

М. З. Грузман

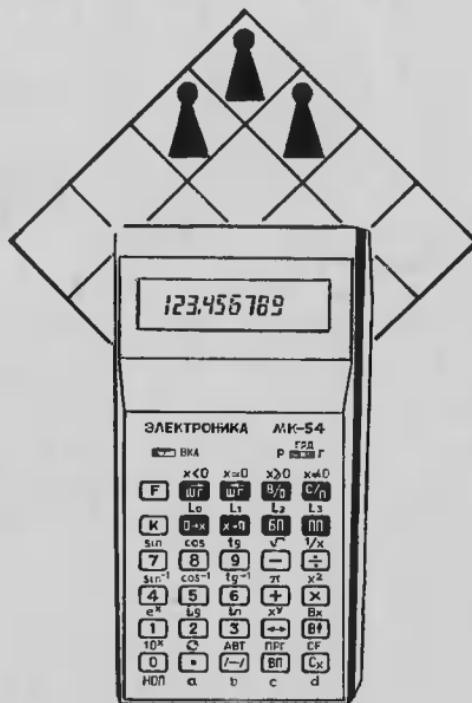
ЛОГИЧЕСКИЕ ИГРЫ С КАЛЬКУЛЯТОРОМ

М. З. Грузман

ЛОГИЧЕСКИЕ ИГРЫ С КАЛЬКУЛЯТОРОМ

КНИГА ДЛЯ УЧАЩИХСЯ 8—10 КЛАССОВ СРЕДНЕЙ ШКОЛЫ

Под редакцией И. Ф. Тесленко



Рецензенты:

заслуженный учитель РСФСР Т. Е. Доцевич;
кандидат технических наук М. А. Тагиров

Грузман М. З.

Г90 Логические игры с калькулятором: Кн. для учащихся 8—10 кл. сред. шк./Под ред. И. Ф. Тесленко.— М.: Просвещение, 1989.— 160 с.: ил.

ISBN 5-09-001594-5

Книга в форме эвристических бесед и активного диалога знакомит учащихся с вычислительными возможностями программируемого калькулятора и основами программирования на нем. Наряду с теоретическим материалом пособие содержит большое количество интересных задач для самостоятельной работы.

Г 4306020000—256 КБ—25—48—1988
103(03)—89

ISBN 5-09-001594-5

ББК 32.973.01

© Издательство «Просвещение», 1989

ПРЕДИСЛОВИЕ

Во времена первых ЭВМ на них в основном работали профессиональные программисты. С развитием микропроцессорной технологии появились общедоступные компьютеры, которые в настоящее время используются учеными и студентами, инженерами и рабочими, экономистами и администраторами, учителями и школьниками. Сфера применения компьютеров чрезвычайно расширилась: от расчета траектории космического корабля до расчета курсового проекта студентом, от управления сложным технологическим процессом до управления бытовыми приборами, от автоматических обучающих систем до выполнения школьниками домашних заданий, от решения сложных научных задач до игр и развлечений. Программирование становится второй грамотностью.

Наиболее простые и доступные в настоящее время компьютеры — это программируемые калькуляторы (ПК). Они в значительной степени реализуют мечту человека об ЭВМ в кармане. Программируемые калькуляторы, о которых пойдет речь в книге, занимают как бы промежуточное место между инженерными калькуляторами и персональными компьютерами: с одной стороны, это наиболее совершенные калькуляторы, с другой — их с полным правом можно отнести к простейшим микроЭВМ. Будем рассматривать калькулятор «Электроника Б3-34» (рис. 1). Калькуляторы «Электроника МК-54» и «Электроника МК-56» имеют ту же систему команд, изменены лишь обозначения некоторых клавиш (см. [55]¹). Все приведенные программы легко перевести на язык этих калькуляторов.

Вы можете научиться программировать, используя в качестве своего помощника ПК. Изучив первую главу, вы сумеете описывать разнообразные алгоритмы, овладеете приемами перевода таких описаний на язык калькулятора. Особое внимание обратите на алгоритмический язык (псевдокод), близкий к алгоритмическому языку школьного курса информатики. В данной книге этот язык рассматривается как язык проектирования (разра-

¹ Здесь и далее номер в квадратных скобках означает ссылку на список литературы, приведенный в конце книги.

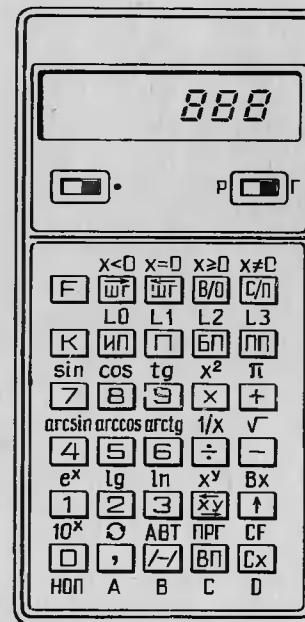
10 УРОКОВ ПО ПРОГРАММИРОВАНИЮ НА КАЛЬКУЛЯТОРЕ

ботки) программ. Параллельно показывается, как осуществляется перевод описания алгоритма на язык ПК. Изучив первую главу, вы сможете, например, переводить на язык ПК алгоритмы из учебника по информатике для 9 класса. Наконец, урок структурного программирования посвящен знакомству с современным подходом к проектированию программы: методом пошаговой детализации (последовательным построением алгоритма).

Вторая глава («Игра и калькулятор») поможет вам развить навыки программирования, которые вы приобретете после изучения первой главы. Некоторые предлагаемые задачи непросты, и для достижения цели следует проявить упорство, настойчивость и изобретательность. Надеемся тем не менее, что вас привлечет рассматриваемый материал: не может быть неинтересной задача составления программы игры человека с машиной!

Желаем успеха в овладении компьютерной грамотностью!

Рис. 1



§ 1.1. УРОК 1. ОПЕРАНДЫ И ОПЕРАЦИИ

Что умеет калькулятор?

Прежде всего калькулятор умеет выполнять операции над числами.

Для набора чисел используются клавиши

.

Для присвоения числу знака минус используется клавиша , а клавиша стирает набранное число.

Нажмите, например, последовательно на клавиши

, и на индикаторе загорится число —265. Сотрите это число с помощью клавиши .

Калькулятор оперирует не только с целыми числами, но и с дробными. Для этого используется клавиша .

Нажмите на клавиши , и на индикаторе загорится число —12.15.

Калькулятор позволяет работать также с числами вида $a=m \cdot 10^p$, где m — десятичное число с десятичной точкой или без нее, называемое мантиссой, p — двухразрядное целое число, называемое порядком.

Пусть, например, требуется набрать число $-2.36 \cdot 10^{-2}$.

Нажмите последовательно на клавиши:

ВП .

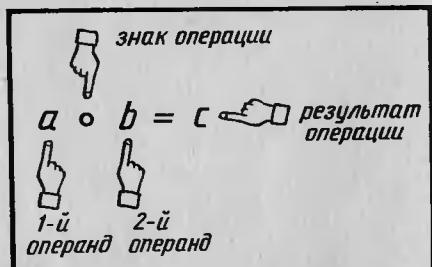
На индикаторе загорится: —2.36—02. Обратите внимание: перед вводом порядка нажимается клавиша — ввод порядка.

Поупражняйтесь в наборе чисел.

Кто наберет наибольшее число? Кто — наименьшее? Наименьшее положительное?

Над числами калькулятор умеет выполнять двухместные и одноместные операции. Если любой паре чисел (a, b) из некоторого множества пар чисел поставлено в соответствие некоторое число c , то будем говорить, что задана двухместная (бинарная) операция. При этом a будем называть первым операндом, b — вторым операндом, а c — результатом операции (рис. 2).

Рис. 2



Рассмотрим примеры.

1. Операция сложения. Каждой паре чисел ставится в соответствие число — их сумма. Здесь операнды — слагаемые. Результат операции — сумма этих слагаемых.

2. Операция деления. Всякой паре чисел (a, b) , такой, что $b \neq 0$, поставлено в соответствие число c — частное от деления a на b . Здесь первый операнд — делимое, второй — делитель. Результат операции — частное.

На калькуляторе можно выполнять такие двухместные операции:

1. Сложение (клавиша $[+]$).

2. Вычитание (клавиша $[-]$).

3. Умножение (клавиша $[x]$).

4. Деление (клавиша $[÷]$).

5. Возведение в степень (клавиши $F [x^y]$).

Отметим, что обозначение x^y расположено вне клавиши (см. рис. 1). Однако в этом и аналогичных случаях мы соответствующее обозначение будем помещать внутри квадратика.

Изучим операционные регистры Y и X. Они так названы потому, что над их содержимым выполняются операции.

Для выполнения бинарной операции

1-й операнд — должен быть занесен в RGY,

2-й операнд — в RGX.

Занесение в RGX осуществляется при наборе чисел на клавиатуре. Содержимое этого регистра загорается на индикаторе. Занесение в RGY осуществляется с помощью клавиши¹

$\boxed{\uparrow}$. При нажатии на эту клавишу содержимое RGX заносится в RGY. После того как операнды занесены в операционные регистры, нажмите на клавишу с обозначением знака операции. Результат загорается на индикаторе (заносится в RGX). Итак, на калькуляторе выполняются такие команды:

$RGY + RGX \rightarrow RGX$ (сложение),

$RGY - RGX \rightarrow RGX$ (вычитание),

$RGY \times RGX \rightarrow RGX$ (умножение),

$RGY \div RGX \rightarrow RGX$ (деление),

$(RGX)^{RGY} \rightarrow RGX$ (возведение в степень), а также пересылки из RGX в RGY: $RGX \rightarrow RGY$.

Пример. Выполнить деление: 1024:8.

Решение.

| | | | | | |
|--------------------|----------------|-------------|-------------|--------------------------------|------------------------|
| $\boxed{1}$ | $\boxed{0}$ | $\boxed{2}$ | $\boxed{4}$ | $1024 \rightarrow RGX$ | $1024 \rightarrow RGY$ |
| $\boxed{\uparrow}$ | | | | $RGX \rightarrow RGY$ | |
| | $\boxed{8}$ | | | $8 \rightarrow RGX$ | |
| | $\boxed{\div}$ | | | $RGY \div RGX \rightarrow RGX$ | |

Результат — 128 — загорается на индикаторе. Поупражняйтесь в выполнении двухместных операций на калькуляторе.

Если всякому числу a из некоторого множества чисел поставлено в соответствие некоторое число b , то будем говорить, что задана одноместная (унарная) операция. При этом a будем называть операндом, b — результатом операции (рис. 3).

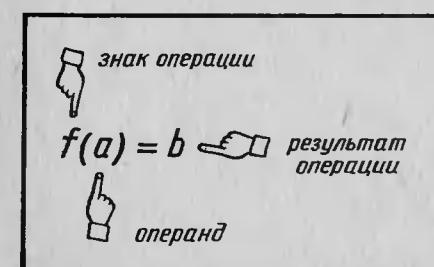


Рис. 3

¹ Для МК-54 и МК-56 используется клавиша $\boxed{B \uparrow}$.

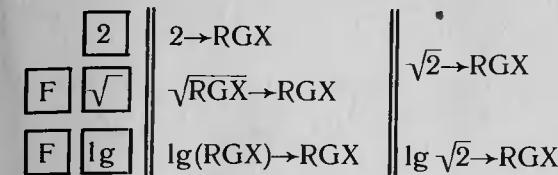
Рассмотрим примеры:

- Нахождение квадратного корня. Каждому неотрицательному числу — операнду ставится в соответствие арифметическое значение корня из этого числа — результат операции.
- Нахождение синуса числа. Всякому числу — операнду ставится в соответствие синус этого числа — результат операции. Калькулятор выполняет такие одноместные операции:
 - Извлечение корня (клавиши $F \sqrt{\square}$).
 - Возведение в квадрат (клавиши $F x^2$).
 - Нахождение обратной величины числа (клавиши $F 1/x$).
 - Нахождение синуса (клавиши $F \sin$).
 - Нахождение косинуса (клавиши $F \cos$).
 - Нахождение тангенса (клавиши $F \operatorname{tg}$).
 - Нахождение арксинуса (клавиши $F \arcsin$).
 - Нахождение арккосинуса (клавиши $F \arccos$).
 - Нахождение арктангенса (клавиши $F \operatorname{arctg}$).
 - Нахождение 10^x (клавиши $F 10^x$).
 - Нахождение натурального логарифма (клавиши $F \ln$).
 - Нахождение e^x (клавиши $F e^x$).
 - Нахождение десятичного логарифма (клавиши $F \lg$).
 - Изменение знака числа (клавиша $/-\square$).

Для выполнения унарной операции на калькуляторе operand должен быть занесен в RGX. Нажимая на клавишу F , а затем на клавишу, над которой расположен знак одноместной операции, получим в RGX результат этой операции. Итак, на калькуляторе выполняются команды вида $f(\text{RGX}) \rightarrow \text{RGX}$.

Рассмотрим такой пример. Вычислить $\lg \sqrt{2}$.

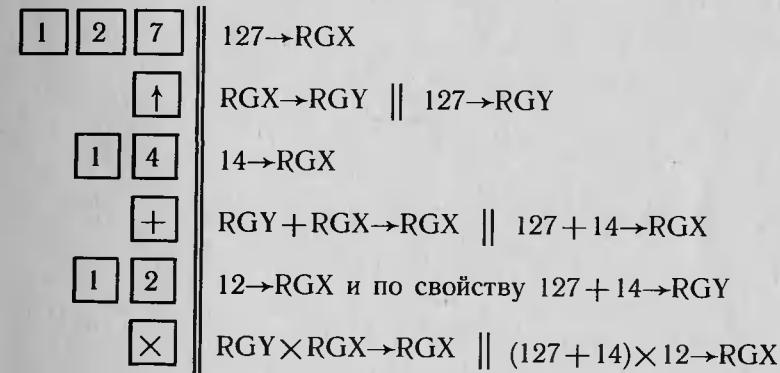
Решение.



Результат 1. 5051497—01 загорается на индикаторе. Поупражняйтесь в выполнении одноместных операций.
Сформулируем следующее свойство калькулятора.

Если на калькуляторе выполнена двухместная или одноместная операция, а затем набрано некоторое число, то результат операции при этом автоматически засыпается в RGY.

Пример. Вычислить $(127 + 14) \cdot 12$.



Результат 1692 загорается на индикаторе.

Познакомимся еще с одним регистром калькулятора. Это регистр предыдущего результата RGX_1 . Чтобы уяснить, как используется этот регистр, сформулируем еще одно свойство калькулятора.

При выполнении одноместной или двухместной операции результат заносится в RGX , а прежнее содержимое RGX заносится в регистр предыдущего результата RGX_1 .

Для того чтобы прочитать содержимое RGX_1 , следует нажать на клавиши $F B_x$. При этом содержимое RGX_1 заносится в RGX и загорается на индикаторе, а прежнее содержимое RGX заносится в RGY .

Пример. Вычислить $1,25^3$.

Решение.

| | | | | |
|---|-------|---|---|--|
| 1 | , | 2 | 5 | 1,25 → RGX |
| F | x^2 | | | $[RGX \rightarrow RGX_1 1,25 \rightarrow RGX_1]$ $(RGX)^2 \rightarrow RGX 1,25^2 \rightarrow RGX$ |
| F | Bx | | | $RGX \rightarrow RGY$ и $RGX_1 \rightarrow RGX 1,25^2 \rightarrow RGY$ и $1,25 \rightarrow RGX$ |
| X | | | | $RGY \times RGX \rightarrow RGX 1,25^2 \cdot 1,25 \rightarrow RGX$ |

Результат 1.953125 загорается на индикаторе.

Познакомимся с командой обмена между содержимым RGX и содержимым RGY. Обозначать эту команду будем так: $RGX \leftrightarrow RGY$.

При нажатии на клавишу¹ содержимое регистра RGX заносится в RGY, а содержимое RGY — в RGX.

Пример. Вычислить $\frac{1,17^2}{2,3+1,5}$.

Решение.

| | | | | |
|-------------------|-------|---|---|---|
| 2 | , | 3 | 2,3 → RGX | |
| | ↑ | | $RGX \rightarrow RGY 2,3 \rightarrow RGY$ | |
| 1 | , | 5 | 1,5 → RGX | |
| | + | | $RGY + RGX \rightarrow RGX 2,3 + 1,5 \rightarrow RGX$ | |
| 1 | , | 1 | 7 | $1,17 \rightarrow RGX$ и $RGX \rightarrow RGY 2,3 + 1,5 \rightarrow RGY$ |
| F | x^2 | | $(RGX)^2 \rightarrow RGX 1,17^2 \rightarrow RGX$ | |
| \leftrightarrow | | | $RGX \leftrightarrow RGY 1,17^2 \rightarrow RGY$ и $2,3 + 1,5 \rightarrow RGX$ | |
| \div | | | $RGY \div RGX \rightarrow RGX 1,17^2 : (2,3 + 1,5) \rightarrow RGX$ | |

Результат 3.6023684—01 загорается на индикаторе.

¹ Для МК-54 и МК-56 клавишу В дальнейшем будем обозначать эту клавишу

Замечание. На калькуляторе «Электроника Б3-34» можно выполнять операцию извлечения числа π . Нажмите на клавиши π , и в RGX занесется число 3.1415926.

Упражнения

1. Вычислите значения выражений:

a) $(1,119 - 10,53 + 3,14) \cdot 12,5$; г) $\ln \sin^2 2$;

б) $\frac{-1,3 \cdot 10^{-2}}{113,6 - 17,82}$; д) $\sqrt{1,2^2 + 1,3^2 + 1,4^2}$;

в) $1,12^7$; е) $\frac{2e^{1/2}}{11,4 + 20,97}$.

2. Выясните, принадлежат ли значения

а) 0,303; б) $-3,302$; в) $-3,303$

области определения функции $y = \lg(x^2 + 3x - 1)$.

3. Проведя вычисления на калькуляторе, сформулируйте гипотезы о значениях пределов:

a) $\lim_{x \rightarrow 0} \frac{\sin x}{x}$; б) $\lim_{x \rightarrow \infty} \ln\left(1 + \frac{1}{x}\right)^x$; в) $\lim_{x \rightarrow 0} \frac{\ln x}{x}$.

4. Какая из функций $y = \frac{1}{2} \operatorname{arctg} x$ или $y = \operatorname{arctg} 2x$ является обратной к функции $y = \operatorname{tg} 2x$?

Указание. Проверьте свое предположение на нескольких значениях. Такая проверка, конечно, не доказательство, однако приобретается дополнительная уверенность.

5. Каков период функции $y = \sin \pi x$?

Указание. Проверьте себя, используя несколько значений x . Если ваше предположение подтверждается, ищите доказательство.

6. Постройте по точкам график функции $y = \frac{x+1}{x^2+1}$.

§ 1.2. УРОК 2. ОПЕРАТОР ПРИСВАИВАНИЯ И ЯЧЕЙКИ ПАМЯТИ КАЛЬКУЛЯТОРА

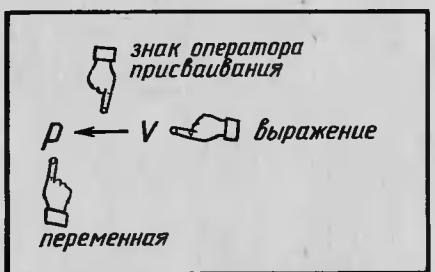
При изучении программирования на том или ином алгоритмическом языке одним из первых операторов рассматривают так называемый оператор присваивания. Для нас это понятие окажется весьма полезным. В дальнейшем будем пользоваться также понятиями переменной и выражения. Запись переменных и выражений на алгоритмических языках отличается от математической, мы же пока будем пользоваться обычными принятыми в математике обозначениями. Переменные будем обозначать большими и малыми латинскими, греческими буквами, возможно с индексами. Например, α , ω , S , x , Σ , φ , a_1 , b_n .

Выражения конструируются из чисел, переменных, функций, знаков операций и скобок. Например,

$$x^2; 2 \sin \alpha + 5,3; \frac{\ln x}{x+y}; \\ 3(a_n + b_n); 121,6.$$

Введем, наконец, обещанный оператор присваивания. Рисунок 4 поясняет принцип его действия. «Работает» этот оператор так:

Рис. 4



1. Вычисляется значение выражения V .

2. Полученное значение присваивается переменной P , при этом старое значение переменной P теряется.

Разумеется, к моменту выполнения оператора присваивания всем переменным, входящим в выражение V , должны быть присвоены числовые значения.

Рассмотрим примеры.

Пример 1. Определить значение переменной y после выполнения последовательности операторов присваивания:

$$\begin{aligned}y &\leftarrow 1 \\x &\leftarrow 2 \\y &\leftarrow y + x \\y &\leftarrow y + 1\end{aligned}$$

Решение. Первый оператор присваивает переменной y значение 1, а второй оператор — переменной x значение 2. Третий оператор выполняется так: а) вычисляется значение выражения $x+y$ при $x=2$ и $y=1$, б) полученное значение 3 присваивается переменной y , прежнее значение 1 этой переменной пропадает. Наконец, четвертый оператор увеличивает значение переменной y на 1.

Итак, после выполнения рассматриваемой последовательности операторов переменной y будет присвоено значение 4.

Пример 2. Пусть переменной y присвоено значение 2. Определить, каким будет значение этой переменной после трехкратного выполнения оператора $y \leftarrow y^2$.

Решение. Новое значение переменной y равно квадрату старого. Таким образом, после трехкратного применения рас-

матриваемого оператора переменной y будет присвоено значение $((2^2)^2)^2 = 256$.

Пример 3 (Н. Вирт). Пусть переменной a присвоено значение -1 , а переменной b — значение 0 . Какими будут значения этих переменных после 1000-кратного выполнения последовательности операторов:

$$\begin{aligned}a &\leftarrow a + 2; \\b &\leftarrow b + a^2\end{aligned}$$

Решение. Первый оператор увеличивает значение a на 2, а второй увеличивает значение переменной b на новое значение переменной a . Таким образом, после однократного выполнения этих двух операторов переменной a будет присвоено значение 1 и переменной b — также значение 1.

Если эти два оператора будут выполнены еще раз, то переменная a получит значение 3, а переменная b — значение 4 и т. д. Заполним таблицу (табл. 1).

Таблица 1

| n | 1 | 2 | 3 | 4 | 5 | 6 | ... |
|-----|---|---|---|----|----|----|-----|
| a | 1 | 3 | 5 | 7 | 9 | 11 | ... |
| b | 1 | 4 | 9 | 16 | 25 | 36 | ... |

Замечаем, что переменная a пробегает последовательность нечетных чисел, а переменная b — последовательность квадратов натуральных чисел. Сформулируем гипотезу:

на n -м шаге переменной a будет присвоено значение $2n-1$, а переменной b — значение n^2 .

Итак, после 1000-кратного повторения переменной a будет присвоено значение 1999, а переменной b — 1 000 000.

Однако ответ можно было получить иначе, если соотношение $(n+1)^2 = n^2 + 2n + 1$ интерпретировать так: «Новый квадрат равен старому, сложенному с очередным нечетным числом». Тогда становится ясным, что оператор $a \leftarrow a + 2$ порождает последовательные нечетные числа, а выполняемый вслед за ним оператор $b \leftarrow b + a^2$ порождает последовательные квадраты.

Работая с оператором присваивания, мы, по сути дела, пользовались принципом магнитофонной ленты для переменной:

переменную можно сравнить с записью на магнитофоне.
Действительно:

- При присваивании переменной нового значения старое ее значение пропадает. Аналогично при записи на магнитофонную ленту старое ее содержимое стирается.
- При использовании значения переменной, т. е. при под-

становке ее значения в вычисляемое арифметическое выражение, значение переменной сохраняется. Аналогично при прослушивании магнитофонной записи эта запись сохраняется.

Теперь познакомимся с тем, как реализуется оператор присваивания на калькуляторе. Мы уже познакомились с операционными регистрами RGX и RGY, а также с регистром предыдущего результата RGX₁. Кроме этих регистров, в калькуляторе имеются 14 адресуемых регистров — ячеек памяти: RG0, RG1, RG2, RG3, RG4, RG5, RG6, RG7, RG8, RG9, RGA, RGB, RGC, RGD, в которых можно хранить начальные данные, а также промежуточные и конечные результаты.

Запись в ячейку памяти (адресуемую ячейку) осуществляется нажатием клавиш: **[П] [n]**¹. Здесь *n* — адрес ячейки памяти. Возможные значения *n* таковы: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D. При нажатии на указанные клавиши содержимое RGX заносится в ячейку с адресом *n*.

Чтение числа, находящегося в ячейке с адресом *n*, осуществляется нажатием клавиш **[ИП] [n]**². При этом содержимое ячейки с адресом *n* заносится в RGX. Итак, мы ввели две новые команды:

$RGX \rightarrow RGn$ (команда записи);
 $RGn \rightarrow RGX$ (команда чтения).

Поработайте немного с ячейками памяти. Наберите, например, на клавиатуре число 5. Нажмите на клавиши **[П] [1]**. При этом число 5 занесется в RG1. Нажмите на клавишу **[Cx]**, а затем на клавиши **[ИП] [1]**. Число 5 опять загорится на индикаторе.

Сформулируем принцип магнитофонной ленты для ячейки памяти:

при записи числа в ячейку памяти прежнее содержимое этой ячейки стирается. При чтении содержимое ячейки памяти сохраняется.

Проверьте, как выполняется этот принцип на калькуляторе!

¹ Для МК-54 и МК-56 роль **[П]** играет **[x→П]**.

² Для МК-54 и МК-56 роль **[ИП]** играет **[П→x]**.

Теперь несложно установить аналогию:

переменная — ячейка памяти, присваивание — запись в ячейку памяти.

Рассмотрим, как осуществляется трансляция (перевод) оператора присваивания на язык калькулятора. Пусть, например, требуется реализовать на калькуляторе оператор присваивания: $a \leftarrow a + b$.

Прежде всего для значения переменной *a* выделим ячейку RGA, а для значения переменной *b* — ячейку RGB. Тогда перевод на язык калькулятора будет выглядеть так:

| | | |
|-------------|------------|--|
| [ИП] | [A] | $RGA \rightarrow RGX$ |
| [ИП] | [B] | $RGB \rightarrow RGX$, при этом $RGX \rightarrow RGY$ |
| [+] | | $RGY + RGX \rightarrow RGX$ |
| [П] | [A] | $RGX \rightarrow RGA$ |

Отметим, что при чтении содержимого ячейки памяти прежнее содержимое RGX заносится в RGY.

Если теперь в ячейки RGA и RGB занести некоторые числа, а затем нажать на указанные клавиши, то содержимое этих ячеек изменится так, как предписывает рассматриваемый оператор. Проверьте!

Рассмотрим еще один пример.

Пример. Составить программу для вычисления выражения

$$y = \frac{a+b}{c+d}.$$

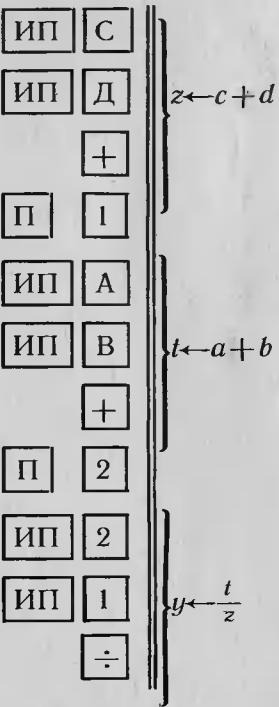
Решение. Внимательный читатель заметил, конечно, что здесь впервые мы употребили термин «программа». Под программой мы пока понимаем последовательность команд, которые нужно выполнить для вычисления выражения. Составить программу — значит указать последовательность клавиш, нажимая на которые в указанном порядке получаем значение выражения.

Запишем последовательность операторов присваивания:

$$\begin{aligned} z &\leftarrow c + d, \\ t &\leftarrow a + b, \\ y &\leftarrow \frac{t}{z}. \end{aligned}$$

Программа для калькулятора выглядит так:

| Распределение памяти | |
|----------------------|-----|
| <i>a</i> | RGA |
| <i>b</i> | RGB |
| <i>c</i> | RGC |
| <i>d</i> | RGD |
| <i>z</i> | RG1 |
| <i>t</i> | RG2 |
| <i>y</i> | RGX |



Конечно, эта программа составлена нерационально. Как составлять более рациональные программы для вычисления выражений, мы рассмотрим позже, когда познакомимся с так называемой польской записью.

Выполнив предварительно занесение исходных данных и нажав на указанные клавиши, получим значение *y* на индикаторе калькулятора.

Отметим одно важное обстоятельство. Если ранее мы находили значения числовых выражений, то теперь составляем программы для вычисления выражений с переменными. По таким программам можно вычислять значения выражений для разных значений переменных. Такую возможность мы получили благодаря адресуемым ячейкам памяти.

Сформулируем **принцип адресности**, в важности которого у вас будет возможность еще не раз убедиться.

В программе можно указывать не только сами числа, над которыми производится та или иная операция, но и адреса ячеек памяти, в которых эти числа хранятся.

Коль скоро в программе указаны адреса ячеек памяти, то, занося в эти ячейки различные числа, можно, пользуясь одной программой, вычислять различные числовые выражения.

Мы получили возможность говорить о программе нахождения площади треугольника по трем сторонам, о программе решения квадратного уравнения, о программе вычисления площади пирамиды, о программе...

Упражнения

1. Какие значения будут присвоены переменным после выполнения последовательности операторов присваивания:

| | | |
|---------------------------|-----------------------------|--------------------------------|
| a) $b \leftarrow 1,$ | б) $y \leftarrow 4,$ | в) $z \leftarrow 1,$ |
| $a \leftarrow b,$ | $y \leftarrow \sqrt{y},$ | $z \leftarrow \lg z + 2z + 1?$ |
| $c \leftarrow a^2 + b^2;$ | $y \leftarrow \frac{1}{y};$ | |

2. Пусть переменным *a*, *b*, *c*, *d* соответственно присвоены значения —1, 0, 0, 0. Какие значения получат эти переменные после 1983-кратного выполнения последовательности операторов присваивания:

$$\begin{aligned}d &\leftarrow d + 3b + 3c + 1, \\c &\leftarrow c + 1, \\a &\leftarrow a + 2, \\b &\leftarrow b + a?\end{aligned}$$

3. Переведите на язык калькулятора такие последовательности операторов присваивания:

| | | |
|---------------------------------|-----------------------|----------------------|
| а) $y \leftarrow \lg \sqrt{y};$ | б) $a \leftarrow -1,$ | в) $a \leftarrow b,$ |
| $b \leftarrow 0,$ | $b \leftarrow -c,$ | $c \leftarrow a.$ |
| $a \leftarrow a + 2,$ | | |
| $b \leftarrow b + a;$ | | |

§ 1.3. УРОК 3. «ЭЛЕКТРОНИКА БЗ-34» — МИКРОЭВМ

Учитель предложил ученику решить квадратное уравнение $x^2 - 5x + 6 = 0$.

Ученик знал, как решаются квадратные уравнения вида $x^2 + px + q = 0$, и предложил начать с вычисления дискриминанта. Пользуясь формулой $D = \left(\frac{p}{2}\right)^2 - q$, он подсчитал, что $D = 0,25$. Далее, запомнив или записав найденное значение дискриминанта, ученик сделал вывод: «Так как дискриминант больше нуля, уравнение имеет два различных действительных корня». Далее по формуле $x_{1,2} = -\frac{p}{2} \pm \sqrt{D}$ были найдены эти корни: $x_1 = 2$ и $x_2 = 3$. Полученный ответ ученик сообщил учителю.

Попробуем проанализировать процесс решения этой задачи учеником. Прежде всего ученик должен запомнить или записать

условие задачи. Далее он пользуется алгоритмом решения квадратного уравнения, который хранится в его памяти. (С понятием алгоритма мы подробнее познакомимся чуть позже, здесь же под алгоритмом решения задачи будем понимать последовательность указаний, которые требуется выполнить, чтобы ее решить.) Затем ученик должен уметь выполнять различные операции: сложение, умножение, деление, вычитание, извлечение квадратного корня. Кроме выполнения этих операций, ученику нужно также проверить условие $D > 0$. В рассматриваемом случае это условие выполняется. Ученик извлек из памяти сведения о том, как следует поступить в этом случае, и, выполнив предписанные действия, закончил решение уравнения. Результаты решения (корни уравнения x_1 и x_2) были сообщены учителю.

Итак, процессу решения задачи учеником присуще:

1. Хранение информации. В рассматриваемом примере ученик запоминал или записывал на листе бумаги условие задачи, а также промежуточные и окончательные результаты. Кроме того, в памяти ученика хранилась информация о способе решения квадратного уравнения, т. е. алгоритм решения задачи.

2. Обработка информации и управление процессом обработки информации. Ученик всякий раз извлекал из своей памяти информацию о том, что следует сделать согласно алгоритму, а также необходимые для этого данные (числа). Производил предписанные операции и запоминал то, что потребуется для дальнейшего.

3. Ввод и вывод информации. Учитель сообщил ученику условие задачи, т. е. в память ученика был осуществлен ввод данных p и q , которые указывают, какое именно квадратное уравнение нужно решать. Кроме того, ученика ранее обучили алгоритму решения произвольного квадратного уравнения, т. е. был осуществлен ввод в память ученика алгоритма решения задачи. После решения уравнения значения корней x_1 и x_2 были сообщены учителю, тем самым был осуществлен вывод результатов.

Если мы хотим поручить машине решение задач, то разумно предположить, что в такой машине должны происходить процессы, аналогичные тем, которые происходят при решении задачи учеником, т. е. вычислительная машина должна иметь перечисленные ниже устройства.

1. Запоминающее устройство. Это устройство предназначено для хранения исходных и промежуточных данных, а также результатов. Кроме того, в память вычислительной машины заносится алгоритм решения задачи. Для этого он должен быть представлен в специальном виде, «понятном» машине, т. е. в виде программы. Сформулируем принцип хранения программы:

Программа заносится в память машины и хранится в ней в кодированном виде.

Запоминающее устройство ЭВМ представляет собой последовательность ячеек памяти, каждой из которых присвоен свой

номер, называемый адресом ячейки памяти. Программа для большинства современных ЭВМ заносится в память машины и хранится там вместе с данными, занимая некоторое количество ячеек, т. е. некоторое пространство в памяти.

В нашем калькуляторе «Электроника Б3-34» есть отдельная память для данных и отдельная память для хранения программы, так называемая программная память. Память для данных состоит из 14 адресуемых ячеек, с которыми мы уже ознакомились. Программная память «Электроники» состоит из 99 ячеек, которым присвоены адреса: 00, 01, 02, ..., 98. В каждой ячейке программной памяти может храниться одна команда. Каждая ячейка состоит из двух разрядов, и команда хранится в ней в виде двухсимвольного кода. Отметим, что ранее сформулированный для адресуемых ячеек памяти принцип магнитофонной ленты справедлив также и для ячеек программной памяти.

2. Процессор. Это устройство предназначено для обработки информации и управления процессом обработки информации. Процессоры современных ЭВМ «умеют» выполнять различные арифметические и логические операции, операции записи информации в ячейки памяти и чтения содержимого этих ячеек, операции переходов и проверки условий и т. д.

Уточним здесь понятие программы.

Программа (точнее, программа на машинном языке) — это алгоритм решения задачи, представленный в виде последовательности команд процессора.

Таким образом, человеку, который решает задачу с помощью ЭВМ, необходимо знать список команд входного языка ЭВМ. С некоторыми командами «Электроники Б3-34» мы уже ознакомились, оставшиеся будут нами рассмотрены несколько позже. В момент выполнения той или иной команды программы в собственной памяти процессора должны находиться данные, которые необходимы для выполнения этой команды. Собственная память процессора «Электроники Б3-34» состоит из операционных регистров RGX и RGY, над содержимым этих регистров выполняются операции. Кроме того, в собственную память процессора входит RGX₁ — регистр предыдущего результата, с которым мы также уже знакомы, и регистры RGZ и RGT, с которыми нам еще предстоит познакомиться.

3. Устройства ввода и вывода. С помощью устройства ввода осуществляется занесение исходных данных и программы в ЭВМ. Ввод исходных данных на «Электронике Б3-34» осуществляется с помощью набора чисел на клавиатуре и занесения их в операционные регистры и адресуемые ячейки. Для ввода программы требуется с клавиатуры последовательно ввести в программную память калькулятора все команды. Ввод данных осуществляется в режиме «Автоматическая работа», а ввод программы — в режиме «Программирование». Переход из режима «АВТ»

в режим «ПРГ» осуществляется нажатием клавиш **F ПРГ**. Переход же из режима «ПРГ» в режим «АВТ» осуществляется нажатием клавиш **F АВТ**. Счет по программе проходит в режиме «АВТ».

С помощью устройства вывода ЭВМ сообщает человеку результаты работы программы. На современных ЭВМ информация может быть выведена, например, на печать или на экран дисплея. На калькуляторе «Электроника Б3-34» результаты счета заносятся в регистры калькулятора: операционные регистры и адресуемые ячейки. Содержимое RGX загорается на индикаторе. Для того чтобы прочитать содержимое других регистров, следует занести их содержимое в RGX.

Рассмотрим пример.

Составить программу для нахождения гипотенузы прямоугольного треугольника по его катетам a и b .

Решение. Воспользуемся формулой $c = \sqrt{a^2 + b^2}$.

Программа для калькулятора имеет вид:

| | | | | | |
|----|----|------------|----------------------|-----|--|
| 00 | ИП | A | Распределение памяти | | |
| 01 | F | x^2 | <i>a</i> | RGA | |
| 02 | ИП | B | <i>b</i> | RGB | |
| 03 | F | x^2 | <i>c</i> | RGX | |
| 04 | + | | | | |
| 05 | F | $\sqrt{ }$ | | | |
| 06 | | C/P | | | |

Слева от вертикальной черты указаны адреса программной памяти, а справа — соответствующие команды. Например, в ячейку 02 программной памяти заносится команда **ИП B**, т. е. команда RGB→RGX — чтение содержимого RGB. Отметим, что здесь впервые мы использовали команду

C/P — «стоп»,

которая прекращает счет. Эта команда необходима, если вы

считаете не вручную, а используете программный режим работы калькулятора.

Для того чтобы начать счет по программе, необходимо задать адрес первой команды программы, т. е. нажать клавиши

БП $n_1 n_2$.

