ЛЕКЦИИ

по методам моделирования

МГТУ им. Н.Э.Баумана

Содержание

[1.09][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] [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.

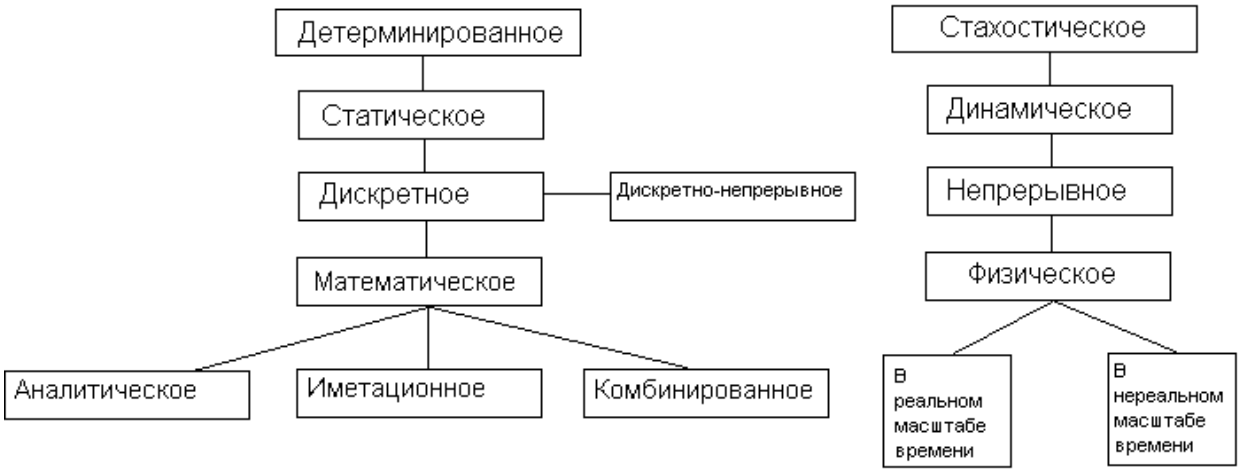
В основе моделирования лежит теория подобия, которая утверждает, что абсолютное подобие имеет место только при замене одного объекта другим точно таким же.

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 протекании 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][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.

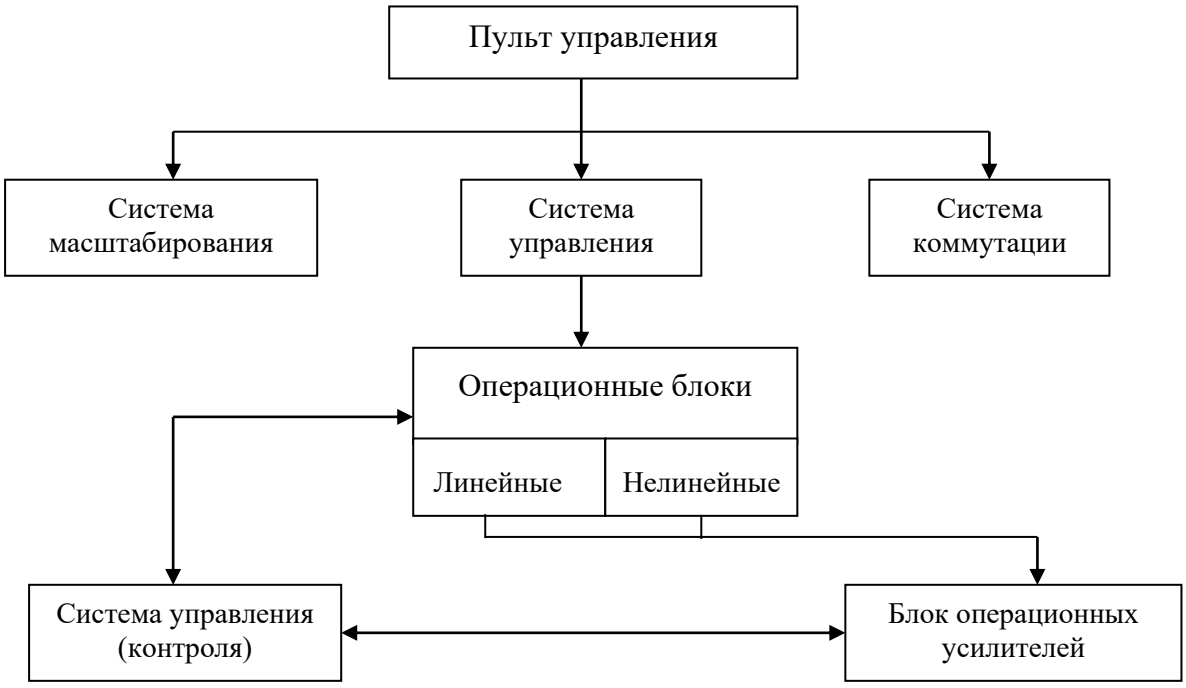
(!) Вспомнить про цифровые компьютеры.

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 (аналоговые 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):

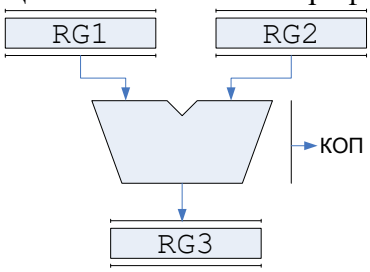
- малые ($n < 10$),
- средние ($10 \le n \le 20$),
- большие аналоговые комплексы ($n > 20$)



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

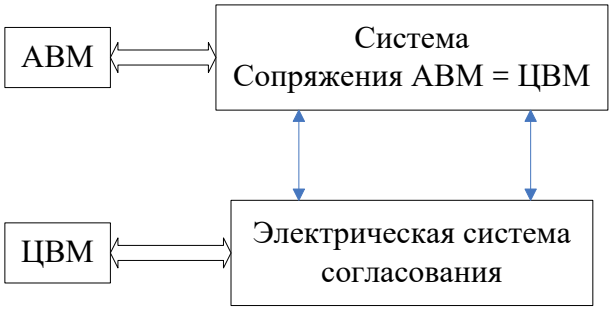
Подклассы гибридных ВМ:

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:

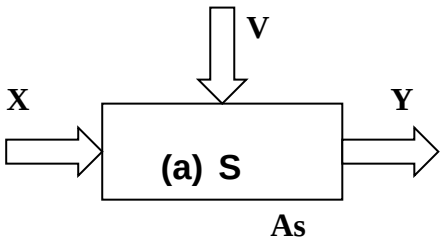
Тип информации	Непрерывный	Дискретный
Изменение значений	Величиной напряжения	Числовым значением
Базовые операции	Арифметические операции и интегрирование	Арифметические операции
Принцип вычисления	Высокопараллельный	Последовательно-параллельный

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

[15.09][Лекция 4]

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:

$x_i \in X, i = \overline{1, n_x}$	$\overrightarrow{x(t)} = \{x_1(t), x_2(t), ..., x_{nx}(t)\}$	Множество входных параметров
$h_j \in H, j = \overline{1, n_h}$	$\overrightarrow{h(t)} = \{h_1(t), h_2(t), ..., h_{nh}(t)\}$	Множество внутренних параметров
$v_k \in V, k = \overline{1, n_v}$	$\overrightarrow{v(t)} = \{v_1(t), v_2(t), ..., v_{nv}(t)\}$	Внешнее воздействие
$y_m \in Y, m = \overline{1, n_m}$	$\overrightarrow{y(t)} = \{y_1(t), y_2(t), ..., y_{ny}(t)\}$	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:

$$\overrightarrow{x(t)} = \{x_1(t), x_2(t), ..., x_{nx}(t)\}$$
$$\overrightarrow{h(t)} = \{h_1(t), h_2(t), ..., h_{nh}(t)\}$$
$$\overrightarrow{v(t)} = \{v_1(t), v_2(t), ..., v_{nv}(t)\}$$

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

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

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

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

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. Весьма 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 $\boldsymbol{F_s}$ 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:

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

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

Если рассматривать процесс функционирования системы как последовательную смену состояний $z_1(t), z_2(t), \ldots, z_k(t)$, 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 $t_0 \le t \le T^*$ полностью определяется начальными условиями $\overrightarrow{z^0(t)} = \{z_1^0(t), z_2^0(t), \ldots, z_{nz}^0(t)\}$, 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 $t_0 \le t \le T^*$, а может быть и дискретным, т.е. квантованным на отрезке Δt: $T^* = m\Delta t$, где m - число интервалов дискретизации.

Типовые математические схемы.

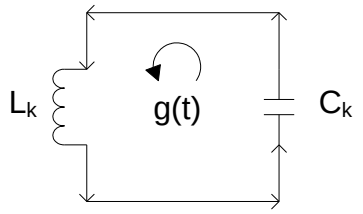
В практике моделирования на первоначальных этапах формализации объектов используют так называемые ***типовые математические схемы***, к которым относят такие хорошо проработанные (разработанные) математические объекты, как дифференциальные алгебраические уравнения, конечные вероятностные автоматы и т.д.

процесс функционирования системы	**типовая математическая схема**	**обозначение**
Непрерывно-детерминированный подход	стандартные ДУ	D-схема
Дискретно-детерминированный подход	конечные автоматы	F-схема
Дискретно-стохастический подход	вероятностные автоматы	P-схема
Непрерывно-стохастический подход	система массового обслуживания	Q-схема
Обобщенные (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])

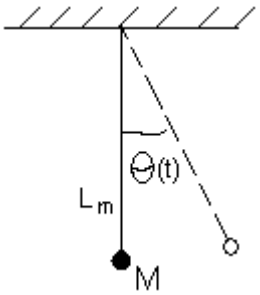
ДУ: $y' = f(y,t)$



$L_k \frac{d^2q(t)}{dt^2} + \frac{q(t)}{C_k} = 0$ - описание процессов в электрическом колебательном контуре на базе ДУ.

q(t) - заряд конденсатора в момент времени t

T = $2\pi\sqrt{L_k C_k}$ - 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

$h_2 = L_k = M_M l_M^2$

$h_1 = 0$

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

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

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:

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

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.

