Среда программирования Pascal ABC

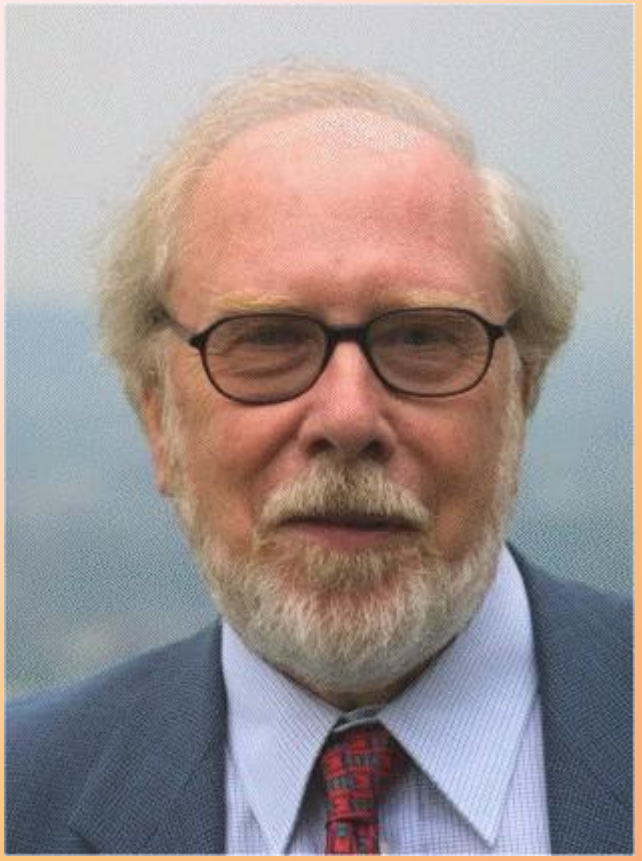
Для учащихся 9-11 классов

Язык назван в честь выдающегося французского математика, физика, литератора и философа Блеза Паскаля, который создал первую в мире механическую машину, складывающую два числа.

Содержание

- Система PascalABC
- Структура программы
- Идентификаторы
- Описание переменных
- Описание констант
- Описание меток
- Описание типов
- Типы данных

Система PascalABC



Язык Паскаль был разработан Никлаусом Виртом в 1970 г. как язык со строгой типизацией и интуитивно понятным синтаксисом. В 80-е годы наиболее известной реализацией стал компилятор Turbo Pascal фирмы Borland, в 90-е ему на смену пришла среда программирования Delphi, которая стала одной из лучших сред для быстрого создания приложений под Windows. Delphi ввела в язык Паскаль ряд удачных объектно-ориентированных расширений, обновленный язык получил название Object Pascal. Из альтернативных реализаций Object Pascal следует отметить многоплатформенный open source компилятор Free Pascal.

Преимущества PascalABC.NET

Современный язык программирования Object Pascal

Язык **PascalABC.NET** позволяет использовать большинство средств, предоставляемых платформой .NET: единая система типов, классы, интерфейсы, исключения, делегаты, перегрузка операций, обобщенные типы (generics), методы расширения, обширные.NET-библиотеки. Добавлен ряд языковых конструкций: описание метода в теле класса, множества произвольных типов, операторы **foreach** и **lock**, внутриблоковые переменные. С другой стороны, **PascalABC.NET** имеет структуру языка Delphi (Object Pascal): внешние процедуры и функции, модули.

