

§ 7

Запись алгоритмов на языках программирования

Язык программирования — формальная знаковая система, предназначенная для записи компьютерных программ.

Компьютерную программу можно считать последовательностью строк символов некоторого алфавита. Современные системы программирования допускают использование визуальных элементов (окон, иконок и др.) для построения программ, в частности для создания интерфейса пользователя. Такое программирование называют визуальным. Тем не менее основная, алгоритмическая часть любой программы строится с использованием символьных средств.

В основной школе вы могли познакомиться со Школьным алгоритмическим языком системы КуМир, языком программирования Pascal (Паскаль) или Python (Питон). И сейчас мы продолжим работать с этими языками.

Желательно установить среду программирования на ваш домашний компьютер.

www

Среда программирования	Сайт с программным обеспечением
Pascal ABC.Net	https://go.prosv.ru/pascalabc
Python	https://go.prosv.ru/Python

7.1. Структурная организация данных

Информация, представленная в виде, пригодном для автоматизированной обработки, называется **данными**. Компьютер оперирует только одним видом данных — отдельными битами (двоичными цифрами). Причём он работает с этими данными в соответствии с набором алгоритмов, которые определяются системой команд центрального процессора.

Задачи, которые решаются с помощью компьютера, редко выражаются на языке битов. Как правило, данные группируются для предоставления чисел, символов, текстов и более сложных структур. Алгоритмы, создаваемые для обработки этих данных, учитывают их структуру.



Под **структурой данных** в общем случае понимают множество элементов данных и множество связей между ними.

Различают простые и сложные структуры данных. Простые структуры не могут быть разделены на составные части, большие чем бит. К ним относятся числовые, символьные, логические и другие данные. Простые структуры служат основой для построения сложных структур — массивов, списков, графов, деревьев и др.

В языках программирования понятие «структуры данных» тесно связано с понятием «типы данных». Любые данные, т. е. константы, переменные, значения функций или выражения, характеризуются своими типами (рис. 2.10). Каждый тип однозначно определяет:

- 1) множество допустимых значений, которые может иметь тот или иной объект описываемого типа;
- 2) множество допустимых операций, которые применимы к объекту описываемого типа;
- 3) объём выделенной памяти для хранения данных указанного типа.

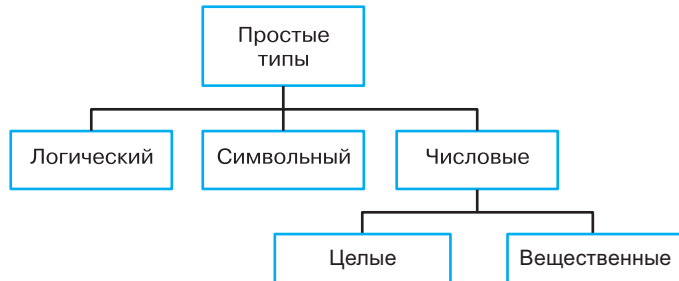


Рис. 2.10. Некоторые простые типы данных

7.2. Некоторые сведения о языках программирования

Основными элементами любого языка программирования являются:

- алфавит языка (латинские буквы, арабские цифры, специальные символы);
- служебные слова, значение которых в языке программирования строго определено;

- постоянные и переменные величины;
- знаки операций (табл. 2.2);
- стандартные функции;
- выражения;
- операторы (языковые конструкции, с помощью которых в программах записываются действия, выполняемые над данными в процессе решения задачи).

Все величины имеют **имена** (идентификаторы), формируемые по определённым правилам:

- имя может состоять из буквы или последовательности букв латинского алфавита, цифр и символа подчёркивания, но начинаться такая последовательность должна с буквы или символа подчёркивания;
- желательно, чтобы имя отражало смысл величины;
- имя не должно совпадать ни с одним из служебных слов.



