ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

Государственное образовательное учреждение высшего профессионального образования
«Уральский государственный университет им. А.М. Горького»

ИОНЦ «Бизнес-информатика»

математико-механический факультет

кафедра математической экономики

УЧЕБНО-МЕТОДИЧЕСКИЙ КОМПЛЕКС

Имитационное моделирование

Учебное пособие

Автор: доцент кафедры
математической экономики
Д.А. Березин

Екатеринбург
2008

§1. Введение в имитационное моделирование

Определение: **Имитационное моделирование** (simulation modelling) - разновидность аналогового моделирования, реализуемого с помощью набора математических инструментальных средств, специальных имитирующих компьютерных программ и технологий программирования, позволяющих посредством процессов-аналогов провести целенаправленное исследование структуры и функций реального сложного процесса в памяти компьютера в режиме «имитации», выполнить оптимизацию некоторых его параметров.

Определение: **Имитационная модель** – специальный программный комплекс, позволяющий имитировать деятельность какого-либо сложного объекта.

Программный комплекс запускает в компьютере параллельные взаимодействующие вычислительные процессы, являющиеся по своим временным параметрам (с точностью до масштабов времени и пространства) аналогами исследуемых процессов. Научное направление Computer Science использует именно такую трактовку имитационного моделирования.

Для создания имитационной модели необходимо специальное программное обеспечение – система моделирования (simulation system). Специфика такой системы определяется технологией работы, набором языковых средств, сервисных программ и приемов моделирования.

Имитационная модель должна отражать большое число параметров, логику и закономерности поведения моделируемого объекта во времени (временная динамика) и в пространстве (пространственная динамика). Моделирование экономических объектов может быть связано также с финансовой динамикой.

Приведем примеры типовых задач, решаемых средствами имитационного моделирования при управлении экономическими объектами:

- моделирование процессов логистики для определения временных и стоимостных параметров;
- управление инвестиционными проектами на различных этапах его жизненного цикла с учетом возможных рисков;

- прогнозирование финансовой деятельности предприятий на конкретный период времени;
- анализ адаптивных свойств, расчет параметров надежности и задержек обработки информации в банковских информационных системах;
- оценка параметров надежности и задержек в централизованных информационных системах с коллективным доступом (кассы продаж ж/д и авиабилетов, системы бронирования и т.д.);
- анализ эксплуатационных параметров корпоративных информационных систем предприятий, пропускной способности информационных каналов и узлов обработки информации;
- моделирование действий курьерской службы в регионе;
- анализ пропускной способности обслуживания населения (торговые комплексы, комбинаты бытового обслуживания, государственные структуры и т.д.).

Приведенный перечень является неполным и охватывает примеры использования имитационных моделей, которые описаны в литературе и применялись на практике.

Среди современных программных комплексов имитационного моделирования можно выделить следующие распространенные пакеты:

«Process Charter» (разработчик - фирма «Scitor», США);

«Powersim» (фирма «Modell Data», Норвегия);

«Ithink» (фирма «High Performance Systems», США);

«Extend+BPR» (фирма «Imagine That!», США);

«ReThink» (фирма «Gensym», США);

«Pilgrim» (разработчики системы – Московский институт статистики и информатики; фирма «МегаТрон», Россия; фирма «Keisy», Нидерланды; фирма «Enit», Эстония).

В данном учебном пособии рассматривается программная версия «Pilgrim-5». Достоинства этой системы:

- ориентация на совместное моделирование материальных, информационных и финансовых процессов;
- наличие развитой CASE-оболочки, позволяющей конструировать многоуровневые модели;
- наличие интерфейсов с базами данных;
- возможность непосредственного анализа результатов моделирования благодаря технологии создания функциональных окон наблюдения за моделью с помощью Visual C++, Delphy или других средств;
- возможность управления моделями непосредственно в процессе их выполнения с помощью специальных диалоговых окон.

§2. Системы массового обслуживания

2.1. Понятие СМО

Построение имитационных моделей в программной среде пакета «Pilgrim» тесно связано с понятием систем массового обслуживания.

Определение: **Система массового обслуживания (СМО)** – это совокупность приборов, каналов, станков, линий обслуживания, на которые в случайные или детерминированные моменты времени поступают заявки на обслуживание.

В качестве примеров СМО можно рассматривать коммутаторы телефонных станций, вычислительные комплексы, торговые терминалы, банковские системы, информационные службы, комбинаты бытового обслуживания и др.

СМО состоит из определенного числа обслуживающих каналов и предназначена для выполнения заявок с разным характером распределения момента времени на обслуживание.

Предметом теории массового обслуживания является построение математических моделей, связывающих заданные условия работы СМО (число заявок, каналов, характер потока заявок и т.п.) с показателями эффективности СМО, описывающими ее способность справляться с потоком заявок.

В качестве показателей эффективности СМО могут рассматриваться среднее число заявок, обслуживаемых в единицу времени, среднее время ожидания обслуживания, среднее число заявок в очереди, вероятность отказа в обслуживании без ожидания, вероятность превышения числа заявок в очереди определенного значения и др.

2.2. Классификация СМО.

Классификация СМО проводится по следующим основным признакам:

По числу каналов обслуживания:

- одноканальные;

- многоканальные.

По характеру поступления заявок:

- с отказами;
- с ожиданием (очередью).

По дисциплине обслуживания:

- с приоритетом (абсолютным или относительным);
- без приоритета.

По организации очереди:

- с ограниченной очередью;
- с неограниченной очередью.

По времени ожидания заявок в очереди:

- с ограниченным временем ожидания;
- с неограниченным временем ожидания.

Классификация СМО не ограничивается приведенными выше признаками, и более подробно описана в специальной литературе.

§3. Законы распределения случайных величин

Определение: **Случайный процесс (вероятностный, стохастический)** – это процесс изменения во времени состояния какой-либо системы в соответствии с вероятностными закономерностями.

Процесс работы СМО представляет собой случайный процесс с дискретными состояниями и непрерывным временем. Это означает, что состояние СМО меняется скачком в случайные моменты появления каких-либо событий (например, прихода новой заявки, окончания обслуживания и т.п.).

Для упрощения математического анализа работы СМО предполагают, что процесс этой работы является марковским (случайным процессом без последст-

вий), т.е. для любого момента времени t_0 вероятностные характеристики процесса в будущем зависят только от его состояния в данный момент t_0 и не зависят от того, когда и как система пришла в это состояние.

Поток заявок, поступаемых на обслуживание в СМО, характеризуется интенсивностью λ – частотой появления заявок или средним числом заявок, поступающих в СМО в единицу времени.

Важное значение для моделирования СМО имеют вероятностные характеристики потока поступающих заявок и времени их обслуживания в СМО.

Рассмотрим законы распределения случайных величин, которые будут использоваться нами в дальнейшем для построения имитационных моделей.

3.1. Нормальное распределение (гауссово)

Является одним из наиболее важных и часто используемых видов непрерывных распределений.

<u>*Определение:*</u> Непрерывная случайная величина t имеет **нормальное распределение** с параметрами m и σ, если плотность распределения вероятности $p(t)$ имеет вид:

$$p(t) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(t-m)^2}{2\sigma^2}},$$

где m – математическое ожидание $M(t)$;

σ – среднее квадратическое отклонение.

Дисперсия $D(t)$ нормально распределенной случайной величины равна σ^2.

Сформулируем теорему, являющейся одной из форм центральной предельной теоремы.

<u>*Теорема:*</u> Пусть $T\{n\} = \sum_{i=1}^{n} t_i$ - случайная величина, являющаяся суммой n независимых случайных величин, распределенных по неизвестному закону и имеющих конечный третий абсолютный момент. Тогда при $n \to \infty$ распределение

случайной величины $t = T\{n\}$ стремится к нормальному с математическим ожиданием $M(t)$ и дисперсией $D(t)$, определяемыми из следующих соотношений:

$$M(t) = \sum_{i=1}^{n} m_i \quad D(t) = \sum_{i=1}^{n} \sigma_i^2 ,$$

где m_i – математическое ожидание, σ_i^2 - дисперсия случайной величины t_i.

Практический смысл этой теоремы очевиден. Любые сложные работы на экономических объектах (например, ремонтные и пусконаладочные работы, обработка и ввод информации и т.д.), состоят из многих коротких последовательных элементарных работ, продолжительность которых является случайной величиной. Согласно теореме, общую продолжительность сложной работы можно считать случайной величиной, распределенной по нормальному закону.

Для приближенной оценки среднего квадратического отклонения σ можно использовать правило «трех сигм», согласно которому отклонение по абсолютной величине нормально распределенной случайной величины от ее математического ожидания практически не превышает утроенного среднего квадратического отклонения (с вероятностью 0,9973). Таким образом, если в результате наблюдений над случайной величиной получены a - ее минимальное значение и b - максимальное значение, то можно полагать, что $\sigma = \dfrac{b-a}{6}$.

3.2. Равномерное распределение

Применяется в случае, когда известно, что случайная величина может принять значение в определенных границах с одинаковой вероятностью.

Плотность вероятностей равномерного распределения на интервале $(m-s, m+s)$ имеет следующий вид:

$$p(t) = \begin{cases} 0, & \text{ïðè} \quad t < m - s; \\ \dfrac{1}{2s}, & \text{ïðè} \quad m - s \le t \le m + s; \\ 0, & \text{ïðè} \quad t > m + s \end{cases}$$

где m – математическое ожидание;

s – максимальное отклонение от математического ожидания.

Дисперсия $D(t)$ равномерно распределенной случайной величины находится по формуле:

$$D(t) = \frac{((m+s)-(m-s))^2}{12} = \frac{s^2}{3}$$

3.3. Экспоненциальное распределение (показательное)

Также занимает важное место при проведении системного анализа экономической деятельности. Этому закону подчиняются многие явления, например, срок безотказной работы различных технических устройств, частота поступлений заказов на предприятие, посещение покупателями магазина-супермаркета и др.

Функция плотности вероятностей экспоненциального распределения с параметром λ задается выражением:

$$p(t) = \begin{cases} \lambda e^{-\lambda t}, & \text{ïðè} \quad t \geq 0; \\ 0, & \text{ïðè} \quad t < 0. \end{cases}$$

Для экспоненциального распределения математическое ожидание $M(t) = \frac{1}{\lambda}$ и дисперсия $D(t) = \left(\frac{1}{\lambda}\right)^2$. Таким образом, характерно, что среднее квадратическое отклонение численно равно математическому ожиданию.

Рассмотрим предельную теорему о суперпозиции потоков. Предположим, что наблюдается k независимых потоков событий. В свою очередь в каждом потоке наблюдается m_j $(j = \overline{1,k})$ элементарных событий. Интервалы времени между событиями – независимые случайные величины, распределенные по неизвестному

закону с математическим ожиданием $\frac{1}{\lambda_j}$. Спроектируем моменты всех событий на общую ось времени и рассмотрим случайный интервал времени $t = T\{k\}$ между двумя событиями полученного суммарного потока, состоящего из n событий, где $n = \sum_{j=1}^{k} m_j$.

<u>*Теорема:*</u> При $n \to \infty$ распределение случайной величины интервала $t = T\{k\}$ в суммарном потоке событий, состоящем из k независимых потоков, стремится к экспоненциальному с математическим ожиданием

$$M(t) = \frac{1}{\sum_{j=1}^{k} \lambda_j}.$$

Таким образом, рассматривая, например, обращения физических и юридических лиц к операционистам коммерческого банка, можно полагать, что в суммарном потоке обращений интервал времени между двумя последовательными обращениями является случайной величиной, распределенной по экспоненциальному закону.

3.4. Обобщенное распределение Эрланга

Используется в случаях, когда длительность какого-либо процесса можно представить как сумму k элементарных последовательных составляющих, распределенных по экспоненциальному закону. Если обозначить математическое ожидание длительности всего процесса как $M(t) = \frac{1}{\lambda}$, при средней длительности элементарной составляющей $\frac{1}{\lambda k}$, то плотность распределения Эрланга представляется следующей формулой:

$$p(t) = \begin{cases} \lambda k \dfrac{(\lambda k t)^{k-1}}{(k-1)!} \lambda e^{-\lambda k t}, & \text{ïðè} \quad t \geq 0; \\ 0, \quad \text{ïðè} \quad t < 0. \end{cases}$$

Дисперсия такого распределения $D(t)=\dfrac{1}{\lambda^2 k}$. Очевидно, что при k=1 мы получаем экспоненциальное распределение.

3.5. Треугольное распределение

Предполагает линейное изменение функции плотности вероятностей. Применяется в случаях, когда имеется информация относительно минимального a, максимального b и наиболее вероятного m значения случайной величины. Функция плотности вероятностей задается формулой:

$$p(t)=\begin{cases} 0, \quad \text{ïðè} \quad t<a; \\ \dfrac{2(t-a)}{(b-a)(m-a)}, \quad \text{ïðè} \quad a\le t<m; \\ \dfrac{2(t-b)}{(b-a)(m-b)}, \quad \text{ïðè} \quad m\le t\le b; \\ 0, \quad \text{ïðè} \quad t>b \end{cases}$$

Математическое ожидание $M(t)$ и дисперсия $D(t)$ определяются следующим образом:

$$M(t)=\frac{a+m+b}{3}; \quad D(t)=\frac{(b-a)^2+(b-m)^2+(m-a)^2}{36};$$

При $m=\dfrac{a+b}{2}$ треугольное распределение случайной величины является симметричным.

§4. Инструментальные средства системы «Pilgrim»

4.1. Этапы построения имитационной модели

Средством построения моделей в системе PILGRIM является графический конструктор Gem (от «Generator models»), представляющий собой набор узлов различного типа. На схеме имитационной модели узел есть графическое изображение некоторого типового процесса, или, можно сказать, что внутри узла работает процесс. При этом элементарный процесс может быть, вообще говоря, представлен несколькими узлами имитационной модели, если этого требует логика его работы.

Таким образом, схема имитационной модели представляет собой направленный граф, вершины которого представляют собой компоненты элементарных процессов, а дуги определяют направление потоков заявок и управляющих воздействий в моделируемой системе.

Конструктор Gem генерирует программу на языке C++, которая в дальнейшем должна быть скомпилирована в составе проекта в среде MS Developer Studio, где предварительно должен быть установлен Visual C++ с библиотеками Pilgrim.

Процесс построения модели в системе PILGRIM включает в себя следующие этапы:

1) Анализ структуры моделируемого процесса;

2) Определение входных и выходных параметров модели;

3) Построение графа модели с помощью конструктора (можно сразу писать программный файл, но конструктор значительно облегчает работу);

4) Описание узлов графа (определение значений параметров узлов модели). Параметры узлов могут быть заданы константами или переменными пользователя. Во втором случае в момент входа в узел очередной заявки значения этих параметров должны быть определены с помощью операторов языка C++;

5) Определение условий выбора узла, если в модели имеются ветвления;

6) Компиляция модели.

4.2. Транзакты, узлы, события, ресурсы. Модельный таймер.

Основной динамической единицей любой модели, работающей под управлением имитатора, является транзакт.

Определение: **Транзакт** - это формальный запрос на какое-либо обслуживание.

В отличие от обычных заявок в СМО, транзакты имеют набор динамически изменяющихся особых свойств и параметров. Пути миграции транзактов по графу модели определяются логикой функционирования компонентов модели в узлах сети.

Транзакт может выполнять следующие действия:

1) порождать группы (семейства) других транзактов;

2) поглощать другие транзакты конкретного семейства;

3) захватывать ресурсы, использовать их на некоторое время и затем освобождать;

4) определять время обслуживания, накапливать информацию о пройденном пути и иметь информацию о своем дальнейшем маршруте.

Основные параметры транзактов:

1) уникальный идентификатор транзакта;

2) идентификатор семейства, к которому принадлежит транзакт;

3) наборы различных ресурсов, захватываемые и используемые транзактом;

4) время жизни транзакта;

5) приоритет транзакта;

6) параметры обслуживания транзакта в узлах, включая вероятностные характеристики.

Примеры транзактов:

а) требование на перечисление денежных средств;

б) заказ на выполнение работы на предприятии;

в) телеграмма, поступающая на коммутатор сообщений;

г) покупатель в магазине;

д) проба загрязненной почвы, ожидающая химического анализа и др.

Узлы графа – это центры обслуживания транзактов. В узлах транзакты могут задерживаться, обслуживаться, порождать семейства новых транзактов, уничтожать другие транзакты. Вид обслуживания транзакта определяется типом узла. Нумерация и присвоение имен узлам графа производится разработчиком модели.

С точки зрения вычислений, в каждом узле порождается независимый вычислительный процесс. Эти процессы выполняются параллельно, реализуются в едином модельном времени, учитывают временную, пространственную и финансовую динамику.

Примеры узлов:

а) счет бухгалтерского учета;

б) производственный или ремонтный участок;

в) транспортное средство, перемещающее ресурсы;

г) информационный центр и др.

Определение: **Событием** называется факт выхода из узла одного транзакта. Интервалы между событиями являются, как правило, случайными величинами. События модели происходят в некотором модельном времени.

Определение: **Модельное время** – это виртуальное время, в котором автоматически упорядочиваются все события, причем не обязательно пропорционально реальному времени, в котором развивается моделируемый процесс.

Определение: **Масштаб времени** – это число, которое задает длительность моделирования одной единицы модельного времени, пересчитанной в секунды, в секундах астрономического реального времени при выполнении модели.

Определение: **Относительный масштаб времени** – это отношение, показывающее, сколько единиц модельного времени содержится в одной единице процессорного времени.

Выделяют четыре разновидности масштаба времени:

1) *Реальный*. Вводится значение выбранной единицы измерения модельного времени, выраженное в секундах. Например, если 1 ед.мод.вр.=1 час, то вводится значение 3600. Относительный масштаб = 1.

2) *Максимально ускоренный.* Вводится значение 0. Время моделирования определяется процессорным временем выполнения модели. Позволяет достигнуть максимального быстродействия.

3) *Пропорционально ускоренный*. Вводится значение, меньшее по сравнению с выбранной единицей модельного времени, выраженной в секундах. Например, если 1 ед.мод.вр.=1 час и введено значение 0.1, то 1 час реального процесса будет моделироваться за 0.1 секунды. При этом относительный масштаб = 1/36000.

4) *Замедленный.* Вводится значение, большее по сравнению с выбранной единицей модельного времени, выраженной в секундах. Например, если 1 ед.мод.вр.=1 час и введено значение 7200, то модель будет выполняться в 2 раза медленнее реального процесса. При этом относительный масштаб = 2.

Для обслуживания транзактов в программной среде «Пилигрим» используются три основных типа **ресурсов**:

- материальные;
- информационные;
- денежные.

В имитационной модели ресурсы характеризуются тремя параметрами:

- мощность – максимальное число ресурсных единиц, которое можно использовать для обслуживания транзактов;
- остаток – число незанятых на данный момент единиц, которые можно использовать для обслуживания транзактов;
- дефицит – количество ресурсных единиц в суммарном запросе транзактов, стоящих в очереди к данному ресурсу.

4.3. Типы узлов имитационной модели

Узлы имитационной модели характеризуются параметрами, задаваемыми пользователем при написании моделирующей программы. Параметры узлов могут иметь как постоянное, так и переменное значение. Первым параметром каждого узла является его символическое имя в модели, отражающее логическое назначение узла. Последним параметром является номер узла, в который должен перейти вошедший в узел транзакт. Ниже описаны принципы работы узлов и назначение дополнительных параметров.

Узел AG - генератор транзактов

Узел AG представляет собой генератор транзактов с бесконечной емкостью. Предназначен для создания транзактов имитационной модели. Транзакты генерируются по одному через определенные пользователем промежутки времени.