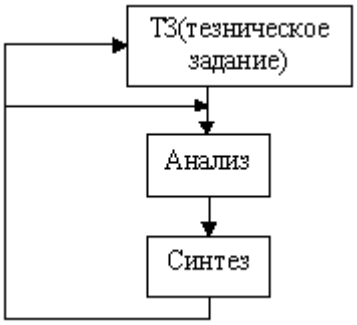
[18.09]|[Лекция 5]

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, приемлимость 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 алгоритимизации и компьютерной реализации, математическая модель сформированная на первом этапе воплощается в конекретную 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 изспользуются для проведения рабочих расчетов по готовой программе. Результаты этих расчетов позволяют проанализировать и сделать выводы о характеристиках процессов функционрирования исследуемой схемы.

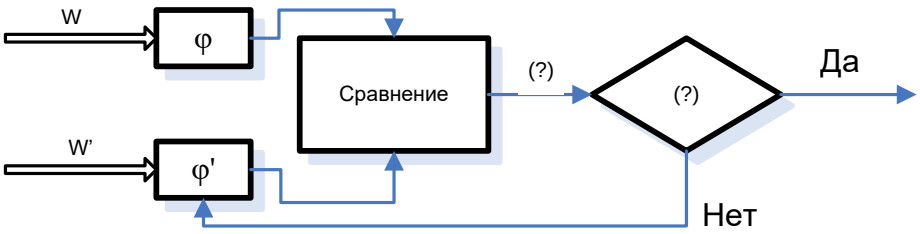
Последовательность действий:

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 ($\Delta y = \left| y_{ориг} - y_{мод} \right|$). Считают что модель адекватна с системой, если вероятность того что отклонение Δy 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 рассоглосование модели и системы, возникает необходимость в её корректировки или изменения.
Выделяют следующие *типы изменений*:

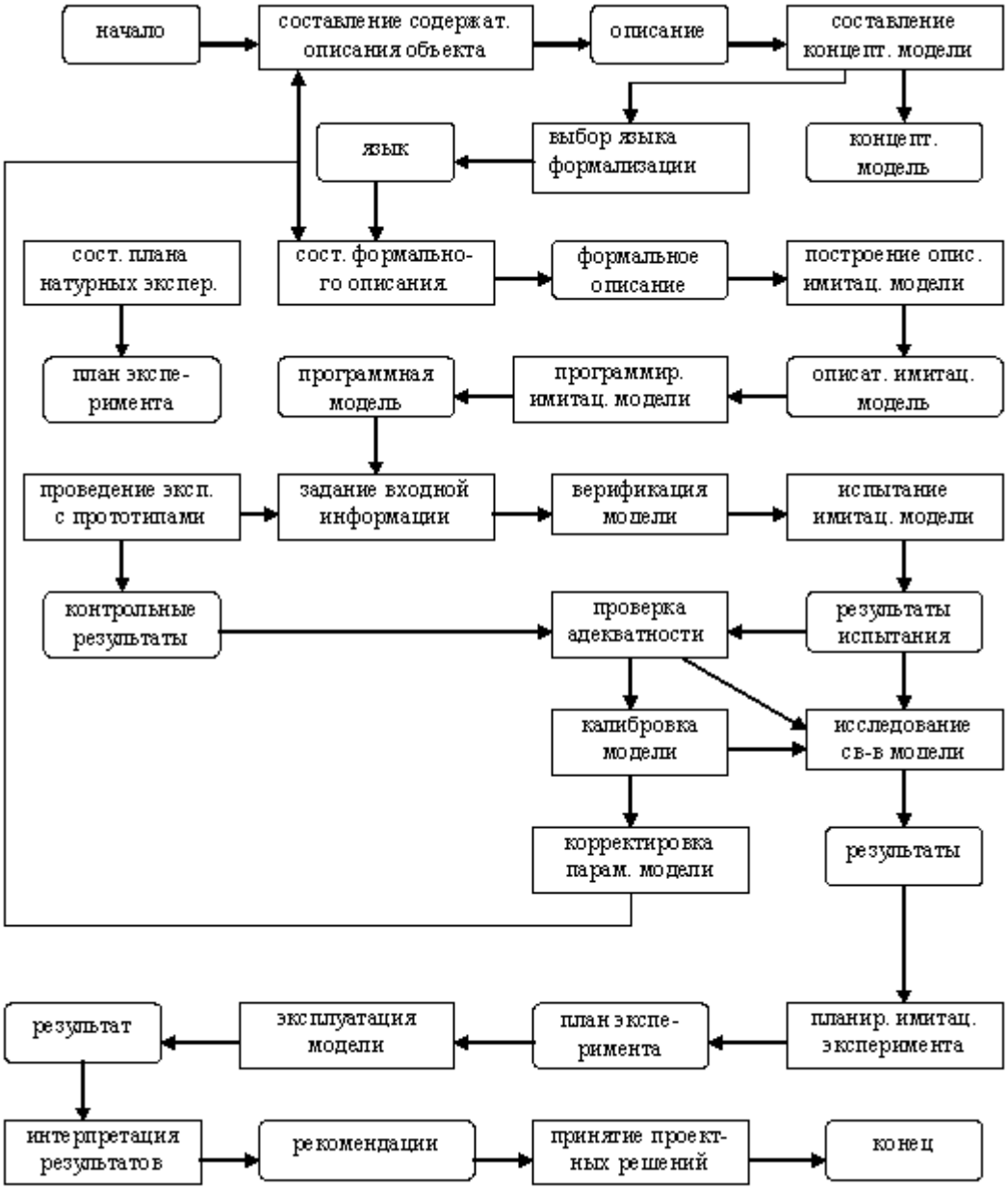
1. Глобальные. Возникают в случае методических ошибок в концептуальной или математической моделе (проще всё начать сначала, чем исправлять).
2. Локальные. Связаны с уточнением некоторых параметров и алгоритмов. Заменить компоненты модели на более точные.

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][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 X и выходы Y. Переменные X 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 X 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: $X_{\min} \le X_i \le X_{\max}$, $i = \overline{1,k}$, k - количество факторов.

Нормализация факторов – преобразование натуральных значений факторов в безразмерные кодированные величины. Переход i-ого значения задается следующей формулой: $X_i = \frac{X_i - X_{i0}}{\Delta X_i}$ - переход безразличного фактора X_i

X_i - натуральное значение фактора

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

ΔX_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 равноценна 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: $U = X_{1u}, X_{2u}, ..., X_{ku}$.
Совокупность значений факторов во всех N точках плана эксперимента образует матрицу плана:

$$\Delta = \begin{pmatrix} X_{11} & \cdots & X_{k1} \\ \vdots & \ddots & \vdots \\ X_{z1} & \cdots & X_{zz} \end{pmatrix}$$

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:

$y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$, где текущий y_i 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.

$M(y) = \beta f(x)$;
β – вектор неизвестных параметров модели:
$\beta = (\beta_0, \beta_1, ..., \beta_h)$, 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 β 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 β 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:

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

ε - 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***.

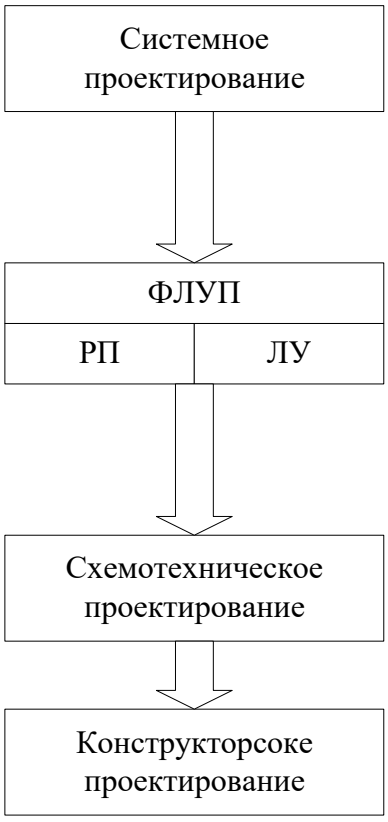
Виды планирования эксперимента.

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

[29.09][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*.



>> Информационная система - ..на самостоятельную разработку..

В качестве объектов на системном уровне проектирования нужно рассматривать процессор, память, каналы и т.д., так же нужно рассматривать ОС.

Далее идет функционально-логический уровень проектирование (ФЛУП). Входными характеристиками являются выходные параметры из системного проектирования.

ФЛУП делится на два уровня:

1) подуровень регистровых передач (нужно так же рассматривать физические характеристики устройств)
2) логический уровень

Следующий уровень: Схемотехнический уровень проектирования (УП).

Здесь и возникает проектирование интегральных схем

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

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

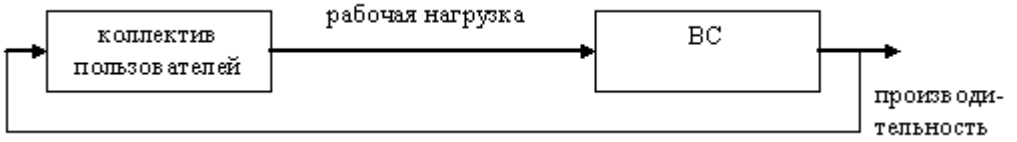
Моделирование на системном уровне

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.

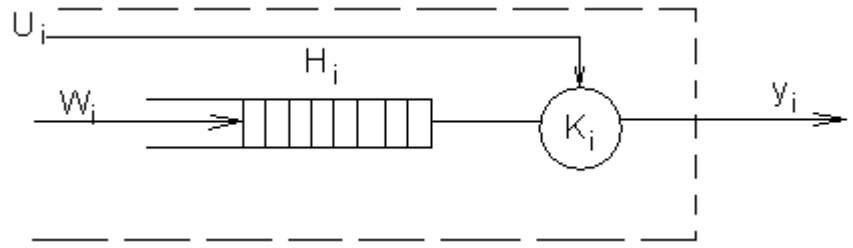
Непрерывно стохастические модели (Q-схемы)

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) состроит 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: $t_i = \{0, t_1, ..., t_n\}$, $t_{i-1} \le t_i \le t_{i+1}$

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 Δt примыкающий к моменту времени t попадает более одного события,

пренебрежительно мала по сравнению с вероятностью того что на этот же интервал Δt 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 Δt 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 Δt составит: $\lim\limits_{t\to 0}\dfrac{P_1(t,\Delta t)}{\Delta t}=\lambda(t)$ - *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 (ω_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, принадлежат 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: $\overline{Z_i}=(Z_i^n,Z_i^k)$
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][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 емкость 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 неограниченна.

