Тема: Язык программирования Pascal

Языки программирования

- **Машинно-ориентированные (низкого уровня)** - каждая команда соответствует одной команде процессора (ассемблер)
- **Языки высокого уровня** – приближены к естественному (английскому) языку, легче воспринимаются человеком, **не зависят от конкретного компьютера**
  - *для обучения*: Бейсик, ЛОГО, Паскаль
  - *профессиональные*: Си, Фортран, Паскаль, Делфи
  - *для задач искусственного интеллекта*: *Пролог, ЛИСП*
  - *для Интернета*: *JavaScript, Java, Perl, PHP, ASP*

Язык Паскаль разработан в 1971 году и назван в честь Блеза Паскаля – французского ученого, изобретателя механической вычислительной машины.

Автор языка Паскаль – швейцарский профессор ***Никлаус Вирт.***

Паскаль – это универсальный язык программирования, позволяющий решать самые разнообразные задачи обработки информации

Алфавит языка

1. Символы, используемые в идентификаторах

- **латинские буквы (A-Z)**

> **заглавные и строчные буквы не различаются**

- **цифры**

> **имя не может начинаться с цифры**

- **знак подчеркивания _**

2. Разделители

- **любой управляющий символ (коды от 0 до 31)**
- **пробел**
- **Комментарий – { }; * ***

3. Специальные символы

- **знаки пунктуации [ ],( ), { }, * *, :=, .. , #, $**
- **знаки операций: буквенные (not, div, or, mod) и небуквенные (+, =, *, /, <, >, <>, <=, >=)**
- **зарезервированные слова (begin, end)**

4. Неиспользуемые символы (буквы рус. алфавита, %, &)

Структура программы

```
program <имя программы>;
uses crt;
const …;{константы}
var …; {переменные}
   { процедуры и функции }
begin
clrscr;
 … {основная программа}
readkey;
end.
```

комментарии в фигурных скобках не обрабатываются

Программа

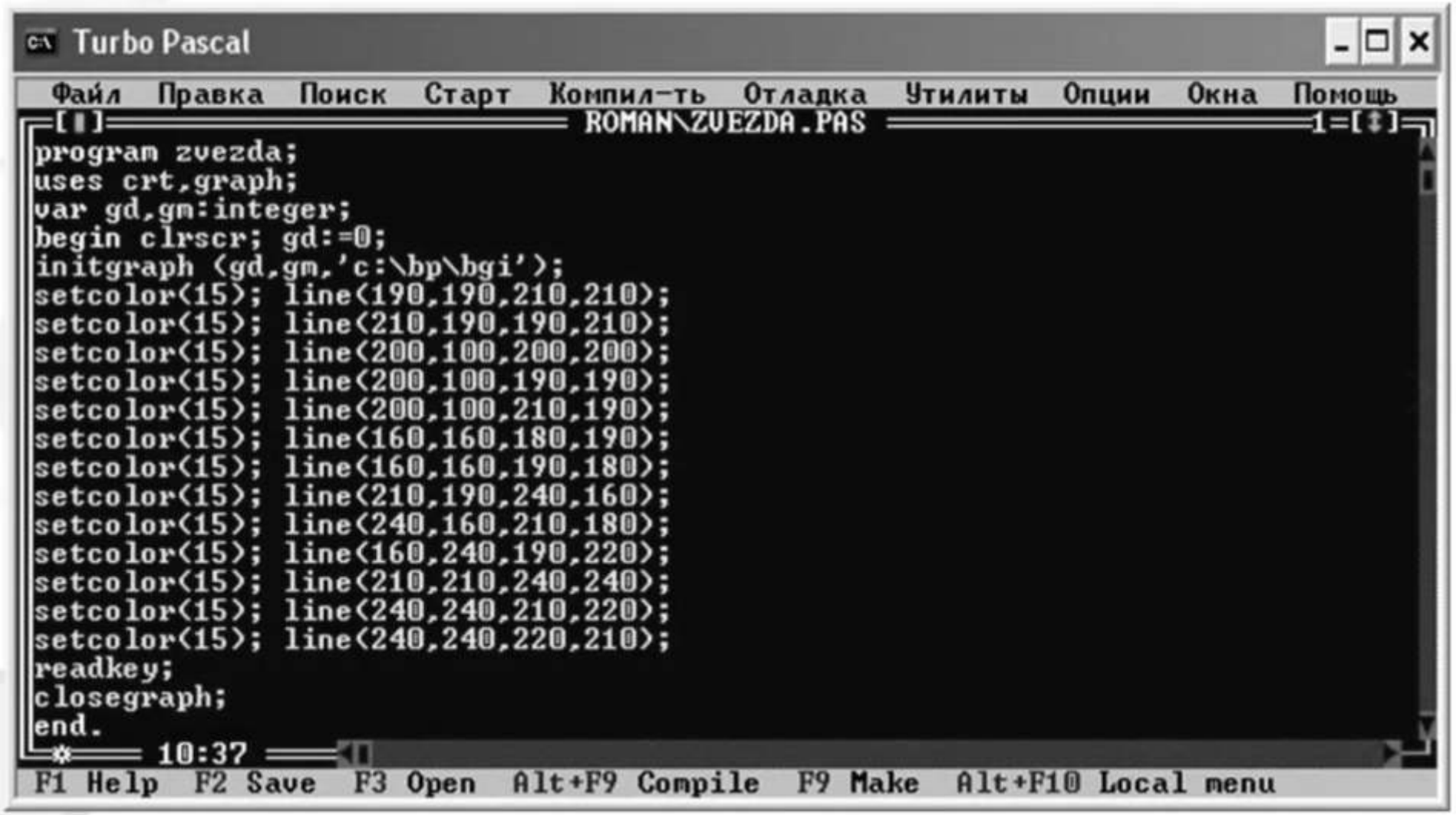
```
program qq;
var s: string;
    K, i, count: integer;
    procedure Rec(p: integer);
    ...
    end;
begin
  writeln('Введите длину слов:');
  read ( K );
  s := '';
  for i:=1 to K do s := s + ' ';
  count := 0;
  Rec ( 1 );
  writeln('Всего ', count, ' слов');
end.
```