Аргументы функции AG (p_1,p_2,p_3,p_4,p_5,p_6,p_7,p_8) имеют следующий смысл:

p_1 – символическое имя узла: строка длиной до 14 символов (char);

p_2 - номер узла-генератора (int);

p_3 - приоритет (int), назначаемый каждому сгенерированному транзакту, - число в диапазоне 1- 32767; при отсутствии приоритета p_3=none;

p_4 – тип функции распределения интервала времени между двумя последовательно сгенерированными транзактами, имеющий значения:

1) norm – нормальное распределение;
2) unif – равномерное распределение;
3) expo – экспоненциальное распределение;
4) erln – обобщенное распределение Эрланга;
5) beta – треугольное распределение

6) none – если интервал между транзактами является детерминированной величиной.

p_5 – величина, зависимая от типа функции распределения (float):

- математическое ожидание интервала времени (float) между двумя последовательно сгенерированными транзактами (p_4 = norm, unif, expo);
- математическое ожидание одного слагаемого этого интервала (p_4 = erln);
- минимальное значение интервала (p_4= beta);
- постоянная величина этого интервала (p_4 = none).

p_6 - величина, зависимая от типа функции распределения (float):

- среднеквадратичное отклонение (p_4 = norm);
- максимальное отклонение от среднего (p_4 = unif);
- значение zero (p_4 = expo, none);
- число слагаемых, входящих в случайный интервал и распределенных по экспоненциальному закону (p_4 = erln);
- наиболее вероятное значение интервала времени между двумя последовательно сгенерированными транзактами (p_4 = beta).

p_7 – величина, зависимая от типа функции распределения (float):

- максимальное значение интервала времени между двумя последовательно сгенерированными транзактами (p_4 = beta);
- значение zero (p_4=norm, unif, expo, erln, none).

p_8 - номер узла (int), в который передается сгенерированный транзакт (узел-приемник).

Параметры генератора могут быть изменены во время работы модели с помощью команды cheg (change generator).

S

Узел SERV – обслуживающий многоканальный прибор

Узел SERV - это обслуживающий прибор (сервер), осуществляющий какое-либо обслуживание транзактов в течение модельного времени, отличного от нуля. С точки зрения имитационной модели обслуживание заключается в задержке транзакта на заданный промежуток времени. Сервер - это одно- или многоканальный обслуживающий прибор, работающий по правилам абсолютных приоритетов или без них и имеющий стек для "прерванных" транзактов (правило относительных приоритетов реализуется в узле типа QUEUE - очередь).

Аргументы функции SERV (p_1,p_2,p_3,p_4,p_5,p_6,p_7,p_8) имеют следующий смысл:

p_1 – символическое имя узла: строка длиной до 14 символов (char);

p_2 - число обслуживающих каналов (int), при этом $1 \le p_2 \le 32767$;

p_3 - дисциплина обслуживания:

- abs - приоритетная, с прерыванием обслуживания менее приоритетного транзакта более приоритетным;
- none – бесприоритетная.

p_4 – тип функции распределения интервала времени обслуживания транзакта в узле SERV:

1) norm – нормальное распределение;
2) unif – равномерное распределение;
3) expo – экспоненциальное распределение;
4) erln – обобщенное распределение Эрланга;
5) beta – треугольное распределение
6) none – если интервал между транзактами является детерминированной величиной.

p_5 – величина, зависимая от типа функции распределения (float):

- математическое ожидание интервала времени обслуживания транзакта (p_4 = norm, unif, expo);
- математическое ожидание одного слагаемого этого интервала (p_4 = erln);
- минимальное значение интервала (p_4= beta);
- постоянная величина этого интервала (p_4 = none).

p_6 - величина, зависимая от типа функции распределения (float):

- среднеквадратичное отклонение времени обслуживания (p_4 = norm);
- максимальное отклонение от среднего времени обслуживания (p_4 = unif);
- значение zero (p_4 = expo, none);
- число слагаемых, входящих в случайный интервал и распределенных по экспоненциальному закону (p_4 = erln);
- наиболее вероятное значение интервала времени обслуживания транзакта (p_4 = beta).

p_7 – величина, зависимая от типа функции распределения (float):

- максимальное значение интервала времени обслуживания транзакта (p_4 = beta);
- значение zero (p_4=norm, unif, expo, erln, none).

p_8 - номер узла (int), в который передается обслуженный транзакт.

Если задана приоритетная дисциплина обслуживания (p_3 = abs), то имеются две возможности для работы с прерванными неприоритетными транзактами:

- они дообслуживаются после ухода приоритетного транзакта, причем сохраняется чистое заданное время обслуживания без учета времени нахождения прерванного транзакта в специальном стеке имитатора;
- характер прерывания обслуживания транзакта был таков, что нужно возобновить обслуживание заново.

Q

Узел QUEUE - очередь транзактов (с приоритетами или без приоритетов)

Узел QUEUE моделирует очередь транзактов. Строится по одному из двух правил: либо транзакты упорядочены в порядке поступления, либо вновь поступающие транзакты поступают в конец своей приоритетной группы (более приоритетные транзакты находятся ближе к началу очереди, а менее приоритетные - ближе к концу). Чем больше численное значение приоритета транзакта, тем он приоритетнее.

Аргументы функции QUEUE (p_1,p_2,p_3):

p_1 – символическое имя узла: строка длиной до 14 символов (char);

p_2 - тип организации очереди (int):

- prty, если очередь с приоритетами;
- none, если очередь без приоритетов.

p_3 – номер узла-приемника (int).

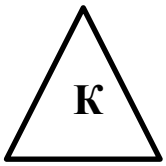
T

Узел TERM - безусловный терминатор транзактов

Узел TERM удаляет из модели входящий в него транзакт и фиксирует время его существования, начиная с момента выхода этого транзакта из генератора.

Аргумент функции TERM (p_1):

p_1 – символическое имя узла: строка длиной до 14 символов (char);



Узел KEY - клапан на пути транзактов

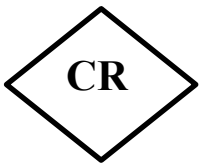
Узел KEY - клапан или ключ - работает в модели по принципу “шлагбаума”. Когда клапан закрыт, транзакт не может в него войти из предшествующего узла. Если же клапан открыт, то транзакт проходит через него в следующий узел без задержки. Среднее время пребывания такого узла в закрытом состоянии подсчитывается автоматически. Для управления этим клапаном или ключом существуют вспомогательные функции hold и rels.

Среднее время задержки - это среднее время пребывания ключа в закрытом состоянии. Число обслуженных транзактов - это число переключений ключа из закрытого состояния в открытое.

Аргументы функции KEY (p_1. p_2):

p_1 – символическое имя узла: строка длиной до 14 символов (char);

p_3 – номер узла-приемника (int).



Узел CREAT - управляемый генератор транзактов (мультипликатор)

Узел CREAT предназначен для создания нового семейства транзактов. Все транзакты в модели принадлежат какому-то семейству. Транзакты, выходящие из обычного генератора AG, принадлежат к семейству с номером 0. Узел CREAT, в отличие от AG - это управляемый генератор. Назначение его самое различное. В замкнутых моделях он применяется для схемы зарядки.

Аргументы функции CREAT (p_1,p_2,p_3,p_4,p_5,p_6) имеют следующий смысл:

p_1 – символическое имя узла: строка длиной до 14 символов (char);

p_2 – идентификатор (int) порождаемого семейства транзактов;

p_3 – число порождаемых транзактов (int);

p_4 – имеет следующие значения:

- copy – для тиражирования параметров порождающего транзакта каждому порожденному (включая время жизни);
- none - для присвоения каждому порожденному транзакту в качестве параметров нулевых значений.

p_5 – номер узла (int), в который направляются порожденные транзакты.

p_6 – номер узла (int), в который направляется порождающий транзакт.

Логика функционирования узла CREAT:

1) через узел проходит порождающий транзакт, который принадлежит семейству f_1, и поступает в узел p_6;

2) одновременно с этим в узле генерируются p_3 новых транзактов, принадлежащих семейству с номером $f_2 = p_2$, которые будут направлены в узел p_5. В общем случае p_5 и p_6 - любые узлы (кроме ag), в частности - это может быть один и тот же узел. Номера семейств f_1 и f_2 в общем случае могут совпадать.

Узел DELET - управляемый терминатор транзактов (демультипликатор)

Узел DELET предназначен для уничтожения группы транзактов, принадлежащих семействам из диапазона, задаваемому параметрами p_2 и p_3. В отличие от терминатора TERM данный узел управляется специальным поглощающим транзактом.

Аргументы функции DELET (p_1,p_2,p_3,p_4,p_5,p_6):

p_1 – символическое имя узла: строка длиной до 14 символов (char);

p_2 – начало (int) диапазона номеров семейств уничтожаемых транзактов;

p_3 – конец (int) диапазона номеров семейств уничтожаемых транзактов;

p_4 – идентификатор (int) семейства, к которому принадлежит поглощающий транзакт;

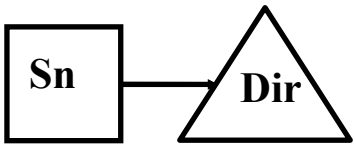
p_5 – число уничтожаемых транзактов (int);

p_6 – номер узла (int), в который направляется поглощающий транзакт.

Логика функционирования узла DELET:

1) в узел входит поглощающий транзакт семейства p_4 и находится там до тех пор, пока в него не поступят p_5 транзактов семейства из диапазона $[p_2;p_3]$. После поступления требуемого количества транзактов они мгновенно уничтожаются, а поглощающий транзакт переходит в узел p_6.

2) если в узел DELET так и не поступят p_5 транзактов, то поглощающий транзакт будет все время находиться в нем, блокируя его для поступления других поглощающих транзактов. Для принудительного изгнания такого транзакта из узла используется функция FREED(i).



Узлы SEND и DIRECT - операции с денежными ресурсами

Программные средства системы PILGRIM хорошо подходят для моделирования финансовой динамики на счетах бухгалтерского учета предприятия (фирмы). Узлом считается счет (субсчет) бухгалтерского учета. Предположим, что номер этого узла - i. Транзакт, вошедший в узел i, - это запрос на проводку со счета i определенной суммы на какой-то другой счет. Для осуществления проводки необходимо, чтобы на счете i была сумма не менее требуемой. При отсутствии такой суммы транзакт ожидает момент поступления на счет i достаточных средств. Таким образом, узел с номером i, который формирует запрос на бухгалтерскую проводку, - это специальная очередь транзактов. Для описания узла-счета с номером i предназначен узел SEND (p_1,p_2,p_3,p_4,p_5) со следующими аргументами:

p_1 – символическое имя узла: строка длиной до 14 символов (char);

p_2 – узел-счет, на который требуется перевести заданную сумму (int).

p_3 – размер заданной суммы (double). В обязательном порядке определяется до второго знака после запятой.

p_4 – возможность работы с приоритетами:

- prty, если требования на перечисление денег удовлетворяются с учетом приоритетов. В случае отсутствия необходимой суммы на счете i транзакты образуют очередь в узле SEND, причем ближе к голове очереди находится более приоритетная группа транзактов. Внутри приоритетной группы транзакты расположены по возрастанию требуемой суммы. Если запрашиваемые суммы одинаковы, то транзакты расположены в хронологическом порядке (правило FIFO – «First In First Out»).
- none, если требования на перечисление не имеют приоритетов. Тогда организация транзактов в очереди осуществляется по правилу FIFO.

p_5 – номер узла типа DIRECT, играющего роль «финансового директора» и выполняющего проводки по мере необходимости.

Событием в узле типа SEND является факт выполнения проводки со счета i на счет p_2. Момент времени этого события - это момент времени проводки, определяемый выводом транзакта из узла SEND.

В каждом узле SEND имеется внутренний атрибут saldo, который отражает остаток средств на счете i. Дефицит средств на счетах бухгалтерского учета содержится в атрибуте defic. Если атрибут saldo в узле i имеет нулевое значение, и в этом узле имеются транзакты (один или несколько), запросившие проводки, то суммарный дефицит затребованных этими транзактами сумм автоматически отражается в атрибуте defic.

Для управления узлом SEND используется функция ASSIGN (p_1,p_2,p_3) со следующим параметрами:

p_1 – номер узла-счета SEND (int);

p_2 – признак:

- add, если требуется добавить средства на счет в размере p_3 ;

- none, если требуется установить сальдо счета в размере p_3 единиц;

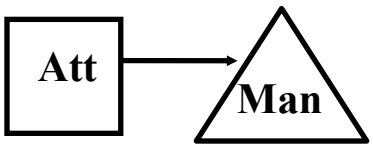
p_3 – размер денежной суммы (double).

Узел DIRECT является своеобразным клапаном, через который могут пройти только те транзакты, проводки по которым можно выполнить. Обслуживание всех узлов SEND, имеющихся в модели может выполняться как одним, так и несколькими узлами типа DIRECT.

Аргументы функции DIRECT (p_1,p_2):

p_1 – символическое имя узла: строка длиной до 14 символов (char);

p_2 – узел-приемник транзакта, выполнившего проводку Этот узел может быть любого типа, кроме DIRECT.



Узлы ATTACH и MANAGE - операции с материальными ресурсами

Предназначены для моделирования динамики использования материальных ресурсов, необходимых для работы в моделируемых процессах. Под ресурсом могут пониматься любые объекты окружающего мира (сырье, оборудование, персонал и т.д.). Объем ресурса в модели измеряется в целых числах.

В модели все виды используемых процессами ресурсов хранятся в специальных узлах-«складах». Узел-«склад» имеет название ATTACH. При посещении транзактом узла ATTACH ему выдается некоторое количество единиц хранящегося там ресурса. Если ресурса на складе меньше, чем требуется транзакту, он остается в узле до тех пор, пока объем имеющегося ресурса не станет больше. Таким образом, узел ATTACH является очередью запросов на получение ресурса. Очередь запросов может быть обыкновенной, а может учитывать приоритет запроса. Внутри приоритетной группы транзакты располагаются в следующем порядке: чем меньше объем запрашиваемого ресурса, тем ближе транзакт находится

к голове своей приоритетной группы. В очереди без приоритетов работает правило FIFO.

Аргументы функции ATTACH (p_1,p_2,p_3,p_4):

p_1 – символическое имя узла: строка длиной до 14 символов (char);

p_2 – требуемое количество элементов ресурса (long);

p_3 – возможность работы с приоритетами:

- prty, если требования на ресурсы удовлетворяются с учетом приоритетов. В случае отсутствия необходимого объема ресурсов транзакты образуют очередь в узле ATTACH, причем ближе к голове очереди находится более приоритетная группа транзактов. Внутри приоритетной группы транзакты расположены по возрастанию требуемого объема ресурсов. Если требования одинаковы, то транзакты расположены в хронологическом порядке (правило FIFO – «First In First Out»).
- none, если требования на перечисление не имеют приоритетов. Тогда организация транзактов в очереди осуществляется по правилу FIFO.

p_4 – номер узла-приемника типа MANAGE. (int), выполняющего функцию «управляющего складом».

Узел MANAGE проверяет наличие требуемого количества ресурса для каждого транзакта, пришедшего в ATTACH. Если ресурса достаточно, транзакт проходит через узел MANAGE и «путешествует» с захваченными единицами по графу модели. Транзакт может несколько раз становиться в очередь к одному и тому же ресурсу, получая дополнительные единицы. Если ресурсов недостаточно, транзакт остается в очереди в узле ATTACH. Таким образом, узел MANAGE - это своеобразный клапан.

Узел MANAGE имеет параметры (p_1,p_2):

p_1 – символическое имя узла: строка длиной до 14 символов (char);

p_2 – узел-приемник транзакта. Таким узлом может быть любой узел модели, кроме MANAGE.

Для изменения объема ресурса, имеющегося на складе ATTACH используется функция SUPPLY (p_1,p_2,p_3) со следующим параметрами:

p_1 – номер узла ATTACH (int);

p_2 – признак:

- add, если требуется добавить ресурс на склад в объеме p_3 единиц;
- none, если требуется заменить складской запас на объем p_3 единиц;

p_3 – количество единиц ресурса (long).

Остаток ресурса и дефицит фиксируется в атрибутах узла rsal и rdef. Дефицит равен суммарному объему запросов всех стоящих в очереди в узле ATTACH транзактов.

Полученный на складе ресурс транзакт может вернуть обратно на склад из любого узла модели с помощью команды DETACH (p_1,p_2), где

p_1 – номер узла-склада ATTACH (int);

p_2 – число возвращаемых единиц ресурса.

Узел PROC – транзактно-управляемый процесс

Специальная функция моделирования транзактно-управляемого непрерывного процесса PROC.

Объединяет в себе возможности имитации:

1) обслуживание в узле подобно SERV с одним каналом;

2) перемещение узла по общему полю данных на координатной сетке;

3) запуск на время активности функции типа float p_2(d), где d - элементарный интервал активности, который определяется системой PILGRIM в процессе моделирования и зависит от параметров p_3,p_4,p_5,p_6.

Интервал требуемого обслуживания транзакта может быть меньше времени пребывания транзакта в этом узле, так как процесс может быть переведен в состояние "пассивен" или "активен" каким-то транзактом из другого узла с помощью функций ACTIV и PASSIV.

Если процесс пассивен, то обслуживание транзакта приостанавливается, а выполнение функции p_2 прерывается до тех пор, пока процесс не будет переведен в активное состояние.

Аргументы функции PROC (p_1,p_2,p_3,p_4,p_5,p_6,p_7,p_8):

p_1 – символическое имя узла: строка длиной до 14 символов (char);

p_2 - возможность подключения внешней программы:

- имя (адрес) программы, написанной пользователем, и моделирующей процесс с помощью формул, дифференциальных уравнений и т.д.;
- dummy, если внешняя программа отсутствует.

p_3 - номер исходной точки, в которую устанавливается узел PROC перед началом моделирования (int);

p_4 - тип функции распределения интервала активности процесса, либо тип координатного пространства. Если это тип функции распределения, то используются условные обозначения:

- norm – нормальное распределение;
- unif – равномерное распределение;
- expo – экспоненциальное распределение;
- erln – обобщенное распределение Эрланга;
- beta – треугольное распределение
- none – если интервал является детерминированной величиной.

Для задания типа пространства используются обозначения:

- earth – поверхность Земли (географические координаты широта и долгота);
- plane – декартова плоскость (прямоугольная система координат);
- cosmos – произвольное пространство;
- none – если режим пространственной имитации в данном узле не нужен.

p_5 – величина (float), зависимая от параметра p_4, определяющего тип функции распределения интервала активности процесса, либо тип координатного пространства:

- математическое ожидание интервала времени активности (p_4 = norm, unif, expo);
- математическое ожидание одного слагаемого этого интервала (p_4 = erln);
- минимальное значение интервала (p_4= beta);
- постоянная величина этого интервала (p_4 = none);
- интервал "непрерывного" нахождения этого узла на координатной сетке (p_4= earth, plane, cosmos).

p_6 – параметр (float) интервала активности обслуживания, зависимый от параметра p_4:

- среднеквадратичное отклонение времени обслуживания (p_4 = norm);
- максимальное отклонение от среднего времени обслуживания (p_4 = unif);
- значение zero (p_4 = expo, none);
- число слагаемых, входящих в случайный интервал и распределенных по экспоненциальному закону (p_4 = erln);
- наиболее вероятное значение интервала времени обслуживания транзакта (p_4 = beta);
- скорость перемещения узла от одной точки пространства к другой (p_4= earth, plane, cosmos).

p_7 – величина (float), зависимая от параметра p_4:

- значение zero (p_4=norm, unif, expo, erln, none).
- максимальное значение интервала времени обслуживания транзакта (p_4 = beta);
- p_8 – номер узла-приемника (int).

Din

Узел DINAM - очередь к процессу с динамическими пространственно-зависимыми приоритетами

Узел DYNAM предназначен для моделирования обслуживания транзактов в очереди с динамическими пространственно-зависимыми приоритетами. Эта функция моделирует оптимально-управляемую очередь, которая находится на входе узла типа PROC.

Аргументы функции DYNAM (p_1,p_2):

p_1 – символическое имя узла: строка длиной до 14 символов (char);

p_2 - номер узла-приемника, в который передается транзакт. Узел PROC обязательно должен использовать свой параметр p_4= earth, plane или cosmos.

Узел DYNAM представляет собой очередь транзактов, направляющихся на обслуживание в узел PROC. Порядок обслуживания в этой очереди пересматривается каждый раз при поступлении в хвост очереди нового транзакта и при переходе первого транзакта в узел PROC.

Функция DYNAM всегда "заглядывает" в узел PROC и анализирует, не следует ли прервать обслуживание находящегося в нем транзакта. Если такое решение будет принято, то вычисляется местонахождение текущей точки пространства, в которой находится узел PROC, далее функция DYNAM извлекает из этого узла транзакт обратно в свою очередь и посылает в него более "выгодный" с точки зрения оптимизации транзакт.

В основу процедуры управления очередью DYNAM положен оригинальный алгоритм оптимизации динамического расписания, обладающий высоким быстродействием и имеющий практическое применение.