Здесь $n_1 n_2$ — адрес первой команды программы. Для рассматриваемой нами программы адрес первой команды — 00, т. е. следует нажать на клавиши

БП 0 0.

Замечание. Если первая команда программы заносится по адресу 00, то вместо клавиш **БП 0 0** можно нажать клавишу **B/O**.

Как воспользоваться этой программой? Прежде всего следует занести ее в программную память калькулятора. Занесение программы в память калькулятора происходит в режиме «ПРГ».

Нажмите на клавиши **F ПРГ**, а затем введите программу, нажав последовательно на клавиши

ИП A F x^2 ИП B F x^2 + F $\sqrt{ }$ C/P.

При вводе программы содержимое счетчика адреса — адрес вводимой команды — загорается в двух последних разрядах индикатора калькулятора. Счет по программе осуществляется в режиме «АВТ». Для установки этого режима нажмите на клавиши **F АВТ**. Так как программа начинается с команды, записанной по адресу 00, нажмите на клавишу **B/O**

или на клавиши **БП 0 0**. Пусть, например, вы желаете найти гипotenузу прямоугольного треугольника с катетами $a=3$ и $b=4$. Занесите начальные данные в ячейки памяти

3 П A 4 П B,

нажмите на клавишу **C/P** — «пуск». На индикаторе загорается длина гипотенузы — 5. Пусть теперь вы пожелали найти гипотенузу c по катетам $a=6$, $b=8$. Нажмите на клавишу **B/O**

(или клавиши **БП** **0** **0**) и затем на клавиши **6** **П**

A **8** **П** **В**, нажмите на клавишу **С/П** — «пуск» и на индикаторе загорится длина гипотенузы — 10. Найдите длину гипотенузы еще для нескольких треугольников!

Упражнения

Составьте программы для нахождения:

а) объема шара по формуле $V = \frac{4}{3} \pi R^3$;

б) площади треугольника по формуле $S = \frac{1}{2} ab \sin \alpha$;

в) площади треугольника по формуле Герона

$$S = \sqrt{p(p-a)(p-b)(p-c)}, \text{ где } p = \frac{1}{2}(a+b+c);$$

г) длины вектора по формуле $|\vec{a}| = \sqrt{a_x^2 + a_y^2 + a_z^2}$;

д) пройденного пути при свободном падении по формуле

$$h = \frac{gt^2}{2}.$$

§ 1.4. УРОК 4. ЧТО ТАКОЕ АЛГОРИТМ

Знакомство с понятием алгоритма начнем с рассмотрения примеров.

Пример 1. Рассмотрим хорошо известную задачу о волке, козе и капусте. Напомним ее содержание. Некий человек должен перевезти через реку волка, козу и капусту. В лодку может поместиться человек либо с волком, либо с козой, либо с капустой. Если оставить волка с козой без человека, то волк съест козу, если же оставить козу наедине с капустой, то коза ее съест. Справивается, каким образом человеку перевезти волка, козу и капусту через реку.

Обозначим берег, на котором первоначально находится человек с волком, козой и капустой, через *A*, а противоположный берег — через *B*.

Человек с этой задачей успешно справится, если последовательно выполнит такие действия:

1. Перевезти козу с *A* на *B*.
2. Возвратиться самому на берег *A*.
3. Перевезти капусту с *A* на *B*.
4. Возвратиться с *B* на *A* вместе с козой.
5. Перевезти волка с *A* на *B*.
6. Возвратиться на берег *A* самому.
7. Перевезти козу с *A* на *B*.
8. Конец.

Приведенная выше система указаний задает алгоритм решения задачи.

Пример 2. Даны 10 мешков, набитых монетами. Известно, что все монеты в девяти мешках не фальшивые. Относительно же одного мешка известно, что в нем либо все монеты фальшивые, либо все настоящие. Фальшивая монета весит 11 г, а настоящая — 10 г. Как с помощью одного взвешивания выяснить, есть ли среди 10 мешков мешок с фальшивыми монетами, если есть — указать какой.

Выяснить требуемое можно, пользуясь такой системой словесных указаний:

1. Пронумеруйте мешки числами от 1 до 10.
2. Из каждого мешка извлеките столько монет, каков его номер.
3. Определите на весах суммарную массу *P* всех извлеченных монет.
4. Проверьте условие: « $P = 550$ г?». Если да, то перейдите к указанию 7, иначе — к следующему указанию.
5. Определите разность *R*, равную $P - 550$.
6. Объявите, что в мешке с номером *R* монеты фальшивые. Перейдите к указанию 8.
7. Объявите, что мешка с фальшивыми монетами нет.
8. Конец.

Эта система указаний задает алгоритм решения задачи. В 1946 году Дж. фон Нейман и Х. Голдстайн предложили задавать алгоритмы в виде схем. Схема рассмотренного алгоритма приведена на рисунке 5.

Пример 3. Известен такой простой способ проверки арбуза на спелость. Арбуз бросают в воду; если арбуз потонул, то его объявляют неспелым, если же всплыл — его объявляют спелым.

Представим процедуру описанной проверки в виде схемы (рис. 6). Пусть теперь у нас не один, а куча арбузов и перед нами поставлена задача подсчета числа спелых и числа неспелых арбузов. Пусть переменная *C* играет роль счетчика спелых арбузов, а переменная *H* — роль счетчика неспелых. То, что следует сделать, проверяя каждый арбуз, изображено на схеме, приведенной на рисунке 7. Первая попытка построить схему алгоритма для подсчета количества спелых и неспелых арбузов приводит нас к мысли многократного повторения схемы, изображенной на рисунке 7, т. е. получим схему, изображенную на рисунке 8.

Такая схема, конечно же, не может нас удовлетворить. Попытаемся найти иное решение. Вам ни о чём не говорит слово «цикл»? Поразмыслив немного, можно получить схему, изображенную на рисунке 9.

Конструкции такого типа называют циклическими или просто циклами. Однако применение такого цикла некорректно. Дело в

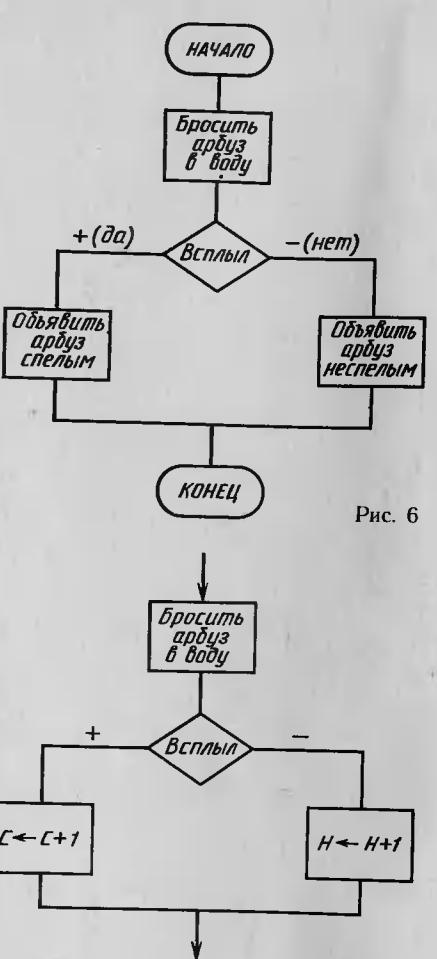
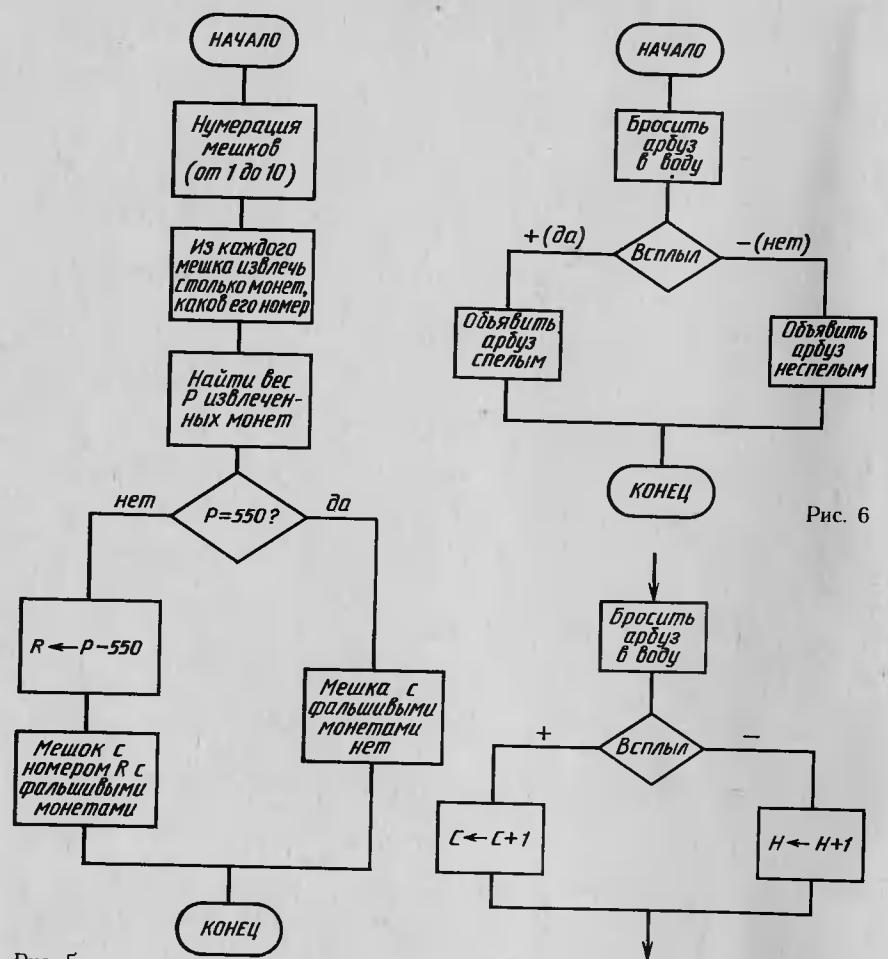


Рис. 7

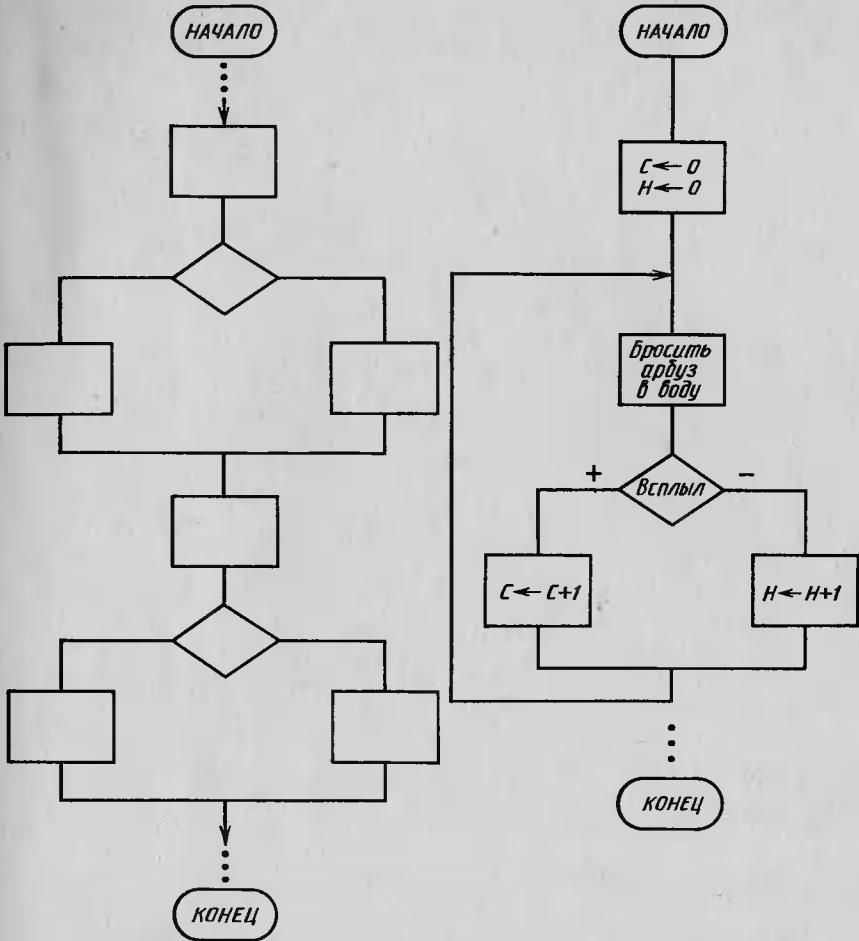


Рис. 9

тому, что рано или поздно возникнет ситуация, когда все арбузы проверены, а схема заставляет нас проверять на спелость очередной арбуз. Выйти из положения можно следующим образом. Перед проверкой арбуза выяснить, остались ли еще в куче арбузы, если это так, то продолжить циклический процесс, в противном случае «выйти из цикла» и окончить процесс подсчета спелых и неспелых арбузов. Итак, наконец, получим схему, изображенную на рисунке 10.

После работы этого алгоритма переменной C будет присвоено значение, равное количеству спелых, а переменной H — значение, равное количеству неспелых арбузов.

В рассмотренных выше примерах мы пользовались:

1. **Блоками-прямоугольниками** (рис. 11), в которых описывается выполнение некоторого действия, после чего предписывается перейти к блоку, к которому ведет выходящая из блока прямоугольника стрелка.

2. **Блоками-ромбами** (рис. 12), которые будем еще называть блоками проверки условия. Выполняется этот блок так:

Проверяется истинность условия P .

Если условие P истинно (выполнено),
то переходим к выполнению блока, к которому ведет стрелка,
помеченная знаком «+»,
иначе (если условие P ложно) переходим к выполнению блока,
к которому ведет стрелка, помеченная знаком «—».

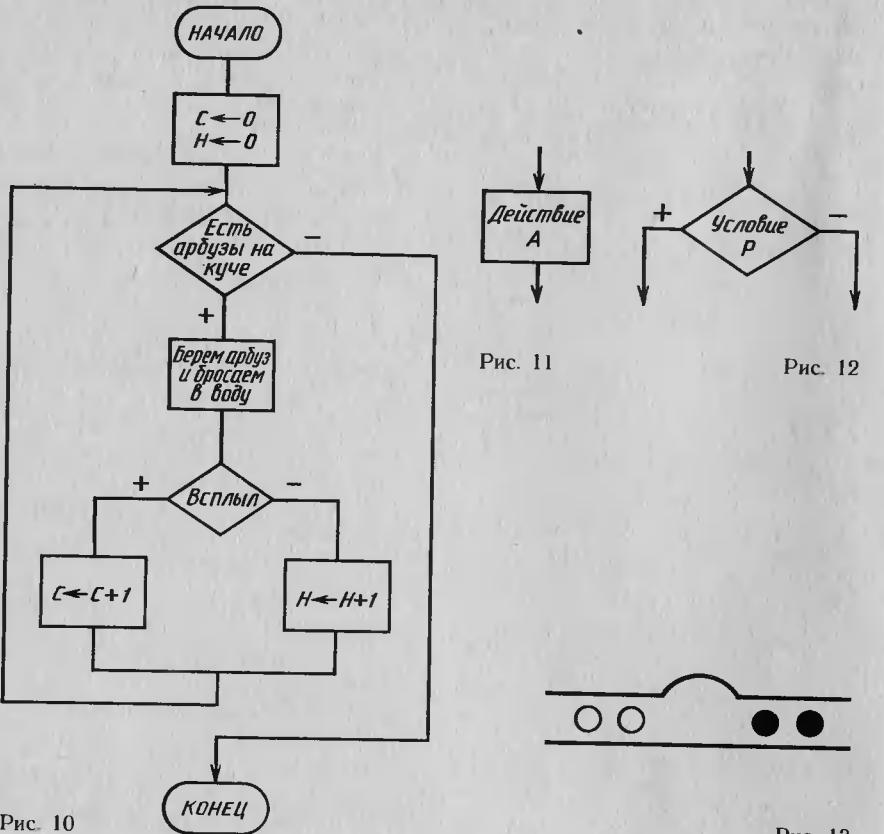


Рис. 11

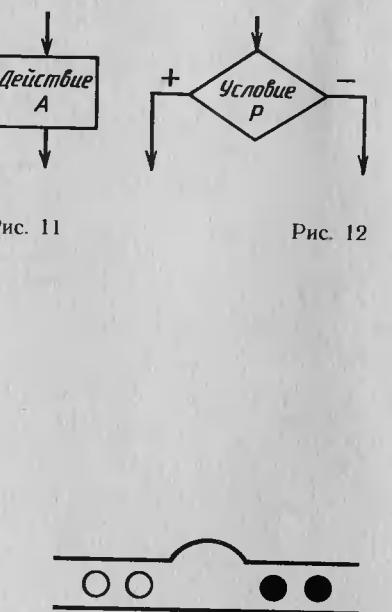


Рис. 13

3. Блоками НАЧАЛО или КОНЕЦ, которые располагаются в начале и конце схемы.

Дадим теперь интуитивное «определение» алгоритма [47]:

под алгоритмом понимают точное предписание о выполнении в определенном порядке некоторой системы указаний для решения всех задач данного типа.

Опишем кратко свойства алгоритмов. (Подробнее см. в [47].)

1. Алгоритм — точное предписание, т. е. алгоритм обладает свойством детерминированности.

От алгоритма требуется, чтобы его реализация не зависела от того, кто им пользуется, и всегда приводила к одним и тем же результатам. Указания, содержащиеся в алгоритме, не должны допускать различных толкований.

2. Свойство массовости алгоритма состоит в том, что он применим не для решения некоторой индивидуальной задачи, а для решения всех задач некоторого типа. Например, говорят

об алгоритме решения квадратного уравнения, об алгоритме нахождения наибольшего общего делителя двух чисел и т. д. В чем состоит массовость алгоритма в нашем примере с арбузами?

3. Результативность алгоритма — свойство, обеспечивающее получение результата за конечное число шагов.

Имеет ли место это свойство в нашем примере с арбузами?

Итак, мы представляем себе, что такое алгоритм. Алгоритмы будем задавать с помощью систем словесных указаний, схем и, разумеется, в виде программ для микроЭВМ «Электроника Б3-34».

Упражнения

1. Изучите инструкцию пользования телефоном-автоматом и представьте ее в виде схемы.

2. Представьте в виде схемы план проведения своих каникул (возможны варианты, зависящие от погоды, планов друзей и т. п.).

3. В узком коридоре, в стене которого имеется одна ниша (рис. 13), встретились двое людей, идущих в одну сторону, с двумя людьми, идущими в другую сторону. Как им разойтись? (В нише может уместиться лишь один человек.) Задайте алгоритм системой словесных указаний и с помощью схемы.

4. Обобщите предыдущую задачу на случай двух групп людей, по n человек в группе.

5. Составьте схему алгоритма нахождения фальшивой монеты среди 3^n монет с помощью рычажных весов.

6. Сформулируйте интуитивное «определение» алгоритма.

7. Ответьте на вопросы:

a) В чем состоит: свойство детерминированности алгоритма; свойство массовости; свойство результативности?

b) Для чего нужны: блоки-ромбы; блоки-прямоугольники?

b) Можно ли считать алгоритмом: кинофильм, рецепт приготовления блюда; правила игры в футбол; инструкцию пользования утюгом; телефонный справочник?

8. Порассуждайте на тему «Свойство массовости алгоритма и принцип адресности».

§ 1.5. УРОК 5. ВЕТВЛЕНИЯ

На уроке 3 мы составляли программы, в которых указанные команды выполнялись одна за другой в порядке их записи. Это так называемые линейные программы. Однако во многих случаях линейные программы не решают поставленной задачи. Так, например, составляя программу решения квадратного уравнения, мы должны указать, как следует поступить в зависимости от

Таблица 3

| | | | |
|-------|-------|-------|-------|
| $k+0$ | БП | $k+0$ | F |
| $k+1$ | n_1 | $k+1$ | n_1 |
| $k+2$ | n_2 | $k+2$ | n_2 |

2. Если
то P выполняется,
следующей будет выполняться команда, записанная
по адресу $k+2$,
иначе n_1n_2 .
следующей выполняется команда, записанная по
адресу n_1n_2 .

Пример. Дано программа:

| | | | |
|----|-----|------------|------------------------------|
| 00 | F | $x \geq 0$ | RGX ≥ 0 |
| 01 | 0 | 5 | 05 — адрес перехода по «—» |
| 02 | F | $\sqrt{ }$ | $\sqrt{RGX} \rightarrow RGX$ |
| 03 | БП | | |
| 04 | 0 | 6 | 06 — адрес перехода |
| 05 | F | x^2 | $(RGX)^2 \rightarrow RGX$ |
| 06 | C/P | | стоп |

Каков будет результат работы этой программы, если перед ее пуском будет набрано: а) число 4; б) число 0; в) число -4 ?

Решение.

а) Набранное число 4 заносится в RGX. Проверяем условие истинно, и следующей будет выполнена команда $F \sqrt{ }$.

Результат, число 2, заносится в RGX. Далее команда БП 0

6 осуществляет переход к команде С/П. Число 2 загорается на индикаторе.

Таблица 2

| Псевдокод | Схема алгоритма |
|--|-----------------|
| <pre> Если Условие P Действие A иначе Действие B Все — если </pre> | |

Как переводится (транслируется) альтернатива с псевдокода или с языка схем алгоритмов на язык калькулятора? Прежде чем ответить на этот вопрос, изучим команды безусловного и условного переходов на нашем калькуляторе.

Команда *безусловного перехода* прерывает последовательный порядок выполнения команд программы и осуществляет переход к выполнению команды по указанному адресу. Эта команда на микроЭВМ «Электроника Б3-34» имеет вид, приведенный в таблице 3. Здесь n_1n_2 — адрес команды, к которой осуществляется переход. Команда безусловного перехода заносится в две последовательные ячейки $k+0$ и $k+1$ программной памяти.

Команды *условного перехода* имеют вид, приведенный в таблице 4. Здесь P — одно из условий $x < 0$, $x \geq 0$, $x = 0$, $x \neq 0$, где x — содержимое RGX, n_1n_2 — адрес перехода. Команда *условного перехода* заносится в две последовательные ячейки $k+0$ и $k+1$ программной памяти. Выполняется эта команда так:

1. Проверяется условие P .

| | |
|-----|-----|
| x | RGI |
| y | RGX |

б) В этом случае проверяемое условие также истинно и, значит, будет выполняться та же последовательность команд. Результат — число 0 — загорается на индикаторе.

в) В этом случае содержимое RGX отрицательно — проверяемое условие ложно. Таким образом, следующей будет выполняться команда $F \boxed{x^2}$, а вслед за ней — команда $\boxed{C/P}$. Результат — число 16 — загорается на индикаторе.

Итак, легко понять, что наша программа извлекает квадратный корень из неотрицательных чисел, а отрицательные возводит в квадрат. Альтернатива на псевдокоде и на языке калькулятора приведена в таблице 5.

Таблица 5

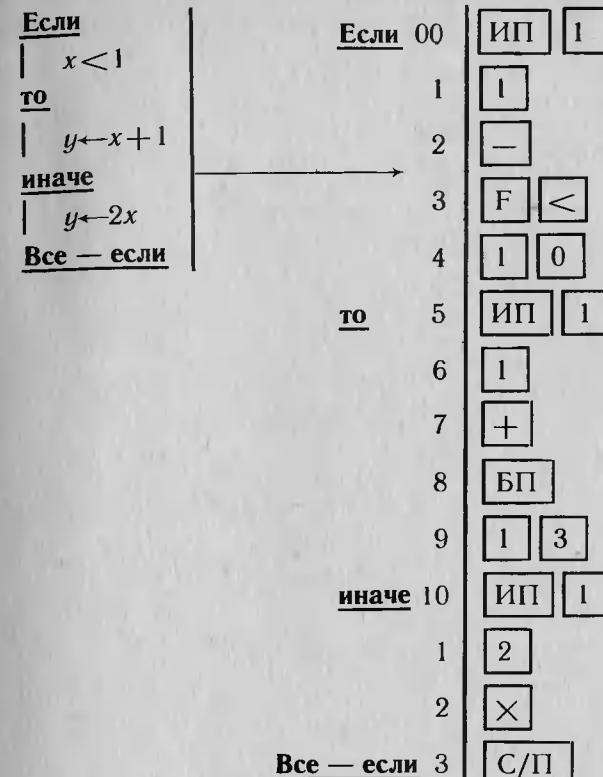
| Псевдокод | Калькулятор |
|--|---|
| <u>Если</u> Условие $P(x)$ <u>то</u> Действие A <u>иначе</u> Действие B <u>Все — если</u> | <u>Если — то — иначе</u> <u>Если</u> $x \rightarrow RGX$ <u>то</u> $F \boxed{P(x)}$ <u>иначе</u> <u>Действие</u> A <u>БП</u> <u>иначе</u> <u>Все — если</u> <u>Все — если</u> <u>иначе</u> <u>Действие</u> B <u>Все — если</u> |

Пользуясь этой таблицей, можно перевести альтернативу с псевдокода на язык калькулятора, при этом следует помнить, что x — число, сравниваемое с нулем. Рекомендуем пользоваться ею. Вы убедитесь, что это очень удобно.

Пример. Составить программу для вычисления выражения

$$y = \begin{cases} x+1, & \text{если } x < 1, \\ 2x, & \text{если } x \geq 1. \end{cases}$$

Решение. Запишем данную альтернативу на псевдокоде, а затем переведем ее на язык калькулятора:



Отметим, что здесь и в дальнейшем вместо \square , \square , \square ,

\square будем пользоваться обозначениями: $<$, \geq , $=$, \neq .

Следующая наша цель — научиться программировать ветвление по более чем двум направлениям. Познакомимся с двумя методами.

Первый метод — использование нескольких альтернатив. Второй метод основан на так называемой косвенной адресации.

Многозначное ветвление будем на псевдокоде обозначать так:

Выбор

Условие 1 | Действие A_1

Условие 2 | Действие A_2

Условие n | Действие A_n

Все — выбор

На схемах ветвления будем обозначать так, как показано на рисунке 14.

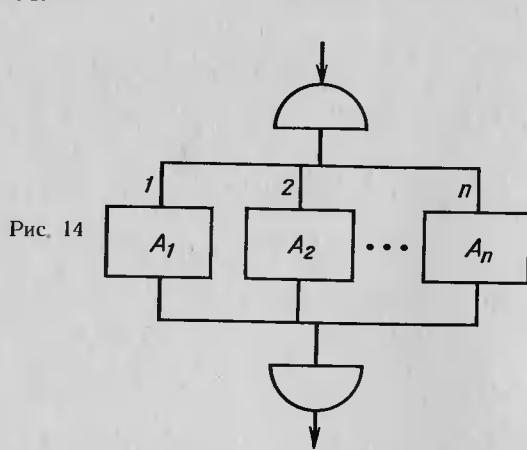


Рис. 14

1. Использование нескольких альтернатив

Пример. Составить программу вычисления выражения

$$y = \begin{cases} x+1, & \text{если } x < 2, \\ x+2, & \text{если } 2 \leq x < 3, \\ x+3, & \text{если } x \geq 3. \end{cases}$$

Решение. Нам нужно перевести на язык калькулятора такое многозначное ветвление:

Выбор

| | | |
|----------------|-----|-----------------------|
| $x < 2$ | $ $ | $y \leftarrow x + 1,$ |
| $2 \leq x < 3$ | $ $ | $y \leftarrow x + 2,$ |
| $x \geq 3$ | $ $ | $y \leftarrow x + 3.$ |

Все — выбор

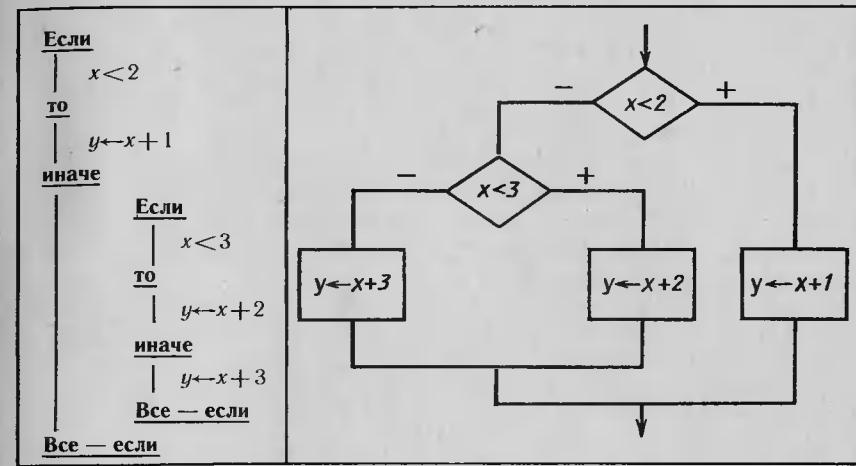
То же самое можно записать с помощью двух альтернатив (табл. 6). Теперь осуществим перевод на язык калькулятора. Будем пользоваться схемой, приведенной в таблице 5. Программа на языке калькулятора дана в таблице 7.

2. Использование косвенной адресации

Познакомимся с командами косвенных переходов. Команда безусловного косвенного перехода имеет вид:

K [БП] n

где n — адрес ячейки памяти калькулятора. Выполняется эта команда так:



Распределение памяти

| x | y |
|-----|-----|
| RG1 | RG2 |

Таблица 7

| | | | | | | | |
|--|-------|----|---|-----|---|---|---|
| <u>Если</u> $x < 2$ <u>то</u> $y \leftarrow x + 1$ <u>иначе</u> <u>Если</u> $x < 3$ <u>то</u> $y \leftarrow x + 2$ <u>иначе</u> $y \leftarrow x + 3$ <u>Все — если</u> <u>Все — если</u> | 00— | ИП | 1 | 2 | — | — | — |
| | —04 | F | < | 1 | 1 | | |
| | 05—08 | ИП | 1 | 1 | + | П | 2 |
| | 09—10 | БП | 2 | 6 | | | |
| | 11— | ИП | 1 | 3 | — | | |
| | —15 | F | < | 2 | 2 | | |
| | 16—19 | ИП | 1 | 2 | + | П | 2 |
| | 20—21 | БП | 2 | 6 | | | |
| | 22—25 | ИП | 1 | 3 | + | П | 2 |
| | 26—27 | ИП | 2 | С/П | | | |

- a) Если RG_n — один из регистров RG7, RG8, RG9, RGA, RGB, RGC, RGД, то после команды косвенного безусловного перехода будет выполняться команда, адрес которой — число, хранящееся в RG_n.
- б) Если RG_n — один из регистров RG0, RG1, RG2, RG3, то после команды косвенного безусловного перехода будет выполняться команда, адрес которой на единицу меньше числа, хранящегося в RG_n.
- в) Если RG_n — один из регистров RG4, RG5, RG6, то после команды безусловного косвенного перехода выполняется команда, адрес которой на единицу больше числа, хранящегося в RG_n.

Команды *условного косвенного перехода* имеют вид:

K [P(x)] [n], где P(x) — условие, а n — адрес ячейки памяти. Смысл косвенной адресации здесь такой же. Например,

команда **K [x < 0] 9** осуществляет переход к команде, адрес которой хранится в RG9, если содержимое RGX больше или равно 0. Если же содержимое меньше 0, то следующей будет выполняться команда, которая расположена непосредственно за командой косвенного условного перехода. Отметим здесь, что принцип косвенной адресации является дальнейшим развитием принципа адресности. Действительно, в рассмотренных командах переход осуществляется не по указанному адресу, а к команде, адрес которой записан по указанному адресу. Ситуация здесь аналогична такой. Если в незнакомом городе требуется найти человека, то вы направляйтесь в справочное бюро и там вам указывают нужный адрес. Так вот, ячейка памяти, адрес которой указан в команде косвенного перехода, и играет роль этого справочного бюро. Число же, хранящееся в этой ячейке, и будет искомым адресом перехода. Косвенная адресация используется не только в командах перехода, и у нас еще будет повод вернуться к этому вопросу. Сейчас же мы займемся снова многозначными ветвленийми.

Пример. Составить программу вычисления выражения

$$y = \begin{cases} x^2, & \text{если } m=1, \\ 2x, & \text{если } m=2, \\ x+1, & \text{если } m=3, \\ 0, & \text{если } m=4, \\ x-1, & \text{если } m=5. \end{cases}$$

Решение. Итак, требуется перевести на язык калькулятора такую структуру выбора:

Выбор

| | |
|-------|----------------------|
| $m=1$ | $y \leftarrow x^2$, |
| $m=2$ | $y \leftarrow 2x$, |
| $m=3$ | $y \leftarrow x+1$, |
| $m=4$ | $y \leftarrow 0$, |
| $m=5$ | $y \leftarrow x-1$. |

Все — выбор

Идея использования здесь косвенной адресации достаточно проста. Коль скоро команда косвенного безусловного перехода осуществляет переход к команде, адрес которой — число, хранящееся в некоторой ячейке памяти, то, меняя содержимое этой ячейки, мы можем заставить программу переходить к выполнению различных команд. Программа для калькулятора приведена в таблице 8.

Таблица 8

| 5m → RG7 | 00—03 | ИП | 1 | 5 | × | П | 7 |
|----------------------|-------|----|-----|---|-------|-----|---|
| | 04 | К | БП | 7 | | | |
| Распределение памяти | 05—07 | ИП | 2 | F | x^2 | С/П | |
| | 10—13 | ИП | 2 | 2 | × | С/П | |
| | 15—18 | ИП | 2 | 1 | + | С/П | |
| | 20—21 | 0 | С/П | | | | |
| | 25—28 | ИП | 2 | 1 | — | С/П | |

В заключение этого урока познакомимся с частным случаем структуры если — то — иначе. Предположим, что в случае истинности некоторого условия P следует выполнить действие A , в противном случае — ничего не выполнять. Такой частный случай альтернативы будем называть усеченной альтернативой или структурой если — то. Схема трансляции этой структуры с псевдокодом приведена ниже (табл. 9).

Таблица 9

| Псевдокод | Калькулятор | | | | | | |
|---|---|---|--------|--|--|--|--|
| <u>Если — то</u> | | | | | | | |
| <u>Если</u> Условие $P(x)$ <u>то</u> Действие A <u>Все — если</u> | <u>Если</u> $x \rightarrow RGX$ <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>F</td><td>$P(x)$</td></tr> <tr><td> </td><td> </td></tr> <tr><td> </td><td> </td></tr> </table> <u>то</u> Действие A <u>адрес</u> <u>Все — если</u> | F | $P(x)$ | | | | |
| F | $P(x)$ | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |

Эту структуру часто используют в связи с приемом, названным нами «приемом предварительного предположения». Объясним суть этого приема на примере. Составим программу вычисления выражения:

$$y = \begin{cases} 0, & \text{если } x < 0, \\ \sqrt{x}, & \text{если } x \geq 0. \end{cases}$$

Решение. Используем прием предварительного предположения.

| | |
|--|--|
| Считать предварительно y равным 0. | <u>Если — то</u> |
| <u>Если</u> $x \geq 0$ <u>то</u> изменить значение y <u>Все — если</u> | <u>Если</u> $x \geq 0$ <u>то</u> $y \leftarrow \sqrt{x}$ <u>Все — если</u> |

Составим программу для калькулятора, используя схему трансляции (табл. 10).

| | | | | | | |
|-------------------|-------------------------|-------|----|------------|-----|---|
| <u>Если</u> | $y \leftarrow 0$ | 00—01 | 0 | П | 2 | |
| <u>Если</u> | $x \geq 0$ | 02— | ИП | 1 | | |
| <u>то</u> | | —04 | F | \geq | 0 | 7 |
| | | 05— | F | $\sqrt{ }$ | | |
| | | —06 | П | 2 | | |
| <u>Все — если</u> | $y \leftarrow \sqrt{x}$ | 07—08 | ИП | 2 | C/P | |

Упражнения

1. Составьте программу решения квадратного уравнения. Если $D < 0$, то на индикаторе должна загораться специальная константа, например 11111111, которая заранее помещается в некоторую ячейку памяти.

2. Составьте программу решения уравнения $ax = b$. Учтите все случаи.

3. Составьте программу для вычисления выражения:

a) $y = \text{sign}(x) = \begin{cases} 1, & \text{если } x > 0, \\ 0, & \text{если } x = 0, \\ -1, & \text{если } x < 0; \end{cases}$

b) $y = \begin{cases} x^2, & \text{если } n \text{ четное,} \\ x^3, & \text{если } n \text{ нечетное.} \end{cases}$

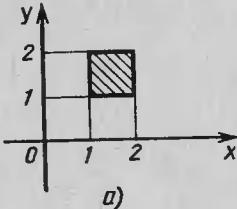
Указание. Используйте то, что если n четное, то $\cos n \pi = 1$; если n нечетное, то $\cos n \pi = -1$.

4. Составьте программу, которая проверяет, принадлежит ли точка заштрихованной области (рис. 15).

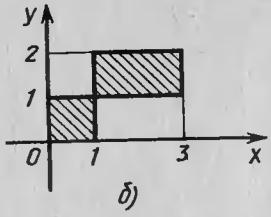
5. Составьте программу для вычисления выражения

$$y = \begin{cases} t^2, & \text{если } n = 1, \\ t^2 + t, & \text{если } n = 2, \\ t^2 + t + 1, & \text{если } n = 3. \end{cases}$$

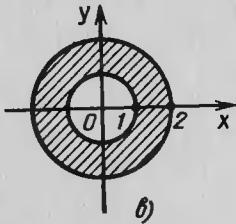
6. Составьте программу, которая проверяла бы существование треугольника с длинами сторон a, b, c . (Пусть в случае «да» загорается заранее заготовленная константа, а в случае «нет» — ERROR.)



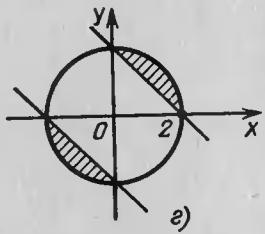
a)



б)



в)



г)

Рис. 15

§ 1.6. УРОК 6. ЦИКЛЫ

На уроке 4, занимаясь конструированием схем, мы встречались с ситуацией, когда некоторые действия повторяются много-кратно, циклически. На данном уроке мы познакомимся с различными видами циклов и научимся составлять циклические программы на калькуляторе.

