Глава 2

АЛГОРИТМЫ И АЛГОРИТМИЗАЦИЯ ЗАДАЧ

2.1. Понятие алгоритма. Свойства алгоритма

Алгоритмом называют строгую, полностью определенную последовательность действий с изменяемыми исходными данными, выполнив которую, получим искомый результат, решим поставленную задачу.

Пример 1.

Записать алгоритм вычисления функции:

$$F = (ax+b)/(cy).$$

При записи алгоритма будем использовать правила приоритета выполнения арифметических действий, операцию присваивания и вспомогательные переменные. Знак операции присваивания следующий: «:=». Запись a := 5 читают так: «пусть a равно 5» или «переменной a присвоим значение 5».

Запись алгоритма может быть такой:

1) z1 := ax;
2) z2 := z1+b;
3) z3 := cy;
4) F := z2/z3.

Использование вспомогательных переменных (z1, z2 и z3) упрощает записи алгоритмов вычисления сложных функций, делает их более обозримыми. В рассмотренной простой задаче они введены для примера.

Свойства алгоритмов

К свойствам алгоритмов относят: определенность, дискретность, результативность и массовость. Все эти свойства вытекают из определения алгоритма. Рассмотрим их подробнее.

Определенность ..... требует от алгоритма быть строгим, четким, понятным. Все действия, символы операций должны быть или общепринятыми, или заранее четко и однозначно определены. Не допускаются двусмысленности, неоднозначности. Например, последовательность действий: 1) y = a#b; 2) z = y@ — не алгоритм, так как операции со знаками # и @ не определены.

Дискретность ..... требует от алгоритма пошаговую запись и выполнение.

Результативность .... предполагает обязательное получение результата. При этом, как говорят, «отрицательный результат — тоже результат». Например, если компьютер выдает на экран дисплея предусмотренное алгоритмом решения сообщение «Решение невозможно ввиду отрицательного подкоренного выражения», то такой алгоритм обладает свойством результативности. Если же запуск на выполнение программы, записанной в соответствии с некоторым алгоритмом, приводит к бесконечным вычислениям, то с результативностью такого алгоритма не все в порядке.

Массовость ..... требует от алгоритма возможность применения его при различных значениях исходных данных, то есть предполагается, что алгоритм должен содержать переменные величины. Например, запись: 1) y := 3+5; 2) z := 2y нельзя считать алгоритмом, так как она не удовлетворяет свойству массовости.

Одна из задач предмета информатики заключается в том, чтобы проследить преобразование информации по цепочке:

задача → алгоритм → программа → → компьютер → результат решения.

Задача ..... формулируется на обычном разговорном языке, в виде формул, соотношений, зависимостей. Это этапы постановки и формализации задачи.

Алгоритм ..... разрабатывается и записывается одним из способов записи, на одном из формальных языков. Этап называют этапом разработки и записи алгоритма.

Программа ..... это тот же алгоритм, но записанный на языке, понятном компьютеру — языке программирования. Программа записывается на этапе программирования задачи.

Компьютер ..... переводит введенную программу с языка программирования на внутренний язык компьютера — язык машинных команд, и решает задачу. Это этап решения задачи.

Результат ..... выдается компьютером в предусмотренном программой виде и анализируется пользователем.

2.2. Способы записи алгоритмов

Выбор способа записи зависит от характера задачи. Алгоритм вычислительного характера можно записать формулой или последовательностью формул. Алгоритм заваривания чая удобно записать словами в пронумерованных пунктах. Алгоритм решения квадратного уравнения будет наиболее понятен при записи словами и формулами.

Из формальных способов записи алгоритмов чаще других будем использовать язык блок-схем и алгоритмический язык. Заметим, что программа также является записью алгоритма на языке программирования.

Запись алгоритмов словами

Словесная запись алгоритма наиболее проста, не требует строгих форматов, правил. Обычно используется запись пронумерованными пунктами. Например, запишем словами алгоритм решения известной детской логической задачи «Волк, коза, капуста и перевозчик».

Пример 2.

Постановка задачи следующая: «На левом берегу реки находятся волк, коза, капуста и перевозчик с лодкой. Перевозчик должен переправить всех на правый берег так, чтобы не оставлять наедине волка с козой и козу с капустой. Как это сделать?».

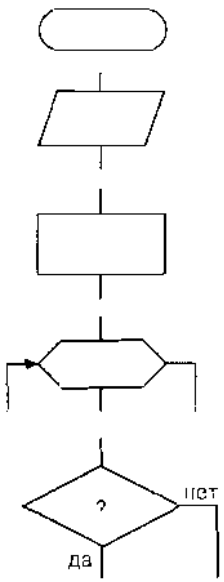
1. Начало алгоритма.
2. Переправить на правый берег козу, оставив на левом волка и капусту.
3. Вернуться на левый берег, оставив козу на правом берегу.
4. Переправить на правый берег капусту.
5. Вернуться на левый берег с козой.
6. Переправить на правый берег волка.
7. Вернуться на левый берег, оставив на правом волка и капусту.
8. Переправить на правый берег козу.
9. Все в сборе на правом берегу.
10. Конец алгоритма.

В качестве упражнений запишите словами алгоритм перехода улицы, алгоритм заварки чая, алгоритмы решения других задач.

