АКАДЕМИЯ НАУК СССР
СИБИРСКОЕ ОТДЕЛЕНИЕ
АВТОМЕТРИЯ

№ 1 1976

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ СИСТЕМ АВТОМАТИЗАЦИИ НАУЧНЫХ ИССЛЕДОВАНИЙ

УДК 681.3.06

Э. А. ТАЛНЫКИН

(*Новосибирск*)

МОДУЛЬНОЕ ПРОГРАММИРОВАНИЕ В ЗАДАЧАХ СБОРА И ОБРАБОТКИ ЭКСПЕРИМЕНТАЛЬНЫХ ДАННЫХ

Введение. В статье обсуждается специфика задач, возникающих при построении программного обеспечения систем сбора и первичной обработки данных. Предлагается возможный подход к организации такой системы, а также принципиальная структура составляющих ее компонентов и средств управления ими в режиме "on line". Отдельные аспекты предполагаемого подхода использовались в течение ряда лет при разработке программного обеспечения машинных комплексов на базе мини-ЭВМ [1—5].

Модульность в программировании является фундаментальным понятием и берет истоки от создания первой библиотеки стандартных подпрограмм. Под модулем будем понимать программный элемент, организованный и оформленный специальным образом, так что им можно воспользоваться, зная лишь реализуемую им функцию (которую можно считать неделимой операцией), а также способ представления входных и выходных данных. Модули обычно организуются в пакеты или библиотеки. Примером такого пакета может служить любая специализированная научная библиотека или библиотека стандартных подпрограмм в некоторой системе программирования. Средства управления здесь определяются способами вызовов подпрограмм и передачи параметров.

В задачах управления экспериментом приходится иметь дело с процессами обмена, которые с точки зрения программирования представляются наиболее сложными, если даже они частично выполнены на уровне операционной системы. В этом отношении набор модулей, реализующих отдельные функции, значительно упростит ситуацию. Стандартные же средства управления модулями здесь недостаточно эффективны. Дело в том, что основным пользователем программного обеспечения является экспериментатор, причем характер управляющих действий зачастую не полностью определен заранее, а выясняется в процессе работы. Если на каждое пробное решение составлять отдельную программу, пусть даже совсем простую, состоящую из последовательности вызовов модулей, то такой способ управления не будет удовлетворять экспериментатора из-за полного отсутствия оперативности. Средства управления пакетом должны обеспечить возможность реализации пробных решений непосредственно, не прибегая к трансляции или пробивке перфокарт. Задание должно вводиться в систему подходящего терминала в терминах, легко доступных экспериментатору. В

этой связи необходим режим пошагового исполнения отдельных операций с возможностью объединения последовательности этих операций в программу, исполняемую автоматически.

Даже в случае, если известно, что комплекс будет работать длительное время по фиксированной программе, предлагаемый подход является эффективным на этапах отладки и тестирования, когда необходимо промоделировать работу сложной установки или проверить правильность работы вспомогательных компонентов. Кроме того, существенно упрощается модификация системы, так как системная часть (управляющая пакетом программа) не зависит от конкретного эксперимента.

Еще одно существенное требование к системе — это гибкость по отношению к составу оборудования и его конфигурации. Выпадение одного из устройств не должно разрушить целостности системы, а включаемое в нее новое устройство должно легко вписываться в общую схему. То же самое можно сказать и о функциях, реализуемых чисто программно (например, алгоритмы обработки).

Для достижения этих целей недостаточно просто модульной организации системы. Перекрестные ссылки между модулями должны некоторым образом упорядочиваться, чтобы обеспечить возможность простой переконфигурации в случае изменения состава оборудования или круга решаемых задач.

Функциональные модули. Назначение и структура модулей данного типа достаточно подробно описаны в [1]. Далее в большей степени будет затронута общая организация пакета, т. е. взаимосвязь модулей друг с другом. Функциональным будем называть модуль, который, получив один раз управление, к моменту возврата в вызвавшую программу выполняет все возложенные на него функции и не требует дополнительных обращений для завершения операций. Таким свойством обладают, например, стандартным образом организованные подпрограммы. В рамках систем сбора и обработки данных на эти модули предлагается возложить функции управления оборудованием, а также выполнение некоторых операций над данными.

