Министерство высшего и среднего специального
образования РСФСР

Куйбышевский ордена Трудового Красного Знамени
авиационный институт имени С.П. Королева

М.А. Кораблин, С.В. Смирнов

МОНИТОР ДЛЯ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ СИСТЕМ С ДИСКРЕТНЫМИ СОБЫТИЯМИ

Учебное пособие

Куйбышев 1980

УДК 519:95

Описываются структура, организация и правила пользования монитором CREATE, предназначенным для моделирования систем с дискретными событиями. Монитор построен на базе языка ПЛ/I в ДОС ЕС и может быть широко использован студентами специальностей 0646-0647 ("АСУ" и "Прикладная математика) при выполнении курсовых и дипломных проектов для программирования имитационных моделей сложных систем. Кроме того, настоящее описание монитора призвано иллюстрировать лекционный материал курса "Моделирование систем" (раздел "Автоматизация программирования моделей"), читаемого для студентов специальности 0646 - "Автоматизированные системы управления".

Монитор CREATE включен в государственный фонд алгоритмов и программ (№ П 004061 от 28.12.79).

Темплан 1980 г. поз. 1138.

Рецензенты: Д.Д.К л о в с к и й, Б.К.Б р ю х а н о в

Утверждено на редакционно-издательском совете института 17.11.78 г.

Введение

Монитор для имитационного моделирования CREATE представляет собой набор подпрограмм, формализующих и воспроизводящих динамические свойства моделируемой системы, которые наряду со стандартными возможностями дисковой операционной системы ДОС ЕС и имеющегося в ней подмножества языка ПЛ/I позволяют пользователю на основе единой системы понятий быстро составлять сложные машинные имитационные программы.

Имитационное моделирование систем на ЭВМ с целью их анализа и синтеза получает в настоящее время все большее распространение. В то же время программные средства подобного моделирования мало доступны широкому кругу пользователей. Такие языки, как СИМСКРИПТ, *SIMULA*, *CSL* практически не используются в СССР ввиду отсутствия эффективных трансляторов; компилятор с одной из версий языка *GPSS* [1], работающий в рамках ОС ЕС, имеется лишь в ограниченном числе организаций; кроме того, практически отсутствуют инструкции по использованию этого языка. Получивший наибольшее распространение в СССР язык СЛЭНГ [2] используется на ЭВМ БЭСМ-6, на этой же ЭВМ может быть реализован и язык НЕДИС [3], в ДОС АСВТ разработан пакет программ имитационного моделирования СКИФ [4], однако доступных широкому кругу пользователей средств имитационного моделирования, которые использовались бы в наиболее распространенной в настоя-

щее время операционной системе отечественных ЭВМ третьего поколения ДОС, практически нет. В этой связи, на наш взгляд, весьма актуальной является разработка простого, гибкого и компактного средства автоматизации программирования имитационных моделей сложных систем, которое может быть освоено пользователем, не имеющим специальной подготовки системного программиста, и использовано на машинах ЕС, получивших наибольшее распространение в последние годы с операционной системой ДОС.

Монитор CREATE лишен многих специальных средств, имеющихся в других более "мощных" языках моделирования. Набор специальных средств организации сбора и вывода статистической информации здесь ограничен. Разработчики CREATE сознательно пошли на это ради упрощения структуры системы, полагая, что пользователь, владеющий языком ПЛ/I, может самостоятельно и без больших усилий решать многие дополнительные, фактически сервисные вопросы моделирования. Монитор, на наш взгляд, облегчает труд программиста, в главном: он позволяет пользователю не заботиться о правильном ходе системного времени в программе модели, выполняя автоматически эту функцию.

С формальной точки зрения CREATE является расширением языка ПЛ/I ДОС ЕС путем введения набора специальных подпрограмм, насчитывающих в описываемом ниже варианте до семисот операторов языка ПЛ/I. Все подпрограммы объединены в два модуля: *ωINITC* и *ωSTART* . Объединение большинства управляющих подпрограмм в общий исходный модуль с несколькими входами позволяет уменьшить объем памяти, требующийся для их реализации. Исходные модули программ монитора CREATE написаны на языке ПЛ/I [5], легко читаются и, что, на наш взгляд, является одним из достоинств, - могут быть легко модифицированы программистом (не обязательно системным) для расширения возможностей имитации. Объектный модуль описываемой ниже версии CREATE записан на МЛ.

К пользователю CREATE предъявляется лишь одно требование: умение программировать на языке ПЛ/I.

В настоящем пособии описаны только правила работы с CREATE (фактически инструкция пользователю). Описание

внутренней структуры и организации системы отсутствует, это объясняется, во-первых, тем обстоятельством, что пособие адресовано студентам, впервые знакомящимся со средствами автоматизации программирования имитационных моделей сложных систем. Многие понятия, свойственные языкам и системам моделирования, например: "императивное управление", "системные часы", схема моделирования", - являются для предполагаемой аудитории читателей не только новыми, но и в известной степени неожиданными, что, как показывает опыт чтения курса "Моделирование систем", затрудняет (во всяком случае на первых порах) их усвоение студентами. Поэтому вводить в пособие еще и раздел внутренней организации монитора, разобраться в которой довольно трудно, мы сочли излишним.

При описании средств CREATE мы старались вести изложение "максимально иллюстративно", это опять-таки объясняется особенностями нашей аудитории. Популярной литературы (имеется в виду учебная) по средствам автоматизации программирования в настоящее время практически нет, отдельные разделы в монографиях, статьи и описание тех или иных языков моделирования обычно носят ярко выраженный отпечаток профессионализма их авторов, - что, естественно, затрудняет восприятие подобного материала студенческой аудиторией. Авторы имели возможность убедиться на собственном опыте, насколько сумбурны и неоднозначны толкования многих понятий, являющихся узловыми в языках и системах моделирования. Поэтому при написании этого пособия мы отказались от заманчивой мысли привести популярное изложение какого-либо существующего и наиболее используемого программного средства моделирования, разработав в основном с чисто учебными целями монитор CREATE. Он прост и, как уже отмечалось выше, лишен "многослойности сервисной шелухи", свойственной более "мощным" языковым средствам моделирования. В этом мы видим его основное достоинство: отсутствие подобной шелухи делает более зримыми и отчетливыми контуры ядра любой моделирующей системы - монитора моделирования - и позволяет ввести читателя в круг основных, узловых понятий имитационных систем, используя в то же

время терминологию, свойственную вполне современным средствам имитационного моделирования.

Читатель, усвоивший подобную терминологию и разобравшийся в основных приемах построения имитационных моделей системы, может в дальнейшем, как нам кажется, без особого труда самостоятельно разработать необходимые ему сервисные программы (стандартный вывод "протокола моделирования", программы генерации случайных факторов, организацию наборов данных и т.д. и т.п.).

Нам представляется, что учебный монитор имитационного моделирования CREATE может служить хорошей основой как для усвоения основных понятий систем и языков имитационного моделирования, так и для использования этих понятий в дальнейшем при разработке "более мощных" программных средств моделирования и применении их для решения задач имитации сложных систем.

I. КОНЦЕПЦИИ МОДЕЛИРОВАНИЯ, РЕАЛИЗОВАННЫЕ В CREATE

Имитация любой достаточно сложной технической (или экономической и т.п.) системы должна проводиться с использованием некоторой единой системы понятий, составляющей концептуальную основу средства имитационного моделирования. Использование подобной системы понятий позволяет, во-первых, легко "читать" программу моделирования (что особенно важно при разработке отдельных ее подпрограмм разными лицами) и, во-вторых, облегчает труд программистов-разработчиков отдельных подмоделей моделируемой системы.

Концептуальную основу системы CREATE составляет объединение двух формальных схем моделирования: схемы событий и схемы работ [I]. При этом моделируемая система классифицируется как система с дискретными событиями, т.е. ее функционирование можно представить в виде последовательности событий или изменений состояния, происходящих мгновенно во времени.

Понятие "система с дискретными событиями" охватывает весьма широкий класс формальных математических схем, описывающих реальные системы; в первую очередь, это системы массового обслуживания и автоматы [6], - модели, широко используемые при исследовании вычислительных устройств и систем. Кроме того, к классу систем с дискретными событиями могут быть в известных условиях отнесены и непрерывные системы, обрабатывающие информацию аналогового типа. Строго говоря, понятие "система с дискретными событиями" не накладывает никаких ограничений на множество возможных состояний системы (это множество может быть как счетным, конечным или бесконеч-

ным, так и несчетным, имеющим мощность континуума). Именно это обстоятельство и делает класс систем с дискретными событиями весьма представительным.

Для подобных систем процесс функционирования моделируемой системы во времени отождествляется при его имитации с последовательностью событий, возникающих в системе в соответствии с закономерностями ее функционирования. В формальное понятие "событие" пользователь может вложить любое конкретное смысловое содержание, определяемое целями и задачами имитации системы. (По схеме событий построены такие языки моделирования, как СИМСКРИПТ [7], СКИФ [4], во многом аналогична ей схема процессов - язык СИМУЛА [8]). События могут возникать при функционировании системы в некоторые моменты времени, в общем случае не совпадающие друг с другом. Например, событие $EV1$: "Приход заявки на расчеты в вычислительной системе" может наступить в некоторый момент времени $t1$, событие же $EV2$: "Окончание расчетов" - в момент времени $t2 = t1 + \Delta$, где Δ - некоторая задержка (в системе может имитироваться как детерминированной, так и стохастической величиной задержки). Кроме того, события в системе могут наступать и одновременно, например, $EV3$: "Приход заявки на расчеты от первого пользователя вычислительной системы" и $EV4$: "Приход заявки на расчеты от второго пользователя". В этом случае имитация реакции системы на одновременное наступление нескольких событий осуществляется в соответствии с приоритетами событий.

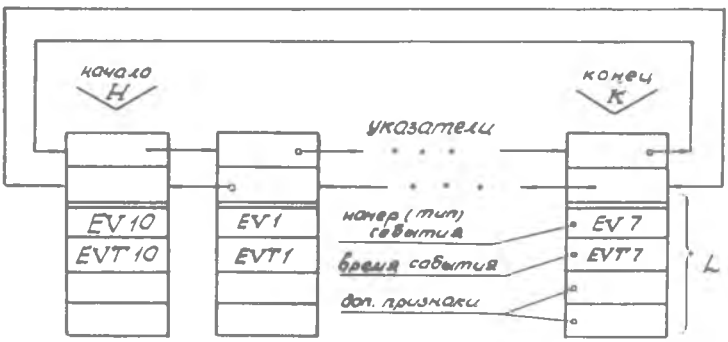
Одновременное возникновение нескольких событий в системе ставит вопрос: на какое из наступивших событий система должна отреагировать, или, другими словами, какую работу начать. Схема работ в имитационной системе связана с проверкой в каждый заданный момент времени условий возможности начать ту или иную работу, - начать реагировать на то или иное событие. Формальное понятие "работа" конкретизируется пользователем. (Схема работ свойственна, например, языку CSL [1], [9]).

Таким образом, с формальной точки зрения имитация процесса функционирования системы во времени связана с генерацией определенных событий в соответствии с закономерностями работы системы и выполнением работ - установлением реакции системы на эти события.

Процесс возникновения событий при имитации функционирования

системы имеет в общем случае стохастический характер: события различных типов могут возникать в различные, заранее непредсказуемые моменты времени, имитируемые в модели с использованием аппарата статистического моделирования (датчики псевдослучайных чисел). Одна из основных задач монитора моделирования - образовать из потока событий рекуррентный поток, - или, иначе говоря, упорядочить события по времени их возникновения в системе.

Для решения этой задачи монитор оперирует с определенной списковой структурой - календарем, в которой собраны метки события (рис. I). Каждая метка $\angle$ события EV состоит из номера события (типа события) - $EV\kappa$, времени его возникновения в системе - $EVT\kappa$ и некоторых дополнительных признаков (подробная организация календаря здесь не рассматривается). Так, для ситуации, приведенной на рис. I, метка события, стоящая в начале списка, определяет событие IO-го типа: $\angle(H)=\{EV10, EVT10,$ доп. признаки $\}$. Завершает календарь метка события 7-го типа: $\angle(K)=\{EV7, EVT7,$ доп. признаки $\}$.