Таблица 2.2

Операции в языках Pascal и Python

Арифметические операции	Pascal	Python
Сложение		+
Вычитание		-
Умножение		*
Деление		/
Целочисленное деление	div	//
Остаток от целочисленного деления	mod	%
Операции отношения	Pascal	Python
Равно	=	==
Не равно	<>	!=
Больше		>
Меньше		<
Больше или равно		>=
Меньше или равно		<=

Окончание табл. 2.2

Логические операции	Pascal	Python
Логическое отрицание	not	
Логическое И	and	
Логическое ИЛИ	or	
Строковые операции	Pascal	Python
Конкатенация (сцепление, присоединение)	+	

Выражение — языковая конструкция для вычисления значения. Выражение может состоять из операндов (констант, переменных, стандартных функций), знаков операций и круглых скобок. Выражения записываются в строку; знаки операций не пропускаются. Порядок выполнения операций определяется скобками и приоритетом операций (табл. 2.3). Операции одинакового приоритета выполняются слева направо, если порядок выполнения не задан явно круглыми скобками. Вычисление выражения с вложенными скобками начинается с внутренних скобок.

Таблица 2.3

Приоритет операций

Приоритет	Операция
1	not
2	*, /, div (//), mod (%), and
3	+, -, or
4	= (==), <> (!=), >, <, >=, <=

Данные, обрабатываемые компьютером, хранятся в памяти. С точки зрения языка программирования память разделена на секции, называемые **переменными**. Каждая переменная имеет **имя**, **тип** и **значение**; значения переменных могут меняться в ходе выполнения программы.

Программа на языке программирования представляет собой последовательность операторов (инструкций, команд). **Оператор** — языковая конструкция, представляющая собой один шаг из последовательности действий или набор описаний (табл. 2.4).

Таблица 2.4

Основные операторы

Pascal	Python
Присваивание	
<code>a := b</code>	<code>a = b</code>
Ввод с клавиатуры	
<code>read(a)</code>	Ввод строки: <code>input()</code> Ввод целого числа: <code>int(input())</code> Ввод вещественного числа: <code>float(input())</code>
Вывод на экран	
<code>write(a)</code>	<code>print(a)</code>
Условный оператор	
<code>if <условие></code> <code>then <блок операторов 1></code> <code>else <блок операторов 2></code>	<code>if <условие>:</code> <блок операторов 1> <code>else:</code> <блок операторов 2>
Цикл с предусловием	
<code>while <условие> do</code> <тело цикла>	<code>while <условие>:</code> <тело цикла>
Цикл с параметром	
<code>for <параметр> :=</code> <нач_знач> <code>do <кон_знач> do</code> <code>begin</code> <операторы> <code>end</code>	<code>for <параметр> in range(n):</code> <тело цикла> <code>for <параметр> in range(k, n):</code> <операторы> <code>for <параметр> in range(k, n, m):</code> <операторы>

Пример 1. В начале этой главы мы обсуждали алгоритмы нахождения простых чисел. Напишем программу, проверяющую, является ли заданное натуральное число n простым.

Самый простой путь решения этой задачи — проверить, имеет ли данное число n ($n \geq 2$) делители на отрезке $[2; n - 1]$.

Если делители есть, число n — составное; если их нет, то — простое.



В программе будем использовать логическую переменную `flag`:

- если `flag = true`, то n — простое число;
- если `flag = false`, то n — составное число (если у числа n есть делители, то «флаг выключаем» с помощью оператора присваивания `flag := false`).

Программа на языке Pascal

```
var
  n, i: longint;
  flag: boolean;
begin
  writeln('Введите n');
  read(n);
  flag := true;
  for i := 2 to n - 1 do
    if n mod i = 0 then flag := false;
  if flag then writeln('Да') else writeln('Нет')
end.
```

Программа на языке Python

```
print ('Введите n')
n = int(input())
flag = True
for i in range (2, n):
    if n % i == 0: flag = False
print ('Да') if flag else print('Нет')
```



Внимание! В программах на языке Python отступы обязательны.



В этой программе мы проверяли, нет ли у числа n делителей из интервала $[2; n - 1]$. Но если $n = a \cdot b$, то меньшее из чисел a , b не больше \sqrt{n} (в противном случае оба числа были бы больше \sqrt{n} , а следовательно, их произведение было бы больше n). Кроме того, из делимости числа n на a автоматически следует, что n делится и на n/a .

Усовершенствуйте приведённую выше программу с учётом этих соображений.