Для каждого конкретного устройства или установки выделяется набор элементарных (по возможности независимых и неделимых далее) операций, которые могут быть реализованы программно и в полной мере обеспечивают возможность управления этим устройством. Для каждой из выделенных операций разрабатывается реализующий ее программный модуль. Это (при условии, что имеются средства управления модулями) доставляет адекватные средства для работы с внешним оборудованием.

Очевидно, что экспериментатору недостаточно возможности активизировать одну из заранее выделенных функций, так как ему всегда необходимо менять некоторые параметры эксперимента. Таким образом, управляющая программа должна уметь передавать отдельным модулям параметры, получаемые от экспериментатора. В этом отношении наиболее приемлемым представляется механизм, уже имеющийся в используемой системе программирования. В конечном счете это означает, что функциональный модуль оформляется в виде стандартной подпрограммы со стандартным механизмом передачи параметров. Наряду с очевидными преимуществами стандартизованного подхода, позволяющего не загромождать систему излишними условностями, можно получить и существенный качественный выигрыш, состоящий в том, что один и тот же модуль может использоваться одновременно как функциональный модуль системы и как библиотечная подпрограмма. Рассматривая модули в последнем качестве, можно реализовать все более и более сложные функции и на любом этапе развития системы в случае необходимости прибегнуть к помощи элементарного

модуля, который используется для построения более сложных функциональных элементов.

На базе такого набора элементарных модулей при дальнейшем развитии системы можно придерживаться широко применяемого в программировании подхода к созданию многоуровневых систем математического обеспечения. Нулевой уровень, имея дело с реальными объектами, создает систему более укрупненных понятий, на которых строится следующий уровень, и т. д.

Такой метод позволяет неограниченно наращивать функциональные возможности, но тем не менее получаемая подсистема модулей четко ограничивается рамками исходного базиса. Если разрабатываемая подсистема модулей ориентировалась на управление некоторым устройством, то при выпадении из рассмотрения данного устройства естественным образом отпадает потребность в этих модулях и причем только в них. Аналогично при необходимости обеспечить работу с новым устройством нужна реализация лишь чисто специфических функций.

На рис. 1 показана принципиальная структура такой системы, где отдельные подсистемы модулей предназначены для решения вполне определенного круга задач, связанных с управлением оборудованием или данными. При разработке подсистемы существенной является лишь внутренняя ее структура и взаимосвязь с управляющей программой. О существовании других подсистем, а тем более логики их функционирования можно не заботиться. Развитие системы может происходить независимо по двум направлениям, одно из которых связано с разработкой новых подсистем, а другое — с расширением внутренних возможностей уже существующих.

Управляющая программа. Во время генерации системы каждому модулю приписывается имя, с помощью которого он может активизироваться. Это имя никак не связано с именем модуля, определяемым во время трансляции, или с именем точки входа, хотя и может с ним совпадать. Активизация модулей оператором осуществляется с помощью директив. Директива представляет собой список параметров, разделенных запятыми, где первый параметр является именем, приписанным соответствующему модулю, и будет называться далее именем директивы. Фактически же управляющая программа не распознает никаких директив и инвариантна по отношению к составу управляемых ею модулей. Все необходимое для работы управляющей программы содержится в таблице информации о модулях, которая формируется во время генерации. Таблица информации о модулях состоит из записей, ключевым полем которых является имя директивы. Кроме того, в таблице содержится информация о типе модуля, его месторасположении, числе и типе параметров, значениях параметров по умолчанию (когда они опущены в списке вызова).