Блок-схемы алгоритмов

Записи алгоритмов на языке блок-схем обладают большой наглядностью. Хорошо просматривается структура алгоритма. Блок-схема представляет собой соединенные линиями блоки различной конфигурации. Вид блоков и последовательность их соединения соответствуют типу и последовательности действий алгоритма.

Ограничимся сокращенным набором блоков (рис. 2.1).



Блок начала и конца алгоритма. В овале пишут слова «начало» и «конец».

Блок ввода и вывода информации. В блоке ввода перечисляют имена данных, подлежащих вводу в алгоритм, в блоке вывода — выводу из алгоритма.

«Процесс» решения задачи. В прямоугольнике блока записывают действия, которые нужно выполнить при решении задачи.

Блок модификации, изменения параметра. Используется, например, в блок-схемах циклических алгоритмов со счетчиком.

Блок «решение». Проверяет выполнение какого-либо условия: выход **да** — при выполнении условия, **нет** — при его невыполнении. **Да** и **нет** иногда заменяют на **1** и **0** или + и – соответственно.

Рис. 2.1. *Блоки блок-схем алгоритмов (начало)*

Алгоритмический язык

Алгоритмический язык — это язык, предназначенный для записи алгоритмов. Как и любой другой язык, он включает: набор символов (алфавит), правила записи алгоритмов (синтаксис) и правила истолкования записей (семантику).

Учебный алгоритмический язык использует русский алфавит, синтаксис и семантику наиболее применяемых языков программирования. Поэтому он является как бы ступенью для понимания и освоения многих языков программирования.

Алгоритмический язык ввиду своей распространенности, простоты и универсальности широко используется для записи алгоритмов решения учебных задач, а также различного рода учебных проверочных и контролирующих тестов. Заметим, что

правила записи алгоритмов на школьном алгоритмическом языке могут различаться, но это не затрудняет чтение и понимание алгоритмов.

Запись алгоритмов на алгоритмическом языке требует определенной строгости и четкости. Все служебные слова выделяются, например, подчеркиваются. Необходимо выдерживать правила смещения записей относительно друг друга. Это смещение подчеркивает структуру алгоритма, делает его легко читаемым, более понятным.

В обобщенной схеме одного из вариантов записи алгоритма на алгоритмическом языке названия записей взяты в треугольные скобки:

```
алг <название алгоритма> (<список переменных и их типы>)
   арг <список аргументов>
   рез <список результатов>
нач <ввод вспомогательных переменных>
   <присваивание начальных значений>
   <последовательность действий в соответствии с алгоритмом>
   выв <список данных, выводимых на печать >
кон
```

В информатике все величины по своему типу делятся на: числовые, литерные (строковые, символьные, текстовые) и логические. Среди числовых переменных различают:

- вещественные числа (обозначение — вещ X);
- целые числа (обозначение — цел X);
- натуральные числа (обозначение — нат X).

Переменные литерного типа обозначаются лит X и представляют собой символы или наборы символов (тексты, строки), заключенные в кавычки:

```
лит X := "вариант 5", лит S := "счастливый билет".
```

Логические переменные могут принимать только два значения: 1 — истина, и 0 — ложь. Будем их обозначать так: лог X.

2.3. Линейные алгоритмы

Различают три типа алгоритмов: **линейные, ветвящиеся и циклические**. Их названия определяются входящими в них типовыми алгоритмическими конструкциями, которые также называют **базовыми структурами**. К основным базовым структурам относят: **следование, ветвление и цикл**. Доказано, что этих трех основных базовых структур достаточно, чтобы построить алгоритм любой сложности.

Самые простые по структуре — линейные алгоритмы. Они не имеют ветвлений и циклов. В блок-схемах отсутствуют блоки «решение» и обратные связи, позволяющие многократно выполнять некоторые действия.

Пример 3.

Записать в виде блок-схемы и на алгоритмическом языке алгоритм вычисления объема V и площади P поверхности цилиндра, если известны радиус его основания R и высота H. Все аргументы вещественные числа.

Решение.

Введем обозначения: pi = 3,14, площадь основания P1, длина окружности основания L и запишем алгоритм решения задачи.

Запись линейного алгоритма в виде блок-схемы показана на рис. 2.2. Запись алгоритма на алгоритмическом языке следующая:

```
алг Площадь поверхности и объем цилиндра (вещ pi, R, H, V, P)
    арг R, H
    рез V, P
нач вещ P1
    P1 := pi · R²; V := p1 · H
    L := 2 · pi · R; P := 2 · P1 + L · H
    выв V, P
кон
```

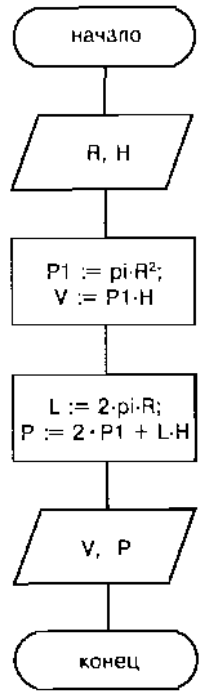


Рис. 2.2. *Блок-схема линейного алгоритма*

Задачи для самостоятельного решения

2.1. Поменять между собой значения двух переменных A и B, воспользовавшись для временного хранения третьей переменной C.

2.2. Поменять между собой значения двух переменных, не используя третью переменную.

