2. Виттих В.А., Симановский Е.А., Цыбатов В.А. Методы и средства анализа ИВК. - В кн.: Автоматизация и научное приборостроение -83: Сб.докл. П Международного сипоз.- Варна (НРБ), 1983, с.7-22.

3. Виттих В.А., Сидоров А.А., Симановский Е.А. Анализ процессов сбора и обработки данных в автоматизированных системах научных исследований методами имитационного моделирования.-Управляющие системы и машины, 1983, № 3, с.89-93.

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ АВТОМАТИЗИРОВАННЫХ СИСТЕМ НАУЧНЫХ ИССЛЕДОВАНИЙ И ОБУЧЕНИЯ

УДК 681.3.06:51

А.Н.Ковшов, Е.Ю.Шахтарин

ПРОБЛЕМНОЕ ПРОГРАММИРОВАНИЕ ЗАДАЧ АСНИ

(г.Свердловск)

Развитие средств автоматизации программирования ориентировано на привлечение все большего числа неквалифицированных пользователей. К сожалению, такая ориентация до сих пор остается в большей степени пожеланием и рекламными заверениями разработчиков этих средств. Стремление создавать ясные и легкие для понимания программы удобными простыми средствами наталкивается на серьезное препятствие. Суть трудностей состоит в общем для языков программирования недостатке. Как универсальные типа ПЛ/I, так и машинно-ориентированные типа АССЕМБЛЕР языки программирования реализуют компромисс между ориентированностью на машину и ориентированностью на пользователя. С одной стороны, для всех языков характерна однотипная основа, отражающая структуру современных ЭВМ. Какой бы сложности ни была решаемая задача, программист вынужден выражать проблемные категории посредством набора элементарных вычислительных действий- условный оператор, цикл, оператор присваивания. Декомпозиция категорий задачи при помощи инструментария такого уровня весьма нетривиальна и требует богатого опыта и высокой квалификации.

Современные вычислительные машины характеризуются двумя принципиальными моментами:во-первых,процесс описывается в терминах действий в порядке их вычисления;во-вторых,данные сохраняются в ячейках памяти,значения которых периодически изменяются соответствующим действием. Эти свойства находят отражение в языках программирования в виде понятия передачи управления и оператора присваивания. Такая машинная ориентация несомненно способствует эффективности программ. Но способствует ли она удобству и эффективности программирования?

Программу создают, основываясь на соображениях ее реализации на выбранном языке. Однако особого выбора нет - программист имеет дело с языками, непосредственно отражающими структуру машины в указанном смысле. Вместо того чтобы заниматься алгоритмизацией задачи по существу, программист концентрирует свое внимание на технических приемах вычислений на ЭВМ, хотя такая ситуация программистом не осознается.

Предлагается в явной форме выделить следующий принцип: составлять алгоритм исходя из структуры задачи, а не из соображений реализации программы на ЭВМ. Необходимо программировать задачу, а не машину.

Бессмысленно предлагать какой-то новый язык для внедрения в практику этого принципа - требуется иной подход.

По-видимому, одномоментного решения проблемы достигнуть нельзя. Наиболее подходящим для реализации упомянутого принципа представляется метод функционального программирования, ранее в основном известный в теоретическом программировании, начинающий постепенно перемещаться в практику [1]. Сторонники этого подхода утверждают,что функциональные программы яснее выражают свои цели, чем традиционные, и поэтому легче для понимания, эксплуатации и,главное, их легче составлять.

Основа функционального языка - понятие функции как операции. Функции связываются в выражения с помощью функциональных форм. Простейшим примером функциональной формы является обычная композиция функций. Значение выражения единственным образом определяется по значениям составляющих его частей.

Функциональный язык имеет небольшое число основных понятий с простой семантикой. Конструкция выражений очень проста. Семантика языка понимается полностью в терминах значений, которыми обладают выражения, а не в терминах действий в порядке вычисления, поэтому

отсутствуют операторы передач управления и присваивания. Возникающая в связи с этим простота семантики делает возможным ясно описывать вычисления в этом языке.

Недостатком функциональных программ является менее эффективное использование вычислительной машины, но в связи с улучшением технологии производства ЭВМ это будет иметь все меньшее значение.

Продемонстрируем на примере основные черты функционального стиля. Используемая в данном случае нотация синтаксически близка к изложенной в [2] . Отметим, что функциональный подход допускает разнообразные нотации, сравнительное обсуждение которых выходит за рамки статьи.

Определим объект как структурированный кортеж. Например, вектор трех чисел есть кортеж (4,5,6). Матрица - кортеж из двух векторов: ((1,2,3), (4,5,6)). Введем набор элементарных функций, в их числе "+" - сложение, " * " - умножение, "транс" - транспонирование. Двоеточием обозначим применение функции к аргументу, например:

+ : (2,3)

Равенством обозначим тождество значений

+ : (2,3) = 5

транс: ((1,2), (3,4) = ((1,3), (2,4))

Отсутствующий аргумент для сложения полагаем равным нулю (в случае умножения - единице):

+ : (3) = + : (0,3) = 3

В качестве функциональных форм рассмотрим (X -объекты, Φ - функции, § % и. - функциональные формы)

композицию

$\Phi 1.\Phi 2 : X = \Phi 2 : (\Phi 1 : X)$

редукцию

% Ф : (X1,X2,...,XK) = Ф : (X1,Ф : (X2,...,Ф:(XK)...))