Алфавит языка

Алфавит - это совокупность допустимых в языке символов. Алфавит включает следующий набор основных символов:

- строчные и прописные латинские буквы:

  A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

  a b c d e f g h i j k l m n o p q r s t u v w x y z
- пробел
- подчеркивание: _
- арабские цифры: 0 1 2 3 4 5 6 7 8 9
- знаки операций: + - * / = <> < > <= >= := @
- ограничители: . , ' ( ) [ ] (. .) { } (* *) .. : ;
- спецификаторы: ^ # $

Алфавит языка

- служебные (зарезервированные) слова:
  ABSOLUTE EXPORTS LIBRARY SET ASSEMBLER EXTERNAL MOD SHL AND FAR NAME SHR ARRAY FILE NIL STRING ASM FOR NEAR THEN ASSEMBLER FORWARD NOT TO BEGIN FUNCTION OBJECT TYPECASE GOTO OF UNITCONST IF OR UNTILCONSTRUCTOR IMPLEMENTATION PACKED USESDESTRUCTOR IN PRIVATE VARDIV INDEX PROCEDURE VIRTUALDO INHERITED PROGRAM WHILEDOWNTO INLINE PUBLIC WITHELSE INTERFACE RECORD XOREND INTERRUPT REPEAT EXPORT LABEL RESIDENT

Структура программы

Программа на языке **PascalABC.NET** имеет следующий вид:

program *имя программы*;
раздел **uses**
раздел описаний
begin
операторы
end.

Первая строка называется ***заголовком программы*** и не является обязательной.
Раздел **uses** начинается с ключевого слова **uses**, за которым следует список имен модулей и пространств имен .NET, перечисляемых через запятую.
Раздел описаний может включать
разделы описания
переменных, констант, меток, типов, процедур и функций, которые следуют друг за другом в произвольном порядке.
Далее следует блок **begin/end**, внутри которого находятся операторы, отделяемые один от другого символом "точка с запятой".
Раздел **uses** и раздел описаний могут отсутствовать.

Идентификаторы

Идентификаторы служат в качестве имен программ, модулей, процедур, функций, типов, переменных и констант. Идентификатором считается любая последовательность латинских букв или цифр, начинающаяся с буквы. Буквой считается также символ подчеркивания "_".

Например, a1, _h, b123 - идентификаторы, а 1a, ф2 - нет.

Описание переменных

- Переменные могут быть описаны в разделе описаний, а также непосредственно внутри любого блока **begin/end**.
- Раздел описания переменных начинается со служебного слова **var**, после которого следуют элементы описания вида
- *список имен*: *тип*;
- или
- *имя*: *тип* := *выражение*;
- или
- *имя* := *выражение*;
- Имена в списке перечисляются через запятую.

Пример описания переменных

- Например:
- **var**

```
a,b,c: integer;
d: real := 3.7;
s := 'Pascal forever';
al := new ArrayList;
p1 := 1;
```

Описание констант

- Раздел описания именованных констант начинается со служебного слова **const**, после которого следуют элементы описания вида
- *имя константы = значение*;
- или
- *имя константы : тип = значение*;

Пример описания констант

```
const
    Pi = 3.14;
    Count = 10;
    Name = 'Mike';
    DigitsSet = ['0'..'9'];
    Arr: array [1..5] of integer = (1,3,5,7,9);
    Rec: record name: string; age: integer end =
  (name: 'Иванов'; age: 23);
    Arr2: array [1..2,1..2] of real = ((1,2),(3,4));
```

