

Основы языка Pascal

Алгоритмы

Алгоритмом называется система формальных правил, четко и однозначно определяющая процесс решения поставленной задачи в виде конечной последовательности действий или операций.

или

Алгоритм - некоторая конечная последовательность правил, определяющая процесс преобразования исходных и промежуточных данные в результате решения задачи.

Требования, предъявляемые к алгоритму:

- **однозначность** — предлагаемые действия должны быть «понятны» компьютеру, а порядок исполнения этих действий должен быть единственно возможным, любая неопределенность или двусмысленность недопустима;
- **массовость** — пригодность алгоритма для решения не только данной задачи, а множества родственных задач, относящихся к общему классу;
- **детерминированность** — повтор результата при повторе исходных данных;
- **корректность** — способность алгоритма давать правильные результаты решения задачи при различных исходных данных;
- **конечность** — решение задачи должно быть получено за конечное число шагов алгоритма, «заикливание» недопустима;
- **эффективность** — для успешного решения задачи должны использоваться ограниченные ресурсы конкретного компьютера (время работы процессора, объем оперативной памяти, быстродействие жесткого диска и др.).

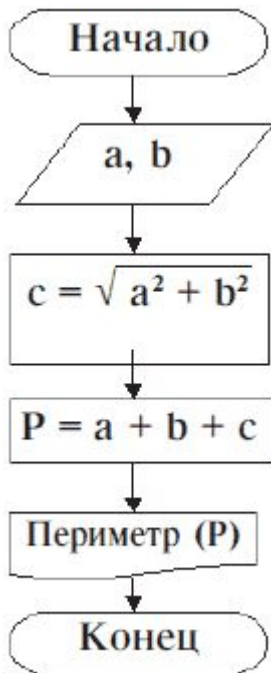
Типы алгоритмов

Алгоритмы подразделяются на **три типа**:

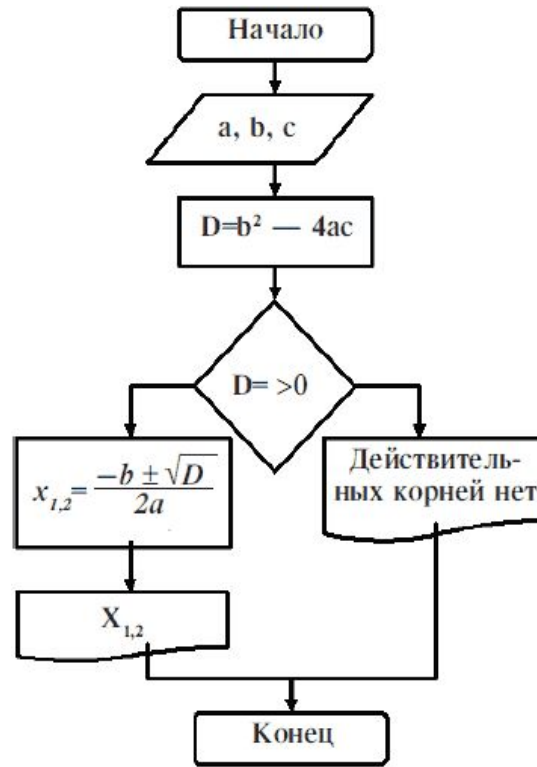
- **Линейный** – алгоритм, в котором все действия выполняются последовательно друг за другом;
- **Разветвляющийся** – алгоритм, в котором в зависимости от выполнения некоторого логического условия вычислительный процесс должен идти по одной или другой ветви;
- **Циклический** – алгоритм, предусматривающий многократное повторение одного и того же действия над новыми исходными данными.

Примеры

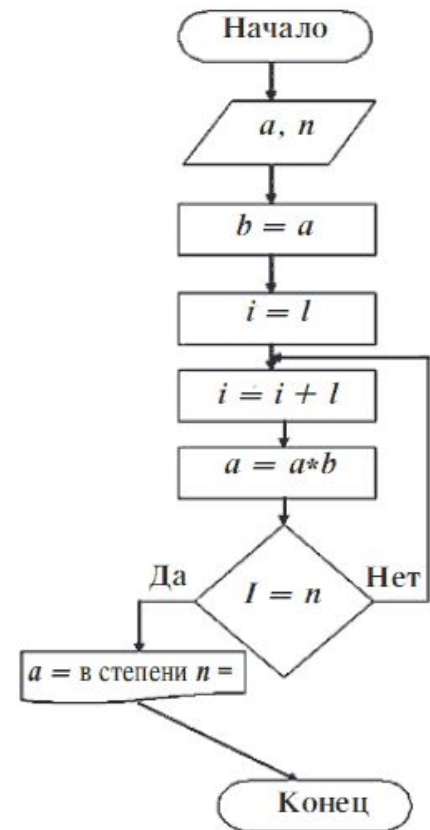
Линейный



Разветвляющийся



Циклический



Средства изображения алгоритмов

- словесный;

содержание этапов вычислений задается на естественном языке в произвольной форме с требуемой детализацией.

