быть перенесена в следующий разряд. Легко проверить, что все возможные здесь случаи исчерпываются следующей таблицей:

a	0	1	0	0	1	1	0	1
b	0	0	1	0	1	0	1	1
c	0	0	0	1	0	1	1	1
s	0	1	1	1	0	0	0	1
p	0	0	0	0	1	1	1	1

(*)

Отсюда ясно, что для осуществления операции суммирования в пределах одного разряда мы должны иметь в машине устройство с тремя входами (отвечающими цифрам a, b и c) и двумя выходами (отвечающими цифрам s и p), работающее в соответствии с таблицей (*), т. е. так, что если ни на один из входов не подается напряжения, то на выходах s и p напряжения тоже нет, если напряжение подается на один из входов, то оно есть на выходе s и отсутствует на выходе p, и т. д. Устройство, действующее по этим правилам, называется *одноразрядным сумматором*. Такое устройство нетрудно фактически реализовать в виде некоторой радиотехнической схемы, составленной из радиоламп или из полупроводниковых элементов. Мы, однако, не будем приводить эти схемы.

§ 3. Элементы программирования

1. Общие сведения. Для решения задачи на УЦВМ весь ход этого решения должен быть представлен как некоторая последовательность тех элементарных операций, которые данная машина может выполнять. Выполнение каждой операции определяется соответствующей командой, а последовательность команд, отвечающая решению данной задачи, называется *программой*. Программирование, т. е. составление программы, — один из основных этапов решения задачи на УЦВМ. Ясно, конечно, что прежде, чем приступать к программированию, нужно выбрать определенный математический метод решения задачи и получить те конкретные формулы, по которым должен происходить расчет.

Программа зависит от тех численных методов, которые мы выбрали для решения задачи (например, для приближенного вычисления интеграла мы можем пользоваться формулой трапеций, формулой прямоугольников или каким-либо иным приемом), и от типа машины, т. е. от набора тех операций, которые может выполнять данная машина. Однако даже если метод счета и тип машины определены, то программа этим еще не определяется однозначно: мы можем разложить

нашу вычислительную задачу в последовательность элементарных операций различными способами. Выбор наиболее рациональной программы для той или иной задачи определяется в значительной мере квалификацией лица, составляющего программу. Мы не можем здесь останавливаться на вопросах программирования сколько-нибудь подробно и ограничимся лишь разбором простейших типичных примеров.

2. Программирование по формулам. Наиболее простой для программирования тип задач — это вычисление по формулам, сводящееся к последовательному выполнению ряда арифметических операций. В этих случаях программирование сводится к рациональному разбиению всей формулы на отдельные операции, к размещению соответствующих команд и исходных данных в памяти машины. Рассмотрим элементарный пример.

Пример. По данному x вычислить значение

$$y=\frac{2x+3}{5x+1}.$$

Все вычисление можно, очевидно, представить в виде такой последовательности элементарных операций:

$$1)\ A_1=2x,\quad 2)\ A_2=A_1+3,\quad 3)\ B_1=5x, \tag{1}$$
$$4)\ B_2=B_1+1,\quad 5)\ y=\frac{A_2}{B_2}.$$

Для того чтобы произвести эти операции на машине, расположим в пяти ячейках памяти машины (скажем, в ячейках с номерами $n+1 \div n+5$) исходные данные. Получим

№ ячейки	Записанное число	№ ячейки	Записанное число
$n+1$	x	$n+4$	5
$n+2$	2	$n+5$	1
$n+3$	3		

В других пяти ячейках запишем команды, отвечающие тем действиям, которые указаны в равенствах (1). Получим такую последовательность команд:

№ ячейки	Операция	1-й адрес	2-й адрес	3-й адрес	Результат операции
$m+1$	Умножить	$n+1$	$n+2$	$n+2$	$2x$
$m+2$	Сложить	$n+2$	$n+3$	$n+2$	$2x+3$
$m+3$	Умножить	$n+1$	$n+4$	$n+1$	$5x$
$m+4$	Сложить	$n+1$	$n+5$	$n+1$	$5x+1$
$m+5$	Разделить	$n+2$	$n+1$	$n+2$	$\frac{2x+3}{5x+1}$

