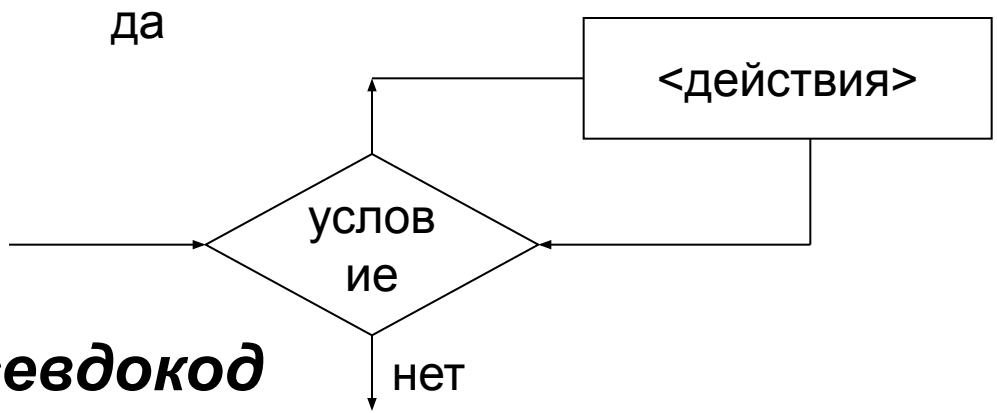
# Циклы

- Цикл-пока (с предусловием)

***Блок-схема***



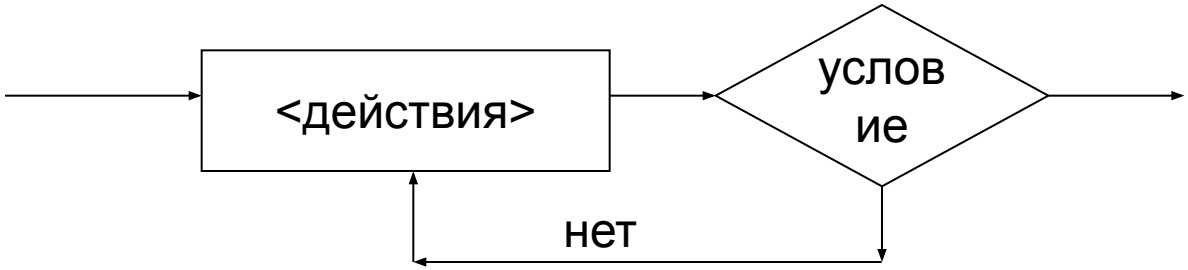

***Псевдокод***

цикл-пока <условие>
    <действия>
кц

Выполнение цикла продолжается пока условие истинно и завершается, когда условие станет ложным.

- Цикл-до (с постусловием)

***Блок-схема***




***Псевдокод***

цикл

<действия>

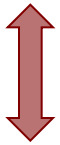
до <условие>

кц

Выполнение цикла продолжается пока условие ложно и завершается, когда условие станет истинным.

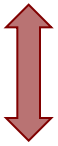
Циклы с предусловием и с постусловием являются взаимозаменяемыми

```
цикл-пока <условие>
    <действия>
кц
```



```
если <условие> то
цикл
    <действия>
до не <условие>
кц
все
```

```
цикл
    <действия>
до <условие>
кц
```



```
<действия>
цикл-пока не <условие>
    <действия>
кц
```

# Язык Pascal

*Назван в честь французского математика Блеза Паскаля (1623-1662). Создан как учебный язык программирования в 1968 –1971 гг. Никлаусом Виртом в Высшей технической школе в Цюрихе. Предназначен для обучения студентов основам структурного программирования.*

# Алфавит языка

## и особенности использования символов

- Символы, используемые при составлении идентификаторов – латинские буквы (строчные и прописные), цифры, подчеркивание _.
- Разделители – пробел, табуляция, новая строка.
- Специальные символы – используются при построении конструкций языка

  + - * / { } [ ] ( ) < > . , ‘ : ; @ # $ ^
- Составные символы – воспринимаются компилятором как единое целое

  <= >= := (* *) ..
- Русские буквы и символы псевдографики – могут использоваться в комментариях и сообщениях.
- Зарезервированные слова – имеют определенный смысл для компилятора.

# Структура программы

**program** <имя программы>;
<раздел подключения модулей>
<раздел описаний>
**begin**
<операторы>
**end**.

Первая строка называется *заголовком программы* и не является обязательной.

<Раздел подключения модулей> начинается со служебного слова **uses**, затем список имен модулей, перечисляемых через запятую.

