

# Язык программирования PASCAL





# Оглавление

История создания языка программирования **История создания языка программирования Turbo Pascal**

Структура программы

Оператор ввода, вывода и присваивания

Графическое представление программы

Выполнение оператора ввода данных с клавиатуры

Выполнение оператора вывода данных на монитор

Типы данных

Объекты **Объекты Pascal**

Условный оператор **Условный оператор IF**

Циклы



# История создания

В 1970 году профессор Никлаус Вирт создал в Цюрихском политехническом университете язык программирования Паскаль (Pascal). Создатель языка назвал его в честь Блеза Паскаля – первого конструктора устройства.

Он создавался как язык, который, с одной стороны, был бы хорошо приспособлен для обучения программированию, а с другой – давал бы возможность эффективно решать самые разнообразные задачи на современных ЭВМ.



# Структура Программы

**Program** <name>;

← Заголовок программы

    константы;  
    переменные;

← Раздел описания объектов

**begin**

<оператор 1 >;  
    <оператор 2>;

← Раздел операторов

**end.**

**Пример**





# ЗАДАЧА

Найдите сумму двух чисел.

```
program SUMMA; } ← Заголовок программы  
  
  var a,b: real; } ← Раздел описания объектов  
  
begin  
  readln (a,b);  
  S:= a+b;  
  writeln (S);  
  readln;  
end. } ← Раздел операторов
```

назад





# Оператор ввода, вывода, присваивания.

1. Оператор ввода данных с клавиатуры.

**Синтаксис:** `readln (x,y,z);` (x,y,z – переменные)

2. Оператор вывода данных на экран

**Синтаксис:** `writeln ('текст ', a,b, 2 * X+4);`

3. Оператор присваивания.