$EVT10 \leqslant EVT1 \leqslant \ldots \leqslant EVT7$

Р и с. I. Структура календаря событий. $\angle$ - метка события

Ведение подобного календаря при моделировании позволяет обеспечить правильный "ход часов системного времени": события в списке упорядочены по времени их появления в системе, а монитор моделирования всегда выбирает из списка событие с наименьшим временем, которое наступает в системе раньше всех остальных (другими словами, происходит в "ближайшем будущем").

Ведение подобного календаря и контроль за правильным ходом часов системного времени - функция монитора моделирования. Пользователь не заботится о контроле времени. В процессе имитации функционирования системы календарь постоянно обновляется: в него вносятся метки новых событий, обусловленных закономерностями работы системы, и исключаются события с минимальным временем, (метка которых базируется по указателю H), или те события, которые в соответствии с логикой работы системы необходимо отменить. Временная упорядоченность списка при этом строго сохраняется.

Ведение такого календаря можно сравнить с заполнением рабочего календаря - пятидневки некоторым административным работником. В процессе своей деятельности он записывает на тот или иной день и час недели текущие дела, которые необходимо решить в этот момент времени (создает "метки событий"). При выполнении того или иного мероприятия соответствующая запись в календаре вычеркивается, но появляются новые "метки", - уведомление о делах, которые необходимо выполнить в последующее время и т.д.

Ведущая программа-монитор моделирования осуществляет включение в список новых событий и удаление из списка ненужных или "устаревших" (происходящих в данный момент времени). Делается это с использованием специальных операторов, которые составляют группу операторов императивного управления [10].

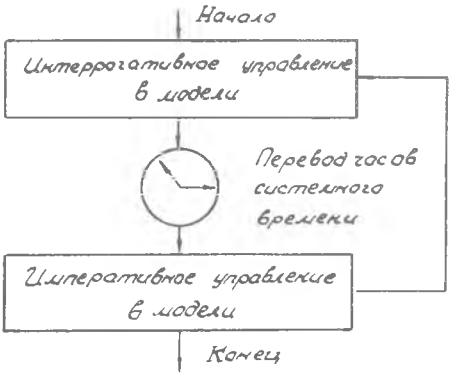
Операторы этой группы связаны с изменениями содержимого календаря. Подобные изменения могут осуществляться по признаку времени. Например, "ввести событие $EV3$ в календарь на момент времени $EVT3 = 4$ (ед. времени)" или "ввести событие $EV1$ в календарь на момент времени $EV4$, который наступит через $T = 5$ (ед. времени)". Кроме того, изменения в календаре могут выполняться по признаку приоритета, например, "ввести событие $EV5$ в календарь перед (после) событием $EV4$ " (временная упорядоченность списка при этом не нарушается, см. подробнее § 3).

Помимо изменения содержимого списка событий - императивного управления - монитор моделирования выполняет функции интеррогативного управления моделью. Операторы второй группы не меняют содержимого календаря и не участвуют в поддержании правильного "хода часов системного времени". При интеррогативном управлении системное время останавливается, при этом имитируются процессы и явления, происходящие в системе мгновенно, при достижении системой

некоторого определенного статуса [IO] (выполнении определенных условий). Например, "имитировать наступление события *EV3* при выполнении условий *BL1* , *BL2* ". В качестве условий *BL1* , *BL2* могут выступать любые из условий, определяемых логикой работы системы.

Функции по интеррогативному и императивному управлению монитор моделирования выполняет попеременно на протяжении всего имитационного процесса (рис. 2).

Существует три различных понятия, которыми приходится оперировать при имитации процессов функционирования систем во времени: реальное время, системное время и машинное время. Реальное время - это время, в котором функционирует моделируемая система, время, определяемое ходом обычных часов. Системное время есть некоторый идентификатор программы моделирования, имитирующий ход часов реального времени.Например, если имитируется процесс функционирования системы с момента ее включения $t=t1$ до момента реального времени $t=t2$, идентификатор системного времени $@STIME$ также определен в области значений $[t1,t2]$: $t1 \leqslant @STIME \leqslant t2$.



Р и с. 2. Управляющие функции монитора

При выполнении в процессе имитации условия $@STIME > t2$ процесс моделирования заканчивается. Пересчет системного времени в модели можно выполнить двумя путями:

I) "двигаясь во времени" с некоторым постоянным малым шагом Δt :

$@STIME = @STIME + \Delta t$

(принцип " Δt " [IO]) или

2) "двигаясь во времени" от события к событию, считая что в промежутке времени между событиями в модели системы никаких изменений не происходит:

@ $STIME = EVT(H)$;*)

$EVT(H)$ – время события, метка которого стоит в начале календаря. (Этот принцип пересчета системного времени называют принципом "особых состояний" [II], принципом "контрольных часов" [I] и т.д.).

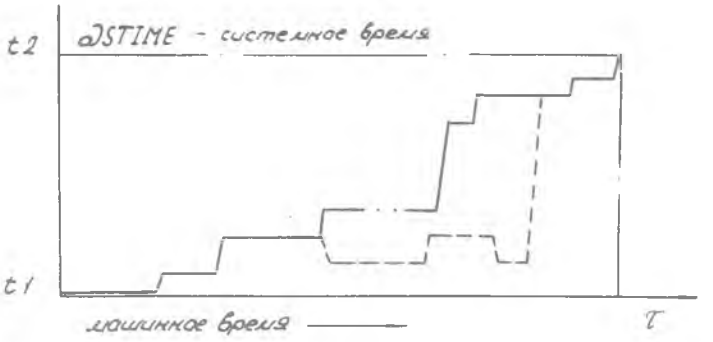
Второй путь естественно более экономичен, поскольку позволяет избежать множества "лишних" прогонов программы, когда в модели системы заведомо ничего не произойдет. Этот метод пересчета системного времени лежит в основе практически всех языковых средств моделирования систем с дискретными событиями, не составляет исключения из этого правила и CREATE. Однако отметим, что в ряде случаев (например, при моделировании процедур дискретизации непрерывных информационных процессов) оказывается целесообразным, а нередко и необходимым, использовать первый путь пересчета системного времени, – средства CREATE не исключают и такой возможности.

Наконец, понятие машинного времени связано лишь с процедурой машинного счета имитационной программы. Время, в течение которого "прогонится" программа модели, определяется (при прочих известных условиях) лишь тем, насколько эффективна программа имитации с машинной "точки зрения", и никак не связано с системным или реальным временем. Впрочем, при правильной работе монитора идентификатор системного времени в машинном времени может только возрастать (рис. 3), имитируя ход часов реального времени. "Эффект сумасшедших часов" (пунктирная кривая на рис. 3) свидетельствует об ошибках в работе монитора моделирования. Величина же τ (см. рис.3) в общем случае никак не связана с величиной $(t2-t1)$ интервала реального времени функционирования системы.

Работа системы в течение часа, суток, года может имитироваться за несколько минут или даже секунд. Впрочем, может быть и обратная ситуация: работа системы измерения, например, быстропеременных процессов в течение нескольких секунд может имитироваться за минуты и даже часы. (Заметим, что модели систем, обсуждаемые в дальнейшем, не относятся к классу моделей реального времени, для которых значения величин $t1$, $t2$ определяют жесткие ограничения на величину τ).

---

*) Здесь и в дальнейшем символ "=" определяет оператор присваивания.



Р и с. 3. Изменение идентификатора системного времени в машинном

В дальнейшем, оперируя с понятием времени, мы всегда будем подразумевать системное время, каждый раз особо оговаривая те ситуации, в которых время имеет иной смысл.

Завершая параграф, отметим присущие монитору CREATE особенности при отображении течения реального времени.

Функционирование исследуемой системы на определенном отрезке реального времени $(t1, t2)$ имитируется в модели изменением системного времени на отрезке $[0, TSIM]$, где $TSIM=t2-t1$. Идентификатор системного времени $@STIME$ имеет атрибуты $DECIMAL\ FLOAT$ (6) (так же, как и $TSIM$), и, следовательно, для того, чтобы во всем диапазоне $[0, TSIM]$ обеспечить постоянную точность приращений Δt системного времени, необходимо согласовать изменения $@STIME$ и диапазон $[0, TSIM]$ с учетом особенностей атрибутов $DECIMAL\ FLOAT$ (6). Например, для максимально возможной точности 0,000001 необходимо $TSIM \leqslant 1.00000E0$, для $0{,}001 \rightarrow TSIM \leqslant 1000.00E0$, для $1 \rightarrow TSIM \leqslant 1E7$ и т.д.

При построении программ моделей может оказаться, что максимально возможное значение $TSIM$, еще обеспечивающее требуемую точность "системных часов", оказывается меньшим величины $t2-t1$. В подобных случаях отображение хода реального времени осуществляется в модели следующим образом:

а) отрезок времени $[0, t2-t1]$ расчленяется на P одинаковых по длине участков:

$[0,\tau]$, $(\tau, 2\tau]$, ..., $((P-1)\tau, P\tau]$,

чтобы при $TSIM=\tau$ "системные часы" на отрезке $[0, TSIM]$ обладали требуемой точностью;

б) при инициализации монитора (см. § 2) ему сообщаются данные о величинах $TSIM=\tau$ и P.

Например, точность изменений @STIME 0,001 и $t2-t1$ = 6000. При этом $t2-t1=P*TSIM$, и для достижения заданной точности можно положить либо $TSIM=1E3$ и $P=6E0$, либо $TSIM=6E2$ и $P=1E1$, либо $TSIM=1E2$ и $P=6E1$ и т.д.

При выполнении имитационной программы на каждом из образованных участков идентификатор системного времени @STIME может принимать значения в диапазоне от 0 до $TSIM$. Параметр P определяет число периодов монотонного роста системного времени(РСВ).

2. ОБЩАЯ ОРГАНИЗАЦИЯ ПРОГРАММЫ МОДЕЛИРОВАНИЯ

Программа моделирования системы состоит из программ, составленных пользователем по обычным правилам языка ПЛ/I с учетом некоторых дополнительных обстоятельств (см. подробнее § 5) и с использованием специальных операторов управления процессом имитации (§ 3, 4), которые дают возможность использовать средства CREATE. Не останавливаясь здесь на деталях программирования и работы конкретных управляющих операторов, приведем пример структуры взаимодействия между программой пользователя и монитором модели, что необходимо для понимания воплощения концептуальных основ моделирования в конкретную имитационную программу.

Программа пользователя составляется из отдельных подпрограмм (процедур языка ПЛ/I), которые могут компилироваться независимо друг от друга [I2], что упрощает процесс исправления ошибок программирования и позволяет в принципе вести программирование и отладку частей имитационной модели параллельно разными исполнителями.

Общая структура программы пользователя иллюстрируется рис.4. В этой структуре помимо главной процедуры, определяющей начальное состояние системы и содержащей основные операторы обращения к средствам CREATE, присутствуют подпрограммы IO-ти различных событий (в общем случае число различных событий может доходить до

```
BEISP: PROCEDURE OPTIONS (MAIN);
    . . . /*Главная процедура*1/. .
END BEISP;
EV1: PROCEDURE; /*Наименование события EV1 */
    . . . /*Программа события EV1 */. . .
END EV1;
EV2: PROCEDURE; /*Наименование события EV2*/
    . . . /*Программа события EV1*/. . .
END EV2;
. . . . . .
EV1ф: PROCEDURE; /*Наименование события EV10*/
    . . . /*Программа события EV10*/. . .
END EV 1ф;
CN1: PROCEDURE BIT(1); /*Булевская функция,
    участвующая в интерроративном управлении */
    . . . /*Условия интерроративного управления
    одним из событий EV */. . . RETURN(A);
END CN1,
. . . . . .
CN4: PROCEDURE BIT(1); . . . /*Условия
    интерроративного управления одним из событий
    EV */. . . RETURN(B); END CN4;
```

Р и с. 4. Общая структура программы пользователя

50, простая модификация позволяет увеличить и это количество) и четыре подпрограммы, определяющие булевские функции, значения которых (*'1'B* или *'0'B*) мы получаем в зависимости от условий интеррогативного управления, задаваемых пользователем.

