

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:

$F = (P'(t), P(t), \lambda)$. Лямбда – 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 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 момент t система уже находилась в состоянии S_1 и за время dt не вышла из него.
2. В момент времени t система находилась в состоянии S_3 и за время dt перешла в состояние S_1 .

Вероятность первого варианта: $P_1(t)(1-\lambda_{12}dt)$

Аналогично вероятность второго варианта: $P_3(t) \lambda_{31}dt$

Найдем вероятность того что система находится в состоянии S_1 . Она равна сумме:

$$P_1(t+dt) = P_1(t)(1-\lambda_{12}dt) + P_3(t) \lambda_{31}dt$$

Раскроем скобки в правой части, перенес P_1 в левую часть и разделим все на dt :

$$[P_1(t+dt) - P_1(t)] / dt = -\lambda_{12}P_1(t) + \lambda_{31}P_3(t)$$

при $dt \rightarrow 0$

$$P_1'(t) = -\lambda_{12}P_1(t) + \lambda_{31}P_3(t) - \text{уравнение Колмогорова для } S_1.$$

Рассмотрим второе состояние S_2 и найдем $P_2(t + dt)$, то есть вероятность того, что система будет находиться в состоянии S_2 . Варианты:

1. В момент времени t система уже была в состоянии S_2 и за время dt не перешла ни в S_3 ни в S_4 .
2. Система была в состоянии S_1 и за время dt пришла в состояние S_2
3. Система была в состоянии S_4 и за время dt пришла в состояние S_2

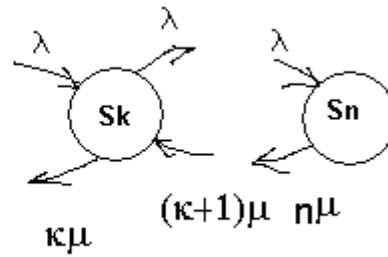
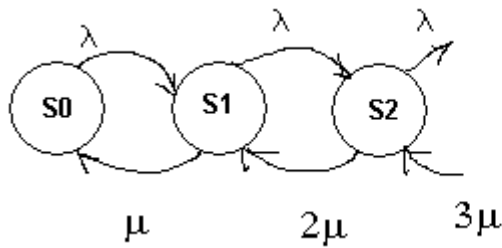
Аналогичные уравнения для всех состояний...

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 S_1 , 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 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.

Разметим граф, проставив у стрелок интенсивности соответствующих потоков событий.



Пусть система в состоянии $S1$, тогда как только закончится обслуживание заявки, занимающей канал, система перейдет в состояние $S0$.

<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.

$P_0 = [1 + \rho/1! + \rho^2/2! + \dots + \rho^n/n!]^{-1}$

$P_k = \rho^k / k! * P_0$

Зная все вероятности состояний, можно найти характеристики эффективности системы.

Вероятность отказа для данной системы. Отказ – все n каналов заняты.

$P_{\text{отказ}} = P_n = \rho^n / n! * P_0$

Относительная пропускная способность q – это вероятность того, что заявка будет принята к обслуживанию.

$q = 1 - P_{\text{отказ}}$

Среднее число заявок обслуживаемых 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 λ равный $1 / (\text{время работы})$ является усредненной характеристикой пользователей, а параметр μ – 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 μ ($t = 1/\mu = n_{\text{пр}}/V_{\text{пр}}$). 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 установившемся 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.

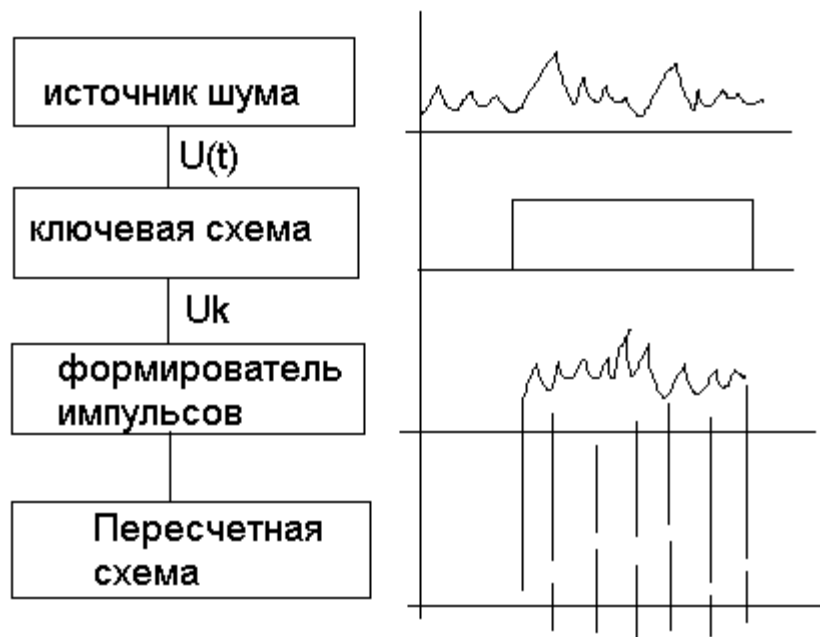
К недостаткам метода также относится большой объем требуемых вычислений, равный n . Отсюда и важность выбора величины n . Уменьшение количества испытаний повышает экономичность расчетов, но ухудшает их точность.

