Моделирование

Кабанов Александр Николаевич
к.ф.-м.н., доцент кафедры кибернетики

Язык моделирования GPSS

- GPSS (General Purpose Simulation System) — система имитационного моделирования общего назначения.
- Это язык моделирования, предназначенный для описания и исследования дискретных моделей систем массового обслуживания (СМО): системы обработки данных, системы транспорта и связи, технологические процессы, предприятия торговли, а также вычислительные системы и разного рода автоматизированные системы.
- Был разработан IBM в США.

GPSS World

- Программа GPSS World разработана компанией «Minuteman Software».
- Используемая для лабораторных работ GPSS World Student Version является свободно распространяемой и ее можно скачать с официального сайта разработчика:

www.minutemansoftware.com.

Транзакты

- Язык основан на схеме транзактов (сообщений).
- Транзакт — формальный объект, который «путешествует» по системе (перемещается от блока к блоку), встречая на пути всевозможные задержки, вызванные занятостью тех или иных единиц оборудования.
- Транзакты имеют прямую аналогию с заявками в СМО. В качестве транзакта может выступать программа обработки информации, телефонный вызов, покупатель в магазине, отказ системы при исследовании надежности и т. д.

Транзакты

- Каждый транзакт обладает совокупностью параметров, которые называются атрибутами транзакта. В процессе имитации атрибуты могут меняться в соответствии с логикой работы исследуемой системы.
- Каждое продвижение транзакта является событием в модели.
- Симулятор регистрирует время наступления каждого из известных на данный момент событий и выполняет их с нарастающей временной последовательностью.

Основные функции

- создание и уничтожение транзактов;
- изменение их атрибутов;
- задержка транзактов;
- изменение маршрутов транзактов в системе.

Программа

- Любая программа на GPSS связана с созданием транзактов, проведением их через последовательность блоков и уничтожением транзактов.
- Алфавит языка GPSS состоит из латинских букв от A до Z, цифр от 0 до 9 и следующих специальных символов: $, #, *, +, -, /, (,), ’, точка, запятая, пробел.

Стандартные числовые атрибуты (СЧА)

- В процессе моделирования язык GPSS автоматически регистрирует и корректирует определенную информацию различных объектов, используемых в модели.
- Доступ к этой информации осуществляется с помощью СЧА, которые однозначно определяют статус объектов модели.
- СЧА меняются в процессе имитации, изменить их может как симулятор, так и пользователь.

Стандартные числовые атрибуты (СЧА)

- Для указания конкретного объекта, по которому необходимо получить требуемую информацию, за именем СЧА должно следовать числовое или символьное имя этого объекта.
- Если используется символьное имя, то между СЧА и именем объекта ставится знак $.
- То есть указывать нужно <имя СЧА>i, либо <имя СЧА>$<имя объекта>, где i обозначает номер объекта.

Некоторые СЧА

- C1 – текущее значение условного времени.
- Pi – значение i-го параметра активного транзакта.
- X$<имя ячейки> или Xi – значение ячейки с указанным именем или указанным номером.
- M1 – время пребывания в модели активного транзакта.
- V$<имя переменной> – вычисленное значение переменной.
- RNi – случайное число в диапазоне 0-999, i – номер датчика случайных чисел.

Блоки и операторы

- Каждый блок языка записывается в отдельной строке и имеет следующую структуру: [метка] операция [операнды] [комментарии].
- Каждое поле отделяется друг от друга пробелами.
- Обязательным является только поле операции, остальные поля могут отсутствовать.

Поля блоков

- Метка является именем-идентификатором блока.
- Поле операндов может содержать от 1 до 7 подполей: A,B,C,D,E,F,G, содержимое которых отделяется друг от друга запятой.
- Для пропуска одного из подполей поля операндов ставится просто запятая, например: A„C.
- Поле комментария должно начинаться с символа «;».
- Комментарии, кроме поля комментариев, могут быть заданы отдельной строкой: любая строка, начинающаяся с символа «*» или «;», тоже будет комментарием.

Генерирование

- Блок GENERATE генерирует поток сообщений — транзактов, поступающих в систему.
- Временные интервалы между поступающими в систему транзактами определяются содержимым поля операндов.

Подполя GENERATE

- A — среднее время между поступлениями транзактов в систему (по умолчанию равно 1);
- B — модификатор времени;
- C — начальная задержка (время появления первого транзакта);
- D — общее число транзактов, которое должно быть сгенерировано этим блоком (по умолчанию — неограниченное число транзактов);
- E — приоритет транзакта, может принимать значения от 0 до 127. Приоритет возрастает в соответствии с номером (по умолчанию равен 0).

Генерирование

- Если в поле B может задано число, то для каждого временного интервала поступления транзактов длительность определяется как значение случайной величины, равномерно распределенной на интервале [A - B, A + B].
- Значение параметров A и B могут задаваться как константами, так и любым СЧА, за исключением СЧА параметра транзакта (эта величина в момент генерации транзакта еще не определена).

Генерирование

- Например, блок GENERATE 10,5 будет генерировать транзакты через интервалы времени, длительность каждого из которых выбирается случайно в пределах от 5 до 15.
- Каждое из этих значений будет выбираться с одинаковой вероятностью.
- Таким образом, блок генерирует случайный поток транзактов, в котором время между транзактами равномерно распределено в диапазоне [A - B, A + B] и имеет среднее значение A.

Генерирование

- В программе может быть несколько блоков GENERATE. Все эти блоки работают параллельно и начинают генерировать транзакты одновременно с момента начала моделирования.
- Необходимо помнить, что смысл единицы времени в языке GPSS (секунда, минута, час, день и т.д.) закладывает пользователь, поэтому при написании программы необходимо все операнды, связанные со временем, привести к единому масштабу.
- Время не может быть отрицательной величиной.

Уничтожение

- Блок уничтожения транзактов — TERMINATE. Обычно для простых программ это последний блок программы.
- Транзакты, попадающие в этот блок, уничтожаются и больше не участвуют в процессе моделирования.
- Никаких других действий этот блок не выполняет, если единственный возможный операнд A в блоке не задан.
- Если же операнд A задан, то его значение вычитается из счетчика числа завершений.
- Операнд A может принимать только положительное целочисленное значение.

Уничтожение

- Первоначальная величина счетчика устанавливается специальным управляющим блоком START и пишется в поле A этого блока.
- Когда в результате входа очередного транзакта в блок TERMINATE значение счетчика становится нулевым или отрицательным, симулятор прекращает моделирование и передает управление программе вывода, которая распечатывает накопленные симулятором данные о модели.