**Синтаксис:** `переменная := выражение;`

**Нельзя присвоить выражению переменную!**

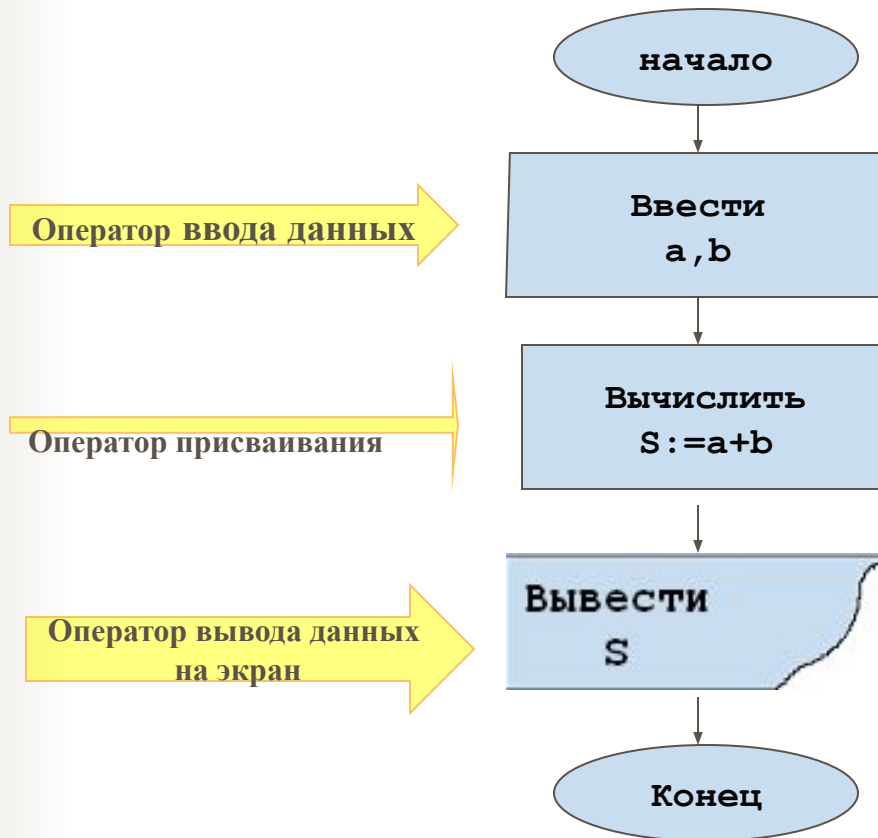
`D:= sqr(b)- 4ac`

~~`b-4ac := D`~~



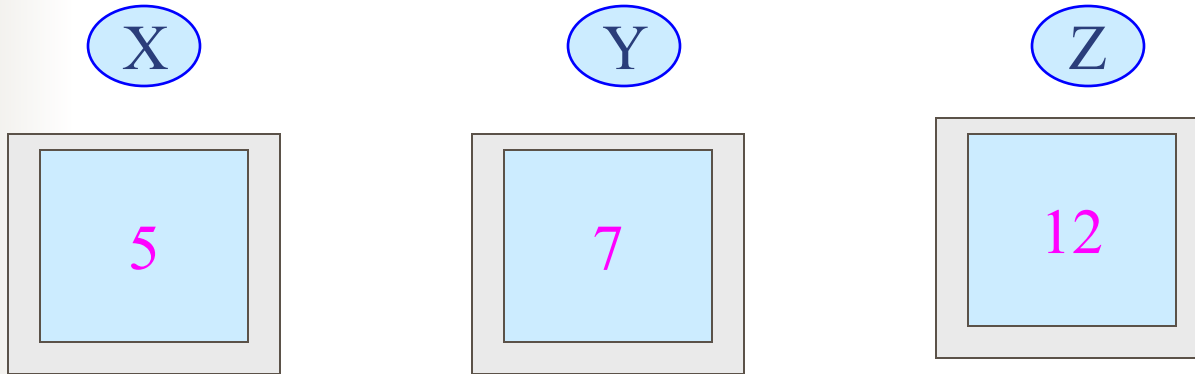


# Графическое представление программ





# Выполнение оператора ввода данных с клавиатуры



1. В памяти компьютера выделяются три ячейки.
2. Им присваиваются имена, заданные пользователем.
3. Пользователь вводит с клавиатуры значения переменных через пробел.
4. После нажатия клавиши Enter значения переменных заносятся в память компьютера.







# Выполнение вывода данных на монитор

```
D:= sqr(b) - 4*a*c;
```

```
writeln ('Дискриминант равен D=', D);
```



1. Компьютер вычисляет значение выражения и помещает его в определённую ячейку (D).
2. Выводит текст-приглашение и значение выражения на монитор.





# Типы данных

Типы данных	Диапазон значений	Объём памяти
<b>Целый тип</b>		
<code>shortint</code>	-128...127	1 Байт со знаком
<code>byte</code>	0...255	1 Байт
<code>word</code>	0...65535	2 Байта
<code>integer</code>	-32768...32767	2 Байта со знаком
<code>longint</code>	-2147483648... 2147483647	4 Байта со знаком
<b>Вещественный тип</b>		
<code>real</code>		4 или 8 в зависимости от точности



# Константы и Переменные

**Константа (const)** — данная, значение которой не меняется в процессе выполнения программы.

**Переменная** — данная, значение которой меняется в процессе выполнения программы.

Константа и переменная характеризуются **ИМЕНЕМ**, **ТИПОМ** и **ЗНАЧЕНИЕМ**.