2.3. Имеются две переменные A и B. Переменной A присвоить их сумму, а переменной B — их разность, не используя третью переменную.

2.4. Определить, чему будут равны переменные A и B в результате выполнения алгоритма: 1) A := 3; 2) B := 5; 3) A := 2A − B; 4) B := 5A − B; 5) A := A − A.

2.5. Два отрезка прямой на плоскости заданы координатами своих концов. Записать алгоритм определения суммы длин этих отрезков.

2.4. Ветвящиеся алгоритмы

Ветвящиеся алгоритмы содержат базовую структуру ветвления. Они содержат блок «решение», который может иметь два и более альтернативных выходов. При работе алгоритма в зависимости от выполнения условий выбирается один из этих выходов, и выполняются соответствующие ему действия. Ветвящиеся алгоритмы, как правило, включают в себя более простую базовую структуру — следование.

Пример 4.

Вычислить значение величины c, определяемое по формулам: c = a+b, если a ≤ b и c = a − b, если a > b.

Рассмотрим работу алгоритма решения задачи по его блок-схеме (рис. 2.3). Пусть блок 2 вводит значения данных a = 5, b = 2. Блок 3 проверяет условие a ≤ b. Условие не выполняется. Поэтому следующим выполняется действие в блоке 5: c = a − b = 5 − 2 = 3. Блок 6 выводит результат c = 3. Конец алгоритма.

Запись алгоритма на алгоритмическом языке следующая:

```
алг Задача. (вещ a,b,c)
    арг a, b
    рез c
нач
    если a ≤ b
       то c = a + b
       иначе c = a — b
    все
    выв c
кон
```

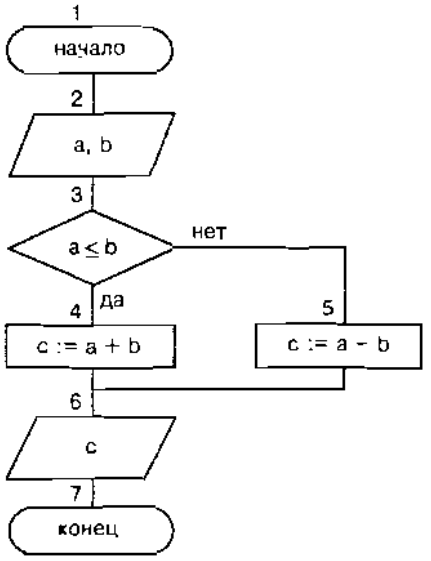


Рис. 2.3. *Блок-схема ветвящегося алгоритма*

Проследите работу алгоритма при a = b = −4, a = 3 и b = −3, a = b = 0, других значениях исходных данных.

Пример 5.

Выполним алгоритмизацию шуточной задачи «Счастливый билет».

Постановка задачи.

Автобусный билет считают «счастливым», если сумма трех первых цифр номера билета равна сумме трех последних цифр. Требуется записать алгоритм определения билета, имеющего счастливый номер.

Формализация задачи.

Введем обозначения и запишем основные соотношения.

Пусть цифры билета: a, b, c, d, e, f.

Суммы цифр: C1 = a+b+c; C2 = d+e+f.

Кроме того, обозначим: S := «билет счастливый», N := «билет несчастливый», R — результат.

В итоге имеем: $\begin{cases} S, \text{ если } C1 = C2; \\ N, \text{ если } C1 \neq C2. \end{cases}$

Запись алгоритма на алгоритмическом языке следующая:

```
алг Счастливый билет (цел a, b, c, d, e, f, лит R)
    арг a, b, c, d, e, f
    рез R
нач цел C1, C2, лит S, N
    C1 := a+b+c; C2 := d+e+f
    лит S = "билет счастливый"; лит N = "билет несчастливый"
    если C1 = C2
       то R := S
       иначе R := N
    все
    выв R
кон
```

При записи блок-схемы (рис. 2.4) показан способ отображения связи блоков при переносах части блоков на другой лист или другую страницу:

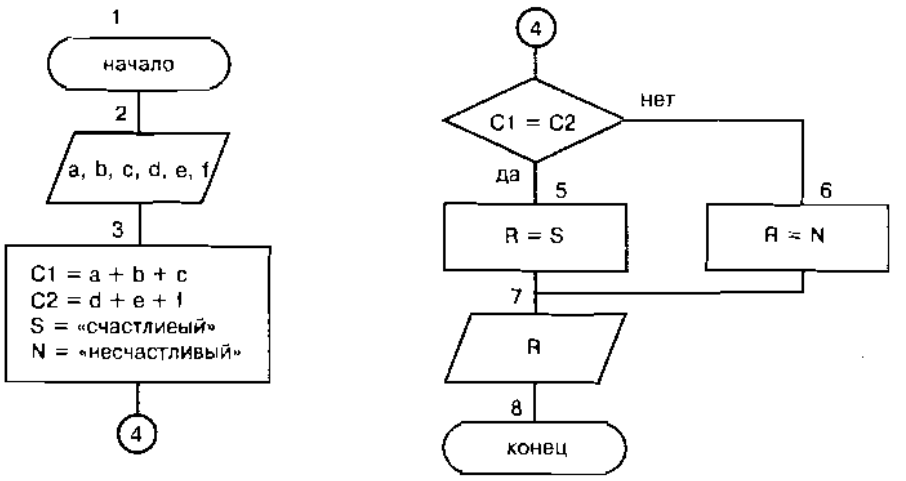


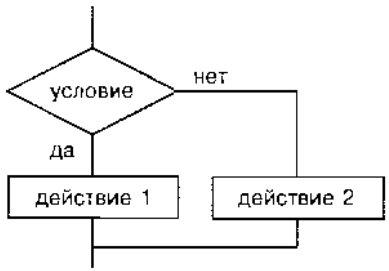
Рис. 2.4. *Блок-схема алгоритма «Счастливый билет»*

