УДК 681.3.06 : 681.322

В. И. ЖУК, Б. И. ШИТИКОВ

(*Москва*)

МОДУЛЬНОЕ ПРОГРАММИРОВАНИЕ НА РАЗГОВОРНОМ ЯЗЫКЕ ОПИСАНИЯ БЛОК-СХЕМ

Введение. Принципы модульного построения программ все шире применяются в системах автоматизации физических экспериментов (САФЭ) [1—3]. Это обусловлено тем, что при проведении больших дорогостоящих физических экспериментов весьма важно обеспечить минимально возможные сроки создания и изменения программных средств САФЭ и в максимально возможной степени освободить экспериментатора от рутинной работы. К одной из основных проблем модульного программирования [4—8], существенных для САФЭ, относится автоматизация компоновки программ из модулей [7], или, иначе, модульных программ.

В дальнейшем под модульным программированием будем понимать один из его видов — компоновку модульных программ на основе библиотеки программных модулей. Модульное программирование осуществляется либо на базовых языках, используемых для создания программных модулей (например, язык ассемблера, ФОРТРАН, АЛГОЛ и т. д.)., либо на специальных макроязыках, лексической единицей которых является имя модуля. Однако для применения этих языков требуется предварительное изучение их. В ряде систем программного обеспечения модульное программирование на специальных макроязыках сводится к заданию только линейной последовательности выполнения модулей [1, 2]. Во многих случаях требуется, чтобы макроязык модульного программирования обеспечивал реализацию разветвлений и циклов в программе. При этом представляется важным, чтобы для пользования этим макроязыком было достаточно минимального предварительного знакомства с ним. В качестве такого языка в данной работе предлагается разговорный язык описания модульных блок-схем программ, сходный с языком ввода в ЭВМ информации о блок-схемах для их автоматического вычерчивания [9]. Функции рассматриваемого в данной работе разговорного языка описания блок-схем программы сводятся к заданию топологической схемы программ (направленного графа изоморфного с требуемым модульным графом программы [7]) и имен модулей.

1. **Модули.** Для организации разветвлений в модульной программе необходимо использовать программные модули, имеющие более одного выхода. Без потери общности рассматриваемого здесь способа программирования ограничимся применением только одновходовых программных модулей.

Примем, что используемые программные модули создаются в рамках некоторой базовой операционной системы. Для примера будем считать, что применяется базовая операционная система, обеспечивающая вызов в оперативную память (ОП) несегментированных программ и сегментов из внешней памяти прямого доступа, где они находятся постоянно. При выполнении сегментированной программы главная программа, вызванная в ОП, остается там на время выполнения всех сегментов и поочередно вызывает сегменты в отведенное для них поле ОП. Кроме того, программа любого из указанных типов (главная или сегмент) может вызвать программу того же или другого типа. В каждую программу любого из этих типов включены все относящиеся к ним под-

программы. Других ограничений на базовую операционную систему не накладывается. К таким операционным системам относятся, например, системы DOS-M «Хьюлетт — Паккард» и DOS/360. Примем, что используются следующие виды программных модулей: несегментированные программы, части сегментированных программ (главные программы и сегменты), закрытые и открытые подпрограммы. Модули-подпрограммы могут иметь более одного выхода. Программные модули остальных видов имеют только один выход. Все программные модули размещены в библиотеке программных модулей, которая может содержать исходные и объектные модули. Кроме программных модулей, имеются еще модули данных, размещенные в библиотеке модулей данных. К ним относятся модули-таблицы, модули исходных данных и результатов обработки. Модули-таблицы содержат параметры, используемые программными модулями. Библиотеки программных модулей и модулей данных размещены во внешней памяти.

2. **Процесс модульного программирования.** Перед началом разработки блок-схемы модульной программы пользователь должен иметь необходимые сведения о применяемом наборе модулей. Эти сведения он может получить из соответствующей документации. Кроме того, в более краткой форме основные сведения о модулях могут быть представлены ему информационно-поисковой системой, в которой собраны описания всех модулей и инструкции по их применению. Процесс модульного программирования удобно разбить на несколько этапов.

