

ОБ АЛГОРИТМАХ ДЛЯ ПЕРЕРАБОТКИ ИНФОРМАЦИИ

А. А. Ляпунов

Излагается способ изображения алгоритмов, который удобно использовать при программировании различных задач.

В кибернетике большую роль играют алгоритмы для переработки информации. Во многих случаях возникает вопрос о реализации того или иного алгоритма, перерабатывающего информацию в вычислительной машине. При этом требуется составить такую программу работы машины, чтобы, каково бы ни было допустимое сообщение на входе машины, на выходе должно появиться сообщение, являющееся переработкой входного сообщения, согласно данному алгоритму.

Мы будем различать понятия «сообщения» и «информации». Сообщением будем называть индивидуальную систему сигналов; информацией — совокупность всех возможных сообщений.

Имеются два вида постановок задач, относящихся к изучению информации,—теоретико-множественная и теоретико-вероятностная. При втором подходе каждое из возможных сообщений рассматривается с ответствующей ему вероятностью.

Когда ставится вопрос о переработке информации, то имеется в виду общий закон переработки всех возможных сообщений, образующих данную информацию. Таким образом, речь идет о законах, дающих возможность решения задач массового характера. Индивидуальная задача возникает тогда, когда уточняется индивидуальное исходное сообщение. Обычно такие законы переработки информации — их называют алгоритмами для переработки информации — представляют собой сводки отдельных правил, каждое из которых способно переработать вполне определенным образом некоторые сообщения, и наборы условий, определяющих порядок применения указанных правил к перерабатываемому сообщению. Примерами таких алгоритмов могут служить алгоритм умножения целых чисел, составляемый из умножений на число, заключенное между 0 и 9, сдвигов слагаемых относительно друг друга и сложений, а также алгоритм Эвклида — нахождения общего наибольшего делителя двух чисел или двух многочленов и т. п.

Теория алгоритмов представляет собой сравнительно новую и очень содержательную ветвь современной математики. В этой области дается точное определение понятия алгоритма, и в связи с этим появляется возможность выяснить вопрос о том, какие типы массовых задач допускают алгоритмическое решение и какие не допускают. Вне этой области понятие алгоритма обычно используется чисто описательно. Однако в связи с развитием автоматических вычислительных машин возникает необходимость широкого использования алгоритмов переработки информации. Становится неудобным ни их чисто описательное использование, ни слишком высокая степень формализации, применяемая в общей теории алгоритмов.

Целью настоящей заметки является изложение такого способа изображения алгоритмов, который оказался удобным для программирования, т. е. вложения разнообразных алгоритмов в машины. Этот способ используется как для программирования вычислительных задач, так и для программирования различных неарифметических алгоритмов, служа-

щих для переработки некоторой информации с помощью вычислительных машин. Обычно такой алгоритм распадается на ряд элементарных актов, каждый из которых выполняется по вполне определенным и достаточно простым правилам, быть может зависящим от целочисленных значений некоторых параметров, и проверки некоторых элементарных условий, определяющих порядок выполнения этих актов. (Эти условия в свою очередь могут содержать некоторые параметры). После выполнения всех актов, отвечающих данному значению параметра, и проверки условий, ему соответствующих, происходит изменение значения данного параметра. Порядок выполнения отдельных актов и проверки необходимых условий, а также порядок последовательных изменений значений параметра заранее точно определен. Это и составляет предложенный алгоритм.

Будем представлять отдельные элементарные акты большими латинскими буквами с индексами, обозначающими параметры, от которых данный акт зависит. Мы будем называть их элементарными операторами. Логические условия будем обозначать малыми латинскими буквами. После каждого логического условия ставится стрелка, служащая для обозначения того, куда следует перейти, если условие не выполнено. Отметим, что элементарные операторы суть функции, преобразующие некоторые множества допустимых сообщений, а логические условия—это предикаты, определенные на множествах сообщений.

Опишем процесс синтеза более сложных операторов из более простых операторов и логических условий. Последовательное выполнение операторов мы будем обозначать как произведение операторов, причем последующий оператор ставится правым множителем. Если в строчке операторов — сомножителей стоит некоторое логическое условие, то, если условие выполнено, мы должны перейти к непосредственно следующему оператору. Если условие не выполнено, то мы переходим к тому оператору или логическому условию, к которому нас ведет стрелка. Так, например, оператор

$$Ap \overline{\overline{BC}} \downarrow$$

при $p=1$ есть ABC, а при $p=0$ есть A.