Работа управляющей программы начинается с ввода директивы. После определения имени директивы находится соответствующая запись в таблице и формируется список параметров вызова. Затем, если сегмент, содержащий модуль, не содержится в оперативной памяти, управляющая программа обеспечивает его подгрузку и передает управление модулю. В случае, если параметры в директиве не соответствуют типам, описанным в таблице, или запись с таким именем в

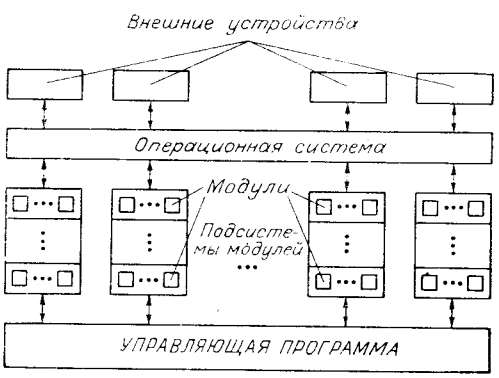

Рис. 1. Структура системы.

таблице не обнаружена, директива отвергается. Получив управление, модуль принимает параметры и начинает выполнять свои функции. По окончании работы модуль производит возврат в управляющую программу, которая вводит следующую директиву. Таким образом получается пошаговый режим работы, о котором говорилось во введении.

Для автоматизации выполнения некоторых типовых операций реализующую их последовательность директив можно хранить в библиотеке в виде стандартного символьного файла. По специальной директиве, в которой указывается имя этого файла, управляющая программа переключается с ввода директив от оператора на чтение их из библиотеки. В некотором смысле символьный файл с директивами можно считать программой, а каждую составляющую ее директиву — макрокомандой. В такой программе может содержаться директива запуска другой программы.

Возможность формирования из директив программы дает еще один эффективный способ структурирования модулей. Здесь, в отличие от структурирования на программном уровне, всякий раз будет производиться интерпретация директив, что обусловливает дополнительные затраты времени и может оказаться критичным в некоторых ситуациях. Преимущества состоят в том, что все это можно сделать во время работы системы. После отработки выделенной последовательности операций для нее может быть разработан отдельный модуль.

Организованная таким образом система представляется удобной и для разработчика. После реализации ядра (управляющей программы) все необходимые функции можно выполнить в виде стандартных модулей, которые будут, безусловно, включаться в систему во время генерации и активизироваться с помощью предопределенных директив. Системные модули отличаются от обычных тем, что имеют доступ к системным базам данных и более широко используют возможности управляющей программы. Примером может служить модуль запуска программы из библиотеки.

Модули с динамической настройкой. Системы сбора и первичной обработки обычно, кроме стандартных машинных носителей, обладают многими другими источниками данных. В этом отношении для экспериментатора идеальна ситуация, когда он может указать, откуда необходимо получить информацию, какие алгоритмы обработки к ней применить и куда записать результат, не меняя предоставленных ему в распоряжение программ. Это означает, что каждая программа уже в готовом для исполнения виде должна иметь возможность «настраиваться», например, на чтение с одного или другого носителя. Стандартные средства программирования таких возможностей не представляют. Обычный подход состоит в выделении в подпрограммы операций чтения и записи для всевозможных носителей или источников информации, а также алгоритмов обработки. Далее составляется очень простая программа, которая читает данные с помощью одной из выделенных подпрограмм, обращается к нескольким подпрограммам, реализующим алгоритмы обработки, и записывает результат также через подпрограмму. Если вся подлежащая обработке информация не помещается в оперативной памяти, такой процесс повторяется для каждой очередной порции (записи или зоны).

Первый недостаток такого способа заключается в том, что составление даже простой программы с необходимостью пройти весь процесс ее подготовки от пробивки перфокарт до загрузки и исполнения значительно снижает общий темп работы. Экспериментатор должен иметь возможность применять одни и те же алгоритмы к данным, поступающим в реальном времени и с автономных носителей. Например, для случая быстрых процессов таким образом можно подобрать один

из алгоритмов фильтрации, который успевает работать в темпе эксперимента и позволит сократить объем хранимой информации.

