повышению производительности.

### **General Purpose System Simulation (GPSS)**

GPSS – общепрограммная система моделирования. Как и любой язык имеет словарь и грамматику, с помощью которых легко могут быть разработаны точные модели систем. Существуют версии I, II, V.

Любой пакет GPSS посторон ...

Моделью сложной дискретной системы является описание её элементов и логических правил их взаимодействия в процессе функционирования. Для определения класса моделируемой системы можно выделить конечный набор абстрактных элементов, называемых объектами. Набор логических правил также ограничен и может быть описан небольшим числом стандартных операций.

Объекты языка подразделяются на семь категорий и 14 типов.

Категория	Типы
-----------	------

Динамическая	Транзакция
Операционная	Блоки
Аппаратная	Устройства памяти, ключи
Вычислительная	Переменные, арифметические, логические, функции
Статистическая	Очереди, таблицы
Запоминающая	Ячейки, матрицы ячеек
Группирующие	Списки, группы

Для облегчения пользователю процесса построения модели разработан т.н. язык блок-диаграмм, который позволяет упростить переход от алгоритма, определяющего процесс функционирования...

Основой пакета является программа, описывающая функционирование выделенного конечного набора объектов, и специальная программа-диспетчер, которая называется симулятор и выполняет следующие функции:

1. Обеспечение заданных маршрутов продвижения динамических объектов
2. Планирование событий, происходящих в наборе, путем регистрации времени наступления каждого события и выполнении их в нарастающей временной последовательности.
3. Регистрация статистической информации по функционированию модели.
4. Продвижение модельного времени.

Динамическими объектами являются:

транзакции (сообщения), которые представляют собой единицы исследуемых потоков и производят ряд определенных действий, продвигаясь по фиксированным структурам, представляющим собой совокупность объектов других категорий.

Операционные объекты:

блоки, задающие логику функционирования модели системы и определяющие пути движения транзактов между объектами аппаратных категорий.

Объекты аппаратной категории:

абстрактные элементы, на которые может быть декомпозировано оборудование реальной системы. Воздействуя на эти объекты, транзакты могут изменять их состояние и влиять на движение других транзактов.

Вычислительные объекты:

служат для описания таких ситуаций, когда связи между компонентами моделируемой системы наиболее просто и компактно отображаются в виде математических соотношений.

Функции:

используя их, пользователь может задавать непрерывную или дискретную функциональную зависимость между аргументом функции и ее значением. Функции GPSS задаются табличным способом с помощью оператора описания функции

Очереди:

в любой системе движение потоков транзакций может быть задержано из-за недоступности ресурсов (например, необходимое устройство уже занято). В этом случае задержанные транзакции становятся в очередь. Учет этих очередей составляет одну из основных функций интерпретатора. Пользователь может специально определить точки модели, в которых необходимо собирать статистику об очередях, т.е. установить регистраторы. В

в этом случае интерпретатор автоматически собирает статистику об очередях (длину, среднее время нахождения транзакции в очереди и т.д.). Вся эта информация является СЧА – стандартным числовым атрибутом и доступна пользователю в процессе моделирования. Интерпретатор также автоматически поддерживает дисциплину обслуживания FIFO. Пользователь может получить стандартную статистическую информацию только от таких очередях. Если же есть необходимость реализовать очередь из транзакций другой дисциплины обслуживания, то для этого используются списки пользователей (позволяют также получить синхронизацию движения транзакций по модели).

Объект таблица предназначен для сбора статистики по случайным величинам, задаваемым пользователем. Таблица состоит из частотных классов, в которые заносится число попаданий конкретной величины (СЧА). Вычисляется математическое ожидание и среднеквадратическое отклонение. В процессе моделирования системы одни объекты взаимодействуют с другими, в результате чего происходит изменение атрибутов и преобразование их значений. Такие преобразования называются событиями.

Транзакции моделируют прохождение по системе соответствующих единиц исследуемого потока. Такое движение может быть разбито на ряд элементарных событий, происходящих в определенные моменты времени.