Способы получения последовательностей случайных чисел.

На практике используются три основных способа генерации случайных чисел:

1. Аппаратный. Случайные числа вырабатываются 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	<ol style="list-style-type: none"> 1. Запас чисел неограничен 2. Расходуется мало операций 3. Не занимает место в оперативной памяти. 	<ol style="list-style-type: none"> 1. Требуется периодическая проверка на случайность 2. Нельзя воспроизводить последовательности 3. Используются специальные устройства. Надо стабилизировать
Табличный	<ol style="list-style-type: none"> 1. Требуется однократная проверка 2. Можно воспроизводить последовательности 	<ol style="list-style-type: none"> 1. Запас чисел ограничен 2. Занимает место в оперативной памяти и требуется время на обращение к памяти
Алгоритмический	<ol style="list-style-type: none"> 1. Однократная проверка 2. Можно многократно воспроизводить последовательности чисел 	<ol style="list-style-type: none"> 1. Запас чисел последовательности ограничен её периодом 2. Требуются затраты машинного времени

	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:

1. Числовые значения заключены в интервале от 0 до 1. $0 \leq F(x) \leq 1$
2. Если $x_1 \leq x_2$, то $F(x_1) \leq F(x_2)$. Т.е. функция распределения неубывающая.
3. $F(x) \rightarrow 0$ при $x \rightarrow -\infty$ и $F(x) \rightarrow 1$ при $x \rightarrow \infty$

Общие свойства плотности вероятности:

1. Связь функции вероятности и плотности вероятности определяется формулой $f(x) = dF(x) / dx$ или $F(x) = \int_{-\infty}^x f(x) dx$
2. Плотность вероятности неотрицательная функция
3. $\int_{-\infty}^{\infty} f(x) dx = 1$

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					

1. Равномерное (вывести функцию и плотность). Оцифровка осей.
2. Выбирается распределение по списку в журнале

Немарковские случайные процессы, сводящиеся к Марковским.

Реальные процессы очень часто обладают последствием 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 $\min \int_{t_1}^{t_2} [\lambda_{i \text{ экв}}(\tau) - \lambda_i(\tau)] d\tau$

где $\lambda_{i \text{ экв}}$ – эквивалентная интенсивность перехода в i -ой группе переходов, заменяющей реальный переход, который обладает интенсивностью λ_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 λ , причем поток отказов Пуассоновский. После отказа устройство восстанавливается. Время восстановления распределено по закону Эрланга третьего порядка. Функция плотности распределения $f_2(t) = 0.5\mu (\mu t)^2 e^{-\mu t}$

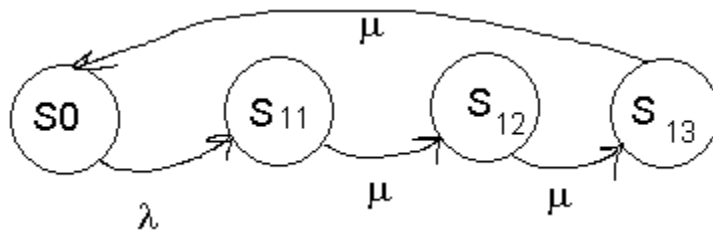
Найти предельные вероятности возможных состояний системы

Пусть S может принимать два возможных состояния:

S_0 – устройство исправно

S_1 – устройство отказало и восстанавливается

Переход S_0 - S_1 осуществляется под воздействием Пуассоновского потока, а переход S_1 - S_0 – потока Эрланга. Представим случайное время восстановления в виде суммы трех случайных времен, распределенных по показательному закону с интенсивностью μ .



$$0 = -\lambda P_0 + \mu P_{13}$$

$$0 = -\mu P_{11} + \lambda P_0$$

$$0 = -\mu P_{12} + \mu P_{11}$$

$$0 = -\mu P_{13} + \mu P_{12}$$

$$P_0 + P_{11} + P_{12} + P_{13} = 1$$

$$P_0 = \mu / \lambda * P_{13}$$

$$P_1 =$$

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 n 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 .. бла-бла-бла.
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 число состояний больше 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 мультипликативного 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 \rightarrow), 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 λ . Используют метод точек, в основе которого лежит генерация случайной переменной R_i , равномерно 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: λ 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.