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 λ - 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:

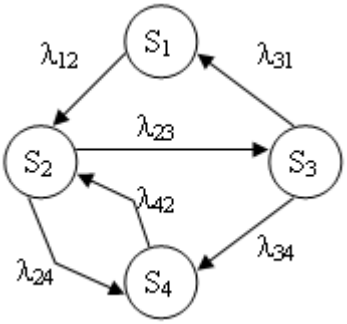
$$\lambda = \lambda(X, U, H)$$

- 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 Δt и найдем, что в момент времени $t + \Delta t$ система будет находится 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 Δt не вышла из него.
2. В момент t система была в состоянии S_3 и за время Δt 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 Δt 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.:

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

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

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

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

Мы вывели уравнение Колмогорова 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 срелке, умноженной на вероятность того состояния, из которого исходит стрелка.

Лабораторная работа №1.
Определить среднее относительное время пребывания системы в предельном стационарном состоянии. Интенсивности переходов из состояния в состояние задаются в виде матрицы размером ≤ 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:

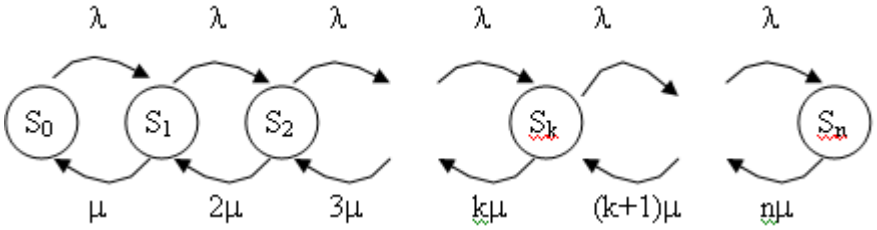
S_0 - все каналы свободны

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 λ.
Определим интенсивность потоков событий, переводящих систему справа на лево.
Пусть система находится в S_1. Тогда, когда закончится обслуживание заявки занимающей этот канал, система перейдет в S_0 => поток, переводящий систему в другое состояние, будет иметь интенсивность перехода μ. Если занято 2 канала, а не один, то интенсивность перехода составит 2μ.

Уравнения Колмогорова:

$$\begin{cases} p_0' = -p_0\lambda + p_1\mu \\ p_1' = -p_1\lambda - p_1\mu + p_0\lambda + p_2 2\mu \\ p_2' = -p_2\lambda - p_2 2\mu + p_1\lambda + p_3 3\mu \\ p_k' = -p_k\lambda - p_k k\mu + p_{k-1}\lambda + p_{k+1}(k+1)\mu \\ p_n' = p_{n-1}\lambda - p_n n\mu \end{cases}$$

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

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

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

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

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

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

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

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

$$p_{ОТК} = p_n = \frac{\rho^n}{n!} p^0$$

- *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 μ 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][Лекция 9]

Самостоятельно: Метод вложенных цепей Маркова

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

Немарковские 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

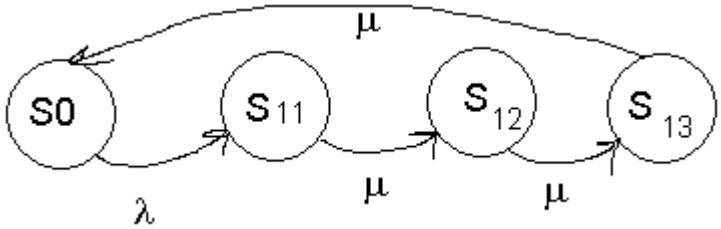
следующее: $\int_{t_1}^{t_2}(\lambda_{iэкв}(\tau)-\lambda_i(\tau))d\tau$, где λ_i - эквивалентная интенсивность перехода в i-ой группе переходов, заменяющей реальный переход, обладающий интенсивностью $\lambda_i(\tau)$.

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 Марковских 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 k 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 λ, причем поток отказов Пуассоновский. После отказа устройство восстанавливается. Время восстановления распределено по закону Эрланга 3 порядка с функцией плотности $f_2(t)=0.5\mu(\mu t)^2e^{(-\mu t)}$.

Найти предельные вероятности возможных состояний системы.
Решение.
Пусть система может принимать 2 возможных состояния:
S_0 - устройство исправно;
S_1 - устройство отказало 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 3х случайных временных интервалов, распределенных по показательному закону с интенсивностью μ.



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

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

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

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

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

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

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

Метод вложенных цепей Маркова.

Вложенные цепи Маркова образуются следующим образом. В исходном случайном 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>

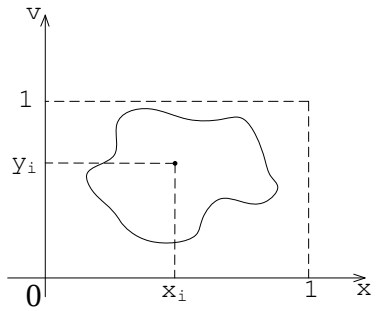
Метод статистических испытаний. Метод Монте-Карло.

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 статическими 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 2x 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 $\varepsilon \le \sqrt{\frac{1}{n}}$.

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.

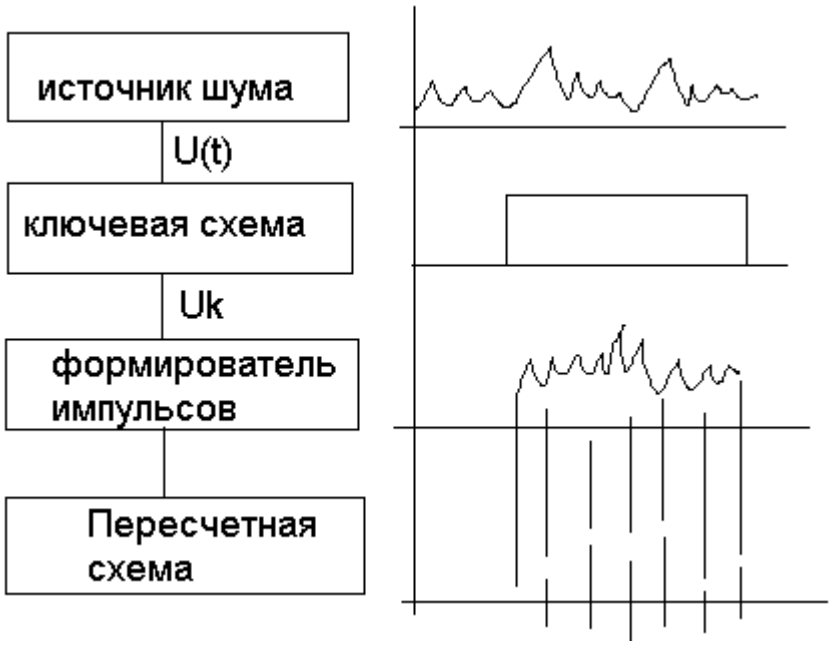
Способы получения псевдослучайных чисел.

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	1. Запас чисел ограничен 2. Занимает место в оперативной памяти и требуется время на

	последовательности	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][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.

Сферы 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 *y* 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 были независимыми, то сумма их последовательностей обладала дисперсией, равной сумме дисперсией. Иначе случайность рядов возрастет при суммировании. Сейчас в системах программирования обычно используют конгруэнтные генераторы по алгоритму, предложенному национальным бюро стандартов США, который имеет длину 2^{24}.

Генерация случайных чисел по алгоритму Зеймана.
{1, 1, 2, 3, 5, 8, 13, 21, …}
mod 10
{1, 1, 2, 3, 5, 8, 3, 1, …}
Переименовываем с помощью какого-либо ГСЧ; пусть всё так и осталось:
{1, 1, 2, 3, 5, 8, 3, 1, …}

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-ция 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][Лекция 11]

Программа генерации случайных чисел на Фортране для машин ЕС (~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**.

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

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:

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

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

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: λ 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.

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

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 σ_X может быть получена по следующей формуле:

$$x = \sigma_X \cdot \sqrt{\frac{12}{N}} \cdot (\sum_{i=1}^{N} R_i - \frac{N}{2}) + M_X$$