4.4. Управление переходами в многослойных моделях

При исследовании и моделировании сложного процесса разработчику бывает целесообразно осуществить декомпозицию данного процесса на ряд подпроцессов и корректно отобразить это в имитационной модели

Декомпозиция в имитационной модели – это детализация одного узла с помощью совокупности других узлов.

Существуют четыре разновидности декомпозиции процессов:

1) общий случай декомпозиции сложного процесса с помощью узлов типа DOWN;
2) декомпозиция процессов перечисления денежных средств с помощью узлов типа PAY;
3) декомпозиция процессов выделения ресурсов с помощью узлов типа RENT;
4) абстрактное объединение группы процессов в один псевдопроцесс с помощью виртуального узла PARENT без образования нового узла.

Down

Узел DOWN - переход на нижерасположенный слой

Предназначен для объединения большой группы узлов в один общий узел, который находится на одном из слоев модели. Этот узел подвергается декомпозиции на слоях модели, которые расположены ниже.

Аргументы функции DOWN (p_1,p_2,p_3):

p_1 – символическое имя узла: строка длиной до 14 символов (char);

p_2 - номер узла-приемника на нижнем слое модели (int);

p_3 - номер узла возврата на данном слое, где расположен узел DOWN.

Pay

Узел PAY - имитация перечисления денежной суммы

Предназначен для имитации перечисления денежных средств. Подлежит детализации на более низком уровне с помощью узлов SEND и DIRECT.

Аргументы функции PAY (p_1,p_2,p_3,p_4,p_5,p_6,p_7):

p_1 – символическое имя узла: строка длиной до 14 символов (char);

p_2 – номер узла-счета типа SEND, на который переводится денежная сумма (int);

p_3 – значение денежной суммы (double);

p_4 – номер узла-счета типа SEND, с которого переводится денежная сумма (int);

p_5 – признак работы с приоритетами (prty или none);

p_6 – номер узла-приемника на нижнем слое (int);

p_7 – номер узла возврата на данном слое модели, где расположен узел PAY (int).

Rent

Узел RENT - имитация получения ресурса со склада

Работа данного узла схожа с работой узла PAY. Подлежит декомпозиции на более низком уровне с помощью узлов ATTACH и MANAGE.

Аргументы функции RENT (p_1,p_2,p_3,p_4,p_5,p_6):

p_1 – символическое имя узла: строка длиной до 14 символов (char);

p_2 – требование на число элементов ресурса (long);

p_3 – номер узла-склада ресурсов ATTACH, с которого необходимо получить ресурсы (int);

p_4 – признак работы с приоритетами (prty или none);

p_5 – номер узла-приемника на нижнем слое (int);

p_6 – номер узла возврата на данном слое модели, где расположен узел RENT (int).

Parent

Узел PARENT - объединение группы процессов

Является особым типом узла, использующимся исключительно как средство создания наглядных многоуровневых моделей в графическом конструкторе GEM. Виртуальный узел типа PARENT содержит ссылку на плоскость, его детализирующую, т.е. используется для описания процессов, которые можно рассмотреть не отдельной плоскости. При этом он не выполняет никаких действий по обработке транзакта. При генерации программного кода заменяется своей декомпозицией.

4.5. Сигнальные управляющие функции

Сигнальные управляющие функции могут использоваться в любом узле (кроме узла-генератора AG), где необходима специальная логика или должны использоваться функциональные уточнения особенностей узла.

Описание функций:

INTERRUPT () – выполнить прерывание модели. После выполнения этой функции в каком-либо узле работа модели прекращается, а экспериментатор может исследовать полученные результаты.

CHEG (p_2,p_3,p_4,p_5,p_6,p_7,p_8) – перенастроить генератор транзактов AG(p_1,p_2,p_3,p_4,p_5,p_6,p_7,p_8) на новые значения параметров. Номер перенастраиваемого генератора содержится в параметре p_2. Аргументы $p_3 \ldots p_8$ содержат новые значения параметров и имеют тот же смысл, что и при настройке генератора AG.

RELS (i) – открыть узел-клапан KEY с номером i. После выполнения RELS клапан принимает состояние «открыт», если до этого он был закрыт.

HOLD (i) – закрыть узел-клапан KEY с номером i. После выполнения HOLD клапан принимает состояние «закрыт», если до этого он был в открытом состоянии.

ACTIV (i) – перевести процесс (узел типа PROC) в активное состояние, если он был пассивен. При этом возобновляется обслуживание транзакта (т.е. отсчет активного времени) и выполнение модели процесса (программы с именем p_2). Функция ACTIV может выполняться из любого узла, номер которого не равен i.

PASSIV (i) – перевести процесс (узел типа PROC) в пассивное состояние, если он был активен. При этом транзакт переводится в стек прерванных транзактов. Прекращается выполнение модели процесса (программы с именем p_2). Функция PASSIV может выполняться из любого узла, номер которого не равен i.

SUPPLY (p_1,p_2,p_3) - изменить объем ресурса, имеющегося на складе ATTACH, где p_1 – номер узла ATTACH; p_2 – признак: add, если требуется добавить ресурс на склад в объеме p_3 единиц; none, если требуется заменить складской запас на объем p_3 единиц; p_3 – количество единиц ресурса.

DETACH (p_1,p_2) - вернуть ресурс обратно на склад, где p_1 – номер узла-склада ATTACH; p_2 – число возвращаемых единиц ресурса.

ASSIGN (p_1,p_2,p_3) – ассигновать на счет SEND денежные средства, где p_1 – номер узла-счета SEND; p_2 – признак: add, если требуется добавить средства на счет в размере p_3 ; none, если требуется установить сальдо счета в размере p_3 единиц; p_3 – размер денежной суммы.

FREED (i) – изгнать уничтожающий транзакт из узла DELET. При непоступлении в узел DELET заданного количества уничтожаемых транзактов (параметр p_5), в данном узле все время находится поглощающий транзакт, блокируя тем самым узел DELET для поступления других поглощающих транзактов. При прохождении транзакта через функцию FREED (i) в каком-либо другом узле, с ее помощью генерируется вспомогательный транзакт, который направляется в узел DELET с номером i, выталкивает из этого узла застрявший поглощающий транзакт, приводит DELET в нормальное состояние и сам погибает. Застрявший тран-

закт направляется в узел p_6 в соответствии с заданными параметрами узла DELET.

SEWT (x) – поместить текущий транзакт в точку пространства с номером x. Функция приписывает транзакту координаты этой точки путем занесения значения x в параметр транзакта t→tx.

SEWK (x,i) – поместить узел с номером i в точку пространства с номером x. Функция приписывает узлу координаты этой точки путем занесения значения x в параметр узла k→kx.

GEOWAY (latA, lonA, latB, lonB) – определить расстояние между точками A и B по их географическим координатам, измеряемым в радианах, причем latA, lonA – широта и долгота точки A, latB, lon B – широта и долгота точки B. Для определения расстояния функция использует основные формулы сферической тригонометрии. Радиусы Земли для разных широт вычисляются по эллипсоиду Красовского.

DEKART (xA, yA, xB, yB) - – определить расстояние между точками A и B на декартовой плоскости в прямоугольной системе координат. Расстояние вычисляется по формуле: $L_{AB} = \sqrt{(xA - xB)^2 + (yA - yB)^2}$.

CHANGE (p_1,p_2) – заменить узел обслуживания очереди. Если состав или количество транзактов в узле-очереди (узлы типа QUEUE, SEND, ATTACH) с номером p_1 перестают удовлетворять каким-то требованиям, то с помощью данной функции можно перенаправить все транзакты из узла p_1 в узел p_2, разгрузив тем самым очередь в узле p_1.

CLCODE – включить блок операторов языка C++. Позволяет разработчику помещать в модель свои программные блоки. При этом нельзя использовать операцию goto не только внутри одного узла, но и для перехода в другой узел. Для принудительного завершения моделирования вместо goto можно использовать только функцию interrupt.

После узлового оператора можно записать блок любых операторов языка C++ в формате:

clcode

{

Группа операторов языка C++

}

Если необходимо использовать блок из одного оператора, то это записывается в виде:

clcode

Оператор языка C++

4.6. Параметры транзактов

Для повышения эффективности создаваемых моделей разработчику доступны следующие параметры транзактов:

t→iu0, t→iu1, t→iu2, t→iu3 – произвольные целочисленные параметры, которые могут использоваться для различных целей, например, для помечивания транзакта;

t→ru0, t→ru1, t→ru2, t→ru3 – произвольные параметры, имеющие вид переменной с плавающей точкой;

t→ga – признак обслуживания транзакта вновь, который устанавливается операцией присваивания ему глобальной константы again. Если перед узлом SERV транзакт пройдет через оператор присваивания t→ga= again, то он при прохождении через SERV получает признак обслуживания заново. После выхода из SERV этот признак теряется;

t→pr – приоритет транзакта;

t→ft – номер семейства, к которому принадлежит транзакт;

t→tx – индекс точки пространства. Позволяет определить координаты lat и lon в массиве space.

4.7. Параметры состояния узлов

Некоторые параметры узлов доступны разработчику для анализа (но не для их изменения). Параметры содержатся в системном массиве addr.

Пусть n – номер узла в модели, тогда:

addr[n]→nc – число каналов в узле;

addr[n]→na – число транзактов, прошедших через узел на данный момент модельного времени;

addr[n]→tn – число транзактов, находящихся в узле в данный момент;

addr[n]→ts – среднее время обслуживания, подсчитанное на данный момент;

addr[n]→op – признак состояния узла типа KEY. Целочисленная переменная, принимающая значение true, если клапан открыт, или false, если он закрыт.

addr[n]→se – признак состояния узла типа DELET. Целочисленная переменная, принимающая значение nil, если в узле нет уничтожающего транзакта.

addr[n]→kx – индекс точки, в которой находится узел типа CREAT, DELET или PROC. Позволяет определить координаты lat и lon.

Параметры узлов можно наблюдать в процессе выполнения имитационной модели.

4.8. Датчики случайных величин

Часто в процессе моделирования возникает необходимость получить случайную величину в формате float, распределенную по какому-либо закону. В системе PILGRIM есть стандартные 32-разрядные датчики псевдослучайных величин. В каждом узле есть свой датчик, независимый от датчиков других узлов. Связь с этими датчиками осуществляется с помощью следующих функций:

normal(m,s) - нормальный закон распределения;

expont(m) - экспоненциальный закон;

unifrm(m,r) - равномерный закон на отрезке [m-r,m+r];

rundum() - равномерный закон на отрезке [0,1];

erlang(e,z) - обобщенный закон Эрланга;

triplex(a,b,c) – треугольный закон распределения,

где m- математическое ожидание (в случаях normal, expont, unifrm);

s - среднеквадратичное отклонение (в случае normal);

e- математическое ожидание величины одного элемента (в случае erlang);

r - максимальное отклонение (в случае unifrm);

z - количество отрезков (в случае erlang), z > 0;

a - минимальное значение (в случае triplex);

b - наиболее вероятное значение (в случае triplex);

c - максимальное значение (в случае triplex).

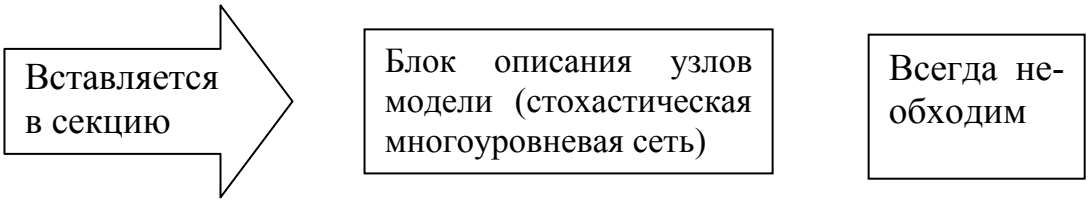
При обращении к одной из этих функций используется датчик того узла, в котором находится транзакт.

4.9. Структура имитационной модели в системе Pilgrim

Модель состоит из двух характерных частей: секции инициализации и блока описания стохастической сети.

№ п/п	Оператор	Необходимость применения в модели
1	#include <Pilgrim.h>	всегда необходим
2	[Глобальные переменные и функции]	только в сложных моделях
3	forward	всегда необходим
4	{	
5	[Локальные переменные модели]	часто необходимы
6	modbeg (p_1,p_2,p_3,p_4,p_5,p_6,p_7,p_8,p_9)	всегда необходим
7	ag (p_1,p_2,p_3,p_4,p_5,p_6,p_7,p_8) ag (p_1,p_2,p_3,p_4,p_5,p_6,p_7,p_8)	всегда необходимы

	…… ag (p_1,p_2,p_3,p_4,p_5,p_6,p_7,p_8)	
8	[Сигнальные функции]	В ресурсных моделях
9	network (p_1,p_2)	всегда необходим
10	{	всегда необходим



11	}	всегда необходим
12	modend (p_1,p_2,p_3,p_4)	всегда необходим
13	return 0;	всегда необходим
14	}	всегда необходим

Рис.4.1. «Структура секции инициализации модели»

Секция инициализации модели состоит из 14 компонентов, причем некоторые из них являются необязательными (на рисунке заключены в квадратные скобки).

Макрооперация #include <Pilgrim.h> подключает моделирующую среду имитатора к модели. Далее следуют описания глобальных переменных и дополнительных функций, используемых в модели.

Собственно модель начинается с оператора forward. За ним, если это необходимо, указываются локальные переменные модели.

Оператор первоначальной настройки MODBEG (p_1,p_2,p_3,p_4,p_5,p_6,p_7,p_8,p_9) осуществляет первоначальную настройку моделирующих программ и инициализацию в памяти ЭВМ графа модели.

Аргументы этой функции имеют следующий смысл:

p_1 – символическое имя узла: строка длиной до 14 символов (char);

p_2 – максимальный номер узла модели (int), причем $2 \leq p_2 \leq m_{max}$, где m_{max} – некоторое граничное значение, задаваемое при установке имитатора на ЭВМ (обычно m_{max} =1024);

p_3 - модельное время, в течение которого необходимо производить моделирование (float);

p_4 - произвольное целое число, используемое для настройки датчиков псевдослучайных величин (long). В каждом узле есть свой независимый датчик. В качестве этого числа полезно использовать значение таймера ЭВМ, обращение к которому имеет следующий вид p_4=(long)time(NULL). В этом случае результаты разных прогонов модели будут разными, имеющими случайные отклонения. При отладке лучше использовать постоянную комбинацию цифр, например, p_4=(long)2013456789.

p_5 – признак режима пространственной имитации (int):

- earth – поверхность Земли (сферические географические координаты широта и долгота);
- plane – декартова плоскость (прямоугольная система координат);
- cosmos – произвольное пространство (ответственность за правильность его представления возлагается на разработчика модели) ;
- none – если пространственная имитация в модели не используется.

p_6 – номер (int) одной из очередей (узел типа QUEUE, ATTACH или SEND), которую необходимо контролировать во времени для анализа динамики задержек в этой очереди с графическим отображением результатов. Например, если указать число 3, и в модели действительно имеется очередь с таким номером, то можно получать изображение динамики этой очереди непосредственно в процессе моделирования.

p_7 – номер (int) одного из процессов (узла типа PROC), который необходимо контролировать как в пространстве, так и во времени с графическим отображением результатов. Если нет необходимости в графической интерпретации, то указывается none.

p_8 – номер (int) терминатора (узел типа TERM), на входе которого необходимо наблюдать интенсивность потока транзактов во время моделирования. Если такой необходимости нет, то указывается none.

p_9 – точность:

- если p_9=1..6, то имитатор будет использовать от 1 до 6 знаков после запятой при выводе результатов;
- если p_9=none, то результаты будут округляться до целых значений.

За оператором MODBEG следуют описания узлов-генераторов AG. Это единственный тип узла, который описывается в секции инициализации модели. Эти генераторы не зависят от состояния других узлов и должны работать до того, как узлы других типов будут приведены в рабочее состояние.

Координатор сети процессов NETWORK (p_1,p_2) осуществляет диспетчеризацию транзактов в узлах модели, планирует события в едином модельном времени и активизирует дискретные или непрерывные компоненты модели, имитирующие внешнюю среду. Координатор выполняет всю модель либо до первой ошибки, либо пока не истечет время, заданное аргументом p_3 в функции MODBEG.

С точки зрения синтаксиса языка программирования эта функция является структурным оператором, за которым следует фигурная скобка { , открывающая блок с описанием графа модели (стохастической сети). Этот блок заканчивается закрывающейся скобкой }. Обращение к этой функции делается только после того, как выполнены MODBEG и все AG.

Аргументы p_1 и p_2 - это имена (адреса) соответствующих программных функций моделирования внешней среды, производящих интегрирование, решение разностных уравнений, вычисление по формулам и т.д. Эти функции возвращают значение типа float. Им координатор передает единственный аргумент float d – интервал времени, в течение которого они получают управление. Аргумент d в функциях p_1 и p_2 это глобальная переменная, содержащая интервал времени между двумя ближайшими событиями. Разработчик модели лишен возможности самостоятельно (т.е. напрямую из модели) вызывать p_1 и p_2 во избе-

жание непредсказуемых или ложных результатов, а переменная d изменяется только координатором NETWORK.

Функции float p_1(d) и float p_2(d), если они необходимы. пишутся пользователем. Они должны производить либо интегрирование, либо решение разностных уравнений на отрезке времени d, который каждый раз передается в качестве параметра. В общем случае, кроме C++, для этого может использоваться Паскаль или пакет математических программ.

Если процессы p_1(d) и p_2(d) не моделируются, то в качестве p_1 и p_2 указывается слово dummy.

Оператор завершения моделирования MODEND (p_1,p_2,p_3,p_4) выполняется после того, как будет нарушено внутреннее условие noerr, определяющее, что в модели зафиксирована грубая ошибка, либо истекло время моделирования, указанное в операторе MODBEG. Данный оператор удаляет все созданные в процессе моделирования управляющие структуры из памяти ЭВМ. Далее эта функция позволяет просмотреть на экране монитора графические результаты и выводит итоговые результаты в файл-отчет на жестком диске.

Аргументы функции MODEND (p_1,p_2,p_3,p_4) следующие:

p_1 – символическое имя специального файла-отчета, в который записываются стандартные характеристики модели, полученные и замеренные в процессе ее выполнения. Это имя (char) является строкой и оформляется по системным соглашениям (например, имя Results1.doc).

p_2 – номер первой страницы отчета (int);

p_3 - число строк на каждой странице (int);

p_4 - имеет два значения:

- page, если в файле-отчете необходимо проставить символ перевода страницы;
- none, если символ перевода страницы не ставится.

§5. Инсталляция программного пакета «Pilgrim-5»

Программный пакет имитационного моделирования «Pilgrim-5» является лицензионным продуктом и поставляется на компакт-диске с уникальным серийным номером.

Дистрибутивный диск содержит каталог с именем «Pilgrim-5-60», который в свою очередь состоит из следующих разделов:

Gem – диалоговый конструктор имитационных моделей;

GeoMaps – карта Московской области для работы с геоинформацией. В версии «Pilgrim-5» геоинформационная подсистема работает в полном объеме. Однако конкретные электронные карты и Win-приложения для работы с геоинформацией создаются пользователем самостоятельно, либо поставляются по дополнительным соглашениям;

Pilgrim5 – содержит библиотеки, необходимые для работы с имитационными моделями, а также примеры моделей в виде скомпилированных cpp-файлов;

System Update – содержит приложение Setupsp5.exe, которое потребуется при инсталляции пакета «Pilgrim» в операционной системе Windows XP. Для операционных систем Windows 98/2000/Me/NT запуск данного приложения не требуется;

Visual C++ 6.0 for Pilgrim – дистрибутив для частичной инсталляции программной среды Microsoft Visual Studio с Visual C++ 6.0, специально адаптированной для работы пакета «Pilgrim».

Для установки программного обеспечения необходимо выполнить следующие действия:

1. Скопировать содержимое компакт-диска на жесткий диск компьютера. В свойствах скопированной папки Pilgrim-5-60 отменить атрибут «Только чтение». При дальнейших действиях компакт-диск не потребуется.

2. В папке «Visual C++ 6.0 for Pilgrim» запустить приложение Setup.exe. После соответствующего запроса ввести уникальный номер, отдельно прилагаемый к программному обеспечению. При запросе пути для папки Common указать сле-

дующий путь: **c:\Visual Studio\Common**. Далее инсталлятор автоматически выберет папку **c:\Visual Studio** для размещения всех остальных папок и файлов. Выбрать режим копирования файлов **Custom** с полной инсталляцией (**Select All**).