- А- закон распределения интервала времени между поступлениями заявок. Наиболее часто используются следующие: экспоненциальное – m , Эрланга – e , гиперэкспоненциальное – h , гамма распределение – Γ , детерминированное – D , произвольное – G .
- В- закон распределения времени обслуживания приборов СМО. Обозначения такие же как и А.
- С- число обслуживающих приборов. Для одноканальной системы – 1, для многоканальной в общем случае – a .
- Д- 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 .
- Е- дисциплина обслуживания. При FIFO данное обозначение может отсутствовать

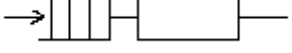
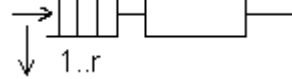
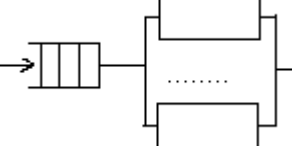
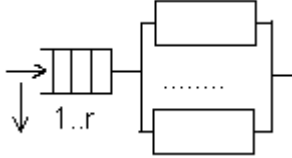
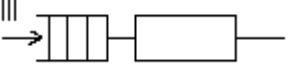
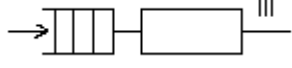
Пример: М/М/1 – СМО с одним обслуживающим прибором, бесконечной очередью, экспоненциальными законами распределения интервалов времени между поступлениями заявок и временем обслуживания. Дисциплина – FIFO

Е/Н/1/ 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.

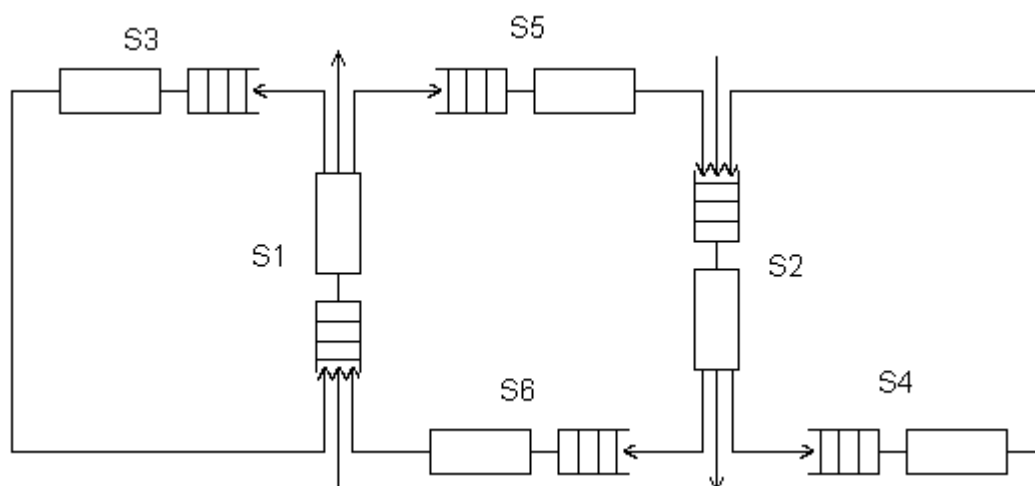
5. Одноканальные СМО с групповым поступлением заявок. Представляют собой один обслуживающий прибор с бесконечной очередью. Перед обслуживанием заявки группируются в пакеты по определенному правилу.
 6. Одноканальные СМО с групповым обслуживанием заявок, представляющие собой один прибор с бесконечной 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 ожидание	$G/G/l$	
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

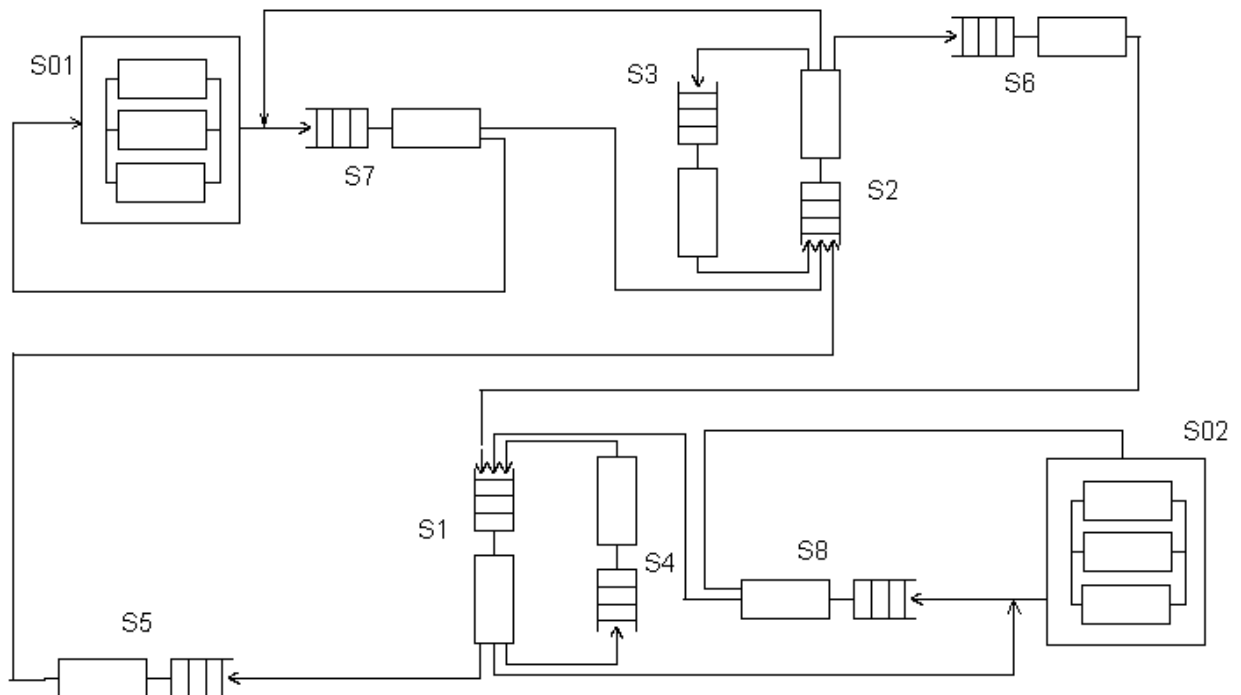
S5,S6 – работу межузловых каналов.