В тот момент, когда тот или иной промежуточный результат перестает быть нужен для дальнейших вычислений, мы можем соответствующую запись «стереть» и использовать содержавшую его ячейку для новой записи. Так мы поступили, например, при выполнении первой команды, записав произведение чисел, хранившихся в ячейках $n+1$ и $n+2$, снова в $(n+2)$-ю ячейку. Это позволяет рациональнее использовать объем памяти машины, не загружая ее ненужными для дальнейшего данными.

Составленную нами программу нужно еще дополнить вначале командой ввода, по которой начальные данные и коды программ вводятся в память машины, и командой перевода исходных данных из десятичной системы в двоичную, поскольку все операции в машине выполняются в двоичной системе, а исходные данные записываются и вводятся в машину обычно в десятичной системе. Далее, после команды «разделить», записанной в $(m+5)$-й ячейке, необходимо поместить еще три команды. По первой из них результат вычисления переводится из двоичной системы в десятичную, по второй печатается ответ, и, наконец, последняя команда — это прекращение работы машины — останов.

Последний шаг в написании программы — это замена буквенных обозначений адресов конкретными числами. Эти номера пишутся четырехзначными числами в восьмеричной системе, начиная с 0000 и т. д. Обычно первые ячейки памяти используются как рабочие ячейки для стандартных операций (ввод, перевод числа из одной системы в другую и т. д.). Например, в машине «Стрела» для этой цели отведены первые 11 ячеек (от 0001 до 0013 в восьмеричных обозначениях) *). Начав заполнение ячеек памяти с 0014, запишем окончательно нашу программу в таком виде:

№ ячейки	Операция или число	1-й адрес	2-й адрес	3-й адрес
0014	Ввод (в ячейки 0015—0032)	0015	0015	
0015	Перевод из десятичной системы в двоичную (ячейки 0026—0032)	0026	0004	0026
0016	Умножение	0026	0027	0027
0017	Сложение	0027	0030	0027
0020	Умножение	0026	0031	0026
0021	Сложение	0026	0032	0026
0022	Деление	0027	0026	0027
0023	Перевод из двоичной системы в десятичную	0027		0027
0024	Печать	0027		

*) При этом ячейка с номером 0000 содержит число «0».

Продолжение

№ ячейки	Операция или число	1-й адрес	2-й адрес	3-й адрес
0025	Останов			
0026	x			
0027	2			
0030	3			
0031	5			
0032	1			
0033				

3. Циклические процессы. Ясно, конечно, что в случаях, подобных только что рассмотренному элементарному примеру, никакого практического смысла применение УЦВМ не имеет. Мы рассмотрели этот пример лишь для того, чтобы показать, как в самом простейшем случае привычные нам формулы переводятся на язык, «понятный» машине. Применение вычислительных машин с программным управлением оказывается эффективным лишь в тех случаях, когда число операций, выполняемых машиной, велико по сравнению с числом команд, которые мы должны фактически ввести в память машины. Во многих задачах такое многократное использование одних и тех же команд достигается благодаря тому, что соответствующая вычислительная схема состоит из многократных повторений отдельных серий операций. Эти повторяющиеся серии называются *циклами*, а соответствующие вычислительные схемы называются *циклическими*. Рассмотрим некоторые простейшие примеры циклических программ.