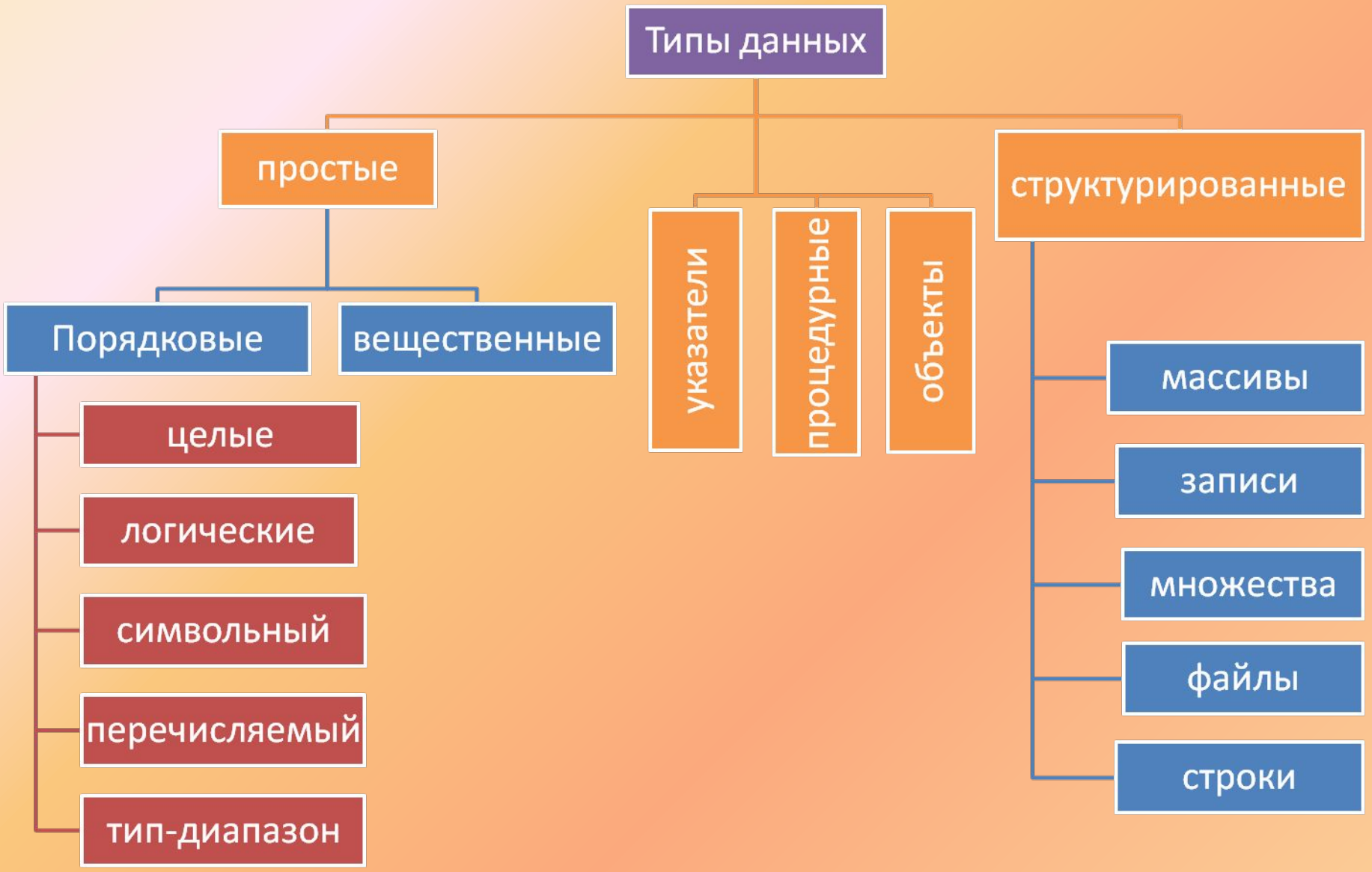
Описание меток

Раздел описания меток начинается с зарезервированного слова **label**, после которого следует список меток, перечисляемых через запятую. В качестве меток могут быть использованы идентификаторы и положительные целые числа:

label a1,l2,777777;

Описание типов

- *Раздел описания типов* начинается со служебного слова **type**, после которого следуют строки вида
- *имя типа = тип*;
- Например,
-
```
type
  myint = integer;
  arr10 = array [1..10] of integer;
  pinteger = ^integer;
  A = class
    i: integer;
    constructor Create(ii: integer);
    begin
      i:=ii;
    end;
  end;
```


Типы данных
простые
Порядковые
целые
логические
символьный
перечисляемый
тип-диапазон
вещественные
указатели
процедурные
объекты
структурированные
массивы
записи
множества
файлы
строки

Простые (порядковые) типы

Тип	Идентификатор	Размер в байтах	Диапазон значений
логический	boolean	1	true, false
символьный	char	1	все символы кода ASCII

Интервальный тип (тип-диапазон) определяется пользователем и формируется только из порядковых типов. Представляет собой подмножество значений в конкретном диапазоне.