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 отправки 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

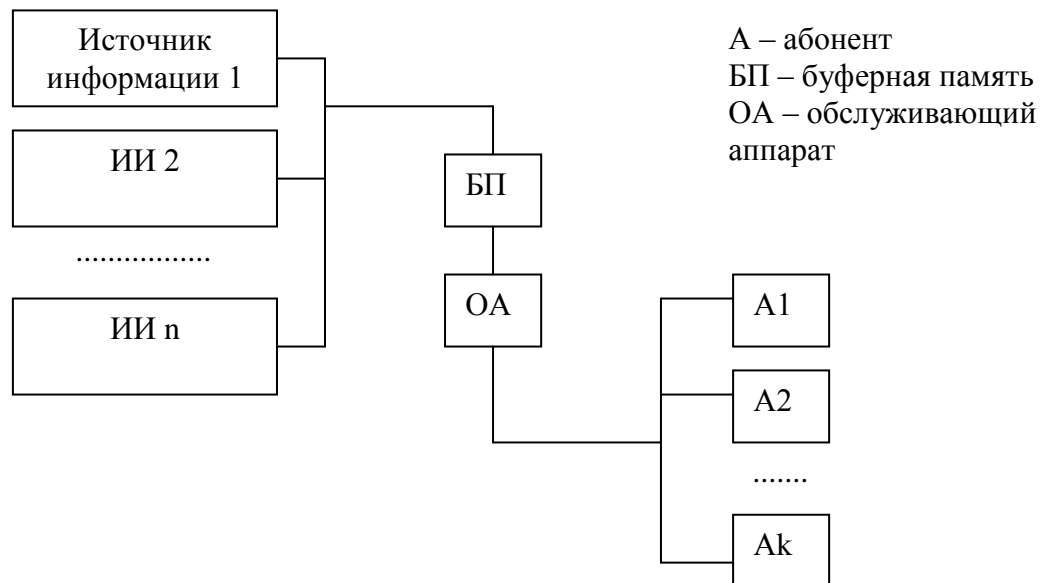
Абоненты с терминалов посылают запросы, которые по каналам связи поступают на узлы коммутации, а оттуда на обработку в свой или соседний сервер.

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.