Программы событий и условий независимы друг от друга. Взаимосвязи между ними осуществляются через монитор моделирования, кроме того, подобные взаимосвязи могут устанавливаться благодаря использованию переменных, общих для нескольких программ событий, (глобальные переменные, описанные с атрибутом *EXTERNAL* [I2]).

Характер взаимосвязей программы пользователя с монитором иллюстрируется схемой (рис. 5. На этой схеме выделены лишь те операторы, которые являются необходимыми для понимания механизма взаимодействия между программой пользователя и монитором CREATE. (Некоторые сокращения, используемые на рис. 5, недопустимы в программе моделирования).

Главная процедура. Два оператора, которые всегда являются первыми специальными операторами при использовании монитора и служат для его инициализации:

DECLARE @PL(n) EXTERNAL; (I)

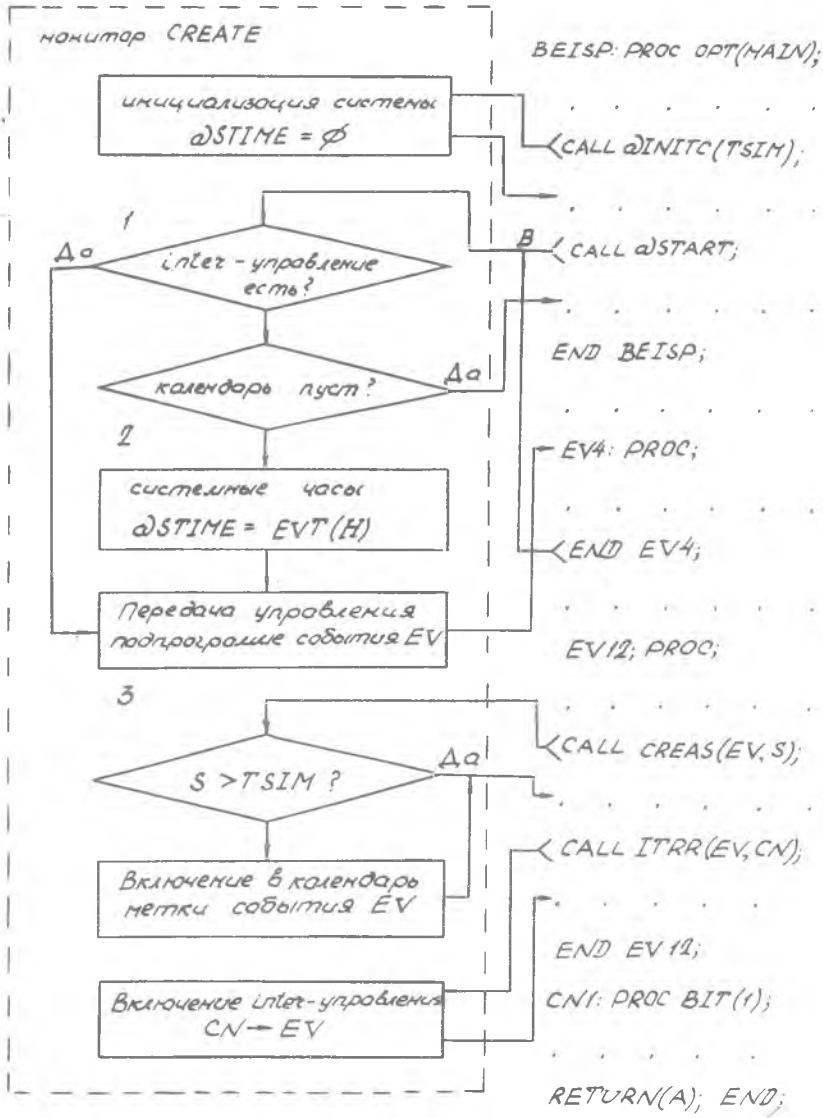
CALL @INITC(TSIM, P, D@PL); (2)

Натуральное *n* в атрибуте размерности массива *@PL*, объявляемого оператором (I), определяет объем памяти, выделяемый монитору для организации календаря событий, $10 \leqslant n \leqslant 32767$; *TSIM* – верхняя граница изменения идентификатора системного времени *@STIME*, а *P* – число периодов монотонного роста системного времени; *D@PL* – размерность массива *@PL*, определенная оператором (I), т.е. *D@PL = n*.

Начало имитации осуществляется оператором *CALL @START*. По окончании процесса моделирования управление передается следующему за ним оператору.

Об определении *TSIM* и *P* подробно говорилось в § I. Если априори известно (практически это всегда именно так), что при выполнении имитационной программы число меток событий в календаре не превышает *m*, то при инициализации необходимо положить *n = 5(m+1)*.

Можно предложить также следующий способ оценки требуемой величины *n*.



Р и с. 5. Структура взаимосвязей программы пользователя с монитором CREATE

Из просмотра программы модели легко указать:

А) ℓ – максимальное число меток событий, пополняющих календарь при выполнении любой из программ событий, и L – число меток, заносимых в календарь до вызова программы *@START* ;

Б) $J=\{j_1, j_2, \ldots, j_K\}$ – множество типов событий, программа каждого из которых способна пополнить календарь ℓ метками событий.

Если априори можно указать вероятность P_j того, что следующее событие в ходе моделирования имеет тип (номер) $j \in J$, то для того, чтобы вероятность переполнения календаря не превышала q , при инициализации необходимо положить
$n = Entier(L + t\ln q / \ln P_j)$.

Программа событий. В программах событий могут присутствовать различные операторы императивного и (или)интеррогативного управления. Механизм действия этих операторов описан подробно в следующих параграфах. На схеме рис. 5 присутствуют лишь два из них: *CALL CREAS(EV,S)* – "внести событие с номером (типа) *EV* в календарь на момент времени *@STIME=S* и *CALL IIRR(EV,CN)* – – "вызвать событие с номером (типа) *EV* , если булевская функция с номером *CN* принимает значение *'1'B* ".

Механизм взаимодействия между программой пользователя и монитором CREATE заключается в следующем. Оператор *CALL @INITC(...)* инициализирует подпрограммы императивного управления и *@START*, подготавливая возможность имитации функционирования системы на отрезке времени $[0, TSIM]$. После выполнения этого оператора управление передается монитором CREATE оператору, следующему в программе пользователя за *CALL@INITC(...)*. Оператор *CALL@ START* осуществляет начало имитации, передавая управление монитору CREATE. Блок I монитора определяет, есть ли в программе пользователя какие-либо события, управляемые интеррогативно (т.е. наступающие в системе при выполнении определенных логических условий). Если такие события есть и условия их возникновения выполнены, то монитор через блок 3 передает управление подпрограмме соответствующего события. Если же интеррогативно управляемых событий в программе пользователя не предусмотрено или ни одно из логических условий, с помощью которых осуществляется интеррогативное управление, не выполнено, монитор передает управление блоку2, который выбирает из календаря событие с наименьшим временем (мет-

ка такого события стоит в начале календарного списка), переводит "стрелки системных часов" на время этого события и передает управление его подпрограмме. По завершении подпрограммы события управление вновь передается монитору (точка *B* на схеме рис. 5), и весь цикл управления повторяется вновь. Операторы императивного и интеррогативного управления, присутствующие в программе пользователя, меняют содержимое календаря (вносят в календарь новые события и исключают ненужные), фиксируют номера событий, управляемых интеррогативно, и условия наступления этих событий в системе. Таким образом, жесткий детерминированный механизм монитора будет все время оперировать с информацией, обновляемой в программе пользователя, что собственно, и составляет основу имитационного процесса.

Рассмотренная схема взаимодействия программы пользователя с монитором CREATE предъявляет к программе пользователя следующие требования:

I) в программе пользователя должны использоваться средства, позволяющие обновлять список-календарь событий с течением системного времени;

2) должны использоваться средства, позволяющие устанавливать (менять) номера событий, управляемых интеррогативно;

3) должны быть предусмотрены средства "обновления" условий интеррогативного управления, определяющих значения булевских функций *CN* .

С формальной точки зрения для того, чтобы монитор нормально функционировал, достаточно выполнения хотя бы одного из этих требований. Однако смысловая содержательная сторона моделирования, представляющая интерес для пользователя, требует от него определенной изобретательности (если не сказать искусства) при составлении программы, удовлетворяющей перечисленным требованиям (всем или их части). Так, в первую очередь пользователь должен очень четко определить смысловое содержание каждого события, участвующего в его программе, условия возникновения этого события в системе, реакцию системы на это событие и т.д. Лишь после этого может решаться вопрос о том, каким образом должно происходить управление этим событием при имитации: императивно или интеррогативно. На этот вопрос в общем случае не удается дать однозначный ответ: все определяется соображениями "удобства" и навыками пользователя

CREATE как программиста. В программе пользователя должны задаваться начальные условия функционирования, свойственные моделируемой системе, определение этих условий должно проводиться до обращения к программе *@START* . И, наконец, пользователь, составляющий имитационную программу, должен очень четко представлять взаимосвязи между событиями, происходящими в системе. Основная функция программы пользователя заключается в том, чтобы наполнить формальную схему моделирования, по которой работает монитор CREATE, конкретным смысловым содержанием.

3. ПОДПРОГРАММЫ ИМПЕРАТИВНОГО УПРАВЛЕНИЯ

CREAS(EV,S) - "внести в календарь событие с номером *EV* на момент времени *S* ". Оператор *CALL CREAS(EV,S)* позволяет включить в список событий метку события *EV* , которое должно произойти в системе в момент времени *S* таким образом, что общая упорядоченность списка по времени не нарушится. События, вносимые в список с одинаковым временем, упорядочиваются по принципу "первый пришел - первый вышел". Например, для следующего фрагмента программы

```
. . . . . .
EV=1; S=5; CALL CREAS(EV,S);
EV=4; S=7,5; CALL CREAS(EV,S);
EV=10; S=5; CALL CREAS(EV,S);
EV=15, S=5; CALL CREAS(EV,S);
. . . . . .
```

метки событий в календаре будут располагаться в такой последовательности:

..., EV1, EV10, EV15, ..., EV4, ...

CREASP(EV,S) - "внести в календарь событие *EV* на момент времени *S* перед другими событиями с временем *S* ". Действие этой подпрограммы позволяет менять порядок включения в календарь событий, "запланированных" в системе на одно и то же время. Например, при выполнении программы

```
. . . . . .
EV=7; S=15; CALL CREAS(EV,S);
EV=4; S=10; CALL CREAS(EV,S);
```

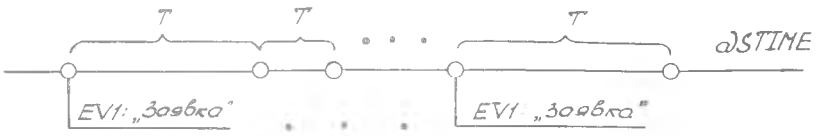
```
EV=11; S=1ϕ; CALL CREAS(EV,S);
EV=2; S=1ϕ; CALL CREASP(EV,S),
EV=17; S=1ϕ; CALL CREASP(EV,S);
. . . . . .
```

в календаре события будут располагаться в такой последовательности . . . , EV2, EV17, EV4, EV11,..., EV7,...

Из этого примера можно заметить, что события, вносимые в список оператором CALL CREASP(EV,S) с одинаковым временем S, также упорядочиваются по принципу "первый пришел - первый вышел". CREAT(EV.T) -"внести в календарь событие EV на момент времени @STIME+T ". Эта подпрограмма позволяет пользователю при планировании будущих событий использовать не "абсолютное" системное время, а интервал T между текущим значением времени @STIME и моментом наступления события в системе. Использование подобного планирования удобно, например, при имитации стохастических рекуррентных потоков, свойственных системам массового обслуживания. Например, при имитации простейшего потока, в котором моменты наступления событий (поступления заявок в систему массового обслуживания) отделены один от другого интервалом, распределенным экспоненциально (величина T на рис. 6), можно воспользоваться следующим механизмом, использующим CREAT(EV,T):

```
EV1: PROCEDURE;          /* Поступление заявки в систему массового обслуживания */
. . . . . .
CALL @RAND(R);                                              (1)
T=-(1/A)*LOG(R);                                            (2)
EV=1; CALL CREAT(EV,T);
END EV1;
```

При этом каждый раз, когда в системе будет наступать событие EV1 : "Заявка", механизм планирования будет вносить в календарь новое событие EV1 : "(Следующая) Заявка" и т.д., - т.е. событие EV1 будет как бы само себя создавать заново - порождать следующее событие (поскольку механизм планирования включен в подпрограмму события EV1). Оператор (I) - обращение к датчику случайных чисел с равномерным распределением (см. подробнее § 6), оператор (2) - алгоритм получения случайной величины, распределенной экспоненциально с параметром A (A - интенсивность потока заявок).