Замечание: Если при запросе оставить предложенный инсталлятором путь c:\Program Files\Microsoft Visual Studio\Common или указать иной путь, то не будет обеспечена совместимость с более ранними версиями системы.

3. При работе в Windows XP из папки «System Update» запустить приложение Setupsp5.exe для актуализации. После этого станут доступными диалоговая среда разработчика, библиотеки программ, интерфейсы ODBC, средства OLE и DLL, а также все средства Pilgrim-5.

4. Допустим, что пакет «Pilgrim-5» устанавливается на диск C. Войти в папку «Pilgrim5» и выполнить процедуру: **Setup.bat c:\Pilgrim-5-60 c:**

После этого на жесткий диск C будут скопированы последние актуальные модули из папки «C:\Pilgrim-5-60\Pilgrim5».

Замечание: Может потребоваться редактирование bat−файла **Setup.bat**, если имена путей и папок на рабочем компьютере отличаются от приведенных выше имен.

5. Папку «Gem» можно поместить далее в любое место файловой системы компьютера, а папки «System Update» и «Visual C++ for Pilgrim» можно удалить, освободив тем самым примерно 400 Мбайт.

В книге: А.А.Емельянов, Е.А.Власова, Р.В.Дума «Имитационное моделирование экономических процессов» М.:Финансы и статистика, 2002, имеются примеры для Microsoft Developer Studio (подраздел 4.5, стр. 147-159). Эти примеры полностью совместимы с Microsoft Visual Studio, если везде путь c:\Msdev заменить на путь c:\Visual Studio\Vc98. Библиотека функционального окна в Microsoft Visual Studio теперь называется MyWindow.dll (а не Mydll.dll – как это было для Developer Studio).

§6. Создание моделей с помощью графического конструктора GEM

Конструктор моделей GEM («Generator Models») позволяет автоматизировать процесс создания графа модели и автоматически генерировать программный код на языке C++. Таким образом, разработчик модели основное внимание может уделить не структуре и синтаксису программы, а структуре и параметрам самой модели и ее узлов. Кроме того, графический конструктор предупреждает разработчика о возможных ошибках и не позволяет выполнять заведомо неверные действия. Тем самым снимается ряд проблем, возникающих при ручном кодировании модели в виде программного файла.

Конструктор GEM состоит из программного файла gem.exe, файлов настроек с расширением «ini», файла помощи и примеров имитационных моделей. При запуске gem.exe перед разработчиком появляется основное окно, содержащее меню, панель «горячих кнопок», панель инструментов, информационную строку (рис.6.1.)

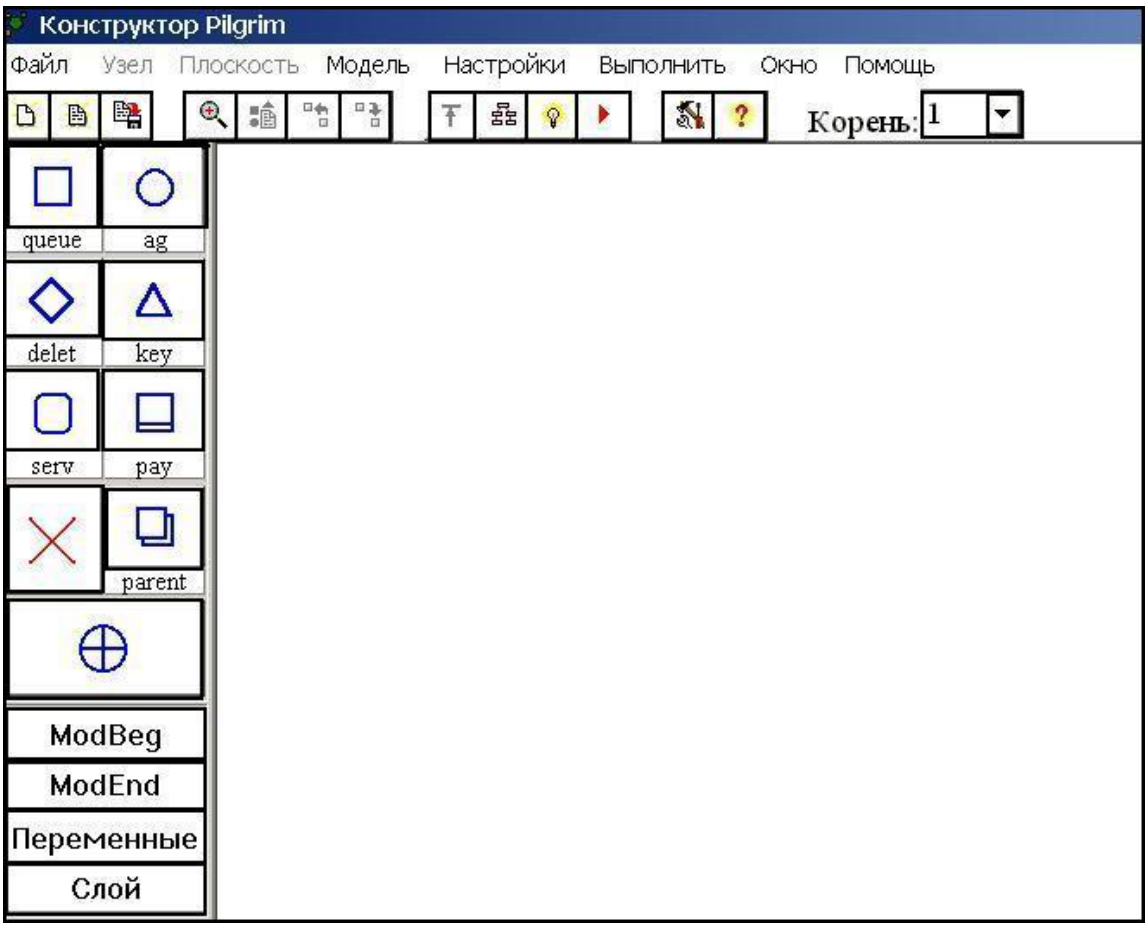


Рис.6.1 «Основное окно конструктора GEM с пустой областью построения графа»

Рабочая область пуста, для редактирования необходимо создать либо новую модель, либо загрузить ранее сохраненную.

Всю информацию о модели конструктор сохраняет в файле с расширением «pgf» (Pilgrim graf file). При создании законченной версии модели, разработчик может сгенерировать соответствующий модели программный файл с расширением «cpp», который подлежит дальнейшей компиляции в среде Visual C++ с подключением необходимых библиотек и ресурсов Pilgrim.

6.1. Редактирование графа модели

В основе любой имитационной модели лежит граф взаимодействия ее процессов. Выделим типовые действия редактирования графа модели:

– добавление новых узлов;

– перемещение узлов в области построения;

– удаление существующих узлов;

– определение маршрутов переходов транзактов, или ссылок.

В конструкторе все перечисленные действия выполняются «перетягиванием» объекта, обозначающего тип требуемого узла или действия, из панели инструментов в область построения графа. Панель инструментов в левой части окна редактора содержит значки, обозначающие узлы системы.

В панели инструментов находится набор значков всех типов узлов системы. Следует заметить, что для нескольких различных типов узлов может существовать одно и то же обозначение. Например, квадратом обозначаются узлы типа QUEUE, SEND и ATTACH. Поэтому под каждым значком с изображением узла находится кнопка с текущим типом создаваемых узлов данного обозначения. Для изменения типа достаточно щелкать мышью по кнопке с именем типа до появления необходимого.

Местоположение узла в плоскости построения графа не несет никакой смысловой нагрузки с точки зрения логики модели. Поэтому конструктор позволяет свободно перемещать узлы так, чтобы граф модели был удобен для восприятия. При перемещении узлов также используется технология перетягивания объектов.

Аналогичным образом выполняется удаление узла: в панели инструментов необходимо захватить значок удаления узла и перетащить его на узел, который требуется удалить. Узел будет удален вместе со всеми входящими и исходящими ссылками.

Дополнительные условия возникают при удалении узлов, содержащих переход на нижнюю плоскость. Например, если требуется удалить, узел типа PARENT, то конструктором проверяется условие пустой детализирующей плоскости. Алгоритм конструктора не позволит удалить узел типа PARENT, пока на детализирующей его плоскости находится хотя бы один узел, так как это приведет к его потере. Чтобы в подобной ситуации упростить удаление множества узлов на плоскости в меню «плоскость» существует специальный пункт «удалить все узлы текущей плоскости».

Несколько по-иному, чем для PARENT, реализовано разрешение на удаление узлов обращения к подпрограмме, т. е. PAY, RENT, DOWN. Если в модели имеется плоскость-подпрограмма для единственного узла PAY, RENT или DOWN, то его нельзя удалить, пока плоскость непустая (так же как в случае с PARENT). Если же к подпрограмме обращаются несколько узлов типа PAY, RENT или DOWN, то конструктор позволяет удалить любой из них, так как в этом случае плоскость- подпрограмма не теряется, поскольку продолжают существовать указывающие на нее ссылки. Таким образом, смысл условий удаления узлов, порождающих плоскости, крайне прост: необходимо, чтобы в модели не был потерян ни один узел.

Создание ссылок, или путей переходов транзактов, происходит следующим образом: в панели инструментов захватывается значок ⊕ и перемещается на узел-источник транзакта. При отпускании кнопки мыши за курсором потянется стрелка, обозначающая ссылку с невыбранным узлом-приемником транзакта. Для выбора узла-приемника необходимо щелкнуть по нему мышью, в результате чего создастся новая ссылка. Созданные ссылки отображаются на экране в виде дуг, имеющих три звена, служащих для удобства отображения графа на плоскости. Две промежуточные точки между звеньями выделены небольшими кругами, захватывая и перемещая которые можно придать стрелке нужную форму.

Следует помнить, что некоторые типы узлов не могут иметь входящие или, напротив, исходящие ссылки. Конструктором GEM постоянно выполняется проверка корректности действий пользователя, запрещающая заведомо недопустимые преобразования графа модели, в частности создание неправильных ссылок.

6.2. Определение параметров узла

Каждый узел модели характеризуется множеством параметров: типом, порядковым номером, именем, принадлежностью к плоскости, ссылками и условиями переходов, встроенным программным текстом, а также непосредственно пара-

метрами, определяемыми спецификой типа узла, такими, как закон распределения для узла типа SERV, приоритет для QUEUE и т.п.

Для просмотра или редактирования параметров узла необходимо дважды щелкнуть по нему левой кнопкой мыши либо один раз щелкнуть по узлу правой кнопкой, в результате чего отобразится всплывающее меню, и выбрать в нем пункт «параметры узла». Появится диалоговое окно, определяющее параметры. На рис.6.2 показано окно параметров узла типа SERV, номер 102, имеющего имя «Производство».

Поясним некоторые компоненты окна и способы работы с ними. Номер узла присваивается конструктором при его создании. Обычно первые 100 номеров зарезервированы для узлов SEND - бухгалтерских счетов. Нумерация создаваемых узлов осуществляется последовательно, с номера 101. Смена номера узла возможна с помощью соседнего с номером диалогового поля, но не рекомендуется для узлов, созданных ранее. Дело в том, что в тексте модели могут существовать привязки к номеру узла, созданные вручную. Конструктор не может их отследить, поэтому при перенумерации логика модели может нарушиться.

Поле «Имя» содержит имя узла, отображаемое на схеме и при выполнении модели. Поле доступно для редактирования. Не рекомендуется использовать имена, не умещающиеся в поле редактирования.

Класс узла может быть выбран из списка. В списке приводятся только те типы узлов, которые имеют одинаковое обозначение. Например, узел типа SEND можно сменить на ATTACH (при этом изменяется набор и смысл параметров узла).

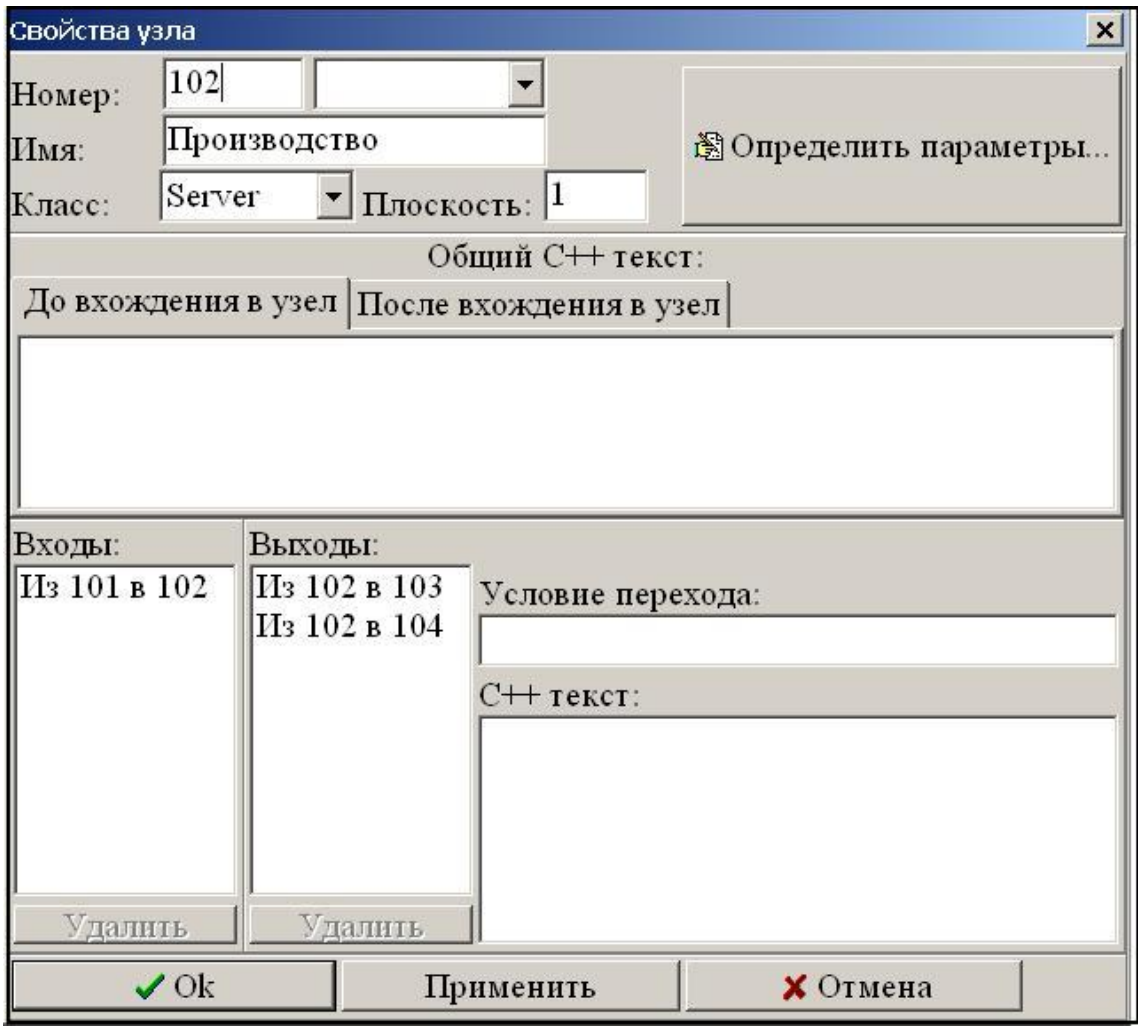


Рис.6.2. «Окно определения параметров узла»

Поле «Плоскость» показывает, к какой плоскости принадлежит узел, и доступно только для просмотра.

Дополнительная гибкость модели обеспечивается за счет использования вставок С++ кода. Панель «Общий С++ текст» позволяет пользователю включить в процедуру обработки узла произвольный текст на языке С++. Текст делится на две части: одна выполняется до вызова функции узла, другая - после нее. Смысл такого разбиения заключается в том, что программный текст, выполняющийся до вызова функции узла, может подготавливать какие-либо переменные, которые функцией будут использованы. Так, например, может быть подсчитано время обслуживания транзакта перед выполнением функции узла типа SERV. Программный текст, следующий после вызова функции узла, на ее выполнение уже никак не влияет и может использоваться для обработки параметров выполненной функции или подготовки параметров для других функций модели.

Узел может содержать несколько исходящих ссылок, по которым способен переместиться транзакт. Выбор маршрута должен осуществляться по условиям. Поэтому в правой нижней части окна определения параметров транзакта имеются поля, определяющие эти условия, а также выполнение соответствующих им индивидуальных фрагментов программного кода. Выбор исходящей ссылки осуществляется щелчком мыши в диалоговом поле «Выходы». Любой вход и выход можно удалить, нажав кнопку внизу списка.

Необходимым компонентом является кнопка «Определить параметры», нажатие на которую вызывает окно определения параметров самой функции обработки узла. Вид появляющегося диалогового окна приведен на рис.6.3.

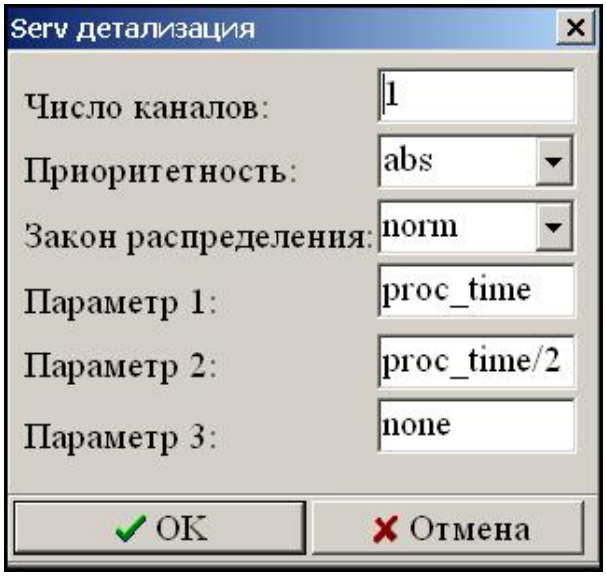


Рис.6.3 «Окно определения параметров функции для узла типа SERV»

Итак, рассмотрена схема определения параметров узла типа SERV. Аналогичным способом определяются параметры для узлов любого типа. При этом окно определения параметров функции узла имеет различный вид. Например, для узла типа QUEUE окно содержит настройку единственного параметра – признака приоритета прохождения транзактов, а для узлов типа TERM параметров функции узла и исходящих ссылок не существует вовсе.

6.3. Определение параметров инициализации и завершения модели

Модель имеет параметры инициализации и завершения, задаваемые функциями MODBEG и MODEND. Определение этих параметров производится через диалоговые окна, вызываемые нажатием кнопок «Modbeg» и «Modend» в основном окне редактора, либо выбором подпунктов основного меню в разделе «Модель».

Окно определения параметров функции MODBEG приведено на рис.6.4. В его правой верхней части записывается начальный текст на C++, если он необходим. Программный текст делится на две части: начальный C++ текст используется для подключения внешних библиотек и настройки глобальных параметров; текст инициализации ресурсов подготавливает параметры конкретных узлов типов ATTACH и SEND. Другие поля окна позволяют редактировать переменные, стандартные для функции MODBEG.

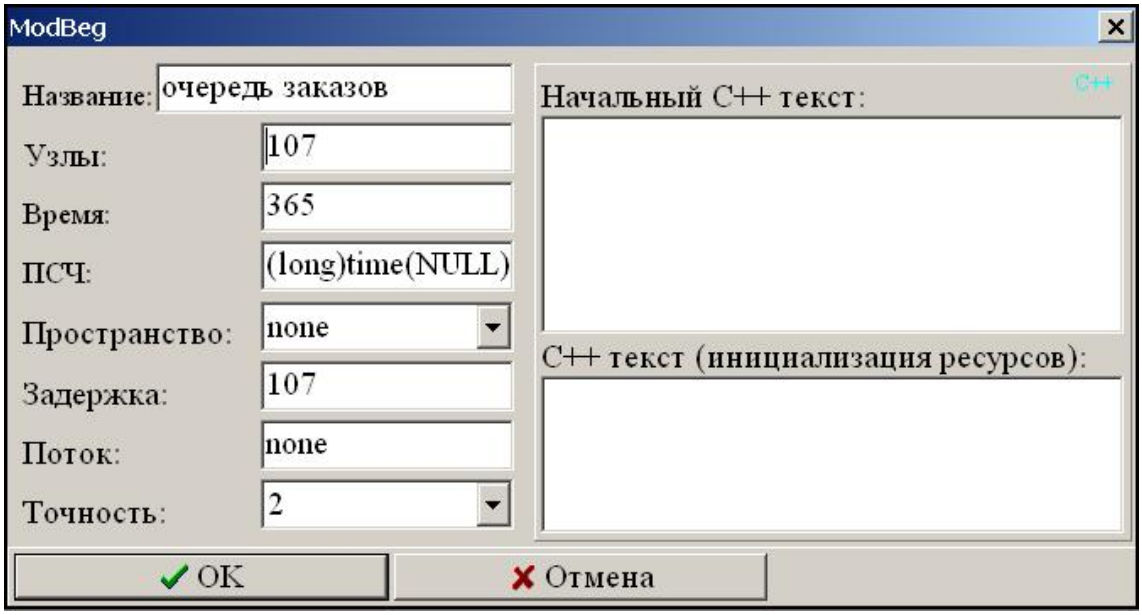


Рис.6.4. «Окно определения параметров инициализации модели»

Редактирование переменных функции MODEND осуществляется через диалоговое окно, приведенное на рис.6.5.

