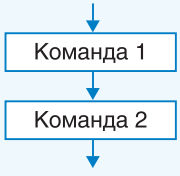
Глава 2
АЛГОРИТМЫ ОБРАБОТКИ СТРОКОВЫХ ВЕЛИЧИН

§ 6. Основные алгоритмические конструкции и типы данных

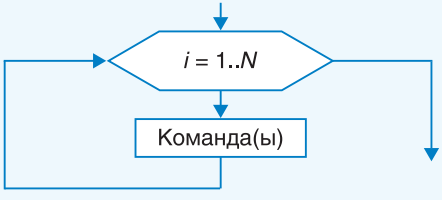
Пример 6.1. Блок-схемы алгоритмических конструкций.

1. Следование

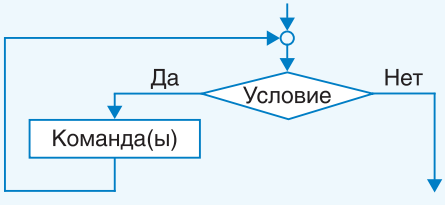


2. Цикл

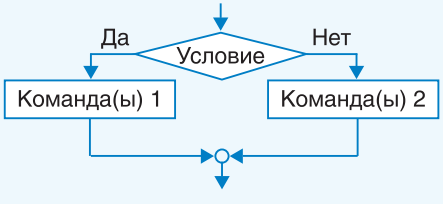
1) цикл с параметром



2) цикл с предусловием



3. Ветвление



6.1. Основные алгоритмические конструкции

Напомним некоторые определения, известные вам из курсов 7-го и 8-го классов.

Алгоритм — конечная последовательность команд, формальное выполнение которых позволяет получить решение задачи для любого допустимого набора исходных данных. Все команды делят на группы:

1. Команды, которые непосредственно выполняются в программе.
2. Команды, изменяющие порядок выполнения других команд.

Любой алгоритм может быть записан с использованием базовых алгоритмических конструкций, а именно: **следование, цикл** и **ветвление** (пример 6.1).

Программа представляет собой запись на некотором формальном языке — языке программирования. Командами в языке программирования считают:

- операторы (оператор присваивания, оператор ветвления, оператор цикла и др.);
- вызовы вспомогательных алгоритмов (встроенных в библиотеки и созданных пользователем).

Команды цикла и ветвления управляют порядком выполнения других команд в программе и относятся

к **командам управления (управляющим конструкциям)** (пример 6.2).

Оператор ветвления — команда, реализующая алгоритмическую конструкцию *ветвление* на языке программирования.

Оператор цикла — команда, реализующая алгоритмическую конструкцию *повторение* на языке программирования.

В Pascal существуют различные возможности управлять тем, сколько раз будет повторяться тело цикла. Может быть задано условие продолжения или окончания работы цикла, а также число повторений тела цикла.

Цикл с предусловием используется в том случае, когда известно условие продолжения работы.

Цикл с параметром используется тогда, когда известно количество повторений.

6.2. Вспомогательные алгоритмы

Вспомогательный алгоритм — алгоритм, который можно использовать в других алгоритмах, указав его имя и, если необходимо, значения параметров.

В языке Pascal используются вспомогательные алгоритмы двух видов: процедуры и функции. Они могут быть с параметрами или без параметров (пример 6.3).

Описание процедур и функций повторяет структуру программы на языке Pascal. Оно может содержать раздел **var** для описания переменных, которые используются только

Пример 6.2. Запись операторов ветвления и цикла на языке Pascal.