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.
```

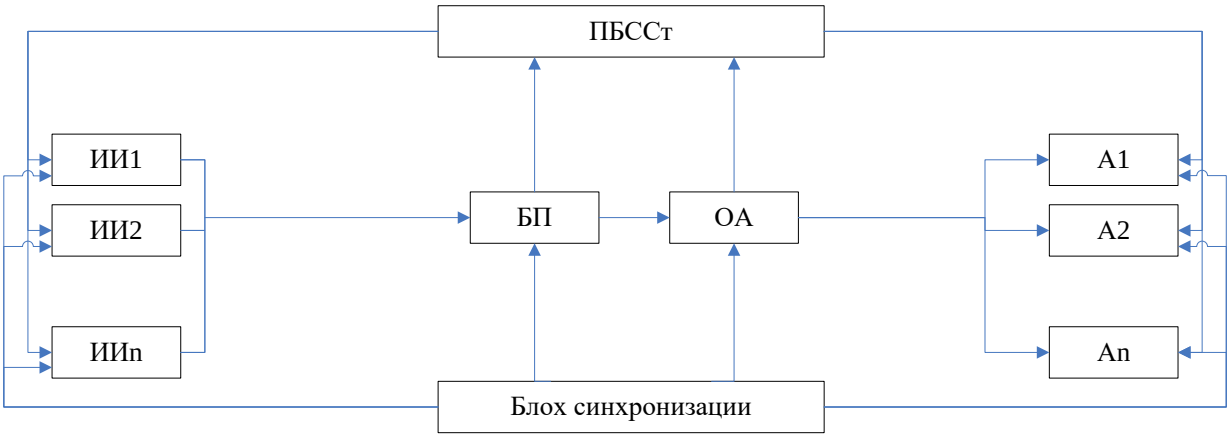
При n = 0 происходит параметрическая настройка или т.н. «установка».
При n = 1 будем получать равномерно распределенную случайную величину.
При n = 2 будет гауссово (нормальное) распределение.

["всем, пожалуйста, поиграться с этим алгоритмом и построить график" (c) 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.

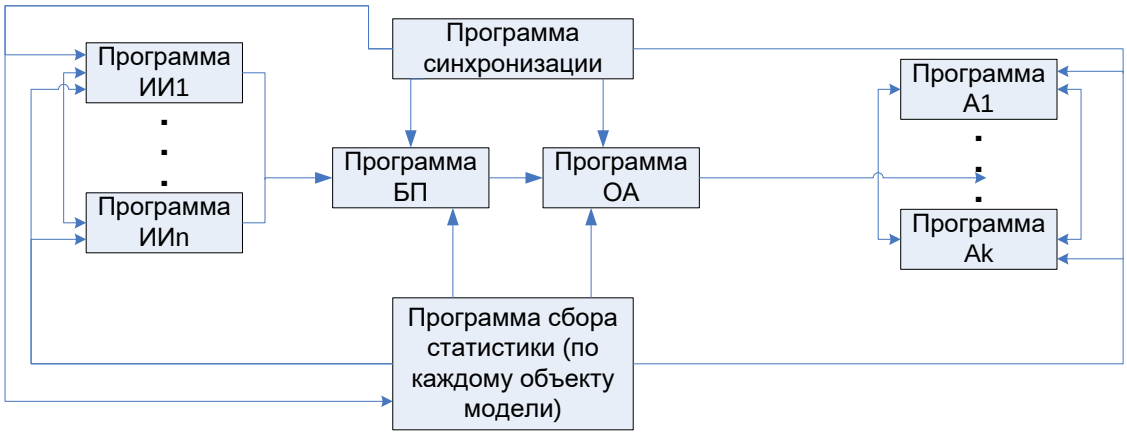


ИИ – источники информации – выдают на вход буферной памяти (БП) независимые друг от друга сообщения. Закон появления сообщений – произвольный, но задан на перед.

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:

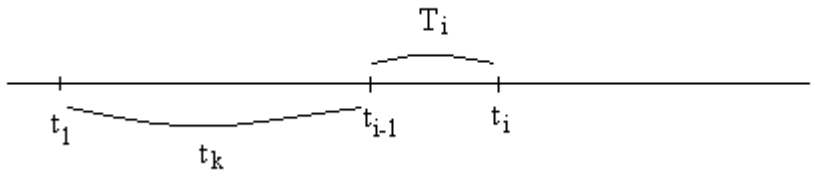


Должна быть обязательно программа сбора статистики (**ПБССт –** программный блок сбора статистики). Причем статистику программа должна собирать по каждому из объектов модели. Так же должна быть программа, которая позволит "оживить" систему

– 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

```
// процедурка равномерного распределения псевдослучайных чисел на итервале [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 разлчным распределением:

Вид распределения	**Выражение**
равномерное на [a,b]	$T_i = a + (b-a)R$
нормальное	$T_i = \sigma_X \sqrt{\frac{12}{n}} (\sum_{i=1}^{n} R_i - \frac{n}{2}) + M_X$, n ~= 12

экспоненциальное	$T_i = -\frac{1}{\lambda}\ln(1-R)$
Эрланга	$T_i = \frac{1}{k\lambda}\sum_{i=1}^{k}\ln(1-R_i)$

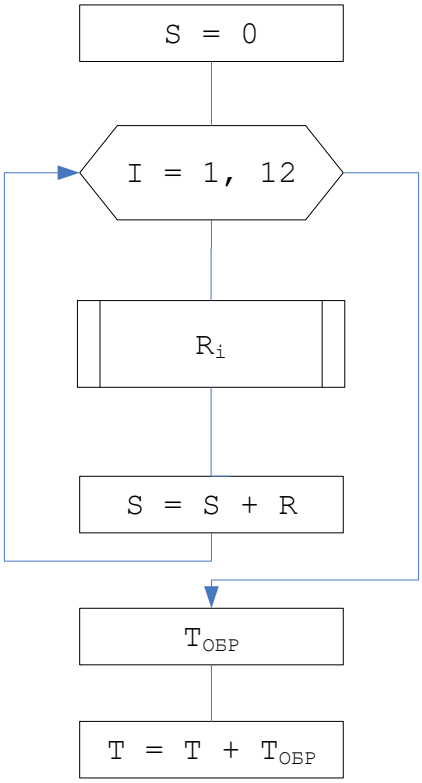
[23.10][Лекция 12]

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 σ_x, то длительность обработки i-ого требования:

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

Схема алгоритма имитатора.



R_i – случайное число с равномерным законом распределения
$T_{ОБР}$ – время обработки очередного сообщения
T – время освобождения ОА
XM – Мат ожидание для заданного закона обратки
DX – СКО (средне квадратичное отклонение) для заданного закона обратоки

$$T_{ОБР} = XM + (S - 6) * DX$$

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.

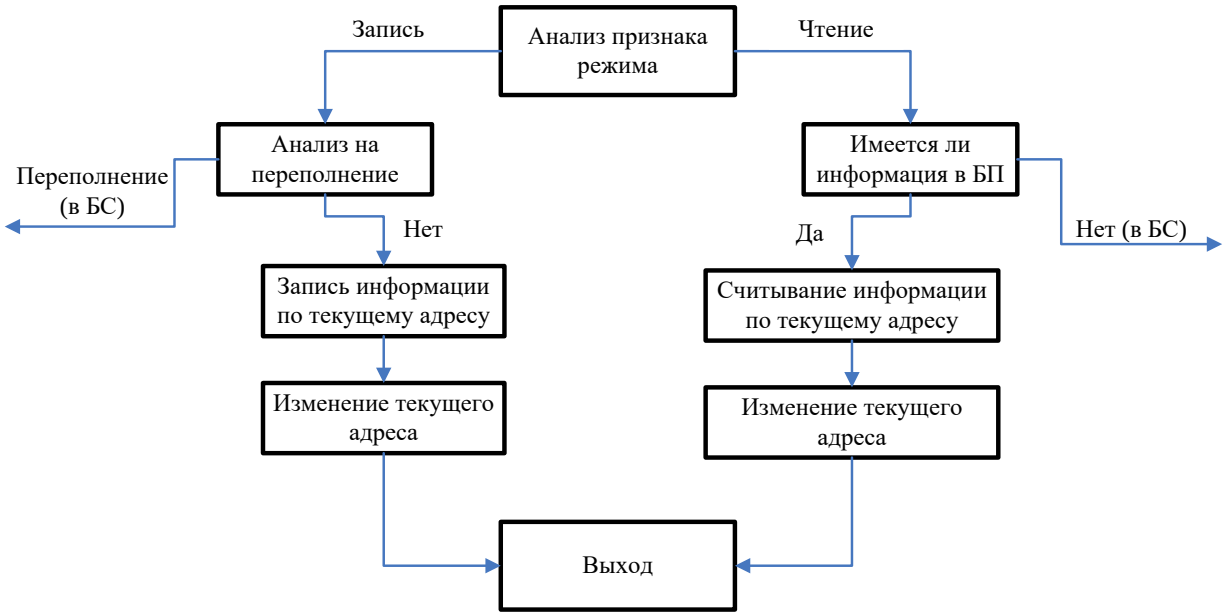
Память - относится к электромеханическому устройству, включающее в себя: среду для запоминания, устройство управления, (информация находится по адресу) база + смещение + [индекс].

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

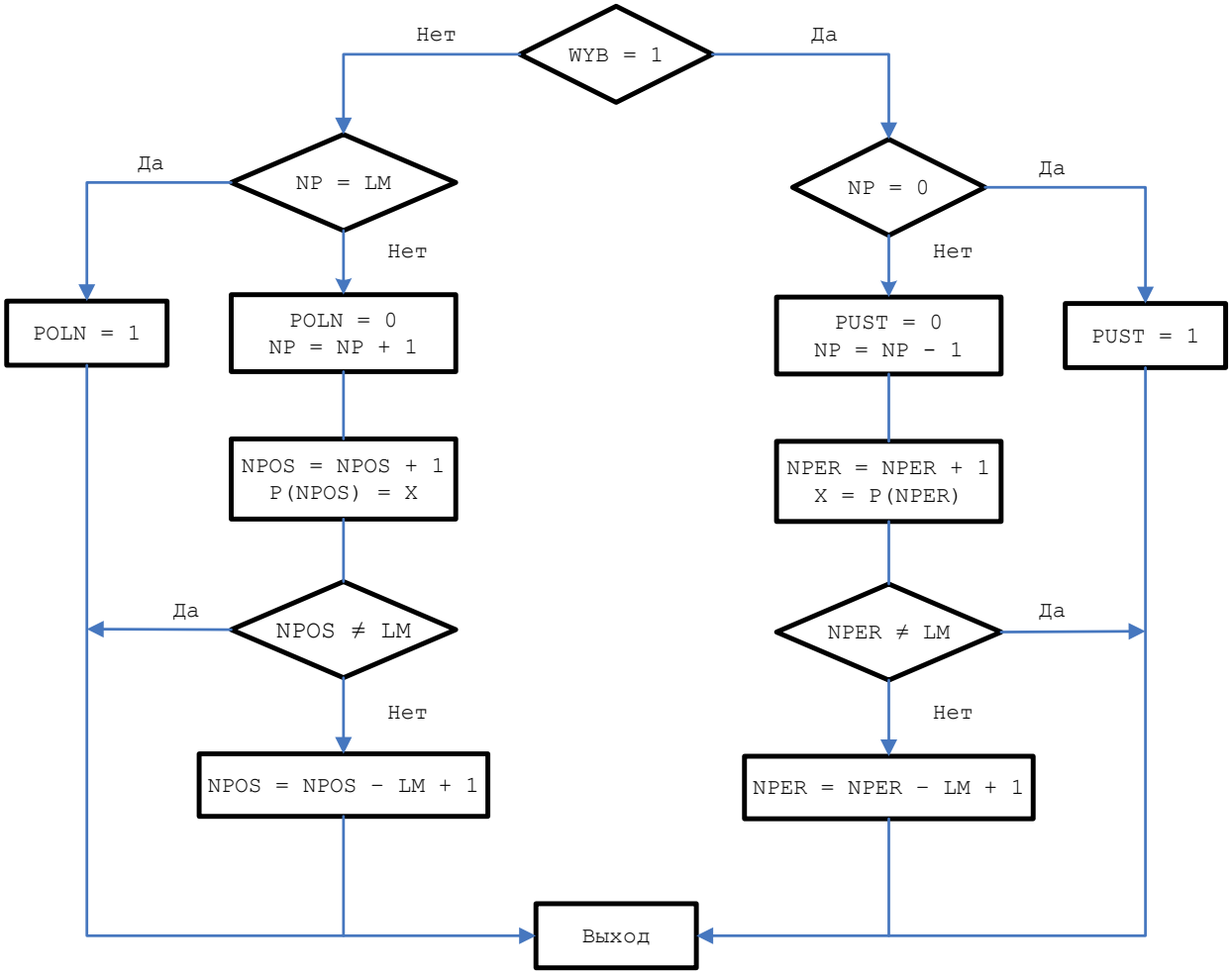
В блок статистики: ошибки записи, ошибки чтения.
Блок буферной памяти должен производить запись и считывание чисел, выдавать сигналы переполнения и отсутствия данных в любой момент времени располагать сведениями о количестве требований (заявок) в блоке. Сама запоминающая среда в простейшем случае имитируется 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 Δt**
2. **Событийный принцип**