Уничтожение

- Например:

```
TERMINATE 1

START 100
```

- Здесь через программу модели пропускается 100 транзактов.
- В программе должен быть хотя бы один блок TERMINATE с заданным операндом A.
- Если в программе несколько блоков TERMINATE, то обычно операнд A задается только в одном блоке; чаще всего — в блоке, относящемся к имитатору интервала времени моделирования (таймеру).

Таймер

- Таймер служит для задания времени моделирования.
- Таймер взаимодействует только с блоком START и никак не связан с содержательной стороной остальных фрагментов модели.
- Пример:

```
GENERATE 480
TERMINATE 1
START 1
```

Задержка

- Для задержки транзактов на определенные интервалы времени предназначен блок ADVANCE.
- Обязательный операнд A задает время задержки транзакта.
- Необязательный операнд B является модификатором времени (разброс интервала времени относительно A – так же, как и в блоке GENERATE).

Задержка

- Любой транзакт входит в блок ADVANCE беспрепятственно. В нем транзакт задерживается на период времени, величина которого определяется операндами A и B. После этого транзакт направляется к следующему блоку.
- Например, ADVANCE 5,2.
- Транзакт будет задержан на случайное время, выбранное из диапазона [3,7].

Занятие устройства

- При входе транзакта в блок SEIZE выполняется операция занятия устройства, имя которого задается операндом A.
- Когда транзакт направляется из какого-нибудь блока в блок SEIZE, симулятор проверяет, свободно ли соответствующее устройство. Если оно не свободно, транзакт не может войти в этот блок. Он остается в предыдущем блоке до тех пор, пока устройство не освободится.
- Если же устройство свободно, то транзакт передвигается в блок SEIZE, занимает устройство и в тот же момент времени направляется к следующему за SEIZE блоку.

Освобождение устройства

- При входе транзакта в блок RELEASE происходит освобождение устройства, имя которого задается операндом A.
- Освободить устройство может только тот транзакт, который его занимает. Если транзакт попытается освободить устройство, занятое другим транзактом, симулятор прервет выполнение модели и выдаст сообщение об ошибке.
- В момент освобождения устройства должен быть решен вопрос о том, какой из задержанных транзактов перед блоком SEIZE имеет право первым занять устройство.

Освобождение устройства

- Когда транзакты задерживаются перед блоком SEIZE, они регистрируются симулятором в списке, где упорядочиваются по приоритетам: любой транзакт с более высоким приоритетом ставится впереди транзакта, имеющего более низкий приоритет.
- Если у двух транзактов одинаковые приоритеты, то они упорядочиваются между собой по времени прихода: впереди ставится транзакт, который раньше обратился к устройству.
- В момент освобождения устройства его занимает тот из задержанных транзактов, который находится в списке первым.

Освобождение устройства

- Транзакт может занимать любое число устройств.
- Освобождать занятые устройства транзакт может в любом порядке.

Пример

- Посетители приходят в кассу кинотеатра через каждые 10-30 секунд, до 30 секунд занимает знакомство с обстановкой, после чего они занимают очередь. Каждый посетитель приобретает у кассира билеты, на что уходит от 15 до 25 секунд. Построить модель работы кассы кинотеатра в течение четырех часов.

Пример

*приход посетителей

GENERATE 20,10

*знакомство с обстановкой

ADVANCE 15,15

* обращение к кассиру

SEIZE KASS

*покупка билета

ADVANCE 20,5

*освобождение кассира

RELEASE KASS

*уход посетителя

TERMINATE

GENERATE 14400 ;таймер

TERMINATE 1

START 1

Очередь

- Некоторые виды статистических данных накапливаются симулятором автоматически. Другие виды данных могут быть получены с помощью специальных блоков.
- При входе транзакта в блок QUEUE он ставится в очередь, имя которой задается операндом A.
- В начальный момент времени, когда очередь пуста, ее длина равна нулю. В момент входа транзакта в блок QUEUE ее длина увеличивается на величину, указанную в поле B. Если операнд B пуст, то длина очереди увеличивается на единицу.

Очередь

- При входе транзакта в блок DEPART длина очереди, имя которой задается операндом A, уменьшается на величину, указанную в операнде B.
- При использовании пустого поля B в блоках QUEUE и DEPART длина очереди в каждый момент времени соответствует текущему числу транзактов в этой очереди.
- Транзакты могут проходить любое число блоков QUEUE и DEPART с произвольными значениями полей A и B, чередующихся в любом порядке.

Очередь

- Необходимо помнить, что данные блоки не влияют на реальное образование очередей транзактов, а служат только для сбора статистических данных.
- Поэтому нужно следить за правильным расположением этих блоков, чтобы не получать отрицательные длины образуемых очередей.
- Симулятор только подсчитывает статистику по очередям и не считает за ошибку отрицательные длины очередей.

Пример

- GENERATE 20,10

ADVANCE 15,15

QUEUE OCH ; включение в очередь

SEIZE KASS ; обращение к кассиру

DEPART OCH ; выход из очереди

ADVANCE 20,5

RELEASE KASS

TERMINATE

GENERATE 14400 ; таймер

TERMINATE 1

START 1

Очередь

- В этой модели момент включения каждого транзакта в очередь OCH совпадает с моментом его обращения к блоку SEIZE, т. к. блок QUEUE выполняется в модельном времени мгновенно.
- Каждый транзакт находится в очереди до тех пор, пока не займет устройство KASS. Момент занятия устройства совпадает с моментом выхода транзакта из очереди.
- В данном случае очередь OCH имеет естественную интерпретацию как очередь посетителей к кассиру, а длина очереди интерпретируется как число посетителей в очереди.
- При наличии в модели очередей симулятор выдает статистику по очередям.

Пример

- В аэропорту производится регистрация пассажиров перед посадкой в самолет. На регистрацию подходят отдельные пассажиры через каждые 10-30 секунд либо туристические группы через каждые 40-80 сек. При этом туристические группы обслуживаются вне очереди. Время обслуживания подчинено экспоненциальному закону и равно в среднем для отдельных пассажиров — 15 секунд, для туристических групп — 25 секунд. Промоделировать работу отдела регистрации, изучив статистику по очереди за 2 ч.

