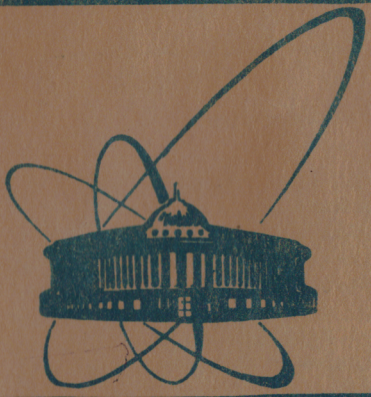

сообщения
объединенного
института
ядерных
исследований
дубна

Р10-80-423

А.И.Островной, И.М.Саламатин

ЯЗЫК ПРОГРАММИРОВАНИЯ ПРИКЛАДНЫХ СИСТЕМ АВТОМАТИЗАЦИИ ЭКСПЕРИМЕНТА

1980

Островной А.И., Саламатин И.М. P10-80-423

Язык программирования прикладных систем автоматизации эксперимента

Описан проблемно-ориентированный язык генерации прикладных систем автоматизации экспериментов - язык САНПО, который позволяет описывать автоматически исполняемые алгоритмы накопления и предварительной обработки информации, поступающей в реальном масштабе времени, имеет средства параллельного программирования, а также является средством диалогового взаимодействия экспериментатора и прикладной системы во время проведения эксперимента. Набор операторов языка открыт и может пополняться. Компилятор реализован на ЭВМ типа СМ-3.

Работа выполнена в Лаборатории нейтронной физики ОИЯИ.

Сообщение Объединенного института ядерных исследований. Дубна 1980

Ostrovnoj A.I., Salamatin I.M. P10-80-423

Programming Language of Applied Systems

1. ВВЕДЕНИЕ

В настоящее время мини-ЭВМ получили широкое распространение в области автоматизации научных исследований /АНИ/ и производственных процессов. Главным препятствием успешному внедрению этой техники являются трудности разработки прикладного программного обеспечения /ПО/. В работе [1] отмечалось, что решение проблемы создания такого ПО следует искать на пути перехода к промышленным методам его разработки.

В данной работе описан язык, ориентированный на определенную технологию производства прикладных систем АНИ, - язык системы САНПО [2] /система автоматического накопления и предварительной обработки информации, поступающей в реальном масштабе времени/. Конечным в принятой технологии выступает этап генерации прикладной системы. В то же время разработанный язык - это не только язык генерации. Существенным является отражение в языке САНПО алгоритмических структур, типичных для режима работы ЭВМ на линии с экспериментальной установкой. Эти структуры названы параллельно исполняемыми процессами. Операторы описания процессов компилируются, и результаты компиляции используются для соответствующей настройки ядра монитора системы САНПО, управляющего автоматически исполняемой частью алгоритма эксперимента. В этом плане язык САНПО выступает в качестве языка программирования работы монитора прикладной системы. Язык включает также инструкции, исполняемые в режиме интерпретации.

Работа выполнена для программирования экспериментов на реакторе ИБР-2 [3].

2. ОПИСАНИЕ ЯЗЫКА

2.1. Основные элементы языка включают понятия имени, константы, инструкции, метки и программы.

Именем является последовательность букв и цифр /не более шести символов/, начинающаяся буквой.

Константы возможны числовые и текстовые. Числовые константы являются числами типа INTEGER или REAL одно- или двукратной точности. Текстовые константы могут быть представлены в кодах ASCII [4] /не более восьми символов/ и RADIX-50 [4] /не более двенадцати символов/.

Инструкции языка формально представляются в виде строк длиной не более 72 символов, каждая из которых ограничена симво-

лами "Возврат каретки" /код 15/ и "Перевод строки" /код 12/. Признаком продолжения инструкции на следующей строке является последовательность символов /* , стоящих в конце текущей строки /перед символом "Возврат каретки"/.

Меткой является имя, сопровождаемое двоеточием и стоящее в начале инструкции.