Если логическое условие состоит в проверке некоторого равенства или неравенства, то проверяемое выражение ставится в качестве аргумента. Так,

$$\overline{\overline{A_i \cdot p (i > n)}} = A_1 \cdot A_2 \cdot \dots \cdot A_n.$$

Такая символика позволяет изображать алгоритмы в виде произведений операторов и логических условий, содержащих параметры. Эти изображения мы будем называть логическими схемами алгоритмов.

Возможны случаи, когда некоторые операторы изменяют значения каких-то параметров или заменяют одни из имеющихся операторов или логических условий другими. Мы будем характеризовать тип оператора или алгоритма тем множеством допустимых сообщений, которые он преобразует. Операторы, преобразующие числовые данные некоторой вычислительной задачи, носят название операторов счета. Они синтезируются из элементарных операторов счета и логических условий задачи.

Переработка состояний памяти в машине определяется программой. Программа представляет собой некоторую последовательность приказов. Если некоторый алгоритм счета реализуется в машине, то каждое его логическое условие и каждый оператор счета должны быть реализованы с помощью приказов машины. Однако для того, чтобы машина могла автоматически реализовать весь алгоритм в целом, этого еще недоста-

точно. Нужно построить ряд операторов управления — это группы приказов, назначение которых состоит в том, чтобы подготовить состояние памяти машины для выполнения очередного оператора счета. Чаще всего это операторы, изменяющие значения параметров. Кроме того, это могут быть операторы переноса исходных числовых данных, операторы формирования очередных операторов, операторы переадресации, операторы ввода параметров (если одна часть программы вырабатывает значение параметра, от которого зависят операторы другой части программы) и т. п.

Произведение операторов счета, логических условий и операторов управления, построенное так, что их выполнение в машине обеспечивает решение исходной задачи согласно предложенному алгоритму счета, называется схемой программы [1]. Схема программы есть алгоритм,рабатывающий не только исходные данные задачи, но и операторы счета.

В настоящее время техника программирования состоит в том, что сначала строится схема счета, затем — схема программы. По этой последней программа расписывается уже механически. Э. З. Любимский, С. С. Камынин и другие [2] под руководством М. Р. Шура-Буры построили программирующую программу, которая строит рабочую программу по данной схеме программы и необходимым сведениям об операторах счета и логических условиях. Ю. И. Янов разработал формальный аппарат для преобразования схем алгоритмов [3, 4, 5, 6]. Р. И. Подловченко [7] предложила метод формальной замены параметров в схемах программ.

В качестве примера рассмотрим задачу о приведении системы линейных алгебраических уравнений к диагональному виду без выбора главных элементов:

$$\sum_{j=1}^n a_{ij}x_j = a_{in+1}.$$

Пусть B_{ij} вычисляет $c = \frac{a_{ji}}{a_{ii}}$ и ставит его на стандартное место, A_{ijk} вычисляет $a'_{jk} = a_{jk} - ca_{ik}$ и ставит его на место a_{jk} , С — заносит нуль на место c . Схема счета имеет вид:

$$\prod_{l=1}^n \prod_{j=1}^n B_{lj} p(i=j) \overline{\downarrow C} \prod_{k=1}^{n+1} A_{ljk},$$

знак $\prod_{l=1}^n$ означает произведение операторов в порядке возрастания параметров от $l=1$ до $l=n$.

Схема программы может быть записана так:

$$\{1 \rightarrow i\} \overline{\uparrow \{1 \rightarrow j\} \downarrow B_{lj} p(i=j) \overline{\uparrow C \downarrow \{1 \rightarrow k\} \downarrow A_{ljk} F(k)p(k > n+1)} \overline{\uparrow F(j)p(j > n)} \times \\ \times F(i)p(i > n). \quad \text{Ост.},$$

где $F(l)$ обозначает оператор, увеличивающий параметр l на единицу, а $\{1 \rightarrow l\}$ обозначает оператор, вносящий единицу на место параметра l .

ЛИТЕРАТУРА

1. А. А. Ляпунов. О логических схемах программ, Труды III Всесоюзного математического съезда т. I, 1956.
2. А. П. Ершов, Э. З. Любимский, С. С. Камынин. Автоматизация программирования, Труды III Всесоюзного математического съезда, т. II, 1956.
3. Ю. И. Янов. О равносильности и преобразованиях схем программ. Труды III Всесоюзного математического съезда, т. I, 1956.
4. Ю. И. Янов. ДАН, 1957, 113, № 1.
5. Ю. И. Янов. ДАН, 1957, 113, № 2.
6. Ю. И. Янов. Статья печатается в настоящем сборнике, стр. 110—119.
7. Р. И. Подловченко. Проблемы кибернетики, вып. I, 1958.

Математический институт
им В. А. Стеклова АН СССР

Поступила в редакцию
27 июня 1957 г.