Принцип Δt.

Принцип Δt заключается в последовательном анализе состояний всех блоков в момент t + Δt 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 Δt появляется

опасность пропуска отдельных событий в системе, что исключает возможность получения адекватных результатов при моделировании.

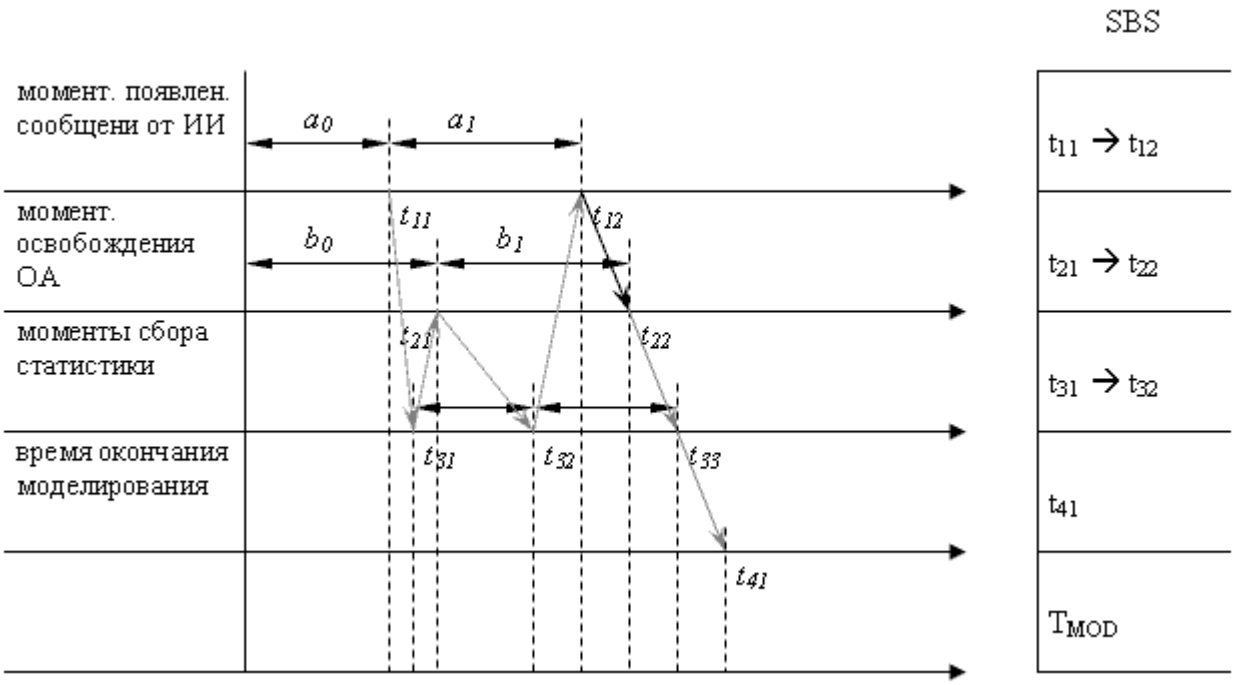
Достоинство: равномерная протяжка времени.

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.

Недостаток событийного принципа: (самостоятельная обработка)

Схема событийного принципа:

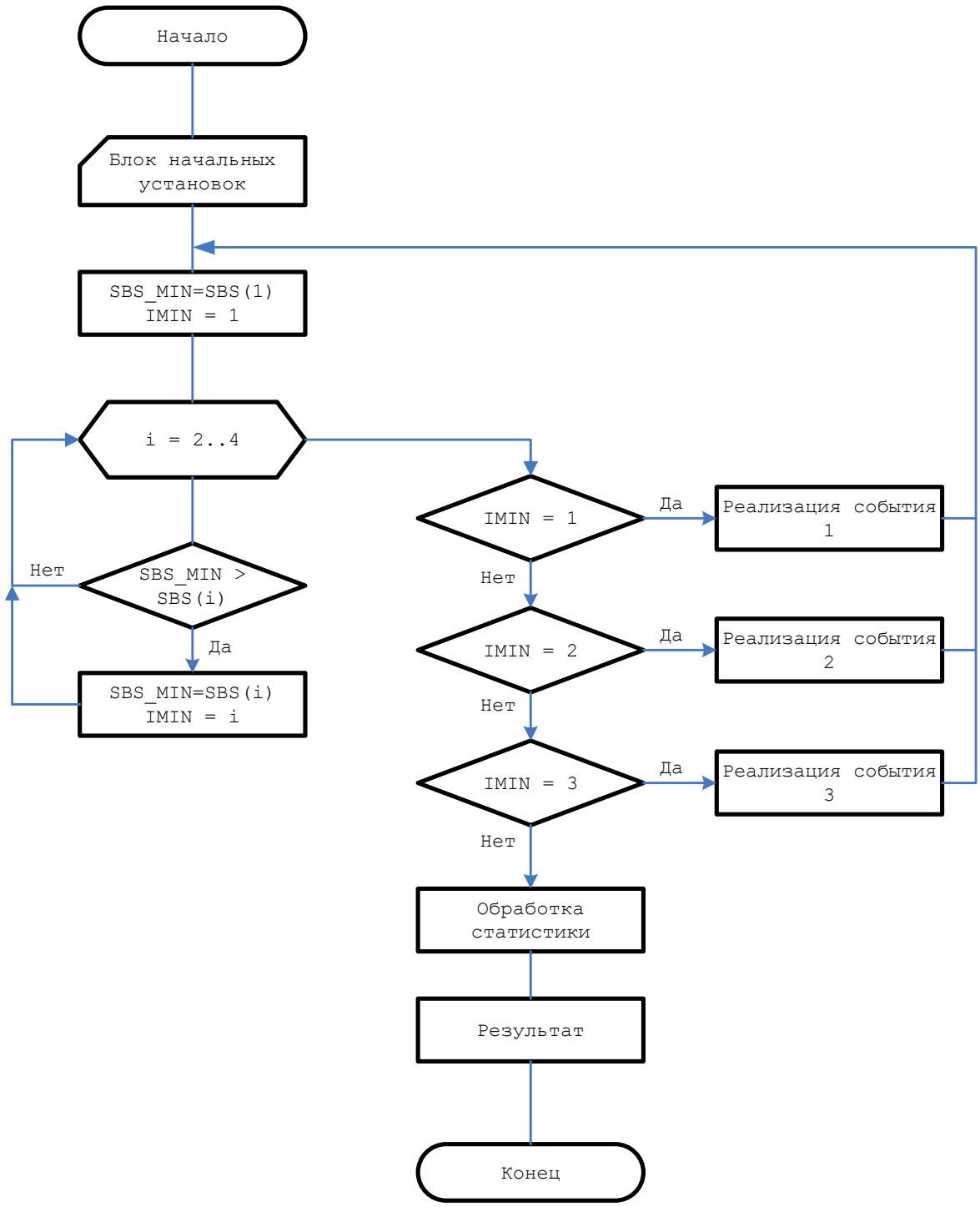


Первая ось: момент появления сообщений
Вторая ось: момент освобождения обслуживающего аппарата
Третья ось: момент сбора статистики (здесь абсолютно равные интервалы, мы сами определяем, когда собирать статистику)
Четвертая ось: время окончания моделирования
Пятая ось: текущее время

t_{11}, t_{12} – моменты появления сообщений на выходе генератора (источника информации)
b_1 – интервал времени обслуживания первого сообщения
$t_{3\,n}$ – 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:

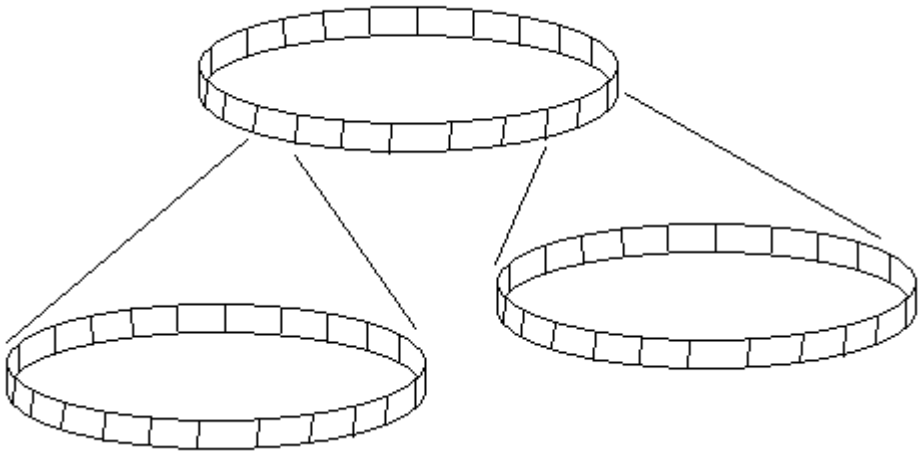
1. В SBS определяется минимальное числовое значение и его номер.
2. Реализуется событие, порождаемое блоком с соответствующим номером, т.е. модельное время = t_{11}. Далее реализуется событие с номером 1, связанное с появлением нового сообщения в ИИ. Реализация этого события заключается в том, что само сообщение записывается в память, а с помощью имитатора ИИ, вырабатывается момент появления следующего события t_{12}. Это время помещается в соответствующую ячейку SBS вместо t_{11}.
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][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* Δt, 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 происшедших 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.