Пример

GENERATE 20,10 ; приход отдельных пассажиров

QUEUE LIN ; включение в очередь

SEIZE REG

DEPART LIN ; выход из очереди

ADVANCE (EXPONENTIAL(1,0,15)) ; регистрация пассажира

RELEASE REG

TERMINATE ; уход пассажира

Пример

GENERATE 60,20„,1 ; приход туристической группы

QUEUE LIN

SEIZE REG

DEPART LIN

ADVANCE (EXPONENTIAL(1,0,25)) ; регистрация группы

RELEASE REG

TERMINATE ; уход группы

GENERATE 720 ; таймер

TERMINATE 1

START 1

Пример

- Отдельные пассажиры и туристические группы встают в одну и ту же очередь и обслуживаются одним регистратором. Внеочередность обслуживания групп в модели обеспечивается заданием приоритета для транзактов, имитирующих туристические группы.
- В программе три самостоятельных сегмента, каждый из которых начинается блоком GENERATE и заканчивается блоком TERMINATE. Они могут быть поставлены в программе в любом порядке. При этом процесс моделирования останется неизменным: все блоки GENERATE работают параллельно.

Отчет

- В результате выполнения модели на печать автоматически выводится информация о наличии транзактов в каждом блоке на момент завершения моделирования, а также информация обо всех устройствах, к которым производилось обращение в модели.

Общая информация отчета

- **START TIME.** Абсолютное системное время на начало рассматриваемого периода. START TIME устанавливается равным абсолютному системному времени, определенному командами RESET или CLEAR.
- **END TIME.** Абсолютное системное время на момент окончания моделирования.
- **BLOCKS.** Количество блоков в программе, исключая блоки описания.
- **FACILITIES.** Количество объектов «устройство» в программе.
- **STORAGES.** Количество объектов «память» в программе.

Имена и блоки в отчете

- **NAME.** Определенные пользователем имена, используемые в программе.
- **VALUE.** Числовое значение, присвоенное имени. Система присваивает значения именам, начиная с 10000. Исключение составляют имена блоков, им присваивается числовое значение в соответствии с порядковым номером в программе.
- **LABEL.** Имя блока, которое ему присвоено.
- **LOC.** Порядковый номер блока в программе.
- **BLOCK TYPE.** Имя блока-оператора в GPSS.

Блоки в отчете

- **ENTRY COUNT.** Количество транзактов, вошедших в данный блок с момента последнего RESET или CLEAR или с момента начала моделирования.
- **CURRENT COUNT.** Количество транзактов, находящихся в блоке на момент окончания моделирования.
- **RETRY.** Количество транзактов, ожидающих выполнения специфических условий, зависящих от состояния объекта данного блока.

Устройства в отчете

- **FACILITY.** Имя или номер объекта «устройство».
- **ENTRIES.** Количество раз, которое устройство было занято, с момента последнего RESET или CLEAR или с момента последнего запуска модели.
- **UTIL.** Средняя загрузка устройства за последний измеряемый период времени (доля системного времени, которое устройство было занято, от общего времени моделирования). Измеряемый период времени отсчитывается от начала моделирования или с момента последнего использования команды RESET или CLEAR.

Устройства в отчете

- **AVE. TIME.** Среднее время нахождения одного транзакта в устройстве.
- **AVAIL.** Состояние доступности устройства на конец моделирования. 1 означает, что устройство доступно, 0 — не доступно.
- **OWNER.** Номер транзакта, который занимает устройство. 0 означает, что устройство свободно.
- **PEND.** Количество транзактов, ожидающих в очереди, чтобы занять устройство через блок PREEMPT.

Устройства в отчете

- **INTER.** Количество транзактов, претендующих на устройство после прерывания.
- **RETRY.** Количество транзактов, ожидающих выполнения специфических условий, зависящих от состояния данного устройства.
- **DELAY.** Количество транзактов, ожидающих в очереди, чтобы занять устройство (включает транзакты, которые пытаются занять устройства через блоки SEIZE и PREEMPT).

Изменение маршрута

- Блок TRANSFER позволяет осуществлять безусловные, статистические и условные переходы. Тип перехода определяется в операнде A, направление перехода — в операндах B, C и D.
- В режиме безусловного перехода операнд A в блоке пуст. Все транзакты переходят к блоку, указанному в поле B. Например:

```
TRANSFER ,NEXT
```

- Если блок, к которому направляется транзакт, в текущий момент системного времени не может его принять, то транзакт остается в блоке TRANSFER и повторяет попытку перехода при каждом пересчете системного времени

Изменение маршрута

- Если в поле A блока TRANSFER записана десятичная дробь, начинающаяся точкой, то блок работает в режиме статистического перехода.
- Здесь десятичная дробь определяет вероятность перехода транзакта к блоку, имя которого указывается в поле C. При этом поле B пустое.
- С вероятностью 1 — <A> транзакт переходит к блоку, следующему за блоком TRANSFER.
- Если оба блока заняты, то транзакт остается в блоке TRANSFER и повторяет попытку перехода к выбранному ранее блоку при каждом изменении системного времени.

Изменение маршрута

- Если требуется разбить поток заявок более чем на два, необходимо внимательно рассматривать вероятности статистических переходов в модели.
- В этом случае, если вы используете несколько блоков статистического перехода TRANSFER, для всех блоков, кроме первого, необходимо пересчитывать вероятности переходов с учетом того, что часть заявок уже была перенаправлена к другому блоку.

Пример

- Известно, что поток покупателей в магазине разбивается на три потока (три отдела в магазине) следующим образом: 50% покупателей направляется в 1-й отдел, 20% — во 2-й и 30% — в 3-й.
- Тогда перенаправление транзактов будет происходить следующим образом:

TRANSFER .5„OTD1

TRANSFER .4„OTD2 ; 20% от оставшихся 50% — это 40%

TRANSFER ,OTD3 ; все остальные — в 3-й отдел

Пример

- Ситуация значительно усложняется, если проценты распределения заявок равны, например 17, 47 и 36. Чтобы упростить моделирование такой ситуации, можно построить дискретную функцию типа D, значением которой будут имена блоков, к которым должны направляться потоки заявок.

TRAN FUNCTION RN1,D3

.17,OTD1/.64,OTD2/1,OTD3

А затем можно обращаться к функции в блоке безусловного перехода

TRANSFER ,FN$TRAN

Условный переход

- Режим условного перехода определяется мнемокодом, заданным в поле A.
- Если в поле A определено значение BOTH, то транзакт первоначально направляется к блоку, имя которого определено в поле B. Если переход невозможен (например, занято устройство), то делается попытка перейти к блоку, чье имя определено в поле C. Если оба блока заняты, то транзакт остается в блоке TRANSFER и повторяет попытку перехода при каждом изменении системного времени.