1. Цикл-пока. Цикл этого типа повторяет некоторое действие A , пока выполняется условие P . Как только условие P становится ложным, осуществляется выход из цикла.

Цикл-пока записывается на псевдокоде и на языке схем алгоритмов так, как показано в таблице 11.

Таблица 11

| Псевдокод | Схема алгоритма |
|---|---|
| <u>Цикл</u> <u>пока</u> Условие P <u>выполняй</u> Действие A <u>Все — цикл</u> | <pre> graph TD P{P} --> A[A] A --> P </pre> |

На язык калькулятора цикл-пока переводится с помощью схемы, приведенной в таблице 12.

Таблица 12

| Псевдокод | Калькулятор | | | | | | |
|--|---|---|------|--|--|----|--|
| <u>Цикл — пока</u> | | | | | | | |
| <u>Подготовить</u> <u>Цикл</u> <u>пока</u> Условие $P(x)$ <u>выполняй</u> Действие A <u>Все — цикл</u> <u>Завершить</u> | <u>Подготовить</u> $x \rightarrow RGX$ <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>F</td><td>P(x)</td></tr> <tr><td> </td><td> </td></tr> </table> <u>адрес</u> <u>Все — цикл</u> <u>Действие</u> A <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>БП</td></tr> <tr><td> </td></tr> </table> <u>адрес</u> <u>пока</u> <u>Завершить</u> | F | P(x) | | | БП | |
| F | P(x) | | | | | | |
| | | | | | | | |
| БП | | | | | | | |
| | | | | | | | |
| | | | | | | | |

Пример. Составить программу для нахождения факториала, используя цикл-пока: $f(n) = n! = 1 \cdot 2 \cdot 3 \cdots \cdot n$.

Решение. На псевдокоде алгоритм нахождения факториала запишется так:

| | |
|--------------------------|----------------------|
| <u>Начало</u> | |
| <u>Ввод</u> (n) | |
| $i \leftarrow 1$ | подготовка цикла |
| $f \leftarrow 1$ | |
| <u>Цикл</u> | |
| <u>пока</u> | |
| $n \geq i$ | условие P |
| <u>повторяй</u> | |
| $f \leftarrow f \cdot i$ | повторяющее действие |
| $i \leftarrow i + 1$ | |
| <u>Все — цикл</u> | завершение цикла |
| <u>Конец</u> | |

Таблица 14

| Псевдокод | Схема алгоритма |
|--|--|
| <pre> Цикл пока $n \geq i$ выполняй $f \leftarrow f \cdot i$ $i \leftarrow i + 1$ все — цикл </pre> | <pre> graph TD P{P} -- "+" --> A[A] A --> P </pre> |

истинным, осуществляется выход из цикла. На язык калькулятора цикл-до переводится в виде таблицы 15.

Пример. Составить программу для нахождения факториала, используя цикл-до: $f(n) = n! = 1 \cdot 2 \cdots \cdot n$.

Решение. Как и в предыдущем примере, начнем с описания алгоритма на псевдокоде.

Начало

Ввод (n)

$f \leftarrow 1$ | подготовка цикла

$i \leftarrow 1$

Цикл

повторяй

$f \leftarrow f \cdot i$
 $i \leftarrow i + 1$

до

$n < i$

все — цикл

индикация f

завершение цикла

Конец

С помощью приведенной выше схемы переведем алгоритм нахождения факториала с псевдокода на язык калькулятора. В данном случае тело цикла — последовательность операторов

$$f \leftarrow f \cdot i, i \leftarrow i + 1$$

повторяется до тех пор, пока не станет истинным условие $n < i$. Программа на языке калькулятора приведена в таблице 16.

Таблица 13

| | | |
|---|-------|--------------|
| $i \leftarrow 1; f \leftarrow 1$ Цикл пока $n \geq i$ выполняй $f \leftarrow f \cdot i$ $i \leftarrow i + 1$ все — цикл | 00—02 | 1 П 1 П 2 |
| | 03— | ИП 3 ИП 1 — |
| | | F \geq |
| | —07 | 1 8 |
| | 08— | ИП 2 ИП 1 |
| | | \times П 2 |
| | | ИП 1 1 + П 1 |
| | | БП |
| | —17 | 0 3 |
| | 18—19 | ИП 2 С/П |

С помощью приведенной выше схемы осуществим перевод алгоритма с псевдокода на язык калькулятора. Ввод числа n , факториал которого мы ищем, будем засыпать в RG3 перед пуском программы.

Подготовка цикла — начальные присваивания:

$$i \leftarrow 1$$

$$f \leftarrow 1$$

Условие P в данном случае — $n \geq i$, а тело цикла, т. е. повторяемые действия, операторы присваивания: $f \leftarrow f \cdot i$; $i \leftarrow i + 1$, т. е. переменная умножается на очередное натуральное число i , пока значение i не превышает n . Завершение цикла — это вычисление ответа — значения переменной f на индикаторе (см. табл. 13).

2. Цикл-до. Цикл-до записывается на псевдокоде и на языке алгоритмических схем так, как показано в таблице 14.

Цикл этого типа повторяет действие A до тех пор, пока не станет истинным условие P . Как только условие P становится

Таблица 15

| Псевдокод | Калькулятор | |
|--|--|--|
| <u>Цикл — до</u> | | |
| Подготовить Цикл <u>повторять</u> <u>до</u> Действие A <u>до</u> Условие $P(x)$ Все — цикл Завершить | повторять <u>до</u> Все — цикл | Подготовить Действие A $x \rightarrow RGX$ <u>адрес</u> <u>повторять</u> Завершить |

| Псевдокод | Калькулятор | |
|---|--|--|
| <u>Цикл n раз (i)</u> | | |
| Подготовить Цикл $n \rightarrow RGI$ $(I=0, \text{ или } 1, \text{ или } 2, \text{ или } 3)$ Тело цикла Все — цикл Завершить | Цикл <u>адрес</u> <u>цикла</u> Завершить | Подготовить Тело цикла Все — цикл Завершить |

Таблица 16

| | | |
|--|-------|--|
| <i>i</i> \leftarrow 1; <i>f</i> \leftarrow 1 Цикл <u>повторять</u> $j \leftarrow i \cdot i$ <u>до</u> <i>n</i> $<$ <i>i</i> Все — цикл | 00—02 | |
| | 03— | |
| | | |
| | | |
| | | |
| | —10 | |
| | 11— | |
| | | |
| | | |
| | —15 | |
| | 16—17 | |

3. Арифметический цикл

В том случае, когда известно, сколько раз должна повторяться группа команд, входящих в тело цикла, цикл будем называть

арифметическим. В разобранном выше примере вычисления факториала натурального числа n заранее известно, что действия

$$\begin{aligned} f &\leftarrow f \cdot i, \\ i &\leftarrow i + 1 \end{aligned}$$

должны выполниться n раз.

Для организации арифметических циклов на микроЭВМ «Электроника Б3-34» имеются специальные средства. Сейчас мы познакомимся с тем, как такие циклы можно наиболее экономно записывать на языке калькулятора.

На псевдокоде арифметические циклы будем записывать так:

Цикл n раз (i)

Тело
 цикла
Все — цикл

Здесь i — параметр цикла. Выполняется этот цикл так. Переменной i присваивается значение 1 и затем выполняются действия, входящие в тело цикла. Переменной i присваивается значение 2 и повторно выполняются действия, входящие в тело цикла, и т. д. Переменной i присваивается значение n , выполняются действия, входящие в тело цикла, и осуществляется выход из цикла. С псевдокода на язык калькулятора арифметический цикл переводится с помощью такой схемы (см. табл. 17). В подготовку цикла обязательно включается занесение числа повторений n в один из регистров: RG0, RG1, RG2, RG3.

Пример. Составить программу нахождения факториала числа n :

$$f(n) = n! = 1 \cdot 2 \cdot \dots \cdot n.$$

Цикл организовать как арифметический.

Решение. Составим программу на псевдокоде и, пользуясь схемой перевода арифметического цикла, получим программу для калькулятора (табл. 18).

Таблица 18

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|--|-------|---|-----|-----|-----|-----|------------------|-------|---|---|----|---|--|----|---|----|---|---|----|---|--|---|----|--|--|---|---|--|--|----|---|------|--|
| Распределение памяти <table border="1"> <tr><td>n</td><td>RG0</td></tr> <tr><td>i</td><td>RG0</td></tr> <tr><td>j</td><td>RG1</td></tr> </table> | n | RG0 | i | RG0 | j | RG1 | $f \leftarrow 1$ | 00—01 | <table border="1"> <tr><td>1</td><td>II</td><td>1</td><td></td></tr> <tr><td>ИП</td><td>I</td><td>ИП</td><td>0</td></tr> <tr><td>X</td><td>II</td><td>1</td><td></td></tr> <tr><td>F</td><td>L0</td><td></td><td></td></tr> <tr><td>0</td><td>2</td><td></td><td></td></tr> <tr><td>ИП</td><td>I</td><td>C/II</td><td></td></tr> </table> | 1 | II | 1 | | ИП | I | ИП | 0 | X | II | 1 | | F | L0 | | | 0 | 2 | | | ИП | I | C/II | |
| n | RG0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| i | RG0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| j | RG1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | II | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ИП | I | ИП | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| X | II | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| F | L0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0 | 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ИП | I | C/II | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| $j \leftarrow j \cdot i$ | 02— | <table border="1"> <tr><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td></tr> </table> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | —07 | <table border="1"> <tr><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td></tr> </table> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 08—09 | <table border="1"> <tr><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td></tr> </table> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | <table border="1"> <tr><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td></tr> </table> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | <table border="1"> <tr><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td></tr> </table> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | <table border="1"> <tr><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td></tr> </table> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Перед пуском этой программы следует занести n в RG0.

Пример. Составить программу для нахождения наибольшего общего делителя двух натуральных чисел x и y .

Решение. Схема этого алгоритма представлена на рисунке 16. Псевдокод и программа для калькулятора, полученная с помощью схем перевода альтернативы и цикла-пока, приведены в таблице 19. Перед пуском этой программы следует занести x в RGA и y в RGB.

Упражнения

1. Составьте программу вычисления выражения:

a) $\frac{1}{2} + \frac{2}{3} + \frac{3}{4} + \dots + \frac{n-1}{n}$;

б) $1 \cdot 2 + 1 \cdot 2 \cdot 3 + \dots + 1 \cdot 2 \cdot 3 \cdot \dots \cdot n$.

2. Составьте программу нахождения n -го члена последовательности:

a) $x_n = \cos^n\left(\frac{1}{n}\right)$;

б) $x_n = \frac{1}{1^2} + \frac{1}{3^2} + \frac{1}{5^2} + \dots + \frac{1}{(2n-1)^2}$;

в) $x_n = \left(1 + \frac{1}{n}\right)^n$; г) $x_n = 1 - \frac{1}{3} + \frac{1}{5} - \dots + (-1)^{n+1} \cdot \frac{1}{2n-1}$.

3. Пусть x_0 — произвольное число, а $x_{n+1} = \frac{1}{2}(x_n + \frac{a}{x_n})$; известно, что при $x_0 > 0$ $\lim_{n \rightarrow \infty} x_n = \sqrt{a}$. Составьте, исходя из этого, программу нахождения квадратного корня числа $a > 0$.

4. Если обозначить через S_m площадь правильного m -угольника, вписанного в круг, то площадь S_{2m} правильного $2m$ -угольника вычисляется по формуле

$$S_{2m} = \frac{m}{2} \sqrt{2 - 2 \sqrt{1 - \left(\frac{2S_m}{m}\right)^2}}.$$

Найдите приближенно число π , составив программу для нахождения площади произвольного правильного 2^n -угольника.

5. Составьте программу для вычисления выражения:

a) $C_n^k = \frac{n!}{k!(n-k)!}$;

б) $A_n^k = n \cdot (n-1) \cdot \dots \cdot (n-k+1)$.

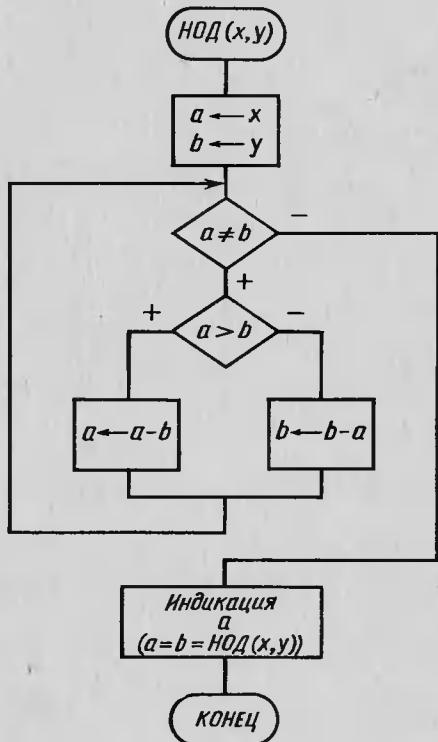


Рис. 16

Таблица 19

| | | | | | | | |
|----------------------|------|-------|----|---|-----|---|---|
| Распределение памяти | Цикл | 00— | ИП | A | ИП | B | — |
| | | — | F | ≠ | | | |
| | | —04 | 2 | 2 | | | |
| | | 05— | ИП | B | ИП | A | — |
| | | — | F | < | | | |
| | | —09 | 1 | 6 | | | |
| | | 10— | ИП | A | ИП | B | |
| | | — | — | П | A | | |
| | | —15 | БП | 2 | 0 | | |
| | | 16— | ИП | B | ИП | A | |
| | | — | — | П | B | | |
| | | —19 | — | — | | | |
| | | 20— | БП | | | | |
| | | —21 | 0 | 0 | | | |
| | | 22—23 | ИП | A | С/П | | |

§ 1.7. УРОК 7. МАССИВЫ И КОСВЕННАЯ АДРЕСАЦИЯ

Познакомимся прежде всего с понятием массива. Мы будем рассматривать лишь так называемые одномерные массивы, которые для краткости будем называть просто массивами. Понять, что такое массив, нам помогут приведенные ниже примеры.

Пример 1.

Рассмотрим совокупность учащихся 9A класса винницкой средней школы № 32. Всякому учащемуся этого класса соот-

ветствует его порядковый номер в журнале. Возникает ситуация, в которой правомерно говорить о массиве учащихся:

9A (1), 9A(2), ..., 9A (30).

Вы уже догадались, что в этом классе 30 учеников: 9A (1) — ученик, записанный в журнале первым, 9A (2) — ученик, записанный в журнале вторым, и т. д., 9A (30) — ученик, записанный в журнале последним. Отметим, что рассматриваемая совокупность учащихся имеет общее имя — «9A». Это так называемое имя массива. Каждому же отдельному учащемуся — элементу массива — соответствует определенное значение индекса, которое указано в скобках сразу же за именем массива. Индексом может быть не только натуральное число, но и некоторая переменная. Например, можно говорить об i -м ученике 9A класса. Чтобы продемонстрировать использование массивов, рассмотрим алгоритм подсчета количества учащихся, посещающих факультатив по программированию.

Схема этого алгоритма и описание на псевдокоде приведены на рисунке 17.

Начало
 $i \leftarrow 1$
 $F \leftarrow 0$
Цикл
пока $i \leq 30$
выполняй
если 9A (i)
 посещает
 факультатив
то $F \leftarrow F + 1$
все — если
 $i \leftarrow i + 1$
все — цикл
конец

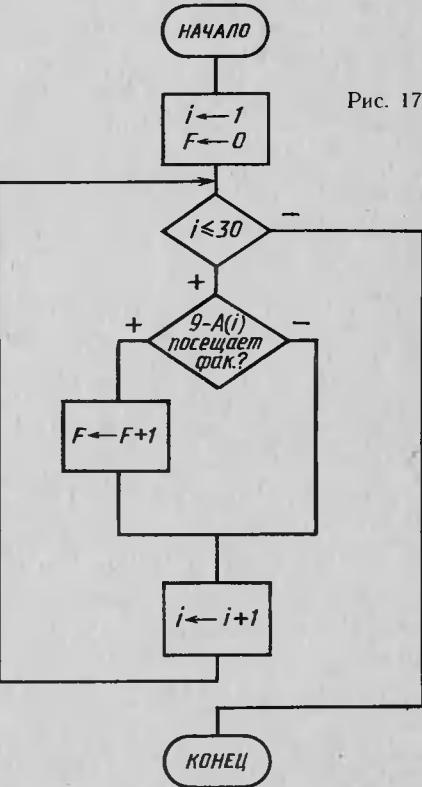


Рис. 17

После завершения работы этого алгоритма переменной F , которая играет роль счетчика, будет присвоено число, равное количеству учащихся 9А класса, посещающих факультатив по программированию.

Пример 2. Рассмотрим массив вершин некоторого n -угольника: A_1, A_2, \dots, A_n . Здесь A — имя массива, а индексы — порядковые номера вершин. В этом примере термин «индекс» оправдывает свое название (индекс в переводе — нижний). В различных языках программирования индекс записывается в скобках после имени массива; мы же будем использовать различные формы записи. Рассмотрим алгоритм нахождения периметра n -угольника. Схема этого алгоритма и описание на псевдокоде приведены на рисунке 18. Расстояние между вершинами A_i и A_j будем обозначать $d(A_i, A_j)$. После окончания работы этого алгоритма переменной p будет присвоено значение, равное периметру n -угольника.

```

Начало
p←0
i←1
Цикл
пока
    i≤n-1
    выполни
        d←d(Ai, Ai+1)
        p←p+d
        i←i+1
    Все — цикл
p←p+d(An, A1)
Конец

```

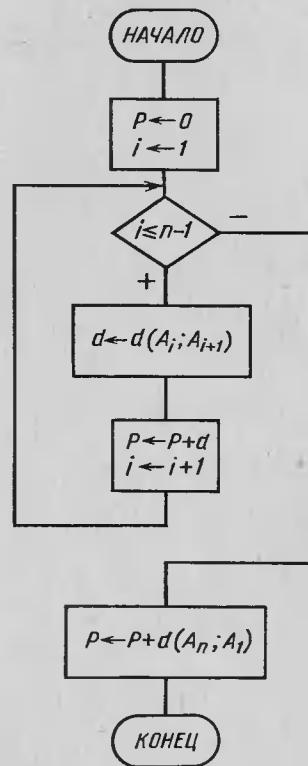


Рис. 18

В дальнейшем мы будем рассматривать так называемые числовые массивы, т. е. массивы, элементы которых — числа. Дело в том, что наш калькулятор позволяет работать именно с такими массивами. Примером числового массива может слу-

жить совокупность первых членов некоторой числовой последовательности:

$$a_1, a_2, \dots, a_n.$$

Не следует думать, однако, что индексы — обязательно последовательные натуральные числа, начиная с 1. Можно рассматривать такой массив: $a_n, a_{n+1}, \dots, a_{n+k}$. Более того, в некоторых языках программирования индексы — не обязательно натуральные числа: ими могут быть объекты иной природы. Мы такие случаи рассматривать не будем.

Пример 3. Схема алгоритма и описание на псевдокоде нахождения наибольшего числа среди чисел a_1, a_2, \dots, a_n изображены на рисунке 19, а описание на псевдокоде приведено ниже. Идея этого алгоритма достаточно проста. Вводится переменная \max , которой вначале присваивается значение a_1 : далее, значение этой переменной сравнивается с a_2 , если оказывается, что $\max < a_2$, то переменной \max будет присвоено значение a_2 . Затем значение переменной \max сравнивается с a_3 и т. д., пока не будут исчерпаны все числа массива.

```

Начало
max←a1
Цикл n раз (i)
    Если
        max < ai
        то
            max←ai
    Все — если
    Все — цикл
Конец

```

Теперь наша ближайшая задача — научиться переводить алгоритмы, содержащие массивы и представленные в виде алгоритмических схем или на псевдокоде, на язык калькулятора. Для этого нам придется предварительно ознакомиться с командами косвенного чтения содержимого ячеек памяти и с командами косвенной записи в ячейки памяти калькулятора. С использованием косвенной адресации мы уже встречались, изучая команды косвенных переходов. Смысл косвенной адресации здесь тот же.

Команды косвенной записи на нашем калькуляторе имеют вид:

[К] [П] n , где n — адрес некоторой ячейки памяти. По этой команде содержимое RGX заносится в ячейку памяти, адрес которой хранится в RG n либо отличается на 1 от этого адреса.

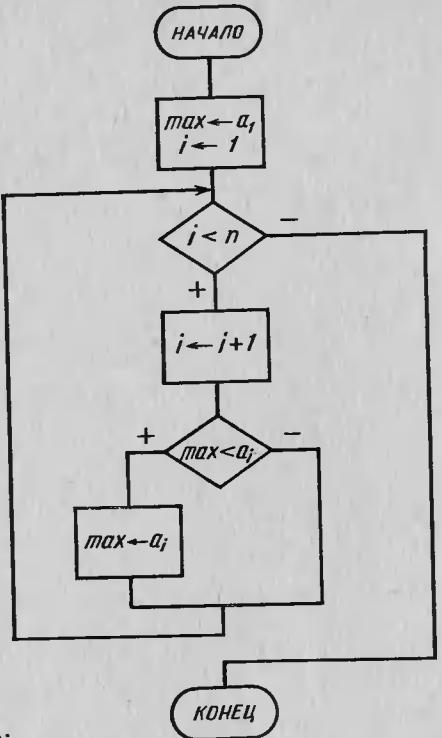


Рис. 19

Более точно:

- a) Если n — одно из значений 0; 1; 2; 3;
то:
1. Содержимое RGn уменьшается на 1.
2. Содержимое RGX заносится в ячейку, адрес которой хранится в RGn .
- b) Если n — одно из значений 4; 5; 6;
то:
1. Содержимое RGn увеличивается на 1.
2. Содержимое RGX заносится в ячейку, адрес которой хранится в RGn .
- v) Если n — одно из значений 7; 8; 9; A; B; C; D;
то:
содержимое RGX заносится в ячейку, адрес которой хранится в RGn .

Отметим здесь, что регистрам RGA; RGB; RGC; RGD присвоены адреса 10; 11; 12; 13 соответственно. Например, последовательно нажмите на клавиши:

| | | |
|---|---|---|
| 9 | П | 7 |
|---|---|---|

| | | |
|---|---|---|
| 1 | 0 | 0 |
|---|---|---|

| | | |
|---|---|---|
| K | П | 7 |
|---|---|---|

При этом число 100 занесется в RG9, так как в команде косвенной адресации

| | | |
|---|---|---|
| K | П | 7 |
|---|---|---|

 указан адрес ячейки RG7, в которую в свою очередь хранится число 9 — адрес ячейки RG9. Чтобы убедиться в этом, прочтите содержимое ячейки RG9. Нажмите на клавиши

| | | |
|----|----|---|
| Cx | ИП | 9 |
|----|----|---|

На индикаторе загорится число 100.

Команды косвенного чтения на калькуляторе имеют вид:

| | | |
|---|----|---|
| K | ИП | n |
|---|----|---|

Опишите, как выполняются эти команды по аналогии с командами косвенной записи.

Теперь уже несложно будет понять, как переводить на язык калькулятора алгоритмы, в которых присутствуют массивы. Вернемся к примеру алгоритма нахождения наибольшего числа среди a_1, a_2, \dots, a_n . Пусть эти числа занесены соответственно в ячейки RG1, RG2, ..., RGn. Учитывая ресурсы нашего калькулятора, будем считать, что $n \leq 11$. Согласно алгоритму значение переменной max всякий раз сравнивается с очередным числом a_i . Для значений переменной max выделим RGC, а для значений переменной i — RGD. Таким образом, команда

| | | |
|---|----|---|
| K | ИП | D |
|---|----|---|

будет означать чтение числа a_i , где i — значение, хранящееся в RGD в момент выполнения этой команды. Итак, программа для калькулятора будет выглядеть так, как указано в таблице 20. Перед пуском этой программы следует занести числа a_i в регистры RG i , а затем набрать n на клавиатуре.

Занесите эту программу в программную память калькулятора. Пусть, например, $a_1 = 1; a_2 = 5; a_3 = 6; a_4 = 4; a_5 = 8; a_6 = 11; a_7 = -2; a_8 = 3; a_9 = 6; a_{10} = 7; a_{11} = 2$. Занесите эти числа в ячейки RG1; RG2; RG3; RG4; RG5; RG6; RG7; RG8; RG9; RGA; RGB соответственно. Наберите на клавиатуре число $n = 11$, а затем осуществите пуск программы, нажав на клавиши

| |
|-----|
| B/O |
|-----|

| |
|-----|
| C/P |
|-----|

. На индикаторе загорится наибольшее из чисел — 11. Далее наберите на клавиатуре число $n = 4$ и нажмите на клавиши

| | |
|-----|-----|
| B/O | C/P |
|-----|-----|

. На индикаторе загорится наибольшее из чисел a_1, a_2, a_3, a_4 , т. е. число 6.

Таблица 20

| | | | | | | | |
|---|-------|----|----|-----|----|---|---|
| <p style="text-align: center;"><u>Цикл <i>n</i> раз (<i>i</i>)</u></p> <p style="text-align: center;">Если</p> <p style="text-align: center;"><i>a_i > max</i></p> <p style="text-align: center;">то</p> <p style="text-align: center;"><i>max←a_i</i></p> <p style="text-align: center;"><u>Все — если</u></p> <p style="text-align: center;"><u>Все — цикл</u></p> | 00 | П | 0 | | | | |
| | 01—02 | ИП | 1 | П | С | | |
| | 03—04 | ИП | 0 | П | Д | | |
| | 05— | ИП | С | К | ИП | Д | — |
| | —00 | F | < | 1 | 2 | | |
| | 10— | К | ИП | Д | | | |
| | —11 | П | С | | | | |
| | 12— | F | L0 | | | | |
| | —13 | 0 | 3 | | | | |
| | 14—15 | ИП | С | С/П | | | |

Упражнения

1. Данна программа (табл. 21):

Выясните, для вычисления какого выражения эта программа составлена.

2. Составьте программу нахождения наименьшего из чисел a_1, a_2, \dots, a_n .

3. Составьте программу для вычисления выражения

$$S = \sum_{i=1}^n a_i \cdot b_i (1 \leq n \leq 5).$$

4. Составьте программу вычисления выражения

$$\sigma = \sum_{i=1}^n \sin a_i.$$

5. Составьте программу вычисления выражения

$$P = a_1 \cdot a_2 \cdot a_3 \cdot \dots \cdot a_n.$$

6. Составьте программу для вычисления выражения

$$P = a_1 \cdot a_2^2 \cdot a_3^3 \cdot \dots \cdot a_n^n.$$

7. Составьте программу для вычисления выражения

$$R = \sum_{a_i \geq c} a_i - \sum_{a_i < c} a_i.$$

Таблица 21

| <u>Набор <i>n</i> Пуск</u> | 00 | П | 0 | | | | |
|----------------------------|------|-------|-----|---|----------------|--|--|
| | цикл | 01—02 | 0 | П | С | | |
| 03— | ИП | 0 | П | Д | | | |
| — | К | ИП | Д | F | x ² | | |
| —09 | ИП | С | + | П | С | | |
| 10—11 | F | L0 | 0 | 3 | | | |
| 12—13 | F | √ | С/П | | | | |

§ 1.8. УРОК 8. ПОЛЬСКАЯ ЗАПИСЬ. ТРАНСЛЯЦИЯ ВЫРАЖЕНИЙ

После этого урока мы должны уметь наиболее рационально составлять программы вычисления арифметических выражений для нашего калькулятора. Напомним, что арифметические выражения конструируются из операндов, т. е. имен переменных и чисел, знаков операций и скобок. Порядок выполнения операций регламентируется старшинством операций (приоритетом) и расстановкой скобок. Нам понадобится здесь новая форма записи арифметических выражений, предложенная польским логиком Лукасевичем. Это так называемая обратная бесскобочная, или польская, запись. Мы увидим, что удобней всего программировать арифметическое выражение, если оно задано именно таким образом. Однако вначале познакомимся еще с одним новым понятием — стеком, или стековой памятью.

Стек будем представлять себе как совокупность вертикально расположенных ячеек памяти. Это устройство по мере поступления в него данных запоминает их последовательность. Верхнюю ячейку стека будем называть его вершиной. Со стеком связаны две основные операции.

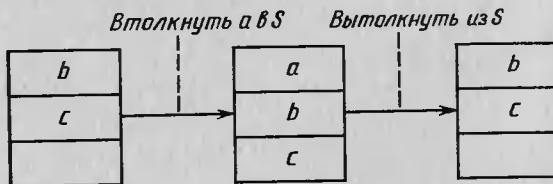
1. Втолкнуть *a* в *S*

Выполняется эта операция так: данное *a* заносится в вершину стека *S* (*S* — имя стека), а ранее хранимые в стеке данные «проталкиваются» в нижние ячейки, т. е. каждое данное пересыпается в ячейку, расположенную под ним.

2. Вытолкнуть из S

Эта операция выполняется так: содержимое вершины стека S выталкивается из него, а остальные данные «подтягиваются», т. е. каждое из них пересыпается в ячейку, расположенную над ним. Эти операции поясняет рисунок 20.

Рис. 20



Итак, стек — память, организованная по принципу: «последний вошел — первый вышел». Чтобы лучше себе представить, как устроен стек, полезно обратиться к аналогии с магазином огнестрельного оружия или с железнодорожным тупиком. Можно также представить себе стек как стопку шашек. Каждая шашка — данное. Втолкнуть данное в стек — значит положить на стопку еще одну шашку; вытолкнуть — значит снять с нее верхнюю шашку. Преимущество стека состоит в том, что при его использовании не нужно заботиться об адресации данных.

Перейдем теперь к рассмотрению польской записи арифметических выражений. В этой записи операции записываются после операндов. Например, $ab +$ означает $a+b$ или $ab+c\times$ означает $(a+b)\cdot c$. Остановимся несколько подробней на том, как вычислять значения выражений, представленных в польской записи. Здесь мы и воспользуемся понятием стека. Пусть цепочка символов $a_1a_2\dots a_n$ — некоторое арифметическое выражение в польской записи (для простоты считаем, что имена переменных однобуквенные). Будем последовательно просматривать символы этой цепочки. Если обозреваемый символ a_i — операнд, то его следует втолкнуть в стек. Если обозреваемый символ — унарная операция, то следует применить эту операцию к содержимому вершины стека и заменить содержимое вершины результатом этой операции. Если же обозреваемый символ — бинарная операция, то ее следует выполнить над содержимым двух верхних ячеек стека (второй операнд в вершине); затем из стека выталкиваются операнды, находящиеся в двух верхних ячейках, а полученный результат операции вталкивается в стек.

Рассмотрим пример. Пусть требуется вычислить выражение $ab+c\ln x$ при некоторых значениях переменных a , b , c . Последовательность необходимых для этого действий приведена на рисунке 21.

Калькулятор «Электроника Б3-34», кроме изученных нами регистров RGX, RGY, RGX₁, имеет также регистры RGZ и RGT. Для нас важно, что регистры RGX, RGY, RGZ и RGT образуют стек с вершиной RGX (рис. 22). Унарные операции выполняются

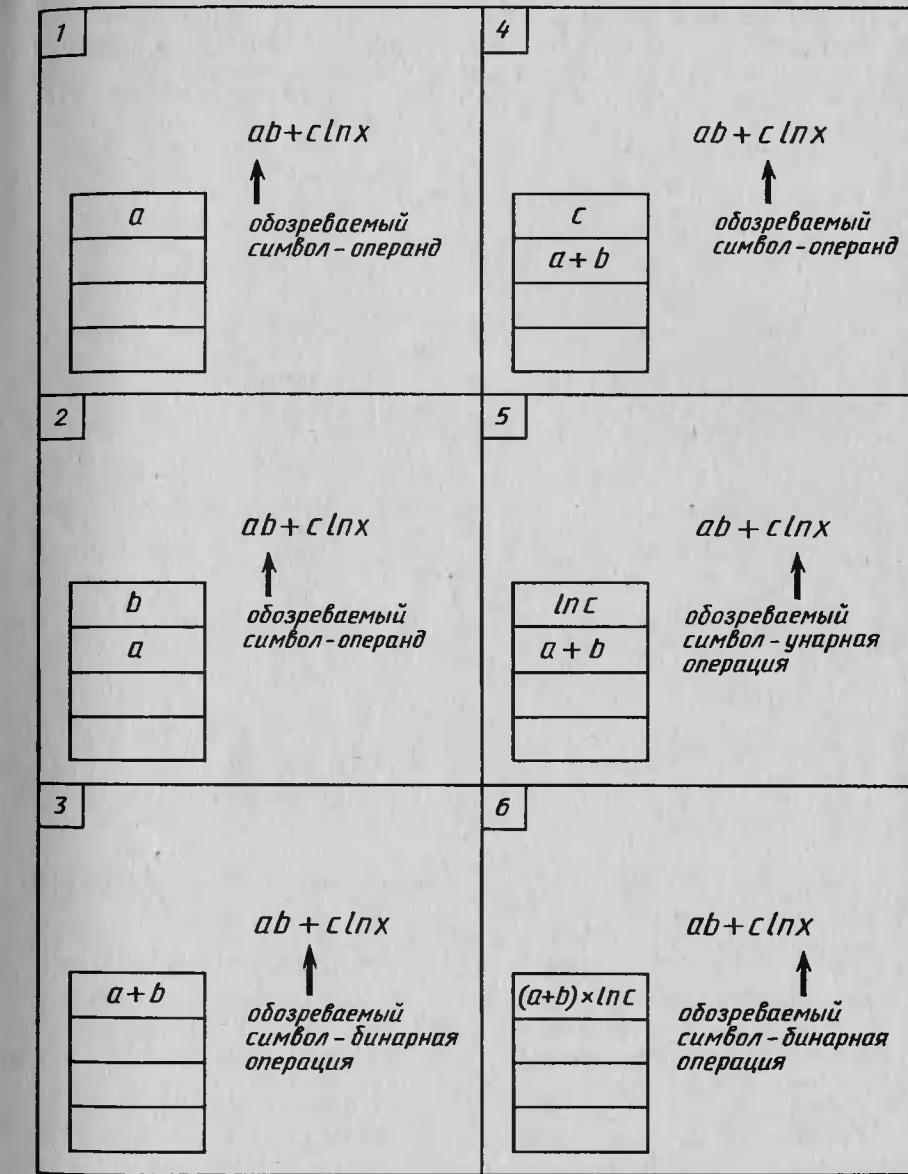


Рис. 21

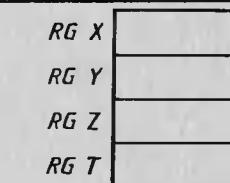
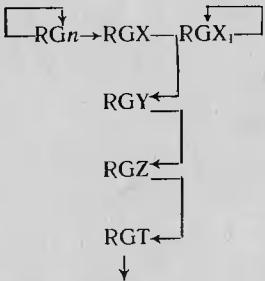


Рис. 22

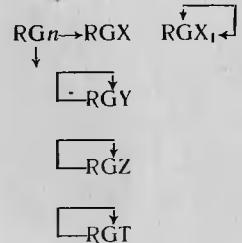
Перемещение чисел в регистрах при выполнении операций

 $f(RGX) \rightarrow RGX \rightarrow RGX_1 \rightarrow$  $\rightarrow RGT$ **F****f***f* — унарная операция $RGX \circ RGY \rightarrow RGX \rightarrow RGX_1 \rightarrow$  $\leftarrow RGZ \leftarrow$ $\leftarrow RGT$ **.***•* — бинарная операция

Перемещение чисел в регистрах при записи и чтении

**ИП****n**

Чтение RGn

**П****n**

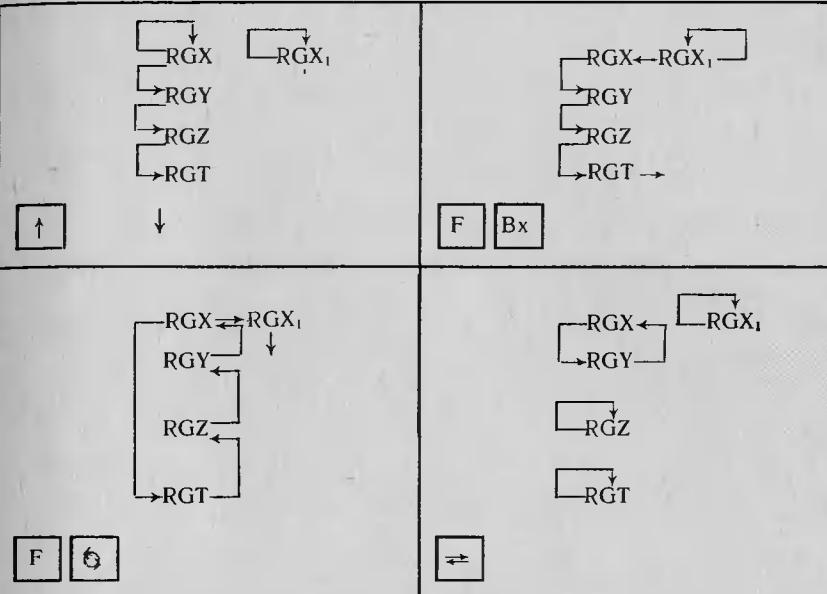
Запись в RGn

над содержимым вершины стека — RGX. Бинарные же операции, как мы помним, выполняются над содержимым регистров RGX и RGY. Перемещение чисел в регистрах стека происходит так, как это предписывает алгоритм вычисления выражения, представленного в польской записи (табл. 22). Перемещение чисел в регистрах при занесении чисел в адресуемые ячейки памяти и чтении содержимого этих ячеек также согласуется с алгоритмом вычисления значения выражения, заданного своей польской записью (табл. 23).

Нам осталось привести схемы перемещения чисел в регистрах при выполнении команд

↑ **F** **Bx** **F** **⌚** **⇄** (табл. 24).

Таблица 23



Команда **↑** используется, когда требуется втолкнуть в стек одно или несколько чисел.

Команда **F Bx**, как мы помним, позволяет прочесть результат предыдущей операции. При этом (табл. 24) числа, находящиеся в регистрах стека, «проталкиваются вниз».

Команду **⌚** удобно использовать для чтения содержимого стековых регистров.

Команда **⇄** производит обмен между содержимым RGX и RGY. Ее используют в случае, когда требуется «поменять местами» операнды некоммутативной бинарной операции.