Проверку, является ли заданное натуральное число $n \geq 2$ простым, мы осуществили методом перебора всех возможных его делителей. Метод перебора используется для решения достаточно широкого круга задач.

Пример 2. Применим метод перебора для поиска наибольшего общего делителя (НОД) двух натуральных чисел a и b .

Начнём перебор с d — наименьшего из чисел a и b . Это первый, очевидный кандидат на роль их наибольшего общего делителя. И далее, пока не найдём d , на которое оба числа делятся нацело, будем уменьшать d на единицу. Как только такое деление выполнится, останавливаем уменьшение d . Полученное значение d и будет наибольшим общим делителем чисел a и b .



Программа на языке Pascal

```
var
  a, b, d: integer;
begin
  write('Введите два числа: ');
  readln(a, b);
  if a < b then d := a
  else d := b;
  while (a mod d <> 0) or (b mod d <> 0) do
    d := d - 1;
  write('НОД = ', d)
end.
```

Программа на языке Python

```
print('Введите два числа: ');
a = int(input())
b = int(input())
if a < b:
    d = a
else:
    d = b
while (a % d != 0) or (b % d != 0):
    d = d - 1
print('НОД =', d)
```

7.3. Анализ программ с помощью трассировочных таблиц

Для анализа свойств алгоритма и проверки его соответствия решаемой задаче используются **трассировочные таблицы**. В них фиксируется пошаговое исполнение алгоритма (программы), что позволяет наглядно представлять значения переменных, изменяющиеся при его выполнении. Поэтому трассировочные таблицы иначе называют таблицами значений.

Используются трассировочные таблицы двух видов:

- 1) таблицы, каждая строка которых отражает результат выполнения одного действия;
- 2) таблицы, каждая строка которых отражает результат выполнения группы действий.



Пример 3. Определим значения переменных *a* и *b*, полученные в результате выполнения следующей программы, записанной на двух языках программирования:

Программа на языке Pascal	Программа на языке Python
<pre> var a, b: integer; begin a := 5; b := 1; while b <= a do begin b := b + 1; a := a - 1; end; writeln(a); writeln(b) end.</pre>	<pre> a = 5 b = 1 while b <= a: b += 1 a -= 1 print(a) print(b)</pre>

Составим трассировочную таблицу первого вида. В её заголовке поместим имена всех переменных, используемых в программе. В отдельном столбце будем записывать команды и условия, имеющиеся в программе. Каждая строка таблицы соответствует одному шагу алгоритма. Чтобы не загромождать таблицу, будем записывать в каждой строке только то значение переменной, которое получено на соответствующем шаге.

№ шага	Команда или условие		Значение выражения	a	b
	Pascal	Python			
1	a := 5	a = 5	5	5	
2	b := 1	b = 1	1		1
3	b <= a		да		
4	b := b + 1	b = b + 1	2		2
5	a := a - 1	a = a - 1	4	4	
6	b <= a		да		
7	b := b + 1	b = b + 1	3		3
8	a := a - 1	a = a - 1	3	3	
9	b <= a		да		
10	b := b + 1	b = b + 1	4		4
11	a := a - 1	a = a - 1	2	2	
12	b <= a		нет		
13	writeln(a)	print(a)		2	
14	writeln(b)	print(b)			4

Из таблицы видно, что в результате работы переменные приняли значения: a = 2 и b = 4.

Пример 4. Определим значение переменной s, полученное в результате выполнения следующей программы, записанной на двух языках программирования:



Программа на языке Pascal	Программа на языке Python
<pre> var s, k, d: integer; begin s := 0; d := 10; for k := 5 to 10 do s := s + d; writeln(s) end.</pre>	<pre> s = 0 d = 10 for k in range(5, 10 + 1): s += d print(s)</pre>

Построим трассировочную таблицу второго вида, отражая в каждой строке результат группы действий. Группу действий ограничим контрольной точкой: выполнение алгоритма продолжится до контрольной точки и приостанавливается после выполнения отмеченной ею строки.

Будем считать, что контрольная точка (КТ) поставлена на строке $s := s + d$.