Второй недостаток касается уже внутренней структуры подпрограммы. Он заключается в том, что связующая программа в описанной ситуации выступает в роли головной, а все подпрограммы являются подчиненными. Головная программа имеет значительные преимущества, связанные с тем, что она не должна помнить о ситуации, сложившейся в процессе работы. Эта программа в каждый момент обращается к подчиненной подпрограмме и после возврата управления продолжает работать с той же точки (в том же контексте). Подчиненная же подпрограмма всякий раз получает управление в точку входа и должна при выходе запоминать свой контекст с помощью некоторой системы флагов или переключателей. Для преобладающего числа алгоритмов это значительно усложняет логику подпрограммы и затрудняет ее разработку. Очевидно, что разработчику всегда удобнее считать программу головной.

Эффективный способ избежать перечисленных затруднений — организация всех модулей ввода, вывода и переработки информации по принципу сопрограмм с динамической настройкой. Для каждого возможного источника и приемника данных, включая автономные носители, а также для всевозможных алгоритмов обработки разрабатывается всего один модуль. Далее во время работы системы (не трансляции или загрузки!) оператор может набирать из них желаемые комбинации, например: чтение, несколько последовательных алгоритмов обработки, каждый из которых работает над результатом предыдущего, и затем запись.

На рис. 2 приведены возможные типы модулей такой системы. Функциональные модули уже обсуждались выше. Входные модули предназначены для чтения данных (каждый со своего носителя) и передачи их другому модулю, причем модуль назначения определяется оператором уже при исполнении. Проходные модули представляют собственно алгоритмы преобразования данных. Они «читают» информацию, поступающую от предыдущего модуля, и передают ее следующему. Наконец, выходные модули предназначены для вывода данных на некоторый носитель. Входные и выходные модули будем называть терминальными.

Пример настройки модулей в цепочку показан на рис. 3. Здесь *A*, *D* — терминальные модули, *B*, *C* — проходные. Когда одному из модулей необходима следующая порция информации, он производит запрос *READ* (это специальная точка входа в управляющей программе). После этого управление в этот модуль возвращается управляющей программой, когда необходимая информация будет получена от стоящего левее по цепочке модуля, и возврат происходит в точку непосредственно за запросом *READ*. Аналогично, накопив очередную порцию данных, модуль производит запрос *WRITE* и управление возвращается ему, когда стоящему правее по цепочке модулю потребуется следующая порция данных. Как видно из приведенного описания, каждый модуль может считаться головной программой, а *READ* и *WRITE* можно рассматривать как подпрограммы чтения и записи, причем одни и те же для всех модулей. Такая организация позволяет при разработке модуля не знать его «соседей», так как забота об этом ложится на управляющую программу.

В таблице информации о модулях содержится признак, определяющий тип модуля, и настройка происходит автоматически, когда

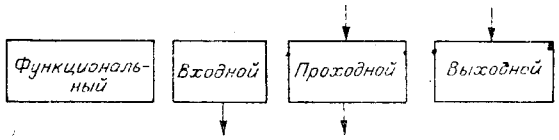

Рис. 2. Типы модулей.

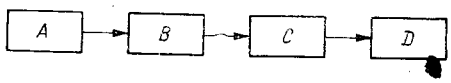

Рис. 3. Пример настройки.

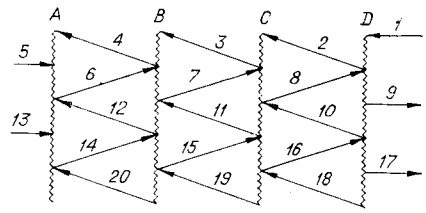

Рис. 4. Схема передачи управления.

оператор вводит соответствующую модулю директиву. Настройка цепочки сводится к формированию вектора переходов, который состоит из записей, обязательным полем которых является адрес передачи управления, устанавливаемый во время настройки на точку входа в модуль. Кроме того, здесь могут содержаться значения базисных регистров и другая информация, необходимая для правильной передачи управления. После того, как в цепочке появится выходной модуль, она готова к работе. Запуск происходит по специальной директиве *GO*, по которой управление передается выходному модулю.