Можно создать собственный тип данных простым перечислением значений, которые может принимать переменная данного типа. Это так называемый **перечисляемый тип данных**.

Целочисленные типы

Тип	Диапазон	Формат	Размер в байтах
Byte	0..255	Беззнаковый	1
ShortInt	–128..127	Знаковый	1
SmallInt	–32768..32767	Знаковый	2
Word	0..65535	Беззнаковый	2
Integer	-32768..32767	Знаковый	2
LongWord	0..4294967295	Беззнаковый	4
LongInt	–2147483648..2147483647	Знаковый	4
QWord	0..18446744073709551615	Беззнаковый	8

Вещественные типы

Идентификатор	Длина (байт)	Диапазон значений
real	6	$2,9x10^{-39}$ - $1,7x10^{38}$
single	4	$1,5x10^{-45}$ - $3,4x10^{38}$
double	8	$5x10^{-324}$ - $1,7x10^{308}$
extended	10	$3,4x10^{-4932}$ - $1,1x10^{4932}$

Структурированные типы

- **Массив** – это структура, занимающая в памяти единую область и состоящая из фиксированного числа компонентов одного типа.
- **Строки** представляет собой последовательность символов. Причем количество этих символов не может быть больше 255 включительно. Такое ограничение характерная черта Pascal.
- **Запись** – это структура, состоящая из фиксированного числа компонент, называемых полями. В разных полях данные могут иметь разный тип.
- **Множества** представляют собой совокупность любого числа элементов, но одного и того же перечисляемого типа.
- **Файлы** для Pascal представляют собой последовательности однотипных данных, которые хранятся на устройствах внешней памяти (кстати, жесткий диск – это тоже внешняя память).

Выражения

Выражение задает правило вычисления некоторого значения. Выражение состоит из констант, переменных, указателей функций, знаков операций и скобок.

Математические операции

Символ операции	Название операции	Пример
*	умножение	2*3 (результат: 6)
/	деление	30/2 (результат: 1.5E+01)
+	сложение	2+3 (результат: 5)
-	вычитание	5-3 (результат: 2)
div	целочисленное деление	5 div 2 (результат: 2)
mod	остаток от деления	5 mod 2 (результат: 1)

Логические операции

Над логическими аргументами в Паскале определены следующие операции:

NOT - логическое отрицание («НЕ»)

AND - логическое умножение ("И")

OR - логическое сложение («ИЛИ»)

XOR - логическое «Исключающее ИЛИ»

A	B	not A	A and B	A or B	A xor B
true	true	false	true	true	false
true	false		false	true	true
false	true	true	false	true	true
false	false		false	false	false

Операции отношения

> - больше **<** - меньше **=** - равно **<>** - не равно

>= - больше или равно **<=** - меньше или равно

В операциях отношения могут принимать участие не только числа, но и символы, строки, множества и указатели.

Приоритет операций

- унарная операция not, унарный минус -, взятие адреса @
- операции типа умножения: * / div mod and
- операции типа сложения: + - or xor
- операции отношения: = <> < > <= >= in

Порядок выполнения операций переопределить можно с помощью скобок.

Например 2*5+10 равно 20, но 2*(5+10) равно 30.