Р и с. 6. Поступление заявок в систему массового обслуживания

В остальном подпрограмма CREAT аналогична *CREAS*: события, вносимые в календарь с помощью CREAT на один и тот же момент системного времени, упорядочиваются так же, как и при использовании *CREAS*.

CREATP(EV,T) – "внести в календарь событие *EV* на момент времени *@STIME+T* перед другими событиями с временем *@STIME+T*". Действие этой подпрограммы аналогично *CREASP* с той лишь разницей, что параметр *T* определяет не "абсолютное" значение системного времени, а некоторую задержку во времени (см. подпрограмму CREAT). Обращение к подпрограмме – *CALL CREATP(EV, T)*. *CREABF(EV,E)* – "внести в календарь событие *EV* перед (*before*) событием *E*". Параметр *E* определяет номер (тип) события, имеющего наименьшее время из всех событий такого же типа, включенных в календарь. При этом событию *EV* будет приписано то значение системного времени, которое имеет событие *E*.

Например, для следующего фрагмента:

```
. . . . . .
E=7; S=4; CALL CREAS(E,S);
S=5; CALL CREAS(E,S);
EV=4; CALL CREABF(EV,E); . . . . .
```

метки событий в списке расположатся в такой последовательности

$$\ldots, \begin{bmatrix} EV4 \\ EVT4=4 \end{bmatrix}, \begin{bmatrix} EV7 \\ EVT7=4 \end{bmatrix}, \begin{bmatrix} EV7 \\ EVT7=5 \end{bmatrix}, \ldots$$

CREAAF(EV,E) – "внести в календарь событие *EV* после (*after*) события *E*". Действие этой подпрограммы аналогично *CREABF* с той лишь разницей, что *EV* записывается в календарь сразу за меткой события *E*. Обращение – *CALL CREAAF(EV,E)*.

Если в календаре не окажется ни одного события типа *E*, выполнение *CREABF* и *CREAAF* соответствует выполнению пустого оператора.

CANCEN (EV) - "исключить событие *EV* из календаря". Из календаря выводится метка события *EV*, имеющая наименьшее время из всех событий такого же типа, включенных в календарь (т.е. событие, "ожидаемое в ближайшем будущем"). Если в списке не окажется событий типа *EV*, выполнение *CANCEL* соответствует выполнению пустого оператора.

Обращение *CALL CANCEL (EV)*.

@ STEN (EV, EVT, EVP) - с помощью этой подпрограммы можно выяснить запланированное время появления события *EV* в системе. При обращении *CALL @ STEN(EV,EVT,EVP)* параметру *EVT* присвоится значение того момента системного времени, а *EVP* - того периода роста системного времени, на которое запланировано в календаре появление события с номером (типа) *EV*. Если в календаре несколько меток событий типа *EV*, параметрам *EVT* и *EVP* присвоятся атрибуты события, ожидаемого в "ближайшем будущем".

Если в списке не окажется событий типа *EV*, то *EVT* присвоится значение (*TSIM+1*), а *EVP* - значение *P* (о величинах *TSIM* и *P* см. § 2).

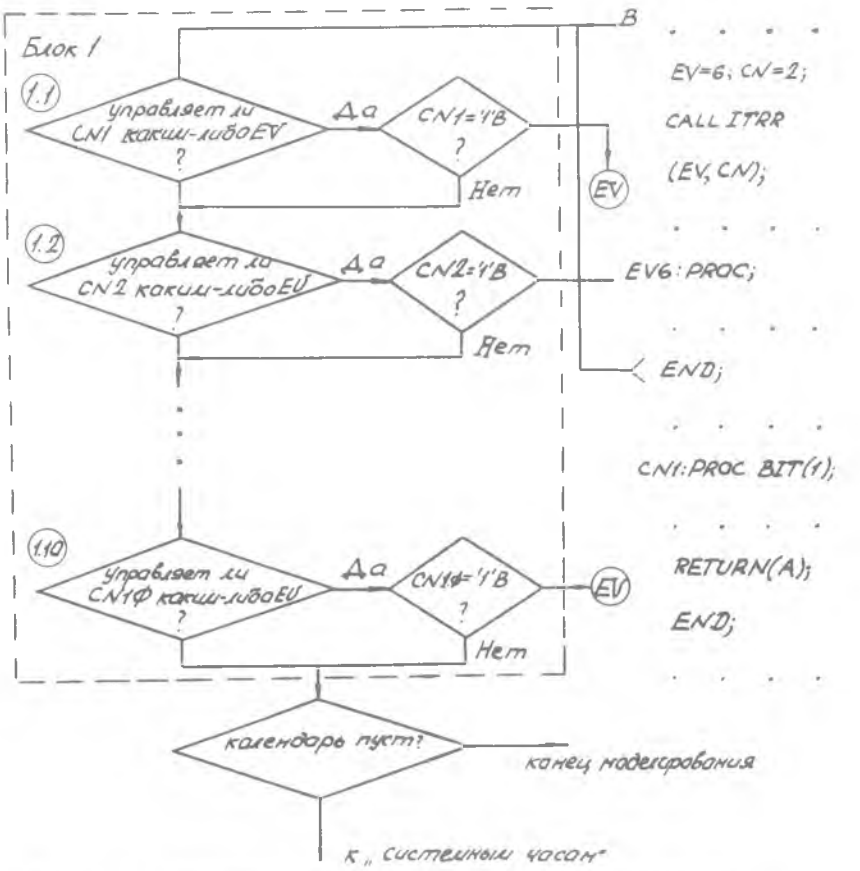
4. ПОДПРОГРАММЫ ИНТЕРРОГАТИВНОГО УПРАВЛЕНИЯ

Интеррогативное управление организуется с помощью пронумерованных булевских программ-функций, реализующих проверку некоторых условий. Конкретное содержание этих программ определяется пользователем. Их число в программе не должно превышать десяти и для их идентификации допустимы имена *CN1*, *CN2*, ..., *CN10*.

Для пояснения механизма интеррогативного управления распишем более подробно структуру блока I рис. 5, отражающего основные взаимосвязи программы пользователя с монитором CREATE.

В программе пользователя (как уже отмечалось выше) при использовании интеррогативного управления должны присутствовать булевские программы функции с именами *CN1*, *CN2*, ... Для того, чтобы "включить" аппарат интеррогативного управления некоторым событием с номером *k (EV k)* функцией с номером *l (CN l)*, в программу пользователя (в любое ее место, после *CALL @INITC(. . .)* необходимо ввести операторы *EV=k; CN=l; CALL ITRR (EV, CN);* При этом блок I (рис. 7) на вопрос, "управляет ли функция каким-

либо событием *EV* ?" отвечает положительно и передает управление блоку проверки условия „*CNi='1'B*?". Если это условие выполнено, управление передается подпрограмме событий с номером *k* .



Р и с. 7. Структура блока интеррогативного управления

В случае, если в программе пользователя нет ни одного *CALL ITRR(EV,CN)*, блоки I.I-I.IO (рис. 7) передадут управление блоку 2 монитора - интеррогативное управление отсутствует. Если в процессе моделирования введенное ранее интеррогативное управление

некоторым событием с помощью функции *CNℓ* необходимо отменить, в программе пользователя достаточно вызвать подпрограмму *ABITRR*:

..., CN=ℓ; CALL ABITRR(CN); ..., –

при этом переключающий блок монитора *I.ℓ* установится в положение "Нет".

Первоначально все переключатели I.I–I.IO установлены в положение "Нет". Изменить положение того или иного из переключающих блоков может только подпрограмма *ITRR* по обращению *CALL ITRR(EV, CN)*.

После начала имитации по оператору *CALL@START* монитор проверяет последовательно положение ключей I.I–I.IO. Если хотя бы один из них установлен в положение "Да", идет проверка соответствующего условия *„CNℓ='1'B?"* и при положительном ответе на этот вопрос управление передается подпрограмме соответствующего события. После отработки этого события в программе пользователя управление передается на точку *B* (см. рис. 5), и все повторяется вновь без пересчета системного времени, имитируя все изменения, происходящие в моделируемой системе мгновенно в реальном времени. Из блока I монитора управление будет передано в блок 2 только тогда, когда не останется ни одной функции *CN*, значение которой равно *'1'B* (ни одной активной функции *CN*).

Это означает, что в программе пользователя значения функций *CN* в процессе имитации должны меняться, – в противном случае монитор никогда не передаст управление блоку 2 – "системным часам". Изменения значений функции *CN*, каждая из которых в программе пользователя представлена своей в общем случае отдельно компилируемой процедурой, осуществляется с помощью аппарата глобальных переменных, имеющих атрибут *EXTERNAL*.

Каждая булевская подпрограмма-функция может управлять только одним событием. Таким образом, одновременно интеррогативно управляемы не более десяти событий. Последовательное выполнение операторов

```
. . . . . ; EV=k; CN=ℓ;
            CALL ITRR(EV,CN);
            . . . . . .
            EV=m; CN=ℓ,
            CALL ITRR(EV,CN); ...
```

автоматически приводит к отмене интеррогативного управления булевской функцией *CNi* для события *EVk* и, разумеется, вводит такое управление для события *EVm*.

5. ПРАВИЛА ПРОГРАММИРОВАНИЯ МОДЕЛИ. ОФОРМЛЕНИЕ ЗАДАНИЯ НА ВЫПОЛНЕНИЕ ИМИТАЦИОННОЙ МОДЕЛИ ПРОГРАММЫ В ДОС ЕС

Программы событий записываются на языке ПЛ/I в виде процедур без параметров и могут иметь имена *EV1* , ..., *EV5ϕ*. В общем случае каждая из этих подпрограмм событий должна транслироваться отдельно, поэтому данные, оперирование с которыми необходимо при выполнении нескольких программ событий, должны быть описаны в них с атрибутом *EXTERNAL* . Традиционное оформление программы события имеет следующий вид:

```
EV1ϕ: PROCEDURE;
      . . . . . .
      END.
```

Несколько программ событий могут быть оформлены и в виде одной процедуры с дополнительными входами:

```
EV1: PROCEDURE;
. . . . . .
     RETURN;
EV2: ENTRY;
     . . . . .
     RETURN;
EV7; ENTRY;
     . . . . .
     END.
```

Обязательным условием в этом случае будет отсутствие в такой процедуре операторов *CALL EVk* , где *k* - номера событий, программы которых объединены в одну процедуру. Это условие связано с устранением возможности рекурсий программ.

Булевские программы-функции, необходимые для организации интеррогативного управления, оформляются в виде подпрограмм-функций с именами *CN1* , ..., *CN1ϕ*:

```
CN5: PROCEDURE BIT(1);
     . . . . . .
     DECLARE A BIT(1);
     RETURN(A); END.
```

Фактические параметры всех операторов императивного и интеррогативного управления, рассмотренных ранее, задаются по правилам, принятым в ПЛ/I ДОС ЕС, и должны иметь атрибуты *DECIMAL FLOAT* (6). При этом возможно задание этих параметров с использованием операторов присваивания:

```
EV=4, S=5; CALL CREAS (EV,S);
```

или непосредственно:

```
CALL CREAS(4Eϕ, ϕ.5E1).
```

Для переменных, присутствующих в нескольких программах событий или булевских программах-функциях и описанных в них с атрибутом *EXTERNAL*, запрещается использовать знак *@* в качестве начальной буквы идентификатора. Включение в любую программу оператора *DECLARE (@STIME,@PRD) EXTERNAL* делает доступным в ней значение системного времени *@STIME* и значение периода роста системного времени *@PRD*.