Поясним, например, одну из приведенных схем. При нажатии на клавиши **ИП** **n** (табл. 23)

содержимое RGZ заносится в RGT,
содержимое RGY заносится в RGZ,
содержимое RGX заносится в RGY,
содержимое RGn заносится в RGX.

Старое содержимое RGT теряется, а содержимое RGX₁ остается прежним.

Теперь наша ближайшая задача — научиться переводить выражения, записанные обычной математической записью, в польскую запись. Несложные выражения можно переводить,

пользуясь интуитивными соображениями. При этом нужно только помнить, что стек устроен по принципу: «последний вошел — первый вышел». Иными словами, операнд, который используется раньше в качестве участника той или иной операции, должен быть помещен в стек позднее. Для перевода более сложных выражений будем пользоваться специальным алгоритмом, к описанию которого мы и приступаем.

Прежде чем переводить выражение в польскую запись, нужно «вытянуть его в цепочку». Например, выражение $\sqrt{a(b+c)}$ следует записать как $\sqrt{(a \times (b + c))}$, а выражение $\frac{a+b}{c+d}$ следует записать как $(a+b) \div (c+d)$. Для возведения в степень придется использовать новый символ \uparrow . Таким образом, выражение x^y записывается как $x \uparrow y$, а x^{yz} записывается как $x \uparrow (y \uparrow z)$. Для описания алгоритма перевода выражений, «вытянутых в цепочку», в польскую запись сопоставим каждому символу a , входящему в это выражение, число $p(a)$, называемое его приоритетом (табл. 25).

Таблица 25

| a | операнд | унарная операция | \uparrow | \div или \times | $+$ или $-$ | (или) | пусто |
|--------|---------|------------------|------------|---------------------|-------------|---------|-------|
| $p(a)$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

Алгоритм перевода выражений будет иметь вид:

Цикл

пока

В x еще не просмотрены все символы.

выполняй

Обозревай следующий символ.

Цикл

пока

Приоритет символа в вершине стека S не превосходит приоритета обозреваемого символа и обозреваемый символ не левая скобка.

выполняй

Вытолкнуть символ из S и затем присоединить его справа к цепочке y , если он не левая скобка.

Все — цикл

Втолкнуть обозреваемый символ в S , если он не правая скобка.

Все — цикл

Цикл

пока

S не пуст

выполняй

Вытолкнуть символ на S и присоединить его справа к цепочке y .

Все — цикл

Перед началом работы алгоритма цепочка y мыслится пустой, т. е. не содержащей символов. Алгоритм использует стек S , который также пуст перед началом работы алгоритма.

Пусть, например, требуется перевести выражение $\frac{a+b}{c+d}$ в польскую запись. «Вытянем» предварительно это выражение в цепочку. Последовательно применяя указания алгоритма, получим польскую запись (рис. 23).

Перейдем, наконец, к составлению программ вычисления арифметических выражений для нашего калькулятора.

Пример 1. Составить программу для вычисления выражения

$$t = \frac{a+b}{c+d}.$$

Решение. Мы уже ранее перевели это выражение в польскую запись:

$$ab + cd + \div$$

Распределим память для исходных данных:

$$\begin{array}{ll} a \rightarrow RGA & b \rightarrow RGB \\ c \rightarrow RGC & d \rightarrow RGD \end{array}$$

По польской записи составляем программу для калькулятора (табл. 26).

Пример 2. Составить программу для вычисления площади треугольника S по формуле Герона: $S = \sqrt{p(p-a)(p-b)(p-c)}$, где $p = \frac{1}{2}(a+b+c)$.

Решение. Запишем каждое из этих выражений в виде цепочки:

$$(a+b+c) \div 2, \\ \sqrt{(p \times (p-a) \times (p-b) \times (p-c))}.$$

Пользуясь приведенным выше алгоритмом, переведем эти выражения в польскую запись. Получим:

$$ab + c + 2 \div, \\ ppa - pb - \times \times pc - \times \sqrt{}$$

| | | | |
|----|------------------------|----------------------|--------|
| 1 | $x = (a+b) \div (c+d)$ | $y =$ | \div |
| 2 | $x = (a+b) \div (c+d)$ | $y =$ | (|
| 3 | $x = (a+b) \div (c+d)$ | $y =$ | a |
| 4 | $x = (a+b) \div (c+d)$ | $y = a$ | + |
| 5 | $x = (a+b) \div (c+d)$ | $y = a$ | b |
| 6 | $x = (a+b) \div (c+d)$ | $y = ab +$ | |
| 7 | $x = (a+b) \div (c+d)$ | $y = ab +$ | \div |
| 8 | $x = (a+b) \div (c+d)$ | $y = ab +$ | (|
| 9 | $x = (a+b) \div (c+d)$ | $y = ab +$ | c |
| 10 | $x = (a+b) \div (c+d)$ | $y = ab + c$ | + |
| 11 | $x = (a+b) \div (c+d)$ | $y = ab + c$ | d |
| 12 | $x = (a+b) \div (c+d)$ | $y = ab + cd + \div$ | |

Распределим память:

$$\begin{array}{ll} a \rightarrow \text{RGA} & c \rightarrow \text{RGC} \\ b \rightarrow \text{RGB} & p \rightarrow \text{RG1} \end{array}$$

Программа для калькулятора представлена в таблице 27.

Рис. 23

| | | | | | | | |
|--|-----|-----|---|----|---|---|--------|
| | 00— | ИП | A | ИП | B | + | |
| | —06 | ИП | C | ИП | D | + | \div |
| | 07 | C/P | | | | | |

Таблица 27

| | | | | | | | |
|--|-----|----------|----------|----|----------------------|--------|---|
| | 00— | ИП | A | ИП | B | + | |
| | —07 | ИП | C | + | 2 | \div | |
| | 08— | П | I | | | | |
| | | ИП | I | ИП | 1 | ИП | A |
| | | — | ИП | I | ИП | B | — |
| | | \times | \times | ИП | 1 | ИП | C |
| | —21 | — | \times | F | $\sqrt{}$ | | |
| | 22 | C/P | | | | | |

(Попробуйте для сравнения составить программу без использования польской записи!)

Упражнения

1. «Расшифруйте» следующие выражения в польской записи:

- а) $abcd + \times +$; б) $ab \ln c - \times \sqrt{d} \div$;
 в) $abc \sin \cos + \times$; г) $a\sqrt{b} \operatorname{tg} + cd - \times \sqrt{}$.

2. Составьте программу для калькулятора, осуществив предварительно перевод в польскую запись:

а) $y = \frac{a+bc}{(b+ac)(c+ab)}$; б) $y = \sqrt{a^2 + b^2 - 2ab \cos C}$.

§ 1.9. УРОК 9. ПОДПРОГРАММЫ

На этом уроке мы познакомимся с очень важным понятием — понятием подпрограммы. Пусть, например, нужно составить программу вычисления выражения $V = \frac{f(a)+f(b)}{2f(a)+f(c)}$, где $y=f(x)=x^2-5x+6$. Можно, конечно, составить программу так, чтобы в трех различных ее местах вычислялось значение функции $f(x)$: в одном месте при $x=a$, в другом — при $x=b$ и в третьем — при $x=c$. Ясно, что при таком подходе в нашей программе трижды будет описана одна и та же совокупность действий, необходимых для вычисления значений функции $y=f(x)$. Возможен, однако, и другой подход. Совокупность действий, вычисляющих $f(x)$, можно расположить в одном месте, т. е. выделить в так называемую подпрограмму, и обращаться к ней всякий раз, когда потребуется. На псевдокоде эта подпрограмма записывается так:

ВЫРАЖЕНИЕ (x, y)

$$y \leftarrow x^2 - 5x + 6$$

возврат

Конец ВЫРАЖЕНИЕ

Здесь **ВЫРАЖЕНИЕ** — имя подпрограммы (имя подпрограммы выбирается произвольно), x, y — формальные параметры, при чем x — входной формальный параметр, а y — выходной формальный параметр. К этой подпрограмме можно обратиться всякий раз, когда потребуется вычислить $f(x)$. После же окончания работы подпрограммы необходимо осуществить переход к основной программе. Такой переход описывается на псевдокоде оператором **ВОЗВРАТ**.

Программу вычисления выражения V на псевдокоде можно описать, например, так:

Начало

Ввод (a, b, c)

Вызов ВЫРАЖЕНИЕ (a, p)

Вызов ВЫРАЖЕНИЕ (b, q)

Вызов ВЫРАЖЕНИЕ (c, r)

$$V \leftarrow \frac{p+q}{2p+r}$$

стоп

Конец

Эта программа содержит три оператора вызова подпрограммы. Рассмотрим, например, оператор:

Вызов ВЫРАЖЕНИЕ (a, p)

Здесь a, p — так называемые фактические параметры. Выполнение этого оператора можно представить себе так:

1. Формальному параметру x подпрограммы присваивается значение входного фактического параметра a .

2. Осуществляется переход к подпрограмме **ВЫРАЖЕНИЕ**, где вычисляется $f(x)$ при $x=a$, которое присваивается формальному выходному параметру y .

3. Значение переменной y присваивается выходному фактическому параметру p и осуществляется переход к основной программе.

Подведем некоторые итоги

1. Часто случается так, что одна и та же совокупность действий выполняется в различных местах программы. В таких ситуациях разумно выполнение этих действий расположить в одном месте и оформить как подпрограмму. В описании подпрограммы указывается ее имя, а также список входных и выходных формальных параметров. В подпрограмме обязателен оператор возврата, который позволяет осуществить переход к основной программе.

2. Для обращения к подпрограмме будем пользоваться оператором вызова подпрограммы, в котором указывается имя вызываемой подпрограммы, а также список фактических входных и выходных параметров.

3. При переходе к подпрограмме ее формальные входные параметры получают значение соответствующих (т. е. занимающих ту же позицию в списке) фактических параметров.

При возврате к основной программе значения формальных выходных параметров присваиваются соответствующим фактическим. Мы сейчас изучим, как составлять подпрограммы и обращаться к ним на нашем калькуляторе.

Прежде всего отметим, что, как и для программы, для подпрограммы выделяется некоторое количество ячеек программной памяти. В подпрограмме помещается команда возврата к вызывающей ее программе (клавиша **В/О**). Вызов подпрограммы на нашем калькуляторе занимает две последовательные ячейки программной памяти. В первой помещается команда

перехода к подпрограмме (клавиша **ПП**), а во второй — адрес первой команды подпрограммы. Обращение к подпрограмме и возврат к вызывающей ее программе на калькуляторе осуществляется так, как показано на рисунке 24.

Нам осталось выяснить один принципиально важный вопрос: как осуществляется обмен информацией между вызывающей программой и подпрограммой? Возможны два подхода, к рассмотрению которых мы и переходим.

Таблица 28



Рис. 24

1. Передача значений

Всякий раз перед переходом к подпрограмме значения входных фактических параметров заносятся в ячейки памяти, выделенные для соответствующих формальных параметров. Результаты же работы подпрограммы, т. е. значения формальных выходных параметров, должны быть занесены в ячейки, выделенные соответствующим фактическим параметрам.

Пусть, например, требуется реализовать на калькуляторе вычисление выражения

$$V = \frac{f(a)+f(b)}{2f(a)+f(c)}, \text{ где } y=f(x)=x^2 - 5x + 6.$$

Псевдокоды программы вычисления V и подпрограммы находятся $f(x)$ нам известны. Займемся распределением памяти.

Для программы

$a \rightarrow \text{RGA}$ $p \rightarrow \text{RG1}$
 $b \rightarrow \text{RGB}$ $q \rightarrow \text{RG2}$
 $c \rightarrow \text{RGC}$ $r \rightarrow \text{RG3}$
 $V \rightarrow \text{RGX}$

Для подпрограммы

$x \rightarrow \text{RG4}$
 $y \rightarrow \text{RGX}$

Программа и подпрограмма будут иметь вид (табл. 28, 29).

2. Передача адресов

При этом подходе в ячейки, выделенные для формальных параметров, заносятся адреса соответствующих фактических. Иными словами, передаются не сами данные, а адреса ячеек, в которых эти данные хранятся. Естественно, что при таком подходе мы будем использовать косвенную адресацию.

Вернемся к нашему примеру вычисления выражения V . Займемся распределением памяти.

Вызывающая программа

| | | | | | | | |
|-----------------------------------|-----|----|---|-----|---|----|---|
| <u>Вызов</u> ВЫРАЖЕНИЕ (a, p) | 00— | ИП | А | П | 4 | | |
| | —04 | ПП | 3 | . | П | 1 | |
| <u>Вызов</u> ВЫРАЖЕНИЕ (b, q) | 05— | ИП | В | П | 4 | | |
| | —09 | ПП | 3 | 0 | П | 2 | |
| <u>Вызов</u> ВЫРАЖЕНИЕ (c, r) | 10— | ИП | С | П | 4 | | |
| | —14 | ПП | 3 | 0 | П | 3 | |
| $V \leftarrow \frac{p+q}{2p+r}$ | 15— | ИП | 1 | ИП | 2 | + | |
| | —24 | ИП | 1 | 2 | × | ИП | 3 |
| | | + | ÷ | С/П | | | |

Таблица 29

| | | | | | | | | |
|---------------------|-----------------------------|-----|-----|---|---|-------|----|---|
| <u>Подпрограмма</u> | $y \leftarrow x^2 - 5x + 6$ | 30— | ИП | 4 | F | x^2 | ИП | 4 |
| | | —37 | 5 | × | — | 6 | + | |
| | | 38 | B/O | | | | | |

Для программы

$a \leftarrow \text{RGA} (10)$ $p \leftarrow \text{RG1}$
 $b \leftarrow \text{RGB} (11)$ $q \leftarrow \text{RG2}$
 $c \leftarrow \text{RGC} (12)$ $r \leftarrow \text{RG3}$
 $V \leftarrow \text{RGX}$

Для подпрограммы

$x \leftarrow \text{RG8}$
 $y \leftarrow \text{RG9}$

Программа и подпрограмма приведены в таблицах 30, 31.

Таблица 30

| Вызывающая программа | | | | | | | | | |
|---------------------------------|-------|----|---|----|---|-----|---|--|--|
| Передача адресов | 00— | 1 | 0 | П | 8 | | | | |
| Вызов ВЫРАЖЕНИЕ (a, p) | —04 | 1 | П | 9 | | | | | |
| | 05—06 | ПП | 3 | 1 | | | | | |
| Передача адресов | 07— | 1 | 1 | П | 8 | | | | |
| | —11 | 2 | П | 9 | | | | | |
| Вызов ВЫРАЖЕНИЕ (b, q) | 12—13 | ПП | 3 | 1 | | | | | |
| Передача адресов | 14— | 1 | 2 | П | 8 | | | | |
| | —18 | 3 | П | 9 | | | | | |
| Вызов ВЫРАЖЕНИЕ (c, r) | 19—20 | ПП | 3 | 1 | | | | | |
| $V \leftarrow \frac{p+q}{2p+r}$ | 21— | ИП | 1 | ИП | 2 | + | . | | |
| | | ИП | 1 | 2 | × | ИП | 3 | | |
| | —30 | | | + | ÷ | С/П | | | |

З а м е ч а н и е.

При работе с одной и той же подпрограммой возможна как передача значений одних параметров, так и передача адресов других. С такими примерами мы будем встречаться в дальнейшем.

Перейдем к рассмотрению примеров.

Пример 1. Составить программу для вычисления выражения

$$S = f(1) + f(2) + \dots + f(n).$$

Решение. Приведем описание алгоритма на псевдокоде:

| Подпрограмма | | | | | | | | | |
|-----------------------------|-----|-----|-------|---|---|---|--|--|--|
| $y \leftarrow x^2 - 5x + 6$ | 31— | К | ИП | 8 | | | | | |
| | | F | x^2 | | | | | | |
| | | К | ИП | 8 | | | | | |
| | | 5 | × | — | 6 | + | | | |
| Возврат | —39 | К | П | 9 | | | | | |
| | 40 | В/О | | | | | | | |

Начало**Ввод (n)**

$S \leftarrow 0$

Цикл n раз (i)**Вызов F(i, y)**

(подпрограмма F присваивает
y значение $f(i)$)

$S \leftarrow S + y$

Все — цикл**Индикация S****Стоп****Конец**

Приведем программу для калькулятора (табл. 32).

Вычислим, например, с помощью этой программы сумму: $1 + 2 + 3 + 4 + \dots + 100$. В этом случае $f(i) = i$.

Подпрограмма приведена в таблице 33.

Занесем программу и подпрограмму в программную память калькулятора. Выполним такие действия:

В/О, набор

| | | |
|---|---|---|
| 1 | 0 | 0 |
|---|---|---|

, пуск.

Примерно через 2,5 минуты на индикаторе загорается ответ: 5050. Вычислим с помощью нашей программы сумму:

$$1^3 + 2^3 + 3^3 + \dots + 100^3.$$

Подпрограмма приведена в таблице 34.

Таблица 32

| | | S | | | | |
|---|-------|-----|----|---|---|---|
| Ввод (<i>n</i>) | 00 | P | 0 | | | |
| <i>S</i> $\leftarrow 0$ | 01—02 | 0 | P | 1 | | |
| Цикл <i>n</i> раз (<i>i</i>) | | | | | | |
| Вызов <i>F</i> (<i>i</i> , <i>y</i>) | 03—04 | ПП | 1 | 2 | | |
| <i>S</i> $\leftarrow S + y$ | 05—07 | ИП | 1 | + | П | 1 |
| Все — цикл | 08—09 | F | L0 | 0 | 3 | |
| Индикация <i>S</i> | 10 | ИП | 1 | | | |
| Стоп | 11 | С/П | | | | |

Таблица 33

| | | F | | | | |
|-------------------------|----|-----|---|--|--|--|
| <i>y</i> $\leftarrow i$ | 12 | ИП | 0 | | | |
| Возврат | 13 | В/О | | | | |

Таблица 34

| | | F | | | | |
|---------------------------|-----|-----|----|---|---|--|
| <i>y</i> $\leftarrow i^3$ | 12— | ИП | 0 | ↑ | × | |
| Возврат | —16 | F | Вх | × | | |
| | 17 | В/О | | | | |

Вычисление продолжается примерно 5 минут. Ответ: 25502500.
Пример 2. Составить программу, которая по десятичной

записи числа $n = abc\dots xyz$ конструировала число $\bar{n} = zyx\dots cba$.
Например, если $n = 152$, то $\bar{n} = 251$.

Решение. Приведем описание алгоритма на псевдокоде.

Начало

Ввод (*n*)

n $\leftarrow 0$

k $\leftarrow n$

Цикл

повторяй

Вызов ОСТАТОК (*k*, 10, *x*)

(Подпрограмма ОСТАТОК (*a*, *b*, *ост*) присваивает переменной *ост* остаток от деления *a* на *b*.)

n $\leftarrow \bar{n} \cdot 10 + x$

k $\leftarrow \frac{k}{10}$

Вызов АНТЬЕ (*k*)

(Эта подпрограмма находит целую часть *k* и присваивает результат опять же переменной *k*.)

до

k $\leqslant 0$

Все — цикл

Индикация *n*

стоп

Конец

Приведем псевдокоды подпрограмм.

ОСТАТОК (*a*, *b*, *ост*)

d $\leftarrow \frac{a}{b}$

Вызов АНТЬЕ (*d*)

ост $\leftarrow a - b \cdot d$

возврат

Конец ОСТАТОК

АНТЬЕ (*y*)

y $\leftarrow [y]$

(переменной *y* присваивается целая часть *y*)

Возврат

Конец АНТЬЕ

Займемся распределением памяти:

| x | n | k | \bar{n} | y | a | b | ост | d |
|-----|-----|-----|-----------|-----|-----|-----|-----|-----|
| RGX | RGX | RGI | RG2 | RGX | RGA | RGB | RGX | RGX |

Приведем программу и подпрограммы для нашего калькулятора (табл. 35, 36, 37).

Таблица 35

| | | | | | | | | |
|---|-------|-----|---|---|---|---|--|--|
| Цикл $k \leftarrow n$ $\bar{n} \leftarrow 0$ | 00 | П | 1 | | | | | |
| | 01—02 | 0 | П | 2 | | | | |
| | 03— | ИП | 1 | П | А | | | |
| | —07 | 1 | 0 | П | В | | | |
| | 08—09 | ПП | 2 | 8 | | | | |
| | 10— | ИП | 2 | 1 | 0 | | | |
| | —15 | × | + | П | 2 | | | |
| | 16—19 | ИП | 1 | 1 | 0 | ÷ | | |
| | 20—22 | ПП | 3 | 9 | П | 1 | | |
| | 23—25 | /— | F | ≥ | 0 | 3 | | |
| | 26 | ИП | 2 | | | | | |
| | 27 | С/П | | | | | | |

Рассмотрим подробнее механизм работы с подпрограммами на нашем калькуляторе. Калькулятор имеет специальный стек возврата, состоящий из 5 ячеек. При обращении к подпрограмме адрес возврата вталкивается в стек, при переходе к вызы-

Таблица 37

| ОСТАТОК | | | | | | | | |
|-------------------------------------|-------|-----|---|----|-----|---|--|--|
| $d \leftarrow \frac{a}{b}$ | 28—30 | ИП | А | ИП | В | ÷ | | |
| Вызов АНТЬЕ (d) | 31—32 | ПП | 3 | 9 | | | | |
| $ост \leftarrow a - b \cdot d$ | 33— | ИП | В | × | /—/ | | | |
| | —37 | ИП | А | + | | | | |
| Возврат | 38 | В/О | | | | | | |

| АНТЬЕ | | | | | | | | |
|--------------------|-----|---|-----|---|---|-------|----|--|
| $y \leftarrow [y]$ | 39— | 1 | , | 8 | F | $1/x$ | — | |
| | | 1 | ВП | 7 | + | F | Вх | |
| | —50 | — | В/О | | | | | |

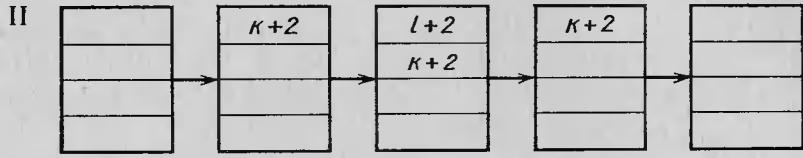
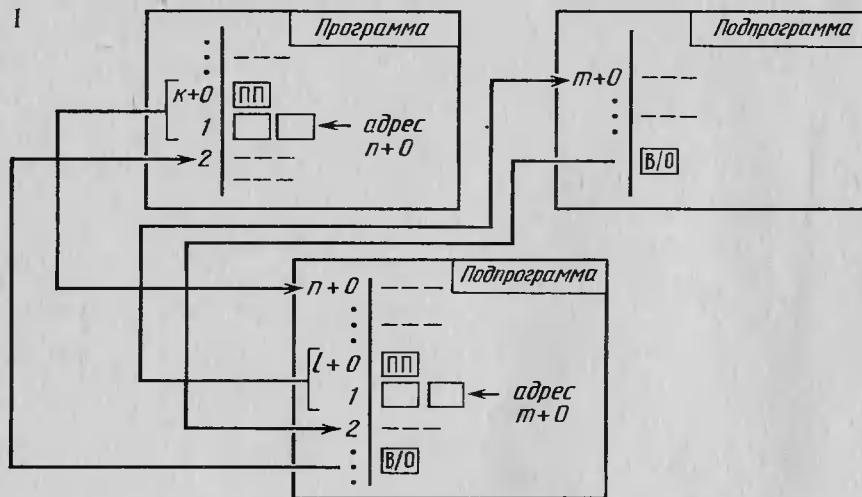
вающей программе (команда **В/О**) содержимое вершины стека выталкивается из него. В качестве примера рассмотрим следующую ситуацию (см. I, с. 72).

Последовательные состояния стека возврата будут иметь вид, изображенный на с. 72, II.

В заключение рассмотрим понятие подпрограммы несколько под иным углом зрения. На нашем калькуляторе нет операции нахождения целой части числа, однако, после того как мы составили подпрограмму АНТЬЕ и занесли ее в программную память калькулятора, можно считать, что такая операция у нас есть! Всякий раз, когда потребуется найти целую часть числа, мы сможем обратиться к этой подпрограмме.

Отметим еще одно чрезвычайно важное свойство подпрограммы: подпрограмма, составленная одним человеком, может быть использована другими людьми в своих программах.

Наконец, подпрограммы играют большую роль при проектировании программ, однако это уже тема нашего следующего урока.



Упражнения

1. Используя программу примера 1 данного урока, вычислите значения выражений:

- $1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{100};$
- $\sin \frac{\pi}{5} + \sin \frac{2\pi}{5} + \dots + \sin \frac{100\pi}{5};$
- $\lg 1 + \lg 2 + \dots + \lg 50;$
- $\sqrt{e} + \sqrt{e^2} + \dots + \sqrt{e^{100}}.$

2. Известно, что показатель a , с которым простое число p входит в разложение на простые множители числа $n!$, находится по формуле

$$a = \left[\frac{n}{p} \right] + \left[\frac{n}{p^2} \right] + \dots + \left[\frac{n}{p^k} \right] + \dots .$$

Проведем описание алгоритма нахождения a на псевдокоде:

```

Начало
Ввод ( $n, p$ )
 $X \leftarrow p$ 
 $a \leftarrow 0$ 
Цикл
  пока
     $\left[ \frac{n}{X} \right] \neq 0$ 
    выполняй
       $a \leftarrow a + \left[ \frac{n}{X} \right]$ 
       $X \leftarrow X \cdot p$ 
  Все — цикл
  Индикация  $a$ 
  стоп
Конец

```

Составьте программу для калькулятора. Выясните, например, на какую наибольшую степень числа $p=7$ делится число $1000000!$.

3. Составьте программу для вычисления выражения

$$V = f(f(\dots f(x)\dots)).$$

Вычислите несколько выражений этого вида.

§ 1.10. УРОК 10. СТРУКТУРНОЕ ПРОГРАММИРОВАНИЕ

На этом уроке мы познакомимся с современным подходом к проектированию программ. Рассматриваемая методика, называемая структурным программированием, была впервые предложена Дейкстрой в 1968 г. и с тех пор получила всеобщее признание.

Как обычно, начнем с рассмотрения примера. В теории чисел известна так называемая гипотеза Гольдбаха, состоящая в предположении:

Всякое четное число, большее или равное 4, можно представить как сумму двух простых чисел.

Эта гипотеза проверена для всех четных чисел, вплоть до числа 1 000 000 000 (см. [31]). Калькулятор «Электроника Б3-34» «работает» с целыми числами, максимальная разрядность которых — 8. Таким образом, гипотеза Гольдбаха для целых чисел, которые представимы на калькуляторе, верна. Интересно составить программу для нашего калькулятора, которая ищет разложение на сумму двух простых чисел данное четное число.

Начнем с общей постановки задачи и затем, постепенно детализируя описание программы, получим программу для нашего калькулятора. Описание программы, которую мы назовем ГОЛЬДБАХ, на «самом высоком уровне» имеет вид:

ГОЛЬДБАХ

Найти разложение четного числа, большего или равного 4, на сумму двух простых чисел.

Конец ГОЛЬДБАХ

1-й шаг детализации

Введем подходящие обозначения. Запишем условие, используя эти обозначения. Выясним, какие у нас входные и выходные переменные.

ГОЛЬДБАХ

Найти разложение четного числа, большего или равного 4, на сумму двух простых чисел

Конец ГОЛЬДБАХ

Ввод (n)

Найти простые p_1 и p_2 , такие, что $n = p_1 + p_2$

Вывод (p_1, p_2)

стоп

Составим табличку переменных и будем всякий раз, когда появятся новые переменные, пополнять ее.

| Переменные | Описание | Распределение памяти |
|------------|-----------------------|----------------------|
| n | Четное число ≥ 4 | |
| p_1 | Простое число | |
| p_2 | Простое число | |
| : | : | |

Такая табличка позволит нам вести строгий учет вводимых переменных. Это делать необходимо, так как память калькулятора ограничена и количество вводимых в рассмотрение переменных может существенно повлиять на разработку программы.

2-й шаг детализации

Для нахождения искомого разложения будем просматривать последовательность простых чисел. Как только разность между данным четным n и очередным простым числом окажется также простым, задачу можно считать решенной.

Два первых шага детализации изображены ниже.

ГОЛЬДБАХ

Найти разложение четного числа, большего или равного 4, на сумму двух простых

Конец ГОЛЬДБАХ

Ввод (n)

Найти простые p_1 и p_2 , такие, что $n = p_1 + p_2$

Вывод (p_1, p_2)

Стоп

Подготовить

Цикл

повторяй

Найти очередное простое p_1

$p_2 \leftarrow n - p_1$

проверить на простоту p_2

до

p_2 простое

Все — цикл

3-й шаг детализации

Итак, нужно уметь находить последовательные простые числа. Это можно сделать так:

положить p_1 равным 1; увеличивать p_1 на 1 до тех пор, пока p_1 не окажется простым.

Пусть теперь переменной p_1 присвоено некоторое число. Для нахождения следующего за ним простого числа будет опять-таки «в цикле» выполнять оператор присваивания $p_1 \leftarrow p_1 + 1$ до тех пор, пока p_1 не окажется простым.

ГОЛЬДБАХ

Найти разложение четного числа, большего или равного 4, на сумму двух простых

Конец ГОЛЬДБАХ

Ввод (n)

Найти простые p_1 и p_2 , такие, что $n = p_1 + p_2$

Вывод (p_1, p_2)

Стоп

Подготовить — $p_1 \leftarrow 1$

Цикл

повторяй

Найти очередное простое p_1

$p_2 \leftarrow n - p_1$

Проверить на простоту p_2

до

p_2 — простое

Все — цикл

Цикл

повторяй

$p_1 \leftarrow p_1 + 1$

Проверить на простоту p_1

до

p_1 — простое

Все — цикл

4-й шаг детализации

Осталось детализировать предложения, связанные с проверкой числа на простоту. Это отдельная интересная задача. Кроме того, такая проверка встречается в двух местах программы. Естественно поэтому выделить проверку числа на простоту в отдельную подпрограмму. Итак, на этом шаге детализации принимаем решение выделить указанную выше проверку в подпрограмму.

ПРОВЕРКА (К, ПРИЗНАК)

Здесь К — входной формальный параметр.

Признак — выходной формальный параметр.

Подпрограмма проверяет, будет ли К ($K > 1$) простым. Если будет, то переменной Признак присваивается значение 0, в противном случае — значение 1. Таким образом, предложения

«проверить p_2 на простоту»
и «проверить p_1 на простоту»
детализируются как

Вызов ПРОВЕРКА (p_2, c)

Вызов ПРОВЕРКА (p_1, d) соответственно.

Предложения же

« p_2 простое» и « p_1 простое»

детализируются соответственно как

« $d = 0$ » и « $c = 0$ ».

Процесс детализации до 4-го шага включительно изображен на с. 77.

На 4-м шаге пополнилась наша табличка переменных, которая теперь имеет вид:

| Переменные | Описание | Распределение памяти |
|------------|---|----------------------|
| n | Четное число ≥ 4 , разложение которого мы ищем | RG0 |
| p_1 | Простые числа. Ищется разложение $n = p_1 + p_2$ | RG1 |
| p_2 | Формальные параметры подпрограммы ПРОВЕРКА, если К простое, то Признак=0, иначе Признак=1 | RG2 |
| K | | RGA |
| Признак | | RGC (RGX) |
| c | Если $p_1(p_2)$ простое, то $d(c)$ присваивается 0, иначе присваивается 1 | RGC (RGX) |
| d | | RGC (RGX) |

ГОЛЬДБАХ

Найти разложение четного числа, большего или равного 4, на сумму двух простых

Конец ГОЛЬДБАХ

Ввод (n)

Найти простые p_1 и p_2 , такие, что $n = p_1 + p_2$

Выход (p_1, p_2)
стоп

Подготовить $p_1 \leftarrow 1$

Цикл

повторяй

Найти
очередное
простое
число p_1
 $p_2 \leftarrow n - p_1$
Проверить
на простоту
число p_2

до

p_1 — простое
Все — цикл

до

p_2 — простое
Все — цикл

Цикл

повторяй

$p_1 \leftarrow p_1 + 1$
Проверить
на простоту p_1

до

p_1 — простое

Все — цикл

Вызов

ПРОВЕРКА

(p_1, d)

$d = 0$

Вызов

ПРОВЕРКА

(p_2, c)

$c = 0$

После 4-го шага детализации программа на псевдокоде принимает вид, показанный на с. 79.

5-й (и он же последний) шаг детализации

Осуществим перевод программы на псевдокод в программу для калькулятора «Электроника Б3-34». Воспользуемся схемами трансляции основных управляющих структур. Заполним программный бланк (табл. 38).

Предложения псевдокода теперь играют роль комментариев. При заполнении бланка операторы псевдокода располагаются напротив соответствующих команд калькулятора.

Перейдем к проектированию подпрограммы ПРОВЕРКА (К, ПРИЗНАК).

Разработка этой подпрограммы методом пошаговой детализации приведена на с. 80. Текст этой подпрограммы на псевдокоде приведен на с. 81. Таблица переменных для этой подпрограммы приведена на с. 80. Программа для калькулятора приведена в таблице 40.

Таблица 38

ПРОГРАММА ГОЛЬДБАХ

КАЛЬКУЛЯТОР
«Электроника Б3-34»ПРОГРАММИСТ Автор
Дата 1.04.81 лист 1 листов 1

КОММЕНТАРИИ

Адреса команд

КОМАНДЫ

Ввод (*n*) $p_1 \leftarrow 1$ ЦиклповторяйЦиклповторяй $p_1 \leftarrow p_1 + 1$ Вызов ПРОВЕРКА
(*p₁, d*)до $d = 0$ Все — цикл $p_2 \leftarrow n - p_1$ Вызов ПРОВЕРКА
(*p₂, c*)до $c = 0$ Все — циклВывод (*p₁, p₂*)Стоп

| | | | | | | |
|-----|---|----|---|---|---|--|
| П | 0 | | | | | |
| 1 | П | 1 | | | | |
| | | | | | | |
| ИП | 1 | 1 | + | П | 1 | |
| П | А | ПП | 2 | 4 | | |
| | | | | | | |
| F | = | 0 | 3 | | | |
| | | | | | | |
| ИП | 0 | ИП | 1 | | | |
| - | П | 2 | | | | |
| П | А | ПП | 2 | 4 | | |
| | | | | | | |
| F | = | 0 | 3 | | | |
| | | | | | | |
| ИП | 2 | ИП | 1 | | | |
| С/П | | | | | | |
| 23 | | | | | | |

ПРОГРАММА ОСТАТОК (*a, b, ост*)КАЛЬКУЛЯТОР
«Электроника Б3-34»ПРОГРАММИСТ Автор
Дата 1.04.81 лист 1 листов 1

КОММЕНТАРИИ

Адреса команд

КОМАНДЫ

58—

| | | | | | | |
|----|---|----|-----|----|---|--|
| ИП | А | ИП | В | ÷ | 1 | |
| , | 8 | F | 1/x | - | 1 | |
| ВП | 7 | + | F | Bx | - | |
| ИП | В | × | /- | | | |
| ИП | А | + | B/O | | | |

 $ост \leftarrow \left[\frac{a}{b} \right]$ $ост \leftarrow a - b \cdot ост$ Возврат

—77

ГОЛЬДБАХ

Ввод (*n*) $p_1 \leftarrow 1$ ЦиклповторяйЦиклповторяй $p_1 \leftarrow p_1 + 1$ Вызов ПРОВЕРКА (*p₁, d*)до $d = 0$ Все — цикл $p_2 \leftarrow n - p_1$ Вызов ПРОВЕРКА (*p₂, c*)до $c = 0$ Все — циклВывод (*p₁, p₂*)стоп

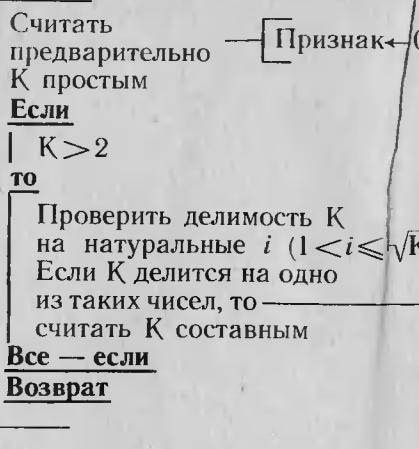
Конец ГОЛЬДБАХ

ПРОВЕРКА (К, Признак)

Проверить на простоту К, ($K > 1$)

Присвоить переменной Признак значение 0, если К — простое, иначе присвоить 1

Конец ПРОВЕРКА



$i \leftarrow 1$
 $LIMIT \leftarrow \sqrt{K}$

Цикл
повторяй

рассмотреть очередное i —> $i \leftarrow i + 1$
 Найти остаток от деления К на i —> Вызов ОСТАТОК (k, i, r)

Если
 | Этот остаток равен 0 —> $r = 0$
 | то
 | | считать К составным —> Признак<=1
 Все — если

до
 | $i > \sqrt{K}$ или К — составное —> $i > LIMIT$, или
 Все — цикл Признак=1

ПРОВЕРКА (К, Признак)

Признак<=0

Если

$K > 2$

то

$i \leftarrow 1$

$LIMIT \leftarrow \sqrt{K}$

Цикл

повторяй

$i \leftarrow i + 1$

Вызов ОСТАТОК (K, i, r)

Если

$r = 0$

то

Признак<=1

Все — если

то

$i > LIMIT$ или Признак=1

Все — цикл

Все — если

возврат

Конец ПРОВЕРКА

Приведем инструкцию по эксплуатации программы ГОЛЬДБАХ.

1. Ввести программу ГОЛЬДБАХ, начиная с адреса 00.
2. Ввести подпрограммы:
 ПРОВЕРКА, начиная с адреса 24;
 ОСТАТОК, начиная с адреса 58.
3. Набор n на клавиатуре, пуск с адреса 00.
4. После окончания работы программы p_1 читается на индикаторе, для прочтения p_2 нажать на клавишу .