Полная и сокращенная формы ветвления

В обычном разговорном языке в условном предложении «если не будет дождя, то мы нойдем гулять, иначе будем читать книги» содержится два действия, из которых выполняется только одно. Условное нредложение «если не будет дождя, то мы пойдем гулять» содержит одно действие. Первое нредложение соотаетствует нолной форме ветвлепия, второе — сокращенной форме.

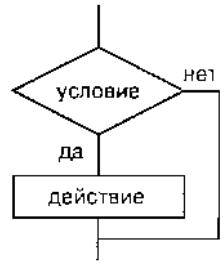
Запись блок-схемой и на алгоритмическом языке полной и сокращеппой форм аетвления показапы на рис. 2.5.

Полная форма ветвления




```
если [условие]
   то [действие 1]
   иначе [действие 2]
все
```

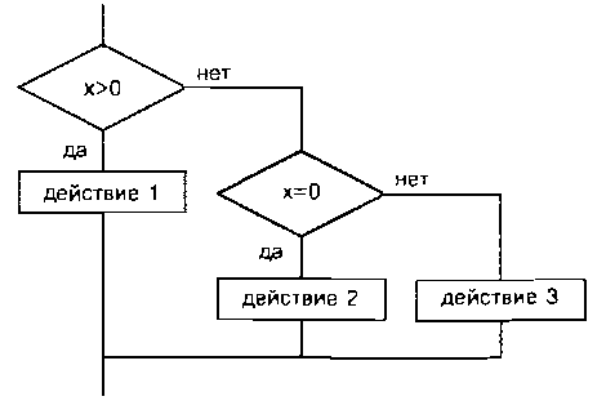
Сокращеннвя форма ветвления




```
если [условие]
   то [действие]
все
```

Рис. 2.5. *Полная и сокращенная формы ветвления*

Нередко в задачах проверяются условия, соответствующие трем и более выходам. Например, если вынолнение условий x>0, x = 0, x<0 требуют трех различных действий, то структура ветвления может быть такой, как показано на рис. 2.6.




```
если x>0
   то [действие 1]
иначе если x=0
         то [действие 2]
         иначе [действие 3]
      все
все
```

Рис. 2.6. *Вариант построения ветвления на три выхода*

Если число альтернатив больше трех, то в таких случаях, как правило, используют конструкцию «выбор» или «выбор-иначе».

Ветвление типа «выбор»

```
выбор
   при условие 1: действия 1
   при условие 2: действия 2
   . . .
   при условие N: действия N
все
```

Ветвление типа «выбор-иначе»

```
выбор
   при условие 1: действия 1
   при условие 2: действия 2
   . . .
   при условие N: действия N
   иначе действия N+1
все
```

В соответствии с рассмотренными на алгоритмическом языке вариантами ветвлений изображаются и блок-схемы. Ниже приведен пример записи и блок-схема (рис. 2.7) алгоритма решения задачи для варианта «выбор-иначе».

```
выбор
    при 1-е место: золотая медаль
    при 2-е место: серебряная медаль
    при 3-е место: бронзоеая медаль
    иначе грамота участника
все
```

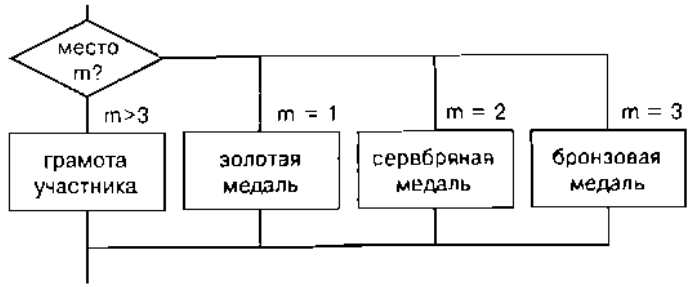


Рис. 2.7. *Блок-схема алгоритма «выбор-иначе»*

Применение сложных условий

Условия, имеющие знаки отношения (>, <, = , ≤, ≥) рассматриваются как логические переменные, которые могут принимать значение ИСТИНА, если условие выполняется, или ЛОЖЬ в противном случае. В блок-схемах выходы блока «решение», соответствующие выполнению или невыполнению условия соответственно помечаются словами и знаками **да** (1, +) — истина и **нет** (0, –) — ложь.

Алгоритмы многих задач записываются намного проще, если в них используются сложные, составные условия. Для связи простых условий будем применять служебные слова И, ИЛИ, НЕ. Например, если требуется задать проверку условия нахождения величины z в границах отрезка [a, b], то это можно записать так:

```
если z≥a И z≤b
    то z находится в границах отрезка [a, b]
все
```


Если, например, нужно проверить находится ли z в хотя бы одном из отрезков [a, b] и [c, d], то запись будет следующей:

```
если z≥a И z≤b ИЛИ z≥c И z≤d
    то z находится в одном из отрезков [a, b] или [c, d]
все
```

Рассмотрим пример решения задачи с использованием сложного условия.