1-й этап. Разработка блок-схемы модульной программы. Получив необходимые сведения о наборе модулей, разрабатываем блок-схему модульной программы, состоящую из элементов, изображающих модули, и соединяющих их линий. Для каждого элемента на блок-схеме указываем имя модуля.

2-й этап. Подготовка блок-схемы программы для ввода ее в ЭВМ. Присваиваем всем элементам блок-схемы произвольные индивидуальные номера. Иными словами, нумеруем вершины графа программы так, чтобы одинаковые номера не встречались. Если бланки для изображения модульных блок-схем разбиты на зоны [10], то вместо номеров элементов можно использовать координаты зон, в которых размещены элементы. Далее нумеруем выходы каждого элемента в пределах его изображения в соответствии с описанием модуля, замещающего этот элемент. Для упрощения нумерации можно ввести ограничение, состоящее в том, что каждый элемент имеет не более двух выходов.

3-й этап. Ввод топологической схемы модульной программы. Вызываем программу ввода блок-схемы, и в режиме диалога вводим в ЭВМ, например с телетайпа, таблицу ТТ, содержащую информацию о топологической схеме программы. В этой таблице указываются номера (координаты) элементов схемы, их выходов и элементов, соединенных с этими выходами. В начале диалога программа ввода может «напомнить» пользователю форму сообщений (язык ввода), с помощью которых пользователь должен ввести в машину информацию о топологической схеме модульной программы, например:

$$N.X{-}A,\ Y{-}B,\ \ldots,\ Z{-}C;$$

N — номер элемента;

$X,\ Y,\ Z$ — номера выходов элемента N;

$A,\ B,\ C$ — номера элементов, соединенных с выходами $X,\ Y,\ Z$ элемента N.

Ответы пользователя в соответствии с приведенным «напоминанием» выглядят на бланке телетайпа следующим образом:

1. 1—3, 2—2, ..., 4—10;
2. 1—4.

4-й этап. Ввод имен программных модулей. В режиме диалога вводим в ЭВМ таблицу имен (ТИ), в которой для каждого номера элемента указывается имя программного модуля и его вид. Каждому имени модуля соответствует его вид, шифр графического символа, изображающего модуль, количество выходов модуля и их нумерация. При необходимости шифр графического символа может быть введен вместе с именем модуля. Вид модуля вводить в ЭВМ не обязательно, но совместный ввод имени и типа модуля позволит проконтролировать их соответствие. В принципе 4-й этап может быть совмещен с 3-м. Разделение этих этапов может быть удобным, так как позволяет обойти 3-й этап, например, в том случае, когда пользователь, не изменяя топологической схемы, захочет изменить только имена и виды модулей в программе.

5-й этап. Ввод имен модулей данных. В режиме диалога вводим в ЭВМ имена модулей данных (таблицы параметров, исходные данные и результаты). При этом программа ввода может сообщать имена модулей данных, «напоминая» при необходимости назначение этих модулей с тем, чтобы пользователь мог произвести выбор нужных имен.

6-й этап. Ввод параметров. В режиме диалога вводим таблицы параметров программных модулей и параметры, не вошедшие в таблицы. При этом также могут быть использованы «подсказки» со стороны ЭВМ.

7-й этап. Ввод данных для отладки. При необходимости для целей отладки пользователь может задать точки (например, номер элемента и его выхода) начала и останова модульной программы.

Программа контролирует введенную информацию и указывает пользователю на допущенные ошибки. При исправлении этих ошибок можно предусмотреть возврат к предыдущим этапам. Программа ввода может проконтролировать наличие имен модулей в каталоге, соответствие имен программных модулей их видам, допустимость соединения модулей, соответствие модулей данных программным модулям, допустимость значений параметров и размещение модулей в памяти. По окончании ввода данных ЭВМ может выдать документы в виде блок-схемы программы и необходимых таблиц.