Главная процедура должна содержать *CALL @INITC(TSIM,P,D@PL)*, где *TSIM*, *P* и *D@PL* определены. Только после выполнения этого оператора возможен доступ к подпрограммам монитора. Для начала процесса имитации необходимо ввести в главную процедуру оператор *CALL @START*.

Задание на выполнение имитационной программы оформляется с использованием обычных управляющих карт ДОС ЕС. В случае, когда используется монитор, программа которого набита на перфокартах, задание на моделирование в общем случае оформляется в следующем виде:

```
// JOB BEISP
// OPTION LINK
// UPSI ϕ1
// EXEC PL/I
BEISP: PROCEDURE OPTIONS (MAIN);
. . . . . .
END;
/*
```

```
// EXEC PL/I
EV1: PROCEDURE;
. . . . . .
END;
/*
// EXEC PL/I
EV2: PROCEDURE;
. . . . . .
END;
/*
. . . . . .
// EXEC PL/I
< модуль @INITC >
/*
// EXEC PL/I
< модуль @START >
/*
// EXEC LNKEDT
// EXEC
<исходные данные > ;
/*
/&
```

Компиляция отдельных модулей программы выполняетоя отдельно, что можно использовать для нахождения невыявленных ошибок программиста. Программа же имитации в целом может быть сформирована только после предварительной компиляции отдельных подпрограмм.

Если необходимость в получении распечатки исходных модулей *@INITC* и *@START* монитора отсутствует, в операторе *OPTION*, предшествующем шагам задания по трансляции этих модулей, задаются режимы *LINK, NOLIST*: *// OPTION LINK, NOLIST*. Перед компиляцией пользователя режим *NOLIST* отменяется оператором *OPTION* с режимами *LINK , LIST* :
// OPTION LINK, LIST.

Предлагаемый способ организации моделирующей программы обладает рядом достоинств:

простота реализации в рамках ДОС;

модульность имитационной программы;

возможность использования для написания программ как языка ПЛ/I, так и языков АССЕМБЛЕР, ФОРТРАН.

При использовании монитора, программа которого записана на МЛ, оформление задания на выполнение программы имитации может быть осуществлено следующим образом:

```
// JOB BEIS P2
// OPTION LINK
// UPSI 01
// EXEC PL/I
BEISP: PROCEDURE OPTIONS (MAIN);
. . . . . .
END;
/*
// EXEC PL/I
EV1: PROCEDURE;
. . . . . .
END;
/*
. . . . . .
// ASSGN SYSIPT, X'280'
INCLUDE
INCLUDE
// EXEC LNKEDT
// EXEC
< исходные данные >
/*
/&
```

Здесь предполагалось, что модули @INITC и @START набора CREATE в объектном виде, т.е. после компиляции, были записаны один за другим на магнитную ленту. При выполнении задания эта МЛ должна быть установлена на магнитофоне с адресом X'280' . На очередном шаге задания с помощью оператора // ASSGN логическому устройству "системный ввод" временно назначается адрес X'280', и управляющие операторы редактора INCLUDE включают объектные модули @INITC и @START в общую программную фазу.

6. ИМИТАЦИЯ СТОХАСТИЧЕСКИХ ЯВЛЕНИЙ. ПРОГРАММНЫЕ ДАТЧИКИ СЛУЧАЙНЫХ ЧИСЕЛ

В состав системы CREATE в описываемом варианте включен ряд программных датчиков случайных чисел с различными распределениями. Соответствующие программы при работе обращаются также к включенному в CREATE датчику равномерно распределенной случайной величины, принадлежащей интервалу (0,1). Ниже приведена его программа, реализующая алгоритм, предложенный в работе [1].

```
(NOFIXEDOVERFLOW):
@RAND: PROCEDURE (R);
       N=1220703125*N;
       IF N<0 THEN N=N+2147483647+1;
       R=N; R=R*0.04656613E-9;
       RETURN;
@INITR: ENTRY (NP);
       N=NP;
DECLARE NP  BINARY FIXED(31), R DECIMAL FLOAT(6),
        N   BINARY FIXED(31) STATIC;
       END;
```

Первоначальное обращение к этой программе служит для инициализации механизма получения псевдослучайной величины:

```
CALL @INITR(IY);
```

(здесь *IY* должно быть описано с атрибутами *BINARY FIXED*(31) и определено пользователем как произвольное нечетное число в диапазоне от 1 до $2^{31}-1$), последующие обращения - для получения реализаций этой величины *R* :

```
CALL @RAND(R) .
```

Далее перечисляются включенные в CREATE программы получения псевдослучайных величин с законами распределения, отличными от равномерного в интервале (0,1)*. Все фактические параметры этих

---

* С методикой построения подобных алгоритмов можно познакомиться, например, в работах [I], [II], [I3].

программы должны иметь атрибуты *DECIMAL FLOAT* (6).

1. *@UNIF(A,B,X)* - доставляет реализацию X случайной величины, равномерно распределенной в интервале (A,B).

2. *@NORM(A,ST,X)* - доставляет реализацию X случайной величины, распределенной по нормальному закону со средним A и стандартным отклонением ST.

3. *@ERLNG(L,K,X)* - доставляет реализацию X случайной величины, распределенной по закону Эрланга K-го порядка со средним $1/L$ [14].

4. *@RINT(A,B,X)* - доставляет X - с равной вероятностью одно из целых значений $A, A+1, \ldots, B-1, B$.

5. *@PASCL(M,P,X)* - доставляет реализацию X случайной величины, распределенной по закону Паскаля (типичное толкование: $P\{X\}$ есть вероятность того, что M-й успех наступит самое большее после $M+X-1$ испытаний по схеме Бернулли при вероятности успеха P).

6. *@BINOM(N,P,X)* - доставляет реализацию X случайной величины, распределенной по биномиальному закону, т.е. X - число "успехов" после N испытаний по схеме Бернулли при вероятности успеха P.

7. *@PSSN(LP,X)* - доставляет реализацию X случайной величины, распределенной по закону Пуассона с параметром LP.

8. *@LINAP(N,A,B,X)* - доставляет реализацию X случайной величины, распределенной в соответствии с интегральной функцией F, получающейся линейной интерполяцией из неравношаговой таблицы, определенной массивами A и B так, что $B(i)=F(A(i))$. Предполагается, что A и B суть одномерные массивы одинаковой длины $N \geq 2$, что первый и последний элементы массива B равны соответственно нулю и единице и что $B(i) \geq B(j), A(i)>A(j)$ при $i > j$.

9. *@HIST(N,G,P,X)* - доставляет реализацию X случайной величины, распределенной в соответствии с законом, определяемым гистограммой. Массив G должен иметь размерность $N+1$, $G(i)$ интерпретируются как упорядоченные по возрастанию границы классовых интервалов гистрограммы. Массив P должен иметь размерность N, причем $\sum_{i=1}^{N} P(i) = 1$. $P(i)$ интерпретируется как вероятность попадания случайной величины в интервал $(G(i), G(i+1))$.

Инициализация датчика *@RAND* необходима при использовании

любого из описанных выше генераторов псевдослучайных чисел, поскольку, как уже отмечалось, каждый из них использует механизм датчика *@RAND*.

При имитации стохастических явлений, происходящих в аналоговой части моделируемых систем, нередко приходится строить программные датчики реализаций (отсчетов) случайных процессов. При этом механизм имитации подобных отсчетов основан на соотношениях теории временных рядов [15]. Наиболее распространенными алгоритмами имитации отсчетов случайных процессов являются алгоритмы авторегрессий. Ниже приводится программа имитации отсчетов марковского случайного процесса, основанная на использовании алгоритма авторегрессии первого порядка [16]:

$$(X - EX)_k = A(X - EX)_{k-1} + B \cdot R.$$

В соответствии с этим алгоритмом идентификатору X присваивается значение реализации отсчета стационарного случайного процесса со средним EX, дисперсией $D = B^2/12(1-A^2)$ и корреляционной функцией вида

$$COR(\tau) = e^{-\gamma\tau}, \quad \tau > 0$$

где $\gamma = -(\ln A)/TT$,

TT – шаг дискретизации процесса;

R – равномерно распределенная в интервале [0,1] случайная величина.

Текст программы:

```
MPR.PROCEDURE(A,B,EX.X);      /* Датчик отсчетов стацио-
нарного марковского процесса */
CALL @RANDU(R); X = EX + A*(X - EX) + B*R; END.
```

(При первом обращении к подпрограмме X должно быть присвоено начальное значение реализации процесса, например, $X = 0$, или $X = EX$). Параметры авторегрессии A и B однозначно определяются через D, γ и TT с помощью соотношений

$$A = \exp(-\gamma TT); \quad B = \sqrt{12D(1-A^2)}.$$

Более сложные алгоритмы имитации случайных процессов обсуждаются в работе [17], задача составления по ним подпрограммы имитации не представляет особой сложности.

7. ОБРАБОТКА РЕЗУЛЬТАТОВ МОДЕЛИРОВАНИЯ И ОТЛАДКА ПРОГРАММЫ ИМИТАЦИИ

Подпрограммы *@EGET(EV), @EPUT(EV), @EEST(EV)* монитора CREATE обеспечивают организацию автоматического сбора, обработки и стандартную выдачу статистики потоков событий в модели.

@EGET(EV) - "организовать сбор статистики потока событий типа *EV*". После вызова подпрограммы монитор производит сбор статистических данных о потоке: образуемом событием типа *EV* без каких-либо других инструкций и указаний пользователя. Одновременно статистика может собираться не более чем для десяти различных типов событий.

Если во время автоматического сбора статистики событий происходит переход в новый период роста системного времени (см. § I), то на печать выдаются результаты сбора и обработки статистики за истекший период роста системного времени и подготавливается возможность сбора статистики в новом периоде. Сбор статистики в новом периоде осуществляется совершенно независимо от прошлого.

@EEST(EV) - "прекратить сбор статистики потока событий типа *EV*". Сбор статистики прекращается, на печать выдается стандартный протокол результатов сбора и обработки статистики потока событий типа *EV*.

Если необходимо лишь выдать на печать текущее содержание протокола сбора и обработки потока событий типа *EV*, а затем продолжать сбор статистики потока этих событий, то необходимо вызвать программу *@EPUT(EV)*.

Выполнение оператора *CALL @ALLST* приводит к выдаче на печать стандартного протокола статистики потоков всех типов событий, для которых монитор осуществляет сбор статистики на момент выполнения указанного оператора. Сбор статистики не прекращается.

Сбор и обработка других результатов моделирования осуществляется в соответствии с хорошо известными алгоритмами математической статистики, вывод результатов производится с использованием стандартных возможностей языка ПЛ/I ДОС ЕС. Эти вопросы здесь не обсуждаются.

В целях отладки имитационной программы последовательность выполнения программ событий и булевских программ функций, реали-

зующаяся в процессе моделирования, может регистрироваться на устройстве печати в виде текстов, содержащих следующие сведения:

1) о виде выполняемой программы (например, "событие 5", "булевская функция 9 (7)" - т.е. выполняется булевская программа-функция 9, управляющая событием 7);

2) о текущих значениях системного времени и периода роста системного времени (*PCB*);

3) о всех обращениях из выполняемой программы к подпрограммам монитора с указанием значений фактических параметров вызываемых процедур.

Для организации подобной регистрации (трассировки) могут быть использованы следующие подпрограммы:

@TRACE(T1,P1,T2,P2) - "организовать трассировку выполнения всех программ событий и булевских программ-функций модели от момента системного времени *T1* периода *P1 PCB* до момента *T2* периода *P2 PCB* ";

@TREV(EV) - "организовать трассировку выполнения программы события типа *EV* ", трассировка будет продолжаться до конца имитации, либо до вызова подпрограммы *@NTREV(EV)* (одновременно этим способом может осуществляться трассировка не более 5 программ);

@TRCN(CN) - "организовать трассировку выполнения булевской программы-функции с номером *CN* ", трассировка будет продолжаться до конца имитации, либо до вызова подпрограммы *@NTRCN(CN)* (одновременно этим способом может проводиться трассировка не более 5 булевских программ-функций).

8. ПРИМЕР МОДЕЛИ СИСТЕМЫ МАССОВОГО ОБСЛУЖИВАНИЯ

С чисто иллюстративной целью рассмотрим следующую систему.