Стандартные функции Pascal

Обращение	Тип аргумента	Тип результата	Примечание
Abs(x)	Целый, вещественный	Целый, вещественный	Модуль аргумента
Sqr(x)	Целый, вещественный	Целый, вещественный	Квадрат аргумента
Sqrt(x)	Целый, вещественный	вещественный	Корень квадратный
Sin(x)	Целый, вещественный	вещественный	Синус, угол в радианах
Cos(x)	Целый, вещественный	вещественный	Косинус, угол в радианах

Стандартные функции Pascal

Обращение	Тип аргумента	Тип результата	Примечание
ArcTan(x)	Целый, вещественный	вещественный	Арктангенс (значение в радианах)
Exp(x)	Целый, вещественный	вещественный	Экспонента
Frac(x)	вещественный	вещественный	Дробная часть числа
Int(x)	Целый, вещественный	вещественный	Целая часть числа
Ln(x)	Целый, вещественный	вещественный	Логарифм натуральный
Pi	-	вещественный	3,141592653

Стандартные функции Pascal

Обращение	Тип аргумента	Тип результата	Примечание
Random	-	вещественный	Псевдослучайное число в интервале [0, 1]
Round(x)	вещественный	целый	Округление до ближайшего целого
Trunc(x)	вещественный	целый	Отбрасывание дробной части числа

Возведение числа в степень

x^y **EXP(y*Ln(x))**

оператор присваивания

<имя_переменной>:=<выражение>

Примеры:

$y = \sqrt{x}$ $y := sqr(x);$

$y = |x - 5|$ $y := abs(x - 5);$

$y = \sin^2 x$ $y := sqr(\sin(x));$

$y = \frac{x + 5}{2}$ $y := (x + 5)/2;$

Ввод и вывод данных

- **Ввод данных**
  - **read(<список ввода>);**
  - **readln(<список ввода>);**

Примеры:

- **read(a,b,c);**{где a,b,c - переменные. Ввод данных осуществляется через пробел}
- **readln(a,b,c);**{где a,b,c - переменные. Ввод данных осуществляется через enter}
- Список вывода может содержать константы, переменные, выражения, формат вывода. Выражения в списке вывода разделяются запятыми.

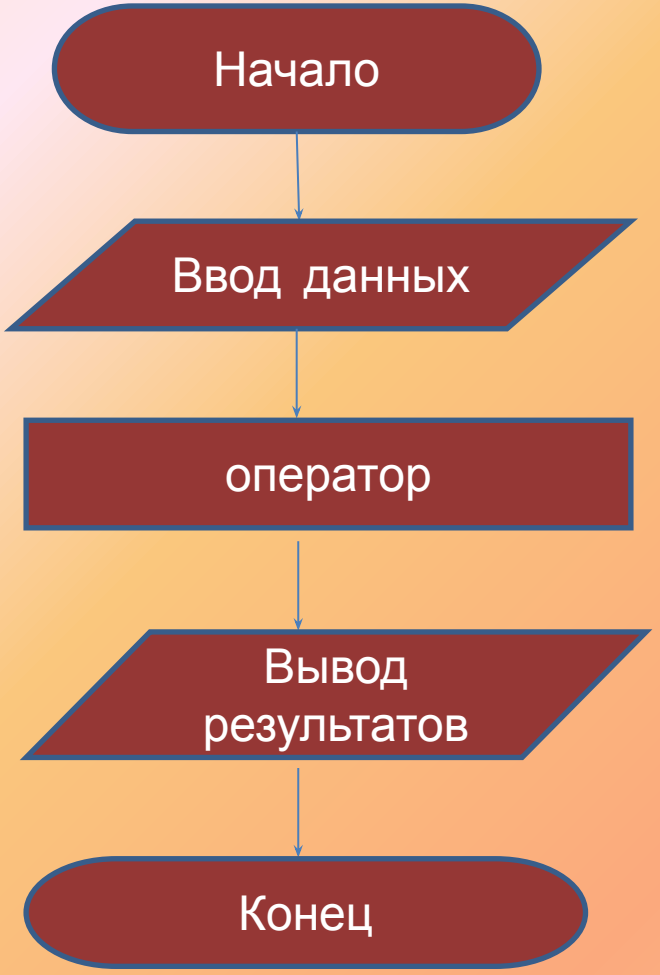
Ввод и вывод данных

- **Вывод данных**
  - **write(<список вывода>);**
  - **writeln(<список вывода>);**