Ветвление в полной форме:

```
if <условие> then
begin
  команды 1;
end
else
begin
  команды 2;
end;
```

Ветвление в сокращенной форме

```
if <условие> then
begin
  команды;
end;
```

Цикл с предусловием

```
while <условие> do
begin
  тело цикла;
end;
```

Цикл с параметром

Параметр увеличивается:

```
for var i := N1 to N2 do
begin
  тело цикла;
end;
```

Параметр уменьшается:

```
for var i := N2 downto N1 do
begin
  тело цикла;
end;
```

Пример 6.3. Общий вид процедуры:

```
procedure <имя>(<список параметров>:тип);
var <описание переменных>
begin
  <команды>
end;
```

Общий вид функции:

```
function <имя>(<список параметров>:тип): тип результата;
var <описание переменных>
begin
  <команды>
  <имя> := <значение>;
end;
```

Функция должна содержать команду вида <имя> := <значение>;. Эта команда определяет, что функция должна вернуть в качестве результата.

Пример 6.4. Вызов процедуры и функции.

Вызов процедуры рисования круга:

```
Circle(250, 125, 30);
```

вызов функций для вычисления квадратного корня и синуса:

```
d := sqrt(2) * sin(x);
```

Пример 6.5. Целочисленные типы данных в PascalABC.

Тип	Диапазон значений	Размер памяти, байт
shortint	−128..127	1
smallint	−32768..32767	2
integer, longint	−2147483648.. 2147483647	4
byte	0..255	1
word	0..65535	2
longword, cardinal	0..4294967295	4

Дополнительно в PascalABC определен тип BigInteger, который не ограничен диапазоном значений и размером памяти.

Пример 6.6. Вещественные типы данных в PascalABC.

Тип	Диапазон значений	Размер памяти, байт
real (double)	$-1.8\cdot10^{308}..1.8\cdot10^{308}$	8
single	$-3.4\cdot10^{38}..3.4\cdot10^{38}$	4
decimal	$-(2^{96}-1)..2^{96}-1$	16

Количество значащих цифр в типе real составляет 15—16, в типе single — 7—8, в типе decimal — 28—29.

Тип real имеет другое название — double. Самое маленькое положительное число типа real приблизительно равно $5.0\cdot10^{-324}$, для типа single оно составляет приблизительно $1.4\cdot10^{-45}$.

внутри данной процедуры или данной функции.

Функции, в отличие от процедур, в результате своего выполнения возвращают значение, которое может быть использовано в выражении. Вызов процедуры является отдельной командой (пример 6.4).

6.3. Типы данных

В языке Pascal используются разные типы данных. Они нужны для выполнения различных операций — с каждым типом данных связан свой набор операций.

Для хранения различных типов данных в памяти компьютера отводится разное количество памяти. Вы уже использовали типы данных, которые называют простыми. В таблице примера 6.5 приведены целочисленные типы данных, используемые в PascalABC.

Все целочисленные типы данных, представленные в таблице, можно разделить на две группы:

- **знаковые** (диапазон значений которых содержит как положительные, так и отрицательные числа);
- **беззнаковые** (диапазон значений содержит только неотрицательные числа).

Для каждого знакового типа есть беззнаковый, занимающий столько же памяти.

Вещественные типы данных позволяют хранить число, представленное в стандартном виде: $a \cdot 10^n$, где $1 \leqslant a < 10$

и n (целое) есть порядок числа, записанного в стандартном виде (пример 6.6).

Значения типа boolean, который называют логическим, занимают 1 байт и принимают одно из двух значений, задаваемых константами true (истина) и false (ложь).

Данные в программу пользователь может вводить с помощью команды read (readln). Для вывода данных используется команда write (writeln).

Оператор присваивания используется для того, чтобы задавать значения переменным и вычислять значение выражения.

При использовании в одном операторе присваивания данных разных типов нужно помнить об их совместимости:

- переменной целочисленного типа нельзя присвоить вещественное значение;
- для данных вещественных типов определены операции «+», «−», «*», «/»;
- для данных целочисленных типов определены операции «+», «−», «*», «**div**», «**mod**».

Целочисленные типы могут быть преобразованы к вещественным, но не наоборот (пример 6.7).