На вычислительный центр (ВЦ), оснащенный двумя ЭВМ, поступает поток заявок на расчеты. При наличии хотя бы одной свободной ЭВМ расчеты, связанные с выполнением поступившей заявки производятся немедленно. В случае занятости обеих машин, заявка ставится в очередь. Очередь ограничена. Если к моменту поступления заявки на ВЦ очередь занята полностью, заявка получает отказ в обслуживании.

Смоделировать работу вычислительного центра с целью определения среднего времени выполнения заявки на расчеты и вероятности получения отказа в обслуживании.

В терминах систем массового обслуживания данная система может быть классифицирована как двухканальная с общей очередью ограниченного объема [16].

Не нарушая общности дальнейшего рассмотрения, примем следующие допущения:

1) поток заявок на ВЦ - простейший, интенсивности A ;

2) время расчетов на ЭВМ-1 - равномерно-распределенная случайная величина в интервале $[T1, T11]$;

3) время расчетов на ЭВМ-2 - равномерно-распределенная случайная величина в интервале $[T2, T22]$;

4) объем очереди ограничен величиной MQ .

(Три первых допущения, естественно, могут быть связаны с выбором совершенно других статистических моделей. На общую структуру программы имитации это не повлияет, изменятся лишь соответствующие алгоритмы моделирования случайных величин).

Для моделирования указанной системы выделим следующие события:

$EV1$: /* Поступление заявки на ВЦ */;

$EV2$: /* Конец расчетов по заявке на ЭВМ-1 */;

$EV3$: /* Конец расчетов по заявке на ЭВМ-2 */.

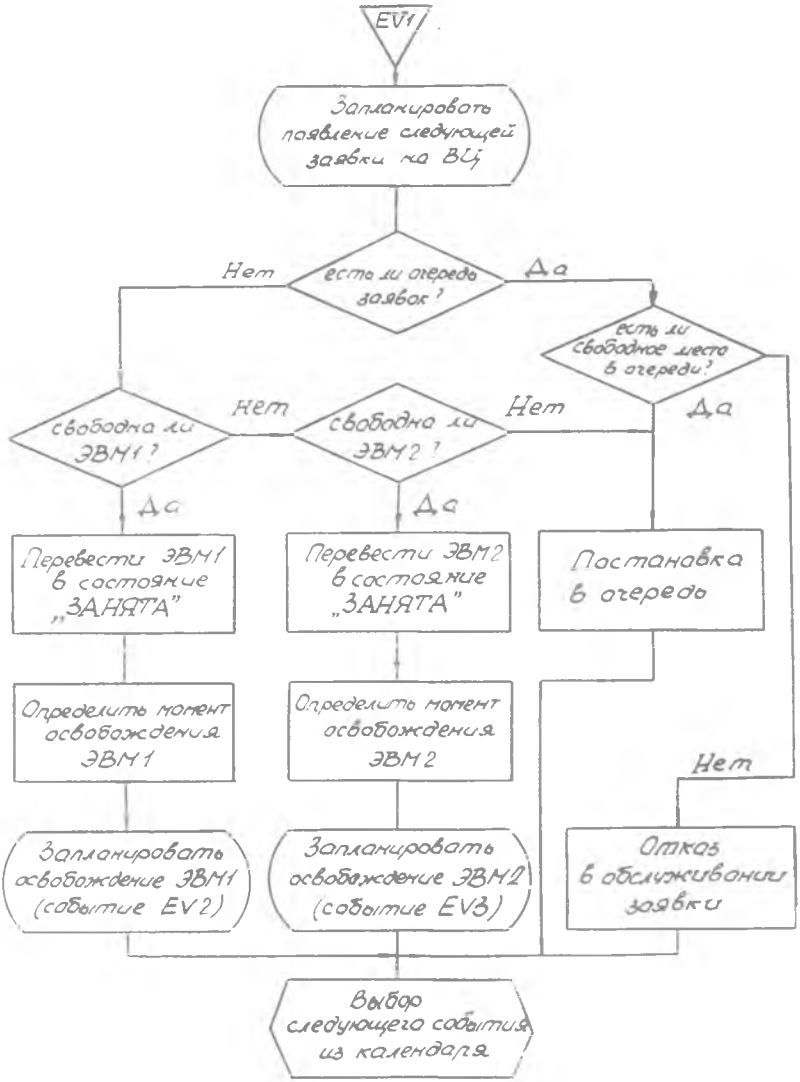
Ниже приведены блок-схемы программ этих событий. (Блок-схема программы события $EV3$ аналогична $EV2$). Блок-схемы программ, приведенные на рис. 8, 9, отражают всю логику работы моделируемой системы и не требуют дополнительных пояснений, следует лишь заметить, что блоки

(Запланировать)

связаны с внесением в календарь меток тех или иных требуемых в процессе имитации событий. В программе моделирования эти блоки связаны с использованием того или иного оператора императивного управления. Блоки

<Выбор события>

связаны с передачей управления из программы пользователя монитору, который осуществляет пересчет системного времени и передает



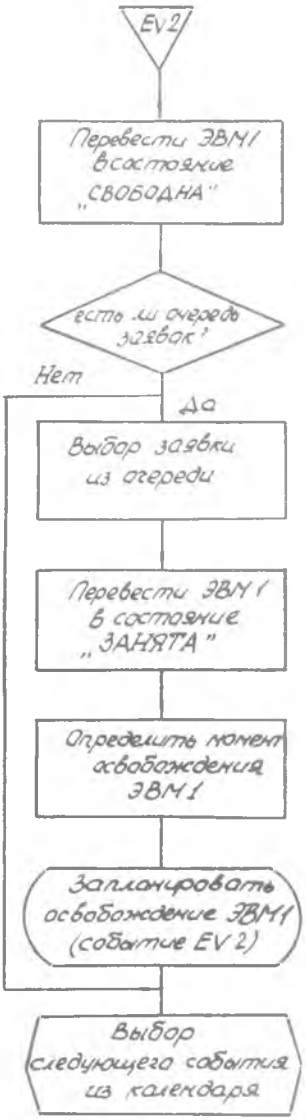
Р и с. 8. Блок-схема программы события *EV1* : "Поступление заявки на ВЦ"

в свою очередь управление ближайшему по времени его возникновения событию.

Подчеркнем, что на приведенных блок-схемах программ событий нашли свое отражение только те операции, которые никак не связаны с ходом системного времени(эти операции происходят в системе по предположению мгновенно).

Ниже приводится программа имитации данной системы с необходимыми комментариями к ней. Программа составлена в соответствии с блок-схемами рис. 8, 9.

```
BEISP:PROCEDURE OPTIONS(MAIN);
DECLARE(P2,ZP,ST2,Z) EXTERNAL;
/* Z  - общее число заявок, участвующих
     в имитации */
GET EDIT (Z)(F(3)); GET EDIT (A,T1,T11,
T2,T22)(E(10,3)); DECLARE @P1(20)
EXTERNAL; CALL @INITC(1E3,1E4,
2E1):
/* Время моделирования P*TSIM задается
   заведомо большим, чем необходимо  ВЦ
   для обработки Z  заявок */
CALL EV ;          /* Установка начальных
                      условий */
CALL EV1;          /* Приход первой заяв-
                      ки на ВЦ. Внесение
                      первых событий  в
                      календарь */
CALL @START ;      /* Начало процесса
                      имитации */
```



Р и с. 9. Блок-схема программы события *EV2*: "Конец расчетов по заявке на ЭВМ-I"

```
PUT EDIT('СРЕДНЕЕ ВРЕМЯ:',STZ/ZP)(SKIP,A,E(10,3));
PUT EDIT('ВЕРОЯТНОСТЬ ОТКАЗА:',PZ/Z)(SKIP,A,E(10,3));
END;
EV: PROCEDURE;   /* Начальные условия модели */
DECLARE (F1,F2) BIT(1),Q(50),IY BINARY FIXED(31),
@STIME,STZ,ZP,PZ,Z) EXTERNAL;
/* F1,F2  - флажки занятости машин. Обе ЭВМ первоначально свободны*/ F1='0'B; F2='0'B;
/* JZ  - счетчик заявок, поступающих на ВЦ */ JZ=0;
/* MQ  - объем очереди заявок */ MQ=50;
/* N,K  - указатели начала и конца очереди заявок */ K=1; N=2;
/* JQ  - счетчик заявок, стоящих в очереди */ JQ=0;
/* STZ  - вспомогательная переменная. Общее время пребывания на ВЦ
        всех заявок, принятых к обработке */ STZ=0;
/* ZP  - счетчик заявок, обработанных на ВЦ */ ZP=0;
/* PZ  - количество отказов в обслуживании */ PZ=0;
/* TH1,TH2  - вспомогательные переменные. TH1 - время начала
        расчетов по очередной заявке на ЭВМ-I, TH2 - на
        ЭВМ-2 */;
/* Q (50) - массив, имитирующий очередь */;
/* Инициализация датчика случайных чисел */
   IY = I22070325; CALL @INITR(IY); RETURN;
EV1: ENTRY;      /* Поступление заявки на ВЦ */;
/* Счет поступающих заявок, проверка условия окончания моделирования */ JZ=JZ+1; IF JZ>Z THEN
RETURN;
/* Планирование появления следующей заявки на ВЦ */ EV=1;
CALL @RAND(R); T=-(1/A*LOG(R)); CALL CREAT(EV,T);
/* Есть ли очередь заявок ? */ IF JQ=0 THEN DO;
/* Свободна ли ЭВМ-I? */ IF F1='0'B THEN DO;
```

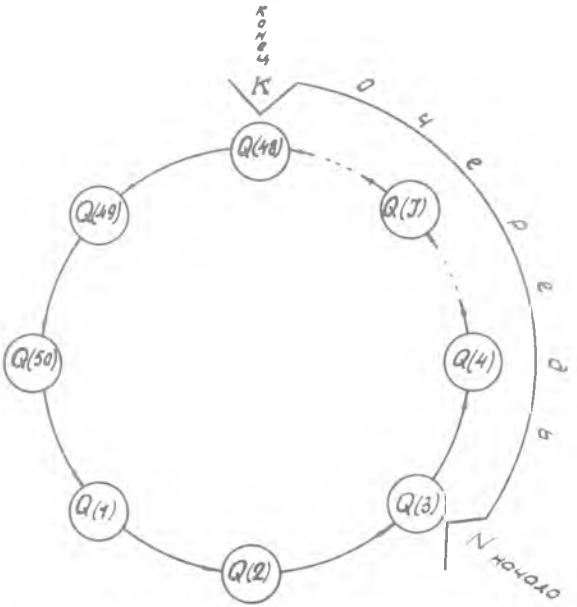
```
/* Занимаем ЭВМ-I расчетами по заявке, определяем момент ее освобождения и планируем событие EV2 */ F1 = '1'B;
TH1=@STIME; CALL @RAND(R); T=T1+(T11-T1)*R;
EV=2; CALL CREAT(EV,T); RETURN; END;
/* Свободна ли ЭВМ-2? */ IF F2='0'B THEN DO;
/* Занимаем ЭВМ-2 расчетами по заявке, определяем момент ее освобождения и планируем событие EV3*/* F2='1'B;
TH2=@STIME; CALL @RAND(R); T=T2+(T22-T2)*R;
EV=2, CALL CREAT(EV,T): RETURN; END; END;
/* Есть ли свободное место в очереди? */IF JQ=MQ THEN DO;
/ * Отказ в обслуживании заявки */ PZ=PZ+1; RETURN; END·
/* Постановка в очередь */ JQ=JQ+1; IFK=MQ THEN K=1;
ELSE K=K+1; Q(K)=@STIME; RETURN;
EV2:ENTRY; /* Конец расчетов по заявке на ЭВМ-I */
/* Освобождаем ЭВМ-I и накапливаем общее время ST2 */
F1='0'B; ST2=ST2+(@STIME-TH1); ZP=ZP+1;
/* Есть ли очередь заявок? */ IF JQ=0 THEN RETURN;
/* Выбор заявки из очереди */ TH1=Q(N); JQ=JQ-1;
IFN=MQ THEN N=1; ELSE N=N+1;
/* Занимаем ЭВМ-I расчетами по заявке, определяем момент ее освобождения и планируем событие EV2 */ F1='1'B; CALL @RAND(R);
T=T1+(T11-T1)*R; EV=2; CALL CREAT(EV,T); RETURN;
EV3: ENTRY; /* Конец расчетов по заявке на ЭВМ-2 */
/* Программа аналогична EV2 */ F2='0'B;
ST2=ST2+(@STIME-TH2); ZP=ZP+1;
IF JQ=0 THEN RETURN TH2=Q(N), JQ=JQ-1;
IFN=MQ THEN N=1; ELSE N=N+1;
F2='1'B; CALL @RAND(R); T=T2+(T22-T2)*R;
EV=3; CALL CREAT(EV,T); END.
```

Дополнительных общих пояснений программа не требует. Отметим лишь, что механизм планирования событий (занесения событий в календарь), используемый в программе, связан с обращением к датчику случайных чисел и использованием алгоритмов имитации случайных величин (см. § 6), в значительной степени традиционных для задач статистического моделирования [16].