Пользуясь нашей программой, получим разложения нескольких четных чисел на сумму двух простых:

$$4=2+2, \quad 52=5+47, \quad 1000=3+997, \quad 150=11+139, \\ 428=7+421, \quad 1030=11+1019, \dots$$

Подведем итоги и сделаем выводы. Мы сконструировали программу разложения четного натурального числа, большего или равного 4, на сумму двух простых чисел. При этом пользовались методикой структурного программирования.

В чем же состоит эта методика?

1. Пошаговая детализация

Мы начали описание программы с общей постановки задачи.

| Переменные | Описание | Распределение памяти |
|------------|--|----------------------|
| K | натуральное число > 1 , проверяемое на простоту | RGA |
| Признак | Переменной признак присваивается 0, если К простое, 1 — в противном случае | RGC (RGX) |
| LIMIT | $LIMIT = \sqrt{K}$ | RGD |
| i | натуральное число | RGB |
| r | остаток от деления К на i | RGX |
| a | Формальные параметры подпрограммы | RGA |
| b | остаток ($a, b, ост$) | RGB |
| ост | | RGX |

Таблица 40

ПРОГРАММА ПРОВЕРКА (К, признак)

| КОММЕНТАРИИ | Адреса команд | КОМАНДЫ | | | | | |
|-----------------------------------|---------------|---------|--------|-----|----------------------|----|---|
| | | 0 | П | С | | | |
| признак←0 | 24—25 | | | | | | |
| <u>Если</u> | 26— | 0 | П | С | | | |
| <u>k > 2</u> | —30 | 2 | ИП | А | — | | |
| <u>то</u> | 31—32 | F | < | 5 | 6 | | |
| <u>i←1</u> | 33—35 | 1 | П | В | | | |
| LIMIT← \sqrt{k} | 36—39 | ИП | А | F | $\sqrt{}$ | П | Д |
| <u>Цикл</u> | 40—41 | ИП | В | 1 | + | П | В |
| <u>повторяй</u> | 42—43 | ПП | 5 | 8 | | | |
| <u>i←i + 1</u> | 44—45 | F | = | 4 | 6 | | |
| <u>Вызов</u> ОСТАТОК (k, i, r) | 46— | 1 | П | С | | | |
| <u>Если</u> | | ИП | Д | ИП | В | — | |
| <u>r = 0</u> | | F | \geq | 5 | 6 | ИП | С |
| <u>то</u> | —55 | 1 | — | F | = | 3 | 6 |
| признак←1 | 56—57 | ИП | С | B/O | | | |
| <u>Все — если</u> | | | | | | | |
| <u>до</u> | | | | | | | |
| <u>i > LIMIT</u> | | | | | | | |
| или | | | | | | | |
| признак = 1 | | | | | | | |
| <u>Все — цикл</u> | | | | | | | |
| <u>Все — если</u> | | | | | | | |
| <u>Возврат</u> | | | | | | | |

Далее, понижая «уровень языка», в несколько шагов трансформировали наше описание в программу для калькулятора.

2. Модульное программирование

Несколько раз по ходу разработки программы мы сталкивались с задачами, решение которых «поручалось» подпрограммам.

Эти подпрограммы можно использовать и при составлении других программ. Например, мы столкнулись с задачей: «Найти остаток от деления двух натуральных чисел» (см. табл. 39). Решая самостоятельно задачи по данному уроку, вы можете использовать эту подпрограмму. Модульный принцип помогает сделать логику программы более обозримой.

3. Структурное кодирование

Разрабатывая программу, мы детализируем предложение «более высокого уровня», используя некоторый стандартный набор «строительных блоков».

Таким образом, блоками являются: альтернатива, цикл-до, цикл-пока, цепочка, выбор, арифметический цикл. Эти строительные блоки будем называть базовыми или основными управляющими структурами. Программы, полученные таким образом, будем называть структурированными. Структурированные программы легко читаются человеком. Их логика легко просматривается.

В дальнейшем, составляя программы, пользуйтесь структурным подходом. Не пожалеете!

Упражнения

I. ТЕОРИЯ ЧИСЕЛ

1. Составьте программу нахождения суммы цифр заданного натурального числа.

2. Составьте программу нахождения суммы делителей данного натурального числа.

3. Составьте программу перевода натурального числа из десятичной в двоичную систему счисления.

4. Натуральное число называется совершенным, если сумма его делителей равна ему самому (в сумму не входит само число). Составьте программу нахождения совершенных чисел и найдите с ее помощью три первых совершенных числа.

5. Составьте программу нахождения НОД и НОК двух натуральных чисел.

6. Число из n цифр называется числом Армстронга, если сумма его цифр, возведенных в n -ю степень, равна самому числу. Например:

$$1634 = 1^4 + 6^4 + 3^4 + 4^4.$$

Составьте программу, находящую все числа Армстронга, состоящие из двух и трех цифр.

7. Два натуральных числа называются дружественными, если сумма делителей одного из них равна другому числу, и наоборот (в сумму делителей не включается само число). Например, числа 220 и 284 дружественные. (Проверьте это!).

Составьте программу, проверяющую, будут ли два данных натуральных числа дружественными.

8. Составьте программу нахождения всех трехзначных чисел $a_1a_2a_3$, таких, что $a_1a_2a_3 = a_1! + a_2! + a_3!$.

9. Гипотеза Симона о факториалае состоит в следующем: только четыре факториала являются произведением трех последовательных целых чисел. Вот два из них:

$$4! = 2 \cdot 3 \cdot 4, \quad 5! = 4 \cdot 5 \cdot 6.$$

Составьте программу, которая помогла бы найти следующее натуральное число, обладающее указанным свойством.

Не смогли бы вы опровергнуть гипотезу Симона?

10. Три натуральных числа a, b, c образуют Пифагорову тройку, если

$$a^2 + b^2 = c^2.$$

Составьте программу, которая помогла бы вам находить Пифагоровы тройки.

11. Составьте программу разложения натурального числа n на простые множители.

12. Составьте программу вычисления a^n наиболее экономным способом. Например:

$$a^7 = (a^2)^3 \cdot a.$$

13. Составьте программу нахождения по заданному натуральному n суммы

$$1! + 2! + \dots + n!.$$

14. Если натуральные числа p и $p+2$ простые, то их называют числами-близнецами. Составьте программу и найдите с ее помощью несколько пар чисел-близнецов.

15. Составьте программу, позволяющую по данному натуральному числу выяснить, представимо ли оно в виде суммы квадратов двух натуральных чисел.

16. Автоморфными называются числа, которые содержатся в последних разрядах их квадрата. Например:

$$5^2 = 25, \quad 25^2 = 625.$$

Составьте программу для нахождения нескольких автоморфных чисел.

17. Простое число Мерсенна — это число, которое может быть представлено в виде

$$2^p - 1, \text{ где } p \text{ тоже простое.}$$

Напишите программу для нахождения ряда таких чисел.

18. Даны натуральные числа x и y , не равные 1.

Составьте программу нахождения натурального числа k , такого, что

$$y^{k-1} < x \leqslant y^k.$$

II. ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА

1. Численные методы решения уравнений (см., например, [34]).

Составьте программу решения уравнений:

- методом половинного деления,
- методом хорд,
- методом касательных,
- комбинированным методом хорд и касательных,
- методом последовательных приближений.

2. Минимизация функций одной переменной (см., например, [13], [54]).

Составьте программу нахождения минимума функций:

- методом половинного деления,
- методом золотого сечения,
- методом Фибоначчи.

3. Решение систем линейных уравнений (см., например, [34]).

a) Составьте описание на псевдокоде алгоритма решения системы линейных уравнений методом Гаусса.

b) Составьте программы решения системы двух линейных уравнений с двумя неизвестными и трех линейных уравнений с тремя неизвестными.

b) Составьте описание на псевдокоде алгоритма решения системы линейных уравнений методом Крамера.

g) Составьте программу решения системы двух линейных уравнений с двумя неизвестными этим методом.

d) Составьте программу нахождения определителя 3-го порядка.

4. Численное интегрирование (см., например, [34]).

Составьте программу нахождения площади криволинейной трапеции:

a) методом прямоугольников, **b)** методом трапеций, **c)** методом Симпсона.

5. Интерполирование (см., например, [34]).

a) Составьте программу нахождения значений интерполяционного многочлена Лагранжа.

b) Составьте описание на псевдокоде алгоритма нахождения значений интерполяционного многочлена Ньютона.

III. ЗАНИМАТЕЛЬНЫЕ ЗАДАЧИ

1. Точка внутри треугольника.

Составьте программу, которая по координатам вершин треугольника и по координатам некоторой точки определяла бы, лежит ли точка внутри треугольника.

2. День недели.

Составьте программу, с помощью которой можно было бы определить день недели по году, месяцу и числу. Определите, в какой день недели будет 1 мая 2000 года, каким днем недели было 9 мая 1945 года.

ИГРА И КАЛЬКУЛЯТОР

3. Расстояние между датами.

Составьте программу, которая находила бы расстояние (в днях) между двумя датами. Определите с ее помощью, сколько прошло дней со дня запуска первого спутника до первого полета человека в космос.

4. Прокладка водопровода.

Докажите, что водопровод любой длины, большей 7 (в метрах), можно построить, используя лишь 3- и 5-метровые трубы.

Составьте программу, которая по длине водопровода находила бы необходимое количество 3- и 5-метровых труб.

5. Задача о сдаче.

В кассе имеются в неограниченном количестве денежные знаки достоинством 1, 3, 5 и 10. Составьте программу автоматического расчета сдачи для любой покупки. Постарайтесь составить программу так, чтобы сдача содержала возможно меньшее число купюр.

6. Расстановка ферзей.

На шахматной доске размером $n \times n$ нужно расставить n ферзей так, чтобы они не угрожали друг другу. Составьте программу, которая находила бы все возможные расстановки.

7. Программирование стека.

Составьте программу работы со стеком, в котором могут храниться неотрицательные целые числа от 0 до $n - 1$.

Операция «втолкнуть» число x в стек интерпретируется как

$$p \leftarrow n \cdot p + x.$$

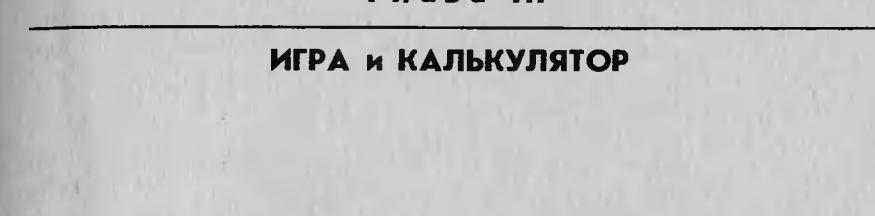
Операция «вытолкнуть» интерпретируется как

$$p \leftarrow \left[\frac{p}{n} \right].$$

Стек пуст тогда и только тогда, когда $p = 1$.

8. Ханойская башня.

В доску вбито три колышка. На первый нанизаны n дисков убывающего диаметра. Требуется, перекладывая диски по одному, расположить их в прежнем порядке на третьем колышке. Ограничение состоит в том, что больший диск запрещено класть на меньший. Составьте программу, которая решает эту задачу.



§ 2.1. КАЛЬКУЛЯТОР ВЫИГРЫВАЕТ

Если бы вы, читатель, находились в 1809 году в Шенбрунне, то могли бы стать свидетелем любопытного зрелища. За шахматной доской, укрепленной на крышке большого ящика, сидел Наполеон Бонапарт. Его противником была большая кукла, одетая турком, укрепленная на ящике. Внутри ящика находился хитроумный механизм. Вместо пальцев кукла имела специальные захваты для передвижения фигур. Автомат был сконструирован Вольфгангом фон Кемпеленом, именовавшим себя придворным советником австрийской императрицы Марии-Терезии, изобретателем и механиком. Великий полководец, не знавший до этого поражений на поле брани, на сей раз был побежден. До конца жизни император был уверен, что играл с «умной» машиной, на самом же деле, как оказалось впоследствии, в ящике находился тщательно замаскированный шахматист небольшого роста.

Мечта об играющем шахматном автомате стала реальностью лишь в XX веке с появлением ЭВМ. В августе 1974 года в Стокгольме впервые был проведен чемпионат мира среди шахматных программ. В нем участвовали 12 программ из 8 стран. Победу одержала советская программа «Каисса».

Возможности программируемых калькуляторов гораздо скромнее возможностей современных ЭВМ, обучить «Электронику» игре в шахматы — задача нереальная, однако более простым играм обучить «Электронику» удается. Последующее изложение посвящено таким играм.

Наша ближайшая задача — уточнить, с какими играми мы будем иметь дело в этой главе. Начнем с рассмотрения одной простой игры.

В одной кучке n спичек, а в другой m . Игроки A и B по очереди берут спички из этих кучек. Каждый имеет право взять любое количество спичек, но только из одной кучки. Побеждает тот, кто заберет последние спички.

Пусть A и B договорились о следующем: игроку A предоставлено право выбора количества спичек в кучках, после чего игрок B выбирает начинаящего игру. Несложно догадаться, что в этом случае игрок B всегда может выиграть. Действительно, пусть игрок B пользуется такой системой указаний:

- Если** в кучках равное число спичек ($n=m$), **то** игрок B предоставляет право первого хода игроку A , игрок B сам выходит первым.
- Если** **иначе** **то** игрок B начинает игру, он своим первым ходом забирает спички из той кучки, в которой их больше, оставляя одинаковое количество спичек в каждой кучке.

3. В дальнейшем B забирает столько же спичек, сколько и A , но из другой кучки и тем самым всякий раз уравнивает количество спичек в кучках.

Ясно, что рассматриваемая игра никогда не закончится вничью, ведь рано или поздно все спички будут забраны. Легко также понять, что если B все время уравнивает количество спичек в кучках, то игрок A никогда не выигрывает. В самом деле, поскольку разрешается брать спички только из одной кучки, то после хода A в кучках всегда будет различное количество спичек. Но тогда A никогда не достигнет позиции, когда в каждой кучке отсутствуют спички (ведь это случай равенства количества спичек в кучках). Итак, придерживаясь правил 1—3, B всегда побеждает. Система указаний 1—3 позволяет игроку B достичь победы вне зависимости от того, как будет действовать A .

Систему указаний, следуя которой один из играющих всегда может выиграть вне зависимости от того, как будет действовать его партнер, будем называть выигрышной стратегией.

В рассмотренной нами игре ситуация с равным количеством спичек в кучках является проигрышной для того, чья очередь делать ход. Если же в кучках неравное количество спичек, то игрок, делавший ход, всегда может выиграть.

Итак, все возникающие по ходу игры позиции либо выигрышные, либо проигрышные для игрока, делающего ход.

Свойство позиции «быть проигрышной для делающего ход» будем называть инвариантом игры.

В данном случае это свойство состоит в следующем: «в кучках равное число спичек». Если обозначить через x текущее количество спичек в первой кучке, а через y — во второй, то инвариантом игры будет соотношение $x=y$.

В рассматриваемой игре значения x и y полностью описывают текущую позицию.

Совокупность параметров, значения которых полностью определяют позицию в игре, будем называть параметрами игры.

Инвариантом, таким образом, является некоторое условие (соотношение), связывающее параметры игры. Именно на знании инварианта и основывается выигрышная стратегия для игрока B . В случае нашей игры B , зная инвариант ($x=y$), неизменно после своего хода оставляет равное количество спичек в кучках, делая позицию проигрышной для A .

Изложим теперь, следуя Трахтенброту (см. [47]), особенности, присущие разобранной выше игре, а также всем играм, которые будем рассматривать в разделе «КАЛЬКУЛЯТОР ВЫИГРЫВАЕТ».

1. В игре участвуют два игрока, ходы которых строго чередуются.
2. Для каждой партии возможен лишь один из двух исходов:
 - выигрывает начинаящий игру;
 - выигрывает его партнер.
3. Каждый ход представляет собой выбор игроком одного из конечно допустимого множества вариантов без участия какого-либо механизма случайного выбора (например, бросания игральной кости).
4. При каждом ходе игрок знает выборы и исходы всех предыдущих ходов данной партии.

В теории игр доказано, что в играх такого типа:

- оба партнера не могут иметь одновременно выигрышной стратегии;
- во всякой игре один из игроков имеет выигрышную стратегию.

Подведем итоги

В играх рассматриваемого типа игрок, для которого существует выигрышная стратегия, всегда побеждает, если ему известен инвариант игры. Игра в этом случае протекает так: этот игрок всегда может своим ходом создать позицию, значения параметров которой связаны определенным известным ему соотношением (инвариант!). С другой стороны, какой бы ход ни сделал его соперник, возникает позиция, значения параметров которой этому соотношению не удовлетворяют.

Перейдем теперь к проектированию на псевдокоде алгоритма игры калькулятора с человеком в игре, обладающие описанными свойствами.

Человеку (как и игроку A в рассматриваемой выше игре) предоставим право выбора значений параметров, описывающих исходную позицию. Калькулятор же (как и игроку B) предоставим право выбора начинаящего игру. Такое распределение ролей делает калькулятор непобедимым в игре с человеком!

Воспользуемся методом пошаговой детализации. Первый шаг детализации представлен на с. 90.

ИМЯ ИГРЫ

Человек с калькулятором; человек выбирает значения параметров игры, описывающих исходную позицию, а калькулятор «решает», кто ходит первым

Конец ИМЯ ИГРЫ

выбор человеком значений параметров игры
выбор калькулятором начидающего игру и формирование позиции, проигрышной для человека (если исходная таковой не является)
 дальнейшее проведение игры по правилам до ее окончания

Детализуем предложения псевдокода, полученного после первого шага:

Выбор человеком значений параметров игры

ввод в память калькулятора значений параметров игры, выбранных человеком

Выбор начидающего игру и формирование позиции, проигрышной для человека (если исходная таковой не является)

Если

исходная позиция благоприятна для начидающего

то

калькулятор «сообщает»:
«Игру начинаю я»
ход калькулятора
формирование новой позиции

иначе

калькулятор «сообщает»:
«Хожу после Вас»

Все — если

Цикл

пока

игра не окончена

выполняй

ход человека
формирование новой позиции
ход калькулятора
формирование новой позиции

Все — цикл

Дальнейшее проведение игры по правилам до ее окончания

Проверка условия «Исходная позиция благоприятна для начидающего» в каждой конкретной игре детализируется после того, как будет найден инвариант игры. Предложения «1-й ход калькулятора» и «Ход калькулятора» для конкретных игр будут детализироваться как последовательность действий, приводящая к позиции, проигрышной для человека. После всякого хода человека или калькулятора возникает новая позиция. Предложение «Формирование новой позиции» состоит в изменении текущих значений параметров игры таким образом, чтобы они описывали вновь возникшую позицию. Проверку условия «игра не окончена» будем поручать арбитру, которым может быть и сам играющий.

Итак, общая схема на псевдокоде алгоритма игры человека с калькулятором имеет вид, изображенный ниже.

Подводя итоги сказанному, отметим, что для составления программ игр рассматриваемого типа следует поступать так:

ИМЯ ИГРЫ

Ввод в память калькулятора значений параметров, описывающих начальную позицию игры.

Анализ исходной позиции

Если

исходная позиция благоприятна для начидающего игру

то

калькулятор сообщает: «Игру начинаю я»
ход калькулятора
формирование новой позиции

иначе

калькулятор сообщает: «Хожу после Вас»

Все — если

Цикл

пока

игра не окончена
(соответствующее решение принимает арбитр после индикации хода калькулятора)

выполняй

ход человека
формирование новой позиции
ход калькулятора
формирование новой позиции

Все — цикл

Конец ИМЯ ИГРЫ

- Найти инвариант игры, т. е. сформулировать свойство позиции «быть проигрышной» в терминах параметров игры.
- Детализировать для конкретной игры предложения общей схемы (с. 91).

3. Осуществить перевод полученного псевдокода на язык программируемого калькулятора. (При этом «работают» схемы трансляции основных управляющих структур, приведенные в первой главе.)

Рассмотрим теперь несколько игр и составим программы для калькулятора, играющего с человеком.

§ 2.2. ИГРА БАШЕ

В 1612 году в городе Лионе вышла в свет книга «Приятные и занимательные задачи». Автор ее — французский математик, поэт и переводчик Баше де Мезириак Гаспар Клод (1581—1638). Испытывая чувство зависти, сообщим читателю также о том, что Баше де Мезириак прекрасно владел французским, итальянским, латинским и греческим языками. Но нас интересуют не его успехи в области изучения языков, а игра, описанная в его книге. Вот один из вариантов этой игры:

На столе выложены n предметов. Двое играющих поочередно забирают несколько предметов, причем заранее договорено, что число забранных за один раз предметов не превышает k ($0 < k \leq n$). Выигрывает тот, кто своим ходом может забрать все оставшиеся предметы.

Читатель может сыграть с кем-либо в игру БАШЕ, забирая поочередно выложенные на столе в ряд предметы, зачеркивая палочки, нарисованные на доске или листе бумаги, передвигая фишку по разбитой на клеточки полосе.

Чтобы калькулятор всегда выигрывал у человека, прежде всего найдем инвариант игры, т. е. условие, связывающее текущие значения параметров игры, при выполнении которого человек проигрывает.

Поиск инварианта удобнее вести, рассуждая «от конца»:



Если калькулятор побеждает, то это означает, что в последнем туре он ходит после человека и забирает все оставшиеся предметы. Для этого нужно добиться такого течения игры, чтобы перед последним туром оставался $(k+1)$ предмет. В самом деле, если перед последним туром останется $k+1$ предмет, то тогда:

- согласно правилам человек не может забрать все оставшиеся предметы;
- после хода человека останется не более k предметов, которые и забирает калькулятор.

Как же достичь калькулятору того, чтобы после предпоследнего тура остался $k+1$ предмет? Легко понять, что необходимо добиться такого течения игры, при котором перед предпоследним туром останется $2 \cdot (k+1)$ предмет. Действительно, если перед предпоследним туром останется $2 \cdot (k+1)$ предмет, то тогда:

- в силу правил игры (забирается не более k предметов) после хода человека не может остататься $k+1$ предмет;
- с другой стороны, так как человек забирает по крайней мере один предмет, на столе останется не более $(k+1)+k$ предметов и калькулятор после своего хода оставляет $k+1$ предмет.

Рассуждая аналогично, устанавливаем, что ситуации с количеством предметов на столе, кратным $k+1$, проигрышны, а позиции, не обладающие этим свойством, выигрышны для того игрока, чья очередь делать ход. (Докажите это методом математической индукции!)

Таким образом, свойство позиции «быть проигрышной» в терминах параметров игры формулируется так:

позиция проигрышна в том и только в том случае, если остаток от деления количества предметов на столе на $k+1$ равен нулю.

Детализируем теперь общую схему (см. с. 91) для случая игры БАШЕ.

Обозначим через x текущее количество предметов на столе.

Ввод в память калькулятора
значений параметров, описывающих
начальную позицию игры

Ввод (n, k)
 $x \leftarrow n$

Анализ исходной позиции

Нахождение остатка
от деления x на $k+1$

Вызов ОСТАТОК ($x, k+1, ост$)

Здесь мы приняли решение о выделении задачи нахождения остатка от деления натурального числа a на натуральное число b в подпрограмму ОСТАТОК. Псевдокод этой подпрограммы имеет вид:

ОСТАТОК ($a, b, ост$)

$$q \leftarrow \left[\frac{a}{b} \right]$$

ост $\leftarrow a - b \cdot q$

Возврат

Конец ОСТАТОК

Исходная позиция благоприятна для начинающего игру

Остаток от деления x на $k+1$ не равен нулю — $ост \neq 0$

Калькулятор сообщает: — Индикация «1»
«Игру начинаю я»

Ход калькулятора — Забрать остаток от деления x на $k+1$

Вызов ХОД КАЛЬКУЛЯТОРА ($x, k+1, ответ$)

На этом этапе детализации нами принято решение о выделении действий по формированию хода калькулятора в подпрограмму ХОД КАЛЬКУЛЯТОРА. Приведем псевдокод этой подпрограммы.

ХОД КАЛЬКУЛЯТОРА ($x, k+1, ответ$)

Вызов ОСТАТОК ($x, k+1, ост$)

ответ \leftarrow ост

Возврат

Конец ХОД КАЛЬКУЛЯТОРА

Формирование новой позиции — Вызов ФНП ($x, ответ$)

Подпрограмма ФНП ($x, ответ$) по текущему количеству предметов на столе x и по значению переменной $ответ$ (количеству предметов, взятых калькулятором или человеком) вычисляет новое значение переменной x (количество предметов, оставшихся на столе). Эта подпрограмма имеет вид:

ФНП ($x, ответ$)

$x \leftarrow x - ответ$

Возврат

Конец ФНП

Калькулятор сообщает: — Индикация «2»

«Хожу после Вас»

Игра не окончена —

Есть предмет на столе (арбитр принимает решение о продолжении или прекращении игры после индикации хода калькулятора)

Ход человека — Ввод (*ответ*)

Итак, программа игры БАШЕ на псевдокоде имеет вид:

БАШЕ

Ввод (n, k)

$x \leftarrow n$

Вызов ОСТАТОК ($x, k+1, ост$)

Если

ост $\neq 0$

то

индикация «1»

Вызов ХОД КАЛЬКУЛЯТОРА ($x, k+1, ответ$)

Вызов ФНП ($x, ответ$)

иначе

индикация «2»

Все — если

Цикл

пока

Есть предметы на столе (арбитр принимает решение о продолжении или прекращении игры после индикации хода калькулятора)

выполняй

Ввод (*ответ*)

Вызов ФНП ($x, ответ$)

Вызов ХОД КАЛЬКУЛЯТОРА ($x, k+1, ответ$)

Вызов ФНП ($x, ответ$)

Все — цикл

Конец БАШЕ

Транслируем теперь псевдокод программы БАШЕ и подпрограмм ОСТАТОК, ХОД КАЛЬКУЛЯТОРА и ФНП на язык «Электроники» (табл. 41—44).

Изложим теперь инструкцию по проведению игры Баше калькулятора с человеком.

Инструкция

1. Арбитр вводит

а) программу БАШЕ (см. табл. 41);

б) подпрограмму ХОД КАЛЬКУЛЯТОРА (см. табл. 43);

в) подпрограмму ФНП (см. табл. 42);

г) подпрограмму ОСТАТОК (см. табл. 44)

в программную память калькулятора.

2. Соперник калькулятора выбирает числа n и k , $0 < k \leq n$. Выбранное количество предметов n выкладывается арбитром на столе. Игрок выполняет такие действия:

Таблица 42

Таблица 41

| ПРОГРАММА БАШЕ | | | | | | | |
|--|--|--|------------|-----|-------|---|---|
| КАЛЬКУЛЯТОР «Электроника Б3-34» | | ПРОГРАММИСТ Автор Дата 10.02.82 лист 1 листов 1 | | | | | |
| Распределение памяти | | | | | | | |
| Переменные | <i>n</i> | <i>x</i> | <i>k+1</i> | ост | ответ | | |
| RG | X | A | B | X | 0 | | |
| КОММЕНТАРИИ | Адреса команд | КОМАНДЫ | | | | | |
| <u>Ввод</u> (<i>n</i> , <i>k</i>) <i>x</i> ← <i>n</i> | 00— | П | A | С/П | ↑ | | |
| | —05 | 1 | + | П | В | | |
| | 06—07 | ПП | 7 | 0 | | | |
| <u>Вызов Остаток</u> (<i>x</i> , <i>k+1</i> , <i>ост</i>) | 08—09 | F | ≠ | 1 | 8 | | |
| | 10—11 | 1 | C/П | | | | |
| | 12—13 | ПП | 3 | 5 | | | |
| <u>Если</u> <i>ост</i> ≠0 <u>то</u> | 14—17 | ПП | 4 | 0 | БП | 2 | 1 |
| | Индикация «1» | | | | | | |
| | <u>Вызов ХК</u> (<i>x</i> , <i>k+1</i> , <i>ответ</i>) | | | | | | |
| <u>иначе</u> | 18—20 | 2 | БП | 2 | 2 | | |
| | Индикация «2» | | | | | | |
| | <u>Все — если</u> | | | | | | |
| <u>Цикл</u> | 21—22 | ИП | 0 | C/П | | | |
| <u>пока</u> есть предметы выполняй | 23 | П | 0 | | | | |
| | <u>Ввод</u> (<i>ответ</i>) | 24—25 | ПП | 4 | 0 | | |
| | <u>Вызов ФНП</u> (<i>x</i> , <i>ответ</i>) | 26—27 | ПП | 3 | 5 | | |
| <u>Вызов ХК</u> (<i>x</i> , <i>k+1</i> , <i>ответ</i>) | 28—29 | ПП | 4 | 0 | | | |
| <u>Вызов ФНП</u> (<i>x</i> , <i>ответ</i>) | 30—31 | БП | 2 | 1 | | | |

| Программа ФНП (<i>x</i> , <i>ответ</i>) | | | | | |
|---|---------------|--|---|--------------|---|
| КАЛЬКУЛЯТОР «Электроника Б3-34» | | ПРОГРАММИСТ Автор Дата 10.02.82 лист 1 листов 1 | | | |
| Распределение памяти | | | | | |
| Переменные | | <i>x</i> | | <i>ответ</i> | |
| RG | | A | | 0 | |
| КОММЕНТАРИИ | Адреса команд | КОМАНДЫ | | | |
| <u>вывод</u> <i>x</i> — ответ <u>Возврат</u> | 40— | ИП | A | ИП | 0 |
| | —44 | П | А | В/О | |

Таблица 43

| ПРОГРАММА ХОД КАЛЬКУЛЯТОРА (<i>x</i> , <i>k+1</i> , <i>ответ</i>) | | | | | |
|---|---------------|--|------------|--------------|--|
| КАЛЬКУЛЯТОР «Электроника Б3-34» | | ПРОГРАММИСТ Автор Дата 10.02.82 лист 1 листов 1 | | | |
| Распределение памяти | | | | | |
| Переменные | | <i>x</i> | <i>k+1</i> | <i>ответ</i> | |
| RG | | A | B | 0 | |
| КОММЕНТАРИИ | Адреса команд | КОМАНДЫ | | | |
| <u>Вызов Остаток</u> (<i>x</i> , <i>k+1</i> , <i>ост</i>) ответ← <i>ост</i> <u>Возврат</u> | 35—36 | ПП | 7 | 0 | |
| | 37 | П | 0 | | |
| | 38 | В/О | | | |

В/О, набор *n*, **С/П**, набор *k*, **С/П**.

3. Калькулятор, «подумав», выбирает начинающего игру. **Если** на индикаторе загорается «1», **то** первым ходит калькулятор.

Если загорается «2», **то** калькулятор ходит вторым.

Таблица 44

ПРОГРАММА ОСТАТОК ($a, b, ост$)КАЛЬКУЛЯТОР
«Электроника Б3-34»ПРОГРАММИСТ Автор
Дата 10.02.82 лист 1 листов 1

Распределение памяти

| Переменные | a | b | $ост$ | | | | |
|--|---------------|---------|-------|----|-----|----|---|
| RG | A | B | X | | | | |
| КОММЕНТАРИИ | Адреса команд | КОМАНДЫ | | | | | |
| $q \leftarrow \left[\frac{a}{b} \right]$ <u>ост</u> $\leftarrow a - b \cdot q$ <u>Возврат</u> | 70— | ИП | A | ИП | В | ÷ | 1 |
| | | , | 8 | F | 1/x | — | 1 |
| | —83 | ВП | 7 | + | F | Вх | — |
| | 84— | ИП | В | × | /—/ | ИП | А |
| | —88 | + | | | | | |
| | 89 | В/О | | | | | |

4. Каждый ход человека осуществляется так:

- человек забирает со стола некоторое количество предметов;
- взятое число предметов заносится в RGX калькулятора (набирается на клавиатуре).

5. Ход калькулятора осуществляется так:

- арбитр нажимает на клавишу **C/P**. Калькулятор, «помняв», высвечивает на индикаторе количество предметов, которые, по «его мнению», следует забрать,
- арбитр убирает это количество предметов со стола.

- Как только все предметы будут забраны, арбитр объявляет победителя.

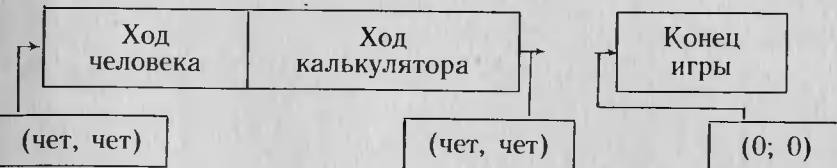
§ 2.3. ИГРА «ДВЕ КУЧКИ КАМНЕЙ»

Имеются две кучки камней. Двоих играющих забирают поочередно камни. Разрешается взять один из любой кучки или по одному камню из обеих кучек. Выигравшим считается тот, после хода которого ни в одной кучке не останется камней.

Для составления программы этой игры прежде всего, как мы уже знаем, необходимо найти инвариант игры. Из правил

игры следует, что игрок своим ходом может менять четность количества камней в одной из кучек или в двух кучках сразу. Но тогда легко понять, что позиции, при которых в каждой кучке четное количество камней, проигрышны для того игрока, чья очередь делать ход.

Тур игры



Действительно, после хода этого игрока возникает одна из ситуаций:

а) В одной из кучек количество камней четно, а в другой — нечетно.

б) В каждой кучке нечетное количество камней.

Забирая в случае а) камень из кучки с нечетным количеством камней и в случае б) по камню из каждой кучки, его соперник опять же создает позицию с четным количеством камней в каждой кучке. Таким образом, после хода нашего игрока будет всегда возникать позиция типа (чет, нечет), (нечет, чет) или (нечет, нечет). Однако тогда ему не удастся забрать последние камни, так как финальная позиция (0,0) — позиция типа (чет, чет).

Таким образом, свойство «быть проигрышной» в терминах параметров игры формулируется так:

Позиция проигрышна тогда и только тогда, когда в каждой кучке четное количество камней.

Детализируем теперь общую схему (см. с. 91) для случая игры «Две кучки камней».

Обозначим через x текущее количество камней в первой, а через y — во второй кучке.

Ввод в память калькулятора значений параметров, описывающих начальную позицию игры

Ввод (m, n)
 $x \leftarrow m$
 $y \leftarrow n$

Анализ исходной позиции

Определить четность числа x .
Определить четность числа y .

Вызов ЧЕТ (x , признак 1)

Вызов ЧЕТ (y , признак 2)

Здесь мы приняли решение о выделении задачи определения четности натурального числа a в подпрограмму ЧЕТ (a , признак). Эта подпрограмма присваивает переменной *признак* значение 0, если число a четное, и значение 1, если a нечетное. Реализация такой подпрограммы может быть различной. Можно было бы воспользоваться тем, что четные числа при делении на 2 дают в остатке 0, а нечетные — 1, т. е. при реализации подпрограммы ЧЕТ можно воспользоваться подпрограммой ОСТАТОК. Мы же воспользуемся тем, что если a четное, то $\cos \pi a = 1$, в случае нечетного $a \cos \pi a = -1$. Таким образом, псевдокод подпрограммы ЧЕТ может быть представлен так:

ЧЕТ (a , признак)

```

Если
     $\cos \pi a \geqslant 0$ 
    то
        признак  $\leftarrow 0$ 
    иначе
        признак  $\leftarrow 1$ 
    Все — если
    Возврат

```

Конец **ЧЕТ**

Замечание. Напомним читателю, что калькулятор должен после своего хода оставлять в каждой кучке четное число камней, т. е. забирать один камень из кучки, если в ней количество камней нечетное. Переменной же *признак* как раз и присваивается количество камней, которое следует забрать из кучки. Этим мы и воспользуемся в дальнейшем.

Исходная позиция благоприятна для начинающего игрока

Хотя бы в одной кучке нечетное количество камней

признак 1 + признак 2 $\neq 0$

Калькулятор сообщает: — Индикация «1»

«Игру начинаю я»

Ход калькулятора

Забрать из кучек такое количество камней, чтобы в каждой кучке количество камней было четным

Вызов ХОД КАЛЬКУЛЯТОРА (x, y , ответ 1, ответ 2)

Здесь мы приняли решение реализовать предложения псевдокода «Ход калькулятора» как подпрограмму. Эта подпрограмма по количествам камней x и y в кучках находит количества камней — ответ 1 и ответ 2, которые «забирает» калькулятор из первой и второй кучек соответственно.

ХОД КАЛЬКУЛЯТОРА (x, y , ответ 1, ответ 2)

Вызов ЧЕТ (x , признак 1)

ответ 1 \leftarrow признак 1

Вызов ЧЕТ (y , признак 2)

ответ 2 \leftarrow признак 2

Возврат

Конец ХОД КАЛЬКУЛЯТОРА

Формирование
новой
позиции

Забрать из первой кучки
количество камней, равное
ответ 1

Забрать из второй кучки
количество камней, равное
ответ 2

Вызов ФНП (x, y , ответ 1, ответ 2)

На этом этапе детализации нами принято решение о выделении действий по формированию новой позиции в подпрограмму ФНП. Псевдокод этой подпрограммы имеет вид:

ФНП (x, y , ответ 1, ответ 2)

$x \leftarrow x -$ ответ 1

$y \leftarrow y -$ ответ 2

Возврат

Конец ФНП

Калькулятор сообщает: — Индикация «2»
«Хожу после вас»

Игра не окончена

Есть камни в кучках (арбитр принимает решение о продолжении или прекращении игры после индикации хода калькулятора)

Ход человека — **Ввод** (ответ 1, ответ 2)

Итак, псевдокод игры «Две кучки камней» имеет вид, представленный ниже. После перевода этого псевдокода и псевдокодов программ ЧЕТ, ФНП и ХОД КАЛЬКУЛЯТОРА получим программу и подпрограммы на языке нашего калькулятора (табл. 45—49).

ДВЕ КУЧКИ КАМНЕЙ

Ввод (m, n)

$x \leftarrow m$

$y \leftarrow n$

Вызов ЧЕТ (x , признак 1)

Вызов ЧЕТ (y , признак 2)

Если

признак 1 + признак 2 $\neq 0$

то

Индикация «1»

Вызов ХОД КАЛЬКУЛЯТОРА ($x, y, \text{ответ } 1, \text{ответ } 2$)

Вызов ФНП ($x, y, \text{ответ } 1, \text{ответ } 2$)

иначе

Индикация «2»

Все — если

Цикл

пока

есть камни в кучках (арбитр принимает решение о продолжении или прекращении игры индикации хода калькулятора)

выполняй

Ввод ($\text{ответ } 1, \text{ответ } 2$)

Вызов ФНП ($x, y, \text{ответ } 1, \text{ответ } 2$)

Вызов ХОД КАЛЬКУЛЯТОРА ($x, y, \text{ответ } 1, \text{ответ } 2$)

Вызов ФНП ($x, y, \text{ответ } 1, \text{ответ } 2$)

Все — цикл

Конец ДВЕ КУЧКИ КАМНЕЙ

Приведем инструкцию по проведению игры «Две кучки камней» калькулятора с человеком.