Программа на языке САНПО представляет собой последовательность инструкций, последней из которых является инструкция конца программы /END. Эта последовательность включает инструкции описания элементов данных, распределения памяти, конфигурации и типа оборудования, схемы параллельных процессов и т.д. /список инструкций приведен в приложении/. Такая совокупность инструкций является описанием прикладной системы автоматического накопления и предварительной обработки экспериментальной информации, на основании которого выполняется генерация собственно ПО эксперимента.

2.2. Структуры данных представлены переменными, буферами и событиями. Переменные - это поименованные объекты языка, принимающие числовые и текстовые значения. В зависимости от значений, которые они принимают, переменные имеют тип INTEGER или REAL одно- или двукратной точности.

Буфера являются поименованными сегментами памяти и аналогичны одномерным массивам в языке FORTRAN-IV. Они содержат числа и специфицируются по типу этих чисел. Буфера могут располагаться как в оперативной памяти, так и на внешних запоминающих устройствах /ВЗУ/.

События - это поименованные объекты языка, выступающие в САНПО в двух качествах. С одной стороны - это логическая переменная, принимающая значения TRUE и FALSE, с другой - событие является некоторым признаком /или сигналом/ состояния элементов данных /ЭД/ системы [2]. Событие как сигнал может находиться в активном состоянии /что соответствует значению TRUE / и в пассивном /значение FALSE /. Операцию приведения данного события в активное состояние мы называем операцией установки флага события или объявления события, а обратную операцию - операцией сброса флага события.

События имеют назначением отражение состояния ЭД в системе. Состояние буферов и переменных отражается событиями с теми же именами. Активное состояние таких событий означает, что информация из данного ЭД готова к обработке, а пассивное - ЭД пуст либо информация, содержащаяся в нем, еще не готова к обработке. Помимо этого события могут отражать любую желаемую ситуацию, сложившуюся в процессе работы прикладной системы, состояние совокупности ЭД и т.д. События используются для организации и синхронизации параллельно исполняемых процессов.

2.3. Инструкции в языке САНПО делятся на декларативные и интерпретируемые.

Декларативные инструкции обрабатываются компилятором САНПО во время генерации прикладной системы. На основании этих инструкций компилятор создает схему оперативной базы данных /ОБД/ и схему параллельно исполняемых процессов. Помимо этого из системных библиотек [5,6] извлекаются необходимые программные модули и настраиваются на указанные конфигурацию ЭВМ и тип оборудования. Результатом этой работы является прикладная система, скомпонованная из указанных в описании программных модулей. Функциональные возможности прикладной системы обеспечиваются составом модулей и соответствуют описанию.

Все декларативные инструкции начинаются символом / (код 57) и включают инструкции распределения памяти, описания ЭД, описания параллельно исполняемых процессов, макросредства, средства периода компиляции. При описании ЭД допускается задание начальных значений и абсолютное распределение памяти для буферов.

Список декларативных инструкций и их краткое описание приведены в приложении /часть А/. Инструкции описания параллельных процессов мы рассмотрим более подробно в следующем параграфе.

Интерпретируемые инструкции исполняются интерпретатором системы во время проведения эксперимента и включают операторы присваивания, цикла, условные операторы и некоторые другие /см. приложение, часть Б/.

Следует особо отметить функциональный оператор. Он реализуется стандартными программами /СП/ системы. Их набор не ограничен и может пополняться, для этого достаточно написанную и отлаженную СП в двоичном перемещаемом формате загрузки записать в системную библиотеку [5,6].

Списки инструкций в приложении не включают инструкций ввода/вывода. Это объясняется тем, что в САНПО работа с оборудованием организуется с помощью подсистем, которые состоят из резидента и набора СП, реализующих отдельные операции. Набор таких СП, их функции и способ исполнения операций оборудованием определяется при реализации подсистемы. Языковые средства могут быть ограничены специализированным набором функциональных операторов /см. приложение/, т.е. СП, но могут содержать и декларативные инструкции, составляющие подмножество языка САНПО и позволяющие ориентировать прикладную систему на конкретное оборудование.