Результат в КТ	k	s	d
Начальные значения	–	0	10
1	5	10	
2	6	20	
3	7	30	
4	8	40	
5	9	50	
6	10	60	

Итак, в результате работы программы переменная приняла значение $s = 60$.



Каким должно быть значение d , чтобы в результате работы программы переменная приняла значение $s = 186$? Существует ли такое значение d , что в результате работы программы переменная примет значение $s = 212$?



Пример 5. Определим значение переменной s , полученное в результате выполнения следующей программы, записанной на двух языках программирования:

Программа на языке Pascal	Программа на языке Python
<pre> var s, i, j: integer; begin s := 0; for i := 1 to 3 do for j := 3 downto i do s := s + i + j; writeln(s) end.</pre>	<pre> s = 0 for i in range(1, 4): for j in range(3, i - 1, -1): s += i + j print(s)</pre>

Трассировочная таблица может иметь вид:

Результат в КТ	i	j	s
Начальные значения	–	–	0
1	1	3	4
2		2	7
3		1	9
4	2	3	14
5		2	18
6	3	3	24
Результат:	24		

Пример 6. Выясним, для чего предназначена следующая программа, записанная на двух языках программирования:



Программа на языке Pascal

```
var
  n: integer; nd: string;
begin
  writeln('Введите натуральное число');
  read(n);
  nd := '';
  while n <> 0 do
    begin
      if n mod 2 = 1 then nd := '1' + nd
      else nd := '0' + nd;
      n := n div 2
    end;
  writeln(nd);
end.
```

Программа на языке Python

```
print('Введите натуральное число')
n = int(input())
nd = ''
while n != 0:
    if n % 2 == 1:
        nd = '1' + nd
    else:
        nd = '0' + nd
    n //= 2
print(nd)
```

Прежде всего обратим внимание на то, что в программе, кроме переменной n целого типа, используется строка nd , для которой символ $+$ обозначает операцию сцепления строк. Начальное значение n вводится с клавиатуры, поэтому зададим его по своему усмотрению, например $n = 12$.

Результат в КТ	nd	n
Начальные значения	''	12
1	'0'	6
2	'00'	3
3	'100'	1
4	'1100'	0
Результат:	1100	



Выполните программу для $n = 25$. Какую задачу, по вашему мнению, решает эта программа?

7.4. Другие приёмы анализа программ

Трассировочная таблица — наглядный, но не универсальный инструмент анализа программ. Например, её затруднительно строить, если в алгоритме много шагов.



Пример 7. Требуется выяснить, какое число будет выведено в результате выполнения следующей программы, записанной на двух языках программирования:

Программа на языке Pascal	Программа на языке Python
<pre> var n, s: integer; begin n := 0; s := 400; while s < 2992 do begin s := s + 12; n := n + 2 end; write(n) end.</pre>	<pre> n = 0 s = 400 while s < 2992: s += 12 n += 2 print(n)</pre>

Трассировочная таблица для этой программы будет содержать не одну сотню строк. Попробуем проанализировать программу иначе.

1. Выясним, какую функцию выполняет каждая из переменных, задействованных в программе.

Начальное значение переменной $s = 400$. При каждом выполнении тела цикла к значению s прибавляется число 12.

Начальное значение переменной $n = 0$. При каждом выполнении тела цикла значение переменной увеличивается на 2: $n = 2$, если тело цикла выполнено 1 раз; $n = 4$ — если 2 раза; $n = 6$ — если 3 раза и т. д. Таким образом, искомое значение n — это $2 \cdot k$, где k — число выполнений тела цикла.

2. Выясним, при каком условии произойдёт выход из цикла. Цикл выполняется, пока $s < 2992$. Следовательно, цикл завершится при достижении s значения, равного или большего 2992.

3. Выясним, сколько раз выполнится тело цикла, вычислив значение выражения: $(2992 - 400)/12 = 216$. После того как тело цикла выполнится 216 раз, значение переменной s будет равно 2992, что является условием выхода из цикла. При этом $n = 2 \cdot 216 = 432$.

Выясните, каким будет результат работы программы, если в ней условие выхода из цикла будет изменено на:

- 1) $s < 2990$; 2) $s \leq 2992$; 3) $s \leq 300$.