Условный переход

- Если в поле A определено значение ALL, то поля B и C содержат имена блоков, а поле D содержит целое число.
- Транзакт последовательно пытается войти в блоки, отстоящие друг от друга на расстояние D, начиная с блока B и заканчивая блоком C до первой успешной попытки. Если ни один из блоков не может принять транзакт, то он остается в блоке TRANSFER и повторяет попытку перехода при каждом изменении системного времени.
- Значение поля D должно задаваться таким образом, чтобы выполнялось условие №C = №B + M*D, где M — любое целое число. Если поле D не задано, то транзакт пытается последовательно войти в каждый блок между B и C.

Условный переход

- Если в поле A определено значение PICK, то поля B и C содержат имена блоков, а транзакт направляется в любой блок между блоками B и C, выбранный случайным образом.

Проверка состояния

- Еще одним блоком, изменяющим маршрут транзакта, является блок GATE.
- Он позволяет изменять путь транзакта в зависимости от состояния моделируемого оборудования и имеет следующую структуру: GATE O A,B.
- В поле O задается проверяемое состояние оборудования в виде мнемокода. В поле A задается имя проверяемой единицы оборудования, в поле B — имя блока, к которому направляется транзакт, если проверяемое условие ложно.

Проверка состояния

- Если поле B пусто, то блок работает в режиме отказа.
- В этом режиме транзакт задерживается в блоке GATE до тех пор, пока не выполнится условие состояния проверяемого объекта.
- Как только это произойдет, транзакт направляется к следующему за GATE блоку.

Проверка состояния

- Если поле B не пусто, то блок работает в режиме условного перехода.
- Если проверяемый в этом режиме объект не находится в требуемом состоянии, транзакт направляется к блоку, указанному в поле B.
- В противном случае транзакт направляется к следующему за GATE блоку.

Мнемокоды

- SE – память пуста.
- SNE – память не пуста.
- SF – память заполнена.
- SNF – память не заполнена.
- U – устройство занято.
- NU – устройство свободно.
- LS – логический переключатель включен.
- LR – логический переключатель выключен

Пример

- Например, в блоке

GATE SF STR

транзакт будет задержан до тех пор, пока память с именем STR не будет полной.

Логические условия

- Блок TEST изменяет маршрут транзакта в зависимости от выполнения разнообразных логических условий, определенных на множестве СЧА. Блок имеет следующую структуру: TEST O A,B,C
- В поле O указывается мнемоника отношения: L — меньше, LE — меньше или равно, E — равно, NE — не равно, G — больше, GE — больше или равно.
- В полях A и B указываются левое и правое значения условия, соответственно. В поле C указывается имя блока, к которому направляется транзакт, если проверяемое условие ложно. Если проверяемое условие истинно, то транзакт переходит к следующему за TEST блоку.

СЧА для работы с очередью

- Q$<имя> – текущая длина очереди.
- QA$<имя> – средняя длина очереди.
- QC$<имя> – число транзактов, вошедших в очередь с начала моделирования.
- QM$<имя> – максимальная длина очереди.
- QT$<имя> – среднее время нахождения транзакта в очереди.

Пример

- Например, при входе транзакта в блок

TEST G Q$OCH,5,OTD1

проверяется длина очереди OCH.

- Если длина очереди больше пяти, то транзакт направляется к следующему за TEST блоку, иначе транзакт переходит к блоку с именем OTD1.

Пример

- В магазине находится два отдела: продовольственный и промтоварный. Около 30% приходящих в магазин покупателей направляются в промтоварный отдел, остальные — в продовольственный. Причем если очередь в промтоварном отделе больше двух человек, а в продовольственном — больше пяти, то покупатели уходят из магазина, не дожидаясь обслуживания. Время прихода и обслуживания покупателей распределено экспоненциально. Среднее значение времени прихода равно соответственно 20 сек, времени обслуживания в продовольственном отделе — 30 сек и в промтоварном — 35 сек. Составить модель, имитирующая работу магазина за 8 ч.

Пример

GENERATE (EXPONENTIAL(1,0,20)) ;приход покупателей

TRANSFER .3„PROM ;выбор покупателем отдела

;работа продовольственного отдела

PROD TEST LE Q$LIN1,5,FIN ;если очередь больше 5 чел. — уход покупателя

QUEUE LIN1 ;поставить в очередь в продовольственный отдел

SEIZE PROD1 ;занять продавца

DEPART LIN1 ;покинуть очередь в продовольственный отдел

ADVANCE (EXPONENTIAL(1,0,30)) ;обслуживание покупателя

Пример

```
RELEASE PROD1 ;освободить продавца
TERMINATE ;уход покупателя
;работа промтоварного отдела
PROM TEST LE Q$LIN1,2,FIN
QUEUE LIN2
SEIZE PROD2
DEPART LIN2
ADVANCE (EXPONENTIAL(1,0,35))
RELEASE PROD2
```

Пример

```
FIN TERMINATE
; таймер
GENERATE 2880
TERMINATE 1
START 1
```

Работа с памятью

- Объект «память» призван имитировать разного рода накопители, используемые в исследуемых системах, в которых может одновременно находиться несколько транзактов.
- Для каждой применяемой памяти пользователь должен указать её ёмкость — максимальное количество транзактов, которые могут одновременно в ней находиться.
- Для указания емкости используется оператор описания памяти STORAGE. Этот блок помещается до первого блока GENERATE.
- Поле метки содержит имя памяти, а операнд A указывает емкость памяти. Например, STR STORAGE 10.

Занятие памяти

- Блок ENTER занимает память: ENTER A,B.
- В поле A блока указывается имя памяти, в которую помещается транзакт, в поле B – число единиц памяти, занимаемых транзактом при входе.
- Когда транзакт входит в блок ENTER, определяется число свободных единиц памяти.
- Если значение операнда B не превышает числа свободных единиц памяти, то число занятых единиц увеличивается на значение операнда B. В этом случае транзакт входит в блок ENTER без задержки.

Занятие памяти

- Если же значение операнда B превышает число свободных единиц памяти, то транзакт задерживается перед входом в блок ENTER.
- Задержанные при обращении к памяти транзакты упорядочиваются по приоритету.
- Если поле B в блоке ENTER пустое, то число занимаемых единиц памяти принимается равным единице.

Занятие памяти

- Если транзакт задержан перед входом в блок ENTER, но для второго транзакта, приходящего после, свободной емкости памяти достаточно, то второй транзакт войдет в блок без задержки.
- Для использования блока ENTER память должна быть обязательно описана ранее командой STORAGE.

Освобождение памяти

- Блок LEAVE освобождает память: LEAVE A,B.
- В поле A указывается имя освобождаемой памяти, в поле B — число освобождаемых единиц.
- В случае пустого поля B число освобождаемых единиц памяти принимается равным единице.
- При входе транзакта в блок LEAVE количество занятых единиц памяти, указанной в поле A, уменьшается на значение операнда B.
- Перед входом в блок транзакты не задерживаются.