Инструкция

1. Арбитр вводит в программную память калькулятора программу и подпрограммы (переключатель Г—Р в положении Р).

2. Соперник калькулятора выбирает числа m и n . Выбранное количество камней выкладывается арбитром на стол. Игрок выполняет действия:

B/O, набор m , **C/P**, набор n , **C/P**.

| ПРОГРАММА ДВЕ КУЧКИ КАМНЕЙ | | | | | | | | | |
|---|-----|-----|-----|--|----------------|-----------|---------|---------|---|
| КАЛЬКУЛЯТОР «Электроника Б3-34» | | | | ПРОГРАММИСТ Автор Дата 20.07.82 лист 1 листов 2 | | | | | |
| Распределение памяти | | | | | | | | | |
| Переменные | m | n | x | y | признак 1 | признак 2 | ответ 1 | ответ 2 | |
| RG | X | X | A | B | 3 | X | 1 | 2 | |
| КОММЕНТАРИЙ | | | | Адреса команд | КОМАНДЫ | | | | |
| <u>Ввод</u> (m, n) | | | | 00— | P | A | | | |
| $x \leftarrow m$ | | | | | C/P | | | | |
| $y \leftarrow n$ | | | | —02 | P | V | | | |
| <u>Вызов</u> ЧЕТ (x , признак 1) | | | | 03— | IП | A | ПП | 4 | 0 |
| <u>Вызов</u> ЧЕТ (y , признак 2) | | | | —06 | P | 3 | | | |
| <u>Если</u> | | | | 07—09 | IП | V | ПП | 4 | 0 |
| признак 1 + признак 2 $\neq 0$ | | | | 10— | IП | 3 | + | | |
| <u>то</u> | | | | | F | \neq | | | |
| Индикация «1» | | | | —13 | 2 | 2 | | | |
| <u>Вызов</u> ХК (x, y , ответ 1, ответ 2) | | | | 14—15 | 1 | C/P | | | |
| <u>Вызов</u> ФНП (x, y , ответ 1, ответ 2) | | | | 16—17 | ПП | 5 | 0 | | |
| <u>иначе</u> | | | | 18—21 | ПП | 6 | 0 | BП | 2 |
| Индикация «2» | | | | | | | | | 5 |
| <u>Все — если</u> | | | | 22—24 | 2 | BП | 2 | 7 | |

Таблица 47

Таблица 46

ПРОГРАММА ДВЕ КУЧКИ КАМНЕЙ

КАЛЬКУЛЯТОР
«Электроника Б3-34»ПРОГРАММИСТ Автор
Дата 20.07.82 лист 2 листов 2

КОММЕНТАРИИ

Адреса команд

КОМАНДЫ

Цикл

Пока

Есть камни

выполняй

Ввод (ответ 1,
ответ 2)Вызов ФНП (x, y ,
ответ 1, ответ 2)Вызов ХК (x, y ,
ответ 1, ответ 2)Вызов ФНП (x, y ,
ответ 1, ответ 2)

Все — цикл

| | | | | | | |
|-------|-----|---|-----|---|---|--|
| | | | | | | |
| 25—26 | ИП | 2 | ИП | 1 | | |
| 27 | С/П | | | | | |
| | | | | | | |
| 28—30 | П | 1 | С/П | П | 2 | |
| 31—32 | ПП | 6 | 0 | | | |
| 33—34 | ПП | 5 | 0 | | | |
| 35—36 | ПП | 6 | 0 | | | |
| 37—38 | БП | 2 | 5 | | | |

3. Калькулятор, «подумав», выбирает начинающего игру. Если на индикаторе загорится 1, то первым ходит калькулятор, если же загорится число 2, то калькулятор ходит вторым.

4. Всякий ход человека осуществляется так:

a) человек забирает со стола либо 1 камень из произвольной кучки, либо по одному камню из каждой кучки.

Таким образом, возможны следующие пары (ответ 1, ответ 2):

(1,0), (0,1), (1,1);

б) человек выполняет действия:

Набор ответ 1, С/П, набор ответ 2, С/П

5. Ход калькулятора осуществляется так:

а) арбитр нажимает на клавишу С/П;

б) количество камней, которое, «по мнению калькулятора», следует забрать из первой кучки, загорается на индикаторе;

в) арбитр нажимает на клавишу \leftrightarrow , и на индикатореПРОГРАММА ФНП (x, y , ответ 1, ответ 2)КАЛЬКУЛЯТОР
«Электроника Б3-34»ПРОГРАММИСТ Автор
Дата 20.07.82 лист 1 листов 1

Распределение памяти

Переменные x y ответ 1 ответ 2

RG A B I 2

КОММЕНТАРИИ Адреса команд КОМАНДЫ

 $x \leftarrow x - \text{ответ 1}$

60— ИП А ИП 1

— П А

— 63 ИП В ИП 2

— П В

64 B/O

Возврат

Таблица 48

ПРОГРАММА ХОД КАЛЬКУЛЯТОРА (x, y , ответ 1, ответ 2)КАЛЬКУЛЯТОР
«Электроника Б3-34»ПРОГРАММИСТ Автор
Дата 20.07.82 лист 1 листов 1

Распределение памяти

Переменные x y ответ 1 ответ 2

RG A B I 2

КОММЕНТАРИИ Адреса команд КОМАНДЫ

Вызов ЧЕТ (x , признак 1)

50—52 ИП А ПП 4 0

П 1

53 ИП В ПП 4 0

П 2

58 B/O

ответ 1—признак 1

Вызов ЧЕТ (y , признак 2)

54—56

57

Возврат

Таблица 49

| ПРОГРАММА ЧЕТ (a, признак) | | |
|------------------------------------|--|--------------|
| КАЛЬКУЛЯТОР «Электроника Б3-34» | ПРОГРАММИСТ Автор Дата 20.07.82 лист 1 листов 1 | |
| Распределение памяти | | |
| Переменные | a | признак |
| RG | X | X |
| КОММЕНТАРИЙ | Адреса команд | КОМАНДЫ |
| <u>Если</u> cos la ≥ 0 | 40— | F л X F cos |
| <u>то</u> | —44 | F \geq 4 8 |
| <u>иначе</u> | 45—47 | 0 БП 4 9 |
| <u>признак←1</u> | | |
| <u>Все — если</u> | 48 | 1 |
| <u>Возврат</u> | 49 | B/O |

загорается количество камней, которое, «по мнению калькулятора», следует забрать из второй кучки;

г) арбитр забирает указанное калькулятором количество камней из кучек.

6. Как только все камни будут забраны, арбитр объявляет победителя.

Отметим, что в эту игру можно играть на шахматной доске, передвигая по очереди короля (см. [20] игра «ОДИНОКИЙ КОРОЛЬ»).

§ 2.4. ИГРА «ВПЕРЕД И ВВЕРХ»

В эту игру играют двое на бумаге в клеточку. В одном из узлов выбирается начальная точка O , а в другом, расположеннном правее и выше,— конечная точка K (см. табл. 50). Пусть отрезок OA содержит m клеток, а отрезок OB — n клеток. Затем выбираются два натуральных числа Δx и Δy , причем $\Delta x \leq m$, $\Delta y \leq n$. Начинающий игру из точки O либо проводит вправо горизонтальную линию, длина которой не превосходит

Δx , либо — вверх вертикальную линию, длина которой не превосходит Δy . Второй игрок также проводит вправо горизонтальную или вверх вертикальную линию, длины которых не превосходят соответственно Δx и Δy , причем начало этой линии совпадает с концом линии, проведенной первым игроком, и т. д. Выигрывает тот, кто первым достигнет точки K .

Для того чтобы хорошо усвоить правила игры, предлагаем в качестве примера партию двух начинающих игроков (табл. 50).

По договоренности здесь $m=9$, $n=6$,
 $\Delta x=8$, $\Delta y=5$.

Первый игрок необдуманно провел линию Oa . Второй игрок провел линию ab , поставив тем самым первого игрока в безвыходное положение. Далее последовало bc , и второй игрок ходом CK одержал победу.

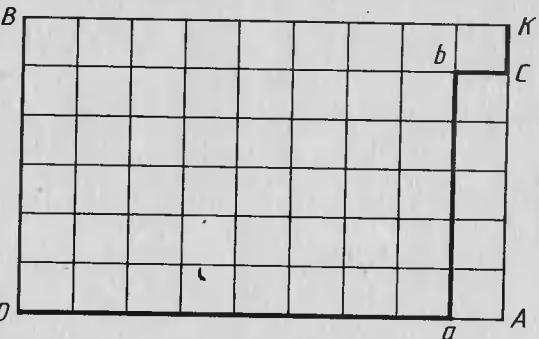


Таблица 50

Рассмотрим внимательно таблицу 51. Легко видеть, что если конец ломаной линии находится в одном из выделенных узлов, то, продолжая линию вверх или вправо на дозволенное число клеток, мы никогда не попадем в выделенный узел. С другой стороны, если конец линии находится не в выделенном узле, то всегда можно продолжить линию так, чтобы ее конец совпал с одним из выделенных узлов. Так как точка K является выделенным узлом, то игрок, сумевший первым продолжить линию так, чтобы ее конец совпал с одним из выделенных узлов, может обеспечить себе выигрыш.

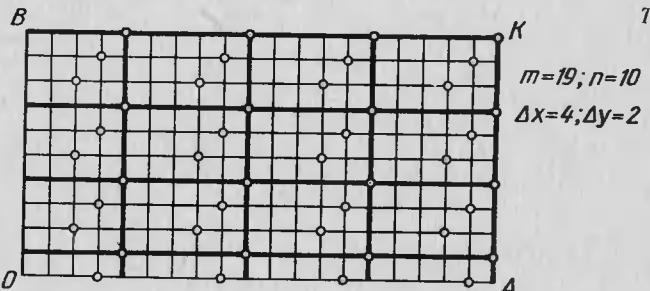


Таблица 51

$m=10$; $n=10$
 $\Delta x=4$; $\Delta y=2$

Нетрудно заметить, что все выделенные узлы, и только они, обладают свойством:

остаток от деления расстояния от узла до AK на $\Delta x + 1$ равен остатку от деления расстояния от узла до BK на $\Delta y + 1$.

Для того чтобы сформулировать инвариант игры в общем случае, введем обозначения:

x — текущее расстояние от конца ломаной линии до правого борта (до AK , см. табл. 51),

y — текущее расстояние от конца ломаной линии до верхнего борта (до BK , см. табл. 51).

Несложно убедиться в том, что свойство позиции «быть проигрышной» в терминах параметров игры можно сформулировать так:

позиция проигрышна в том и только в том случае, когда остаток от деления x на $\Delta x + 1$ равен остатку от деления y на $\Delta y + 1$.

Детализируем теперь общую схему (см. с. 91) для случая игры «ВПЕРЕД И ВВЕРХ».

Ввод в память калькулятора значений параметров, описывающих начальную позицию игры

Ввод ($m, n, \Delta x, \Delta y$)
 $x \leftarrow m$
 $y \leftarrow n$

Анализ исходной позиции

Найти остаток от деления x на $\Delta x + 1$

Вызов ОСТАТОК ($x, \Delta x + 1, ост 1$)

Найти остаток от деления y на $\Delta y + 1$

Вызов ОСТАТОК ($y, \Delta y + 1, ост 2$)

Здесь мы ввели в рассмотрение подпрограмму ОСТАТОК (см. табл. 44).

Исходная позиция благоприятна для начинающего игру

$ост 1 \neq ост 2$

Калькулятор сообщает: «Игру начинаю я»

Индикация «1»

Калькулятор сообщает: «Хожу после Вас»

Индикация «2»

Ход КАЛЬКУЛЯТОРА

Продлить ломаную так, чтобы после хода остатки от деления x на $\Delta x + 1$ и y на $\Delta y + 1$ были равны

Вызов ХОД КАЛЬКУЛЯТОРА ($x, y, вперед, вверх$)

Спроектируем теперь подпрограмму ХОД КАЛЬКУЛЯТОРА. Эта подпрограмма по текущим значениям x и y формирует ход калькулятора, т. е. вычисляет количество клеток, на которое необходимо продлить ломаную вправо (переменная «вперед»), и количество клеток, на которое необходимо продлить ломаную вверх (переменная «вверх»).

ХОД КАЛЬКУЛЯТОРА ($x, y, вперед, вверх$)

$вперед \leftarrow 0$

$вверх \leftarrow 0$

Вызов ОСТАТОК ($x, \Delta x + 1, ост 1$)

Вызов ОСТАТОК ($y, \Delta y + 1, ост 2$)

Разность $\leftarrow ост 2 - ост 1$

Если

Разность < 0

то

$вперед \leftarrow -\text{Разность}$

иначе

$вверх \leftarrow \text{Разность}$

Все — если

Возврат

Конец ХОД КАЛЬКУЛЯТОРА

Формирование новой позиции

Изменить текущие значения x и y в соответствии с ходом калькулятора или человека

Вызов ФНП ($x, y, вперед, вверх$)

Псевдокод подпрограммы ФНП имеет вид:
ФНП ($x, y, вперед, вверх$)

$x \leftarrow y$ — вперед

$y \leftarrow y$ — вверх

Возврат

Конец ФНП

Ход человека

Ввод ($вперед, вверх$)

Игра не окончена

Не достигнут правый верхний узел (арбитр принимает решение о продолжении или прекращении игры после индикации хода калькулятора)

Осуществляя теперь «сборку», получим псевдокод игры «ВПЕРЕД и ВВЕРХ» (с. 110).

После трансляции этого псевдокода и псевдокодов подпрограмм получим программу и подпрограммы (табл. 52—55) на языке калькулятора «Электроника Б3-34».

ВПЕРЕД И ВВЕРХ

Ввод ($m, n, \Delta x, \Delta y$)

$x \leftarrow m$

$y \leftarrow n$

Вызов ОСТАТОК ($x, \Delta x + 1, ост 1$)

Вызов ОСТАТОК ($y, \Delta y + 1, ост 2$)

Если

$ост 1 \neq ост 2$

то

Индикация «1»

Вызов ХОД КАЛЬКУЛЯТОРА ($x, y, вперед, вверх$)

Вызов ФНП ($x, y, вперед, вверх$)

Иначе

Индикация «2»

Все — если

Цикл

пока

не достигнут правый верхний узел
(арбитр принимает решение о прекращении
или продолжении игры после индикации
хода калькулятора)

выполняй

Ввод ($вперед, вверх$)

Вызов ФНП ($x, y, вперед, вверх$)

Вызов ХОД КАЛЬКУЛЯТОРА ($x, y, вперед, вверх$)

Вызов ФНП ($x, y, вперед, вверх$)

Все — цикл

Конец ВПЕРЕД И ВВЕРХ

Изложим инструкцию по проведению игры «ВПЕРЕД И ВВЕРХ» калькулятора с человеком.

Инструкция

1. Арбитр вводит:

- программу ВПЕРЕД И ВВЕРХ (см. табл. 52, 53);
- подпрограмму ОСТАТОК (см. табл. 44);
- подпрограмму ХОД КАЛЬКУЛЯТОРА (см. табл. 54);
- подпрограмму ФНП (см. табл. 55)

в программную память калькулятора.

| ПРОГРАММА «ВПЕРЕД И ВВЕРХ» | | | | | | | | | |
|--|-----|---------------|----------------|----------------|---------|-----------------|----------|---------|---|
| КАЛЬКУЛЯТОР «Электроника Б3-34» | | ПРОГРАММИСТ | | Автор | | лист 1 листов 2 | | | |
| Распределение памяти | | | | | | | | | |
| Переменные | x | y | $\Delta x + 1$ | $\Delta y + 1$ | $ост 1$ | $ост 2$ | $вперед$ | $вверх$ | |
| RG | 1 | 2 | 3 | 4 | 5 | X | 6 | 7 | |
| КОММЕНТАРИИ | | Адреса команд | | КОМАНДЫ | | | | | |
| <u>Вызов</u> ОСТАТОК ($x, \Delta x + 1, ост 1$) | | 00— | | ИП | 1 | П | А | ИП | 3 |
| <u>Вызов</u> ОСТАТОК ($y, \Delta y + 1, ост 2$) | | —06 | | П | В | ПП | 7 | 0 | |
| <u>Если</u> | | 07— | | П | 5 | | | | |
| $ост 1 \neq ост 2$ | | —12 | | ИП | 2 | П | А | ИП | 4 |
| <u>то</u> | | Индикация «1» | | П | В | ПП | 7 | 0 | |
| <u>иначе</u> | | 13— | | ИП | 5 | — | | | |
| <u>Вызов</u> ХК ($x, y, вперед, вверх$) | | —16 | | F | = | 2 | 5 | | |
| <u>Вызов</u> ФНП ($x, y, вперед, вверх$) | | | | | | | | | |
| <u>иначе</u> | | 17—18 | | 1 | С/П | | | | |
| <u>Все — если</u> | | 19—20 | | ПП | 4 | 2 | | | |
| | | 21—24 | | ПП | 9 | 0 | БП | 2 | 8 |
| | | 25—27 | | 2 | БП | 3 | 0 | | |
| | | | | | | | | | |

Таблица 54

Таблица 53

ПРОГРАММА «ВПЕРЕД И ВВЕРХ»

КАЛЬКУЛЯТОР
«Электроника Б3-34»ПРОГРАММИСТ Автор
Дата 5.06.83 лист 2 листов 2

КОММЕНТАРИИ

Адреса
команд

КОМАНДЫ

ЦиклпокаНе достигнут
правый верхний узел
выполняйВвод (вперед, вверх)
Вызов ФНП (x, y,
вперед, вверх)Вызов ХК (x, y,
вперед, вверх)Вызов ФНП (x, y,
вперед, вверх)Все — цикл

28—29

| | | | | | | | | | |
|-----|---|-----|---|---|--|--|--|--|--|
| ИП | 7 | ИП | 6 | | | | | | |
| С/П | | | | | | | | | |
| П | 6 | С/П | П | 7 | | | | | |
| ПП | 9 | 0 | | | | | | | |
| ПП | 4 | 2 | | | | | | | |
| ПП | 9 | 0 | | | | | | | |
| БП | 2 | 8 | | | | | | | |

30

31—33

34—35

36—37

38—39

40—41

2. Соперник калькулятора выбирает параметры игры m , n , Δx , Δy , затем арбитр на бумаге в клеточку рисует прямоугольник размером $m \times n$.

Игрок выполняет действия:

Набор m ,

| | |
|---|---|
| П | 1 |
|---|---|

,Набор n ,

| | |
|---|---|
| П | 2 |
|---|---|

,Набор $\Delta x + 1$,

| | |
|---|---|
| П | 3 |
|---|---|

,Набор $\Delta y + 1$,

| | |
|---|---|
| П | 4 |
|---|---|

.

3. Арбитр или игрок нажимает клавишу

| |
|-----|
| С/П |
|-----|

; калькулятор, «подумав», выбирает начинающего игру.

Если на индикаторе загорается 1,
то первым ходит калькулятор.
Если же загорается 2,
то калькулятор ходит вторым.

ПРОГРАММА ХОД КАЛЬКУЛЯТОРА (x, y, вперед, вверх)КАЛЬКУЛЯТОР
«Электроника Б3-34»ПРОГРАММИСТ Автор
Дата 5.06.83 Лист 1 Листов 1

Распределение памяти

Переменные x y $\Delta x + 1$ $\Delta y + 1$ ост 1 ост 2 вперед вверх разность

RG 1 2 3 4 5 X 6 7 X

КОММЕНТАРИИ

Адреса
команд

КОМАНДЫ

вперед↔0

42—

вверх↔0

—44

Вызов ОСТАТОК
(x, $\Delta x + 1$, ост 1)

45—

Вызов ОСТАТОК
(y, $\Delta y + 1$, ост 2)разность↔ост 2—ост 1

—51

Еслиразность < 0

52—

вперед↔—разность

—57

иначевверх↔разность

58—59

Все — еслиВозврат

60—61

62—65

66

67

| | | | | | | | | | |
|-----|---|----|----|----|---|--|--|--|--|
| 0 | П | 6 | | | | | | | |
| П | 7 | | | | | | | | |
| ИП | 1 | П | А | ИП | 3 | | | | |
| П | В | ПП | 7 | 0 | | | | | |
| П | 5 | | | | | | | | |
| ИП | 2 | П | А | ИП | 4 | | | | |
| П | В | ПП | 7 | 0 | | | | | |
| | | | | | | | | | |
| ИП | 5 | — | | | | | | | |
| | | | | | | | | | |
| F | < | 6 | 6 | | | | | | |
| /—/ | П | 6 | БП | 6 | 7 | | | | |
| П | 7 | | | | | | | | |
| В/О | | | | | | | | | |

Таблица 55

ПРОГРАММА ФНП (*x, y, вперед, вверх*)КАЛЬКУЛЯТОР
«Электроника Б3-34»ПРОГРАММИСТ Автор
Дата 5.06.83 лист 1 листов 1

Распределение памяти

| Переменные | <i>x</i> | <i>y</i> | <i>вперед</i> | | <i>вверх</i> | |
|------------------------------|---------------|----------|---------------|----|--------------|---|
| RG | 1 | 2 | 6 | | 7 | |
| КОММЕНТАРИИ | Адреса команд | | КОМАНДЫ | | | |
| <i>x</i> — <i>x</i> — вперед | 90— | ИП | 1 | ИП | 6 | — |
| | —93 | П | 1 | | | |
| | 94— | ИП | 2 | ИП | 7 | — |
| | —97 | П | 2 | | | |
| | 98 | В/О | | | | |

4. Всякий ход человека осуществляется так: игрок продлевает линию в соответствии с правилами игры. Пусть «вперед» — количество клеток, на которые продлевается линия по горизонтали, а «вверх» — количество клеток, на которое продлевается линия по вертикали. В соответствии с правилами игры либо значение «вперед», либо значение «вверх» равно 0.

Далее игрок выполняет действия:

Набор «вперед», С/П, набор «вверх», С/П.

5. Всякий ход калькулятора осуществляется нажатием арбитром клавиши С/П. То количество, на которое, «по мнению калькулятора», следует удлинить линию по горизонтали, загорается на дисплее. Далее арбитр нажимает на клавишу ➡, и на индикаторе загорается количество клеток, на которое, «по мнению калькулятора», следует удлинить линию по вертикали. Арбитр производит удлинение линии в соответствии с «указаниями» калькулятора.

6. Как только будет достигнут верхний правый узел, арбитр объявляет победителя.

Замечание. Отметим, что игра «ВПЕРЕД И ВВЕРХ» является обобщением игры, приведенной в 2.1.

§ 2.5. ИГРА Р. ГАСКЕЛА И М. ВАНИГАНА

Познакомим читателя с еще одной игрой, заимствованной нами из книги крупнейшего специалиста по программированию Д. Кнута «искусство программирования для ЭВМ» (т. 1, с. 120). Вот ее описание.

Играют двое. Имеется n фишек; первый игрок берет любое количество фишек; единственное, чего он не может, это взять их все. Затем он передает ход партнеру, и далее они ходят по очереди, причем каждый игрок забирает одну или больше фишек, но не больше, чем удвоенное количество фишек, взятых предыдущим игроком. Выигрывает тот, кому достанется последняя фишка.

Существует простой алгоритм, зная который игрок, имеющий право выбора начинающего игру, всегда обеспечит себе победу. Однако найти и доказать правильность этого алгоритма значительно сложнее, чем во всех рассмотренных выше играх.

В дальнейшем нам понадобится приведенная ниже теорема.

ТЕОРЕМА

Всякое натуральное число n можно, и притом единственным образом, представить в виде суммы

$$\text{такой, что } \begin{aligned} n = & F_1 + F_2 + \dots + F_{k-1} + F_k \\ F_i > & 2F_{i+1} \quad (i=1, 2, \dots, k-1), \end{aligned} \quad (1)$$

где F_1, F_2, \dots, F_k — числа Фибоначчи.

Напомним читателю, что числами Фибоначчи называются члены последовательности Фибоначчи:

$$0, 1, 1, 2, 3, 5, 8, 13, \dots$$

Каждый член этой последовательности, начиная с третьего, равен сумме двух предшествующих.

Доказательство сформулированной выше теоремы мы здесь не приводим. Опишем, однако, алгоритм получения такого разложения.

В качестве F_1 выбирается наибольшее из чисел Фибоначчи, не превосходящее n . Если $n - F_1 \neq 0$, то в качестве F_2 выбирают наибольшее из чисел Фибоначчи, не превосходящее $n - F_1$. Далее, если $n - (F_1 + F_2) \neq 0$, то в качестве F_3 выбирают наибольшее из чисел Фибоначчи, не превосходящее $n - (F_1 + F_2)$, и т. д., до тех пор пока разность между n и суммой уже полученных слагаемых F_1, F_2, \dots, F_k не окажется равной нулю.

Пользуясь этим алгоритмом, получим, например, такие разложения:

$$5=5;$$

$$11=8+3;$$

$$30=21+8+1;$$

$$120=89+21+8+2.$$

Введем в рассмотрение функцию f , заданную на множестве неотрицательных целых чисел.

Положим

$$f(0)=+\infty,$$

$$f(n)=F_k,$$

т. е. натуральному n ставится в соответствие наименьшее из чисел Фибоначчи в разложении (1) этого числа.

Например:

$$f(5)=5;$$

$$f(11)=3;$$

$$f(30)=1;$$

$$f(120)=2.$$

Продолжим теперь рассмотрение игры Гаскела и Ванигана. всякая позиция в этой игре описывается парой $(x, \Delta x)$, где x — количество оставшихся фишек, а Δx — максимальное количество фишек, которое разрешается взять игроку, делающему ход.

Позиции, для которых $f(x) \leq \Delta x$, будем называть позициями первого типа, а позиции, в которых $f(x) > \Delta x$, — позициями второго типа. Можно доказать, что:

1. Если $(x, \Delta x)$ — позиция первого типа,
то, забирая $f(x)$ фишек, мы получаем позицию второго типа.
2. Любой ход в позиции второго типа приводит к позиции первого типа.

Кроме того, легко видеть, что финальная позиция — позиция второго типа. Действительно, $f(0)=+\infty$ и, следовательно, $f(0) > \Delta x$.

Таким образом, любой ход в позиции второго типа не является выигрышным.

Итак, свойство позиции «быть проигрышной» в терминах параметров игры формулируется так:

Позиция проигрышна в том и только в том случае, если $f(x) > \Delta x$.

Детализируем теперь общую схему (см. с. 91) для случая игры Гаскела и Ванигана.

Ввод в память калькулятора значений параметров, описывающих начальную позицию игры

Ввод
 $x \leftarrow n$
 $\Delta x \leftarrow x - 1$

Анализ исходной позиции

Найти $y = f(x)$

Вызов ХВОСТ (x, y)

Здесь мы принимаем решение о выделении нахождения значения функции $y = f(x)$ в подпрограмму ХВОСТ.

Спроектируем эту подпрограмму методом пошаговой детализации:

ХВОСТ (x, y)

Вычислить $f(x)$

Найденное значение присвоить y

Возврат

Конец ХВОСТ

$R \leftarrow x$

Цикл

повторяй

Найти наибольший член y последовательности Фибоначчи, не превосходящий R
 $R \leftarrow R - y$

до

$R = 0$

Все — цикл

Подпрограмма ХВОСТ в свою очередь обращается к подпрограмме ФИБ, разработка которой имеет вид:

ФИБ (R, y)

Найти наибольший член y последовательности Фибоначчи, не превосходящий R

Возврат

Конец ФИБ

Подготовить

$y \leftarrow 1$

$F \leftarrow 1$

Цикл

повторяй

Найти очередной член последовательности Фибоначчи F и предшествующий ему член y
 $V \leftarrow F$
 $F \leftarrow F + y$
 $y \leftarrow V$

до

$R < F$

Все — цикл

Ниже приведены эти программы в «собранном виде».

ХВОСТ (x, y)

$R \leftarrow x$

Цикл

повторяй

Вызов ФИБ (R, y)

$R \leftarrow R - y$

до

$R = 0$

Все — цикл

Возврат

Конец ХВОСТ

ФИБ (R, y)

$y \leftarrow 1$

$F \leftarrow 1$

Цикл

повторяй

$V \leftarrow F$

$F \leftarrow F + y$

$y \leftarrow V$

до

$R < F$

Все — цикл

Возврат

Конец ФИБ

Исходная позиция
благоприятна для начинающего

$\Delta x \geq y$

Калькулятор сообщает:
«Игру начинаю я»

Индикация «1»

ХОД КАЛЬКУЛЯТОРА

Вызов ХОД КАЛЬКУЛЯТОРА ($x, \Delta x, \text{ответ}$)

Подпрограмма ХОД КАЛЬКУЛЯТОРА по заданному текущему количеству фишек x и по количеству фишек Δx , которое разрешается забрать, находит количество фишек, которое «забирает» калькулятор. Найденное значение присваивается переменной «ответ».

Спроектируем эту подпрограмму.

ХОД КАЛЬКУЛЯТОРА ($x, \Delta x, \text{ответ}$)

«Забрать» такое количество фишек, чтобы возникла позиция, проигрышная для человека

Возврат

Конец ХОД КАЛЬКУЛЯТОРА

Если

правила разрешают забрать все фишки

$\Delta x \geq x$

то

забрать все оставшиеся фишки

$\text{ответ} \leftarrow x$

иначе

забрать $f(x)$ фишек

Вызов ХВОСТ (x, y)
 $\text{ответ} \leftarrow y$

Все — если

Осуществляя «сборку», получим:

ХОД КАЛЬКУЛЯТОРА ($x, \Delta x, \text{ответ}$)

Если

$\Delta x \geq x$

то

$\text{ответ} \leftarrow x$

иначе

Вызов ХВОСТ (x, y)
 $\text{ответ} \leftarrow y$

Все — если

Возврат

Конец ХОД КАЛЬКУЛЯТОРА

Формирование
новой позиции

Вызов ФНП ($x, \Delta x, \text{ответ}$)

Программа ФНП по текущему количеству фишек x и по значению переменной «ответ» (количество фишек, взятых калькулятором или человеком) вычисляет новое значение переменной x (количество фишек, оставшихся после хода) и значение переменной Δx (количество фишек, не более которого можно будет взять следующим ходом).

Псевдокод этой подпрограммы имеет вид:

ФНП (x , Δx , ответ)

$x \leftarrow x$ — ответ

$\Delta x \leftarrow 2$ · ответ

Возврат

Конец ФНП

Калькулятор сообщает:
«Хожу после Вас»

Игра не
окончена

Еще есть не забранные фишки (арбитр принимает решение о продолжении или прекращении игры после индикации хода калькулятора)

Ход человека

Ввод (ответ)

Итак, программа игры ГАСКЕЛА И ВАНИГАНА на псевдокоде имеет вид, приведенный на с. 122. После перевода этого псевдокода и псевдокодов подпрограмм ХВОСТ, ФИБ, ХОД КАЛЬКУЛЯТОРА и ФНП получим программу и подпрограммы (см. табл. 56, 57, 58, 59, 60, 61) на языке калькулятора «Электроника Б3-34».

Изложим теперь инструкцию по проведению игры ГАСКЕЛА И ВАНИГАНА калькулятора с человеком.

Инструкция

1. Арбитр вводит:

- программу ГАСКЕЛА И ВАНИГАНА (табл. 56, 57);
- подпрограмму ХВОСТ (табл. 59);
- подпрограмму ФИБ (табл. 60);
- подпрограмму ХОД КАЛЬКУЛЯТОРА (табл. 58);
- подпрограмму ФНП (табл. 61)

в программную память калькулятора.

2. Соперник калькулятора выбирает число n . Выбранное количество фишек n выкладывается арбитром на столе. Игрок выполняет такие действия:

B/O, набор n , **C/P**.

3. Калькулятор, «подумав», выбирает начинаяющего игру. Если на индикаторе загорается 1, то первым ходит калькулятор. Если же загорается 2, то калькулятор ходит вторым.

4. Всякий ход человека осуществляется так:

а) человек забирает со стола некоторое количество фишек в соответствии с правилами игры;

| ПРОГРАММА ГАСКЕЛ И ВАНИГАН | | | | | |
|---|---------------|------------------------------------|------------|-------------------|--------|
| КАЛЬКУЛЯТОР «Электроника Б3-34» | | ПРОГРАММИСТ Дата 9.05.83 лист 1 | | Автор листов 2 | |
| Распределение памяти | | | | | |
| Переменные | n | x | Δx | y | ответ |
| RG | X | 0 | 1 | 2 | 3 |
| КОММЕНТАРИИ | Адреса команд | Команды | | | |
| <u>Ввод</u> (n) | 00— | P | 0 | | |
| $x \leftarrow n$ | | 1 | — | | |
| $\Delta x \leftarrow x - 1$ | | | | | |
| <u>Вызов</u> ХВОСТ (x , y) | —03 | P | 1 | | |
| <u>Если</u> | 04—05 | ПП | 5 | 0 | |
| $\Delta x \geq y$ | 06— | ИП | 1 | ИП | 2 |
| <u>то</u> | | | | | — |
| Индикация «1» | —10 | F | \geq | 1 | 9 |
| <u>Вызов</u> ХК (x , Δx , ответ) | 11—12 | 1 | С/П | | |
| <u>Вызов</u> ФНП (x , Δx , ответ) | 13—14 | ПП | 3 | 5 | |
| <u>иначе</u> | | | | | |
| Индикация «2» | 15—18 | ПП | 7 | 7 | БП 2 2 |
| <u>Все — если</u> | 19—21 | 2 | БП | 2 | 3 |

б) взятое число фишек заносится в RGX калькулятора (набирается на клавиатуре).

5. Ход калькулятора осуществляется так:

а) арбитр нажимает на клавишу **C/P**, калькулятор, «подумав», высвечивает на индикаторе количество фишек, которое, «по его мнению», следует забрать;

б) арбитр убирает это количество фишек со стола.

6. Как только все фишки будут забраны, арбитр объявляет победителя. Им, как мы знаем, будет калькулятор!

ГАСКЕЛ И ВАНИГАН

Ввод (*n*) $x \leftarrow n$ $\Delta x \leftarrow x - 1$ **Вызов** ХВОСТ (*x, y*)**Если** $\Delta x \geq y$ **то**

Индикация «1»

Вызов ХОД КАЛЬКУЛЯТОРА (*x, Δx, ответ*)**Вызов** ФНП (*x, Δx, ответ*)**иначе**

Индикация «2»

Все — если**Цикл****пока**

Есть еще не забранные фишки
(арбитр принимает решение о продолжении
или прекращении игры после индикации
хода калькулятора)

выполняй**Ввод** (*ответ*)**Вызов** ФНП (*x, Δx, ответ*)**Вызов** ХОД КАЛЬКУЛЯТОРА (*x, Δx, ответ*)**Вызов** ФНП (*x, Δx, ответ*)**Все — цикл****Конец** ГАСКЕЛ И ВАНИГАН**§ 2.6. КАЛЬКУЛЯТОР И СЛУЧАЙ**

В играх, которые мы рассматривали до сих пор, существовала выигрышная стратегия для одного из игроков. Случай в таких играх не имеет никакого значения. Существуют игры, исход которых зависит только от случая, а знания и мастерство не нужны. Примером таких игр могут быть игра в спортлото, идеальная рулетка, кости и т. д. В них применяются так называемые датчики, или генераторы случайных чисел. Например, в качестве генератора случайных чисел 1, 2, 3, 4, 5, 6 можно использовать игральную кость. Генераторами могут быть рулетки, урны, таблицы и т. д.

Для получения случайных чисел с помощью ЭВМ используются различные алгоритмы генерации так называемых

| ПРОГРАММА ГАСКЕЛ И ВАНИГАН | | | | | |
|--|---------------|---------|----------|-----|--|
| КАЛЬКУЛЯТОР «Электроника Б3-34» | ПРОГРАММИСТ | | Автор | | |
| | Дата 9.05.83 | лист 2 | листов 2 | | |
| КОММЕНТАРИИ | Адреса команд | КОМАНДЫ | | | |
| Цикл пока Есть еще не забранные фишки выполняй Ввод (<i>ответ</i>) Вызов ФНП (<i>x, Δx, ответ</i>) Вызов ХК (<i>x, Δx, ответ</i>) Вызов ФНП (<i>x, Δx, ответ</i>) Все — цикл | 22—23 | ИП | 3 | С/П | |
| | 24 | П | 3 | | |
| | 25—26 | ПП | 7 | 7 | |
| | 27—28 | ПП | 3 | 5 | |
| | 29—30 | ПП | 7 | 7 | |
| | 31—32 | БП | 2 | 2 | |
| | | | | | |

псевдослучайных чисел. Один из первых таких алгоритмов, принадлежащий Джону фон Нейману (1946 г.), известен как метод середины квадрата. Опишем его. Пусть задано четырехзначное число $x_1 = 9568$. Возведем его в квадрат, $x_1^2 = 91\ 546\ 624$. Возьмем четыре средние цифры этого числа и обозначим $x_2 = 5466$. Далее $x_2^2 = 29\ 8771\ 56$. Положим $x_3 = 8771$. Аналогично $x_3^2 = 76\ 9304\ 41$, а $x_4 = 9304$ и т. д. В качестве псевдослучайных чисел Джон фон Нейман предлагал использовать значения 0,9568; 0,5466; 0,8771; 0,9304; ...

Предлагаем читателю в качестве упражнения составить для калькулятора «Электроника Б3-34» программу получения псевдослучайных чисел этим методом. Мы же для этой цели воспользуемся методом Н. М. Коробова, который легко программируется на калькуляторе. Изложим суть этого метода. Положим $x_0 = 1$. Последовательность случайных чисел x_n задается соотношением

$$x_{i+1} = kx_i \pmod{p},$$

где запись $a \pmod{b}$ означает остаток от деления a на b .