Примеры:

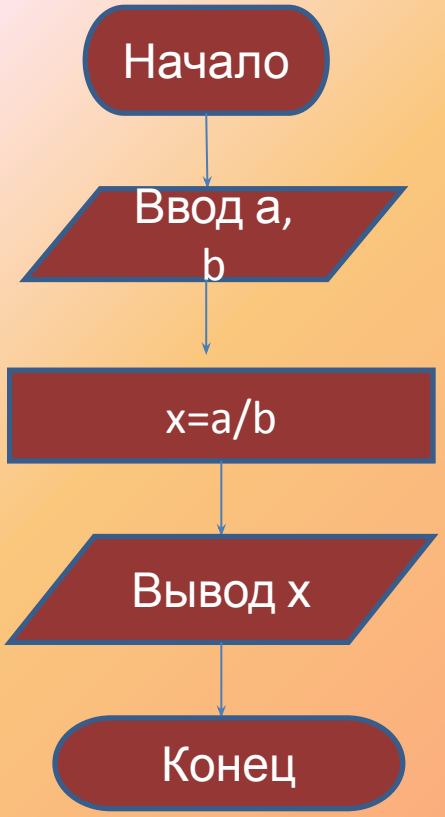
- **write(a,b,c);**{где a,b,c - переменные. После вывода данных на экран, курсор останется на последнем символе}
- **writeln(a,b,c);**{где a,b,c - переменные. После вывода данных на экран, курсор перейдет на новую строку)}
- Окончание **ln** в имени процедуры означает, что курсор автоматически будет переведен в начало следующей строки экрана.

Линейный алгоритм




```
Program имя_программы;
  var
    {описание данных}
  begin
    readln(ввод данных);
    оператор
    writeln(вывод
результатов);
  end;
```

Пример: Даны 2 целых числа, найти частное этих чисел




```
program E1;
  var
    a,b: integer;
    r: real;
  begin
    readln(a,b);
    x := a/b;
    writeln(x);
  end;
```

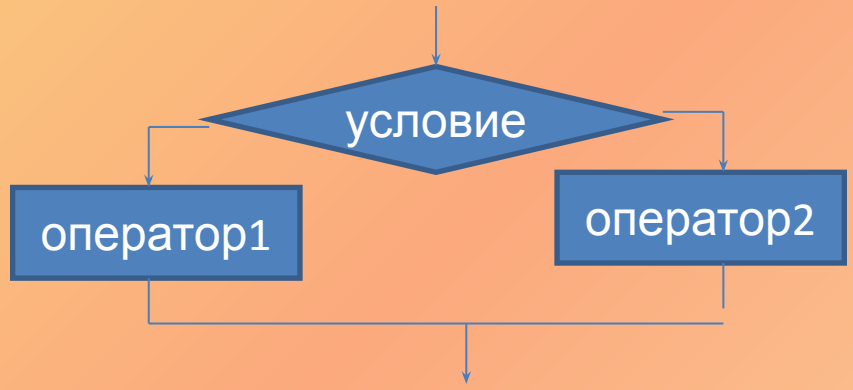
Задачи на линейный алгоритм

- Вычислите длину окружности, площадь круга и объём шара одного и того же заданного радиуса.
- Вычислите периметр и площадь прямоугольного треугольника по длинам двух его катетов.
- По координатам трёх вершин некоторого треугольника найдите его площадь и периметр.
- Вычислите дробную часть среднего геометрического трёх заданных вещественных чисел.
- Вычислите площадь треугольника по формуле Герона.