Освобождение памяти

- Транзакт не должен освобождать большее число единиц памяти, чем их всего занято.
- Если же транзакт пытается это сделать, то симулятор выдает на печать сообщение об ошибке и прекращает выполнение модели.
- Транзакт не обязан освобождать такое же число единиц памяти, какое занимал.
- Он может также освобождать память, которую не занимал.

Освобождение памяти

- В тот момент модельного времени, когда транзакт освобождает память, симулятор просматривает список задержанных у памяти транзактов, если они есть.
- Для каждого очередного транзакта проверяется, может ли он теперь быть обслужен памятью.
- Если такая возможность есть, то симулятор перемещает этот транзакт в блок ENTER, и в результате число занятых единиц памяти соответствующим образом увеличивается.

Освобождение памяти

- Транзакт имеет право занимать и освобождать любое количество памятей, при этом операции занятия и освобождения могут чередоваться в произвольном порядке.

Пример

- Автомобили подъезжают к бензозаправочной станции в среднем каждые 2-6 минут. На станции есть две бензоколонки, каждая из которых используется в среднем 4-6 минут. Автостоянка при станции рассчитана на 4 автомобиля. Если подъехавший автомобиль застает обе бензоколонки занятыми, то он встает в очередь на автостоянку. Если же все места на автостоянке заняты, то автомобиль проезжает мимо. Промоделировать работу станции за 12 часов.

Пример

STO STORAGE 4 ; места под автостоянку

COL STORAGE 2 ; бензоколонки

GENERATE 4,2 ; приезд автомобиля

GATE SNF STO,FIN ; если места заняты — проезжает

ENTER STO ; занять место на автостоянке

ENTER COL ; занять бензоколонку

LEAVE STO ; освободить автостоянку

Пример

ADVANCE 5,1 ; заправиться

LEAVE COL ; освободить бензоколонку

FIN TERMINATE ; покинуть станцию

; таймер

GENERATE 720

TERMINATE 1

START 1

Переменная

- Для использования переменной ее необходимо описать оператором VARIABLE (арифметическая переменная), FVARIABLE (арифметическая переменная с плавающей точкой) или BVARIABLE (булевская переменная).
- В поле метки оператора записывается имя переменной, а в операнде A — арифметическое выражение, составляемое из СЧА, знаков арифметических операций и круглых скобок.

Переменная

- Используются следующие арифметические операции: +, –, # (умножение), /, @ (остаток от деления), \ (целое от деления), ^ (возведение в степень).
- Деление на ноль не считается ошибкой, и результатом такого деления является ноль. Остаток от деления на ноль также считается равным нулю.
- При использовании переменной в программе указывается СЧА переменной: V$<имя переменной>.

Переменная

- Переменная является единственным объектом языка, по которому по окончанию моделирования в отчете не выдается никакой информации.
- Поэтому, если по окончанию моделирования необходимо проанализировать значение переменной, то можно присвоить ее значение ячейке, значение которой выводится в результирующем отчете.

Ячейки

- Ячейки служат для хранения некоторых постоянных или изменяющихся значений данных программы. Ячейка может обозначаться как именем, так и числом.
- Для работы с ячейками используется блок SAVEVALUE. В поле A этого блока указывается номер или имя ячейки, сохраняющей значение, и вид изменения этого значения («+» — накопление, «–» — уменьшение).
- В поле B содержится либо СЧА, либо число, которое добавляется или вычитается, или заменяет содержимое ячейки.

Ячейки

- Например, SAVEVALUE 10+,1 означает, что при поступлении транзакта в блок к содержимому 10-й ячейки прибавляется единица.
- SAVEVALUE FRT,V$VAR1 означает, что при поступлении транзакта в этот блок в ячейку с именем FRT записывается значение переменной VAR1.
- При необходимости обращения к ячейке указывается СЧА ячейки: X$<имя ячейки> (если ячейка задана именем) или XN (если ячейка задана номером N).

Ячейки

- Перед началом имитации содержимое всех используемых в программе ячеек устанавливается в 0.
- Если же требуется задать значение какой-либо из ячеек до начала моделирования, то для этого используется блок INITIAL, в поле A которого задается СЧА ячейки, а в поле B — присваиваемое значение.
- Например, INITIAL X$UCH1,10 — присвоить ячейке с именем UCH1 значение 10.
- Этот оператор должен помещаться до первого блока GENERATE.

Пример

- Производство деталей включает длительный процесс сборки, заканчивающийся коротким периодом обжига в печи. Поскольку содержание печи обходится дорого, несколько сборщиков используют одну печь, в которой одновременно можно обжигать только одну деталь. Сборщик не может начать новую сборку, пока не вытащит из печи предыдущую деталь. Время сборки детали равно 25-35 минут. Время обжига равно 6-10 минут. Зарплата сборщика составляет 60 рублей в час, стоимость эксплуатации печи — 3200 рублей за 8 часов. Цена материала, идущего на изготовление одного изделия, равна 100 рублям, стоимость готового изделия — 380 рублей.

Пример

- Необходимо определить оптимальное количество сборщиков, исходя из максимизации прибыли за неделю (8-часовой рабочий день без выходных).
- В качестве транзактов в модели будем рассматривать самих сборщиков, подразумевая, что в каждый момент времени один сборщик производит одну деталь.

Пример

SBOR VARIABLE N ; при прогоне модели вместо N ставим конкретное целое число

ZATR VARIABLE 3200#7+V$SBOR#60#56 ; затраты на содержание печи и зарплату сборщиков

PRIB VARIABLE X$IZD-V$ZATR ; результирующая прибыль

GENERATE „,V$SBOR

PROD ADVANCE 30,5 ; процесс сборки изделия

SEIZE PECH

ADVANCE 8,2

Пример

```
RELEASE PECH
SAVEVALUE IZD+,280
TRANSFER ,PROD
GENERATE (56#60)
SAVEVALUE REZULT,V$PRIB
TERMINATE 1
START 1
```

Матрицы

- Матрицы служат для хранения некоторых постоянных или изменяющихся значений данных программы в виде массивов.
- Чтобы использовать в программе матрицу, необходимо сначала ее описать оператором MATRIX.
- В поле метки оператора записывается имя матрицы. Поле A в операторе не используется, поля B и C содержат числовые значения, определяющие количество строк и количество столбцов в матрице, соответственно.

Матрицы

- Начальные значения всех элементов матрицы равны нулю.
- Если необходимо присвоить всем элементам матрицы одинаковые значения, отличные от нуля, используется оператор INITIAL, в поле A которого задается имя матрицы, в поле B — присваиваемое значение.