- блок-схемный;

это графическое изображение алгоритма, в котором каждый этап процесса обработки данных представляется в виде геометрических фигур (блоков) имеющих определенную конфигурацию в зависимости от характера выполняемых операций.

- языки программирования

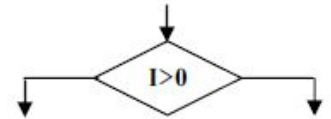
это формальные языки общения человека с ЭВМ, предназначенные для описания совокупности инструкций, выполнение которых обеспечивает правильное решение требуемой задачи.

Запись блоков

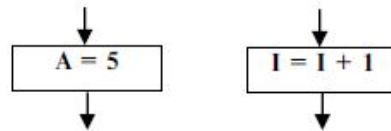
- 1 Начало и конец алгоритма (программы):



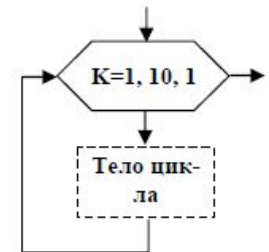
- 5 Блок ветвления (проверки условия):



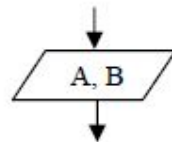
- 2 Блок присваивания:



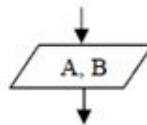
- 6 Блок цикла с параметром:



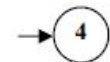
- 3 Ввод данных с клавиатуры:



- 4 Вывод информации на экран (на печать):



- 7 Нумерация блоков, значок перехода на блок с указанным номером:



АЛФАВИТ ЯЗЫКА

Под алфавитом языка понимают совокупность допустимых символов.

Основные группы символов:

- символы, используемые в идентификаторах
- разделители
- специальные символы
- неиспользуемые символы

Идентификатор

Идентификатор - это имя любого объекта языка.

Он может состоять из **латинских букв (a...z)**, **цифр (0...9)** и **знака подчеркивания** и не должен начинаться с цифры.

Прописные и строчные буквы в идентификаторах и зарезервированных словах считаются **идентичными**. Длина идентификатора не ограничена, но значимыми являются лишь первые **63 символа**.

Разделители

Разделители используются для отделения друг от друга идентификаторов, чисел и зарезервированных слов.

К разделителям относятся, например, **пробел и комментарий**.

В любом месте программы, где разрешается один пробел, их можно вставить любое количество.

Комментарии заключаются либо в фигурные скобки { комментарий 1 }, либо в символы (* комментарий 2 *) и могут занимать любое количество строк.

Специальные знаки

К специальным знакам относятся **знаки пунктуации** (. () [] .. : ;), **знаки операций** и **зарезервированные слова**.

Знаки операций могут быть как символьные (+, -, *, / и т. д.), так и буквенными (**mod**, **div**, **not**).

Зарезервированные слова являются служебными и не могут быть переопределены пользователем, т.е. их нельзя использовать как имена пользовательских объектов.

Неиспользуемые символы - используются только в комментариях и символьных строках, но не в языке. К ним относятся все **русские буквы**, а также символы %, &, ! и т.п.

Структура программы

Program ... ; { Заголовок программы }

Uses ... ; { Подключение модулей }

Label ... ; { Раздел объявления меток }

Type ... ; { Раздел объявления новых типов }

Const ... ; { Раздел объявления констант }

Var ... ; { Раздел объявления переменных }

Procedure ... ; { Описание своих процедур }

Function ... ; { Описание своих функций }

Begin { начало основной программы }

...;

{ Операторы }

...;

End.

ПРИМЕР: *Простейшая программа.*

```
program prim_1;
```

```
begin
```

```
  write('Привет!')
```

```
  экрaне }
```

```
end.
```

{эта строка текста появится на

Подраздел описания модулей

Подключение модуля:

USES Модуль;

где USES – зарезервированное слово;

Модуль – имя подключаемого модуля

Подраздел описания меток

Метка – точка перехода.

Данный подраздел начинается со слова **LABEL**, за которым следует список меток:

LABEL 1,77,190;

В качестве метки могут использоваться целые число без знака. Метка в теле программы ставиться перед оператором и отделяется от него двоеточием.