3. **Трансляция и редактирование.** По окончании ввода информации о модульной программе таблицы ТТ и ТИ преобразуются входным транслятором в объектные таблицы, набор и структура которых зависят от способов последующей их обработки. В программе ввода может быть указан вид последующей обработки компиляции или интерпретации объектных таблиц. В первом случае формируется текст модульной программы, соответствующей разработанной блок-схеме. Во втором случае к интерпретатору присоединяются модули-подпрограммы. Анализ показывает, что при совместном использовании модулей-программ, сегментов и подпрограмм целесообразно оставить за последними только функции управляющих операторов. При этом набор модулей-подпрограмм может быть существенно ограничен. Все вызываемые интерпретатором модули представлены на машинном языке.

Так как модули-программы и модули-сегменты для их выполнения каждый раз вызываются из внешней памяти с прямым доступом, то время интерпретации не должно быть заметно больше времени выполнения скомпилированной модульной программы. В связи с этим режим интерпретации модульной программы представляется более предпочтительным, так как при несущественной потере быстродейст-

вия допускает более простую реализацию, чем режим компиляции. Отметим, что модульная программа может сохраниться в банке модульных программ в виде объектных таблиц, сформированных входным транслятором, и ее редактирование можно свести к редактированию указанных объектных таблиц.

В качестве иллюстрации рассмотрим один из возможных алгоритмов интерпретации.

Интерпретатор использует следующие таблицы:

таблица указателей связи (ТУС), в которой номер каждого следующего искомого элемента определяется в зависимости от номеров предыдущего элемента и его выхода, соединенного со входом искомого элемента;

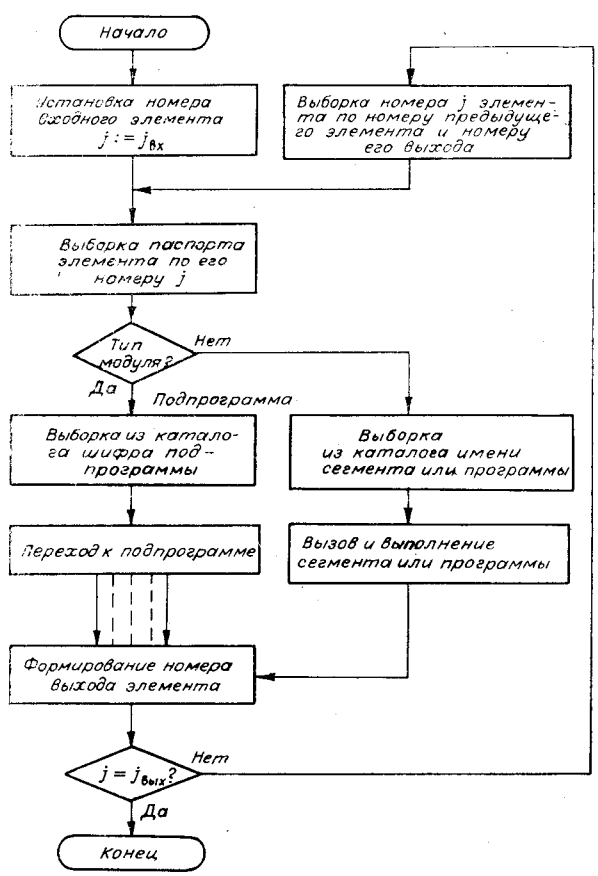
текущий каталог имен (ТКИ) модулей-программ и модулей-сегментов, используемых в интерпретируемой программе;

текущий каталог модулей-подпрограмм (ТКМ), применяемых в интерпретируемой программе;

таблица паспортов элементов (ТПЭ) блок-схемы интерпретируемой программы, где для каждого номера элемента указан номер текущего каталога, соответствующего виду программного модуля (таблицы ТКИ, ТКМ), и номер программного модуля в текущем каталоге.

В таблице ТУС элементы, имеющие один выход, являются элементами цепного списка, а элементы, имеющие более одного выхода, представляют элементы узлового списка.