Матрицы

- Для изменения значения отдельного элемента матрицы используется блок MSAVEVALUE. В поле A блока указывается имя матрицы, после которого может быть указан «+», «–» или ничего. Поля B и C служат для выбора конкретного элемента матрицы и содержат номер строки и номер столбца, соответственно. В поле D указывается значение, которое должно быть добавлено, вычтено или присвоено элементу матрицы.
- Если после имени матрицы не стоит никакого знака, то значение D присваивается элементу. Если после имени матрицы стоит знак «+» или «–», то указанное значение добавляется или вычитается из текущего значения элемента, соответственно.

Матрицы

- При необходимости обращения к элементу матрицы указывается СЧА элемента: MX$<имя матрицы>(I, J) (если матрица задана именем) или XN(I, J) (если матрица задана номером N). Здесь I означает номер строки, J — номер столбца.

Приоритет

- Каждый транзакт может иметь свой приоритет — от 0 до 127. Чем больше номер, тем больше приоритет. Предпочтение в системе отдается транзактам с большим приоритетом, ранее поступившим.
- Для изменения приоритета транзакта в процессе его путешествия по системе используется блок PRIORITY. Поле этого блока определяет значение присваиваемого приоритета.
- Например, при прохождении через блок PRIORITY 3 транзакту будет присвоен приоритет 3.

Параметры

- Каждый транзакт может иметь до 100 параметров (атрибутов). Значения параметрам присваиваются с помощью блока ASSIGN.
- В поле A этого блока указывается номер или имя параметра и вид его изменения, в поле B определяется записываемое в параметр значение, в поле C задается при необходимости модификатор значения B в виде имени функции, значение которой умножается на B.
- Приписывая к номеру параметра в поле A символ + или –, можно обеспечить не запись значения поля B в параметр, а добавление или вычитание этого значения из значения параметра.

Параметры

- Например, ASSIGN 1,10 – занести 10 в P1.
- ASSIGN 2+,V$VAR1,EXP – добавить в P2 значение V$VAR1*FN$EXP.
- ASSIGN TRE–,S$STR – вычесть из P$TRE значение текущего содержимого памяти STR.

Параметры

- Используя блок ASSIGN, можно организовывать циклы в программе.
- Например, если необходимо прогнать транзакт 10 раз через блок ADVANCE, это можно осуществить следующим образом:

ASSIGN 1,10 ; занести 10 в P1 транзакта

PROD ADVANCE 52

ASSIGN 1-,1 ; вычесть 1 из P1

TEST E P1,0,PROD ; продолжать цикл пока счетчик не обнулится

Пример

- В магазине электротоваров работают два консультанта. Посетители заходят в магазин в среднем каждые 2 минуты. Покупатели осматривают товар в среднем в течение 10 минут, после чего примерно 70% из них обращаются к консультанту за помощью (время на консультацию составляет в среднем 5 минут). После осмотра товара и получения консультации примерно 40% посетителей уходят без покупки. Остальные покупатели с выбранным товаром направляются к кассе. Кассир обслуживает клиентов в среднем 3 минуты. Стоимость покупки распределена равномерно на интервале от 500 до 20000 рублей.

Пример

- Известно, что примерно у 3-х процентов покупателей есть карточка со скидкой в 10% , а у 12-ти процентов покупателей есть карточка со скидкой в 5%. Необходимо оценить прибыль магазина за 10-часовой рабочий день.
- Наличие карточки со скидкой у покупателя мы опишем функцией SKIDKA и будем записывать ее значение в 1-й параметр транзакта. Стоимость производимой покупки опишем с помощью функции POKUP, прибыль магазина опишем переменной SUM. Консультанты в модели будут представлены памятью CONS, кассир — устройством KAS.

Пример

CONS STORAGE 2

POKUP FUNCTION RN1,C2

0,500/1,20000

SKIDKA FUNCTION RN1,D3

.03,0.10/.15,0.05/1,0

SUM VARIABLE FN$POKUP#(1-P1)

Пример

GENERATE (EXPONENTIAL(1,0,2))

ASSIGN 1,FN$SKIDKA

ADVANCE (EXPONENTIAL(1,0,10))

TRANSFER .3,„OSM

ENTER CONS

ADVANCE (EXPONENTIAL(1,0,5))

LEAVE CONS

OSM TRANSFER .4,„UXOD

Пример

```
SEIZE KAS
ADVANCE (EXPONENTIAL(1,0,3))
SAVEVALUE PRIB+,V$SUM
RELEASE KAS
UXOD TERMINATE
GENERATE 600
TERMINATE 1
START 1
```

Таблицы

- Для описания таблицы используется блок TABLE. В поле метки этого блока задается имя таблицы, в поле A — аргумент таблицы в виде СЧА (исследуемая случайная величина). В поле B указывается верхняя граница первого частотного интервала, в поле C — ширина интервалов, а в поле D — их число, включающее оба полубесконечных интервала.
- Если необходимо исследовать время, проводимое транзактом в очереди, то используется блок QTABLE, в поле A которого указывается имя очереди, время нахождения в которой нас интересует. Остальные поля аналогичны блоку TABLE.

Таблицы

- Например, если нас интересует гистограмма времени, проводимого одним транзактом в очереди LIN, то мы можем описать таблицу следующим блоком:

```
TBL QTABLE LIN,10,20,5
```

Таблицы

- Если таблица описана, то транзакты могут фиксировать в ней информацию с помощью блока TABULATE. В поле A этого блока указывается имя таблицы, в которой накапливается информация.
- При входе транзакта в блок TABULATE вычисляется значение аргумента указанной таблицы и определяется, в какой из интервалов таблицы это значение попадает. После этого счетчик соответствующей интервальной частоты увеличивается на 1.
- Если используется блок QTABLE, то блок TABULATE не нужен, вычисление искомой характеристики в этом случае происходит автоматически.

Логические переключатели

- Логические переключатели могут находиться в двух положениях: «включен» и «выключен». Перед началом выполнения программы все переключатели устанавливаются в положение «выключен».
- Для работы с логическими переключателями используется блок LOGIC. При поступлении транзакта в блок состояние логического переключателя, номер или имя которого указан в поле A, меняется в соответствии со следующей мнемоникой.

Логические переключатели

- LOGIC R 1 — логический переключатель с номером 1 устанавливается в состояние «выключен».
- LOGIC S 1 — логический переключатель с номером 1 устанавливается в состояние «включен».
- LOGIC I 1 — состояние логического переключателя с номером 1 инвертируется.
- Состояние логического переключателя может быть проверено в любой части модели с помощью блока GATE или с помощью СЧА LS$<имя логического переключателя>, который принимает значение 1, если логический переключатель «включен», и 0 — в противном случае.