и групповое применение

§ Ф : (X1,X2,...,XK) = (Ф : X1,Ф : X2,...,Ф : XK)

Тогда алгоритм сп - вычисления скалярного произведения векторов - составить пары соответствующих компонент векторов, перемножить, результаты сложить - выглядит следующим образом:

сп = транс . § * . % +

Применение программы к конкретному объекту, например,

X = ((1,2,3), (4,5,6)) и этапы вычисления функционального выражения выглядят так:

```
сп = транс . §   . % +
х = ( (I,2,3), (4,5,6) )
у = сп : х =
    сп : ( (I,2,3), (4,5,6 ) ) =
транс . § * . % + : ( (I,2,3), (4,5,6) ) =
§ * . % + : (-(I,4), (2,5), (3,6) ) =
% + : (4,I0,I8) = 32
```

Из соображений эффективности для проблемных приложений АСНИ не имеет смысла создавать универсальный функциональный язык. Скорее всего, определенный класс задач АСНИ потребует соответствующего языка. Необходимо выбрать объекты, определить базисные функции, соответствующие элементарным операциям над объектами, и определить множество функциональных форм, которые соответствуют способам структуирования задачи. Возможны такие системы, в которых допускается построение новых функциональных форм [2] .

Универсальность арифметических функций и композиции очевидна. Не менее фундаментальны редукция и групповое применение. Продемонстрируем применение последних в задаче иного класса.

Допустим, что имеется прибор, который выдает в систему сообщение вида:имя прибора,измеренная величина.Серия испытаний состоит в накоплении множества поступающих в произвольном порядке сообщени от К приборов (XI,...,ХК), где Х - сообщение. В ходе работы системы накапливается комплект из Т серий (УI,...,УТ). В целях обработки необходимо каждую серию упорядочить по виду сообщения, т.е. по имени прибора.

Допустим, что функциональный базис содержит элементарную функцию вкл - не нарушающее порядок включение объекта в упорядочен ное (по атрибуту А) множество:

```
вкл: ( (Х,У), ( (AI,BI),...(AH,BH)  )   (АК - I Х  АК ) =
( (AI,BI),...,(АК-I, ВК-I), (Х,У), (АК,ВК),...,(AH,BH) )
вкл: (Х,У) = ( (Х,У) )
```

Тогда задача "каждую серию упорядочить" решается следующим алгоритмом:

```
    § % вкл: (УI,...,УТ)
    Например,
§ % вкл:(( (МАФС,287), (АЦП,I), (МАФС,200) ),
        ( (АЦП,5), (МАФС,I50), (АЦП,3) ) ) =
        ( % вкл: ( (МАФС,297),(АЦП,I), (МАФС,200) ),
          % вкл: ( (АЦП,5), (МАФС,I50), (АЦП,3) ) )=
```

```
    (вкл : ( (МАФС,287), вкл : ( (АЦП,I),вкл :
     : (МАФС,200) ) ),
вкл : ( (АЦП,5), вкл : ( (МАФС,I50), вкл : (АЦП,3) ) ) ) =
(вкл : ( (МАФС,287), вкл : ( (АЦП,I), ( (МАФС,200) ) ) ),
 вкл : ( (АЦП,5), вкл : ( (МАФС,I50),( (АЦП,3) ) ) ) ) =
(вкл : ( (МАФС,287),( (АЦП,I), (МАФС,200) ) ),
 вкл : ( (АЦП,5),((АЦП,3), (МАФС,I50) ) ) ) =
( ( (АЦП,I),(МАФС,287).(МАФС.200) ),
((АЦП,5),(АЦП,3),(МАФС,I50) ) )
```

При решении очерченного класса проблемных задач еще большей наглядности и компактности следует ожидать от применения специализированных функциональных форм. На наш взгляд, допущение диктуемого задачей многообразия функциональных форм является принципиальным моментом, поскольку именно в этом направлении видится возможность наиболее полной реализации принципа программирования от проблемной задачи. Функциональные формы являются способом структурирования самой задачи, в отличие от того как циклы, условные операторы и т.п.- это способы структурирования программы для ЭВМ.Независимо от применения функционального стиля или иной методики при алгоритмизации должен отрабатываться этап структуризации задачи, аналогичный выдвижению функциональных форм.

Изложенный подход алгоритмизации проблемных задач ориентирован прежде всего на их постановку. Следует рассматривать его как связующее звено между неформализованной постановкой задачи и программой для ЭВМ.

Инструментальная поддержка метода требует решения вопросов эффективности создаваемых функциональных программ. Поэтому инструментальная система должна содержать универсальное ядро, которое развивается в нужном проблемном аспекте не только внутренними средствами,а также должна быть открытой для расширения внешними машинно ориентированными пополнениями.

Литература

1. Хендерсон П. Функциональное программирование. Применение и реализация.-М.:Мир, 1983.-439 с.

2. Backus J. Can programming be liberated from the von Neumann style? A functional style and its algebra of programs. Comm. ACM 21(8), 1978, 613-641.