2.3.1. Для пояснения инструкций описания параллельных процессов введем понятия элементарного процесса и процесса.

Элементарный процесс /ЭП/ - это операция системы /один или несколько операторов/, ассоциированная с элементом данных ОБД, и средства программной индикации состояния данных в виде ап-

парата событий. Назначение ЭП - выполнить действия, ассоциированные с соответствующим ЭД. ЭП может находиться в активном или пассивном состоянии в зависимости от состояния соответствующего ЭД. Инициируются только активные ЭП. В результате обработки одного ЭД могут быть выработаны сигналы о готовности к обработке других ЭД /объявлены соответствующие события и тем самым приведены в активное состояние новые ЭП/. Два ЭП называются функционально связанными, если в результате исполнения одного из них всегда или при определенных условиях другой ЭП приводится в активное состояние.

Процесс - это совокупность функционально связанных ЭП с определенными для них приоритетными отношениями, которые устанавливаются специальными инструкциями и регламентируют соответствующую дисциплину исполнения ЭП. Инициирование ЭП и обеспечение дисциплины исполнения процессов выполняется монитором системы САНПО.

2.3.2. Описать элементарные процессы можно декларативными инструкциями 11 и 12 /см. приложение, часть А/. Инструкцией под номером 11 описываются так называемые медленные ЭП, они исполняются в режиме интерпретации. ЭП, описанные другой инструкцией /номер 12/, обеспечиваются более быстрым механизмом обслуживания, но в них могут использоваться только функциональные операторы.

2.3.3. Приоритетные отношения ЭП возможны двух типов: абсолютные и относительные. Абсолютные приоритеты ЭП в конечном счете определяются порядковым номером данного ЭД в таблице ОБД. Эта таблица формируется в три этапа:

1. В таблицу заносятся последовательно ЭД в соответствии с порядком следования их описаний в программе.

2. Позиции ЭД в таблице упорядочиваются в соответствии с порядком следования имен ЭД в использованных в программе инструкциях /HIERARCHY /см. приложение, часть А, пункт 10/. Для каждой такой инструкции позиции ЭД устанавливаются в порядке убывания абсолютных приоритетов ЭД /возрастания номеров позиций/.

3. Абсолютные приоритеты ЭД корректируются в соответствии с порядком, заданным инструкцией /PRIORITY /см. приложение, часть А, пункт 9/.

Относительные приоритеты определяются иерархическими отношениями, которые носят характер подчинения. Эти отношения устанавливаются декларативной инструкцией под номером 10 (HIERARCHY). Стрелка в инструкциях означает, что событие, имя которого стоит слева, обладает меньшим приоритетом по отношению к событию, имя которого стоит справа от стрелки.

Заметим, что при исполнении совокупности ЭП принимаются во

внимание и влияют на порядок их исполнения приоритетные отношения только активных ЭП. Приоритетные отношения пассивных ЭП игнорируются до тех пор, пока эти ЭП не станут активными.

2.3.4. Инструкции описания параллельных процессов позволяют описать совокупность связанных параллельно исполняемых процессов средствами высокого уровня, при этом их синхронизация обеспечивается монитором САНПО на основе описанных на языке приоритетных отношений отдельных ЭП. Синхронизация обеспечивается без использования семафоров, а единицей программирования параллельных процессов является ЭП. Для синхронизации процессов могут быть использованы инструкции 6,7,8 и 9 /приложение, часть Б/, обеспечивающие управление состоянием ЭП. При необходимости допускается создание дополнительных средств синхронизации в виде функциональных операторов языка.