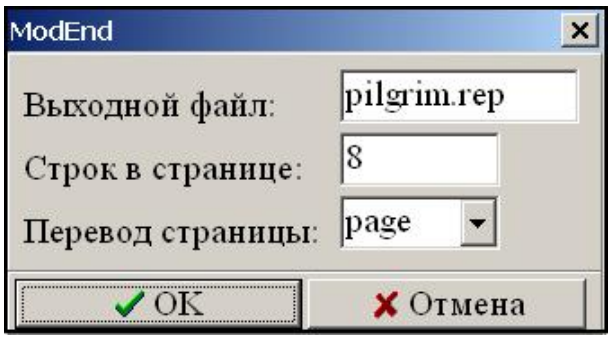


Рис.6.5. «Окно определения переменных функции MODEND»

6.4. Работа в плоскостях модели

При работе с большой моделью удобно пользоваться набором плоскостей построения. Для этого конструктор предлагает набор плоскостей с номерами 1 - 9, фрагменты графов которых не пересекаются на уровне маршрутов транзактов. В каждой из плоскостей могут находиться узлы типа PARENT, PAY, RENT, или DOWN, в свою очередь порождающие новые плоскости. Порождаемые плоскости имеют номера, начинающиеся с 10.

Возможны следующие операции перехода с уровня на уровень: переход на детализирующую плоскость, подъем на плоскость узла-родителя и переключение между корневыми плоскостями. Для каждой открытой плоскости конструктор предоставляет собственное окно построения.

Переключение между корневыми плоскостями осуществляется через верхнюю панель управления. В правой части панели расположено специальное выпадающее меню, позволяющее выбирать номер корневой плоскости. При этом в информационной строке появится надпись «Корень» и номер плоскости.

Для перехода на плоскость, детализирующую узел, необходимо дважды щелкнуть мышью по узлу-родителю. При этом в рабочей области экрана будет активизирована детализирующая плоскость, в информационной строке на панели «имя плоскости» появятся имя узла-родителя, а также номер текущей плоскости. Для плоскости необходимо задать вход и выход, как номера узлов, в которые попадает транзакт при переходе на плоскость из порождающего узла и из которого

происходит возврат на верхнюю плоскость. Узел-вход и узел-выход обозначены на графе входящей стрелкой из левого верхнего угла рабочей области и исходящей в правый верхний угол области соответственно. До тех пор пока вход и выход плоскости не определены, в информационной строке панели красным цветом будет выделено «Вход не назначен» и «Выход не назначен». Чтобы назначить вход и выход, необходимо на левой панели нажать кнопку «Слой». При этом будет выведено диалоговое окно (рис. 6.6).

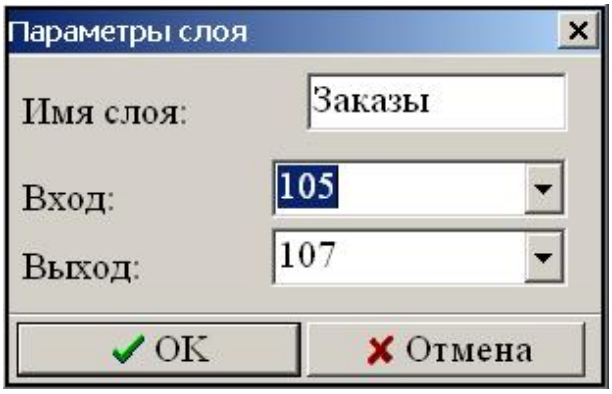


Рис.6.6. «Окно определения параметров плоскости»

Возврат на плоскость, которой принадлежит порождающий узел (т.е. на уровень выше), выполняется нажатием кнопки «подняться на уровень вверх» в панели «горячих кнопок» или выбором аналогичного подпункта основного меню из раздела «Уровень». Для удобства навигации в многослойной модели предусмотрено диалоговое окно, отображающее иерархическую зависимость между плоскостями: или узлами модели (рис. 6.7).

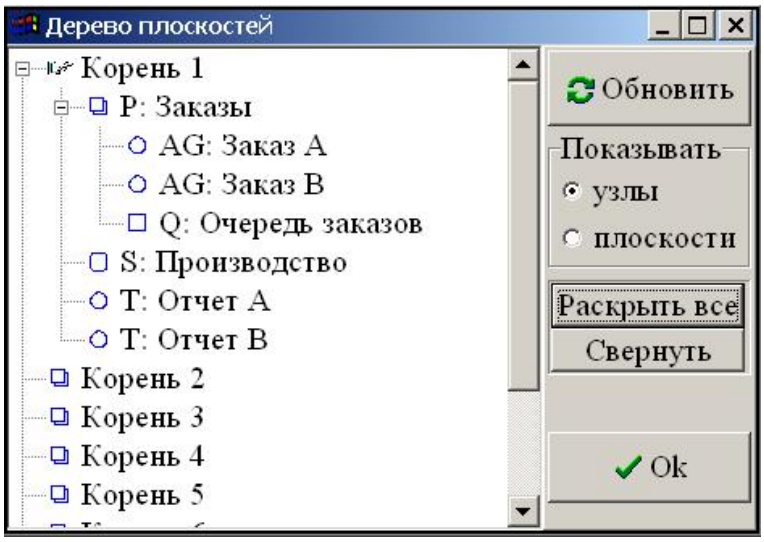


Рис.6.7. «Окно определения иерархии плоскостей модели»

6.5. Определение переменных модели

Конструктор обеспечивает пользователя простым блоком диалога, который позволяет задавать имя, тип и начальное значение переменных (рис. 6.8).

При вводе новой переменной обязательными параметрами являются имя и тип, значение пользователь может указывать по своему усмотрению.

Рис.6.8. «Окно описания переменных»

6.6. Дополнительные функции

Конструктор содержит ряд функций, позволяющих сделать работу пользователя более удобной и простой. Ниже приведен перечень сервисных функций, реализованных в конструкторе.

Изменение настроек экрана

Изменение настроек экрана выполняется через диалоговое окно, изображенное на рис.6.9 (основное меню, пункт «Настройки»).

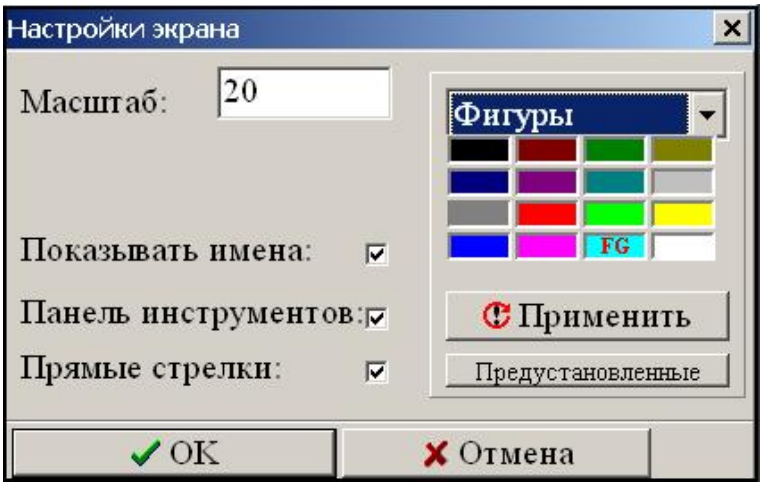


Рис.6.9. «Окно настройки экрана»

Пользователь-разработчик модели может по своему усмотрению изменять масштаб, цвет фигур и фона, показать или скрыть панель инструментов и установить признак прямых стрелок. Можно работать на мониторах с различным разрешением и размером диагонали, поэтому очевидна необходимость изменения масштаба в области построения графа (т.е. размеров фигур, обозначающих узлы, стрелок, надписей и других изображений). Механизм масштабирования дополняют линейки прокрутки, располагающиеся по бокам рабочей области построения графа.

Если признак прямых стрелок выключен, то при перемещении узла внутри плоскости промежуточные звеньевые точки стрелок, привязанных к нему, будут оставаться на месте, а при включенном признаке стрелки будут постоянно иметь прямой вид.

Проверка корректности модели

Конструктор проводит проверку двумя независимыми способами:

- не позволяет осуществлять заведомо ложные действия при редактировании пользователем модели;
- имеет возможность проверки графа в целом.

Результат проверки графа выдается в окне, приведенном на рис.6.10, как список ошибок возможных и ошибок явных, не позволяющих генерировать про-

граммный Pilgrim-файл. К явным ошибкам относятся отсутствие выходов или входов узла и неопределенные вход/выход плоскости.

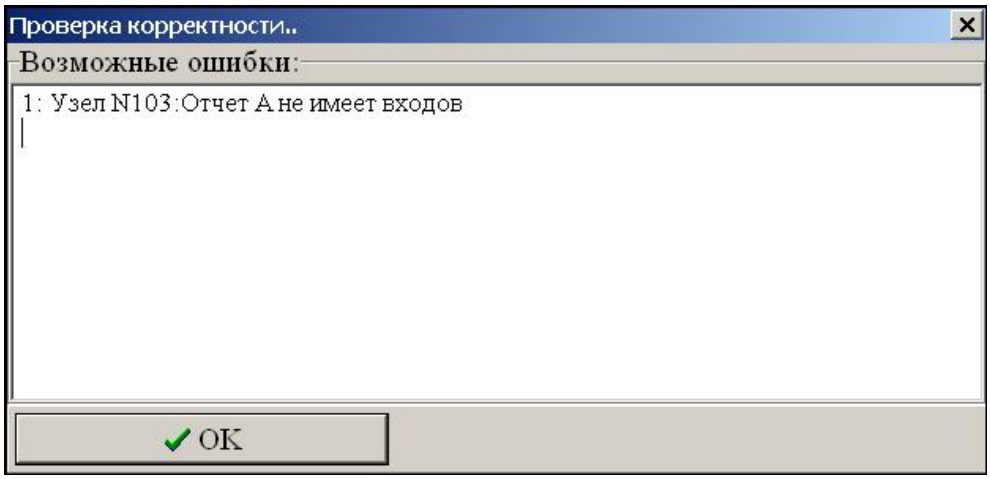


Рис.6.10. «Окно проверки корректности графа модели»

Процедура проверки корректности графа вызывается автоматически при запросе пользователя сгенерировать программный Pilgrim-файл, а также в любой момент времени из пункта «Выполнить» основного меню. При этом осуществляется проверка модели целиком, независимо от того, в какой плоскости работает пользователь.

План бухгалтерских счетов

При моделировании бухгалтерских проводок необходимо знать номер счета-приемника. Поскольку номер любого счета - это номер узла, его моделирующего, то для того, чтобы проще ориентироваться в их нумерации, существует информационное окно, отображающее номера и названия счетов. Это окно может быть вызвано выбором подпункта «План счетов» пункта «Модель» основного меню программы.

Поиск узла

Функция вызывается через меню «Узел» либо через «горячую кнопку» на верхней панели (рис.6.11). Поиск может осуществляться по номеру или по параметрам узла, таким, как имя или тип.

Результаты поиска выводятся в виде списка, откуда можно перейти на любой узел, редактировать его параметры или скопировать узел в буфер.

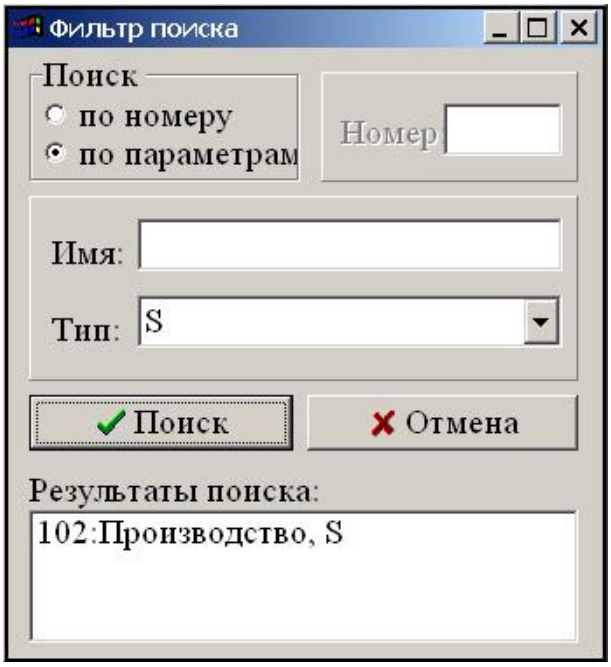


Рис.6.11. «Окно поиска узла»

Копирование/вставка узла

Для упрощения создания набора узлов с рядом совпадающих параметров в конструкторе предусмотрена функция копирования/вставки узла. Чтобы скопировать узел в буфер, необходимо выделить его щелчком мыши и выбрать пункт «Копировать» из меню «Узел». Вставка в текущую плоскость осуществляется выбором пункта «Вставить» меню «Узел». При этом будет создан узел того же типа, что и прототип, с аналогичными параметрами.

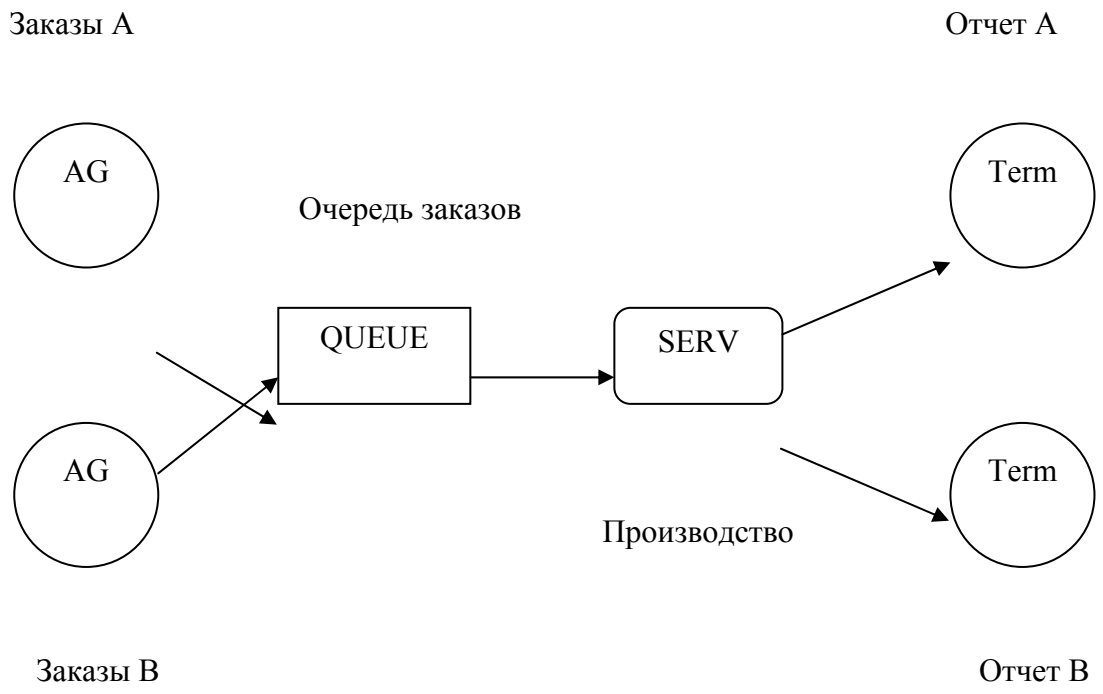
Очистить плоскость

Функция удаляет все узлы текущей плоскости. Вызывается через меню «плоскость». Операция недопустима и выдаст сообщение об ошибке в случае, если хотя бы один из узлов плоскости содержит декомпозицию.

§7. Пример создания имитационной модели экономического процесса

В качестве примера, иллюстрирующего создание моделей с помощью конструктора, рассмотрим следующий объект. Имеется предприятие, принимающее заказы на изготовление продукции типов А и В. Заказы на продукцию типа А поступают в среднем раз в 30 дней, а на продукцию типа В - раз в 5 дней. Для производства продукции типа А необходимо 10 дней, а для продукции типа В - 2 дня. Одновременно может идти производство только по одному заказу. При этом заказы на продукцию А являются первоочередными. Все сведения о выполнении заказов типов А и В фиксируются в отдельных документах. Модель строится для исследования очереди задержек заказов.

Схематично модель можно представить следующим образом:



Необходимо заметить, что модель является достаточно простой и легко может быть построена в одной плоскости конструктора. Но для демонстрации механизма построения многоуровневых моделей создадим двухуровневую модель.

Опишем создание имитационной модели с помощью графического конструктора как последовательность технологических шагов.

Шаг I. Создание графа модели.

1. Рассмотрим в качестве автономных процессов:

- производство и фиксацию выполненных заказов;
- формирование заказов и очередь заказов на производство.

2. Определим единицу модельного времени для последующего формирования параметров модели. Очевидно, в данном примере за единицу времени удобно взять один рабочий день.

3. Запустим конструктор GEM на выполнение.

4. В основном меню выберем «Файл» → «Создать». В результате, в рабочей области экрана появится пустое окно с именем «Плоскость 1». Это пустая корневая плоскость.

5. Из панели узлов в левой части экрана перетащим в плоскость построения узлы типов PARENT и SERV. В панели узлов также найдем узел типа AG и щелкнем мышью по надписи под узлом AG. Надпись на панели сменится на TERM. Добавим в плоскость построения два узла типа TERM. Расположим узлы в рабочей области так, как показано на рис.7.1.

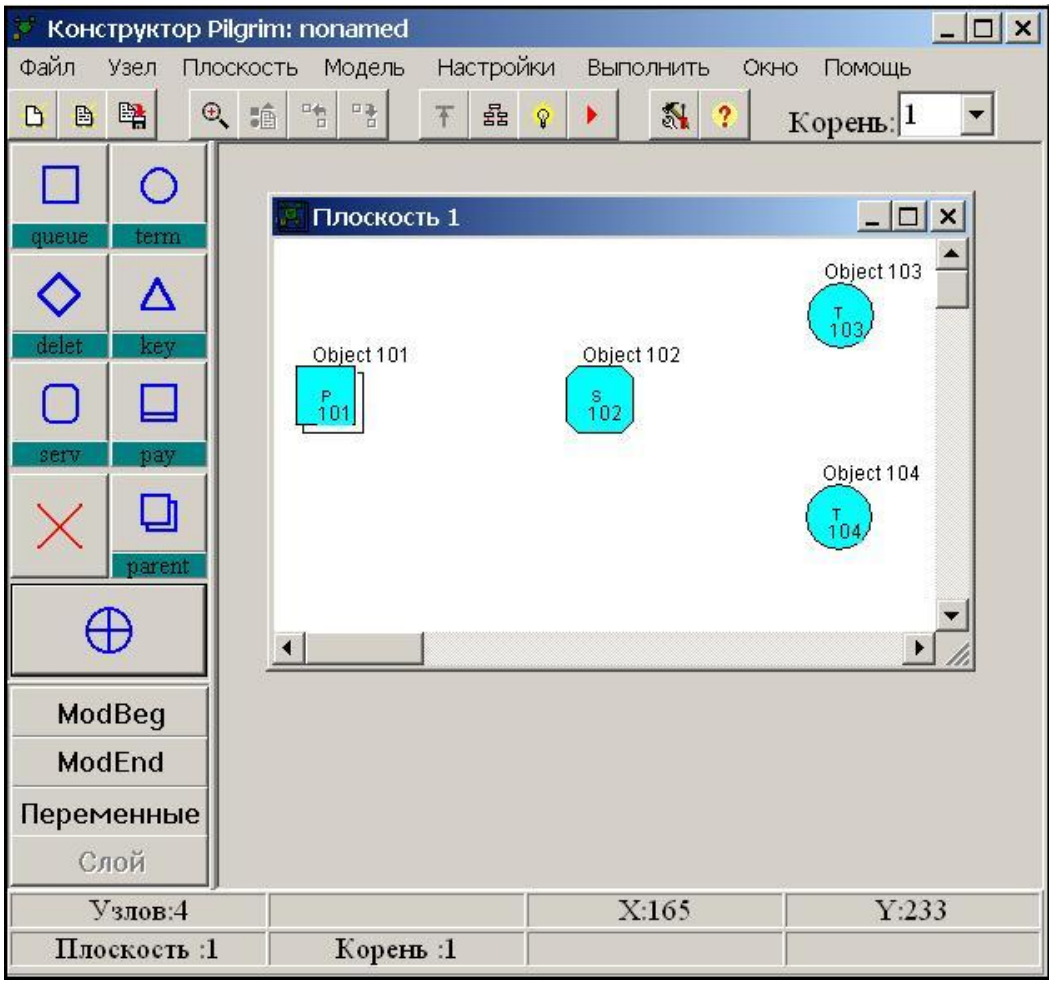


Рис.7.1. «Окно создания узлов в корневой плоскости»

6. Зададим имена узлов. Для этого нужно дважды щелкнуть мышью по каждому из узлов (за исключением узла 101) и в появившемся диалоговом окне в поле «Имя» вместо находящегося там «Object...» ввести: «Производство» для узла 102; «Отчет А» для узла 103, «Отчет В» для узла 104. Узел 101 будет содержать подуровень формирования заказов и будет рассмотрен позже.