<Раздел описаний> может включать разделы описания переменных (Var), констант (Const), типов (Type), процедур (Procedure), функций (Function), которые следуют друг за другом в произвольном порядке.

<Раздел подключения модулей> и <раздел описаний> могут отсутствовать.

Операторы отделяются один от другого символом "точка с запятой".

# Идентификаторы и служебные слова

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

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

**Служебные слова** служат для оформления конструкций языка и не могут быть использованы в качестве имен.

*Список всех служебных слов языка **Pascal ABC** приведен ниже:*

***and, array, as, begin, break, case, class, const,constructor, continue, destructor, div, do, downto, else, end, exit, external, externalsync, file, finalization, for, forward, function, if, in, inherited, initialization, is, mod, not, of, or, private, procedure, program, property, protected, public, record, repeat, set, shl, shr, sizeof, string, then, to, type, unit, until, uses, var, while, with, xor.***

# Описание переменных

**Раздел описания переменных** начинается со служебного слова **var**, после которого следуют строки вида

**<список имен переменных>: тип;**

Имена в списке перечисляются через запятую. Например:

```
var
    a,b,c: integer;
    d: real;
    e, f: integer;
    s,s1: string;
    ch: char;
   Mass: array [1..5] of integer ;
```

# Обзор типов

В **Pascal ABC** имеются следующие типы:

- **integer** (целый)
- **byte** (байтовый)
- **char** (символьный)
- перечислимый
- тип-диапазон

- **boolean** (логический)
- **real** (вещественный)
- **complex** (комплексный)
- **string** (строковый)
- **array**(массив)
- **record** (запись)
- указатель
- процедурный
- **file**(файловый)

# Классификация типов

- Типы integer, byte, char, перечислимый и диапазонный называются ***порядковыми***. Только значения этих типов могут быть индексами массивов и фигурировать в качестве выражения-переключателя в операторе **case**. Переменная-параметр цикла **for** также должна иметь перечислимый тип.
- Все порядковые типы, а также типы **boolean**, **real** и **complex** называются ***простыми*** типами

# Описание констант

**Раздел описания именованных констант** начинается со служебного слова **const**, после которого следуют строки вида

<имя константы> = <значение>;

или <имя константы >: <тип> = <значение>;

Например:

```
const
    Pi = 3.14;
    Count_n: integer = 10;
    Name = 'Mike';
```

# Описание типов

**Раздел описания типов** начинается со служебного слова **type**, после которого следуют строки вида

<имя типа> = <тип>;

Например,

```
type
    myint = integer;
    mass = array [1..10] of integer;
```

# Операторы вывода

- Для вывода в окно вывода используются стандартные процедуры **write** и **writeln**. Они могут вызываться как без параметров, так и со списком параметров. Параметры в списке перечисляются через запятую и должны иметь простой тип (кроме перечислимого типа и интервального типа, построенного на базе перечислимого), либо тип **string**, либо тип указателя. Процедура **writeln** после вывода осуществляет переход на следующую строку.

Например,

```
Var a, b : integer;
…
A:=15; b:=123;
Writeln('A=',a); Write ('B=',b);
```

```
A=15
B=123 _
```

_ обозначает положение курсора после вывода.

# Форматный вывод

- В процедурах вывода **write** и **writeln** после каждого выводимого значения типа может указываться ***формат вывода***, представляющий собой двоеточие, после которого следует целое число или выражение. Это число или выражение задает ***ширину поля вывода***, то есть количество позиций, отводимых под выводимое значение.
- Если длина выводимого значения меньше ширины поля вывода, то выводимый текст дополняется ***слева*** пробелами до нужной ширины; в результате выводимое значение ***выравнивается по правому краю***.
- Если длина выводимого значения больше ширины поля вывода, то формат вывода игнорируется.

# Примеры форматного вывода

- Например, если ***a***, ***b*** - целые переменные, то при выполнении операторов

  a:=-2437; b:=13555;writeln(a:6,'Привет!':9);
  writeln(b:1);

  в окно вывода будет выведен текст:

```
_-2437_ _Привет!
13555
```

  Для вещественных значений можно также использовать формат :***m:n***, где ***m*** и ***n*** - целые значения. Значение ***m*** задает *ширину поля вывода*, а значение ***n*** - *количество знаков после десятичной точки*. Например:

  writeln(-14.859:10:3); // _ _ _-14.859
  writeln(-14.859:10:5); // _-14.85900
  writeln(-14.859:10:0); // _ _ _ _ _ _ _-15
  writeln((0,1):10:1); // _(0.0,1.0)

  Вещественные и комплексные значения с форматом вывода вида :***m*** всегда выводятся в экспоненциальной форме.