3. ЗАКЛЮЧЕНИЕ

Данный язык генерации прикладных систем для проведения физических экспериментов позволяет описывать автоматически исполняемые алгоритмы накопления и предварительной обработки экспериментальной информации, поступающей в реальном масштабе времени, а также является средством диалогового взаимодействия экспериментатора и прикладной системы во время проведения эксперимента. Набор операций открыт и может пополняться путем создания прикладных стандартных программ на языках MACRO-II и FORTRAN-IV. Содержание и объем этих операций определяется потребностями эксперимента, а их названия могут отражать назначение и область приложения.

Язык позволяет описывать связи программных модулей между собой и с наборами данных. Его конструкция не предполагает ориентации на какую-либо конкретную проблемную область, более того, введена формальная возможность изменить проблемную ориентацию языка без модификации компилятора. Тем не менее реализация компилятора и других технологических средств выполнена в первую очередь для программирования спектрометрических экспериментов на реакторе ИБР-2 [3], и временные характеристики генерируемых прикладных систем определены именно этим кругом задач.

Использование языка САНПО обеспечивает преемственность развития прикладных систем конкретных экспериментов, простоту изменения и модификации их состава и структуры, позволяет облегчить процесс сопровождения ПО различных экспериментов.

Язык САНПО достаточно прост и может быть использован без специальной подготовки в области системного программирования.

Отличительными особенностями данного языка являются следующие:

1/ используется метод программирования связанных параллель-

но исполняемых процессов средствами высокого уровня без использования семафоров;

2/ предложенные средства программирования параллельных процессов являются средствами описания алгоритмических структур.

Авторы полагают, что дальнейшее развитие средств и методов программирования алгоритмических структур позволит существенно сократить затраты на программирование, отладку прикладных систем реального времени и, что не менее важно, на доказательство соответствия способа функционирования полученных систем исходному заданию.

Компилятор языка САНПО реализован для ЭВМ типа СМ-3.

В заключение мы пользуемся случаем, чтобы поблагодарить В.П.Ширикова, Ю.М.Останевича, Г.П.Жукова, Л.Б.Пикельнера и коллег за полезные обсуждения и помощь в работе и Т.Б.Журавлеву за оформление рукописи.

ПРИЛОЖЕНИЕ

ИНСТРУКЦИИ ЯЗЫКА САНПО

Инструкции представлены в виде моделей, где выражения, заключенные в квадратные скобки, можно опускать, а фигурные скобки означают, что на данном месте должно стоять одно из заключенных в них выражений. Угловые скобки заключают в себе обозначение одного понятия фразой из нескольких слов, которое объясняется отдельно для каждой инструкции.

Имена, состоящие из малых букв латинского алфавита, обозначают объекты, смысл которых поясняется отдельно для каждой инструкции.

Символы $\Delta\Delta\Delta$ обозначают повторение предшествующей им конструкции, начало которой отмечено символом Δ, а символом $\sqcup$ обозначен пробел. Буквы n с индексами обозначают целые числа /десятичные или восьмеричные/. Символы m с индексами являются целыми числами и обозначают величину объема памяти. В общем случае m можно представить в виде n[K][{ W / B }], где K обозначает множитель 1024, а W и B - единицы, в которых указан объем памяти / W - слово, B - байт/. Сочетание букв с индексами обозначает арифметическое выражение, принимающее целые значения и задающее объем памяти. Такое выражение составляется из целых констант и конструкций вида (name), где name - это имя уже описанного буфера, соединенных знаками сложения (+), вычитания (-), умножения (*) и деления (/). Конструкция вида (name) принята для обозначения длины буфера, имя которого указано в скобках. Арифметическое выражение вида em по форме совпадает с арифметическим выражением в языке FORTRAN-IV /7/

23

с ограничением на глубину вложенности скобок /не более двух/. Не упомянутые выше элементы моделей, например, запятые, круглые скобки, имена, состоящие из больших букв латинского алфавита и т.д., используются как собственные элементы языка, т.е. они должны присутствовать в инструкциях в таком же виде, как и в моделях.