7. Определим маршруты транзактов. Для этого захватим изображение круга с перекрестием ⊕ на панели узлов и перетащим на узел «Производство». После отпускания кнопки мыши за курсором потянется стрелка. Далее необходимо щелкнуть мышью по узлу «Отчет А», в результате чего появится стрелка, символизирующая маршрут прохождения транзактов из узла «Производство» в «Отчет А». Аналогичную операцию повторим, задав маршруты из узла «Производство» в «Отчет В» и из «Object 101» в «Производство».

Замечание: Необходимо сначала указывать узел- источник и потом - узел-приемник. Последующая смена направления невозможна. При создании неверного маршрута необходимо его удалить через свойства узла и потом задать новый.

По окончании этого этапа должен появиться экран, подобный приведенному на рис. 7.2. При коротких дугах между узлами указатели стрелок могут не отображаться, однако это не влияет на суть модели.

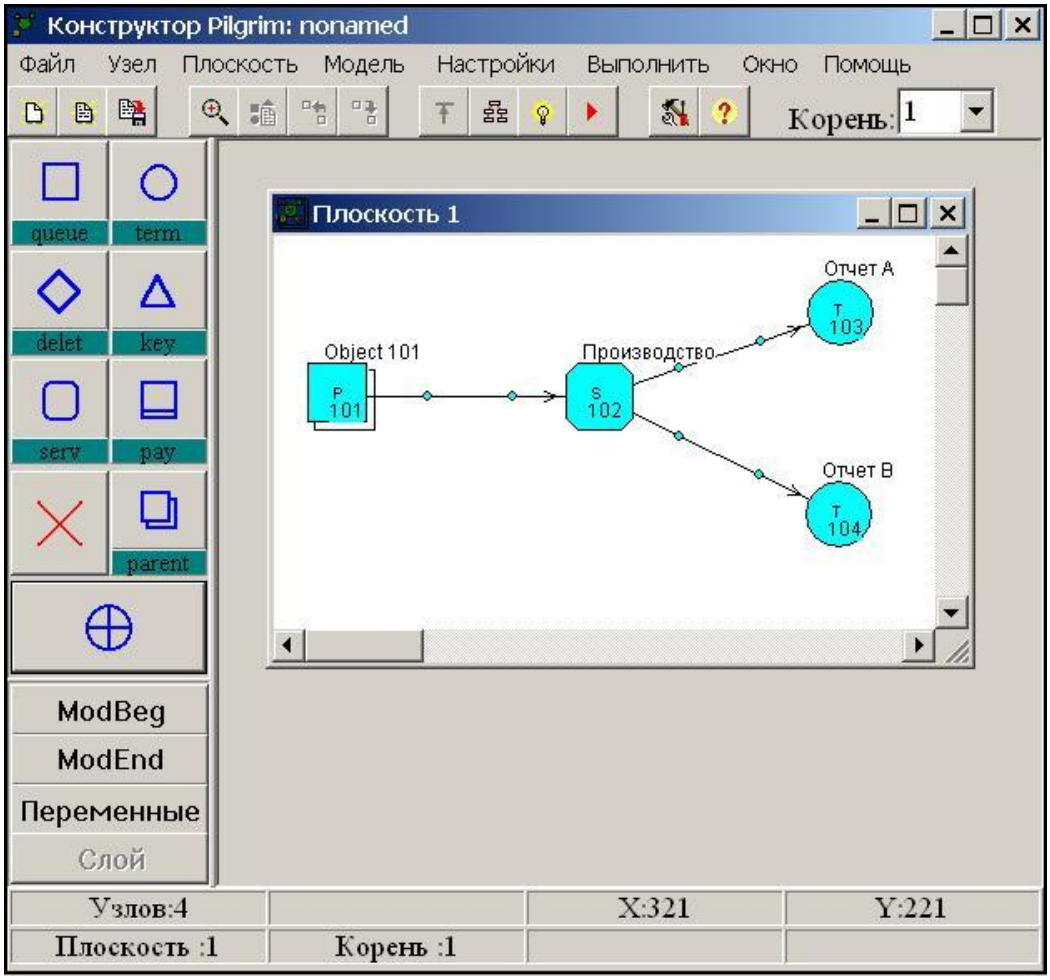


Рис.7.2. «Окно графа корневой плоскости»

8. Создадим плоскость формирования заявок. Дважды щелкнем по узлу «Object 101», в результате чего получим новую пустую плоскость 10 с именем «Object 101». В этой плоскости теми же операциями, что и ранее, создадим граф, как показано на рис. 7.3. Останемся в этой плоскости.

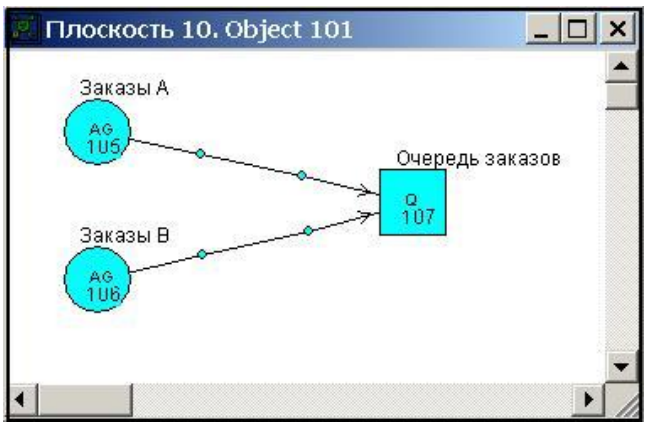


Рис.7.3. «Окно графа плоскости 10»

9. Можно увидеть, что в информационной строке в нижней части экрана красным цветом отображены надписи «Вход не назначен» и «Выход не назначен». В нашем случае плоскость 10 не будет иметь входа, так как сама предназначена для генерации заявок, однако выход необходимо назначить. В левой части экрана щелкнем по кнопке «Слой». В диалоговом окне введем имя слоя «Заказы» и выход 107. Вход установим на значение «none», т.е. отсутствующий. Результат приведен на рис. 7.4. После нажатия кнопки «ОК» одна из красных надписей в информационной строке будет заменена на «Выход: 107».

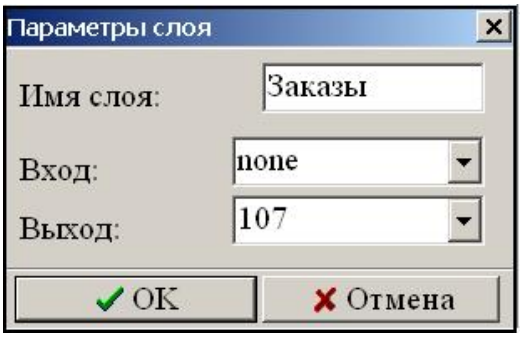


Рис.7.4. «Окно определения параметров слоя 10»

Шаг II. Определение глобальных переменных модели. Определим переменные модели. Очевидно, для работы может понадобиться переменная времени обслуживания заявки. На левой панели щелкнем по кнопке «Переменные». Появится соответствующее диалоговое окно (рис. 7.5). В нем необходимо ввести: имя пе-

ременной proc_time, тип переменной float. Значение можно не указывать, так как оно будет определяться при выполнении модели в зависимости от типа продукции.

Рис.7.5 «Окно определения глобальных переменных модели»

Шаг III. Определение свойств узлов.

1. Определим свойства узлов плоскости 10. Узел 105 будет формировать заказы типа А, узел 106 - типа В. Необходимо дважды щелкнуть мышью по узлу 105, в результате чего появится диалоговое окно «Свойства узла». В нем щелкнуть по кнопке «Определить параметры». В результате появится окно определения параметров узлов типа AG (рис. 7.6). Очевидно, для заявок типа А значение «Приоритет» нужно указать равным 1. На выборе закона распределения не будем останавливаться, выберем его как norm - нормальное. В строке «Параметр 1» введем значение математического ожидания 30, а строке «Параметр 2» примем значение отклонения равным 10. Повторим процедуру для узла 106. При этом значение приоритета укажем 2, математическое ожидание примем равным 5, а отклонение - равным 2. Теперь генераторы подготовлены.

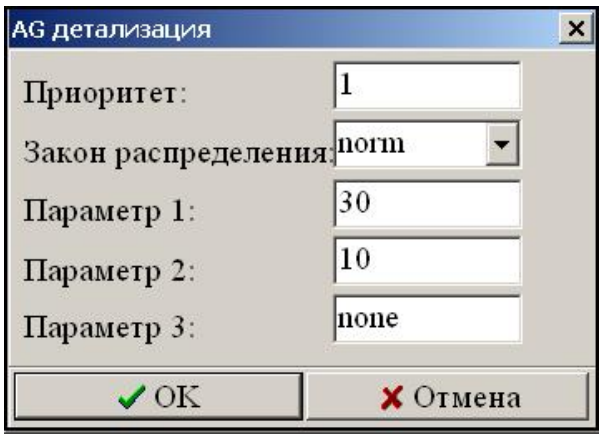


Рис.7.6 «Окно определения переменных узла типа AG Заказы А»

2. Аналогичным образом определим параметры для узла 107 «Очередь заявок». Здесь всего один параметр, его необходимо установить в значение «prty». Это будет означать, заказы проходят очередь в соответствии с приоритетами. Заявки на производство продукции типа А будут проходить прежде заявок на продукцию типа В.

3. Определим свойства узлов плоскости 1. Начнем с узла 102 типа SERV.

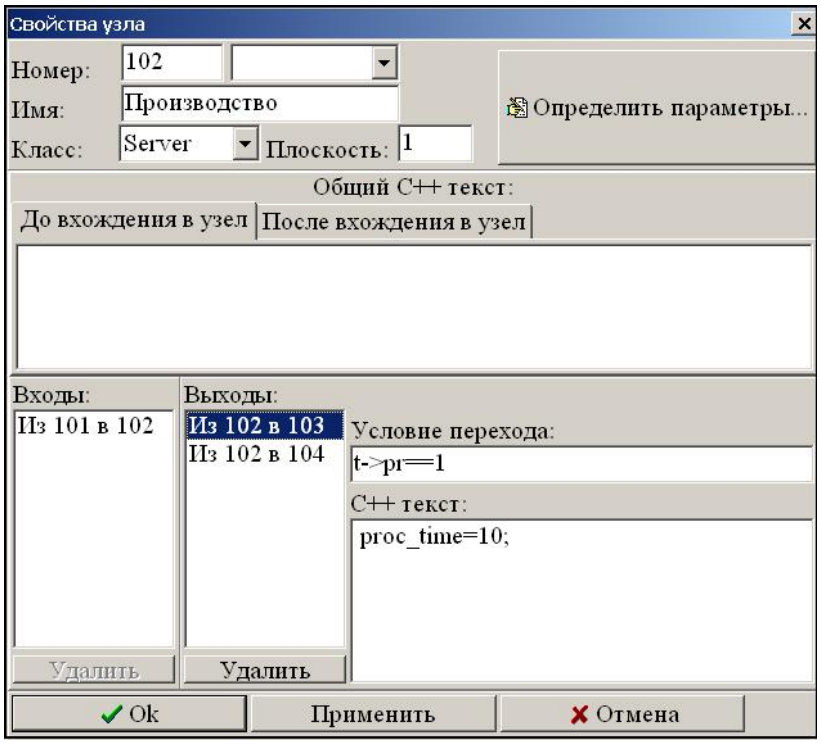


Рис.7.7 «Окно определения общих свойств узла 102 Производство»

Дважды щелкнем по узлу мышью - появится диалоговое окно «Свойства узла» (рис. 7.7). В поле «Выходы» выберем строку «Из 102 в 103». В поле «Условие перехода» укажем «t→pr ==1», Это значит, что по этой ссылке перейдут только те транзакты, которые в качестве параметра t→pr (приоритет) имеют значение 1. В поле «С++ текст», располагающееся под «Условием перехода», введем строку «proc_time = 10;». Далее, выбрав в поле «Выходы» строку» Из 102 в 104», в поле «С++ текст» напишем «proc time = 2;». Щелкнем по кнопке «Определить параметры». В появившемся диалоговом окне «Serv детализация» зададим параметры: число каналов = 1; приоритетность = abs; закон распределения = norm; параметр 1 (математическое ожидание) = proc_time; параметр 2 (отклонение) = proc_time/2; параметр 3 (дополнительно) = none (рис. 7.8). Таким образом, определен обслуживающий прибор с одним каналом и временем обслуживания, зависящим от приоритета транзакта.

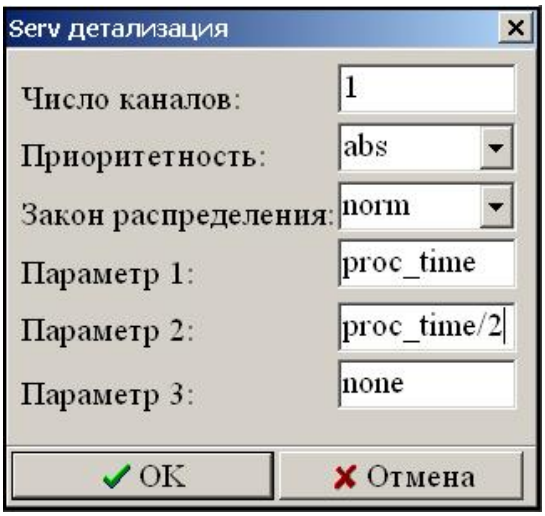


Рис.7.8 «Окно определения переменных функции SERV для узла Производство»

4. Для узлов типа TERM параметров не существует.

Шаг IV. Определение параметров функций MODBEG и MODEND.

1. Определим параметры функции MODBEG. В левой панели инструментов щелкнем по кнопке MODBEG, либо выберем соответствующий пункт меню раздела «Модель». В появившемся диалоговом окне зададим имя модели «Очередь заказов». Параметр «Узлы» обозначает порядковый номер последнего

узла модели и рассчитывается автоматически. Поле «Время» подразумевает время выполнения модели в моделируемых единицах времени. Поскольку за единицу времени взят один день, то рассмотрим модель на отрезке времени 365 единиц, т.е. один год. Параметр «ПСЧ» служит для настройки датчика случайных чисел. Значение, отображенное по умолчанию, позволяет при каждом запуске получать различные результаты моделирования. Поле «Пространство» используется при пространственном моделировании и в нашем случае не понадобится. В поле «Задержка» необходимо ввести номер узла типа QUEUE, информация о котором будет отображаться при выполнении модели, т.е. в нашем случае это номер 107. Поле «Поток» и «Точность» оставим без изменений. Результат определения параметров в диалоговом окне приведен на рис.7.9.

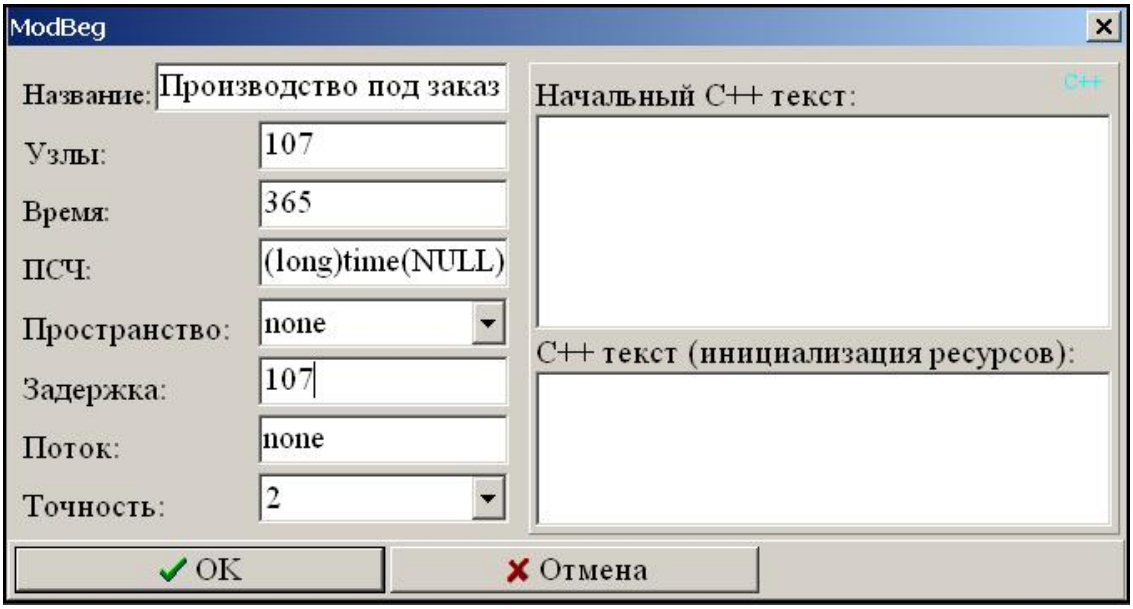


Рис.7.9 «Окно определения параметров функции MODBEG»

2. Определим параметры функции MODEND. Для нашей задачи оставим параметры MODEND, заданные по умолчанию. Вызов диалогового окна для ввода параметров осуществляется аналогично вызову окна MODBEG.

Шаг V. Генерация текста имитационной модели в операторах Pilgrim.

Итак, модель готова. Для нее сформирован граф, заданы переменные, определены параметры узлов и функций. Для генерации программного файла необходимо в основном меню выбрать «Выполнить», а затем - «Генерировать C++

файл». При этом конструктором сначала будет выполнена проверка модели, в нашем случае не обнаружившая никаких ошибок или подозрительных участков на графе модели. После нажатия кнопки ОК будет выведено стандартное диалоговое окно, предлагающее сохранить файл с расширением «cpp». Сохраним файл под именем PRODUCT. Файл «product.cpp» можно далее компилировать в среде Visual C++.