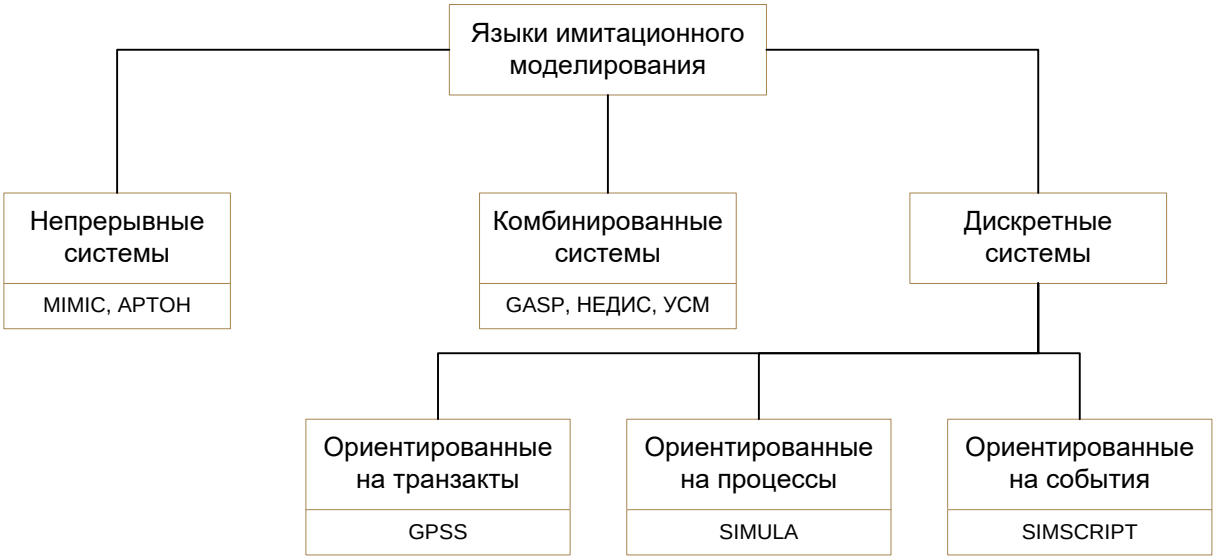
Для программирования модели могут использоваться следующие языки:
1. Универсальные алгоритмические языки высокого уровня.
2. Специализированные языки моделирования: языки, реализующие событийный подход, подход сканирования активностей, языки, реализующие процессно-ориентированный подход.
3. Проблемно-ориентированные 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 функциям. 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, единым 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 элементам* 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:

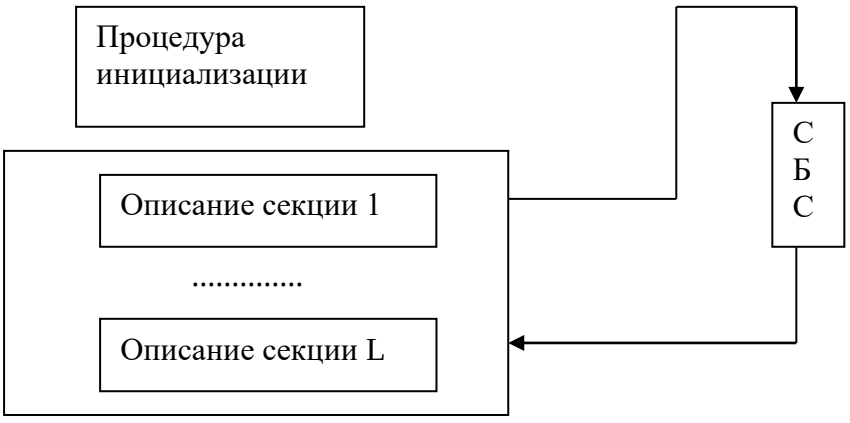
1. Первая задача сводится к тому, чтобы описать правила, описывающие виды процессов, происходящих в системе.
2. Вторая задача заключается в том, чтобы описать правила задания атрибутов или задать правила генерации этих значений. При этом система определяется на конкретном уровне детализации в терминах 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.

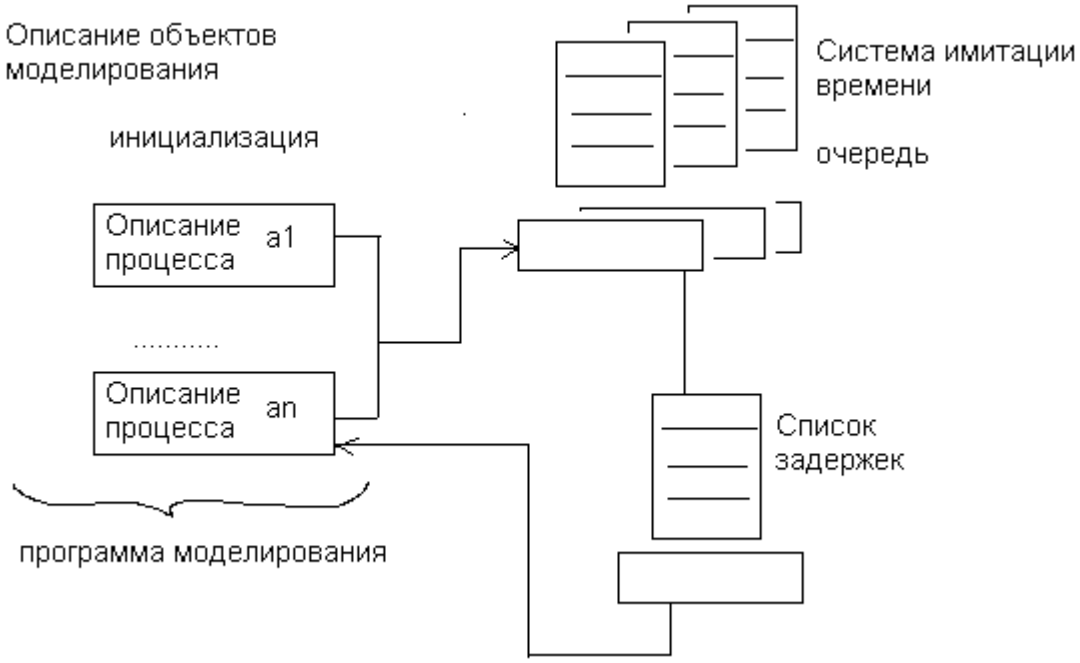


Языки, ориентированные на процессы.

Моделирующая программа организуется в виде набора описаний процесса. Каждый из которых описывает одни 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][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 автоматизирования 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 состояние 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 продукционные правила применимы для анализа статических объектов, т.е. правила описывают причинно следственные связи и не описывают динамику процесса. Поэтому при интеллектуально моделировании, т.е. для моделирования динамики системы, на основе знаний о протекающих процессах в РДО используют модифицированное продукционное правило.
Как оно выглядит:

ЕЛСИ <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: $R(t) = \{r_i: i = 1, \ldots, N(t)\}$
Где Ri – 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: $C_i(t) = \{C_{ij}(t): j = 1 \ldots M_i\}$
Где Cij – значение j-ого параметра i-ого ресурса
Mi – 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 отношении к системе причин, т.е. в систему пришел новый временный ресурс.

В отличие от регулярных событий, нерегулярные носят стахостический характрет.

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 временная 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 продукционной 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.

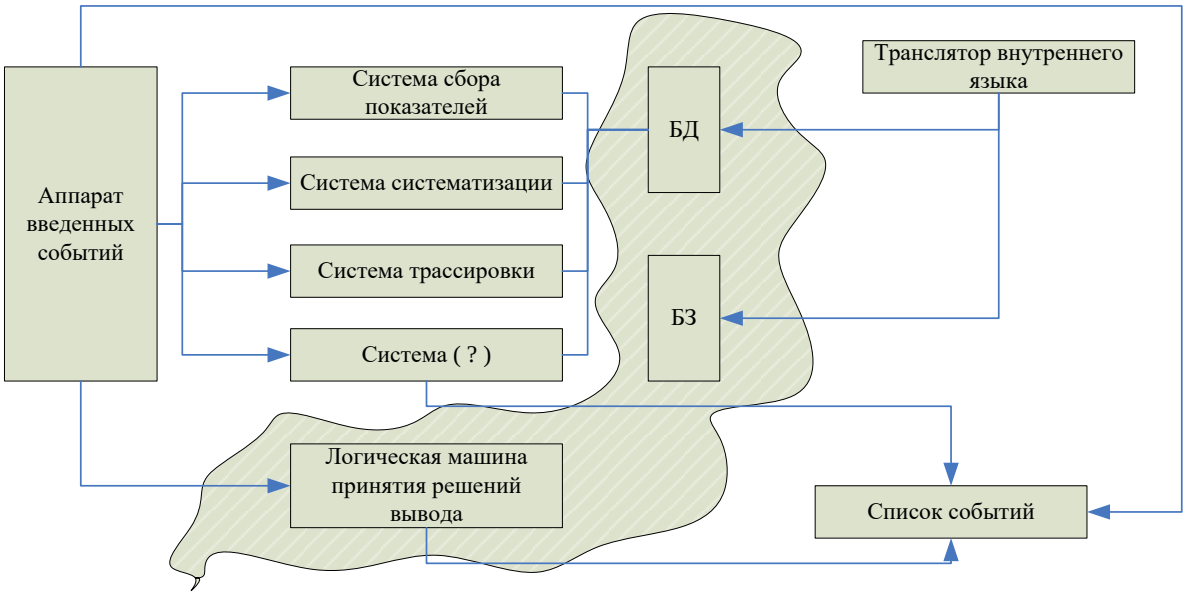
Лабораторная работа №5.
Создать концептуальную модель функционирования метро «Бауманская» в час пик. Структура включает в себя:
Равномерное прибытие поездов, как в противофазе, так и вместе.
3 эскалатора (Супербабушка может включить два верх, один вниз и наоборот)
Выходные двери от 4 до 1.