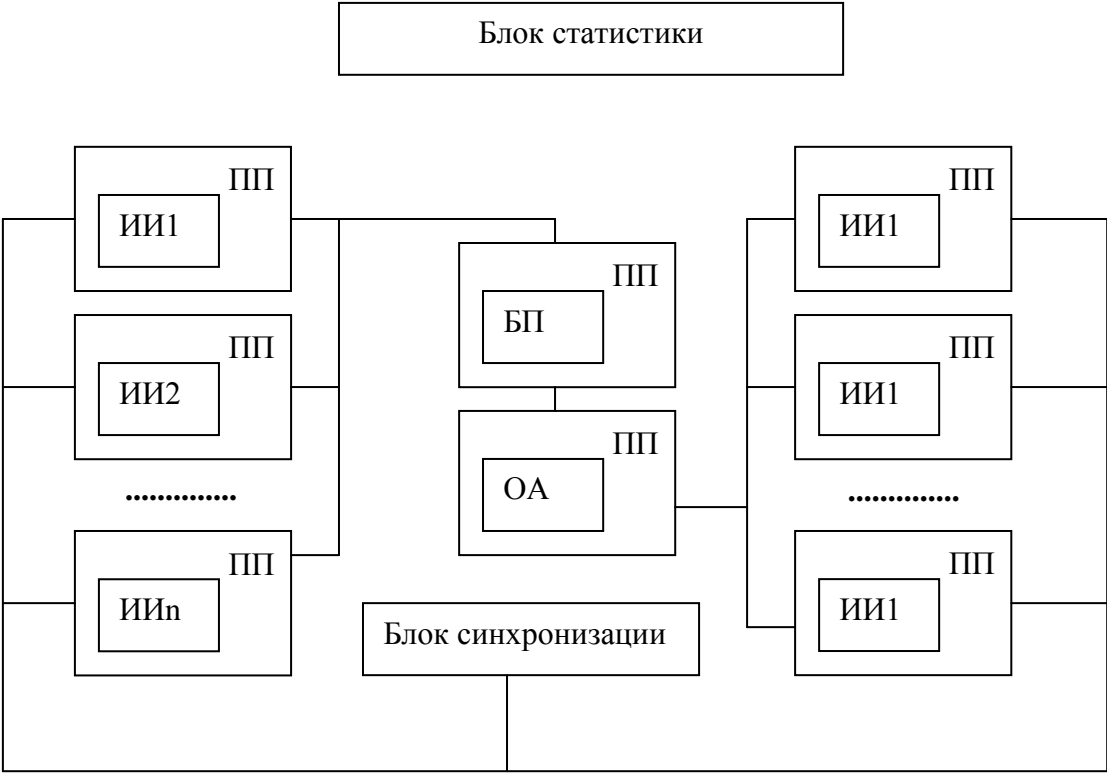


Источники информации подают на вход буферной памяти независимо друг от друга поток сообщений. Закон появления этих сообщений произволен, но обязательно задан. В буферной памяти сообщения как правило записываются навалом (как пришли) и выбираются по одному в обслуживающий аппарат (как правило по принципу 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. $T = T + T_i$.

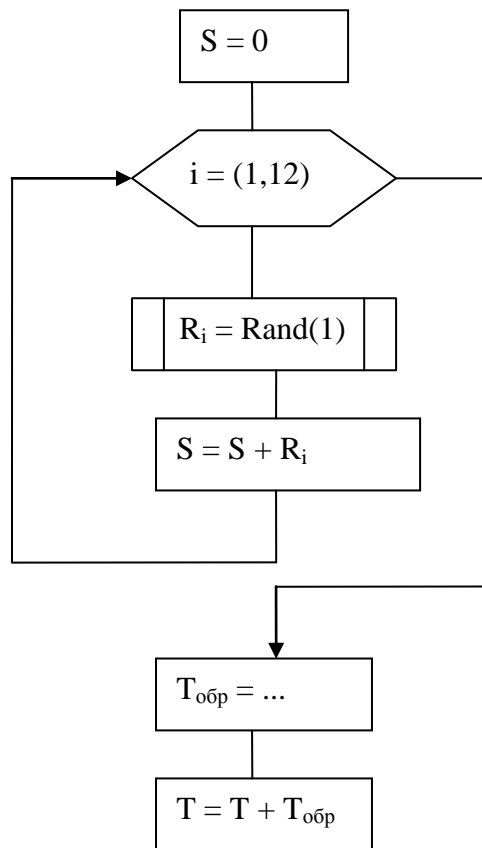
Типы распределения	Выражение для расчета времени
Равномерное на отрезке [A;B]	$T_i = A + (B - A) \cdot R_i$
Экспоненциальное	$T_i = 1/\lambda \ln(1 - R_i)$
Нормальное (Гауссово)	$T_i = \sigma_x \sqrt{\frac{12}{n} (\sum_{i=1}^n R_i - \frac{n}{2})} + m_x$
Распределение Эрланга	$T_i = 1/(k\lambda) \ln(1 - R_i)$

Моделирование работы обслуживающего аппарата.

Программа, имитирующая работу обслуживающего аппарата – это набор процедур, вырабатывающих случайные отрезки времени, соответствующие длительностям обслуживания требования. Например,

M_x, σ_x , нормальное

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



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 текущем адресу, изменение текущего адреса на 1