Подраздел описания типов



```
graph TD; A[Подраздел описания типов] --> B[Стандартные]; A --> C[Пользовательские];
```

Стандартные

(предопределенные разработчиками языка)

Пользовательские

(определяемы программистом в программе)

Стандартные (основные)
типы:

- Целые типы
- Вещественные типы
- Логический тип
- Символьный тип
- Строковый тип

TYPE

Имя типа=(идентификатор1,
идентификатор 2, ...
идентификатор N);

Целые типы

Shortint (-128 ... 127, 1 байт),

Integer (-32767 ... 32768, 2 байта),

Longint (-2147483648 ... 2147483647, 4 байта),

Byte (0 ... 255, 1 байт),

Word (0 ... 65535, 2 байта).

Вещественные типы

Real (занимает 6 байт, диапазон от $2.9E-39$ до $1.7E+38$ по модулю, точность 11-12 значащих цифр)

Single (занимает 4 байта, диапазон от $1.5E-45$ до $3.4E+38$ по модулю, точность 7-8 значащих цифр)

Double (занимает 8 байт, диапазон от $5.0E-324$ до $1.7E+308$ по модулю, точность 15-16 значащих цифр)

Extended (занимает 10 байт, диапазон от $3.4E-4932$ до $1.1E+4932$ по модулю, точность 19-20 значащих цифр).

Comp (занимает 8 байт, диапазон от $-9.2E-18$ до $9.2E+18$, хранятся точно, поскольку это целые числа)

Арифметические функции

Функция	Назначение	Функция	Назначение
<code>abs (x)</code>	абсолютное значение аргумента	<code>exp (x)</code>	e^x
<code>sqr (x)</code>	квадрат аргумента	<code>ln (x)</code>	натуральный логарифм
<code>sqrt (x)</code>	квадратный корень аргумента	<code>int (x)</code>	целая часть числа
<code>cos (x)</code>	косинус аргумента	<code>frac (x)</code>	дробная часть числа
<code>sin (x)</code>	синус аргумента	<code>round (x)</code>	округляет вещественное число до ближайшего целого.
<code>arctan (x)</code>	арктангенс аргумента	<code>trunc (x)</code>	выдает целую часть вещественного числа, отбрасывая дробную.

Логический тип (Boolean)

Переменные логического типа **Boolean** занимают в памяти один байт и могут принимать одно из двух значений **False** - ложное или **True** - истинное.

AND	false	true
false	false	false
true	false	true

XOR	false	true
false	false	true
true	true	false

OR	false	true
false	false	true
true	true	true

- отрицание (превращает
false в true, а true в false)

Символьный тип (Char)

Символьный тип Char позволяет работать с символами, которые записываются двумя способами:

в одинарных кавычках или по их коду, например 'a', 'B', '*' или, что то же самое, #97, #130, #42.

Функция **Ord** выдает код соответствующего символа, который может быть от 0 до 255. Обратной функцией, которая по коду выдает соответствующий символ, является функция **Chr**.

Подраздел описания констант

Константы – такие объекты программы , которые не могут изменять своего значения.

Описание:

CONST Имя=Значение;

Пример:

CONST Pi = 3.1415;

где

CONST – зарезервированное слово;

Имя – имя константы;

Значение – значение константы.

Подраздел описания переменных

Переменная – объект программы , который может изменять своего значение в процессе выполнения.

Пример:

Описание:

VAR Имя:тип;

VAR a:real;
b:integer;

где

VAR – зарезервированное слово;

Имя – идентификатор переменной;

тип – тип переменной.

Оператор присваивания

Оператор присваивания используется для задания значения переменных и имеет следующий синтаксис:

имя_переменной := выражение;

Вычисляется выражение, стоящее в правой части оператора, после чего его значение записывается в переменную, имя которой стоит слева.

Операторы ввода

Read(<список переменных через запятую>);

Readln(<список переменных>);

Readln;

Операторы вывода

Write(<СПИСОК ВЫВОДА>);

Writeln(<СПИСОК ВЫВОДА>);

Writeln;

ПРИМЕР: Простые вычисления.

```
program vvod_vyvod;  
const  n=1.5;  
var    y1,y2:real;  
        x:byte;  
  
begin  
writeln('Введите натуральное число <= 255');  
readln(x);  
y1:=cos(n); y2:=cos(x);  
writeln('n=',n,' y1=',y1:7:4, cos(Pi/2):8:4);  
writeln('x=',x:3,' y2=',y2:7:4);  
end.
```

Условный оператор

IF Условие THEN Оператор1 ELSE Оператор2 ;

где Условие - выражение или переменная логического типа (boolean);
Оператор1 и Оператор2 - простой или составной оператор.