Пример 6.

Заданы длины a, b и c трех отрезков прямой (a, b и c — целые, положительные числа). Можно ли использовать эти отрезки в качестве сторон треугольника?

Решение.

Условие, при выполнении которого треугольник построить нельзя следующее:

```
a>b+c ИЛИ b>a+c ИЛИ c>a+b
```

Поэтому алгоритм решения задачи можно записать так:

```
алг Построение треугольника (цел a, b, c, лит z)
    арг a, b, c
    рез z
нач
    если НЕ (a>b+c ИЛИ b>a+c ИЛИ c>a+b)
то z := "Можно"
иначе z := "Нельзя"
    все
    вын z
кон
```

Задания для самостоятельной работы

Замечание. При решении задач желательно придерживаться рассмотренной последовательности действий. После *постановки задачи* следует этап *разработки алгоритма* ее решения, который совмещается с *формализацией задачи*. При формализации вводятся обозначе-

ния, присваиваются имена переменным, константам, записываются расчетные формулы, соотношения. При хорошей формализации задачи увеличивается вероятность ее правильного решения. К тому же, без соответствующих определений значения введенных обозначений со временем забываются и на восстановление их смысла требуется время. Четкая и подробная формализация задачи упрощает запись алгоритма. На этапе *записи алгоритма* также следует строго придерживаться рассмотренных выше правил.

Записать алгоритмы решения следующих задач.

2.6. Заданы три натуральных числа. Вывести «Нет», если среди чисел нет равных, «Да», если есть одна пара равных и «Все равны», если все числа равны.

2.7. Известны стороны двух треугольников a, b, c и d, e, f. Определить, какой треугольник (первый или второй) имеет большую площадь.

2.8. Заданы три положительных числа a, b и c. Определить, являются ли они последовательно стоящими элементами арифметической или геометрической прогрессии. Если да, то напечатать значение разности (знаменателя) прогрессии.

2.9. Заданы три стороны треугольника x, y и z. Определить, является ли треугольник прямоугольным. Если да, то отпечатать какая сторона служит гипотенузой.

2.10. Заданы длины a, b, c и d четырех отрезков прямой. Проверить, могут ли эти отрезки быть сторонами квадрата, прямоугольника.

2.5. Структуры данных. Массивы

Структуры данных определяют классификацию данных и отношения между ними. Структуры могут быть простыми (элементарными) и сложными (составными). Различают следующие структуры: константу, переменную, массив, запись и таблицу. Константу и переменную можно считать элементарными данными, единичными, неделимыми элементами более сложных систем организации данных.

Константа ..... это число, текст или логическое значение, которые не изменяются в процессе реализации алгоритма, решения задачи на компьютере.

Переменная ..... это единица организации данных, которой в процессе обработки информации могут присваиваться различные значения. Переменная имеет имя — **идентификатор** и тип: числовой, символьный (литерный, строковый), логический. Переменные могут сопровождаться словами, указывающими на их тип.

Массив ..... это объединенное одним именем (идентификатором массива) множество однотипных элементов. К основным параметрам массивов относят его тип (числовой, символьный, логический), размерность (одномерный, двухмерный и т.д.) и размер (количество элементов массива в каждом измерении).

Вид записи массива на различных языках может различаться. Например, массив, задающий значения роста учащихся некоторого класса, имеющего N учеников на алгоритмическом языке может быть задан одномерным массивом так: нат R[1:N]. Величина N — максимальный номер элементов массива. Этот же массив на языке программирования Qbasic записывается так: r(n) или r(1 TO n).

Элементами массива являются переменные с номерами (индексами). Имена переменных совпадают с именем массива. Пусть задан массив роста каждого ученика класса. Тогда массив R[1:N] можно раскрыть до элементов так:

R[1:N] = [156, 162,...,R[i],..., 164];

R[1:N] = [R[1], R[2],..., R[i],..., R[N]],

где индексы (в квадратных скобках) определяют номера элементов массива; R[i] — любой (i-й) элемент массива.

В примере рассмотрен одномерный (линейный) массив. Примером двухмерного массива может служить таблица умножения нат T[1:9,1:9]. В нем каждый элемент T[i, j] равен произведению индексов. Индексы в двухмерном массиве определяют положение

элемента в таблице: i — номер строки, j — номер столбца. Заметим, что не только двухмерный, но и одномерный (линейный) массив иногда называют таблицей и на алгоритмическом языке записывают таб цел G[1:N].

Запись ..... это такая структура организации данных, которая позволяет объединять данные разных типов. Элементами записи являются поля. Запись и поля записи имеют имена. Каждое поле может быть переменной, массивом, записью более низкого уровня. Примером записи может служить одна строка таблицы.

Таблица может быть определена как объединение записей (строк) или как объединение массивов (столбцов). При этом массивы могут быть разного типа. В нижеприведенной таблице собраны сведения об учениках класса: номер по порядку, фамилия, год рождения, рост, вес и участие в спортивной жизни школы (1 — да, 0 — нет).