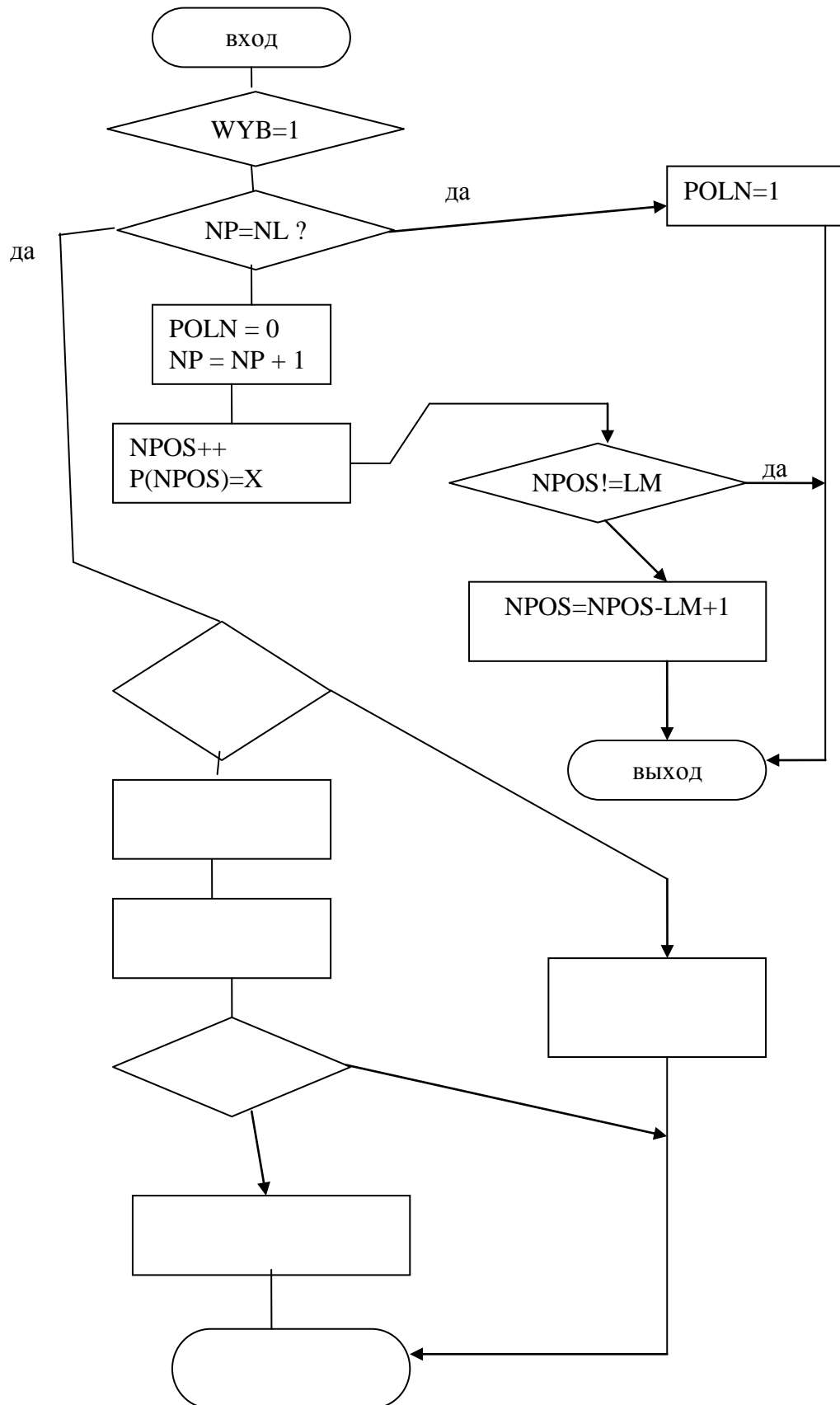
выход

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.

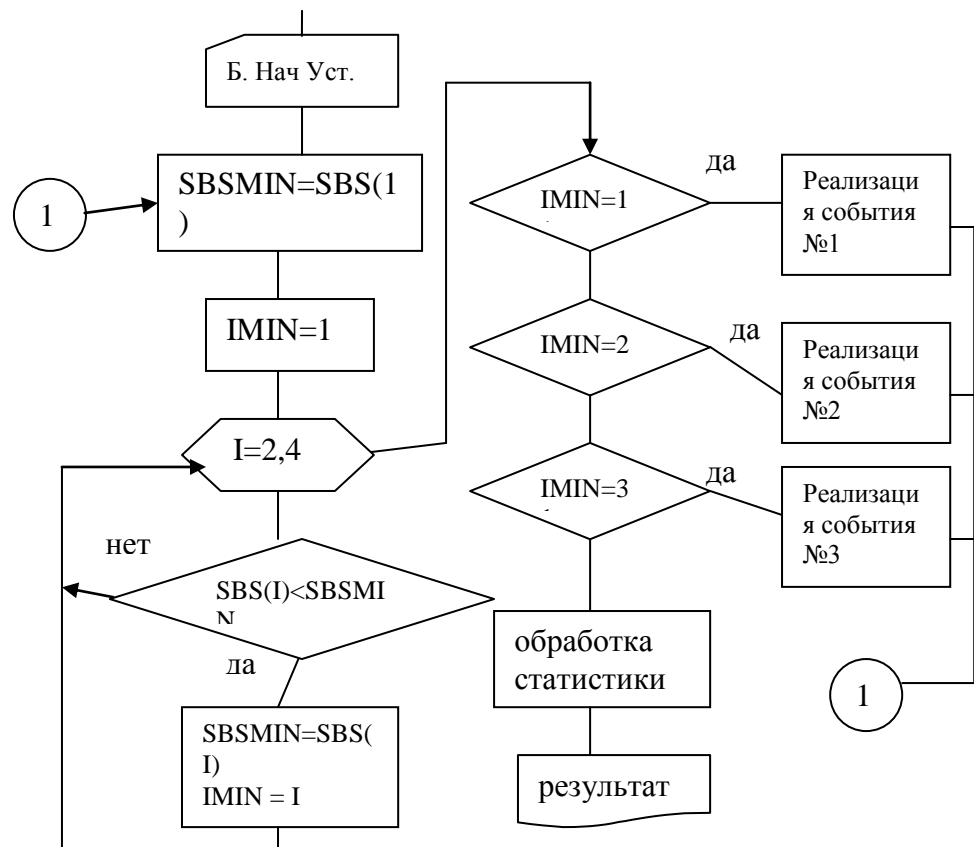
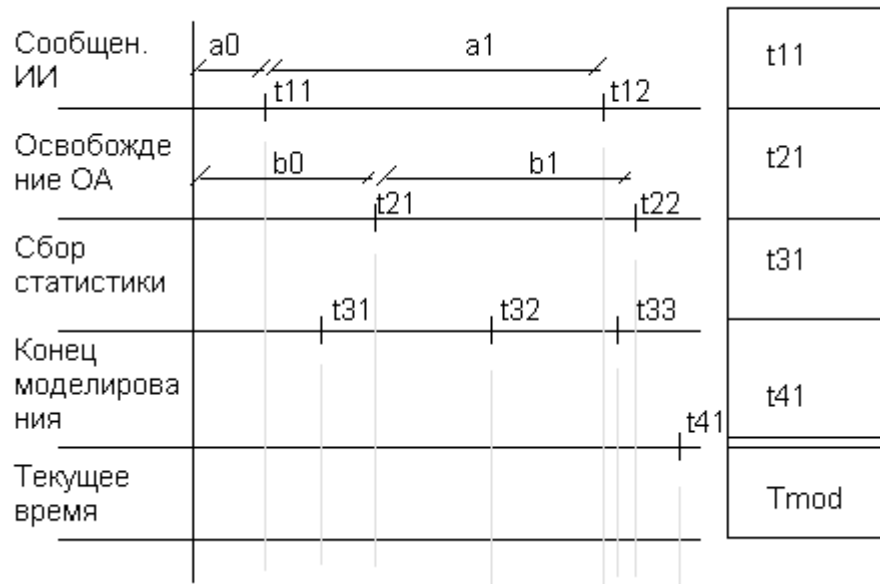
Принцип Δt : вся временная ось разбивается на равные отрезки, кратные Δt . Далее просматривается какие системные события происходят в эти промежутки. Недостаток данного алгоритма: если Δt выбрано большим, то большая вероятность, что какие-то события будут пропущены, и этого даже не будет заметно. В случае малого Δt результатов придется ждать долго.