А. ДЕКЛАРАТИВНЫЕ ИНСТРУКЦИИ

1. Объем выделенной оперативной памяти /ОП/ и версия прикладной системы /ПС/ задаются инструкцией

$$/\text{CORE}\sqcup m_{\text{оп}}\left[\left\{\begin{Bmatrix}/\text{LDA}\\ /\text{RTSJ}\\ /\text{RTFB}\end{Bmatrix}\left[\begin{Bmatrix}/\text{LOCK}\\ /\text{UNLOCK}\end{Bmatrix}\right]\right\}\right].$$

Здесь $m_{\text{оп}}$ - значение объема ОП; LDA - перфоленточная версия ПС; RTSJ и RTFB - варианты операционной системы /ОС/ RT-11 /4/; LOCK и UNLOCK регулируют состав резидента ОС RT-11 /4/. По умолчанию принимаются значения параметров /RTSJ и /UNLOCK.

Пример: /CORE 18KW.

2. Области памяти на внешних запоминающих устройствах /ВЗУ/ называются полями и задаются инструкцией

$$/\text{FIELD}^{*}\, m_{\ell}\sqcup n_{\text{п}}[\sqcup \text{dev:}]\left(\left\{\begin{bmatrix} n_{\text{н}}\\ \text{NEW}^{*}\\ \text{OLD}^{*}\end{bmatrix}\left\{\begin{matrix}:\, n_{\text{к}}\\ ;\, em_{\text{п}}\end{matrix}\right\}\ \text{filename.ext}[=em_{\text{п}}]\right\}\right),$$

m_{ℓ} - объем минимальной единицы обмена между ОП и ВЗУ, на котором располагается описываемое поле /длина физической записи/; $n_{\text{п}}$ - номер поля /от 1 до 63_{10} /; dev - имя устройства /имена устройств задаются так же, как в ОС RT-11 /3/ /; $n_{\text{н}}$ и $n_{\text{к}}$ - адреса соответственно начала и конца поля; $em_{\text{п}}$ - длина поля; filename и ext составляют спецификацию файла /имя и расширение имени/ в принятом в RT-11 формате; характеристики OLD и NEW сообщают, существует или должен быть создан указанный файл. Если использована характеристика NEW, то после знака равенства задается длина поля, а если OLD, то считается, что указанный файл уже существует /длина после знака равенства игнорируется/. По умолчанию принимается характеристика NEW.

Пример: /FIELD* 256W 1 SY:(EXP.DAT=256W*10000).

3. Описание переменных и присвоение им начальных значений выполняется инструкцией

/[D[OUBLE␣][␣]] [{ I[NTEGER␣][␣] / R[EAL␣][␣] }] V[ARIABLE]␣Δ

{ name / n* name_Н % } [= < числовая константа >] [,ΔΔΔ] ,

name - имя переменной; конструкция n* name_Н % означает, что описывается n имен, состоящих из последовательности символов, в начале которой стоит name_Н, а в конце - символьное представление порядкового номера /от 1 до n /. Например, описание 3* A% равносильно описанию A1, A2, A3; в случае наличия знака равенства переменной присваивается начальное значение, равное указанной константе.

Пример: /DOUB REAL VAR K23, 4* T% = 3.141592650.

4. Описание буферов и распределение памяти для них выполняется инструкцией

/[D[OUBLE␣][␣]] [{ I[NTEGER␣][␣] / R[EAL␣][␣] }] B[UFFER]␣Δ { name([n_П :] { n_Н ␣ n_К / n_Н . em_Б / em_Б }) / n* name_Н % ([n_П :] em_Б) } [, ΔΔΔ],

name - имя буфера; конструкция n* name_Н % описана выше /приложение, часть А, п.3/; n_П - номер поля, на котором располагается описываемый буфер; n_Н и n_К - адреса соответственно начала и конца буфера; em_Б - длина буфера.

Пример: /INT BUF A(13:10KB), B(100W).