На рис. 4 изображена принципиальная схема передачи управления при работе цепочки, показанной на рис. 3. Стрелками изображены передачи управления и пути следования данных, а волнистыми линиями (условно) — работа модуля. Необходимо отметить также, что обменные переходы от модуля к модулю реализуются управляющей программой, так что стрелки на рис. 4 также условны. При первой передаче управления модулю ему передаются также указанные в директиве параметры.

Итак, после запуска цепочки управление передается выходному модулю *D* (см. *1* на рис. 4). Вначале выходной модуль делает подготовительную работу: принимает параметры, открывает выходной файл и т. п., после чего по запросу *READ* управление попадает в точку входа модуля *C 2*. Модуль *C*, проделав подготовительную работу, также производит запрос *READ 3*, и этот процесс повторяется, пока управление не попадает в точку входа входного модуля *A 4*. Входной модуль принимает параметры, открывает входной файл, читает одну запись данных *5* и по запросу *WRITE* управление передается системой 6 в модуль *B*, в точку, где он произвел ранее запрос *READ*. Это продолжается *7*, *8* до тех пор, пока управление не попадает в выходной модуль, в точку, где он в последний раз передал управление системе. Выходной модуль переписывает поступившие данные на внешний носитель *9* и снова производит запрос *READ 10*. Далее процесс повторяется, пока не будет переработана вся информация. Первая порция данных проходит по стрелкам *5—9*, вторая — по стрелкам *13—17* и т. д.

Запросы *READ* и *WRITE* в одном модуле не обязательно должны чередоваться, а могут поступать асинхронно. Например, модуль, сжимающий информацию, может произвести несколько запросов *READ* и один запрос *WRITE*.

Приведенный подход позволяет разрабатывать модули обработки информации, по существу, не зависящие от источников информации, так как в такой схеме имеется всего один логический источник — подпрограмма *READ*, а физическими источниками могут быть любые носители данных, которые определяются уже во время работы системы.

Появляется также возможность применять последовательно несколько алгоритмов обработки без перезаписи промежуточной информации. В то же время цепочку можно в любом месте «разорвать», снабдив ее в месте разрыва модулями записи и чтения для носителя с высокой скоростью доступа. Эту операцию управляющая программа может делать автоматически, если все модули цепочки не помещаются в оперативную память или два модуля находятся в различных сегментах, перекрывающих одну область в памяти.

Практика использования такого подхода показала его эффективность в задачах обработки изображений, статистической обработки, при работе с текстовой информацией.

Язык управления. Для управления системой предлагается язык директив, обеспечивающий простоту в принятии решений. Формат директив следующий:

параметр 0, параметр 1, ..., параметр N

Параметр 0 определяет имя директивы, а остальные — дополнительную информацию, которая будет передана модулю: числа, текстовые строки, восьмеричные и шестнадцатиричные константы и т. п. Некоторые из параметров могут опускаться. Для этого необходимо ввести подряд две запятые, а по умолчанию передается значение параметра, определенное во время генерации системы (таблицы информации о модулях).

Как уже упоминалось ранее, из директив может составляться программа, которая будет исполняться автоматически. Отдельная директива в программе выступает уже в роли макрокоманды. В программе могут использоваться все без исключения директивы, которые можно вводить с пульта оператора; кроме того, имеются некоторые дополнительные возможности.