Пример

- Паспортный стол работает с 9 до 18 часов с перерывом на обед с 13 до 14 часов. Посетители приходят в среднем каждые 5 минут, причем все сначала направляются к начальнику паспортного стола, который работает с каждым посетителем в среднем 4 минуты.
- После начальника примерно 5% посетителей покидают отделение (получен отказ либо вопрос решен), а остальные направляются в отдел прописки, в котором работают три паспортистки. Время приема посетителя в отделе прописки равно в среднем 12 минутам (с каждым посетителем). Время прихода и время обслуживания в системе распределено экспоненциально.

Пример

- Те посетители, кто стоял в очереди и не успел обслужиться до обеда, обслуживаются после перерыва в первую очередь. Будем считать, что во время обеденного перерыва никто не приходит.
- Примерно за полчаса до окончания рабочего дня просят не занимать очередь к начальнику паспортного стола, и подошедшие в это время посетители не обслуживаются.
- Проверить, успеют ли все посетители, стоящие в очереди, обслужиться до конца рабочего дня. Протабулировать время нахождения посетителей в очередях к начальнику паспортного стола и в отдел прописки.

Пример

PROP STORAGE 3

TAB1 QTABLE OCH_NACH,10,10,10

TAB2 QTABLE OCH_PROP,10,10,10

RAZN VARIABLE N$VXOD-N$UXOD ;количество посетителей, которые встали в очередь, но не успели обслужиться до конца рабочего дня

Пример

```
;работа начальника паспортного стола
GENERATE (EXPONENTIAL(1,0,5))
GATE LR TIME,FIN ;если рабочий день закончился — уход
VXOD QUEUE OCH_NACH
GATE LR OBED ;ожидание окончания обеда
SEIZE NACH
DEPART OCH_NACH
ADVANCE (EXPONENTIAL(1,0,4))
RELEASE NACH
TRANSFER .05„UXOD
```

Пример

```
;работа отдела прописки
QUEUE OCH_PROP
GATE LR OBED ; ожидание окончания обеда
ENTER PROP
DEPART OCH_PROP
ADVANCE (EXPONENTIAL(1,0,12))
LEAVE PROP
UXOD TERMINATE
FIN TERMINATE
```

Пример

```
GENERATE 240„,1 ; начало рабочего дня
LOGIC S OBED ; начало обеда
ADVANCE 60
LOGIC R OBED ; окончание обеда
ADVANCE 210
LOGIC S TIME ; за 30 минут до конца рабочего дня
ADVANCE 30
SAVEVALUE NO_OBSL,V$RAZN ;подсчет не обслуженных
TERMINATE 1
START 1
```

Синхронизация транзактов

- Для моделирования одновременного начала нескольких процессов предназначен блок SPLIT. В момент входа транзакта в блок SPLIT создается несколько копий этого транзакта. Число копий задается в поле A.
- Все копии переходят в блок, определенный в поле B. Исходный (порождающий) транзакт переходит к блоку, следующему за SPLIT.
- Если поле C блока SPLIT пустое, то все копии идентичны породившему их транзакту.

Синхронизация транзактов

- Например, при входе транзакта в блок SPLIT 4,NEXT порождается четыре транзакта, идентичных вошедшему, и передается в блок с меткой NEXT. Породивший их транзакт передается в блок, записанный после блока SPLIT. Всего из этого блока SPLIT выходит пять транзактов.
- Если поле C непустое, то его значение интерпретируется как номер или имя параметра транзакта. Пусть N — значение этого параметра в момент входа транзакта в блок SPLIT. Тогда в момент выхода из SPLIT данный параметр у исходного транзакта будет иметь значение N+1, а у копий транзактов соответственно N+2, N+3, ..., N+K, где K — общее число вышедших из блока SPLIT транзактов.

Синхронизация транзактов

- Например, если транзакт, имеющий нуль в десятом параметре, войдет в блок SPLIT 2,BLOCK,10 то параметр P10 у этого транзакта приобретет значение 1, а у копий — соответственно 2 и 3.
- Транзакты-копии могут двигаться в модели независимо друг от друга. Копии могут проходить блоки SPLIT и порождать новые копии.
- Множество, состоящее из исходного транзакта и всех его копий, называется семейством транзактов. Копия члена семейства является членом того же семейства. Т.о. любой транзакт — член только одного семейства.

Синхронизация транзактов

- Для одновременного завершения нескольких процессов используется блок ASSEMBLE.
- Этот блок собирает заданное в поле A число транзактов одного семейства и превращает их в один транзакт.
- Первый из транзактов какого-либо семейства, вошедший в блок, задерживается до тех пор, пока в этом блоке не накопится заданное число транзактов того же семейства. После этого первый транзакт выходит из блока ASSEMBLE, а остальные транзакты этого семейства уничтожаются.

Синхронизация транзактов

- В одном блоке ASSEMBLE могут одновременно проходить сборку транзакты, принадлежащие к разным семействам.
- Например, если в блок ASSEMBLE 4 поступают транзакты разных семейств, то транзакты каждого семейства собираются по четыре и каждая четверка превращается в один транзакт.

Синхронизация транзактов

- Блок GATHER работает аналогично блоку ASSEMBLE с тем отличием, что транзакты, попав в блок GATHER, не уничтожаются, а только задерживаются, и после того, как в блоке накапливается заданное число транзактов, они все переходят к следующему блоку.

Синхронизация транзактов

- Блок MATCH предназначен для синхронизации процессов.
- Если в программе встречается этот блок, то обязательно где-то должен быть еще один такой же блок, но с другим именем.
- В поле A блока указывается имя парного блока MATCH. Когда транзакт попадает в блок, определяется второй блок MATCH и проверяется, находится ли в нем транзакт этого же семейства.
- Если да, то транзакты выходят из обоих блоков одновременно.
- Если в парном блоке нет транзакта, то в первом блоке транзакт задерживается до тех пор, пока во второй блок не поступит транзакт этого же семейства. Т.о. можно

Синхронизация транзактов

- Блок MATCH предназначен для синхронизации процессов.
- Если в программе встречается этот блок, то обязательно где-то должен быть еще один такой же блок, но с другим именем.
- В поле A блока указывается имя парного блока MATCH. Когда транзакт попадает в блок, определяется второй блок MATCH и проверяется, находится ли в нем транзакт этого же семейства.
- Если да, то транзакты выходят из обоих блоков одновременно.
- Если в парном блоке нет транзакта, то в первом блоке транзакт задерживается до тех пор, пока во второй блок не поступит транзакт этого же семейства. Т.о. можно

Пример

- Промоделировать сборку изделий рабочими A, B и C. Изделия в разобранном виде поступают каждые 200-400 мин. Каждое из них разделяется между рабочими A и B, которые параллельно готовят свою часть изделия к сборке. Подготовка состоит из двух фаз, причем после первой фазы производится сверка с одновременным участием обоих рабочих, а затем A и B независимо выполняют вторую фазу работы. На первой фазе рабочий A тратит на работу 80-120 мин, рабочий B — 60-100 мин; на второй фазе рабочий A тратит 45-55 минут, рабочий B — 60-100 минут. После окончания работы рабочими A и B рабочий C выполняет сборку изделия за 45-55 мин, причем он может начинать сборку только тогда, когда оба первых рабочих закончат свою работу.