# Математические функции

Функция	Тип аргумента	Тип результата
<code>abs(x)</code> - модуль $x$	целый или вещественный	Целый, вещественный
<code>sqr(x)</code> - квадрат	-//-	-//-
<code>sqrt(x)</code> - корень	Целые и дробные	Вещественный
<code>sin(x)</code> - синус	Целый и вещественный	Вещественный
<code>cos(x)</code> - косинус	Целый и вещественный	Вещественный
Pi - Пи	Нет	Вещественный
<code>round(x)</code> - округлить до целого	Вещественный	Вещественный
<code>trunc(x)</code> -целая часть от $(x)$	Вещественный	Вещественный
<code>int(x)</code> -целая часть от $(x)$	Вещественный	Вещественный
<code>frac(x)</code> - дробная часть от $(x)$	вещественный	Вещественный



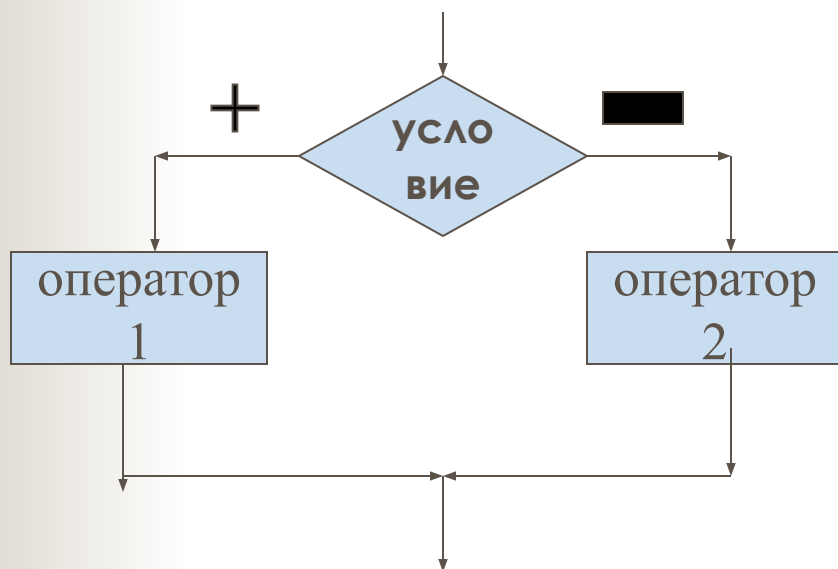


# Условный оператор IF

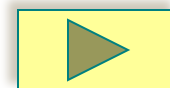


Полная форма условного оператора.

**Синтаксис:** `if <условие> then <оператор 1>  
else <оператор 2> ;`



1. Проверка условия.
2. Если условие истинно, то выполняется оператор после then. Если ложно, то выполняется оператор после else.
3. Выполняется оператор следующий за If.

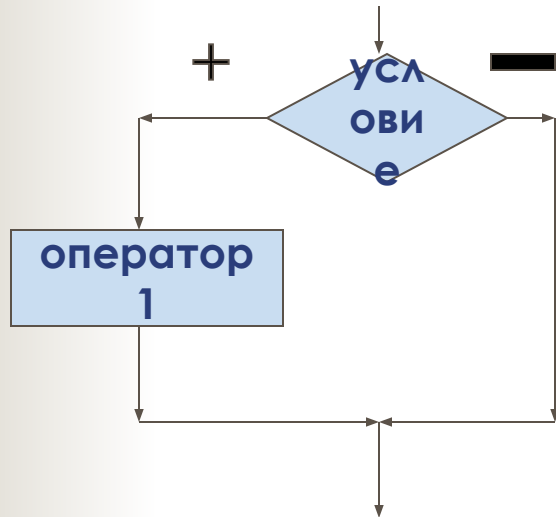




# Условный оператор IF



Сокращённая форма условного оператора.



**Синтаксис:**

**if < условие> then < оператор1>;**

Если надо выполнить последовательность действий (несколько операторов), то их надо заключать в операторные скобки. Операторными скобками называется пара зарезервированных слов «begin ... end».