Номера подпрограмм и каталогов могут служить их относительными адресами в оперативной памяти. Номера подпрограмм определяются входным транслятором по именам подпрограмм, указанным в исходной таблице ТИ. Таблица ТКМ может отсутствовать. В этом случае в таблице ТПЭ указывается непосредственно относительный адрес подпрограммы вместо его номера.



Входной транслятор составляет таблицу ТУС по таблице ТТ, а таблицы ТКИ, ТКМ, ТПЭ — по таблице ТИ. Кроме этих четырех интерпретируемых таблиц, интерпретатором могут быть использованы и другие, например, таблицы параметров подпрограмм.

При составлении таблицы ТУС транслятор может определить входной элемент блок-схемы модульной программы как элемент, не указанный в таблице ТТ в качестве указателя связи, а выходной элемент — как элемент, не имеющий указателя связи. Такое определение входных и выходных элементов соответствует блок-схеме интерпретатора, показанной на

рисунке. Другой вариант определения входных и выходных элементов заключается в том, что пользователь присваивает этим элементам определенные номера, например «1» — для входного, «0» — для выходного элементов.

Объем интерпретируемых таблиц может быть сокращен, если для представления линейных частей модульной программы использовать гнездовые списки [11].

В заключение отметим, что для пользования описанным языком модульного программирования достаточно минимального предварительного знакомства с ним. Данный подход предполагается применить для оперативного создания модульных программ при автоматизации сбора и обработки данных в физических экспериментах.

ЛИТЕРАТУРА

1. R. M. Brown, M. A. Fisherkeller, A. E. Gromme, J. V. Levy. The SLAG high-energy spectrometer data acquisition and analysis system.— "Proc. of the IEEE", 1966, vol. 54, № 12.
2. Н. Н. Воробьева, Л. С. Нефедьева. Язык общений в системах приема и обработки физической информации.— Препринт № 10-4595. Дубна, Изд. ОИЯИ, 1969.
3. Ю. Я. Кузьмин. Об одной альтернативе программирования эксперимента.— В кн.: Кибернетизация научного эксперимента. (Учен. зап. Латв. Гос. ун-та им. П. Стучки). Рига, 1973, т. 196, вып. 5.
4. I. Klingels. Some thoughts on the future of modular programming.— "Data Proc.", London, 1971, vol. 13. № 4.
5. Е. А. Жоголев. Система модульного программирования (СИМП).— В кн.: Вычислительные методы и программирование. Вып. XVII. М., Изд. МГУ, 1971.
6. Е. А. Жоголев. Принципы построения многоязычной системы модульного программирования.— «Кибернетика», 1974, № 4.
7. И. Н. Парасюк, И. В. Сергиенко. Некоторые вопросы разработки и исследования одного класса универсально-специализированных автоматизированных систем обработки данных.— «Кибернетика», Киев, 1973, № 6.
8. И. Н. Парасюк, И. В. Сергиенко, Н. И. Тукалевская. Универсально-специализированная автоматизированная система обработки данных на ЦВМ (система УСОД).— «Упр. сист. и маш.», Киев, 1974, № 2.
9. W. G. Repsher. BELLFLOW draws flow diagrams automatically.— "Bell. Laboratories Record", New York, 1971, vol. 49, № 7.
10. Обработка данных и программирование. Схемы алгоритмов и программ. Правила выполнения. ГОСТ 19427-74.
11. А. И. Китов. Программирование информационно-логических задач. М., «Сов. радио», 1967.

Поступила в редакцию 18 февраля 1975 г.

УДК 681.3.06

С. В. БРЕДИХИН, П. М. ПЕСЛЯК

(*Новосибирск*)

ПРОСТАЯ СИСТЕМА ПРОГРАММИРОВАНИЯ ДЛЯ САМАС

ВВЕДЕНИЕ

Система программирования SICS (аббревиатура от simple interactive CAMAC system) предназначена для описания и исполнения простейших действий с аппаратурой САМАС. Основные пользователи