1) *Вычисление квадратного корня.* Предположим, что мы должны вычислить с заданной точностью квадратный корень из некоторого положительного числа a. Для решения этой задачи можно воспользоваться следующим фактом (см. вып. 1, гл. 3). Каково бы ни было положительное число a, последовательность,

$$x_0 = a;\ x_1 = \frac{1}{2}\left(x_0 + \frac{a}{x_0}\right);\quad x_2 = \frac{1}{2}\left(x_1 + \frac{a}{x_1}\right);\ \ldots$$

$$\ldots;\ x_{n+1} = \frac{1}{2}\left(x_n + \frac{a}{x_n}\right) \qquad (2)$$

сходится и ее предел равен $\sqrt{a}$. Вычисляя последовательно x_1, x_2, ... и т. д., мы можем продолжить процесс до тех пор, пока не будет достигнута некоторая заданная точность, например до тех пор, пока разность между x_n и предыдущим значением x_{n-1} не станет меньше заданной величины.

Следовательно, для вычисления $\sqrt{a}$ на машине мы должны ввести в три ячейки памяти число $a = x_0$, принятое нами за нулевое

приближение, число ε, определяющее точность, и число $\frac{1}{2}$. Далее вычисление $\sqrt{a}$ осуществляется по следующей программе:

№ ячейки	Название операции	1-й адрес	2-й адрес	3-й адрес	Результат операции
0014	Ввод	0016	0015		Ввод массива
0015	Перевод $10 \to 2$	0030	0003	0030	Перевод исходных данных в двоичную систему
0016	Сложение поразрядное	0031		0027	Засылка x_n из ячейки «0031» (где x_n остается) в «0027»
0017	Деление	0030	0027	0031	$\frac{a}{x_n}$
0020	Сложение	0031	0027	0031	$x_n + \frac{a}{x_n}$
0021	Умножение	0031	0033	0031	$x_{n+1} = \frac{1}{2}\left(x_n + \frac{a}{x_n}\right)$
0022	Вычитание	0031	0027	0027	$x_{n+1} - x_n$
0023	Пер. управления $\lvert \leqslant \rvert$	0032	0027	0016	Проверка того, достигнута ли заданная точность (сравнение $x_{n-1} - x_n$ и ε), и окончание цикла, если точность достигнута
0024	Перевод $2 \to 10$	0031		0031	Перевод результата в десятичную систему
0025	Печать	0031			Печать результата
0026	Останов				
0027					Рабочая ячейка
0030	a				
0031	x_0				
0032	ε				
0033	$\frac{1}{2}$				
0034					

2) *Составление таблиц функций.* Другой типичный пример циклического процесса счета — это вычисление значений различных функций — показательной, тригонометрической, логарифмической при различных значениях аргумента, т. е. составлениие таблиц элементарных функций.

Рассмотрим, например, функцию $\sin x$. По формуле Тейлора

$$\sin x = x - \frac{x^3}{3!} + \ldots + (-1)^n \frac{x^{2n+1}}{(2n+1)!} + R_{n+1}, \qquad (3)$$

причем остаточный член R_{n+1} не превосходит

$$\frac{x^{2n+3}}{(2n+3)!}.$$

Обозначив k-й член суммы, стоящей в (3) справа, через u_k и положив $s_k = u_1 + u_2 + \ldots + u_k$, получим, что

$$u_{k+1} = -x^2 \frac{u_k}{a_k}, \quad \text{где} \quad a_k = 2k(2k+1), \tag{4}$$

и

$$s_{k+1} = s_k + u_{k+1} \qquad (k = 1, 2, \ldots), \quad s_1 = x. \tag{5}$$

Наконец, легко проверить, что

$$a_{k+1} = a_k + 8k + 6; \quad a_1 = 6 \qquad (k = 1, 2, \ldots). \tag{6}$$

Итак, мы приходим к следующей схеме вычислений: за первое приближение для $\sin x$ принимается $s_1 = x$. Далее, после того как получено k-е приближение s_k ($k = 1, 2, \ldots$), для нахождения следующего приближения s_{k+1} находятся сперва коэффициент a_{k+1} (по формуле (6)), затем величина u_{k+1} (по формуле (4)) и, наконец, s_{k+1} (по формуле (5)). Если же величина u_{k+1} оказывается меньше, чем заданное ε, то s_k принимается за значение $\sin x$, оно отпечатывается и машина переходит к вычислению $\sin x$ при новом значении x. Этот ход вычислений можно осуществить при помощи следующей программы:

№ ячейки	Операция	1-й адрес	2-й адрес	3-й адрес	Результат операции
0014	Ввод	0043	0015		Ввод массива
0015	Перевод $10 \to 2$	0044	0010	0044	Запись исходных данных к двоичной системе
0016	Сложение поразрядное	0050		0062	Перенос x в стандартную ячейку для u_k
0017	Сложение поразрядное	0062		0063	Перенос u_1 в стандартную ячейку для S
0020	Умножение	0062	0062	0064	x^2
0021	Вычитание		0064	0064	$-x^2$
0022	Умножение	0045	0060	0065	$8(k-1)$
0023	Сложение	0065	0046	0065	$8(k-1)+6$
0024	Сложение	0057	0065	0057	a_k
0025	Сложение	0060	0044	0060	k
0026	Деление	0062	0057	0065	$\frac{u_k}{a_k}$
0027	Умножение	0065	0064	0062	$-x^2 \frac{u_k}{a_k} = u_{k+1}$
0030	Сложение	0063	0062	0063	$S_{k+1} = S_k + u_{k+1}$
0031	Передача управления $1 \leqslant 1$	0047	0062	0022	Конец вычисл. $\sin x$

Продолжение

№ ячейки	Операция	1-й адрес	2-й адрес	3-й адрес	Результат операции
0032	Сложение поразрядное	0063		0050	Перенос $\sin x_i \to x_i$
0033	Сложение поразрядное			0060	Перенос $0 \to k$
0034	Сложение поразрядное			0057	Перенос $0 \to a_0$
0035	Сложение адресов	0016	0056	0016	Изменение 1-го адреса команды 0016
0036	Сложение адресов	0032	0055	0032	Изменение 3-го адреса команды 0032
0037	Сложение	0061	0044	0061	$i \to i + 1$
0040	Передача управления	0061	0054	0016	Конец табулирования
0041	Перевод $2 \to 10$	0050	0003	0050	
0042	Печать	0050	0003		
0043	Останов				
0044	1				
0045	8				
0046	6				
0047	ε				
0050	x_1				
0051	x_2				
0052	x_3				
0053	x_4				
0054	4				
0055	1 III адр.				
0056	1 I адр.				
0057	0 (a_k)				
0060	0 (k)				
0061	Раб. ячейка для i				
0062	Станд. ячейка для u_k				
0063	Станд. ячейка для S				
0064	Станд. ячейка для $-x^2$				
0065					

Аналогичные программы можно составить для вычисления других элементарных функций ($\cos x$, e^x, $\ln x$ и т. п.).

4. Блок-схемное программирование. Подпрограммы. При составлении программ для более или менее сложных задач удобно разбивать такие программы на отдельные части, так называемые блоки, отвечающие отдельным частным задачам. Это облегчает составление программы; кроме того, одни и те же блоки могут входить в качестве составных частей (стандартных программ) в программы различных задач. Рассмотрим такой элементарный пример. Требуется вычислить

приближенное значение интеграла

$$J=\int_a^b f(x)\,dx \tag{7}$$

с помощью метода прямоугольников (см. вып. 1, гл. 12). Вычисления здесь естественно разбить на две части (два блока):

1) вычисление входящих в формулу прямоугольников значений функции $f(x)$ в точках x_i;

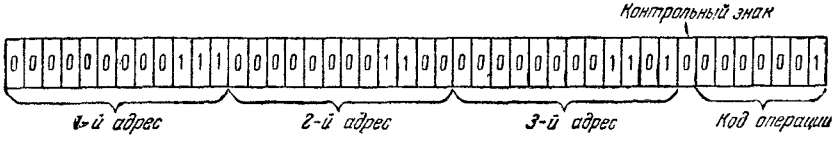
2) вычисление суммы

$$S=\frac{b-a}{n}\sum_{i=1}^{n} f(x_i), \tag{8}$$

представляющей собой приближенное значение интеграла (7). Программа для вычисления $f(x_i)$ зависит от вида функции $f(x)$. Напротив, программа для вычисления суммы (8) не связана с выбором функции $f(x)$.