Число p рекомендуется выбирать простым, вида $2p_1 + 1$, где p_1 также простое. По выбранному p подбирают число k , близкое к $\frac{1}{2}p$, из чисел вида $p - 3^m$, где m натуральное. Приведем рекомендованные пары чисел

ПРОГРАММА ХВОСТ (x, y)

| | | |
|---|-----------------------------|--------------------------|
| КАЛЬКУЛЯТОР «Электроника Б3-34» | ПРОГРАММИСТ Дата 9.05.83 | Автор лист 1 листов 1 |
| Распределение памяти | | |
| Переменные | x | y |
| RG | 0 | 2 |
| КОММЕНТАРИИ | Адреса команд | КОМАНДЫ |
| <u>Если</u> $Ax \geq x$ <u>то</u> <u>ответ</u> $\leftarrow x$ <u>иначе</u> <u>Вызов</u> ХВОСТ (x, y) <u>ответ</u> $\leftarrow y$ <u>Все — если</u> <u>Возврат</u> | 35— | ИП 1 ИП 0 — |
| | —39 | F \geq 4 4 |
| | 40— | ИП 0 П 3 |
| | —43 | БП 4 7 |
| | 44—45 | ПП 5 0 |
| | 46—47 | ИП 2 П 3 |
| | 48 | В/О |
| | | |
| $R \leftarrow x$ | | |
| <u>Цикл</u> <u>повторяй</u> <u>Вызов</u> ФИБ (x, y) $R \leftarrow R - y$ <u>до</u> $R = 0$ <u>Все — цикл</u> <u>Возврат</u> | | |
| 50—51 | | |
| 52—53 | | |
| 54— | | |
| —57 | | |
| 58—59 | | |
| 60 | | |

ДАТЧИК (x, y, k, p, A, B) $r \leftarrow kx$ $r_1 \leftarrow \frac{r}{p}$ Вызов АНТЬЕ (r_1) $x \leftarrow r - p \cdot r_1$
 $y \leftarrow (B - A) \cdot \frac{x}{p} + A + 0.5$ Вызов АНТЬЕ (y)ВозвратКонец ДАТЧИК

Подпрограмма для калькулятора приведена в таблице 62. Эта подпрограмма по предыдущему случайному числу x из промежутка $[1; p-1]$ генерирует новое случайное число x из этого промежутка по методу Н. М. Коробова, при этом также генерируется целое случайное число из промежутка $[A; B]$ по формуле (*).

$$(p; k): (2027; 1298), (5087; 2900),$$

$$(10\ 079; 3518), (20\ 183; 3622); (40\ 127; 20\ 444), (50\ 147; 30\ 464), \\(100\ 103; 41\ 054), (220\ 919; 161\ 870).$$

Все члены последовательности x_i — целые числа из промежутка $[1; p-1]$. С помощью последовательности x_i можно построить последовательность y_i случайных чисел, принадлежащих промежутку $[A; B]$:

$$y_i = [(B - A) \cdot \frac{x_i}{p} + A + 0.5] \quad (*)$$

(символ $[\alpha]$ означает целую часть числа α).

В дальнейшем нам понадобится подпрограмма, генерирующая целые псевдослучайные числа, равномерно распределенные на промежутке $[A; B]$, где A и B — целые числа. Описание этой подпрограммы на псевдокоде имеет вид:

Таблица 61

Таблица 60

| ПРОГРАММА ФИБ (R, y) | | | | | |
|---|-------|-----------------------------|---------|--------------------------|--|
| КАЛЬКУЛЯТОР «Электроника Б3-34» | | ПРОГРАММИСТ Дата 9.05.83 | | Автор лист 1 листов 1 | |
| Распределение памяти | | | | | |
| Переменные | y | R | F | V | |
| RG | 2 | 4 | 5 | 6 | |
| КОММЕНТАРИИ | | Адреса команд | КОМАНДЫ | | |
| $y \leftarrow 1$ $F \leftarrow 1$ <u>Цикл</u> повторяй $V \leftarrow F$ $F \leftarrow F + y$ $y \leftarrow V$ до $R < F$ Все — цикл Возврат | 61— | 1 П 2 | | | |
| | —63 | П 5 | | | |
| | 64—65 | ИП 5 П 6 | | | |
| | 66—68 | ИП 2 + П 5 | | | |
| | 69—70 | ИП 6 П 2 | | | |
| | 71— | ИП 4 ИП 5 — | | | |
| | —75 | F < 6 4 | | | |
| | 76 | B/O | | | |

Подпрограмма ДАТЧИК содержит обращение к подпрограмме АНТЬЕ (a), которая меняет значение a на его целую часть. Подпрограмма АНТЬЕ (a) на языке калькулятора приведена в таблице 63.

Рассмотрим тестовый пример.

Пусть $p=2027$, $k=1298$, $A=-1$, $B=1$, $x_0=1$. Для тестирования подпрограммы заменим 72-ю команду (B/O) на С/П БП 50 и введем программу ДАТЧИК и подпрограмму АНТЬЕ в программную память калькулятора. Выполним занесения:

$$\begin{aligned} A + 0,5 &= -0,5 \rightarrow RGA; \\ B - A &= 2 \rightarrow RGB; \\ p &= 2027 \rightarrow RGD; \end{aligned}$$

| ПРОГРАММА ФНП ($x, \Delta x, \text{ответ}$) | | | | | |
|---|-------|-----------------------------|----------------|--------------------------|--|
| КАЛЬКУЛЯТОР «Электроника Б3-34» | | ПРОГРАММИСТ Дата 9.05.83 | | Автор лист 1 листов 1 | |
| Распределение памяти | | | | | |
| Переменные | x | Δx | ответ | | |
| RG | 0 | 1 | 3 | | |
| КОММЕНТАРИИ | | Адреса команд | КОМАНДЫ | | |
| $x \leftarrow x - \text{ответ}$ $\Delta x \leftarrow 2 \cdot \text{ответ}$ <u>Возврат</u> | 77— | ИП 0 ИП 3 — | | | |
| | —80 | П 0 | | | |
| | 81—84 | ИП 3 2 × П 1 | | | |
| | 85 | B/O | | | |

Таблица 63

| ПРОГРАММА АНТЬЕ (a) | | | | | |
|------------------------------------|-----|---|---------|---|--|
| КАЛЬКУЛЯТОР «Электроника Б3-34» | | ПРОГРАММИСТ Дата 9.06.81 лист 1 листов 1 | | | |
| Распределение памяти | | | | | |
| Переменные | a | | — | | |
| RG | | RGX | | — | |
| КОММЕНТАРИИ | | Адреса команд | КОМАНДЫ | | |
| $a \leftarrow [a]$ | 75— | ↑ 1 , 8 F 1/x | | | |
| | | — 1 ВП 7 + | | | |
| | —86 | F Bx — | | | |
| | 87 | B/O | | | |
| Возврат | | | | | |

§ 2.7. ИГРА «ПОТОПИ КОРАБЛЬ»

| ПРОГРАММА ДАТЧИК (x, y, k, p, A, B) | | | | | | | | |
|---|-------|--|---------|-----|---------|-------|-------|-----|
| КАЛЬКУЛЯТОР «Электроника Б3-34» | | ПРОГРАММИСТ Автор Дата 10.08.82 лист 1 листов 1 | | | | | | |
| Распределение памяти | | | | | | | | |
| Переменные | x | y | k | p | $A+0,5$ | $B-A$ | r_1 | r |
| RG | 9 | X | C | D | A | B | X | 8 |
| КОММЕНТАРИИ | | Адреса команд | КОМАНДЫ | | | | | |
| $r \leftarrow kx$ $r_1 \leftarrow \frac{r}{p}$ <u>Вызов АНТЬЕ (r_1)</u> $x \leftarrow r - p \cdot r_1$ $y \leftarrow (B - A) \frac{x}{p} + A + 0,5$ <u>Вызов АНТЬЕ (y)</u> <u>Возврат</u> | 50— | ИП | 9 | ИП | С | | | |
| | —53 | Х | П | 8 | | | | |
| | 54—55 | ИП | D | ÷ | | | | |
| | 56—57 | ПП | 7 | 5 | | | | |
| | 58— | ИП | D | × | /—/ | ИП | 8 | |
| | —63 | + | П | 9 | | | | |
| | 64— | ИП | D | ÷ | ИП | В | | |
| | —69 | Х | ИП | A | + | | | |
| | 70—71 | ПП | 7 | 5 | | | | |
| | 72 | В/О | | | | | | |

$$k = 1298 \rightarrow RGC; \\ x_0 = 1 \rightarrow RG9.$$

В этом случае получим такую последовательность псевдослучайных чисел:

$$0, -1, -1, 0, 1, -1, 0, 1, 0, 0, 1, 1, \dots$$

Время генерации одного числа примерно 13 с.

Перейдем к рассмотрению игр, исход которых зависит как от случая, так и от умения игроков.

Игра происходит на координатной плоскости. Один из двух играющих «помещает» свой корабль в одну из точек с целочисленными координатами, сообщает расстояние от начала координат до корабля второму игроку и затем передвигает корабль в произвольную соседнюю точку с целочисленными координатами или оставляет корабль на месте. Затем второй игрок «производит выстрел», т. е. указывает первому игроку координаты точки, в которую он посыпает снаряд. Если расстояние от корабля до снаряда меньше величины r — радиуса поражения, то корабль объявляется потопленным (величина r устанавливается по договоренности), в противном случае первый игрок сообщает, каково расстояние между снарядом и кораблем, и передвигает корабль в соседнюю точку с целочисленными координатами или оставляет корабль на месте до тех пор, пока второй игрок не «поразит корабль». Затем игроки меняются ролями. Победителем считается тот, кто быстрее потопит корабль противника. Играть в эту игру гораздо интереснее с помощью калькулятора, который имитирует движение корабля и сообщает всякий раз расстояние от снаряда до корабля. Составим программы, введя обозначения: (x_k, y_k) — координаты корабля, (x_c, y_c) — координаты точки падения снаряда, r — радиус поражения, d — расстояние от снаряда до корабля, s — счетчик количества выстрелов, $ERROR$ — константа «корабль потоплен».

Опишем на псевдокоде алгоритм этой игры.

ПОТОПИ КОРАБЛЬ

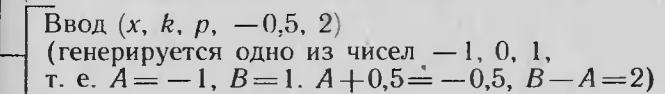
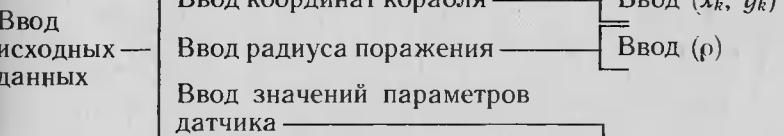
Ввод исходных данных

Вычисление расстояния от корабля
до начала координат

Проведение стрельбы по кораблю
до его уничтожения

Конец ПОТОПИ КОРАБЛЬ

Детализируем теперь предложения псевдокода.



Вычислить расстояние
от корабля до начала
координат

Вызов ЕВКЛИД ($x_k, y_k, 0, 0, d$)

Имитация движения корабля

Здесь нами принято решение о выделении в подпрограмму нахождение расстояния между двумя точками плоскости. Эта подпрограмма на псевдокоде имеет вид:

ЕВКЛИД (x_1, y_1, x_2, y_2, d)

$$d \leftarrow \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Возврат

Конец ЕВКЛИД

Проведение
стрельбы
по кораблю
до его
уничтожения

Подготовить

Цикл
повторять

Сообщение человеку
информации для выстрела
Произвести выстрел по кораблю
Имитация движения корабля
Вычисление расстояния между снарядом
и кораблем
до
корабль потоплен

Все — цикл

Завершить

Подготовить

Установить счетчик
количества выстрелов
в исходное положение

$$\text{сч} \leftarrow 0$$

Сообщение человеку
информации для
выстрела

индикация d

Провести
выстрел по
кораблю

Ввод координат
падения снаряда
Увеличение счетчика
выстрелов

Ввод (x_c, y_c)

$$\text{сч} \leftarrow \text{сч} + 1$$

Получить новую
абсциссу

Получить случайное
целое y из $[-1; 1]$
 $x_k \leftarrow x_k + y$

Вызов ДАТЧИК
($x, y, k, p, -1, 1$)

Получить новую
ординату

Получить случайное
целое y из $[-1; 1]$.
 $y_k \leftarrow y_k + y$

Вызов ДАТЧИК
($x, y, k, p, -1, 1$)

Вычисление расстояния
между кораблями

Вызов ЕВКЛИД (x_k, y_k, x_c, y_c, d)

Корабль —————
потоплен —————
Расстояние от снаряда
до корабля меньше
радиуса поражения ————— $d < \rho$

Завершить —————
Ввод сообщения
«корабль
потоплен» —————
индикация
ERROR

Таким образом, псевдокод игры имеет вид, изображенный на с. 134.

Программа для калькулятора, полученная по этому псевдокоду, приведена в таблицах 64 и 65.

Изложим инструкцию проведения игры «ПОТОПИ КОРАБЛЬ» с помощью калькулятора. Предварительно заметим, что игроку, который ведет стрельбу, рекомендуется отображать «ход событий» на координатной плоскости, вычерченной на бумаге в клеточку.

Инструкция

1. Кто-либо из играющих осуществляет ввод.
а) программы ПОТОПИ КОРАБЛЬ (см. табл. 64);
б) подпрограммы ЕВКЛИД (см. табл. 65);
в) подпрограммы АНТЬЕ (см. табл. 63)

в программную память калькулятора.

2. Выбираем значение ρ — радиус поражения.

3. Первый игрок осуществляет ввод данных для датчика случайных чисел и ввод величины p :

Таблица 65

ПРОГРАММА ЕВКЛИД

КАЛЬКУЛЯТОР
«Электроника Б3-34»ПРОГРАММИСТ Автор
Дата 11.03.83 лист 1 листов 1

Распределение памяти

| Переменные | x_1 | y_1 | x_2 | y_2 | d |
|--|---------------|--------------|-------|-------|-----|
| RG | 1 | 2 | 3 | 4 | 5 |
| КОММЕНТАРИИ | Адреса команд | КОМАНДЫ | | | |
| | | | | | |
| сч←0 | 00—01 | 0 П 0 | | | |
| <u>Вызов ЕВКЛИД</u> ($x_k, y_k, 0, 0, d$) | 02— | П 3 П 4 | | | |
| <u>Цикл</u> | —05 | ПП 3 3 | | | |
| <u>повторяй</u> | 06—07 | ИП 5 С/П | | | |
| <u>Индикация d</u> | 08— | П 3 С/П | | | |
| <u>Ввод</u> (x_c, y_c) | —10 | П 4 | | | |
| сч←сч + 1 | 11—14 | ИП 0 1 + П 0 | | | |
| <u>Вызов ДАТЧИК</u> ($x, y, k, p, -1, 1$) | 15—16 | ПП 5 0 | | | |
| $x_k \leftarrow x_k + y$ | 17—19 | ИП 1 + П 1 | | | |
| <u>Вызов ДАТЧИК</u> ($x, y, k, p, -1, 1$) | 20—21 | ПП 5 0 | | | |
| $y_k \leftarrow y_k + 1$ | 22—24 | ИП 2 + П 2 | | | |
| <u>Вызов ЕВКЛИД</u> (x_k, y_k, x_c, y_c, d) | 25—26 | ПП 3 3 | | | |
| <u>до</u> $ d < p$ | 27— | ИП 6 — F < | | | |
| <u>Все — цикл</u> | —30 | 0 6 | | | |
| <u>Индикация «ERROR»</u> | 31 | F ln | | | |
| | 32 | C/P | | | |

Набор k , [П] [С],Набор p , [П] [D],Набор -0.5 , [П] [A],

Набор 2, [П] [B],

Набор x_0 , [П] [9],Набор ρ , [П] [6].

4. Первый игрок тайком от второго осуществляет ввод значений x_k и y_k — координат точки, в которую он помещает корабль:

Набор x_k , [П] [1], Набор y_k , [П] [2].

5. Первый игрок нажимает на клавиши

[B/O] [C/P].

Через несколько секунд на индикаторе высвечивается расстояние d от начала координат до корабля.

6. Второй игрок вводит координаты точки, в которую он посыпает снаряд:

Набор x_c , С/П, Набор y_c , С/П.

7. Через некоторое время либо на индикаторе высвечивается «*ERROR*», что означает «корабль потоплен» и следует перейти к п. 8 инструкции, либо высвечивается на индикаторе расстояние d от точки падения снаряда до корабля, тогда второй игрок возвращается стрельбу, т. е. переходит к п. 6 инструкции.

8. Первый игрок нажимает на клавиши

ИП 0,

и на индикаторе высвечивается количество выстрелов, которое произвел второй игрок, пока не потопил корабль.

9. Игроки меняются ролями.

10. Победителем объявляется тот, кто затратил меньше снарядов на уничтожение корабля противника.

ПОТОПИ КОРАБЛЬ

Ввод (x_k, y_k)

Ввод (ρ)

Ввод ($x, k, p, -0.5, 2$)

Вызов ЕВКЛИД ($x_k, y_k, 0, 0, d$)

$c\leftarrow 0$

Цикл

повторяй

Индикация d

Ввод (x_c, y_c)

$c\leftarrow c+1$

Вызов ДАТЧИК ($x, y, k, p, -1, 1$)

$x_k \leftarrow x_k + y$

Вызов ДАТЧИК ($x, y, k, p, -1, 1$)

$y_k \leftarrow y_k + y$

Вызов ЕВКЛИД (x_k, y_k, x_c, y_c, d)

до

$d < \rho$

Все — цикл

Индикация «*ERROR*»

Конец ПОТОПИ КОРАБЛЬ

§ 2.8. ИГРА «ВОВРЕМЯ ОСТАНОВИСЬ»

Играют двое, осуществляя ход поочередно. Ход состоит в следующем: игрок несколько раз подбрасывает игральную кость, грани которой занумерованы числами 0, 1, 2, 3, 4, 5, пока

не решает прекратить или пока не выпадет 0. Если игрок решил прекратить подбрасывание кости до выпадения 0, то ему за этот ход засчитывается сумма всех выпавших очков. Если же выпадает 0, то «сгорают» все очки, выпавшие за данный ход. Очки, полученные за все ходы, суммируются. Выигрывает игрок, набравший наибольшую сумму очков за n ходов. Значение n устанавливается по договоренности.

В эту игру можно играть с помощью калькулятора, используя вместо игральной кости генератор случайных чисел.

Спроектируем на псевдокоде алгоритм, позволяющий с помощью калькулятора проводить рассматриваемую игру.

Введем обозначения:

S — общая сумма набранных очков в игре,

Σ — сумма очков, набранных за один ход,

y — количество очков, набранных за 1 бросание кости.

ВОВРЕМЯ ОСТАНОВИСЬ

Ввод общей суммы очков, набранных до данного хода

Сделать ход, придерживаясь правил

Увеличить общую сумму набранных очков на количество очков, набранных данным ходом

Вывод общей суммы набранных очков

Конец ВОВРЕМЯ ОСТАНОВИСЬ

Детализируем предложения псевдокода.

Ввод общей суммы очков, набранных до данного хода — Ввод (S)

Сделать ход, придерживаясь правил —

Подготовить — $\Sigma \leftarrow 0$

Цикл

повторяй

Имитация подбрасывания игральной кости

Получение новой суммы очков, набранных за ход

до

ход закончен — на индикаторе «0»

Все — цикл

Вызов ДАТЧИК ($x, y, k, p, 0.5$)

$\Sigma \leftarrow \Sigma + y$

Если

$y = 0$

то

$\Sigma \leftarrow 0$

Все — если

Индикация Σ

Увеличить общую сумму
набранных очков

на количество очков, ————— $S \leftarrow S + \Sigma$

набранных данным ходом

Вывод общей суммы
набранных очков ————— Индикация S

Полученный после детализации псевдокод будет иметь вид:

ВОВРЕМЯ ОСТАНОВИСЬ

Ввод (S)

$\Sigma \leftarrow 0$

Цикл

повторяй

вызов ДАТЧИК ($x, y, k, p, 0.5$)
 $\Sigma \leftarrow \Sigma + y$

Если

$y = 0$

то

$\Sigma \leftarrow 0$

Все — если

Индикация Σ

до

на индикаторе «0»

Все — цикл

$S \leftarrow S + \Sigma$

Индикация S

Конец ВОВРЕМЯ ОСТАНОВИСЬ

Таблица 66

| ПРОГРАММА ВОВРЕМЯ ОСТАНОВИСЬ | | | | | | | | | |
|--|----------|-----|--|---------|-----|-----------|---------|-----|--|
| КАЛЬКУЛЯТОР «Электроника Б3-34» | | | ПРОГРАММИСТ Автор Дата 10.02.83 лист 1 листов 1 | | | | | | |
| Распределение памяти | | | | | | | | | |
| Переменные | Σ | x | y | k | p | $A + 0.5$ | $B - A$ | S | |
| RG | I | 9 | 2 | C | D | A | B | 0 | |
| КОММЕНТАРИИ | | | Адреса команд | КОМАНДЫ | | | | | |
| Ввод (S) $\Sigma \leftarrow 0$ | | | 00 | P | 0 | | | | |

| КОММЕНТАРИИ | Адреса команд | КОМАНДЫ | | | | | | | |
|--|---------------|---------|---|-----|---|---|--|--|--|
| | | 0 | P | I | | | | | |
| Цикл | 01—02 | 0 | P | I | | | | | |
| повторяй | 03—05 | PП | 5 | 0 | P | 2 | | | |
| Вызов ДАТЧИК ($x, y, k, p, 0.5$) | 06—08 | ИП | 1 | + | P | 1 | | | |
| $\Sigma \leftarrow \Sigma + y$ | 09— | ИП | 2 | | | | | | |
| Если | | F | = | | | | | | |
| $y = 0$ | | 1 | 3 | | | | | | |
| то | | P | 1 | | | | | | |
| $\Sigma \leftarrow 0$ | | ИП | 1 | C/P | | | | | |
| Все — если | | F | = | | | | | | |
| Индикация Σ | | 0 | 3 | | | | | | |
| до | | ИП | 0 | ИП | 1 | | | | |
| на индикаторе «0» | | + | P | 0 | | | | | |
| Все — цикл | | C/P | | | | | | | |
| $S \leftarrow S + \Sigma$ | | | | | | | | | |
| Индикация S | | | | | | | | | |

Транслируя полученный псевдокод на язык «Электроники Б3-34», получим программу (табл. 66), которая и поможет нам проводить игру «ВОВРЕМЯ ОСТАНОВИСЬ».

Изложим инструкцию по проведению игры «ВОВРЕМЯ ОСТАНОВИСЬ».

Инструкция

- Кто-либо из играющих осуществляет ввод:
- программы ВОВРЕМЯ ОСТАНОВИСЬ (см. табл. 66);
- подпрограммы ДАТЧИК (см. табл. 62);
- подпрограммы АНТЬЕ (см. табл. 63)

в программную память калькулятора.

- Выбирается число n — количество ходов в партии.
- Осуществляется ввод данных, необходимых для работы датчика случайных чисел:

Набор k ,

| | |
|---|---|
| P | C |
|---|---|

,
Набор 5,

| | |
|---|---|
| P | B |
|---|---|

,
Набор 0,5,

| | |
|---|---|
| P | A |
|---|---|

.

Набор p ,

| | |
|---|---|
| P | D |
|---|---|

,
Набор x_0 ,

| | |
|---|---|
| P | 9 |
|---|---|

,

4. Далее игроки ходят по очереди.

Ход состоит в следующем. Игрок нажимает на клавишу **B/O**, осуществляет набор общей суммы S , имеющейся у него перед ходом, и несколько раз нажимает на клавишу **C/P**,

при этом всякий раз на индикаторе высвечивается значение Σ — текущей суммы очков, набранных за данный ход. Как только игрок решает прекратить дальнейший набор очков, он нажимает на клавиши **0 C/P**, если же «выпадает» 0, он просто нажимает на клавишу **C/P**. В обоих случаях на индикаторе высвечивается новое значение S , которое игроку нужно запомнить или записать.

5. Побеждает тот, кто за n ходов наберет большую сумму очков.

§ 2.9. МАТРИЧНЫЕ ИГРЫ

В заключение главы «Калькулятор и случай» мы рассмотрим так называемые матричные игры. Приведем пример такой игры.

Каждый из двух игроков А и В выбирает одну из двух имеющихся у него стратегий (см. с. 139). Выбор стратегий производится тайком от соперника (номер избранной стратегии записывается на листке бумаги так, чтобы не видел соперник). Затем игроки сообщают друг другу об избранных ими стратегиях (показывают друг другу листки с записью). На пересечении соответствующих строки и столбца таблицы, называемой платежной матрицей, читается количество очков, набранных игроком А. Отметим, что платежная матрица составляется «с точки зрения игрока А». Если элемент платежной матрицы — положительное число, то А выигрывает, а если отрицательное, то проигрывает указанное количество очков. Например, если А выбрал стратегию 1, а В — стратегию 2, то А проигрывает 2 очка. Если же А и В выбрали стратегию 2, то А выигрывает 8 очков.

В игры такого типа интересно играть двум участникам с помощью калькулятора. Игроки тайком друг от друга «сообщают» калькулятору об избранных ими стратегиях, а калькулятор «сообщает», каков соответствующий этим стратегиям элемент платежной матрицы. Кроме того, калькулятор подсчитывает количество набранных за несколько партий очков «с точки зрения игрока А».

| | | Стратегия В | |
|-------------|---|-------------|----|
| | | 1 | 2 |
| Стратегия А | 1 | 1 | -2 |
| | 2 | -7 | 8 |

Составим программу для калькулятора. Введем обозначения:
 i — стратегия, выбранная игроком А,
 j — стратегия, выбранная игроком В,
 a_{ij} — элемент платежной матрицы, соответствующий стратегиям i и j ,

Σ — сумма набранных очков «с точки зрения игрока А».

Каждому элементу a_{ij} платежной матрицы поставим в соответствие его номер по формуле

$$\text{номер} = j + n \cdot (i - 1),$$

где n — размерность платежной матрицы, или, что то же самое, количество стратегий, имеющихся в распоряжении каждого игрока. При распределении памяти всякому элементу платежной матрицы выделим ту ячейку памяти, каков его номер. Программа приведена в таблице 67. Отметим, что, хотя программа составлена для произвольного n , ресурсы нашего калькулятора позволяют играть лишь в случае матриц размерностей 2×2 и 3×3 .

Инструкция

1. Программа (см. табл. 67) заносится в программную память калькулятора.

2. Одним из игроков (с ведома второго) в регистры памяти калькулятора вводятся элементы платежной матрицы.

a) В случае $n=2$:

$$\begin{array}{lll} a_{11} \rightarrow RG1 & a_{12} \rightarrow RG2 \\ a_{21} \rightarrow RG3 & a_{22} \rightarrow RG4 \end{array}$$

b) В случае $n=3$:

$$\begin{array}{lll} a_{11} \rightarrow RG1 & a_{12} \rightarrow RG2 & a_{13} \rightarrow RG3 \\ a_{21} \rightarrow RG4 & a_{22} \rightarrow RG5 & a_{23} \rightarrow RG6 \\ a_{31} \rightarrow RG7 & a_{32} \rightarrow RG8 & a_{33} \rightarrow RG9 \end{array}$$

3. Размерность игры n заносится в RG0. Значение 0 заносится в RGC.

УПРАЖНЕНИЯ

ПРОГРАММА МАТРИЧНАЯ ИГРА $n \times n$

| КАЛЬКУЛЯТОР «Электроника Б3-34» | | ПРОГРАММИСТ Автор Дата 20.07.83 лист 1 листов 1 | | | | | | |
|--|-------|--|----------|----------|-------|-----|----|---|
| Распределение памяти | | | | | | | | |
| Переменные | i | j | a_{ij} | Σ | номер | n | | |
| RG | A | B | номер | C | D | 0 | | |
| КОММЕНТАРИИ | | Адреса команд | КОМАНДЫ | | | | | |
| Номер $\leftarrow j + n(i - 1)$ $\Sigma \leftarrow \Sigma + a_{ij}$ Индикация a_{ij} | 00 | | ИП | A | I | - | ИП | 0 |
| | -07 | | X | ИП | B | + | П | D |
| | 08- | | K | ИП | D | ИП | C | |
| | -11 | | + | П | C | | | |
| | 12 | | K | ИП | D | | | |
| | 13-15 | | C/P | БП | 0 | 0 | | |

4. Игрок А тайком от игрока В заносит в RGA номер выбранной им стратегии. Игроку А следует не забывать после этого стереть информацию с индикатора. Игрок В заносит в RGB номер выбранной им стратегии. Затем осуществляется пуск программы. На индикаторе при этом загорается соответствующий элемент платежной матрицы.

5. Процедура, описанная в п. 4, повторяется некоторое установленное по договоренности количество раз.

6. Один из игроков нажимает на клавиши ИП С. На индикаторе загорается сумма очков, набранных игроком А. Если эта сумма больше нуля, то победителем объявляется А. Если же эта сумма отрицательна, то победа присуждается игроку В.

В случае нуля на индикаторе фиксируется ничейный исход. Можно составить также программу матричной игры калькулятора с человеком. При этом калькулятор «должен пользоваться» датчиком случайных чисел для выбора той или иной стратегии.

1. ИГРА БОЛТЯНСКОГО [8]

Двое играют в такую игру: первый называет натуральное число от 2 до 9; второй умножает это число на произвольное натуральное число от 2 до 9, затем первый умножает результат на любое натуральное число от 2 до 9 и т. д. Выигрывает тот, у кого впервые получится произведение, большее заданного положительного C .

Выясните, при каких C существует выигрышная стратегия у первого игрока и при каких C — у второго. Спроектируйте алгоритм и составьте программу игры калькулятора с человеком.

2. ИГРА ШЕНОНА [41]

Игра происходит на произвольном графе с двумя выделенными вершинами. Два игрока, которых назовем «Вырубить» и «Закоротить», играют по очереди. При каждом ходе «Вырубить» удаляет из графа одно ребро, а «Закоротить» закрепляет за собой какое-либо из оставшихся ребер и тем самым защищает его от удаления в будущем игроком «Вырубить». Игрок «Закоротить» выигрывает, если он сумеет сохранить путь, соединяющий две выделенные вершины; в противном случае выигрывает «Вырубить».

Составьте программу для игры калькулятора с человеком на достаточно простом графе.

3. ИГРА НИМБЫ [19]

Поле для игры — прямоугольная область, разбитая на клетки. За каждым игроком закрепляется одно из направлений, вертикальное или горизонтальное, и ходы делаются по очереди. При данном ходе игрок, движущийся по горизонтали, захватывает незанятую клетку; далее этот игрок занимает эту клетку и все клетки в этой же строке, которые могут быть достигнуты без пересечения границы поля или клеток, занятых противником. Второй игрок занимает клетки в столбцах аналогичным образом. Игра заканчивается, когда одному из игроков не останется клетки для очередного хода, что означает поражение.

Выясните, в каком случае существует выигрышная стратегия для первого игрока, в каком — для второго. Спроектируйте соответствующие алгоритмы.

4. ИГРА «НИМ» [14]

Играют двое, делая ходы по очереди. Имеется n наборов предметов. Каждый набор содержит m_i предметов, $i = 1, 2, \dots, n$. Игрок при очередном ходе берет один или несколько предметов из какого-то одного набора. Игра продолжается до тех пор, пока все предметы не будут взяты; игрок, сделавший последний ход, считается победителем.

Выясните, в каком случае существует выигрышная стратегия у первого игрока и в каком — у второго. Спроектируйте соответствующие алгоритмы.

5. ИГРА «ОДИНОКИЙ ФЕРЗЬ» [20]

Игра происходит на шахматной доске. Играют двое и ходят по очереди. На произвольной клетке поля устанавливается ферзь. Каждый игрок за один ход может передвинуть ферзя либо на несколько клеток вниз по вертикали, либо на несколько клеток влево по горизонтали, либо на несколько клеток влево-вниз по диагонали. Выигрывает тот, кто своим ходом сумеет поставить ферзя на поле a1 (левый нижний угол).

Выясните, при каком начальном положении ферзя выигрывает первый игрок и при каком — второй. Спроектируйте алгоритм и составьте программу для калькулятора, играющего с человеком.

6. ИГРА ЦЗЯНЬШИЦЫ [35]

Играют двое и ходят по очереди. Положив на землю две кучки камней, играющие поочередно берут камни из этих кучек, соблюдая следующие правила:

а) Из одной кучки можно брать любое количество камней (даже всю кучку).

б) Можно брать камни одновременно из двух кучек, но непременно по одинаковому количеству из каждой кучки.

Выигрывает тот, кто своим ходом сможет забрать все лежащие на земле камни.

Выясните, в каком случае существует выигрышная стратегия у первого игрока, в каком — у второго. Спроектируйте алгоритм и составьте программу, которая помогла бы вам выигрывать эту игру у вашего соперника.

7. ИГРА «ПОБЕЖДАЕТ ЧЕТ» [35]

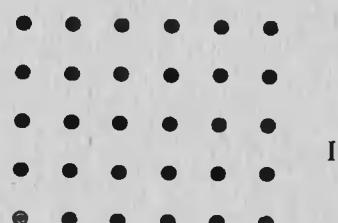
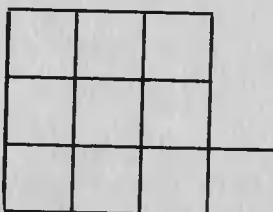
Из 27 спичек, лежащих на столе, двое играющих поочередно забирают не менее одной и не более четырех спичек. Выигравшим считается тот, у кого по окончании игры окажется четное количество спичек.

Выясните, у кого из игроков есть выигрышная стратегия. Спроектируйте алгоритм и составьте программу для калькулятора, играющего с человеком.

8. ИГРА «НОВЫЕ КРЕСТИКИ-НОЛИКИ» [16]

К доске для игры в обычные крестики-нолики добавлена одна клетка (I). Может ли игрок, делающий первый ход, обеспечить себе выигрыш?

Спроектируйте алгоритм и составьте программу для калькулятора, играющего с человеком.



9. ИГРА «37» [24]

Положите на стол пять костяшек домино, у которых число очков равно соответственно 1, 2, 3, 4, 5. Играют двое и ходят по очереди. Первый игрок кладет монету на произвольную костяшку, например на 5, что дает ему 5 очков, затем второй игрок перекладывает монету, скажем, на 3, прибавив 3 к 5, получает при этом 8 очков; затем первый игрок кладет монету, допустим, на 1 и получает сумму очков, равную 9, и т. д. Тот игрок, который наберет 37 или принудит своего противника превзойти эту сумму, выигрывает. (При каждом ходе монету перекладывают на другую костяшку.)

Докажите, что первый игрок всегда может выиграть. Спроектируйте алгоритм и составьте программу для калькулятора, играющего с человеком.

10. ИГРА «ДЕСЯТЬ КАРТ» [24]

Выложите в ряд 10 игральных карт. Первый игрок может перевернуть либо одну карту, либо две соседние. В дальнейшем, осуществляя ходы по очереди, игроки могут перевернуть либо одну, либо две соседние карты. Выигрывает тот, кто перевернет последнюю карту.

Докажите, что первый игрок всегда может выиграть. Рассмотрите эту игру для случая $2n$ карт. При каких n выигрывает первый игрок? Отметим, что ответ на этот вопрос нам неизвестен.

11. ИГРА «СИМ» [16]

На листе бумаги изображен правильный шестиугольник. Играют в «СИМ» двое. Каждый игрок соединяет отрезком две произвольные вершины своим цветом. Проигравшим считается тот, кто первым вычертит треугольник своего цвета.

Докажите, что в этой игре ничейный исход невозможен.

12. ИГРА «ЩЕЛК» [16]

Для игры в «ЩЕЛК» фишки выстраивают в форме прямоугольника (с. 142, II). Играют в «ЩЕЛК» вдвоем, игроки делают ходы по очереди. Каждый ход состоит в следующем: игрок выбирает любую фишку, мысленно проводит через ее центр два взаимно перпендикулярных луча: один горизонтальный вправо и один вертикальный вверх. Все фишки, оказавшиеся внутри прямого угла, образованного лучами, игрок снимает. Выигрывает тот, кто заставит противника взять «отправленную» фишку, стоящую в левом нижнем углу.

Докажите, что первый игрок всегда может выиграть. Спроектируйте алгоритм и составьте программу для калькулятора, играющего с человеком, в случае игр $2 \times n$ и $n \times n$.

13. ИГРА «ЦВЕТНЫЕ КРУЖКИ» (новая игра)

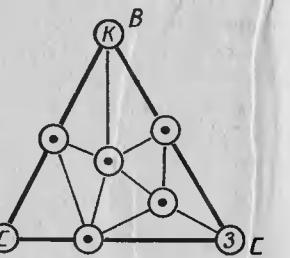
На листе бумаги расположено несколько кружков, например, так, как показано на рисунке 25.

Самый левый кружок закрашен черным цветом, а самый правый — синим. Игроки ходят по очереди, закрашивая черным или синим цветом выбранный ими еще не закрашенный кружок.



Рис. 25

Рис. 26



Игрок объявляется проигравшим, если после его хода окажется, что два соседних кружка закрашены разными цветами.

Докажите, что игра не может закончиться ничейным исходом. Выясните, в каком случае существует выигрышная стратегия у первого игрока, в каком — у второго. Спроектируйте алгоритмы, реализующие эти стратегии.

14. ИГРА «ЦВЕТНЫЕ ВЕРШИНЫ» (новая игра)

Треугольник разбит на меньшие треугольники, например, так, как показано на рисунке 26. В вершинах треугольников помещены кружки. Вершины «большого» исходного треугольника раскрашены: одна — в синий, одна — в зеленый, а одна — в красный цвет. Остальные вершины не закрашены. Играют двое или трое, поочередно закрашивая еще не закрашенную вершину произвольно выбранным цветом; однако кружки, лежащие на ребре BC , нельзя закрашивать синим, лежащие на AC — красным, а лежащие на AB — зеленым цветом. Проигравшим объявляется тот игрок, после хода которого окажется, что вершины некоторого «малого» треугольника (треугольника разбиения) закрашены тремя различными цветами.

Докажите, что игра не может окончиться ничейным исходом.