5. Описание событий выполняется инструкцией

/EV[ENT]␣Δ { name / n* name_Н % } [, ΔΔΔ],

name - имя события; конструкция n* name_Н % описана в п.3 части А приложения.

Пример: /EVENT FLAG1, 5* FLG%.

6. Начальные значения буферов задаются инструкцией

/DATA ␣ name = Δ [n*] < константа > [, ΔΔΔ],

name - имя буфера; n - число повторений следующей за звездочкой константы /по умолчанию принимается равным 1/. Константы могут быть текстовыми и числовыми. Перечисленные в инструкции константы размещаются в буфере последовательно непосредственно друг за другом.

Пример: /DATA B=1,2,3,4,50*7,46*177777V.

7. Имена периода компиляции определяются инструкцией

/LET ␣ Δ name_ПК = { name / < константа > } [, ΔΔΔ],

name_ПК - имя, определяемое как имя периода компиляции;
name - ранее определенное имя. Имена периода компиляции заменяются во время компиляции программы на их значения /имена или константы/.

8. Загрузка стандартных программ на рабочее поле прикладной системы на этапе генерации выполняется инструкцией

/GET ␣ Δ name [, ΔΔΔ],

name - имя стандартной программы.

9. Абсолютные приоритеты можно установить инструкцией

/PRIORITY ␣ [n:] Δ < имя > [, ΔΔΔ],

где < имя > - имя, ранее определенного ЭД; n - номер позиции в таблице ЭД первого имени из списка /чем больше номер, тем меньше абсолютный приоритет/. Элементам данных, имена которых указаны в списке, присваиваются последовательные номера позиций, т.е. их абсолютные приоритеты располагаются в порядке убывания.

10. Иерархические связи элементов данных в оперативной базе данных устанавливаются инструкцией

/HI[ERARCHY]␣ Δ name_1 → name_2 [{ , / ; } ΔΔΔ],

name_1 и name_2 - имена событий или буферов; name_2 будет иметь более высокий относительный приоритет, чем name_1.

11. Описание медленного /интерпретируемого/ элементарного процесса имеет вид

/CASE ␣ name ␣ DO
.
.
.
/END[␣]C[ASE],

name - имя события или буфера; многоточие обозначает последовательность интерпретируемых инструкций.

12. Описание быстрого ЭП может быть выполнено другой инструкцией:

/CASE ␣name ␣DO ␣$_{\Delta}$f[, $_{\Delta\Delta\Delta}$],

name - имя буфера или события; f - функциональный оператор.

Пример: /CASE FLAG DO SORT1(A,B,C), SORT2(A,B,D), WR(A, BUF) .

13. Макроопределение имеет вид

/MACRO [␣]. name [(⟨список формальных параметров⟩)]

.

.

.

/END[␣]M[ACRO],

name - имя макрооператора; список формальных параметров - это последовательность имен, отделенных друг от друга запятыми; многоточие означает последовательность инструкций /могут быть и декларативные инструкции/; /END MACRO - конец макроопределения.

14. Макрооператор имеет вид

/.name [(⟨список фактических параметров⟩)] ,

name - имя макрооператора; список фактических параметров - это последовательность имен элементов данных и констант, отделенных друг от друга запятыми.

15. Конец программы /описания прикладной системы/ определяется инструкцией

/END.

Б. ИНТЕРПРЕТИРУЕМЫЕ ИНСТРУКЦИИ

1. Оператор присваивания имеет вид

name = ⟨арифметическое выражение⟩ ,

name - имя переменной; арифметическое выражение аналогично принятому в языке FORTRAN-IV [7].

Пример: K =(K*2-1) *N.

2. Функциональный оператор имеет вид

name [(⟨список фактических параметров⟩)] ,

name - имя стандартной программы; список фактических параметров - это последовательность имен элементов данных и констант, отделенных друг от друга запятыми /не более шестнадцати параметров/. Набор таких операторов открыт, и для введения нового функционального оператора достаточно написать на языке MACRO-II [8] или FORTRAN-IV [7] стандартную программу, отладить ее и поместить в библиотеку системы [5,6].