Все простые типы, кроме вещественного, называются **порядковыми**. Значения только этих типов могут быть параметрами цикла **for**. Для порядковых типов используются функции ord, pred и succ, а также процедуры inc и dec (пример 6.8

Пример 6.7. Приведение типов.

```
var a, b, c: real;
    x, y, z: integer;
begin
  a := 1; b := 3; x := 6; y := 4;
  //вычисления с вещественным типом
  c := a / b; writeln(c);
  //преобразование к вещественному
  c := x / y; writeln(c);
  //вычисления с целым типом
  z := x div y; writeln(z);
  //преобразование к вещественному
  c := x div y; writeln(c);
end.
```

Результат:

Окно вывода

```
0.333333333333333
1.5
1
1
```

Пример 6.8. Процедуры и функции для работы с порядковыми типами.

Функция	Описание
Ord(a)	Порядковый номер значения a
Pred(x)	Значение, предшествующее x
Succ(x)	Значение, следующее за x

Процедура	Описание
Inc(i);	Увеличивает значение переменной i на 1
Inc(i, n)	Увеличивает значение переменной i на n
Dec(i);	Уменьшает значение переменной i на 1
Dec(i, n)	Уменьшает значение переменной i на n

Пример 6.9. Фрагмент программы для работы с порядковыми типами.

```
var a, c: integer;
    b: boolean;
begin
  a := 10; c := 3; b := true;
  writeln(ord(a));
  writeln(pred(b));
  writeln(succ(c));
  inc(b); writeln(b); dec(c, a);
  writeln(c);
end.
```

Окно вывода

```
10
False
4
False
-7
```

Пример 6.10.

V. Программа:

```
uses GraphABC;
const slovo = 'Pascal';
begin
  SetFontColor(clRed);
  writeln(slovo);
end.
```

VI. Тестирование.

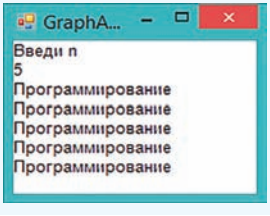
Запустить программу. Результат:

Пример 6.11.

V. Программа:

```
uses GraphABC;
const sl = 'Программирование';
var n: integer;
begin
  writeln ('Введи n ');
  read(n); writeln (n);
  for var i := 1 to n do
    writeln(sl);
end.
```

VI. Тестирование. Результат при $n = 5$:



и пример 6.9). Все переменные, которые используются в программе, должны быть описаны в разделе **var**. Если данные не изменяются в процессе работы программы, то они могут быть описаны как константы в разделе **const**. Например:

```
const slovo = 'Привет';
      Pi = 3.1416;
```

Для работы с графическими данными используются команды библиотеки GraphABC (см. *Приложение 2*, с. 159—160).

6.4. Примеры задач

Пример 6.10. Описать слово «Pascal» как константу. Вывести слово на экран красным цветом.

Этапы выполнения задания

I—II. Результат работы не зависит от исходных данных.

III. Алгоритм решения задачи.

1. Установить красный цвет (команда находится в библиотеке GraphABC).
2. Вывести константу.

IV. В программе нет переменных.

Пример 6.11. Написать программу, которая выведет заданное слово на экран n раз. Значение n вводится.

Этапы выполнения задания

I. Исходные данные: переменная n.

II. Результат: n слов.

III. Алгоритм решения задачи.

1. Описываем слово как константу.
2. Вводим значения n.
3. Воспользуемся циклом **for** для вывода слова n раз.
4. Выводим слова в цикле.

IV. Описание переменных: n — integer.

Пример 6.12. Написать программу, которая выведет на экран n раз одно из двух слов. Выбор слова осуществляется случайным образом. Значение n вводится. Посчитайте, сколько раз было выведено каждое слово.

Этапы выполнения задания

I. Исходные данные: переменная n.

II. Результат: n раз выведено одно из двух слов и сообщение о том, сколько раз выведено каждое из слов.

III. Алгоритм решения задачи.

1. Слова описываем как константы.

2. Вводим значения n.

3. Инициализируем нулем переменные k1 и k2, которые будут подсчитывать, сколько раз выведено каждое слово.

4. Для вывода слов используем цикл **for**.

4.1. Сгенерируем случайное число x на промежутке $[0; 2)$.