Поэтому стали привязывать время однозначно к событию. Такой принцип лишен недостатков «принципа Δt ». 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 Δt .** 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 Δt 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.



t_{11}, t_{12} – время появления событий....
 t_{21}, t_{22} - Время обслуживания первого сообщения
 t_{31}, t_{32}, t_{33} - Моменты сбора статистики:
 t_{41} – момент окончания моделирования

SBS – список будущих событий.

Методика реализации событийного принципа:

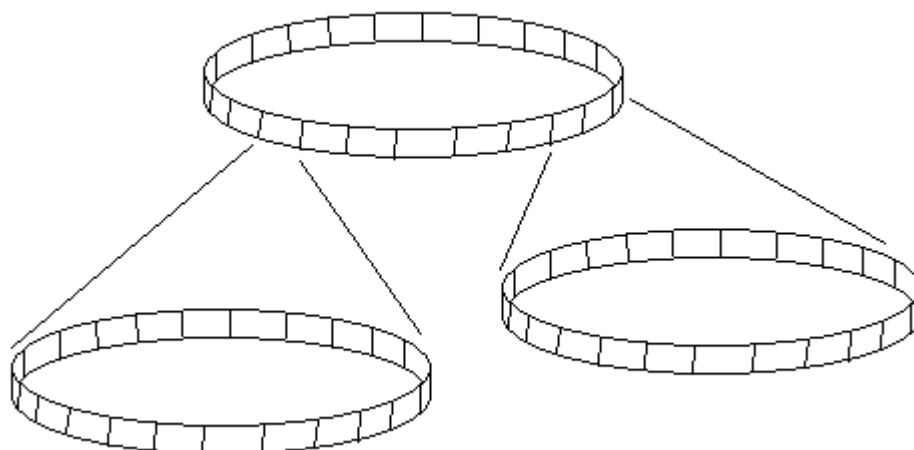
1. Для всех активных блоков (блоков, порождающих события) заводится свой элемент в одномерном массиве – списке будущих событий
2. В качестве подготовительной операции в список будущих событий заносится время ближайшего события от любого активного блока. Активизируя программный имитатор источника информации 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 Дельфты.

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 Δt , 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.

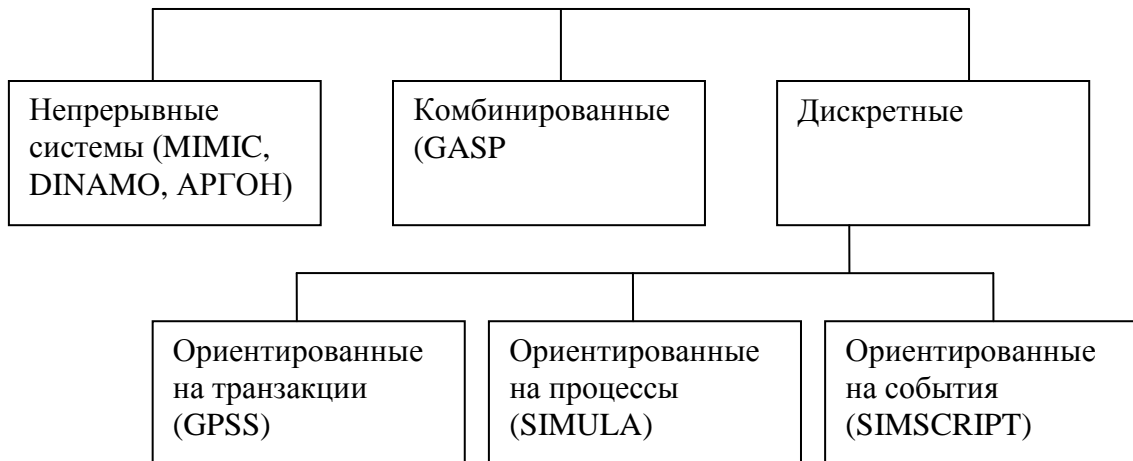
Качество языков моделирования характеризуется:

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 функциями. 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 единым 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:

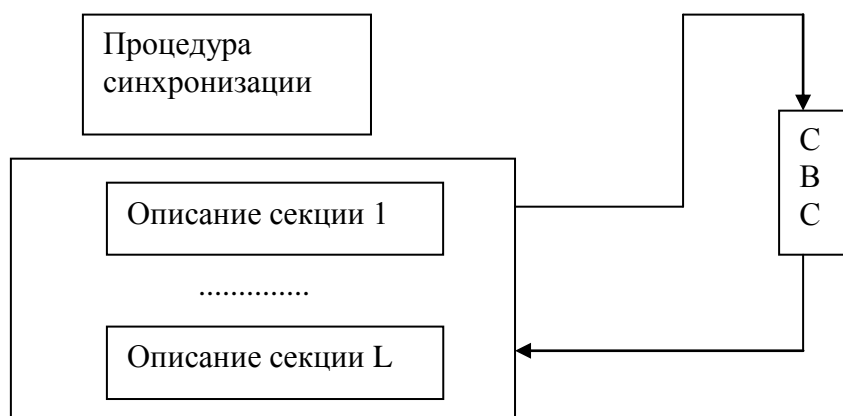
1. Необходимо задать правила, определяющие виды процессов, происходящих в системе.
2. Указать значения атрибутов или задать правила генерации этих атрибутов. При этом как правило системы описываются на определенном уровне детализации в терминах множества описаний процессов, каждый из которых включает 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.

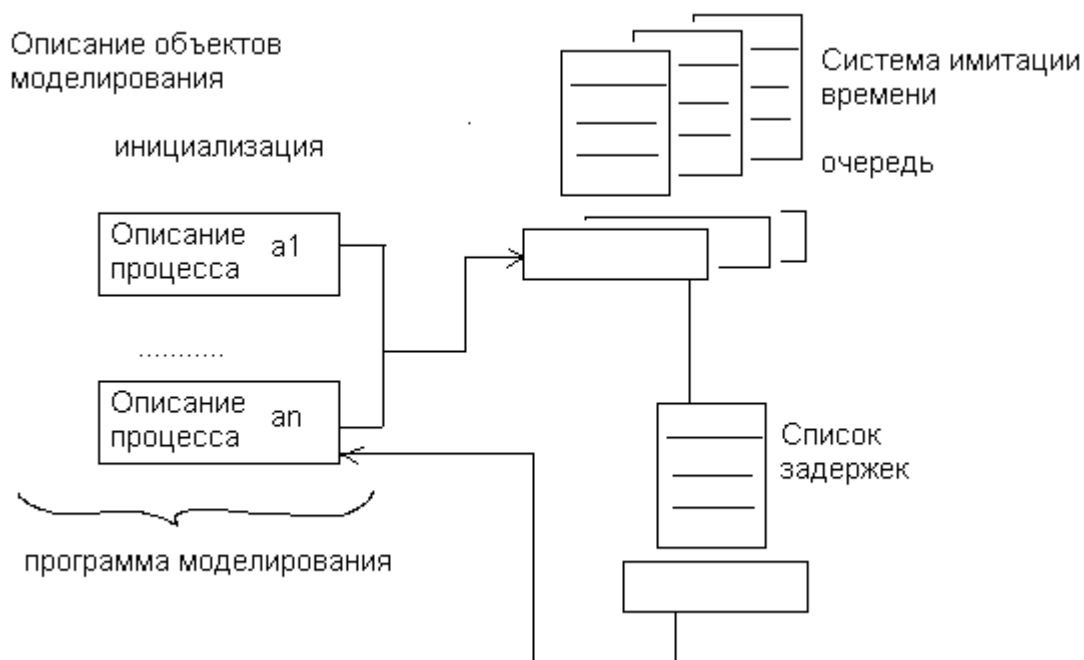
Моделирующая программа организована в виде совокупности секций. Секции включают в себя события (процедуры). Процедура событий состоит из набора операций, которые в общем случае выполняются после завершения какой-либо активности. Выполнение процедуры синхронизируется во времени списком будущих событий.



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 Условие, То Событие1 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 – общецелевая 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 посторен ...

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 систем 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

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 очередей. 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 объекту 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_1 модели будет называться суммарное время по всем реализованным прогонам, а текущим значением относительного времени модели – C_1 – системное время в пределах одного прогона.

Классификация блоков GPSS:

Блоки используются для описания функций моделируемой системы и управляют движением транзактов. У каждого блока имеется два стандартных числовых атрибута:

- W_n – счетчик входов в блок. Как правило содержит номер транзакта, входящего в данный блок.
- N_n – 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 операндов, затем, если необходимо, комментарий (через ; - точку с запятой)