Пример 8. Получив на вход некоторое натуральное число x , программа, записанная на двух языках программирования, выводит два числа — m и n :

Программа на языке Pascal	Программа на языке Python
<pre> var x, m, n: integer; begin readln(x); m := 0; n := 1; while x > 0 do begin m := m + 1; n := n * (x mod 10); x := x div 10; end; writeln(m); write(n) end.</pre>	<pre> x = int(input()) m = 0; n = 1 while x > 0: m += 1 n *= (x % 10) x //= 10 print(m); print(n)</pre>

Известно, что при некотором значении x были выведены числа 5 и 25. Выясним, сколько существует разных значений x , при вводе которых может быть получен такой результат.

Выясним, какие именно данные накапливаются в переменных.

Начальное значение переменной x задаётся пользователем. Это целочисленная переменная. В цикле её значение изменяется по правилу, заданному командой

$$x := x \text{ div } 10 \qquad (x \text{ //= } 10)$$

При таком преобразовании значение переменной x уменьшается в 10 раз и дробная часть результата отбрасывается. Можно сказать, что при каждом выполнении тела цикла от значения переменной x «отсекается» одна цифра справа.

Начальное значение переменной $m = 0$. При каждом выполнении цикла значение переменной m увеличивается на единицу. Можно сказать, что в m подсчитывается количество цифр, «отсечённых» от x .

Начальное значение переменной $n = 1$. В цикле значение переменной n изменяется по правилу, заданному командой:

$$n := n * (x \text{ mod } 10) \qquad (n *= (x \% 10))$$

Здесь $x \text{ mod } 10$ ($x \% 10$) — не что иное, как последняя цифра числа x . Таким образом, в переменной n накапливается произведение цифр числа x , взятых справа налево.

Выход из цикла осуществляется при $x \leq 0$, т. е. когда все значащие цифры этого числа будут рассмотрены.

Следовательно, если на экран первой выводится цифра 5, то исходное число пятизначное. Второе число указывает на то, что 25 — это произведение всех цифр исходного числа x .

Рассмотрим варианты пятизначных чисел, произведение цифр которых равно 25. Например, 11551, 51151 и т. д. Очевидно, в записи любого из таких чисел должны быть две пятёрки и три единицы. Применение известной вам формулы из комбинаторики позволяет вычислить количество разных чисел, удовлетворяющих такому условию, — это 10.

О какой формуле идёт речь? Приведите эту формулу и выполните соответствующие вычисления.

Укажите наибольшее и наименьшее числа, удовлетворяющие условию задачи.

Выпишите все числа, удовлетворяющие условию задачи.

САМОЕ ГЛАВНОЕ

Компьютерную программу можно считать последовательностью строк символов некоторого алфавита. Современные системы программирования и языки допускают использование визуальных элементов (окон, иконок и др.) для построения программ, в частности для создания интерфейса пользователя. Тем не менее основная, алгоритмическая, часть любой программы строится с использованием символьных средств.

Компьютер оперирует только одним видом данных — отдельными битами (двоичными цифрами). Задачи, решаемые с помощью компьютера, оперируют данными, сгруппированными для представления чисел, символов, текстов и более сложных структур. Алгоритмы для обработки этих данных создаются с учётом их структуры — множества элементов данных и множества связей между ними.

Различают простые и сложные структуры данных. Простые структуры данных не могут быть разделены на составные части, большие, чем бит. К ним относятся числовые, символьные, логические и другие данные. Простые структуры данных служат основой для построения сложных структур данных — массивов, списков, графов, деревьев и др.

Для анализа свойств алгоритма и проверки его соответствия решаемой задаче используются трассировочные таблицы. В них фиксируется пошаговое исполнение алгоритма (программы), что позволяет наглядно представлять значения переменных, изменяющиеся при его выполнении. Используются трассировочные таблицы двух видов:



- 1) таблицы, каждая строка которых отражает результат одного действия;
- 2) таблицы, каждая строка которых отражает результат выполнения группы действий.