4.2. Если x = 0, то выведем первое слово и увеличим значение переменной k1 на 1.

4.3. Иначе выведем второе слово и увеличим значение переменной k2 на 1.

5. Выводим сообщения о количестве слов.

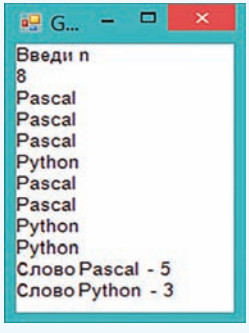
IV. Описание переменных: n, k1, k2, x — integer.

Пример 6.12.

V. Программа:

```
uses GraphABC;
const sl1 = 'Pascal';
      sl2 = 'Python';
var n, k1, k2, x: integer;
begin
  writeln ('Введи n ');
  read(n);
  writeln (n);
  k1 := 0; k2 := 0;
  for var i := 1 to n do
  begin
    x := random(2);
    if x = 0 then
    begin
      writeln(sl1);
      k1 := k1 + 1;
    end
    else
    begin
      writeln(sl2);
      k2 := k2 + 1;
    end
  end;
  writeln('Слово ',sl1,' - ',k1);
  writeln('Слово ',sl2,' - ',k2);
end.
```

VI. Тестирование. Запустить программу. Результат при $n = 8$:



1. Что такое алгоритм?
2. Назовите основные алгоритмические конструкции.
3. Что понимают под вспомогательным алгоритмом?
4. Чем отличаются различные целочисленные типы данных друг от друга?
5. Что нужно помнить о совместимости типов данных?
6. Какие арифметические операции определены для целочисленных типов данных? Для вещественных типов данных?

Упражнения

1 Измените константу в программе примера 6.10 на свое имя. Используя команды графической библиотеки для работы с текстом, измените шрифт, размер символов, фон, начертание букв.

2 Измените программу примера 6.11 так, чтобы выполнялись указанные условия.

1. Каждое слово должно выводиться случайным цветом.
2*. Расстояние между словами должно быть 50 пикселей.
3*. Каждое новое слово должно выводиться шрифтом на 5 пунктов больше, чем предыдущее. Отрегулируйте расстояние между словами так, чтобы слова при выводе не перекрывали друг друга.

3 Запустите программу из примера 6.12 несколько раз. Какие результаты получили?

4 Измените программу из примера 6.12 так, чтобы случайным образом выбиралось одно из трех слов. Выводите каждое слово своим цветом (например, первое — красным, второе — синим, третье — зеленым).

§ 7. Строковые величины

В первых языках программирования строкового типа данных не было; программист должен был сам строить функции для работы со строками.

В 1962 г. был разработан язык SNOBOL (StriNg Oriented symBOlic Language), ориентированный на работу со строками. В конце 60-х гг. XX в. строковые типы данных появились в языках Algol и Fortran.

Две строки, в отличие от двух чисел, нельзя прочитать с помощью одной команды `read`, поскольку пробел для строк не разделитель, а такой же символ, как и все остальные. Необходимо использовать две команды `readln`.

Если использовать две команды `read`, то первая строка будет считана так, как нужно, а вторая строка будет пустой (она не будет вводиться). Это происходит потому, что первая команда `read` считывает данные до нажатия клавиши Enter. Вторая команда `read` прочитает один символ — символ нажатия клавиши Enter.

7.1. Ввод, вывод, присваивание строковых величин

Современные компьютеры способны обрабатывать данные, представленные различными способами: числа, тексты, графику, звуки. Вы уже знаете, как на языке программирования Pascal можно работать с целыми и вещественными числами, выполнять простейшие графические построения. Обработка текстовых данных является сегодня наиболее актуальной — это обработка различных поисковых запросов в Интернете, перевод текстов с одного языка на другой, озвучивание компьютером печатного текста и др.

В языке Pascal для работы с текстовыми данными используется тип **`string` (строка).** Строки состоят из набора последовательно расположенных символов и используются для хранения текста. Они могут иметь произ-