15. РАЗНОВИДНОСТЬ ИГРЫ БАШЕ

В эту игру играют так же, как и в игру БАШЕ, однако игрок, взявший последний предмет, объявляется проигравшим.

Выясните, в каком случае существует выигрышная стратегия у первого игрока, в каком — у второго. Спроектируйте алгоритм и составьте программу для калькулятора, играющего с человеком.

16. РАЗНОВИДНОСТЬ ИГРЫ «ВПЕРЕД И ВВЕРХ»

В эту игру играют так же, как и в обычную игру «ВПЕРЕД И ВВЕРХ», однако игрок, достигший вершины (правой верхней точки), объявляется проигравшим.

Выясните, в каком случае существует выигрышная стратегия у первого игрока и в каком — у второго. Спроектируйте алгоритм и составьте программу для калькулятора, играющего с человеком.

17. ЕЩЕ ОДНА РАЗНОВИДНОСТЬ ИГРЫ БАШЕ

Один из игроков выкладывает на столе некоторое количество предметов и решает, какое максимальное количество предметов

разрешается забрать за один ход. Кроме того, этот игрок решает, кто объявляется победителем: тот, кто взял последний предмет, или тот, кто заставил сделать это своего противника. Второй же игрок выбирает начинаящего игрбу.

Спроектируйте алгоритм и составьте программу для калькулятора, играющего с человеком. (Калькулятор «выступает в роли» второго игрока.)

18. ИГРА «ВЕРОЯТНОСТНЫЙ ВАРИАНТ КРЕСТИКОВ-НОЛИКОВ» [19]

Пронумеруем клетки обычной игры в крестики-нолики 3×3 числами от 1 до 9. Игрок, пытающийся поместить крестик или нолик в клетку i с вероятностью p_i , не может этого сделать, не теряя ход. Центральная клетка всегда имеет большую вероятность потери хода, чем любая из остальных.

Составьте программу для калькулятора и испытайте для различных наборов p_i разнообразные стратегии игры.

19. ИГРА «ТРИ КОСТИ» [32]

Каждый из игроков подбрасывает три кости и подсчитывает сумму выпавших очков. Выигрывает тот, у кого эта сумма совпала с одним из двух названных до броска чисел.

Проведите «машинный эксперимент», используя программу «ДАТЧИК» (см. табл. 62). Какие числа следует называть перед броском? Выработайте рекомендации и попробуйте обосновать их теоретически.

Составьте программу для калькулятора, генерирующего случайные числа 0 и 1 с вероятностью p_1 и p_2 соответственно. Постарайтесь придумать игру, для реализации которой на калькуляторе пригодилась бы составленная программа.

21. ИГРА «НАЗОВИ НАИБОЛЬШЕЕ ЧИСЛО» [41]

Один игрок по секрету от другого выбирает любую последовательность из n натуральных чисел. Эти числа записываются и тасуются подобно колоде из n карт, после чего они по одному предъявляются второму игроку, которого мы назовем Оракулом. Просматривая по очереди предъявляемые числа, Оракул может по желанию прервать игру, увидев некоторое число и заявив: «Остановись, это число — наибольшее в данной последовательности». Как только Оракул это произнесет, немедленно открывается весь набор чисел и определяется, было ли заявление Оракула верным. Если это так, то выигрывает Оракул, в противном случае выигрывает первый игрок.

Проведите «машинный эксперимент», используя программу «ДАТЧИК» (см. приложение). Когда следует прерывать игру? Сформулируйте рекомендации и постарайтесь их обосновать.

22. ИГРА «НАБЕРИ 15» [17]

На 15 картах, выложенных в ряд на столе, изображены числа от 1 до 15. Играют двое, по очереди забирая одну из карт со стола. Выигрывает тот, у кого на руках раньше окажется три карты, дающие в сумме 15.

Докажите, что существует беспроигрышная стратегия как у начинающего игрока, так и у второго игрока.

23. ИГРА «УПРОЩЕННЫЙ ВАРИАНТ ЦЗЯНЬШИЦЗЫ» [12]

Из двух кучек, содержащих n и m предметов, двое играющих по очереди забирают предметы. Каждый игрок имеет право из любой кучки взять один предмет или из двух кучек — одинаковое количество предметов. Выигрывает тот, кто заберет последний предмет.

Выясните, в каком случае существует выигрышная стратегия у первого игрока и в каком — у второго. Спроектируйте алгоритм и составьте программу для калькулятора, играющего с человеком.

24. Составьте программу генерирования случайных чисел по методу фон Неймана (см. § 2.6).

25. Последовательность случайных чисел, равномерно распределенных на $(0; 1)$, может быть задана рекуррентным соотношением:

$$x_0 = 0,002,$$
$$x_{i+1} = \{11x_i + \pi\}.$$

Здесь $\{\cdot\}$ — символ дробной части [54].

Составьте программу для получения случайных чисел этим методом для калькулятора.

26. ИГРА Г. ФРОЙДЕНТАЛЯ

Дан конечный направленный граф («решетка») с одним наивысшим и одним наизнешним узлами (не совпадающими между собой). Играют двое; они поочередно ставят на узлы фишку; если некоторый узел уже занят, ставить на все низшие запрещено. Тот, кто вынужден поставить на наивысший узел, проигрывает.

Докажите, что начинающий обладает выигрывающей стратегией.

27. ИГРА «ПОГОНЯ»

Играют двое: «Догоняющий» и «Убегающий». Оба игрока ставят на шахматную доску по королю и ходят по очереди, передвигая своего короля на соседнее (по горизонтали, вертикали или диагонали) поле. Цель «Догоняющего» — поставить своего короля на поле, соседнее с полем, на котором стоит король «Убегающего».

Докажите, что «Догоняющий» всегда сможет этого добиться. Составьте программу для калькулятора, которая «играет» за «Догоняющего».

28. ИГРА «ТРИГЕКС» [30]

Игра проводится на специальном игровом поле, изображенном на рисунке 27. В игре участвуют двое, ходят по очереди. Один из соперников ходит белыми шашками, другой — черными. Сделать ход — это значит установить шашку своего цвета в один из кружков игрового поля. Победителем считается тот,

кто сумеет первым поставить три своих шашки вдоль одной из девяти прямых.

Докажите, что существует выигрышная стратегия у первого игрока.

Спроектируйте соответствующий алгоритм.

29. ИГРА «ВЕЛИКИЙ КОМБИНАТОР» [51]

Один из игроков (загадывающий) записывает секретную комбинацию из любых четырех цифр от 1 до 6 (повторения допускаются), называемую кодом. Второй игрок (отгадывающий) пытается раскрыть код, высказывая разумные предположения, называемые пробами. Каждая проба, как и код, представляет собой произвольную комбинацию в диапазоне от 1 до 6. Отгадывающий игрок сообщает пробу загадывающему, и тот должен ответить, сколько цифр в пробе совпадают с цифрами кода как по положению, так и по величине и сколько из остальных цифр пробы входят в код, но стоят на другом месте. Так, на пробу 1123 при коде 4221 будет получен ответ: «Одна цифра совпадает и стоит на том же месте, и еще одна совпадает, но стоит на другом месте». Игра продолжается до тех пор, пока отгадывающий не назовет пробу, в точности совпадающую с кодом, т. е. пока не отгадает код. После этого игроки меняются ролями. Победителем считается тот из игроков, кто определит код противника за меньшее число проб.

Составьте программу, которая по задуманному числу и по ответу отгадывающего находит число точных попаданий и число совпадений по значению.

30. ИГРА С. ФОМИНА [52]

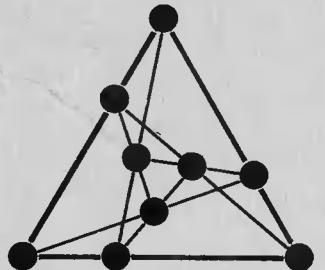
Имеется несколько кучек камней. Двое играют в игру, ход которой состоит в том, что игрок разбивает каждую кучку, состоящую более чем из одного камня, на две меньшие кучки. Ходы делаются поочередно до тех пор, пока во всех кучках не останется по одному камню. Победителем считается игрок, сделавший последний ход.

Выясните, в каком случае выигрывает начинающий, а в каком — его противник. Спроектируйте соответствующие алгоритмы.

31. ИГРА ГРАНДИ [42]

Перед двумя играющими расположена единственная кучка предметов, например стопка монет. Первый игрок делит исходную стопку на две новые стопки, которые должны быть неравными. В дальнейшем каждый из игроков, когда приходит его очередь делать ход, делает то же самое с одной из стопок монет. Игра продолжается до тех пор, пока все стопки не будут содержать по одной-двух монеты, после чего игра заканчивается, поскольку дальнейшее ее продолжение невозможно. Проигрывает тот из игроков, кто первым окажется в положении, когда игра не может продолжаться.

Исследуйте игру Гранди с небольшим количеством предметов.



32. ИГРА «БРИДЖ-ИТ» [15]

На рисунке 28 показана специальная доска для игры «БРИДЖ-ИТ». На квадратном поле отмечены узлы двух прямоугольных решеток, вдвинутых одна в другую; узлы одной решетки обозначены кружками, узлы другой — квадратиками. Участники игры — их двое — по очереди проводят вертикальные и горизонтальные линии — мостики, соединяя два одинаковых узла. Один из игроков соединяет кружки, другой — квадратики. Диагональными линиями узлы соединять не разрешается, а вертикальные и горизонтальные линии противника нигде не должны пересекаться. Выигрывает тот, кто первым построит ломаную линию, соединяющую две противоположные стороны игрового поля: игрок, соединяющий кружки, — верх и низ, а игрок, соединяющий квадратики, — правую и левую сторону поля.

Докажите, что существует выигрышная стратегия для начинающего игрока.

33. Написан многочлен [7]: $x^{10} + *x^9 + \dots + *x^2 + *x + 1$. Двое играют в такую игру. Сначала первый заменяет любую из звездочек некоторым числом, затем второй заменяет числом любую из оставшихся звездочек, затем снова первый заменяет одну из звездочек числом и т. д. (всего 9 ходов). Если у полученного многочлена не будет действительных корней, то выигрывает первый игрок, а если будет хотя бы один корень — выигрывает второй.

Докажите, что существует выигрышная стратегия у второго игрока.

34. Даны две кучки спичек. В начале игры в одной кучке m спичек, в другой — n спичек, $m > n$. Двое игроков по очереди берут из кучки спички. За один ход игрок берет из одной кучки любое (отличное от нуля) число спичек, кратное числу спичек в другой кучке. Выигрывает игрок, взявший последнюю спичку в одной из кучек.

Докажите, что если $m > 2n$, то игрок, делающий первый ход, может обеспечить себе выигрыш. При каких α верно следующее утверждение: если $m > \alpha n$, то игрок, делающий первый ход, может обеспечить себе выигрыш? [44]

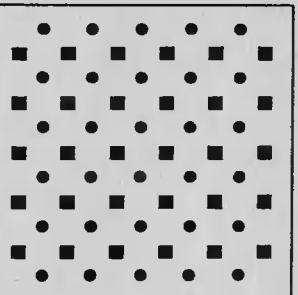


Рис. 28

35. Фишка стоит в углу шахматной доски размером $n \times n$ клеток. Каждый из двух играющих по очереди передвигает ее на соседнее поле (имеющее общую сторону с тем, на котором стоит фишка). Второй раз ходить на поле, где фишка уже побывала, нельзя. Проигрывает тот, кому некуда ходить. [44]

Выясните, в каком случае существует выигрышная стратегия у начинающего игрока и в каком — у второго игрока.

36. Имеется система уравнений [9]:

$$\begin{cases} *x + *y + *z = 0 \\ *x + *y + *z = 0 \\ *x + *y + *z = 0. \end{cases}$$

Два человека поочередно вписывают вместо звездочек числа. Докажите, что начинающий всегда может добиться того, чтобы система имела ненулевое решение.

37. Имеется набор одинаковых фишек, и двое играющих поочередно кладут их на круглый стол, причем фишкки должны не задевать друг друга. Выигрывает тот, кто поставил последнюю фишку.

Выясните, у кого из играющих есть выигрышная стратегия. Спроектируйте алгоритм, реализующий эту стратегию.

38. Двое играют в такую игру: перед ними на бумаге в цепочку нарисовано n минусов. Каждый по очереди переправляет один или два соседних минуса на плюс. Выигрывает тот, кто переправит последний минус.

Докажите, что для любого n существует выигрышная стратегия для начинающего игрока. Составить программу для калькулятора, реализующую эту стратегию (см. [50]).

39. ИГРА «ХОД КОНЕМ» [20]

Игра «ХОД КОНЕМ» отличается от игры «ОДИНОКИЙ ФЕРЗЬ» тем, что здесь игроки поочередно переставляют на шахматной доске не ферзя, а коня. Ходить конем можно на два поля вниз и потом на одно поле вправо или влево или на два поля влево и потом на одно поле вверх или вниз. Выигрывает тот, кто поставит своим очередным ходом коня в положение, при котором противнику будет некуда ходить.

Выясните, при каком начальном положении коня выигрывает начинающий игрока и при каком — его противник.

40. ИГРА НА МАТРИЦЕ [19]

Два игрока, «нечетный» и «четный», по очереди ставят единицы и нули в незанятые позиции поля $N \times N$. Каждый из игроков может ставить 1 или 0 в произвольную свободную позицию, тем самым занимая ее. Заметим, что «нечетный» не обязательно должен пользоваться только единицами, а «четный» — нулями. Игра продолжается до заполнения всех позиций. После этого суммируются числа вдоль каждой строки, каждого столбца и главных диагоналей. Число ODD нечетных сумм далее сравни-

вается с числом $EVEN$ четных сумм. Если $ODD > EVEN$, выигрывает «нечетный»; если $EVEN > ODD$, выигрывает «четный»; если $ODD = EVEN$, результат считается ничейным.

Постройте выигрышную стратегию для одного из игроков в этой игре.

41. ИГРА «ОБОБЩЕННЫЙ НИМ»

В эту игру играют так же, как и в обычный «НИМ», однако разрешается одновременно брать предметы не более чем из k кучек (число k устанавливается по договоренности в начале игры).

Выясните, в каких начальных позициях существует выигрышная стратегия для начинающего играть и в каких — для его соперника.

42. ИГРА «ОСОБЫЕ ТОЧКИ» [5]

Задано несколько красных и несколько синих точек, некоторые из которых соединены между собой. Точка называется особой, если более половины из соединенных с ней точек имеют цвет, отличный от ее цвета. Двое играющих ходят поочередно, перекрашивая одну из особых точек в другой цвет. Побеждает тот, после хода которого не останется ни одной особой точки.

Докажите, что игра не заканчивается ничейным исходом.

43. ИГРА «ДЕТЕРМИНАНТ» (студенческий фольклор).

Задана таблица (матрица) 3×3 . Двое играющих поочередно вставляют в еще не заполненные клетки таблицы произвольные числа. Игра продолжается до полного заполнения таблицы. Побеждает начинающий игру, если окажется, что детерминант матрицы положителен. В противном случае побеждает его соперник.

Составьте программу для калькулятора, которая после окончания игры «назовет» победителя.

44. ИГРА «ПРИНЦЕССА И ЧУДОВИЩЕ» [49]

Американским ученым Р. Айзексом сформулирована задача: «В абсолютно темном помещении заключены Принцесса, которая может двигаться с любой скоростью и в любом направлении, и Чудовище, скорость которого ограничена. Чудовище захватывает Принцессу, если расстояние между ними становится меньше заданного. Какой должна быть стратегия Принцессы, чтобы она могла максимально оттянуть поимку?»

Предлагаем игру по мотивам этой задачи.

Игра происходит на целочисленной решетке размером $n \times n$ (рис. 29). Несколько играющим поочередно предоставляется возможность выступить в роли Чудовища. Выигравшим считается тот, кто на поимку Принцессы затратит наименьшее количество ходов. В начальной позиции Принцесса находится в точке $P(0, n)$, а Чудовище в точке $Ч(n, 0)$. Всякий ход состоит в следующем: Принцесса перемещается случайным образом в произвольный узел решетки. Игрок-Чудовище передвигается, изменяя каждую из координат не более чем на K единиц. (Естественно, что ходы обеих сторон не совпадают.)

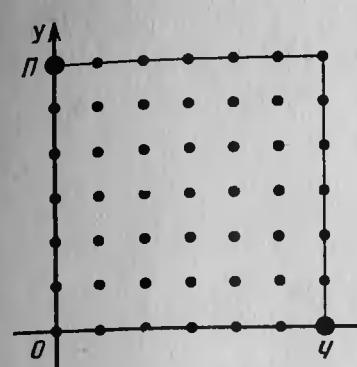


Рис. 29

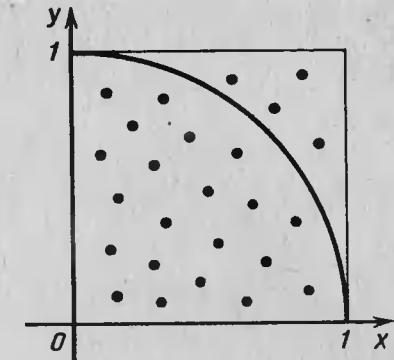


Рис. 30

венно, что игроку не известно, где находится Принцесса, ведь игра происходит в тёмной комнате!) Значение K устанавливается в начале игры по договоренности. Если окажется, что расстояние между Чудовищем и Принцессой не превосходит некоторого R , то Принцесса поймана и к игре приступает следующий игрок.

П р и м е ч а н и я .

1. Значение R можно рассматривать как параметр, характеризующий упитанность Принцессы.

2. Начинающим Чудовищам рекомендуется в начале игры устанавливать значение R не слишком малым.

Составьте программу для проведения игры «Принцесса и Чудовище» с помощью калькулятора. Эта программа должна имитировать движение Принцессы, сигнализировать о поимке Принцессы и подсчитывать число затраченных на поимку ходов.

45. Составьте программу для калькулятора, играющего с человеком в матричную игру 2×2 и 3×3 .

46. Найдите приближение числа π методом Монте-Карло, используя датчик случайных чисел, равномерно распределенных на промежутке $[0; 1]$. Сделать это можно так: получить в квадрате (рис. 30) n случайных точек. Пусть m — число точек, попавших внутрь четверти круга. Площадь четверти круга приближенно равна $\frac{\pi}{4}$.

Но тогда получим:

$$\pi \approx \frac{4m}{n}.$$

ТЕСТИРОВАНИЕ И ОТЛАДКА ПРОГРАММ

При составлении программ, как и в любом виде человеческой деятельности, возможно появление различного рода ошибок. Поэтому, прежде чем полученная программа будет использована, необходимо убедиться в том, что она действительно решает ту задачу, для которой предназначена. Существуют строгие аналитические методы доказательства правильности программ (см., например, [3]).

На практике, однако, чаще всего контроль правильности программ осуществляют, подвергая их специальной проверке, называемой тестированием.

Тестирование — процесс выполнения программы или ее части с намерением найти ошибку.

Тестируют программы следующим образом:

- 1) Не пользуясь программой, определяют ожидаемые результаты ее работы в некоторых частных случаях (при специально подобранных значениях исходных данных).
- 2) По составленной программе находят результаты ее работы в рассматриваемых частных случаях.
- 3) Сравнивают предполагаемые и полученные результаты. Несовпадение их свидетельствует о наличии ошибок в программе (предполагается, что в тестах ошибок нет).

Тестирование представляет собой серьезную проблему уже хотя бы потому, что проверка правильности работы программы при всевозможных значениях исходных данных неосуществима.

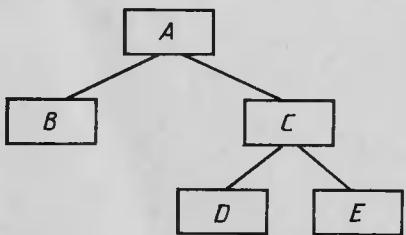


Рис. 31



Рис. 32

Подбор тестов не должен быть произвольным. Обычно на практике тесты составляют таким образом:

- а) каждая команда программы выполнялась хотя бы один раз.
- б) исходные данные для тестовых проверок были самыми разнообразными.

Часто решить ту или иную задачу помогает прием разбиения ее на части (подзадачи). Так мы поступали при проектировании сложных программ, выделяя отдельные их части в подпрограммы. Так можно поступать и в случае тестирования.

Рассмотрим один из методов — тестирование сверху вниз, или именуемое тестированием. Для разъяснения его сути рассмотрим программу, структура которой изображена на рисунке 31. Вначале проектируется и кодируется основная программа А. Подпрограммы же В и С заменяются имитирующими результаты их работы «подыгрывающими» подпрограммами, называемыми иногда заглушками. Таким образом, не имея еще подпрограмм В и С, удается протестировать модуль А. Далее проектируется и кодируется один из вызываемых А модулей, например С. Заменяя Д и Е заглушками (вместо В мы поставили заглушку ранее), тестируем «связку» А — С. Аналогично предыдущему последовательно присоединяются модули В, Д, Е.

Предположим, что на одном из шагов тестирования обнаружена ошибка. Наиболее правдоподобно, что она находится в последнем присоединенном модуле.

В качестве примера рассмотрим программу ГОЛЬДБАХ (см. § 1.10). Структура этой программы изображена на рисунке 32. Покажем, как протестировать модуль Гольдбах, не имея подпрограммы «Проверка». Заменим эту подпрограмму заглушкой. Начиная с шага 24 введем в программную память калькулятора

ИП 5 С/П В/О,

а в RG5 поместим специальную константу 11111111, которая будет сигнализировать о прерывании программы (тем самым мы сможем отличать прерывание от окончания программы).

Замечание. Если бы у нас были обращения к нескольким подпрограммам (пришлось бы использовать несколько заглушек), то *i*-я заглушка выглядела бы, например, так:

ИП 5 i X С/П В/О.

Такая заглушка сообщает нам о том, работу какой подпрограммы следует имитировать.

Теперь осуществляем прогонку нашей программы. После всякого прерывания «подыгрываем» нашей программе вручную:

1. Читаем содержимое RGA ИП А .

2. Если число на индикаторе простое, нажимаем на клавишу **0**, иначе на клавишу **1** (именно так должна работать подпрограмма).

3. Нажимаем на клавишу **C/P**.

После окончания работы программы на индикаторе загорается p_1 ; нажав на клавишу **↔**, читаем p_2 . Такое тестирование дало нам:

$$4=2+2, \quad 6=3+3, \quad 8=3+5, \quad 20=3+17, \quad 28=5+23, \dots$$

Аналогично после разработки и кодирования подпрограммы ПРОВЕРКА проводим тестирование «связки» ГОЛЬДБАХ — ПРОВЕРКА, заменив подпрограмму ОСТАТОК такой же заглушкой.

Отметим, что нисходящее тестирование не следует воспринимать догматически. Иногда (в случае простых программ) рациональнее закодировать программу вместе со всеми подпрограммами и тестируировать ее в окончательном виде. В других случаях одна или несколько необходимых подпрограмм могли быть разработаны и проверены ранее, при решении другой задачи, и поэтому нет необходимости слепо следовать тому, что предписывает нисходящий метод.

Если при тестировании будут обнаружены ошибки, то их необходимо исправить, или, как говорят, отладить программу.

Отладка — процесс установления точной природы ошибки, ее локализация и исправление.

В программах встречаются ошибки различной природы. Об их поиске и устранении смотри в [10], [29], [37]. Отметим, что применение без крайней необходимости различного рода ухищрений ради экономии нескольких ячеек памяти затрудняет отладку.

После завершения отладки модуль опять тестируется, затем, в случае обнаружения ошибок, повторно производится отладка и т. д. до тех пор, пока не возникает уверенность в том, что программа действительно работает верно.

Рассмотрим в заключение этого приложения, как устраниТЬ ошибки ввода программы в программную память калькулятора, с которыми вы уже наверняка сталкивались. Для того чтобы успешно бороться с такими ошибками, необходимо знать, что:

1. Каждой команде калькулятора соответствует двухсимвольный код (см. табл. на с. 156). Например, команде **C/P** соответствует код 50, а команде **↑** — код **OE**.

2. Каждая команда программы имеет свой порядковый номер, называемый адресом. Адрес представляет собой двузначное число от 00 до 97.

3. В режиме «Программирование» на индикаторе высвечиваются коды трех последних введенных команд программы, а также адрес следующей за ними команды.

Поиск и устранение ошибок ввода происходит в такой последовательности:

1. Если программа (подпрограмма) начинается с адреса 00, то в режиме «Автоматическая работа» нажимаем на клавишу **B/O**.

Если же адрес начала не 00, то нажимаем на клавишу **B/P**, а затем набираем на клавиатуре адрес начала программы многочного модуля.

2. Переходим в режим «Программирование». На адресном счетчике устанавливается при этом адрес, с которого начинается программа (подпрограмма).

3. С помощью клавиши **ШГ**, предназначенной для пошагового прохождения программы в сторону увеличения адресов, сверяем по кодам правильность введенных команд. При обнаружении неверно введенной команды поступаем так:

a) С помощью клавиши **ШГ**, предназначенной для пошагового прохождения программы в сторону уменьшения адресов, возвращаемся к ошибочно введенной команде. (Напомним, что команды перехода занимают две последовательные ячейки программной памяти.)

б) Вводим в программную память нужную команду.

Если в программную память занесена лишняя команда, то исключаем ее нажатием клавиши **K** **НОП**. При этом запишется команда «Нет операции».

Иногда полезно использовать реализованную на калькуляторе возможность наблюдения пошагового выполнения программы. Для этого в режиме «Автоматическая работа» нажимаем на клавишу **ПП** необходимое число раз, сравнивая при этом результаты работы по введенной программе с результатами теста.

ТАБЛИЦА КОДОВ КОМАНД

| Первый знак кода | Второй знак кода | | | | | | | |
|------------------------|------------------|---------|---------|-----------------|-------------------------------|-----------------|------------------|-----------------|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | |
| 0 | FBx | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | | + | - | x | ÷ | = | F10 ^x | Fe ^x |
| 2 | | Fπ | F√ | Fx ² | F ¹ / _x | Fx [#] | F○ | |
| 4 | | Π0 | Π1 | Π2 | Π3 | Π4 | Π5 | Π6 |
| 5 | | C/П | BП | B/O | ПП | KНОП | | |
| 6 | | ИΠ0 | ИΠ1 | ИΠ2 | ИΠ3 | ИΠ4 | ИΠ5 | ИΠ6 |
| 7 | | kx ≠ 00 | kx ≠ 01 | kx ≠ 02 | kx ≠ 03 | kx ≠ 04 | kx ≠ 05 | kx ≠ 06 |
| 8 | | КБΠ0 | КБΠ1 | КБΠ2 | КБΠ3 | КБΠ4 | КБΠ5 | КБΠ6 |
| 9 | | kx ≥ 00 | kx ≥ 01 | kx ≥ 02 | kx ≥ 03 | kx ≥ 04 | kx ≥ 05 | kx ≥ 06 |
| - | | KПП0 | KПП1 | KПП2 | KПП3 | KПП4 | KПП5 | KПП6 |
| L | | KП0 | KП1 | KП2 | KП3 | KП4 | KП5 | KП6 |
| | | kx < 00 | kx < 01 | kx < 02 | kx < 03 | kx < 04 | kx < 05 | kx < 06 |
| Г | | KИП0 | KИП1 | KИП2 | KИП3 | KИП4 | KИП5 | KИП6 |
| E | | kx = 00 | kx = 01 | kx = 02 | kx = 03 | kx = 04 | kx = 05 | kx = 06 |

(продолжение)

| Первый знак кода | Второй знак кода | | | | | | | |
|------------------------|------------------|---------|----------|----------|---------|---------|---------|--------|
| | 7 | 8 | 9 | - | L | | Г | E |
| 0 | 7 | 8 | 9 | , | /—/ | ВП | Cx | ↑ |
| 1 | Flg | Fln | F arcsin | F arccos | F arctg | F sin | F cos | F tg |
| 2 | | | | | | | | |
| 4 | Π7 | Π8 | Π9 | ΠA | ΠB | ΠC | ΠD | |
| 5 | Fx ≠ 0 | FL2 | Fx ≥ 0 | FL3 | FL1 | Fx < 0 | FL0 | Fx = 0 |
| 6 | ИΠ7 | ИΠ8 | ИΠ9 | ИΠA | ИΠB | ИΠC | ИΠD | |
| 7 | Kx ≠ 07 | Kx ≠ 08 | Kx ≠ 09 | Kx ≠ 0A | Kx ≠ 0B | Kx ≠ 0C | Kx ≠ 0D | |
| 8 | КБΠ7 | КБΠ8 | КБΠ9 | КБΠA | КБΠB | КБΠC | КБΠD | |
| 9 | Kx ≥ 07 | Kx ≥ 08 | Kx ≥ 09 | Kx ≥ 0A | Kx ≥ 0B | Kx ≥ 0C | Kx ≥ 0D | |
| - | KПП7 | KПП8 | KПП9 | KППA | KППB | KППC | KППD | |
| L | KП7 | KП8 | KП9 | KПA | KПB | KПC | KПD | |
| | Kx < 07 | Kx < 08 | Kx < 09 | Kx ≤ 0A | Kx < 0B | Kx < 0C | Kx < 0D | |
| Г | KИП7 | KИП8 | KИП9 | KИПA | KИПB | KИПC | KИПD | |
| E | Kx = 07 | Kx = 08 | Kx = 09 | Kx = 0A | Kx = 0B | Kx = 0C | Kx = 0D | |

ЛИТЕРАТУРА

1. А брамов С. А. Математические построения и программирование.— М.: Наука, 1978.
2. А брамов С. А. Элементы программирования.— М.: Наука, 1982.
3. А ндерсон Р. Доказательство правильности программ.— М.: Мир, 1982.
4. А хо А., Х опкрофт Дж., У льман Дж. Построение и анализ вычислительных алгоритмов.— М.: Мир, 1979.
5. Башмаков М. И., Беккер Б. М., Гольховой В. М. Задачи по математике. Алгебра и анализ.— М.: Наука, 1982.
6. Белый Ю. А. Считывающая микроЭлектроника.— М.: Наука, 1983.
7. Бернштейн И. М458.//Квант.— 1978.— № 6.
8. Болтянский В. Г. М781.//Квант.— 1982.— № 9.
9. Болтянский В. Г., Розендорн Э. Р. ХХI школьная математическая олимпиада в Москве.//Математическое просвещение.— 1961.— № 6.
10. Ван Тассел Д. Стиль, разработка, эффективность, отладка и испытание программ.— М.: Мир, 1981.
11. В ирт Н. Систематическое программирование.— М.: Мир, 1971.
12. В ишеньский В. А., Ядренко М. И. Вибрани математичні задачі.— Киев: Вища школа, 1974.
13. Воробьев Н. Н. Числа Фибоначчи.— М.: Наука, 1984.
14. Гардиер М. Математические головоломки и развлечения.— М.: Мир, 1971.
15. Гардиер М. Математические досуги.— М.: Мир, 1972.
16. Гардиер М. Математические новеллы.— М.: Мир, 1974.
17. Гардиер М. Есть идея! — М.: Мир, 1982.
18. Гильде В., Альтрихтер З. С микрокалькулятором в руках.— М.: Мир, 1982.
19. Гудман С., Хидетниеми С. Введение в разработку и анализ алгоритмов.— М.: Мир, 1981.
20. Гусев В. А., Орлов А. И., Розенталь А. Л. Внеклассная работа по математике в 6—8 классах.— М.: Просвещение, 1977.
21. Дал У., Дейкстра Э., Хоор К. Структурное программирование.— М.: Мир, 1975.
22. Демидович Н. Б., Монахов В. М. Программирование и ЭВМ.— М.: Просвещение, 1977.
23. Дейкстра Э. Дисциплина программирования.— М.: Мир, 1978.
24. Дьюдени Г. Э. 520 головоломок.— М.: Мир, 1975.
25. Ершов А. П., Звенигородский Г. А., Первин Ю. А. Школьная информатика (концепция, состояние, перспективы).— Новосибирск, 1979.
26. Игнатьев Е. И. В царстве смекалки.— М.: Наука, 1978.
27. Ионин Ю., Курляндчик Л. Поиск инварианта.//Квант.— 1976.— № 2.
28. Искусство программирования.//Квант.— 1979.— Начиная с № 9.

29. И одан Э. Структурное программирование и конструирование программ.— М.: Мир, 1979.
30. Ка саткин В. Н. Логическое программирование в занимательных задачах.— Киев: Техника, 1980.
31. Ка ц М., У лам С. Математика и логика. Ретроспектива и перспективы.— М.: Мир, 1971.
32. К нут Д. Искусство программирования для ЭВМ.— М.: Мир, 1976.— Т. I.
33. Ковалев М. П., Шварцбурд С. И. Электроника помогает считать.— М.: Просвещение, 1978.
34. Козин А. С., Лященко Н. Я. Вычислительная математика.— Киев: Радянська школа, 1983.
35. Кордемский Б. А. Математическая смекалка.— М.: Государственное издательство технико-теоретической литературы, 1954.
36. Крайль Г. Что умеет мой микрокалькулятор.— М.: Мир, 1981.
37. Майерс Г. Искусство тестирования программ.— М.: Финансы и статистика, 1982.
38. М ейер Б., Бодуэн К. Методы программирования.— М.: Мир, 1982.— Т. 1—2.
39. Минаева С. С. Вычисления на уроках и внеклассных занятиях по математике.— М.: Просвещение, 1983.
40. Монахов В. М., Лапчик М. П., Демидович Н. Б., Червочкина Л. П. Формирование алгоритмической культуры школьника при обучении математике.— М.: Просвещение, 1978.
41. Нивергельт Ю., Фаррар Дж., Рейнгольд Э. Машинный подход к решению математических задач.— М.: Мир, 1977.
42. Нильсон Н. Искусственный интеллект.— М.: Мир, 1973.
43. Оре О. Приглашение в теорию чисел.— М.: Наука, 1980.
44. Розов Н., Смолянский Н. Олимпиады по математике.//Квант.— 1978.— № 10.
45. Соболь И. М. Метод Монте-Карло.— М.: Наука, 1968.
46. Сойфер А. Ю. Две игры.//Квант.— 1972.— № 4.
47. Трахтенброт Б. А. Алгоритмы и вычислительные автоматы.— М.: Советское радио, 1974.
48. Трохименко Я. К., Любич Ф. Д. Инженерные расчеты на микрокалькуляторах.— Киев: Техника, 1980.
49. Тьмеладзе З. Теория игр.//Квант.— 1977.— № 8.
50. Условия задач вступительной контрольной работы в ЗМШ 1972 года.//Квант.— 1972.— № 1.
51. Уззерел Ч. Этюды для программистов.— М.: Мир, 1982.
52. Фомин С. М324.//Квант.— 1976.— № 1.
53. Фройденталь Г. Математика как педагогическая задача.— М.: Просвещение, 1982.
54. Цветков А. И. Прикладные программы для микроЭВМ «Электроника Б3-21».— М.: Финансы и статистика, 1982.
55. Цветков А. И., Епонечников В. А. Прикладные программы для микроЭВМ «Электроника Б3-34», «Электроника МК-56», «Электроника МК-54».— М.: Финансы и статистика, 1984.
56. Чакань А. Что умеет карманная ЭВМ? — М.: Радио и связь, 1982.

ОГЛАВЛЕНИЕ

| | |
|---|----|
| Предисловие | 3 |
| Глава I. 10 УРОКОВ ПО ПРОГРАММИРОВАНИЮ | |
| § 1.1. Урок 1. Операнды и операции | 5 |
| § 1.2. Урок 2. Оператор присваивания и ячейки памяти калькулятора | 11 |
| § 1.3. Урок 3. «Электроника Б3-34» — микроЭВМ | 17 |
| § 1.4. Урок 4. Что такое алгоритм | 22 |
| § 1.5. Урок 5. Ветвления | 27 |
| § 1.6. Урок 6. Циклы | 38 |
| § 1.7. Урок 7. Массивы и косвенная адресация | 46 |
| § 1.8. Урок 8. Польская запись. Трансляция выражений | 53 |
| § 1.9. Урок 9. Подпрограммы | 62 |
| § 1.10. Урок 10. Структурное программирование | 73 |
| Упражнения | 83 |

Глава II. ИГРА И КАЛЬКУЛЯТОР

| | |
|---|-----|
| § 2.1 Калькулятор выигрывает | 87 |
| § 2.2. Игра БАШЕ | 92 |
| § 2.3. Игра «ДВЕ КУЧКИ КАМНЕЙ» | 98 |
| § 2.4. Игра «ВПЕРЕД И ВВЕРХ» | 106 |
| § 2.5. Игра Р. ГАСКЕЛА И М. ВАНИГАНА | 115 |
| § 2.6. Калькулятор и случай | 122 |
| § 2.7. Игра «ПОТОПИ КОРАБЛЬ» | 129 |
| § 2.8. Игра «ВОВРЕМЯ ОСТАНОВИСЬ» | 134 |
| § 2.9. Матричные игры | 138 |
| Упражнения | 141 |
| Приложение. Тестирование и отладка программ | 152 |
| Литература | 158 |

Учебное издание

Грузман Михаил Зиновьевич

ЛОГИЧЕСКИЕ ИГРЫ С КАЛЬКУЛЯТОРОМ

Зав. редакцией Р. А. Хабиб. Редактор А. К. Компанец. Младшие редакторы Л. Е. Козырева, Е. А. Сафонова. Художественный редактор Е. Н. Карасик. Технический редактор Н. Н. Матвеева. Корректор Н. С. Соболева.

ИБ № 10297

Сдано в набор 07.05.87. Подписано к печати 28.11.88. Формат 60 × 90^{1/16}. Бумага офсетная № 2. Гарнитура литературная. Печать офсетная. Усл. печ. л. 10. Усл. кр.-отт. 10,5. Уч.-изд. л. 7,23. Тираж 200 000 экз. Заказ 1693. Цена 30 коп.

Ордена Трудового Красного Знамени издательство «Просвещение». Государственного комитета РСФСР по делам издательств, полиграфии и книжной торговли. 129846, Москва, 3-й проезд Марыиной рощи, 41.

Смоленский полиграфкомбинат Росглавполиграфпрома Государственного комитета РСФСР по делам издательств, полиграфии и книжной торговли. 214020, г. Смоленск, ул. Смольянинова, 1.

30 к.