Вопросы и задания

1. Что такое язык программирования? Опишите состав и интерфейс среды разработки программ на используемом вами языке программирования.
2. Приведите примеры структур данных, используемых в языке программирования Pascal (Python).
3. Кратко охарактеризуйте основные элементы языка программирования Pascal (Python).
4. Для чего предназначены трассировочные таблицы?
5. Вещественные числа x , y , z являются исходными данными для следующего алгоритма:
 - 1) переменной m присвоить значение x ;
 - 2) сравнить значения m и y : если y больше m , переменной m присвоить значение y ;
 - 3) сравнить значения m и z : если z больше m , переменной m присвоить значение z .
 Выясните, какую задачу решает этот алгоритм. Запишите его на языке программирования Pascal (Python). Решите аналогичную задачу для чисел x , y , z и w .
6. Определите значение переменной n , которое будет получено в результате выполнения следующей программы, записанной на двух языках программирования:

Программа на языке Pascal	Программа на языке Python
<pre> var s, n: integer; begin s := 0; n := 1; while sqr(s + 2) < 125 do begin n := n * 2; s := s + 2; end; writeln(n) end.</pre>	<pre> s = 0; n = 1 while (s + 2) ** 2 < 125: n *= 2 s += 2 print(n)</pre>

7. Определите значение переменной s , которое будет получено в результате выполнения следующей программы, записанной на двух языках программирования:

Программа на языке Pascal	Программа на языке Python
<pre> var s, i, j: integer; begin s := 0; for i := 1 to 3 do for j := i to 4 do s := s + 2 * i - j; writeln(s) end. </pre>	<pre> s = 0 for i in range(1, 4): for j in range(i, 5): s += 2 * i - j print(s) </pre>

8. Требуется выяснить, какое число будет выведено в результате выполнения следующей программы, записанной на двух языках программирования:

Программа на языке Pascal	Программа на языке Python
<pre> var n, s: integer; begin n := 0; s := 1000; while s >= 100 do begin s := s - 2; n := n + 1 end; write(n) end. </pre>	<pre> n = 0 s = 1000 while s >= 100: s -= 2 n += 1 print(n) </pre>

9. Получив на вход число x , приведённая ниже программа, записанная на двух языках программирования, выводит два числа — m и n .

Программа на языке Pascal

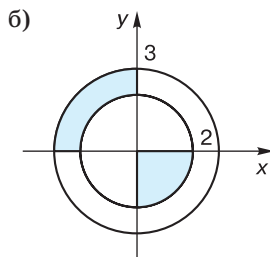
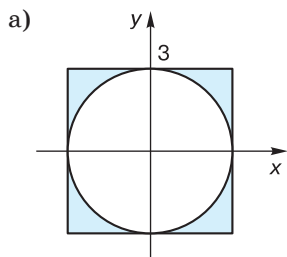
```
var x, m, n: integer;
begin
  readln(x);
  m := 0; n := 0;
  while x > 0 do
    begin
      if n < x mod 10 then n := x mod 10;
      m := m + 1;
      x := x div 10;
    end;
  writeln(m); write(n)
end.
```

Программа на языке Python

```
x = int(input())
m = 0; n = 0
while x > 0:
    if n < x % 10:
        n = x % 10
    m += 1
    x //= 10
print(m); print(n)
```

- Известно, что при некотором значении x были выведены числа 4 и 8. Укажите наибольшее и наименьшее из таких чисел x . Сколько всего существует таких x ?
10. Напишите программу, выводящую на экран все чётные трёхзначные числа.
11. Напишите программу, подсчитывающую сумму квадратов всех чисел от 1 до n .
12. Напишите программу, позволяющую определить, входит ли заданная цифра в некоторое целое неотрицательное число.
13. Разработайте программу перевода десятичного натурального числа n в троичную систему счисления.

14. Разработайте программу, которая выводит сообщение «Да», если точка с координатами (x, y) принадлежит закрашенной области, и «Нет» в противном случае.



15. Шифр кодового замка является двузначным числом. Буратино забыл код, но помнит, что сумма цифр этого числа, сложенная с их произведением, равна самому числу. Напишите программу, выводящую на экран все возможные варианты кода, чтобы Буратино смог быстрее открыть замок. Решите задачу методом перебора.