Механизм моделирования очереди заявок, поступающих на ВЦ, требует некоторых дополнительных пояснений. Как уже отмечалось выше, очередь общая (для обеих ЭВМ), ограниченного объема. Дисциплина очереди: "первым пришел - первым будешь обслужен". При моделировании очереди заявок используются идентификаторы: MQ, JQ, N, K, смысл которых определен комментариями к программе. Очередь имитируется в массиве $Q(50)$. N - указатель первого элемента, стоящего в очереди, K - последнего. Число элементов массива определяет максимальную длину очереди, а содержание того или иного элемента определяется временем поступления в очередь соответствующей заявки на расчеты: $Q(K)$=@$STIME$. Массив $Q(50)$ закольцован таким образом, как показано на рис. 10, при этом элемент массива $Q(N)$ всегда определяет время поступления на ВЦ заявки, стоящей в очереди первой, а $Q(K)$ - время поступления на ВЦ заявки, стоящей в очереди последней.

В процессе имитации N и K могут только увеличиваться, но после достижения своего максимального значения MQ вновь сбрасываются в единицу, таким образом очередь заявок моделируется в информационной структуре, соответствующей структуре циклического линейного списка [18]. Начальные значения указателей K = 1, N = 2 выбраны таким образом, что при постановке в очередь первой заявки после выполнения оператора $K=K+1$ оказывается $K=N$ (начало и конец очереди совпадают - в очереди одна заявка).

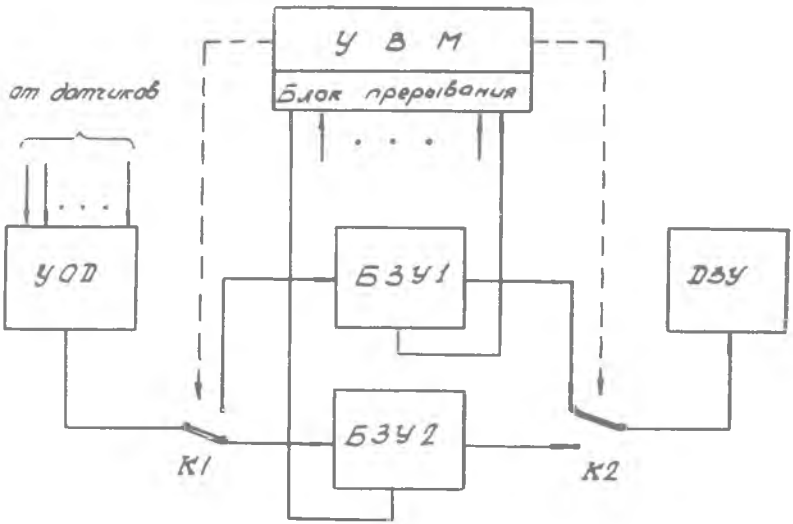
Имитационная программа для этого примера может быть оформлена и другим образом (с использованием других средств CREATE), в этом плане все определяет опыт и искусство программиста. Здесь мы несколько пожертвовали эффективностью программы ради ее наглядности и удобства чтения.



Р и с. IO. Кольцевая структура модели очереди

9. ПРИМЕР МОДЕЛИ ПОДСИСТЕМЫ СБОРА И ПРЕДВАРИТЕЛЬНОЙ ОБРАБОТКИ ИНФОРМАЦИИ

В качестве второго примера рассмотрим структуру подсистемы САЭ, представленную на рис. II. Это наиболее распространенная на практике структура системы сбора экспериментальных данных с использованием двухтактного режима работы буферных запоминающих устройств(БЗУ), когда один из буферов принимает данные от устройства опроса датчиков (УОД), второй в это же время переписывает информацию в долговременное запоминающее устройство (ДЗУ) [I9]. В случае, когда БЗУ, находящееся в режиме приема информации, заполнено на x % своего объема, на регистр прерываний УВМ выставляется сигнал запроса на переключение ключей $K1$, $K2$. Поскольку УВМ



Р и с. II. Структура управляемого двухтактного буфера

в САЭ выполняет самые различные функции по обработке результатов эксперимента и управлению, мгновенно отреагировать на запрос БЗУ в общем случае УВМ не может. Проходит некоторое время $T1$ между появлением запроса и переключением ключей $K1$, $K2$. За время $T1$ приемное БЗУ может заполниться "более чем на 100% своего объема", что связано с потерями информации. Моделирование этой структуры может проводиться, например, с целью определения α – уровня заполнения БЗУ, при котором вероятность потерь информации принимает заданное значение.

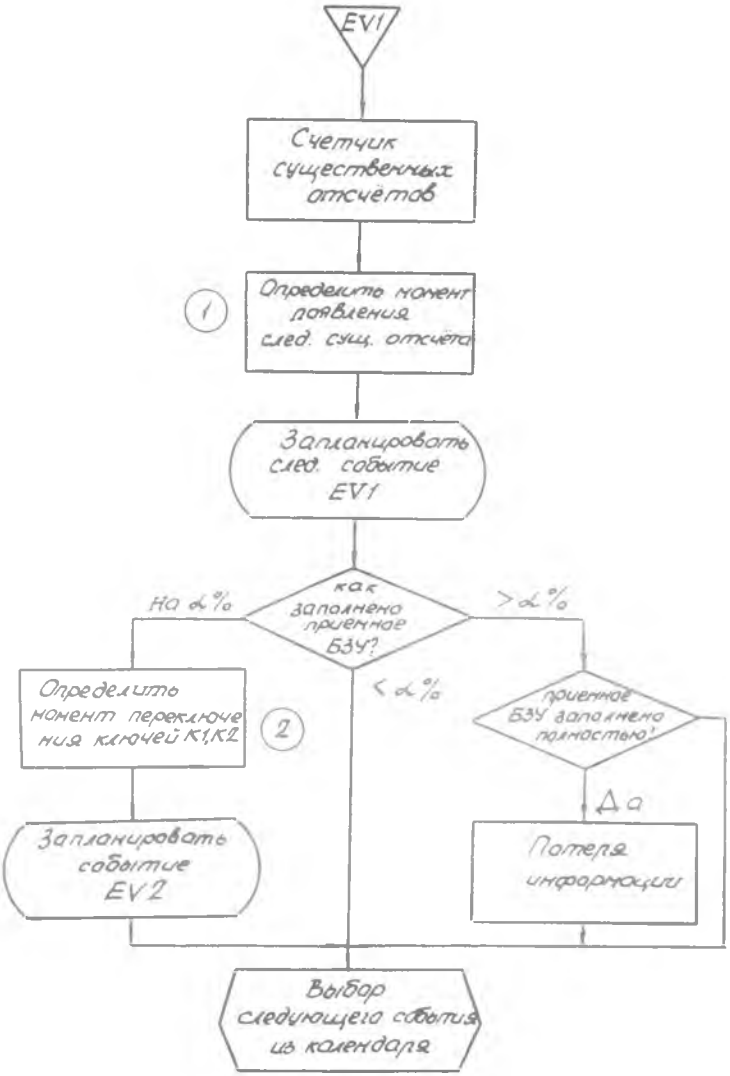
УОД может представлять из себя, например, локальный коммутатор с предсказателем нулевого порядка, выходной информацией которого являются "существенные" отсчеты от каждого из датчиков.

При разработке программы имитационного моделирования данной структуры выделим следующие типы событий:

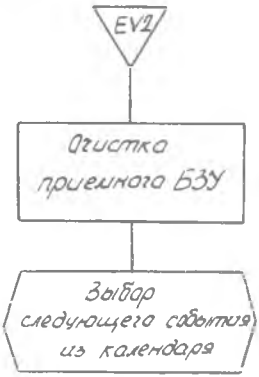
$EV1$: /* Появление существенного отсчета на выходе УОД */;

$EV2$: /* Переключение ключей */.

Структура программы события $EV1$ поясняется блок-схемой рис. I2. Блоки I, 2 функционируют по обычным алгоритмам статистического моделирования. (Закон распределения вероятностей вели-



Р и с. I2. Блок-схема программы события *EV1* : "Появление существенного отсчета на выходе УОД"



Р и с. I3. Блок-схема программы события *EV2*: "Переключение ключей"

чины *T1* выбирается при моделировании с учетом специфики работы конкретной УВМ в составе САЭ). Программа события *EV2* (рис. I3) имитирует "очистку" приемного БЗУ и переключение ключей *K1* , *K2*.

Ниже приведен текст программы имитационного моделирования с необходимыми комментариями. (Алгоритмы имитации случайных величин *T* и *T1* в программе опущены. Подразумевается, что вместо многоточия могут быть использованы конкретные алгоритмы статистического моделирования, определяемые видом законов распределения вероятностей соответствующих случайных величин).

```
BEISP: PROCEDURE OPTIONS(MAIN);
/* Z  - общее число "существенных" отсчетов, участвующих в имитации */ GET EDIT(Z)(F(3)); DECLARE @PL(2Ø)EXTERNAL;
CALL @INITC(1E3,1E4,2E1) :   /* Инициализация монитора */;
CALL EV;   /* Установка начальных условий */;
CALL EV1;  /* Появление первого "существенного" отсчета на выходе УОД */;
CALL @START;   /* Начало имитации */;
PUT EDIT   ('Вероятность потерь информации:',PZ/Z );
(SKIP,A,E(1Ø,3));
DECLARE(PZ,Z)EXTERNAL; END;
EV: PROCEDURE;   /* Начальные условия */;
/* JZ    - счетчик отсчетов, поступающих с УОД */ JZ=Ø;
/* JP    - счетчик отсчетов, размещаемых в приемном БЗУ */ JP=Ø;
/* V     - объем буферного запоминающего устройства */ V=4Ø;
/* PZ    - счетчик потерянных отсчетов */ PZ=Ø;
/* A     - уровень заполнения БЗУ */ A=35;
DECLARE(PZ,Z)EXTERNAL; RETURN;
```

```
EV1:          /* Появление существенного отсчета */ ENTRY;
JZ=JZ+1; JP=JP+1; IF JZ>Z THEN RETURN;
EV=1; T=···   /* T - интервал времени до появления следующего
существенного отсчета на выходе УОД */;
CALL CREAT(EV,T);
IF JP=A THEN DO; EV=2; T1=·····;
CALL CREAT(EV,T1); END;
IF JP>=V+1 THEN PZ=PZ+1; RETURN;
EV2: /* Переключение ключей */ ENTRY;
JP=0; END.
```

Эта же программа может быть составлена с использованием аппарата интеррогативного управления. По-прежнему определим события *EV1* и *EV2* и оформим дополнительно новое событие *EV3* : "Запрос от БЗУ на переключение ключей", управляемое интеррогативно с помощью функции *CN1* ;

```
BEISP: PROCEDURE OPTIONS(MAIN);
DECLARE @PL(20) EXTERNAL;
GET EDIT(z)(F(3)); CALL @INITC(1E3,1E4,2E1); CALL EV;
/* Введение интеррогативного управления */
CN=1; EV=3; CALL ITRR(EV,CN);
CALL EV1; CALL @START;
PUT EDIT      ('Вероятность потерь информации',PZ/Z
(SKIP,A,E(10,3));
DECLARE(z,PZ) EXTERNAL; END;
EV: PROCEDURE;
DECLARE(PZ,Z) EXTERNAL;
JZ=0; JP=0; V=40; A=35; PZ=0;
RETURN;
EV1: ENTRY; JZ=JZ+1; JP=JP+1; IF JZ>Z
THEN RETURN; EV=1; T=. . . . . ;
CALL CREAT(EV,T); IF JP>=V+1
THEN PZ=PZ+1; RETURN;
EV2: ENTRY; JP=0; RETURN;
```

```
EV3: ENTRY; EV=2; T1= . . . . . ;
CALL CREAT(EV, T1); RETURN;

CN1: ENTRY BIT(1);
DECLARE C BIT(1), IFJP=A THEN;
C= '1'B; ELSE C='0'B; RETURN(C);
END.
```

При этом программа события *EV3* : "Запрос от БЗУ на переключение ключей" будет вызываться каждый раз, когда приемное БЗУ заполнено до уровня *A* .