Таблица 2.1

N	*Фамилия F[1:N]*	*Год рожд. G[1:N]*	*Рост R[1:N]*	*Вес W[1:N]*	*Спорт S[1:N]*
1	*Иванов*	*1984*	*156*	*52.2*	*0*
2	*Петров*	*1983*	*162*	*61.5*	*1*
...	...	...	...	...	...
N	*Сидоров*	*1984*	*164*	*59.4*	*0*

Показанная таблица состоит из шести столбцов. Каждый столбец — массив (поле). Перечислим их и запишем на алгоритмическом языке.

Первый массив — номера учеников в списке. Это натуральный ряд чисел нат N[1:N].

Второй — фамилии учеников — массив литерных (строковых) величин лит F[1:N].

Третий и четвертый массивы — год рождения и рост — массивы целых чисел цел G[1:N] и цел R[1:N].

Пятый массив — вес учеников — массив вещественных чисел вещ W[1:N].

Последний, шестой массив — занятие спортом. Это массив логических величин лог S[1:N].

Строки таблицы занимают записи. Например, запись во второй строке таблицы следующая:

"нат N[2] лит F[2] цел G[2] цел R[2] вещ W[2] лог S[2]",

что соответствует

"2 Петров 1983 162 61.5 1".

2.6. Циклические алгоритмы

Циклические алгоритмы содержат базовую структуру — **цикл**. Они могут также включать участки, характерные как для линейных, так и ветвящихся алгоритмов. От линейных и ветвящихся алгоритмов циклические алгоритмы отличаются наличием в структуре алгоритма **обратной связи**, которая позволяет некоторые действия повторять многократно, циклически. Эти действия составляют тело цикла.

Циклические алгоритмы делятся на алгоритмы с известным и неизвестным заранее числом повторений. Их условно называют соответственно алгоритмы с циклом типа «для» и алгоритмы с циклом типа «пока». В обоих случаях окончание циклического процесса определяется поставленным заранее условием.

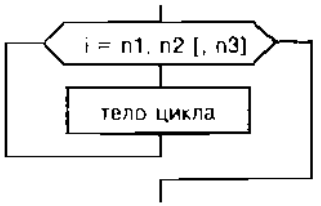
Циклы типа «для»

В циклических алгоритмах типа «для» этим условием служит явно заданное количество повторений. Блок-схемой и на алгоритмическом языке такой тип базовой структуры «цикл» записывается в общем виде, как показано на рис. 2.8.

При решении задачи определения суммы четных натуральных чисел от 2 до 100 структура цикла типа «для» может быть записана, как показано на рис. 2.9.

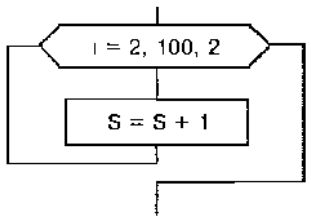
В примере использован блок модификации параметра. В общем виде запись внутри блока следующая:

<параметр> = <начальное значение>, <конечное значение>, [<шаг изменения>]




```
для i от n1 до n2 [шаг n3]
нц
    выполнение действий,
    входящих в тело цикла
кц
```

Рис. 2.8. *Структура цикла типа «для»*




```
для i от 2 до 100 шаг 2
нц
    S = S + i
кц
```

Рис. 2.9. *Пример записи цикла типа «для»*

Запись i = 2, 100, 2 указывает на то, что параметр i изменяется от начального значения 2 до конечного значения 100 с шагом 2, то есть последовательно принимает четные значения 2, 4, 6, ..., 100.

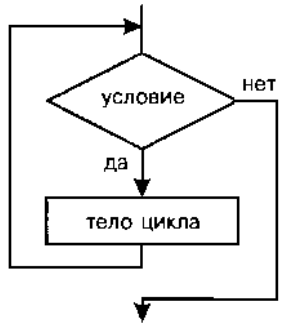
Заметим, что шаг равный единице не записывается, он принимается «по умолчанию».

Циклы типа «пока»

В циклических алгоритмах с неизвестным заранее числом повторений (циклы тина «нока») явно число новторений не задано, как, например, в задаче с такой постаноакой: «Сколько нужно взять, начиная с единицы, носледовательно расположенных чисел натурального ряда, чтобы их сумма нревысила 1000?».

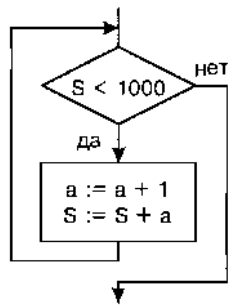
В виде блок-схемы и на алгоритмическом языке в общем виде структуру цикла типа «пока» можно записать, как показано на рис. 2.10.

При решении задачи онределения количества натуральных чисел, начиная с единицы, сумма которых превысит 1000, структура цикла «нока» может быть записана, как показано на рис. 2.11.




```
пока <условие>
нц
    выполнение действий,
    входящих в тело цикла
кц
```

Рис. 2.10. *Структура цикла типа «пока»*




```
пока S < 1000
нц
    a := a + 1
    S := S + a
кц
```

Рис. 2.11. *Пример записи алгоритма типа «пока»*

Характерными примерами задач, требующих для своего решения нростроения циклических алгоритмов, служат различные задачи обработки массиаов. Счетчик циклов (счетчик повторений), работающий по формуле i := i+1 аключается в блок модификации цараметра и может последовательно перебрать все индексы — номера элементоа обрабатываемого массивв. Перебрать все номера требуется, например, для получения суммы (S) элементов (R[i]) массива R[1:N], что получаем в цикле но формуле S := S+R[i].