Вся статистика: сколько времени стоишь в очереди и т.п.
Определить, куда лучше втискиваться в очередь, т.е. где лучше всего находиться…

[10.11][Лекция 15]

Модель сложной дискретной системы в РДО представляет собой динамическую продукционную систему. *Базой данных* этой продукционной 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:



АВС – аппарат введения событий

Основными элементами РДО имитатора является: **динамическая продукционная 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 предназначенной 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:

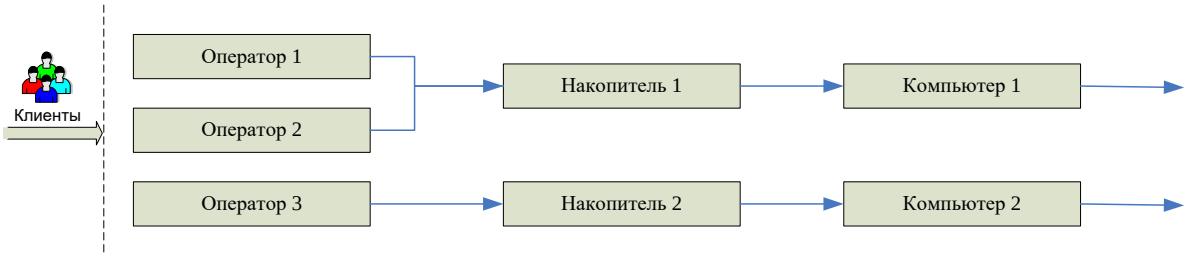
a. Реализует графический объектно-ориентированный интерфейс
b. Позволяет структурировать информацию по имитационным исследованиям
c. Автоматически поддерживает целостность информации
d. Режимы работы эксперта и пользователя

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.

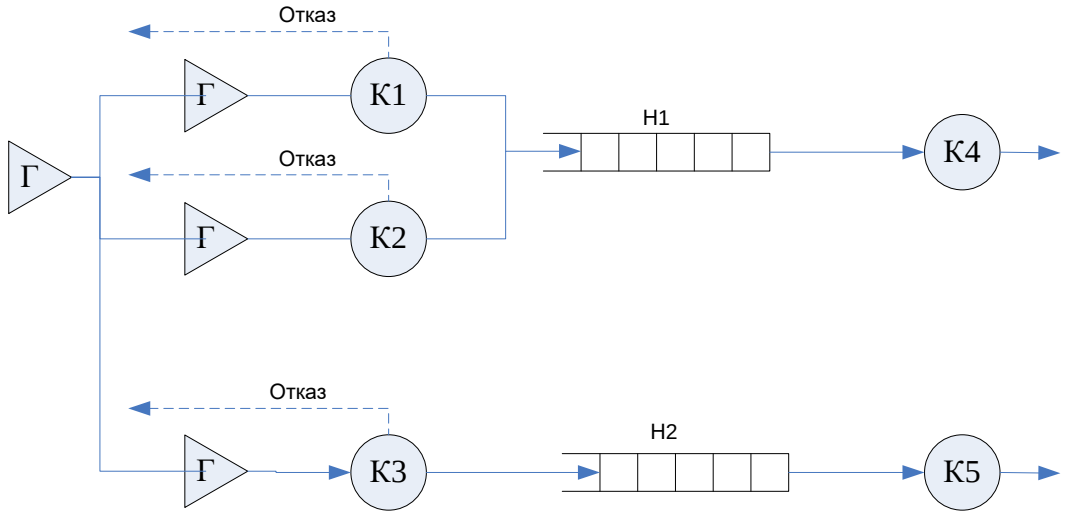
В информационный центр приходят клиенты через интервал времени 10±2 минуты. Если все три имеющихся оператора заняты, клиенту отказывают в обслуживании. Операторы имеют разную производительность и могут обеспечивать обслуживание среднего запроса пользователя за 20 +- 5; 40 +- 10; 40 +- 20. Клиенты стремятся 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.



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

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 значеняия 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 динамки: *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]|Лекция 16]

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

Язык General Purpose System Simulation (GPSS)

Язык GPSS – общецелевая 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. Изменятся 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.

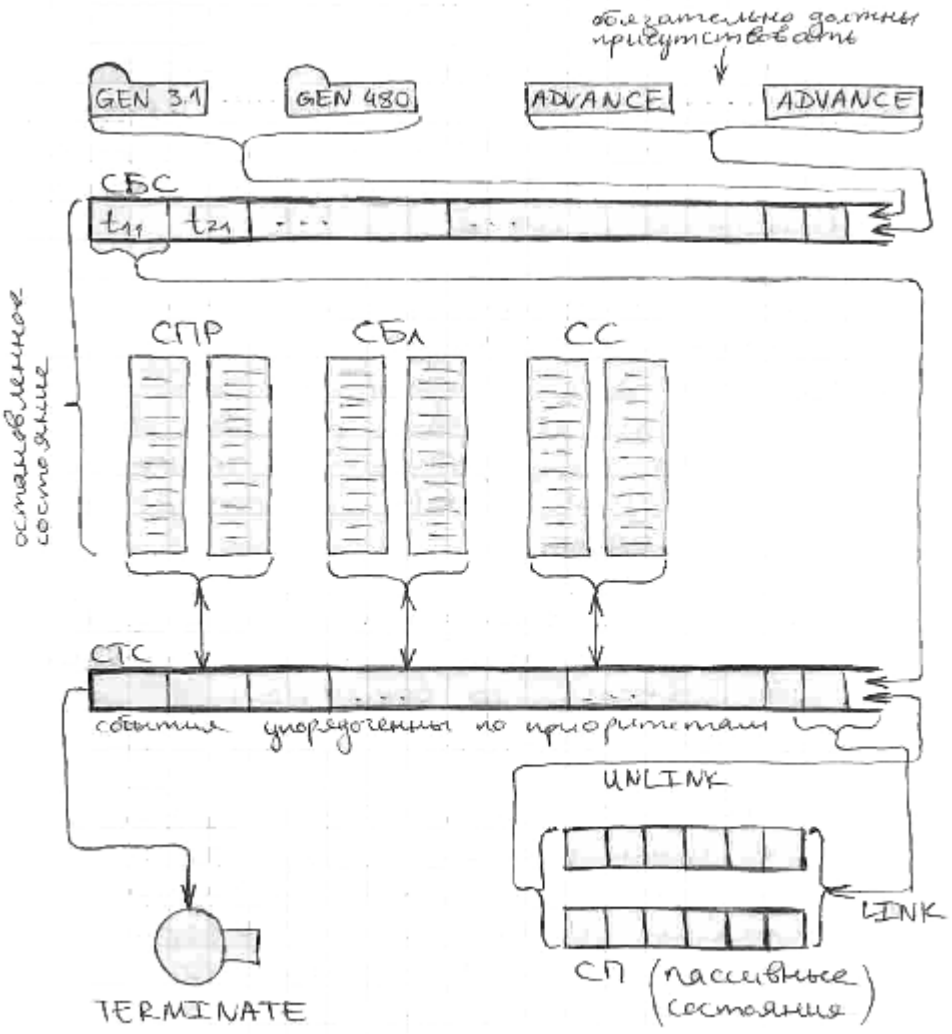
Оба счетчика меняют свое содержимое автоматически.

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 операндов, 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][Лекция 17]

В начальный момент (при выполнении оператора управляющего START, который начинает фазу интерпретации модели) управляющая программа обращается ко всем блокам GENEREATE 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 возращены 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 A, а модификатор в поле B. Если поле задержки постоянно, то поле B может быть пустым. А если нулевое, то и поле A 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 P – 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 операндов 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 A задается среднее время между поступлением отдельных транзактов. Как и в блоке ADVANCE, это поле может быть модифицировано с помощью модификатора находящегося в поле B (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 A.
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 A. Что приводит к ошибке. Избежать её можно поместив после блока GENERATE блок, не задерживающий транзакт.