Приведенные программы являются иллюстративными. Для имитации функционирования простых систем, возможно, нет необходимости в использовании специализированных программных средств моделирования. Удобства, предоставляемые пользователю этими средствами, можно в полной мере оценить при написании и отладке имитационных программ достаточно сложных систем, состоящих из многих компонентов, функционирующих в условиях жесткой (синхронной или асинхронной связи друг с другом).

Литература

1. Нейлор Т. Машинные имитационные эксперименты с моделями экономических систем. М., "Мир", 1975, с. 500.
2. Калиниченко Л.А. СЛЭНГ - экспериментальный язык программирования, ориентированный на описание и моделирование вычислительных машин и систем. - В сб.: Теория автоматов. Киев, Институт кибернетики АН УССР. Вып. I, 1967, с. 34-42.
3. Глушков В.М. и др. Программные средства моделирования непрерывно-дискретных систем. Киев, "Наукова думка" 1975, с. 150.
4. Калиниченко Л.А., Щербин В.М. СКИФ - независимый от языков программирования пакет программ, ориентированный на имитационное моделирование систем с дискретными событиями. "Управляющие системы и машины", 1976, № 4, с. 37-42.

5. Скотт Р., Сэндак Н. ПЛ/I для программистов. М., "Статистика", I977, с. 222.
6. Бусленко И.П., Калашников В.В., Коваленко И.Н. Лекции по теории сложных систем. М., "Советское радио", I973, с. 440.
7. Марковиц Г., Хаусснер Б., Карр Г. СИМСКРИПТ. Алгоритмический язык для моделирования. М., "Советское радио", I966, с. I52.
8. Дал У.-И., Мюрхауг Б., Нюгорд К. СИМУЛА-67. Универсальный язык программирования. М.,"МИР", I969, с. I60.
9. Мартин Ф. Моделирование на вычислительных машинах. М., "Советское радио", I972, с. 287.
I0. Дал У.-И. Языки для моделирования систем с дискретными событиями. - В сб.: Языки программирования. М., "Мир", I972, с. 344-403.
II. Бусленко Н.П. Моделирование сложных систем. М., "Наука", I968, с. 356.
I2. Джермейн К. Программирование на IBM/360. М.,"Мир", I973, с. 870.
I3. Соболь И.М. Численные методы Монте-Карло. М.,"Наука" I973, с. 3II.
I4. Тараканов К.В., Овчаров Л.А., Тырышкин А.Н. Аналитические методы исследования систем. М., "Советское радио", I974, с. 240.
I5. Бокс Дж., Дженкинс Г., Анализ временных рядов. Прогноз и управление. Вып. I, М., "Мир", I974, с. 406.
I6. Дерябкин В.П., Кораблин М.А. Моделирование систем на ЭВМ. Куйбышевский авиационный институт, I977, с. 82.
I7. Быков В.В. Цифровое моделирование в статистической радиотехнике. М., "Советское радио", I97I, с. 328.
I8. Кнут Д. Искусство программирования для ЭВМ. Т.I. Основные алгоритмы. М., "Мир", I976, с. 736.
I9. Соучек Б. Мини-ЭВМ в системах обработки информации. М., "Мир", I976, с. 520.

Приложение I

ФУНКЦИОНАЛЬНЫЕ МОДУЛИ МОНИТОРА

№№ пп	Имя программы со списком формальных параметров	Функции	Допустимые значения фактических параметров *
I	*@INITC(TSIM, P, D@PL)*	Инициализирует монитор	$0 \leqslant TSIM \leqslant 7{,}2 \cdot 10^{75}$ $1 \leqslant P \leqslant 32766$ $10 \leqslant D@PL \leqslant 32767$ **
2	*@START*	Осуществляет запуск программы имитации	
3	*CREAS(E,T)*	Планирует вызов события с номером E в момент времени T	$1 \leqslant E \leqslant 50$ $0 \leqslant T \leqslant 7{,}2 \cdot 10^{75}$
4	*CREASP(E,T)*	То же с приоритетом	

* Фактические параметры должны иметь атрибуты *DECIMAL FLOAT* (6); в таблице и во всех программах для одноименных параметров ограничения одинаковы.

** Для правильной работы имитационной программы необходимо выполнение более жесткого условия $10 \leqslant D@PL \leqslant n$, где n – размерность массива *@PL*, объявляемого при инициализации монитора (см. § I).

Продолжение таблицы

№№ пп	Имя программы со списком формальных параметров	Функции	Допустимые значения фактических параметров *
5	*CREAT(E,T)*	Эквивалентна *CREAS(E,T+@STIME)*	
6	*CREATP(E,T)*	То же с приоритетом	
7	*CRSF(E,T,PJ)*	Планирует вызов события *E* в момент времени *T* *PJ*-го периода РСВ	*1≤PJ≤32767*
8	*CRSFP(E,T,PJ)*	То же с приоритетом	
9	*CREABF(E,EJ)*	Планирует вызов события *E* перед событием *EJ* ***	*1≤EJ≤50*
10	*CREAAF(E,EJ)*	То же вслед за событием *EJ*	
11	*CANCEL(EJ)*	Исключает событие *EJ* из календаря будущих событий	
12	*ITRR(E,C)*	Вводит для события *E* интеррогативное управление, осуществляемое булевой программой-функцией с номером *C*	*1≤C≤10*
13	*ABITRR(C)*	Отменяет интеррогативное управление, осуществляемое булевой программой-функцией *C*	

*** Параметр *EJ* здесь и далее определяет номер события, имеющего наименьшее время из всех событий с тем же номером, хранящихся в календаре событий, т.е. "ближайшего в будущем" из всех событий с номером *EJ* .

Продолжение таблицы

№№ пп	Имя программы со списком формальных параметров	Функции	Допустимые значения фактических параметров *
I4	@NEXT(EJ,EI,TI,PI)	Доставляет EI-номер, TI-время, PI -период РСВ события, запланированного вслед за событием EJ (если EJ=0, то выдаются характеристики события, следующего за текущим событием)	EI, TI, PI – определяются монитором
I5	@STEV(EJ,TI,PI)	Доставляет TI-время, PI -период РСВ события EJ	
I6	@EGET(E)	Назначает сбор статистики потока события E	
I7	@EEST(E)	Прекращает сбор статистики потока события E	
I8	@EPUT(E)	Выдает на SYSLIST статистику потока события E	
I9	@ALLST	Выдает на SYSLIST всю собранную статистику	
20	@STOP	Прекращает моделирование и выполняет функции @ALLST	
2I	@TRACE(T,P,TJ,PJ)	Планирует трассировку всех программ событий и булевых программ-функций, начиная с момента T периода P РСВ и кончая моментом TJ периода PJ РСВ	$0 \leq T \leq 7{,}2 \cdot 10^{75}$

Окончание таблицы

№№ пп	Имя программы со списком формальных параметров	Функции	Допустимые значения фактических параметров *
22	@TREV(E)	Планирует трассировку программы события E	
23	@NTREV(E)	Прекращает трассировку программы события E	
24	@TRCN(C)	Планирует трассировку булевой программы-функции C	
25	@NTRCN(C)	Прекращает трассировку булевой программы-функции C	
26	@ISOS	Планирует вызов программы @SOS в случае обнаружения монитором контролируемой ошибки пользователя	

Приложение П

СООБЩЕНИЯ МОНИТОРА

№№ пп	Текст сообщения *	Комментарий **	Номер программы-корреспондента (см. П I)
I	Моделирование начато	Выдается при обращении к *@START*	2
2	Моделирование окончено o ***	Выдается перед передачей управления оператору, следующему за оператором вызова *@START*	—"—
3	Число завершенных периодов РСВ равно ...	Выдается при переходе в новый период монотонного роста системного времени (РСВ)	—"—
4	Переполнение счетчика аппарата сбора статистики события ...	Сообщение сопровождается выдачей стандартного протокола статистики указанного события, после чего сбор статистики этого события возобновляется и производится независимо от прошлого	—"—

* Выдается на *SYSLIST*

**Расшифровку идентификаторов см. ПI

*** Кружком здесь и далее отмечаются сообщения, дополняемые информацией о текущих значениях системного времени и периода РСВ

Продолжение таблицы

№ пп	Текст сообщения	Комментарий	Номер программы-корреспондента (см. П1)
5	Бул. функция ... (...)°	Сообщение о вызове булевой программы-функции при трассировке. В скобках указывается номер управляемого события	21 или 24
6	Событие ...°	Сообщение о вызове программы события при трассировке	21 или 22
7	Параметр < верх. граница изменения @STIME > вне доп. диапазона	TSIM < 0	I
8	Параметр < число периодов РСВ > вне доп. диапазона	P < 1 или P > 32766	—— " ——
9	Параметр < размерность @PL > вне доп. диапазона	D@PL < 10 или D@PL > 32767	—— " ——
I0	Параметр(ы) < номер события > вне доп. диапазона	E < I или E > 50, либо EJ < I или EJ > 50	3-I2 I4-I8, 22, 23
II	Параметр < период РСВ > вне доп. диапазона	PJ < I или PJ > 32767	7, 8
I2	Параметр < номер бул. функции > вне доп. диапазона	C < 0 или C > I0	I2, I3, 24,25
I3	Переполнение СБС	Памяти, выделенной монитору при инициализации, недостаточно	3-I0

Окончание таблицы

№№ пп	Текст сообщения	Комментарий	Номер программы-корреспондента (см. П1)
I4	План.сбора статистики большего числа событий, чем допустимо	Одновременно статистика может собираться не более, чем о IO событиях	I6
I5	Статистика события ... не собиралась	Потребована выдача статистики события, сбор которой не производился	I7, I8
I6	План.тр.большего числа событий (бул. функций), чем допустимо	Одновременно индивидуальная трассировка может проводиться не более, чем для 5 программ событий и 5 булевых программ-функций	22, 24
I7	Ошибка при задании параметров	$T>TSIM$ или $TJ>TSIM$ или $P<1$ или $P>32766$ или $PJ<1$ или $PJ>32767$ или $P>PJ$ или $PJ=P$, но $T>TJ$; Станд. исправление: $T=@STIME$, $P=@PRD$, $TJ=TSIM$, $PJ=@DRD$	2I
I8	Авар.окончание	Дальнейшее выполнение имитационной программы невозможно	
I9	Вызов @SOS	Выдается перед передачей управления программе пользователя @SOS	

С О Д Е Р Ж А Н И Е

Михаил Александрович Кораблин
Сергей Викторович Смирнов

МОНИТОР ДЛЯ ИМИТАЦИОННОГО
МОДЕЛИРОВАНИЯ СИСТЕМ
С ДИСКРЕТНЫМИ СОБЫТИЯМИ

Учебное пособие

Редактор Н.Б. К а с а т к и н а
Техн.редактор Н.М. К а л е н ю к
Корректор Е.Д.А н т о н о в а

Подписано в печать 7.05.80. ЕО 00377. Формат 60x84 1/16
Бумага оберточная белая. Печать оперативная. Усл.п.л.3,25.
Уч.-изд.л. 3,0. Тираж 600 экз. Заказ № 3295 Цена 10 коп.

Куйбышевский ордена Трудового Красного Знамени
авиационный институт имени С.П.Королева,
г. Куйбышев, ул. Молодогвардейская, 151.

Областная типография им. В.П.Мяги,
г. Куйбышев, ул. Венцека, 60.