Перед пачалом цикла сумме и другим, изменяемым в нроцессе работы алгоритма неремепным присваиваются начальные значения.

Цикл типа «для» со счетчиком

Рассмотрим задачу, требующую построения циклического алгоритма с известным заранее числом новторений — цикл типа «для» со счетчиком.

Пример 7.

Известен рост снортсменов стартовой нятерки баскетбольной команды: 195, 205, 202, 198, 192. Требуется разработать и записать алгоритм онределения среднего роста снортсменов.

Формализация задачи.

Для решения задачи требуется ностроить циклический алгоритм с заранее известным числом циклов. Это цикл типа «для», цикл «со счетчиком».

Рост спортсменов представим в виде одномерного массива целых чисел (линейной таблицы):

цел таб R[1:5] =
цел таб R[195, 205, 202, 198, 192] =
цел таб R[R[1], R[2], R[3], R[4], R[5]].

Введем всномогательную переменную S. Это сумма элементов массива, которую после выхода из цикла разделим на число снортсменов и получим искомый результат.

В алгоритме будем использовать следующие формулы:

S := 0 ..... задание начальных условий, нрисвоение начальных значений;

S := S+R[i].... наканливание суммы элементов массива, в каждом цикле новая сумма равна старой сумме плюс величина очередного, i-го элемента массива;

SR := S/5..... определение среднего роста баскетболистов.

В блок-схемс алгоритма решения задачи на рис. 2.12 буквами «ос» обозначена линия обратной связи, обеспечиающая цикличность процесса вычислеиий.

Для проверки условия окончания циклического процесса использован блок 4 модификации параметра i.

Занись алгоритма на алгоритмическом языке следующая:

```
алг Средний рост (цел таб R[1:5], вещ SR)
   арг R
   рез SR
нач цел S
   S := 0
   для i от 1 до 5
   нц
      S := S+R[i]
   кц
   SR := S/5
   выв SR
кон
```

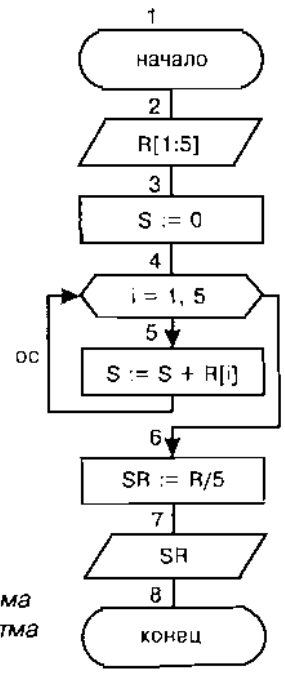


Рис. 2.12. *Блок–схема циклического алгоритма «Средний рост»*

Для паглядности работу алгоритма можно развернуть, как показано на рис 2.13:

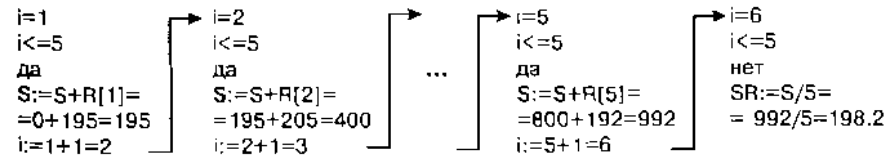


Рис. 2.13. *Развернутый по циклам алгоритм «Средний рост»*

В такой записи значение параметра i соответствует порядковому номеру повтора. Ниже следует проверка условия продолжения циклов. Если условие выполняется, то выполняются действия в теле цикла: накопление суммы и увеличение на единицу значения счетчика. Стрелкой показана обратная связь, обеспечивающая циклический процесс. После пятого цикла значение параметра (i = 6) обеспечивает выход из цикла, после чего вычисляется среднее значение роста баскетболистов.

Алгоритм решения следующей задачи — циклический типа «пока», с неизвестным заранее числом циклов, циклический алгоритм «без счетчика».

Пример 8.

Заданы два целых числа A и B. Определить их наибольший общий делитель.

Для определения наибольшего общего делителя (НОД) двух чисел используют алгоритм Евклида, один из вариантов которого можно записать следующей последовательностью пунктов:

1. Начало алгоритма.
2. Сравниваем числа A и B по величине. Если A ≠ B, то переход к п3, иначе к п4.
3. Если A>B, то A = A − B, иначе B = B − A. Переход к п2.
4. НОД = A.
5. Конец алгоритма.

Работу алгоритма покажем на конкретном примере. Пусть A = 55, B = 15. Шаги работы алгоритма оформим в виде табл. 2.2.

Таблица 2.2

A	55	40	25	10	10	5
B	15	15	15	15	5	5

Анализ алгоритма показывает, что это циклический алгоритм с заранее неизвестным числом циклов. Выход из цикла осуществляется по условию равенства чисел A и B.