Основной задачей симулятора является определение времени, распределение событий и расположение их в правильной временной последовательности и выполнение соответствующий последовательностей действий при наступлении каждого из событий. Все отрезки времени описываются целыми числами (кроме GPSS ..), поэтому перед составлением модели необходимо провести временное масштабирование для всех временных параметров.

## Особенности модельного времени GPSS – САМОСТОЯТЕЛЬНО

Каждому объектам соответствуют атрибуты, описывающие состояние объекта в данный момент времени. Значения атрибутов могут быть арифметические или логические. Большая часть атрибутов недоступна пользователю. А вот атрибуты, которые доступны, которые можно адресовать, называются стандартными числовыми или стандартными логическими. Атрибуты имеют имена и эти имена как правило очень короткие. При создании своих собственных имен желательно использовать более трех символов.

В языке GPSS имеется два основных типа объектов: транзакции и блоки (?). Практически все изменения состояний модели происходит при переходе транзакций в блоки (?). С блоками непосредственно связаны

- операционные блоки, изменяющие процесс моделирования,
- блоки вывода и печать промежуточных результатов,
- команды управляющие процессом моделирования,
- команды осуществляющие редактирование результатов.

Всем блокам присваиваются порядковые номера.

Транзакты представляют собой описание динамических процессов реальной системы. Они могут определять реальные физические объекты и нефизические объекты.

Пример: канальная программа

Транзакты можно генерировать и уничтожать.

Основной атрибут любого транзакта – это его параметры, число которых для каждого транзакта колеблется от 0 до 1020. Параметры обозначаются как  $P_x$ : номер параметра x + тип параметра. может быть слово – S, полуслово – H, байт – B, плавающая точка – L. Также у транзакта есть такая характеристика как приоритетность. Меняется от 0 до 100000. Когда два транзакта соперничают за что-то, первым обрабатывается тот, у которого приоритет выше. Если приоритеты одинаковы, то сначала обрабатывается тот, у которого время ожидания обработки больше. В одном задании может выполняться как один так и несколько прогонов модели. При этом текущим значением абсолютного времени A<sub>1</sub> модели будет называться суммарное время по всем реализованным прогонам, а текущим значением относительного времени модели – C<sub>1</sub> – системное время в пределах одного прогона.

### **Классификация блоков GPSS:**

Блоки используются для описания функций моделируемой системы и управляют движение транзактов. У каждого блока имеется два стандартных числовых атрибута:

- W<sub>n</sub> – счетчик входов в блок. Как правило содержит номер транзакта, входящего в данных блок.
- N<sub>n</sub> – общий счетчик транзактов, поступивших в блок с начального момента моделирования.

Оба счетчика меняют свое содержимое автоматически.

1. Временная задержка ADVANCE  
Генерация и уничтожение транзактов GENERATE TERMINATE SPLIT ASSEMBLE  
Синхронизация движения нескольких транзактов MATCH GATHER  
Изменение параметров транзакции ASSIGN INDEX MARK  
Изменение приоритетов PRIORITY
  2. Блоки, изменяющие последовательность передвижения транзактов (блоки передачи управления) TRANSFER, LOOP, TEST, GATE
  3. Блоки, связанные с группирующей категорией. JOIN REMOVE EXAMINE SCAN  
ALTER
  4. Блоки, сохраняющие значения. SAVEVALUE ASAVEVALUE
  5. Блоки, организующие использование объектов аппаратной категории. Устройства.  
SEIZE RELEASE (парные команды)  
PREEMPT RETURN  
FAVAIL FUNAVAIL
- Памяти.  
ENTER LEAVE  
SAVAIL SUNAVAIL
- Ключи. LOGIC
6. Блоки, обеспечивающие получение статистических результатов.  
QUEUE DEPART  
TABULATE TABLE
  7. Специальные блоки  
HELP TRACE UNTRACE PRINT и еще куча

8.  
Блоки для организации цепей  
LINK UNLINK
9.  
Вспомогательные  
LOAD SAVE .....

Каждый блок определяется с помощью отдельной команды.

В общем случае:

сначала идет нумерация (как в васике), затем обязательно поле метки, затем поле операции, поле operandов, затем, если необходимо, комментарий (через ; - точку с запятой)