Условный оператор

Полный условный оператор

- **IF** *условие* **THEN** *оператор1*
  ELSE *оператор2*;
- **IF** *условие* **THEN**
  BEGIN
  оператор1_1;
  оператор1_2;
  END
  ELSE
  BEGIN
  оператор2_1;
  оператор2_2;
  END;



Перед **ELSE** точка с запятой никогда не ставится!!!

Условный оператор

Неполный условный оператор

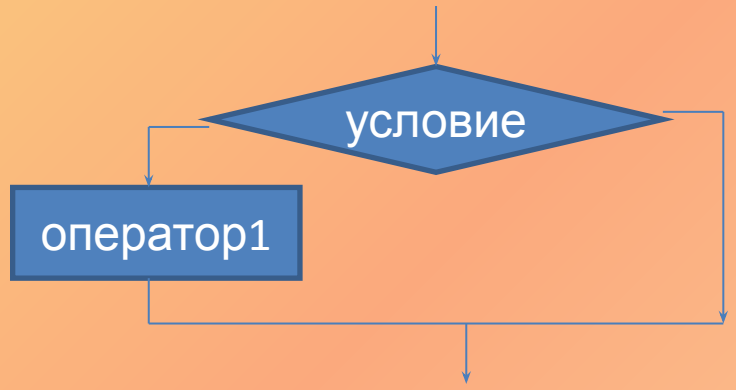
- **IF** *условие* **THEN** *оператор1 ;*
- **IF** *условие* **THEN**

  BEGIN

  оператор1_1;

  оператор1_2;

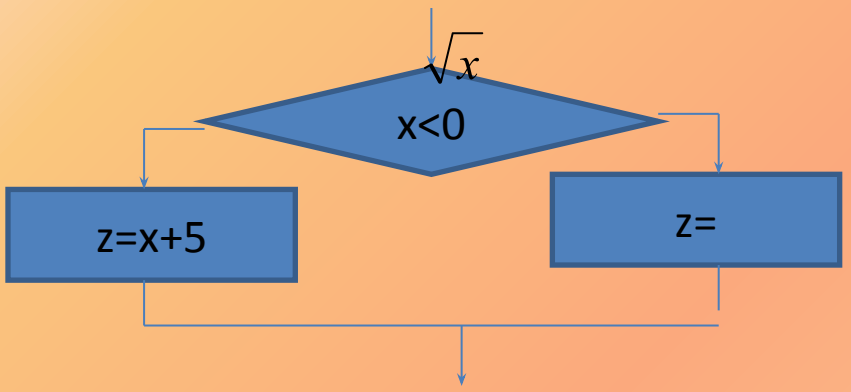
  END;



условие - это логическое выражение, в зависимости от которого выбирается одна из двух альтернативных ветвей алгоритма. Если значение условия истинно (TRUE), то будет выполняться *оператор 1*, записанный после ключевого слова **then**. В противном случае будет выполнен *оператор 2*, следующий за словом **else**, при этом *оператор 1*пропускается. После выполнения указанных операторов программа переходит к выполеннию команды, стоящей непосредственно после оператора **if**.

Пример: Вычислите значение функции

$$z = \begin{cases} x + 5, & \text{при } x < 0 \\ \sqrt{x}, & \text{при } x \geq 0 \end{cases}$$



IF *x<0* **THEN** *z:=x+5* **ELSE** *z:=sqrt(x);*

Задачи на условный оператор

- Определите, является ли заданное целое число *A* нечётным числом.
- Определите, имеется ли среди заданных целых чисел *A*, *B*, *C* хотя бы одно чётное.
- Даны три числа. Выберите те из них, которые принадлежат заданному отрезку [*a*, *b*].
- Для заданных вещественных чисел *a*, *b* и *c* определите, имеет ли уравнение $ax^2 + bx + c = 0$ хотя бы одно вещественное решение.
- Вычислите площадь кольца, ширина которого равна *H*, а отношение радиуса большей окружности к радиусу меньшей окружности равно *D*.
- Заданы площади круга и квадрата. Определите, поместится ли квадрат в круге.