Пример: F1(A,B,C,D,E,F) .

3. Оператор перехода имеет вид

GOTO ␣ { name / [{±}n] },

name - метка; n - дистанция /число операторов/ от данного оператора до того, которому нужно передать управление. Знак плюс перед n обозначает, что отсчитывать нужно вниз, к концу программы, а минус - к ее началу. По умолчанию принимается знак плюс.

4. Условный оператор /аналогичен соответствующему оператору в языке FORTRAN-IV [7]/ имеет вид

IF(⟨арифметическое выражение⟩) ␣ [name$_1$][,[name$_2$],[name$_3$]],

name$_1$, name$_2$ и name$_3$ - либо метки, либо относительные номера инструкций /см. описание оператора GOTO /.

5. Оператор цикла представляется в виде

name$_1$: DO␣name$_2$ ␣ i=n$_1$,n$_2$[,n$_3$]

.

.

.

name$_2$: CONTINUE␣name$_1$,

name$_1$ и name$_2$ - метки; i - параметр цикла /переменная типа INTEGER /; n$_1$ - начальное значение, n$_2$ - конечное и n$_3$ - шаг изменения параметра цикла /по умолчанию n$_3$ = 1 /; многоточие обозначает последовательность интерпретируемых инструкций.

6. Оператор ожидания объявления события или их комбинации имеет вид

WAIT[TRUE] ␣$_{\Delta}$ name$_1$ [{.AND. / .OR.} name$_2$ $_{\Delta\Delta\Delta}$],

name$_1$ и name$_2$ - имена событий или буферов; .AND. означает операцию дизъюнкции, а .OR. - операцию конъюнкции.

Результатом исполнения этой инструкции будет задержка интерпретации последовательности инструкций до момента, когда логическое выражение примет значение .TRUE. .

Пример: WAIT READY .AND. FLAG1.

7. Оператор ожидания сброса флага события имеет вид

$$\text{WAIT F[ALSE]}\_\text{name}_1\Delta\ [\{.\text{AND.}\ .\text{OR.}\}\ \text{name}_2\ \Delta\Delta\Delta],$$

name_1 и name_2 - имена событий или буферов; .AND. и .OR.обозначают соответственно операции дизъюнкции и конъюнкции.

Данная инструкция выполняется аналогично предыдущей /номер 6, часть Б/, но переход к интерпретации следующей инструкции происходит, если логическое выражение приняло значение .FALSE. .

8. Оператор объявления событий имеет вид

$$\text{UP}\_\Delta\text{name}[,\ \Delta\Delta\Delta],$$

name - имя буфера или события.

Пример: UP A,B,FLG.

9. Оператор сброса флагов событий имеет вид

$$\text{DOWN}\_\Delta\text{name}[,\ \Delta\Delta\Delta],$$

name - имя буфера или события.

ЛИТЕРАТУРА

1. Нестерихин Ю.Е. и др. Организация систем автоматизации научных исследований /проблемы, методы, перспективы/. Автометрия , 1974, №4.
2. Островной А.И., Саламатин И.М. ОИЯИ, Р10-11349, Дубна, 1978.
3. Ананьев В.Д. и др. ОИЯИ, Р3-10888, Дубна, 1977.
4. RT-11 System Reference Manual (Dec-11- ORUGA-C-D), Dec. Maynard, Massachusetts, 1975, p.755.
5. Балука Г., Саламатин И.М., Хрыкин А.С. ОИЯИ, 10-12545, Дубна, 1979.
6. Балука Г., Саламатин И.М., Хрыкин А.С. ОИЯИ, 10-12546, Дубна, 1979.
7. PDP-11 Fortran Language Reference Manual (Dec-11-IFLRA-B-D), Dec. Maynard, Massachusetts, 1974, p.173.

Рукопись поступила в издательский отдел
18 июня 1980 года.