Блок-схема алгоритма показана на рис. 2.14. Запись алгоритма Евклида на алгоритмическом языке следующая:

```
алг Алгоритм Евклида (цел A, B, НОД)
   арг A, B
   рез НОД
   нач
   пока A ≠ B
   нц
      если A>B
         то A = A − B
         иначе B = B − A
      все
   кц
   НОД = A
   выв НОД
кон
```

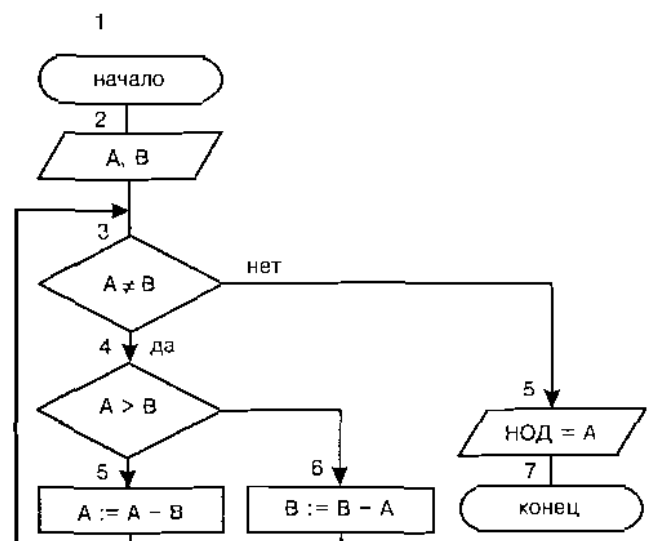


Рис. 2.14. *Блк-схема алгоритма Евклида*

Перед тем, как предложить варианты заданий для самостоятельной работы по закреплению материала, запишем алгоритм решения одного из вариантов типовых задач. Варианты даны в конце главы.

Пример 9.

Задан массив вещественных чисел A[0:N]. Записать алгоритм определения среднего арифметического положительных элементов и количества нулевых элементов массива.

Формализация задачи

Введем следующие обозначения:

S1 и K1 ..... соответственно сумма и количество положительных чисел массива (A[i]>0); вычисляются по формулам S1 = S1+A[i] и K1 = K1+1;

SR = S1/K1 . среднее арифметическое положительных чисел A[i]>0;

K0..... количество нулевых элементов массива; K0 = K0+1, если A[i] = 0.

Блок-схема алгоритма приведена на рис. 2.15. Запись алгоритма на алгоритмическом языке показана ниже.

```
алг Задача (вещ таб A[0:N], вещ SR, цел N, K0)
    арг N, A[0:N]
    рез SR, K0
нач вещ S1, цел K1, i, S1 := 0, K1 := 0, K0 := 0
    для i от 0 до N
    нц
        если A[i]>0
            то S1 := S1+A[i]; K1 := K1+1
        все
        если A[i] = 0
        то K0 := K0+1
        все
    кц
    SR := S1/K1
    выв SR, K0
кон
```

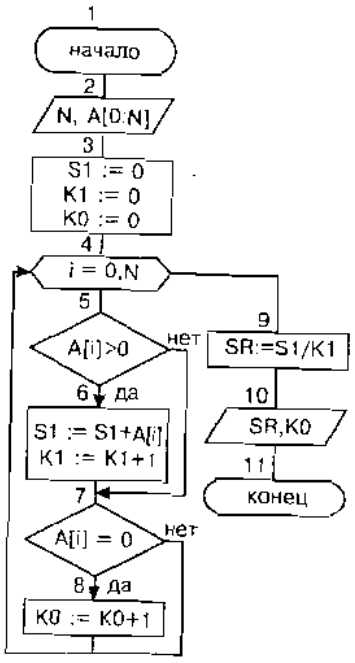


Рис. 2.15. *Блок-схема алгоритма*

Задания для самостоятельной работы

2.11. Составьте таблицу вариантов проверочной работы по записи циклических алгоритмов обработки массивов. Строки таблицы — номера вариантов. Столбцы соответствуют пунктам задания. В качестве примера покажем первый вариант работы (табл. 2.3).

Таблица 2.3

Вариант задания

	A[0:N]	B[1:M]	S>0	S<0	S все	K>0	K<0	K=0	K<C	SR все	SR>0
1	x		x				x	x			x

Первые две колонки задания отмечены именами массивов с различным заданием размерности. Начиная с третьей колонки, записаны задания: найти сумму положительных элементов массива, сумму отрицательных, сумму всех, количество положительных, отрицательных, равных нулю, меньших заданного числа C, среднее арифметическое всех, среднее положительное положительных. Задания, которые вошли в первый вариант, отмечены крестиками.

Число заданий можно увеличить, добавив, например, изменение знака отрицательных, положительных чисел, определение количества чисел в массиве больших заданного числа C, равных ему, определение среднего арифметического отрицательных чисел, рассмотрения в массиве элементов только с четными или нечетными номерами и т. д. Желательно в каждом варианте кроме задания массива иметь не более трех-пяти заданий (крестиков).

2.12. В составленной таблице выберите вариант работы и решите его. Вариант задачи, показанной в примере, может быть записан следующим образом: в массиве целых чисел A[0:N] определить сумму положительных, количество отрицательных, количество равных нулю элементов и среднее арифметическое положительных элементов массива. Алгоритм решения записать в виде блок-схемы и на алгоритмическом языке.