Программная модель, автоматически сгенерированная конструктором и помещенная в cpp-файл, имеет следующий вид:

```
#include <Pilgrim.h>
 forward
{
 int fw;
 int proc_time;
 modbeg ("очередь заказов", 107, 365, (long)time(NULL), none, 107, none,none, 2);
 ag ("Заказ A",  105, 1, norm, 30, 10, none, 107);
 ag ("Заказ B",  106, 2, norm, 2, 5, none, 107);
 network (dummy, dummy)
 {
  top (102):
       if ( t->pr ==1 )
            {
             proc_time = 10;
             fw=103;
            }
           else
            {
             proc_time = 2;
             fw=104;
            }
   serv ("Производство", 1, abs, norm, proc_time, proc_time/2, none, fw);
            place;
   top (103):
       term ("Отчет A");
            place;
```

```
  top (104):
      term ("Отчет B");
          place;
  top (107):
      queue ("Очередь заказов", prty, 102);
          place;
  fault (123);
 }
 modend ("pilgrim.rep", 1, 8, page);
 return 0;
}
```

§8. Обработка имитационной модели в системе Pilgrim

8.1. Формирование программного файла модели

Рассмотрим процесс запуска модели на примере построенной модели PRODUCT:

1. Запустить приложение *Microsoft Visual C++*.
2. В меню файл выбрать *New*, далее вкладку *Projects* и *Win32 Application.*

   В строке *Project name* указать название проекта (например, *«Production»*), а в строке *Location* указать путь к проекту, например, *C:\Pilgrim-5-60\Pilgrim5\Samples*. Нажать *OK.*

   После появления окна с запросом «*Какой вид приложения вы хотите создать?»* выбрать *An empty project*. Нажать *Finish* и еще раз *OK*. Таким образом, создана рабочая область *Production.dsw* для проекта.
3. Скопировать из папки *C:\Pilgrim-5-60\Pilgrim5\Projec*ts в папку *C:\Pilgrim-5-60\Pilgrim5\Samples\Production* следующие четыре файла: *Palette.bmp, Pilgrim.ico, UserHid.h, UserRes.rc.*
4. В меню выбрать *Project -> Add To Project ->Files* и добавить в проект эти четыре файла и файл *Product.cpp*, сгенерированный конструктором GEM.

Таким же образом добавить файлы *Comctl32.lib, Pilgrim.lib* из папки *C:\Visual Studio\VC98\Lib.*

5. Выбрать в меню *Build ->Build Production.exe.* После этого, в случае отсутствия ошибок, в папке *C:\Pilgrim-5-60\Pilgrim5\Samples\ Production \Debug* сформируется файл *Production.exe.*
6. Запустить полученную модель *Production.exe*.

После запуска имитационной модели в папке *C:\Pilgrim-5-60\Pilgrim5\Samples\ Production \Debug* сформируется текстовый файл результатов с именем, указанным в тексте модели. В нашем примере это файл «Pilgrim.rep». Для просмотра файла результатов можно использовать любой текстовый редактор.

8.2. Трактовка результатов моделирования

Файл результатов представляет собой таблицу, содержащую в зависимости от параметров модели до 12 столбцов:

1) *№ узла.* Указаны номера узлов модели;

2) *Наименование узла.* Смысловое название узла в модели, присвоенное разработчиком;

3) *Тип узла.* Указан тип узла: AG, SERV, KEY, QUEUE, TERM, CREAT, DELET, PROC, DYNAM, SEND, DIRECT, ATTACH, MANAGE, PAY, RENT, DOWN или PARENT;

4) *Точка.* Номер последней точки пространства, в которой находится узел типа CREAT, DELET или PROC в момент окончания моделирования;

5) *Загрузка (%=).* Коэффициент использования транзактами узлов типа SERV или PROC в процентах. Для узла типа KEY – доля времени пребывания в закрытом состоянии;

6) *Путь (км).* Если производятся пространственные перемещения узлов типа PROC, CREAT или DELET, то подсчитывается пройденный путь. Для пространства типа GEO путь считается в километрах;

7) *M [t] среднее время.* Среднее значение времени задержки транзакта в узле или иной интервал времени, зависящий от типа узла:

- для SERV - это среднее время пребывания в узле (оно может быть больше времени обслуживания у неприоритетных транзактов при p_3=abs, т.е. при наличии приоритетных транзактов и правила абсолютных приоритетов);
- для QUEUE - среднее время задержки в очереди;
- для AG - среднее время между двумя сгенерированными транзактами;
- для TERM или DELET - среднее время существования транзакта;
- для KEY - среднее время пребывания в закрытом состоянии;
- для CREAT и DYNAM - всегда нулевое значение;
- для PROC при p_4=none, p_4=norm, p_4=expo или p_4=unif - среднее время пребывания в узле (оно может быть больше времени обслуживания транзакта при переводе узла в пассивное состояние);
- для PROC при p_4=earth, p_4=plane или p_4=cosmos -суммарное время пребывания транзакта в узлах DYNAM и PROC с учетом возможных возвратов транзактов из PROC в DYNAM.

8) *C^2 [t] квадрат коэффициента вариации* – отношение дисперсии временного интервала к квадрату его среднего значения;

9) *Счетчик входов и hold*. Число транзактов:

- прошедших через узел;
- сгенерированных транзактов (для AG или CREAT);
- уничтоженных (для TERM или DELET);
- выполнивших операцию HOLD из другого узла в отношении узла KEY;

10) *Количество каналов.* Число каналов в узле;

11) *Осталось транзактов.* Количество транзактов, которые остались в узле на момент завершения моделирования;

12) *Состояние узла в этот момент.* Состояние узла в момент окончания прогона модели:

- узел может быть открыт (свободен), закрыт для входа очередного транзакта, активен или пассивен (узел PROC);
- В узле типа SEND может быть положительное сальдо на момент завершения модели (денежная сумма с буквой S) и отрицательное сальдо (сумма с буквой D).
- В узле типа ATTACH может быть остаток ресурса на момент завершения модели (денежная сумма с буквой S) и дефицит (сумма с буквой D).
- В узлах типа PAY и RENT указывается количество переходов транзактов на нижние уровни - слои модели (на момент завершения модели).

Пример: В результате прогона модели «*Production*» был получена следующая таблица результатов:

НАЗВАНИЕ МОДЕЛИ: очередь заказов ВРЕМЯ МОДЕЛИРОВАНИЯ: 365.36										
№ узла	Наименование узла	Тип узла	Точка	Загрузка(%=), Путь (км)	M [t] среднее время	C^2 [t] квадрат коэф. вариац.	Счетчик входов и hold	Кол. кан.	Ост. тр.	Сост. узла
102	Производство	serv	-	%= 59.0	3.78	2.39	110	1	1	закрыт
103	Отчет А	term	-	-	28.43	0.13	10	0	0	открыт
104	Отчет В	term	-	-	4.85	0.73	99	0	0	открыт
105	Заказ А	ag	-	-	27.22	0.15	14	1	1	открыт
106	Заказ В	ag	-	-	3.59	1.08	100	1	1	открыт
107	Очередь	queue	-	-	3.60	2.78	113	1	3	от-

	заказов									крыт

В таблице результатов прогона модели мы видим статистические данные о производстве за период, равный 365 дней. За это время средняя загрузка производственных мощностей составила 59% (загрузка узла 102). Среднее время выполнения заказа – почти 4 дня (M[t] узла 102) при среднем времени ожидания заказа очереди 3.60 дня (M[t] узла 107). Довольно большое время ожидания связано с тем, что заказы А выполняются в течение длительного промежутка времени, вследствие чего образуются очереди.

Среднее время пребывания заказа А от момента его поступления на предприятие до выполнения составило 28.43 дней (M[t] узла 103) при небольшой вариации (C^2[t] узла 103). Среднее время пребывания заказа В – 4.85 дней.

В течение рассматриваемого промежутка поступило 14 заказов А (счетчик входов узла 105) и 100 заказов В (счетчик входов узла 106), из которых выполнено соответственно 10 (счетчик входов узла 103) и 99 (счетчик входов узла 104). Один заказ А остался в процессе исполнения в узле SERV (остаток транзактов узла 102), три заказа остались в очереди на выполнение.

8.3. Трассировка модели

Часто синтаксически правильная модель выдает неправильные результаты, что свидетельствует либо о неадекватности модели, либо о логической (семантической) ошибке в ее описании. Для отладки модели можно использовать режим трассировки. В данный режим модель переводится с помощью меню модели или панели ее инструментов. Можно запустить модель сразу в режиме трассировки. Диалоговое окно трассировки по умолчанию изменяется после каждого события. Учитывая, что число событий может быть очень велико, существуют следующие режимы трассировки, позволяющие ускорить процесс отладки:

- выйти в режим трассировки после конкретного события;

- перейти в трассировку, если какой-то транзакт входит в определенный узел или выходит из него;
- отслеживать путь конкретного транзакта по сложному графу модели;
- выйти в режим трассировки по показанию модельного таймера.

Управление трассировкой осуществляется из общего меню модельного окна.

§9. Развитие пользовательского интерфейса в имитационных моделях

9.1. Проектирование моделей с диалоговым окном для управления параметрами

Выше мы рассмотрели процесс создания имитационной модели с помощью графического конструктора GEM. При этом числовые параметры модели определялись в параметрах узлов. Изменение значений этих параметров предполагает либо корректировку графа модели с дальнейшей компиляцией и созданием нового cpp-файла, либо ручное внесение правок в ранее скомпилированный cpp-файл.

Следует отметить, что для создания таких моделей от разработчика не требуется глубоких знаний языков программирования, поскольку он может обойтись средствами графического конструктора GEM и готовым алгоритмом сбора и прогона модели.

Более удобными с пользовательской точки зрения являются модели, в которых параметры можно переопределять в диалоговом окне. Проектирование таких моделей требует от разработчика знаний Visual C++.

Рассмотрим следующую задачу. Имеется остановка маршрутного такси. С определенными интервалами времени на остановку приходят пассажиры и подъезжают такси. Такси уезжает, когда в него сядет 10 пассажиров. Если пришедший пассажир не обнаруживает стоящего такси, он встает в очередь на посадку. Если

такси подъехало на пустую остановку, оно ждет пассажиров. Если такси подъехало, а на остановке уже стоит такси, оно встает в очередь такси.

Схема модели выглядит следующим образом:

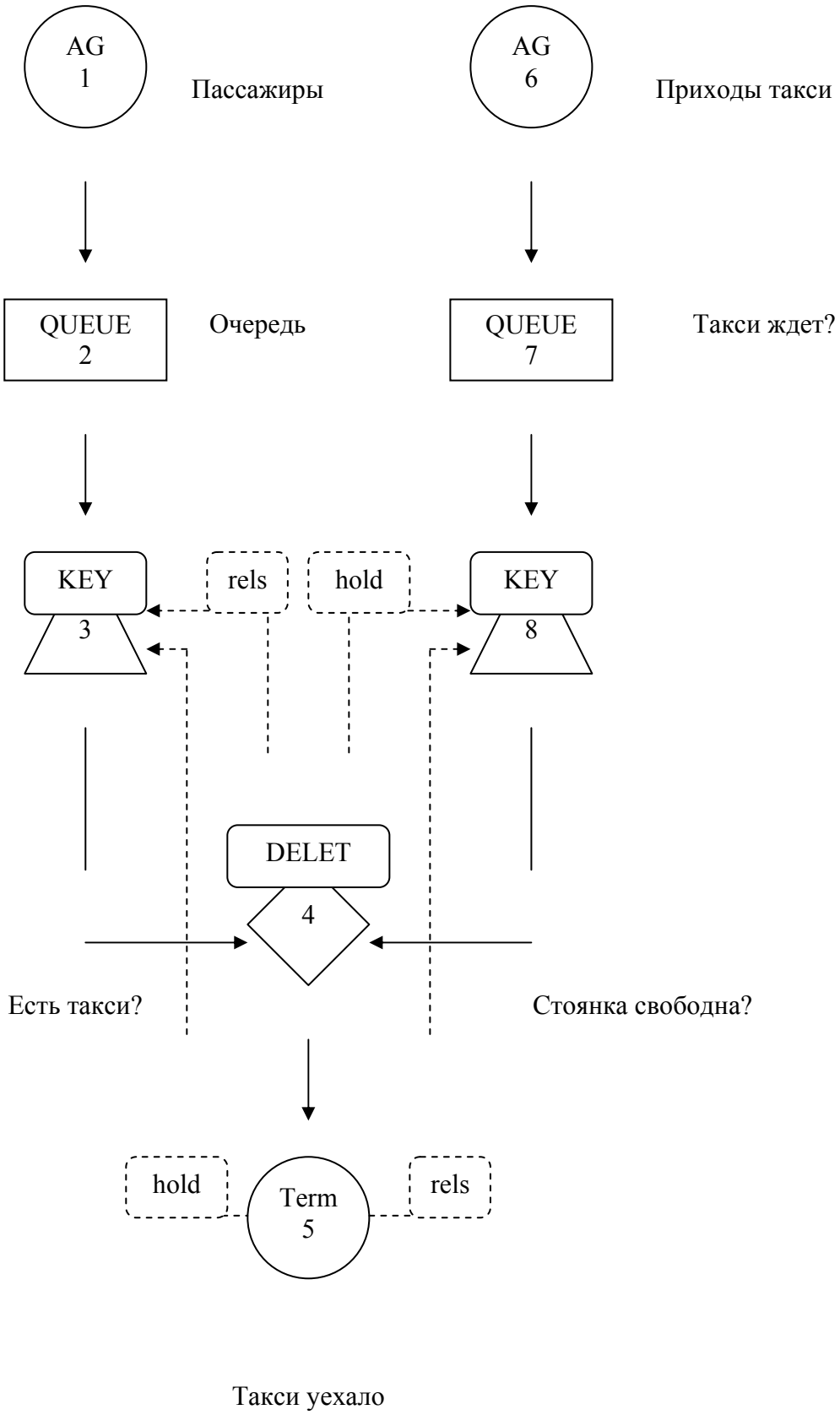


Рис.9.1 «Схема модели взаимного исключения и координации процесса такси-пассажиры»

Данная задача иллюстрирует принцип работы узла DELET: первый вошедший в него транзакт становится как бы тележкой, в которую собираются транзакты, входящие следом. Как только в узле наберется заданное количество транзактов, они уничтожаются, а первый переходит в следующий узел: тележка уезжает, а предметы, собранные в нее, перестают существовать как заявки на обслуживание в системе.

Генератор 1 создает транзакты, имитирующие пассажиров. Здесь работает теорема о суперпозиции потоков случайных событий. Событием является приход пассажира на остановку. Потоком событий является последовательность приходов на остановку каждого конкретного пассажира. Интервалы между приходами являются случайными числами. Если эти потоки для очень большого числа N пассажиров спроектировать на общую ось времени, то интервал между двумя соседними событиями (приходами пассажиров на остановку), возможно принадлежащих разным потокам, будет являться случайной величиной, распределение которой подчиняется **экспоненциальному** закону.

Интервал генерации транзактов, имитирующих такси, имеет нормальное распределение, если предполагать, что остановку обслуживают такси из одного парка, а не нескольких. Использование в описании генератора 6 **нормального** закона распределения интервала генерации означает, что время между приездами такси на остановку чаще оказывается ближе к своему среднему значению и реже - дальше от него (чем больше отклонение интервала от среднего, тем реже это бывает).

Узлы 2 и 7 имитируют соответственно очереди пассажиров и такси. Ключ 8 в начале работы модели находится в открытом состоянии (по умолчанию), а ключ 3 закрывается при приходе в очередь 2 первого транзакта. Это делается для того, чтобы в узел DELET первым вошел транзакт-такси, а не пассажир (иначе пассажир станет "тележкой").

Как только в DELET войдет первый транзакт-такси, ключ 8 закрывается, а ключ 3 открывается. Теперь выходящие из генератора 6 транзакты-такси стоят в очереди 7, а транзакты-пассажиры заходят в узел DELET (идет заполнение такси). Когда в узле накопится 10 транзактов-пассажиров, транзакт-такси перейдет в терминатор (заполненное такси уедет). Ключ 8 открывается, чтобы в DELET мог зайти следующий транзакт-такси. Ключ 3 закрывается, чтобы транзакты-пассажиры стояли в очереди 2 до прихода в DELET транзакта-такси.

Таким образом, в модели ключи 3 и 8 всегда находятся в противоположных состояниях - если один открыт, то другой закрыт. Тем самым чередуется доступ к узлу DELET разных типов транзактов - такси и пассажиров.

Предположив, например, что средний интервал прихода пассажиров равен 1 минуте, средний интервал прихода такси равен 10 минутам, при общем времени моделирования 10 часов, сформируем следующую программу:

```
#include <pilgrim.h>
float win01=600.0; /* время моделирования */
float win02=1.0;  /* интервал прихода пассажиров */
float win03=10.0; /* интервал прихода такси */
forward
{
  modbeg ("Маршр.такси",8,win01,(long)time(NULL),none,2,none,5,2);
     ag ("Пассажиры",1,none,expo,win02,zero,zero,2);
     ag ("Такси",6,none,norm,win03,win03/3,zero,7);
     network (dummy,dummy)
      {
            top (2): queue ("Пассажиры", none, 3);
            clcode
            if (addr[2]->na == 1)
                  hold (3); /* начальное размыкание ключа */
            place;
            top (3): key ("Есть такси?", 4);
                  place;
            top (4): delet ("Посадка в такси", 0, 10, 5);
```

```
                rels (3);
                hold (8);
                place;
            top (5): term ("Такси уехало");
                hold (3);
                rels (8);
                place;
            top (7): queue ("Такси", none, 4);
                place;
            top (8): key ("Остановка свободна?", 4);
                place;
                fault (123);
        }
        modend ("pilgrim.rep", 1, 12, page);
        return 0;
}
```

Отметим, что в этой модели нет ни одного узла обслуживания (узлов типа SERV и PROC), но в очередях могут возникать существенные задержки.

Модифицируем нашу программу таким образом, чтобы пользователь мог в диалоговом окне самостоятельно регулировать параметры интервалов прихода пассажиров, прибытия такси, а также времени моделирования.

Для создания программного файла модели с диалоговым окном, помимо файлов, указанных в п.8.1 используется типовая программа *Parametr.cpp*, настраиваемая на конкретную модель, и файл ресурсов *UserRes.rc*.

Файл ресурсов имеет две части: неизменяемую (стандартную) и изменяемую. Макет файла – это *UserRes.rc*, находящийся в разделе *C:\Pilgrim-5-60\Pilgrim5\Projec*ts. Ниже приведена изменяемая часть описания ресурсов. Трактовка операторов языка описания ресурсов приведена в Приложении 1.

```
//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//
//              ИЗМЕНЯЕМАЯ ЧАСТЬ ОПИСАНИЯ РЕСУРСОВ
```

```
//
/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

PARMBOX DIALOG DISCARDABLE  1, 28, 315, 202
STYLE WS_POPUP | WS_CAPTION
CAPTION "Диалоговая настройка модели"
BEGIN
   GROUPBOX    "Параметры бизнес-роцесса",100,5,3,245,30,WS_TABSTOP
   LTEXT       "Фирма:",-1,10,20,30,8,NOT WS_GROUP
   EDITTEXT    DM_TITLE,40,18,200,12
   LTEXT       "Единица времени",-1,78,38,80,8,NOT WS_GROUP
   LTEXT       "Время моделирования",-1,63,53,95,8,NOT WS_GROUP
   LTEXT       "Прибытие пассажиров",-1,63,68,95,8,NOT WS_GROUP
   LTEXT       "Интервал подхода такси",-1,63,83,95,8,NOT WS_GROUP
   LTEXT       "Окошко 4",-1, 63, 98, 95,8,NOT WS_GROUP
   LTEXT       "Окошко 5",-1, 63,113, 95,8,NOT WS_GROUP
   LTEXT       "Окошко 6",-1, 63,128, 95,8,NOT WS_GROUP
   LTEXT       "Окошко 7",-1, 63,143, 95,8,NOT WS_GROUP
   LTEXT       "Окошко 8",-1, 63,158, 95,8,NOT WS_GROUP
   LTEXT       "Окошко 9",-1, 63,173, 95,8,NOT WS_GROUP
   LTEXT       "Окошко 10",-1, 63,188, 95,8,NOT WS_GROUP
   LTEXT       "Период времени",-1,229, 38, 80,8,NOT WS_GROUP
   LTEXT       "Окошко 11",-1,214, 53, 95,8,NOT WS_GROUP
   LTEXT       "Окошко 12",-1,214, 68, 95,8,NOT WS_GROUP
   LTEXT       "Окошко 13",-1,214, 83, 95,8,NOT WS_GROUP
   LTEXT       "Окошко 14",-1,214, 98, 95,8,NOT WS_GROUP
   LTEXT       "Окошко 15",-1,214,113, 95,8,NOT WS_GROUP
   LTEXT       "Окошко 16",-1,214,128, 95,8,NOT WS_GROUP
   LTEXT       "Окошко 17",-1,214,143, 95,8,NOT WS_GROUP
   LTEXT       "Окошко 18",-1,214,158, 95,8,NOT WS_GROUP
   LTEXT       "Окошко 19",-1,214,173, 95,8,NOT WS_GROUP
   LTEXT       "Окошко 20",-1,214,188, 95,8,NOT WS_GROUP
   EDITTEXT       DM_T1,   5, 36,70,12
   EDITTEXT       DM_P1,   5, 51,55,12
   EDITTEXT       DM_P2,   5, 66,55,12
```

```
    EDITTEXT      DM_P3,   5, 81,55,12
    EDITTEXT      DM_P4,   5, 96,55,12
    EDITTEXT      DM_P5,   5,111,55,12
    EDITTEXT      DM_P6,   5,126,55,12
    EDITTEXT      DM_P7,   5,141,55,12
    EDITTEXT      DM_P8,   5,156,55,12
    EDITTEXT      DM_P9,   5,171,55,12
    EDITTEXT      DM_P10,  5,186,55,12
    EDITTEXT      DM_T2, 156, 36,70,12
    EDITTEXT      DM_P11,156, 51,55,12
    EDITTEXT      DM_P12,156, 66,55,12
    EDITTEXT      DM_P13,156, 81,55,12
    EDITTEXT      DM_P14,156, 96,55,12
    EDITTEXT      DM_P15,156,111,55,12
    EDITTEXT      DM_P16,156,126,55,12
    EDITTEXT      DM_P17,156,141,55,12
    EDITTEXT      DM_P18,156,156,55,12
    EDITTEXT      DM_P19,156,171,55,12
    EDITTEXT      DM_P20,156,186,55,12
    PUSHBUTTON     "Готово",IDOK,    265, 7,34,12
    PUSHBUTTON     "Отмена",IDCANCEL,265,21,34,12
  END
```

Этот файл обеспечивает вывод диалогового окна для управления 20 параметрами модели (рис.9.2).