При выполнении Условия (значение true) выполняется Оператор1, а Оператор2 игнорируется.

При невыполнении Условия (значение false) выполняется Оператор2, а Оператор1 игнорируется. Внимание: перед ELSE «;» не ставится!

усеченный вид оператора IF

IF Условие THEN Оператор1 ;

При этом Оператор2 вместе с ключевым словом ELSE отсутствует:

Оператор выбора (CASE)

```
CASE <ключ_выбора> OF  
C1 : <оператор1>;  
C2 : <оператор2>;  
...  
CN : <операторN>;  
[ELSE <оператор0>;]  
END;
```

Здесь <ключ_выбора> - это выражение порядкового типа, в зависимости от значения которого принимается решение;

C1,...,CN - значения, с которыми сравнивается значение <ключа>;
<оператор1>,..., <операторN> - оператор (возможно составные), из которых выполняется тот, с константой которого происходит первое совпадение значения <ключа>,

<оператор0> выполнится, если значение ключа не совпадает ни с одной из констант C1,...,CN.

ПРИМЕР 1. Вводится целое число, если это цифра, то определить четная она или нет, а если число, то определить попадает ли оно в диапазон от 10 до 100, если нет, то выдать соответствующее сообщение.

```
program chislo;  
var    i:integer;  
begin  
  write('Введите целое число: ');  
  readln(i);  
  case i of  
    0,2,4,6,8 : writeln('Четная цифра');  
    1,3,5,7,9 : writeln('Нечетная цифра');  
    10...100,200 : writeln('Число от 10 до 100 или 200');  
  else writeln('Число либо отрицательное, либо > 100, но не 200');  
  end;  
  readln  
end.
```

Цикл

Циклом называется вычислительный процесс, в котором один или несколько операторов повторяются некоторое количество раз.

Каждое повторение называется **итерацией**, поэтому циклические процессы еще называются **итерационными**.

Повторяющиеся операторы в цикле называются **телом цикла**.

Операторы цикла:

- цикл с параметром;
- цикл с предусловием;
- цикл с постусловием.

Цикл с параметром (FOR ...TO ... DO)

FOR i := N3 TO K3 DO Оператор ;

FOR i := N3 DOWNTO K3 DO Оператор ;

где **i** – переменная-счетчик ;

N3 и **K3** – соответственно начальное и конечное значения переменной **i**;

Оператор – любой простой или составной оператор, являющийся телом цикла.

Пример 2. Найти сумму целых,
положительных чисел, больших 20, меньших
100 и кратных 3.

```
program Summa;  
var I,Sum:integer;  
begin  
Sum:=0;  
For i:=20 to 100 do  
    If (i mod 3) = 0 then Sum:=Sum+i  
writeln('Сумма чисел = ', Sum);  
readln  
end.
```


Цикл с предусловием (WHILE ... DO ...)

WHILE Условие **DO** Оператор ;

где **Условие** - выражение или переменная логического типа (boolean); **Оператор** - простой или составной оператор.

Цикл с постусловием (REPEAT ... UNTIL)

REPEAT

Оператор1;

Оператор2;

.....

ОператорN;

UNTIL Условие ;

где **Оператор1, Оператор2, ОператорN** - операторы тела цикла;
Условие - выражение или переменная логического типа (boolean).

Пример 2. Вычислить сумму чисел Фибоначчи, значение которых не превосходит 500.

```
program Fib;  
var a,b,c:word;      i,n:byte;  
begin  
a:=1; {a=F(0), a соответствует F(i-2)}  
b:=1; {b=F(1), b соответствует F(i-1)}  
c:=a+b;  
Sum:=a+b+c;  
While c<=500 do  
begin  
a:=b; b:=c; {в качестве a и b берется следующая  
           пара чисел}  
c:=a+b;  
Sum:=Sum+c;  
end;  
writeln('Сумма чисел Фиббоначчи =',sum);  
readln  
end.
```

```
program Fib;  
var a,b,c:word;      i,n:byte;  
begin  
Repeat  
a:=1; {a=F(0), a соответствует F(i-2)}  
b:=1; {b=F(1), b соответствует F(i-1)}  
Sum:=a+b;  
Repeat  
c:=a+b;  
Sum:=Sum+c;  
a:=b; b:=c;  
Until c>500;  
writeln('Сумма чисел Фиббоначчи =',sum);  
readln  
end.
```