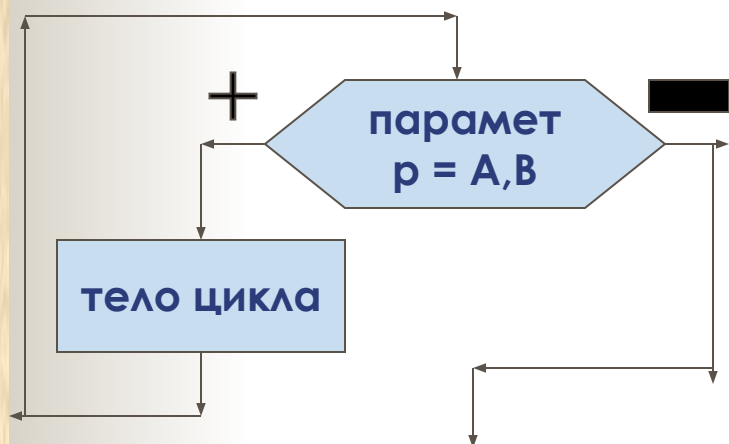


# ЦИКЛЫ

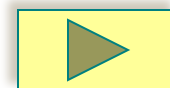
## 1. Оператор цикла с параметром.

Оператор цикла с параметром применяют тогда, когда заранее известно число повторений одной и той же последовательности действий.

**Синтаксис:** for <параметр> := A to B do <тело цикла>;



1. Вычисляются значения выражения A и B.
2. Если  $A \leq B$ , то параметр последовательно принимает значения равные A, A+1 ... B-1, B и для каждого из этих значений выполняется тело цикла.
3. Если  $A > B$ , то тело цикла не выполнится ни разу.



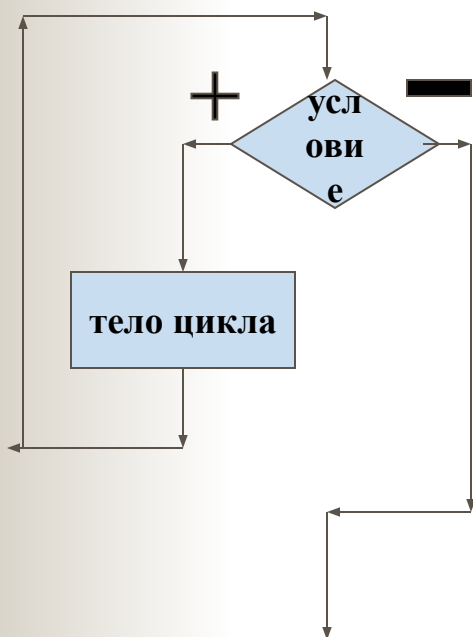


# ЦИКЛЫ

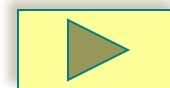
## 2. Оператор цикла с предусловием

Оператор цикла с предусловием используется тогда, когда число повторений оператора цикла заранее не известно, а задаётся некоторое условие продолжения цикла.

**Синтаксис:** `while <условие> do <тело цикла>;`



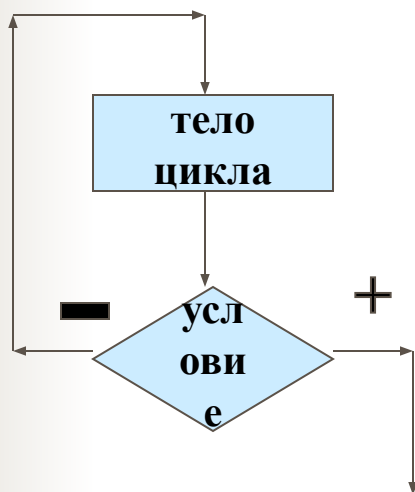
1. Проверка условия, записанного после слова `while`.
2. Если оно выполняется, то выполняется тело цикла, затем вновь проверка условия и т.д.
3. Как только при проверке окажется, что условие не соблюдается, то тело цикла выполняться не будет и программа перейдёт к выполнению следующего оператора.





# ЦИКЛЫ

## 3. Оператор цикла с постусловием.



**Синтаксис:** repeat <тело цикла> until <условие>;

1. Выполняется тело цикла.
2. Проверяется условие : если оно истинно, то программа выходит из цикла; а при невыполнении условия тело цикла повторяется.