В данном примере количество окошек избыточно, поскольку задействовано только 3 из 20 для отображения и управления параметрами модели: 1) float win01 - «Время моделирования»; 2) float win02 - «Прибытие пассажиров»; 3) float win03 - «Интервал подхода такси». Зарезервировано также окошко с номером 11 для отображения параметра float win11, который в модели не используется.

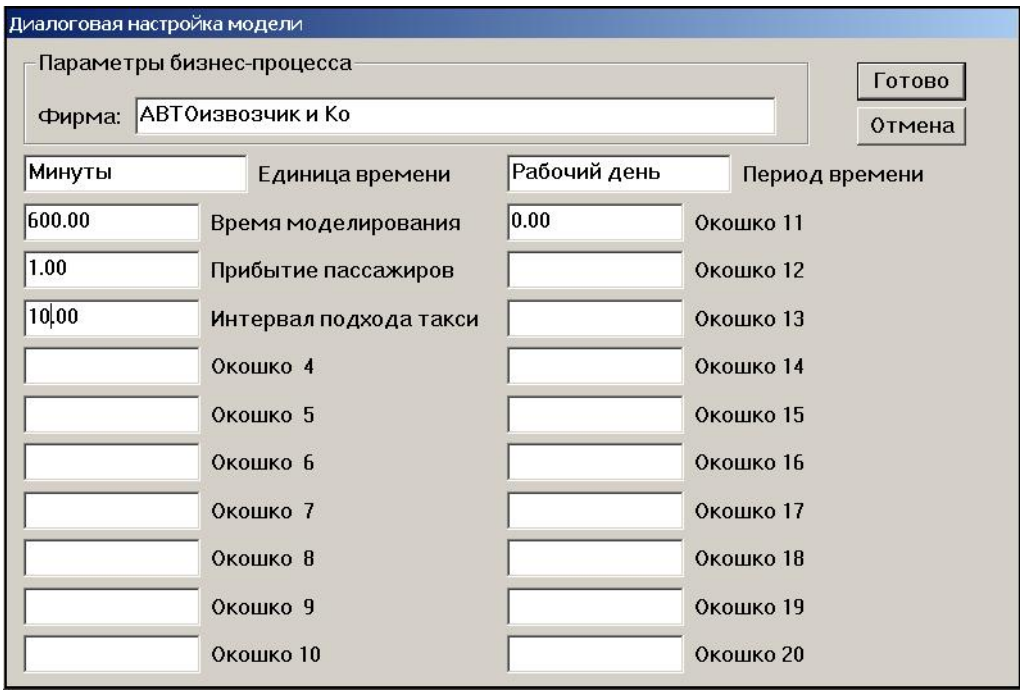


Рис.9.2 «Диалоговое окно для управления параметрами модели»

Далее рассмотрим текст программы, которая выводит численные значения параметров модели и позволяет их корректировать. Обращение к этой программе производится с помощью вызова Parametr из любого места (или любого узла) модели. Описание макета такой программы приведено ниже:

```
#include <Pilgrim.h>
#include "UserHid.h"

extern char  commtext[];
extern char  lefttext[];
extern float win01;
extern float win02;
extern float win03;
extern char  rightext[];
extern float win11;

/*********************************************************************/
/*          Диалог настройки параметров процесса              */
/*********************************************************************/
```

```
BOOL CALLBACK ParametrFunc (HWND hdwnd, UINT message, WPARAM
                                    wParam, LPARAM lParam)
 {
 switch(message)
   {
   case WM_INITDIALOG:
     SetDlgItemText (hdwnd,DM_TITLE,commtext);
     SetDlgItemText (hdwnd,DM_T1,   lefttext);
     sprintf (str,"%4.2f",win01);
     SetDlgItemText (hdwnd,DM_P1, str);
     sprintf (str,"%4.2f",win02);
     SetDlgItemText (hdwnd,DM_P2, str);
     sprintf(str,"%4.2f",win03);
     SetDlgItemText (hdwnd,DM_P3, str);
     SetDlgItemText (hdwnd,DM_T2, rightext);
     sprintf (str,"%4.2f",win11);
     SetDlgItemText (hdwnd,DM_P11, str );
     return FALSE;

   case WM_COMMAND:
     switch (wParam)
       {
       case IDOK:
         GetDlgItemText (hdwnd,DM_TITLE,commtext,80);
         GetDlgItemText (hdwnd,DM_T1,   lefttext,80);
         GetDlgItemText (hdwnd,DM_P1, str,  80);
         sscanf (str,"%f",&win01);
         GetDlgItemText (hdwnd,DM_P2, str,  80);
         sscanf (str,"%f",&win02);
         GetDlgItemText (hdwnd,DM_P3, str,  80);
         sscanf (str,"%f",&win03);
         GetDlgItemText (hdwnd,DM_T2, rightext, 80);
         GetDlgItemText (hdwnd,DM_P11, str,  80);
         sscanf (str,"%f",&win11);
         EndDialog (hdwnd,TRUE);
```

```
            break;
          case IDCANCEL:
            EndDialog (hdwnd,FALSE);
            break;
          default:
            return FALSE;
          }
        break;
      default:
        return FALSE;
      }
  return TRUE;
  }
```

Начальные значения переменных win01, win02, win03, win11, а также заголовки и надписи задаются в головной части текста модели:

```
#include <Pilgrim.h>                    // Модель с диалоговой настройкой
char commtext[81]="АВТОизвозчик и Ко";  // Название фирмы
char lefttext[81]="Минуты";             // Время измеряется в минутах
float win01=600.0;                      // Время моделирования
float win02=1.0;                        // Интервал прихода пассажиров
float win03=10.0;                       // Интервал прихода такси
char rightext[81]="Рабочий день";       // Название интервала моделир-я
float win11=0.0;                        // Резерв(окошко 1 правого столбца)

forward
  {
  Parametr;                             // Диалог настройки параметров
  modbeg ("Маршр.такси",8,win01,long)time(NULL),none,2,none,5,2);
  ag ("Пассажиры",1,none,expo,win02,zero,zero,2);
  ag ("Такси",6,none,norm,win03,win03/3,zero,7);
  network (dummy,dummy)
    {
        .. .. ..
        .. .. ..
```

```
  }
modend ("Res_taxi.doc",1,12,page);
return 0;
}
```

После возможной корректировки файлов *UserRes.rc* и *Parametr.cpp* сборка и компиляция модели выполняется аналогично процедуре, указанной в п.8.1., с добавлением к проекту файлов *UserRes.rc.* и *Parametr.cpp.*

Приложение 1 «Описание ресурсов для создания диалогового окна управления моделью»

1. Оператор CAPTION. Определяет заголовок диалога.

Синтаксис:

CAPTION Text

Пример:

CAPTION «Диалоговая настройка модели»

Параметры:

Text представляет собой ASCII-строку, заключенную в двойные кавычки.

2. Оператор DIALOG. Определяет диалог.

Синтаксис:

Id_name DIALOG [Loading] [Memory] X, Y, Width, Height

(квадратные скобки указывают на необязательное использование параметра)

Пример:

PARMBOX DIALOG DISCARDABLE 1, 28, 315, 202

Параметры:

- Id - name - уникальное имя;
- Loading - способ загрузки ресурса. Значение RELOAD указывает, что ресурс загружается в память при старте программы, а значение LOADONCALL - что загрузка будет при первом обращении к нему;
- Memory - определяет, как система обращается с памятью, в которой расположен ресурс. Эта память может быть зафиксированной (FIXED), перемещаемой (MOVABLE) или выгружаемой (DISCARDABLE), когда в ней нет необходимости;
- X и Y - координаты левого верхнего угла диалогового окна относительно левого верхнего угла экрана;

- Width и Height - ширина и высота окна.

Сразу за оператором DIALOG может следовать оператор STYLE, определяющий стиль диалога. Стиль задается любой комбинацией элементарных стилей окна, сформированной с помощью логического оператора ИЛИ (вертикальная черта). Например, при определении диалогов иногда используется следующая комбинация стилей: DS_MODALFRAME | WS_POPUP | WS_VISABLE | WS_CAPTION | WS_SYSMENU.

3. Оператор EDITTEXT. Определяет окна ввода регулируемого параметра.

Синтаксис:

EDITTEXT Id, X, Y, Width, Height [, Style]

Пример:

EDITTEXT DM_P9, 5, 171, 55, 12

Параметры:

- Id - целое число или уникальное имя, ассоциированное с элементом управления;
- X и Y - координаты левого верхнего угла окна ввода (относительно левого верхнего угла окна диалога);
- Width и Height - ширина и высота окна;
- Style - задается любой комбинацией элементарных стилей, сформированной с помощью логического оператора ИЛИ. Обычно включают элементарные стили WS_TABSTOP, WS_GROUP, WS_VSCROLL, WS_HSCROLL и WS_DISABLE. Стили по умолчанию - это ES_LEFT, WS_BORDER и WS_TABSTOP. Другие стили задаются при описании оператора STYLE.

4. Оператор GROUPBOX. Определяет рамку группы - элемент, представляющий собой прямоугольную рамку с текстовым заголовком, используемую для визуального группирования других элементов управления.

Синтаксис:

GROUPBOX Text, Id, X, У, Width, Height [, Style]

Пример:

GROUPBOX «Параметры бизнес-процесса», 100, 5, 3, 245, 30, WS_TABSTOP

Параметры:

- Text - выводимый текст, представляет собой ASCII-строку, заключенную в двойные кавычки;
- Id - целое число или уникальное имя, ассоциированное с элементом управления;
- X и Y - координаты левого верхнего угла рамки (относительно левого верхнего угла окна диалога);
- Width и Height- ширина и высота рамки;
- Style - задается любой комбинацией элементарных стилей BS_GROUPBOX, WS_TABSTOP и WS_DISABLE, сформированной с помощью логического оператора ИЛИ. Стиль по умолчанию определяется как BS_GROUPBOX; другие стили задаются при описании оператора STYLE.

5. Оператор LTEXT. Определяет статический элемент управления, отображающий текст, который выравнивается по левому краю ограничивающего прямоугольника. Он применяется исключительно в диалогах.

Синтаксис:

LTEXT Text, Id, X, У, Width, Height [, Style]

Пример:

LTEXT "Окошко 20", -1, 214,188, 95, 8, NOT WS_GROUP

Параметры:

- Text - выводимый текст, представляет собой ASCII-строку, заключенную в двойные кавычки;
- Id - целое число или уникальное имя, ассоциированное со статическим текстом;

- X и У - координаты левого верхнего угла элемента управления (относительно левого верхнего угла окна диалога);
- Width и Height - размеры элемента управления;
- Style - задается любой комбинацией элементарных стилей SS_LEFT, WS_GROUP и WS_TABSTOP, сформированной с помощью логических операторов или (вертикальная черта), НЕ (NOT). Стиль по умолчанию определяется как SS_LEFT | WS_GROUP; другие стили задаются при описании оператора STYLE.

6. Оператор PUSHBUTTON. Определяет кнопку (push-button). Кнопка представляет собой прямоугольник со слегка закругленными краями, содержащий текст или графическое изображение, посылаемые родительскому окну при воздействии на эту кнопку щелчком мыши.

Синтаксис:

PUSHBUTTON Text, Id, X, У, Width, Height [, Style]

Пример:

PUSHBUTTON "Готово", IDOK, 265, 7, 34, 12

Параметры:

- Text - текст, отображаемый на «поверхности» кнопки (ASCII - строка, заключенная в двойные кавычки);
- Id - целое число или уникальное имя, ассоциированное с элементом управления;
- X и Y - координаты левого верхнего угла элемента управления (относительно левого верхнего угла окна диалога);
- Width и Height - размеры элемента управления;
- Style - задается любой комбинацией элементарного стиля BS_PUSHBUTTON со стилями WS_DISABLE, WS_GROUP и WS_TABSTOP, сформированной с помощью логических операторов ИЛИ (вертикальная черта), НЕ (NOT). Стиль по умолчанию оп-

ределяется как комбинация BS_PUSHBUTTON | WS_GROUP; другие стили задаются при описании оператора STYLE.

7. Оператор STYLE. Определяет стиль создаваемого диалога. Используется вместе с оператором DIALOG.

Синтаксис:

STYLE Sti1es

Пример:

STYLE WS_POPUP | WS_CAPTION

Параметры:

Stiles – стиль диалога по умолчанию определяется как комбинация WS_POPUP, WS_BORDER и WS_SYSMENU. Ниже приводятся другие элементарные стили, которые могут использоваться для определения стиля окна, и вызываемые ими эффекты.

Стиль	**Эффект**
DS_MODALFRAME	Окно диалога имеет широкую рамку, не позволяющую изменять размеры окна
DS_NOIDLEMSG	Уничтожение в очереди сообщений сообщения WM_ENTERIDLE, которое обычно посылается при активизации диалога его родительскому окну
DS_SYSMODAL	Отображение системного модального диалога (до его завершения продолжение работы с любой программой невозможно)
WS_BORDER	Окно имеет рамку
WS_CAPTION	Окно имеет титульную полосу, в которой отображается заголовок
WS_CHILD	Окно порождено другим окном

WS_CHILDWINDOW	То же, что и WS_CHILD
WS_CLIPCHILDREN	В процессе рисования родительского окна не разрешается рисование в порожденных окнах
WS_CLIPSIBLINGS	Перерисовка одного из порожденных окон не влечет за собой перерисовку других
WS_DISABLED	Неактивное окно, не получающее фокуса ввода
WS_DLGFRAME	Рисуется окно с рамкой, как у обычного диалога; заголовок его отсутствует
WS_GROUP	Определяет первый элемент в группе элементов управления
WS_HSCROLL	Окно имеет горизонтальную линейку прокрутки
WS_ICONIC	Отображение окна свернутым до иконки
WS_MAXIMIZED	Отображение окна в полноэкранном представлении
WS_MAXIMIZEBOX	Окно имеет кнопку полноэкранного представления (максимизации) в правом верхнем углу
WS_MINIMIZE	Отображение окна свернутым до иконки
WS_MINIMIZEBOX	Окно имеет кнопку минимизации в правом верхнем углу
WS_OVERLAPPED	Определение окна с рамкой и заголовком
WS_OVERLAPPEDWINDOW	Определяется как комбинация стилей: WS_OVERLAPPED | WS_SYSMENU | WS_CAPTION | WS_THICKFRAME | WS_MINIMIZEBOX |WS_MAXIMIZEBOX
WS_POPUP	Определение всплывающего (popup) окна
WS_POPUPWINDOW	Создание всплывающего окна со следующей комбинацией стилей: WS_POPUP | WS_BORDER | WS_SYSMENU
WS_SIZEBOX	В правом верхнем углу окно содержит элемент управления, позволяющий изменять его разме-

	ры (sizebox)
WS_SYSMENU	Окно имеет кнопку системного меню в левом верхнем углу
WS_TABSTOP	Используется для элементов управления, которые можно выбрать клавишей табуляции
WS_THICKFRAME	Окно имеет толстую рамку, используемую для изменения его размеров
WS_VISIBLE	Окно изначально видимо
WS_VSCROLL	Окно с вертикальной линейкой прокрутки

ОГЛАВЛЕНИЕ

§1. ВВЕДЕНИЕ В ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ 2

§2. СИСТЕМЫ МАССОВОГО ОБСЛУЖИВАНИЯ 5

2.1. Понятие СМО 5
2.2. Классификация СМО. 5

§3. ЗАКОНЫ РАСПРЕДЕЛЕНИЯ СЛУЧАЙНЫХ ВЕЛИЧИН 6

3.1. Нормальное распределение (гауссово) 7
3.2. Равномерное распределение 8
3.3. Экспоненциальное распределение (показательное) 9
3.4. Обобщенное распределение Эрланга 10
3.5. Треугольное распределение 11

§4. ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА СИСТЕМЫ «PILGRIM» 12

4.1. Этапы построения имитационной модели 12
4.2. Транзакты, узлы, события, ресурсы. Модельный таймер. 13
4.3. Типы узлов имитационной модели 16
Узел AG - генератор транзактов 16
Узел SERV – обслуживающий многоканальный прибор 18
Узел QUEUE - очередь транзактов (с приоритетами или без приоритетов) 20
Узел TERM - безусловный терминатор транзактов 20
Узел KEY - клапан на пути транзактов 21
Узел CREAT - управляемый генератор транзактов (мультипликатор) 21
Узел DELET - управляемый терминатор транзактов (демультипликатор) 22
Узлы SEND и DIRECT - операции с денежными ресурсами 23
Узлы ATTACH и MANAGE - операции с материальными ресурсами .. 25
Узел PROC – транзактно-управляемый процесс 27
Узел DINAM - очередь к процессу с динамическими пространственно-зависимыми приоритетами 30
4.4. Управление переходами в многослойных моделях 31
Узел DOWN - переход на нижерасположенный слой 31
Узел PAY - имитация перечисления денежной суммы 32
Узел RENT - имитация получения ресурса со склада 32
Узел PARENT - объединение группы процессов 33
4.5. Сигнальные управляющие функции 33
4.6. Параметры транзактов 36
4.7. Параметры состояния узлов 36
4.8. Датчики случайных величин 37
4.9. Структура имитационной модели в системе Pilgrim 38

§5. ИНСТАЛЛЯЦИЯ ПРОГРАММНОГО ПАКЕТА «PILGRIM-5» 43

§6. СОЗДАНИЕ МОДЕЛЕЙ С ПОМОЩЬЮ ГРАФИЧЕСКОГО КОНСТРУКТОРА GEM 45

6.1. Редактирование графа модели 46
6.2. Определение параметров узла 48
6.3. Определение параметров инициализации и завершения модели . 52
6.4. Работа в плоскостях модели 53
6.5. Определение переменных модели 55
6.6. Дополнительные функции 55
Изменение настроек экрана 55
Проверка корректности модели 56
План бухгалтерских счетов 57
Поиск узла 57
Копирование/вставка узла 58
Очистить плоскость 58

§7. ПРИМЕР СОЗДАНИЯ ИМИТАЦИОННОЙ МОДЕЛИ ЭКОНОМИЧЕСКОГО ПРОЦЕССА 60

§8. ОБРАБОТКА ИМИТАЦИОННОЙ МОДЕЛИ В СИСТЕМЕ PILGRIM 70

8.1. Формирование программного файла модели 70
8.2. Трактовка результатов моделирования 71
8.3. Трассировка модели 74

§9. РАЗВИТИЕ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА В ИМИТАЦИОННЫХ МОДЕЛЯХ 75

9.1. Проектирование моделей с диалоговым окном для управления параметрами 75

ПРИЛОЖЕНИЕ 1 «ОПИСАНИЕ РЕСУРСОВ ДЛЯ СОЗДАНИЯ ДИАЛОГОВОГО ОКНА УПРАВЛЕНИЯ МОДЕЛЬЮ» 86