В поле C записывается начальная задержка. Заданное в этом поле число без модификации определяет интервал времени до создания данным блоком первого транзакта. По отношению к A 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 E 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 парамтра формата «байт», 20 формата «..», 3 формата «..», 4 формата «..» (?)
```

Блок TERMINATE A
Удаляет транзакты из системы. Он используется для обозначения окончания пути транзакта.
Поле A 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 A указывается какой параметр поступившего транзакта должен быть изменен. Следующий непосредственно за номером транзакта символ указывает что нужно сделать с записанным в поле B целым числом. Прибавить (+), вычесть (-) или заменить этим числом.
Если в поле C указано значение, то оно интерпретируется как номер функции. Определяется значение этой функции, а результат используется для модификации целого числа, указанного в поле B. Произведение помещается в параметр, указанный в поле A.
B 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.
A – параметр транзакта, используемый для организации цикла. Оно может быть именем, числом, стандартным числовым атрибутом.
B – метка (имя) блока начального блока цикла.
Когда транзакт входит в блок LOOP транзакт указанный в операнде A уменьшается на 1. A затем проверяется его значение на равенство нулю. Если нулю не равно, то транзакт переходит в блок указанный в операнде B, если равен, то в следующий блок.

[24.11]|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 создасться.

GATHER A

Действие блока GATHER аналогично действию 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 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
```

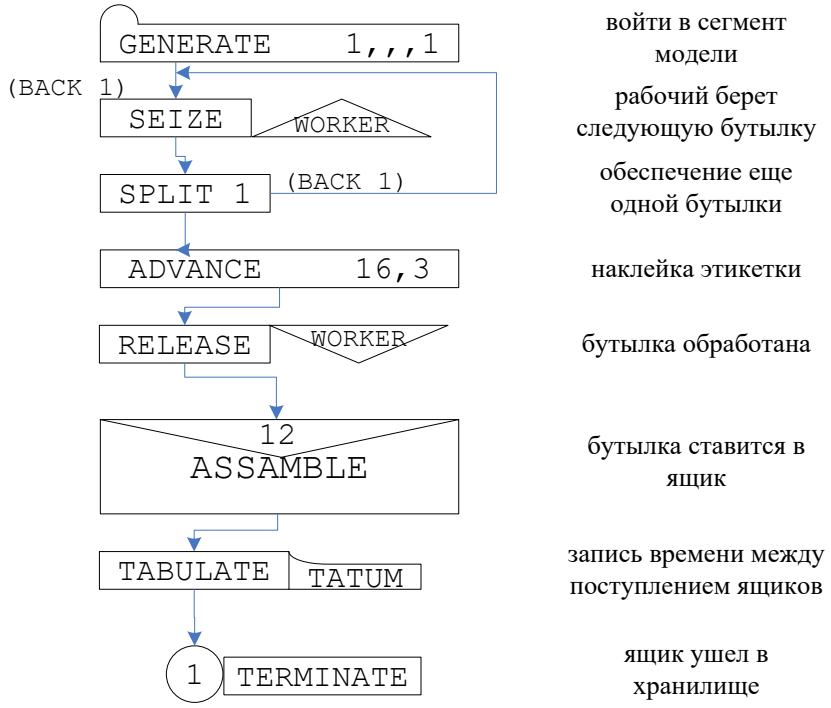
```
            GATHER      8

FINAL       ASSEMBLE    9
            SEIZE       Sborka
            ADVANCE     4,2
            RELEASE     Sborka

            TERMINATE   1
            START       80
```

Если вместо GATHER поставить ASSEBMLE. Что будет?

Задача. Рассмотрим часть производственного процесса на небольшом винном заводике в Колифорнии. Объем продукции не оправдает приобритение машины для автоматической наклейки этикеток на бутылке. Эту операцию проделывает один рабочий в ручную. 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 осущетсвляет необходимое условие принадлежности всех входящих транзактов одному ансамблю.

MATCH A

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

В поле A каждого блока MATCH 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. Иначе поступающий транзакт будет ожидать постуления транзакта того же ансамбля в сопряженный блок MATCH. После чего они оба будут пропущены в следующие за блоками MATCH блоки.

[27.11]|Лекция 19]

Задача. Построить программу моделирования для исследования обработки 500 деталей. Детали поступают с интервалом времени 300+-50 ед. Обработку производит 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 так же 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 B стоит мнемоническое значение PR), он подразумевает разрешение прерывания в зависимости от приоритетности транзактов.

Для последующей обработки прерванных транзактов существуют следующие возможности.

В поле C 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 A. 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 E 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 A – номер или диапазон номеров, переводимых в состояние недоступности
Поля B–H – для задания специальных режимов.

FAVAIL A
Делает доступным устройство с указанным номером или диапазоном в поле A. 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 предназначеные 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  столькло  сколько
содеражалось 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 пли 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 сотоянии 1.

В начале моделирования ключи могут быть установлены в состояние 1 с помощью команды INITIAL. А изменение состояние 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 O A,B
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 **A** определяет номер объекта аппаратной категории (устройства, памяти или ключа).
Если в поле B указано наименование или номер блока, то вместо отказа блок GATE будет посылать транзакт на указанный адрес. Следовательно, если поле B пусто, то блок работает в режиме отказа, - нет – в режиме перехода.

Существуют специальные логические атрибуты, описывающие состояние устройств, памятей, 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 O A,B,C

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 C. Если поле C 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 A и B. 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 вынуждены 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 TRANSEFER записана десятичная дробь, то производится случайный выбор между блоками В и С. Вероятность перехода в блок С и задает 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,,MET            // трехзначное число, записанное в параметре 5 транзакта, интерпретируется как вероятность (в долях от тысячи) того, что транзакт будет передаваться на метку MET, 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][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 A. 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 B. 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 A интерпретируется как номер очереди.

Поле B 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 B 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.

A – номер очереди

B – начальное значение

C – шаг таблицы

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

Предусмотрено несколько режимов табулирования, которые указываются в поле A. Знак минус за величиной, указанной в поле A, 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 A 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 A 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 пересылает 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 освободившейся 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 горизотальных отрезков. Для C-функции необходимо задать координаты всех точек, которые являются концами отрезков.

Действия необходимые для определения функций.

1. Присвоить функции имя. Имя либо числовое 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: $x_1<x_2< \ldots <x_n$.

Самостоятельно:
Написать следующие типы функций в GPSS World.
1. Моделирование Пуассоновского потока.
2. Моделирование Гипер-экспоненциального распределения.
3. K-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]|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 ординарности 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 ~R(0,1) преобразовать эти числа в соответствии с функцией, обратной к экспоненциальной. Т.е. мы получаем: $t \cdot \ln(...)$, где t – разыгранный интервал времени.

Разработчиками GPSS была выполнения апроксимция 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 A используют среднее значение интервала времени $t = \frac{1}{\lambda}$, где λ – интенсивность пуассоновского потока
2. В качестве операнда B 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 следуют 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

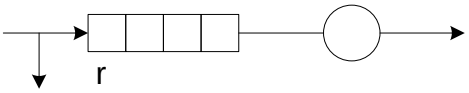
- A – закон распределения интервалов времени между поступлениями заявок. Наиболее часто используемое обозначение:
  M - экспоненциальное
  E – эрлангово
  H – гипер-экспоненциальное
  (?) – гамма
  D – детерминированное
  G – произвольное
- B – закон распределения времени обслуживания в приборах. Приняты те же обозначения, как и для интервалов между появлениями заявок.
- C – число обслуживающих приборов. Для одноканальной – 1, для многоканальной – *l*.
- D – число мест в очереди.
  n или r – 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.

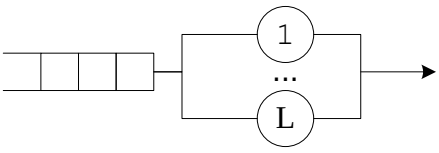
E/H/L/r/LIFO

G/G/1 - Одноканальные системы с ожиданием:

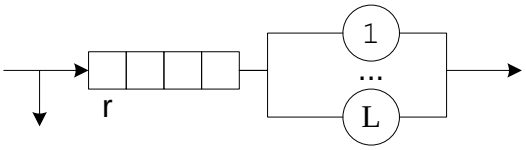
G/G/1/r- Одноканальная система с потерями:



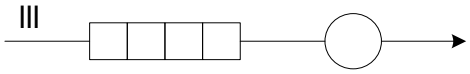
G/G/*l* - Многоканальная система с ожиданием



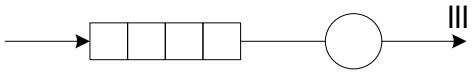
G/G/1/r – Многоканальная система с потерями



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:

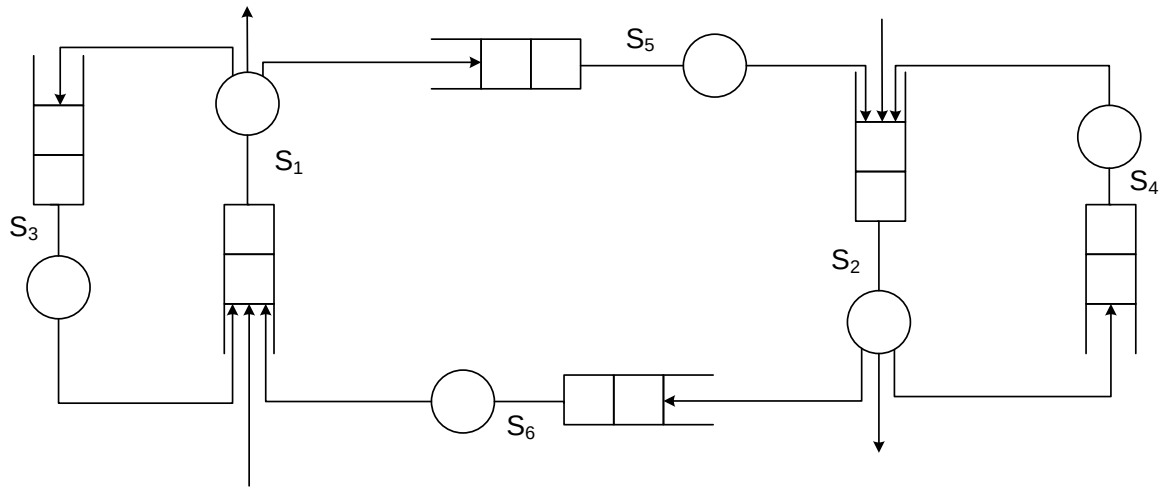
1. *Одноканальное СМО с ожиданием*. Представляет собой один обслуживающий прибор с бесконечной очередью. Является наиболее распространенной при исследовании СДС. Формализует функционирование практически любого числа узла вычислительной сети.
2. *Одноканальная СМО с потерями*. Один обслуживающий прибор 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 соответсвующего 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]|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 третей 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


            GENERATE    30,FN$XPDIS       ;30 умноженное на значение функции
            ASSIGN      1,FN$PROCESS
STAGE1      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 пересылается либо обратно в блок с меткой REVOKE1, REVOKE2, REVOKE3 с вероятностью 40%, либо с вероятность 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 применим 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.