Оператор выбора (варианта)

- case *выражение* of
  вариант : *оператор*;
  ...
  вариант : *оператор*;
  end;
- case *выражение* of
  вариант : *оператор*;
  ...
  вариант : *оператор*;
  else *оператор*
  end;

Пример:

```
case ch of
'A'..'Z', 'a'..'z' : WriteLn ('Буква');
 '0'..'9'          : WriteLn ('Цифра');
'+', '-', '*', '/' : WriteLn ('Оператор');
else WriteLn ('Специальный символ')
end;
```

Циклы

Если заранее известно количество необходимых повторений, то цикл называется ***арифметическим***. Если же количество повторений заранее неизвестно, то говорят об ***итерационном*** цикле.

В итерационных циклах производится проверка некоторого условия, и в зависимости от результата этой проверки происходит либо выход из цикла, либо повторение выполнения тела цикла. Если проверка условия производится перед выполнением блока операторов, то такой итерационный цикл называется циклом ***с предусловием*** (цикл "пока"), а если проверка производится после выполнения тела цикла, то это цикл ***с постусловием*** (цикл "до").

Особенность этих циклов заключается в том, что тело цикла с постусловием всегда выполняется хотя бы один раз, а тело цикла с предусловием может ни разу не выполниться.

Арифметические циклы

- **FOR** *переменная*:= нач_*знач* **TO** кон_*знач* **DO** *оператор;*
- **FOR** *переменная*:= нач_*знач* **TO** кон_*знач* **DO** ***BEGIN***

  оператор1;

  оператор2;

  …

  END;

Переменная цикла, начальное и конечное значения должны иметь порядковый тип. Со словом **TO**, значение переменной цикла **увеличивается** на 1 при каждой итерации цикла.

Арифметические циклы

- **FOR** *переменная*:= нач_*знач* **DOWNTO** кон_*знач* **DO** *оператор;*
- **FOR** *переменная*:= нач_*знач* **DOWNTO** кон_*знач* **DO** ***BEGIN***

  оператор1;

  оператор2;

  …

  END;

Со словом **DOWNTO**, значение переменной цикла **уменьшается** на 1 при каждой итерации цикла. Не следует самостоятельно изменять значение управляющей переменной внутри цикла.

Пример 1. Квадраты чисел от 2-х до 10-и.

```
FOR x:=2 TO 10 DO
  WriteLn(x*x);
```

Пример 2. Латинский алфавит.

```
FOR ch:='A' TO 'Z' DO
  Writeln(ch);
```

Пример 3. Использование цикла с downto.

```
FOR i:=10 DOWNTO 1 DO
  WriteLn(i);
```

Пример 4. Использование составного оператора.

```
FOR x:=1 TO 10 DO
begin
y:=2*x+3;
WriteLn('f(',x,')=',y);
end;
```

Цикл с предусловием

- **WHILE** *выражение* **DO**
  оператор;
- **WHILE** *выражение* **DO**
  BEGIN
  оператор1;
  оператор2;
  …
  END;

Оператор после **DO** будет выполняться до тех пор, пока логическое *выражение* принимает истинное значение. Логическое *выражение* является условием возобновления цикла. Его истинность проверяется каждый раз перед очередным повторением *оператора* цикла, который будет выполняться лишь до тех пор, пока логическое *выражение* истинно. Как только логическое *выражение* принимает значение ложь, осуществляется переход к оператору, следующему за **WHILE**.

Цикл с постусловием

REPEAT
оператор;
оператор;
...
оператор
UNTIL *выражение;*

Операторы между словами **REPEAT** и **UNTIL** повторяются, пока логическое *выражение* является ложным. Как только логическое *выражение* становится истинным, происходит выход из цикла.

Так как *выражение* оценивается после выполнения *операторов*, то в любом случае *операторы* выполнятся хотя бы один раз.