Макрокоманды могут помечаться метками, на которые возможен переход:

```
M:...
...
JUMP. M
```

Некоторые последовательности макрокоманд могут быть выделены в циклы, например, в случае

```
DO, N
...
...
END
```

последовательность макрокоманд, заключенная между *DO* и *END*, повторяется N раз. В позициях параметров типа целых допускаются простые арифметические выражения:

```
LET, N=1
```

Параметр в макрокоманде перехода может иметь вид $*+N$ или $*-N$; в этом случае переход осуществляется по номеру макрокоманды относительно самой макрокоманды перехода. В выражениях в параметрах других макрокоманд символ * представляет код возврата предыдущего исполняемого модуля. Возможен также переход по переключателю:

```
GO, N, M1, M2
GO, *, M, *-5, *+2
```

В первом случае в зависимости от значения переменной N переход осуществляется на метку $M1$, если $N=1$; на метку $M2$, если $N=2$; иначе — на следующую макрокоманду. Во втором случае происходит передача управления по значению кода возврата предыдущего модуля.

Заключение. Необходимо отметить требования к ЭВМ, допускающей реализацию такой системы. Реализация полных возможностей управления модулями может быть проведена на ЭВМ М-6000 или

ЕС-1010 с памятью 16K слов и дисковой операционной системой (см., например, [5]); в [1, 2] описана реализация программного комплекса на ЭВМ без дисков, построенного по этим же принципам и допускающего проведение нескольких экспериментов в реальном времени.

ЛИТЕРАТУРА

1. Э. А. Талныкин. Программное обеспечение экспериментального информационно-измерительного комплекса.— В кн.: Вопросы построения систем автоматизации научных исследований. Новосибирск, Изд. ИАиЭ СО АН СССР, 1974.
2. С. М. Казаков, Э. А. Талныкин. Программное обеспечение машинного комплекса обработки экспериментальных данных.— В кн.: Системы автоматизации научных исследований. (Тезисы докладов конференции). Рига, «Зинатне», 1973.
3. С. В. Бредихин, А. Н. Гинзбург, Б. А. Мелешихин, П. М. Песляк, Э. А. Талныкин. Принципы построения программного обеспечения магистрального информационно-измерительного комплекса.— В кн.: Труды III Всесоюзного симпозиума «Системное и теоретическое программирование». Кишинев, Изд. КГУ, 1974.
4. Л. А. Андрианов, М. А. Ахмаметьев, П. Я. Белоусов, Ю. Г. Кириллов, А. М. Остапенко, С. Н. Ремесленникова, Э. А. Талныкин. Автоматизированная система с управлением и обработкой на базе ЭВМ HP2116B для изучения динамики флюоресценции монослоя живых клеток.— «Автометрия», 1975, № 2.
5. А. М. Остапенко, Э. А. Талныкин, Н. С. Яковенко. ФОТ — диалоговая система обработки данных.— «Автометрия», 1976, № 1.

Поступила в редакцию 25 июня 1975 г.

УДК 681.3.06

А. М. ОСТАПЕНКО, Э. А. ТАЛНЫКИН, Н. С. ЯКОВЕНКО

(*Новосибирск*)

ФОТ — ДИАЛОГОВАЯ СИСТЕМА ОБРАБОТКИ ДАННЫХ

Введение. В статье описывается конкретная реализация диалоговой системы обработки данных ФОТ, построенной на принципах, изложенных в [1], откуда будет взята введенная там терминология, поэтому читателю целесообразно сначала прочесть работу [1]. Более подробно будет описана структура системы и две подсистемы модулей, предназначенные для работы со стандартными носителями информации и микроденситометром "Fotomation", представляющим собой цифровую систему с высоким разрешением для считывания и записи полутоновых изображений на фотопленку. Организация системы в совокупности с предлагаемым набором модулей полностью избавляют разработчика алгоритмов обработки от необходимости программировать операции над данными, располагающимися на внешних носителях. Программы обработки в системе ФОТ, по существу, не зависят от источников информации, которые определяются уже при работе системы и могут быть реальными изображениями, их копиями на одном из носителей или результатами работы других алгоритмов. Для каждой из операций: чтение, запись или конкретный алгоритм обработки — разрабатывается один программный модуль. Далее в режиме диалога оператор может набирать из имеющихся модулей всевозможные комбинации, такие, например, как чтение изображения с некоторого носителя, несколько последовательных алгоритмов обработки, каждый из которых работает над результатом предыдущего, и запись на другой