Точность получающегося таким образом результата (т. е. абсолютная величина разности $J-S$) зависит, очевидно, от двух факторов: точности самой формулы прямоугольников *) и точности, с которой находятся значения функции f в точках x_i.

5. Коды команд. Операции над командами. Выше при составлении программ мы пользовались словесными обозначениями операций, например «сложить», «умножить» и т. д. Но для ввода команд в машину эти словесные обозначения необходимо заменить численными, записанными по двоичной системе, т. е. кодами этих команд. Поэтому те ячейки памяти машины, в которые введены команды, заполняются так же, как и при вводе числовых данных, некоторой последовательностью нулей и единиц. Из общего числа имеющихся в каждой ячейке рязрядов несколько разрядов отводятся для записи кода команды, а остальные — для записи адресов. Например, в машине «Стрела» в каждой ячейке имеется 43 разряда. Из них по 12 разрядов отводится на запись каждого из адресов, а шесть — на код команды (один разряд отводится для контрольного знака). Таким образом, команда «сложить числа, находящиеся в ячейках 7 и 12, и результат записать в ячейку 13» в коде машины «Стрела» запишется так:



*) Об оценке точности различных формул для приближенного вычисления интегралов см. вып. 1, гл. 2, § 2.

(«0000001» по коду «Стрелы» означает сложение). То обстоятельство, что команды, введенные в машину, по виду ничем не отличаются от числовых данных, не вызывает каких-либо неудобств. Напротив, это дает возможность обращаться с командами, как с обычными числами, например «складывая» их *), а это в свою очередь позволяет сильно упростить программирование. Рассмотрим для иллюстрации сказанного следующий простой пример. Предположим, что нам нужно составить программу для суммирования тысячи чисел. Можно, конечно, ввести их в память машины, например в ячейки с $(n+1)$-й по $(n+1000)$-ю, а затем составить программу следующим образом: 1-я команда:

сложить	$n+1$	$n+2$	$n+2$

2-я команда:

сложить	$n+2$	$n+3$	$n+3$

999-я команда:

сложить	$n+999$	$n+1000$	$n+1000$

Можно, однако, решить эту задачу более экономно следующим образом. Запишем снова те числа, которые нужно сложить, в ячейках памяти от $(n+1)$-й до $(n+1000)$-й. После этого в ячейку, скажем, с номером $n+1001$ запишем:

	0001	0001	0001

Пусть теперь в ячейке с номером $m+1$ записана команда:

сложить	$n+1$	$n+2$	$n+2$

, дающая сложение двух первых чисел. Далее, в ячейку с номером $m+2$ запишем такую команду:

поадресно сложить	$m+1$	$n+1001$	$m+1$

В ячейку с номером $m+3$ поместим команду перехода к ячейке с номером $m+1$, которая теперь уже будет содержать команду:

сложить	$n+2$	$n+3$	$n+3$

,

*) Следует иметь в виду, что при операциях над командами применяются специальные операции сложения: сложение кодов операций, поадресное и поразрядное сложение.

по которой к сумме двух первых чисел будет прибавлено третье и результат будет записан в ячейку с номером $n+3$. Ясно, что цикл из таких трех команд обеспечит сложение всех чисел, записанных в ячейках $n+1, \ldots, n+1000$. Остается еще обеспечить печать ответа и останов машины по окончании работы.

Таким образом, применение операции сложения команд позволило нам заменить длинную цепь однотипных команд небольшим числом операций.

6. Об автоматизации программирования. Несмотря на наличие таких приемов, как использование стандартных подпрограмм, и другие усовершенствования, облегчающие и упрощающие программирование, составление программы часто бывает весьма трудоемким процессом, требующим во много раз больше времени, чем сам счет на УЦВМ. В первую очередь это относится к большим современным быстродействующим машинам. Поэтому сейчас все большее значение приобретают различные методы *автоматизации программирования*.