Пример

GENERATE 300,100 ;поступление изделий

SPLIT 1,MANB ;разделение изделий

SEIZE RABA ;занять рабочего А

ADVANCE 100,20 ; 1-я фаза

FAZ1A MATCH FAZ1B ;ждать, если В не закончил 1-ю фазу

ADVANCE 50,5 ; 2-я фаза

RELEASE RABA

TRANSFER ,MANC

Пример

MANB SEIZE RABB ;занять рабочего B

ADVANCE 80,20

FAZ1B MATCH FAZ1A ;ждать, если A не закончил 1-ю фазу

ADVANCE 80,20

RELEASE RABB

Пример

```
MANC ASSEMBLE 2 ;ждать обе части изделия
SEIZE RABC ;занять рабочего С
ADVANCE 50, 5
RELEASE RABC
TERMINATE 1 ;завершение сборки
START 1000
```

Захват

- Блок PREEMPT — захватить устройство. Транзакт, попадающий в этот блок, захватывает устройство, имя которого указано в поле A блока.
- Если при захвате устройства оно свободно, то транзакт просто занимает устройство, в этом случае блок PREEMPT работает аналогично блоку SEIZE.

Захват

- Если при входе транзакта в блок PREEMPT устройство занято другим транзактом, то в этом случае транзакт входит в блок PREEMPT, а устройство прерывает обслуживание занимающего его транзакта и переключается на обслуживание транзакта, вошедшего в блок PREEMPT.
- При этом из состояния «занято» устройство переходит в состояние «захвачено».
- Когда транзакт, захватывающий устройство, освободит его, устройство возобновит прерванное обслуживание другого транзакта и перейдет в состояние «занято».

Захват

- Если прерываемый транзакт в момент прерывания находится в блоке ADVANCE, то, начиная с момента прерывания, отсчет времени пребывания транзакта в этом блоке прекращается до тех пор, пока не будет восстановлено обслуживание транзакта.
- Таким образом, в момент восстановления прерванного обслуживания транзакта время, оставшееся этому транзакту до выхода из блока ADVANCE, такое же, каким оно считалось непосредственно в момент прерывания.
- Такое прерывание обслуживания называется прерыванием с последующим дообслуживанием.

Захват

- Кроме поля A, в блоке PREEMPT могут быть заданы операнды B, C, D и E.
- Операнд B записывается в виде обозначения PR, задающего приоритетный режим работы блока.
- В этом режиме транзакт захватывает устройство, если оно свободно или обслуживает менее приоритетный транзакт.
- Прерывание обслуживания менее приоритетного транзакта происходит с последующим дообслуживанием.

Захват

- В поле C может быть указана метка какого-либо блока, на который будет передан прерванный транзакт. При этом прерванный транзакт продолжает претендовать на данное устройство.
- В поле D блока может быть задан номер параметра транзакта. Тогда, если прерванный транзакт находится в блоке ADVANCE, то вычисляется остаток времени обслуживания (время дообслуживания), и полученное значение помещается в параметр, заданный в поле D. Прерванный транзакт при этом будет послан в блок, указанный в поле C. Прерванный транзакт продолжает претендовать на данное устройство.

Захват

- Если в поле E блока записано обозначение RE, то прерванный транзакт больше не будет претендовать на данное устройство.
- Если поле E не задано, а поле C указано, то прерванный транзакт не может быть уничтожен до тех пор, пока он явно не освободит устройство. Чтобы не забывать явно освободить устройство, обычно поля C и E применяют одновременно.

Освобождение

- Блок RETURN освобождает устройство.
- Этот блок используется в паре с блоком PREEMPT.
- Если транзакт захватил устройство посредством блока PREEMPT, то освободить его он может только в блоке RETURN.
- Имя освобождаемого устройства задается в поле A блока.

Циклы

- Для организации циклов используется блок LOOP.
- Поле A этого блока содержит имя или номер параметра, который выполняет функцию счетчика циклов.
- Каждый раз при поступлении транзакта в блок LOOP из указанного параметра вычитается единица, и полученная разность снова записывается в данный параметр.
- Как только значение параметра становится равным нулю, транзакт направляется в блок, следующий за блоком LOOP. Если значение параметра остается положительным, то транзакт направляется к блоку, указанному в поле B.

Циклы

- Например, если необходимо, чтобы через блок ADVANCE все транзакты проходили по 10 раз, то этот процесс можно промоделировать следующим образом:

ASSIGN 1,10

FACIL ADVANCE 34,12

LOOP 1,FACIL

Системное время

- В СЧА C1 и AC1 хранится текущее значение системного времени.
- СЧА C1 содержит значение относительного системного времени (с момента последнего блока RESET).
- СЧА AC1 содержит значение абсолютного системного времени (с момента последнего блока CLEAR).
- Данные СЧА доступны пользователю в любой точке программы.

Системное время

- Каждый транзакт при генерации снабжается отметкой времени.
- Время пребывания транзакта в модели содержится в СЧА M1 и MP и отсчитывается от момента рождения:
- M1 = AC1 — <дата рождения>.
- СЧА M1 возвращает время пребывания транзакта в модели, СЧА MPi или MP$<имя параметра> возвращает значение, равное абсолютному системному времени минус значение соответствующего параметра транзакта.

Системное время

- «Дату рождения», зафиксированную блоком GENERATE, можно изменить в любом месте программы, используя блок MARK с пустым полем A.
- Блок MARK с пустым полем A изменяет «дату рождения» на текущее системное время.
- Если в блоке MARK используется поле A, то в этом поле содержится номер или имя параметра транзакта. В этом случае транзакт сохраняет «дату рождения», а в указанном параметре записывается текущее значение AC1.
- Например, работа блока MARK 10 эквивалентна работе блока ASSIGN 10,C1.