Не имея возможности останавливаться здесь на описании этих методов сколько-нибудь подробно, укажем лишь их основную идею. Она состоит в том, чтобы передать функции перевода словесного описания тех или иных вычислений в последовательность команд, записанных в коде машины, самой вычислительной машине. Иначе говоря, математик пишет ход решения той или иной задачи в виде словесного описания, пользуясь некоторым заранее фиксированным набором понятий и терминов. Далее такое словесное описание вводится в УЦВМ (при этом, конечно, каждая буква текста изображается определенной комбинацией нулей и единиц, так же как это делается, например, в телеграфии); после этого сама вычислительная машина с помощью некоторой универсальной программы-транслятора переводит это словесное описание в программу, записанную в коде данной машины. Для того чтобы такой переход от словесного описания к программе мог быть автоматизирован, необходимо, чтобы это описание было составлено с соблюдением определенных формальных правил и с четко ограниченным запасом слов. Существует несколько таких стандартизованных формальных «языков», используемых для автоматического программирования. Наиболее распространенные из них — это алгоритмический язык АЛГОЛ и язык ФОРТРАН *). Каждый такой язык может быть использован независимо от того, на какой машине в дальнейшем будет проводиться счет. Напротив, программа-транслятор, преобразующая словесную запись в машинные команды, зависит от выбора языка и

*) АЛГОЛ — сокращение английских слов «algoritmic language» (алгоритмический язык), а ФОРТРАН — комбинация слогов слов «formula translating» (перевод на язык формул).

от типа машины (но не зависит от той конкретной задачи, которая должна быть сосчитана).

Введение таких формальных языков и программ-трансляторов позволяет значительно сократить трудоемкую и кропотливую работу по программированию.

§ 4. Некоторые вопросы организации работы на УЦВМ

1. Условия, определяющие эффективность применения УЦВМ. Как уже говорилось выше, для решения на УЦВМ той или иной задачи должна быть составлена отвечающая этой задаче программа, т. е. указана в соответствующем коде последовательность тех элементарных операций, к которой сводится задача. Если бы число отдельных команд в программе было таким же, как и число тех операций, которые необходимы для решения данной задачи, то применение УЦВМ было бы лишено всякого смысла, так как при этом составление программы занимало бы не меньше времени, чем выполнение всех расчетов вручную. Однако при решении всякой задачи отдельные циклы операций приходится повторять несколько, а иногда и очень много раз (мы видели это уже на таких простых примерах, как программа извлечения квадратного корня; в еще большей степени это относится к более сложным задачам). Поэтому число команд в программе (разумно составленной) во много раз меньше числа операций, выполняемых по этой программе машиной. Особенно эффективно применение УЦВМ в тех задачах, где приходится многократно повторять вычисления с различными данными, но по одной и той же схеме. С другой стороны, существуют и такие задачи, в которых применение УЦВМ оказывается неэффективным из-за того, что они при сравнительно небольшом объеме счета требуют для решения их на УЦВМ составления длинной и громоздкой программы.

Умение правильно решить вопрос о целесообразности применения УЦВМ для той или иной конкретной задачи — первое условие рационального использования вычислительной техники.

2. Основные этапы решения задачи с применением УЦВМ. Решение той или иной прикладной задачи с помощью УЦВМ слагается из следующих основных этапов.

1) *Математическая формулировка задачи.* Всякая задача, предназначенная для численного ее решения на УЦВМ, должна быть прежде всего четко сформулирована именно как математическая задача. Иначе говоря, та физическая, техническая или какая-либо другая проблема, которая подлежит решению, должна быть представлена как задача о решении каких-либо уравнений, вычислении интегралов и т. п. Следует иметь в виду, что этот этап работы, требующий обычно совместной работы физиков или инженеров,