глобальные переменные

процедура

строка из K пробелов

Текст программы



Структура программ на языке Паскаль в IDE LAZARUS

```
program LAZA_OBORUD;

uses
Forms, Unit1, dbflaz, tachartlazaruspkg

{$R *.res}

begin
  RequireDerivedFormResource:=True;
  Application.Scaled:=True;
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.Run;
end.
```

```
unit Unit1;                    НАЧАЛО МОДУЛЯ

interface

uses
  Classes, SysUtils, DB, BufDataset, dbf, SQLDB,
SQLite3Conn;
const
  Help_String: array[1..3] of string =('111','222','333');
type
  TForm1 = class(TForm)
           Button1: TButton;
           Button2: TButton;
     ComboBox1: TComboBox;
           Label17: TLabel;
               Edit1: TEdit;
SpeedButton17: TSpeedButton;
            Image1: TImage;
            Memo1: TMemo;
    procedure Button10Click(Sender: TObject);
    procedure Button11Click(Sender: TObject);
```

Private **ПРОДОЛЖЕНИЕ МОДУЛЯ**

ПРИВАТ МАТЕРИАЛ

public

МАТЕРИАЛ ДЛЯ ОБЩЕСТВЕННОСТИ

end;

```
var
        MyStr  : TStream;
         Form1: TForm1;
NomerStroki: integer;
         MyDir: string;

implementation

{$R *.lfm}
procedure TForm1.DBEdit1Click(Sender: TObject);
begin
Index_Stroki.Text:=SQLQuery1.Fields.FieldByName('IndexMy').As
String;
end;
…
END.
```

Основные понятия

Константа – постоянная величина, имеющая имя.

Переменная – изменяющаяся величина, имеющая имя (ячейка памяти).

Процедура – вспомогательный алгоритм, описывающий некоторые действия (рисование окружности).

Функция – вспомогательный алгоритм для выполнения вычислений (вычисление квадратного корня, **sin**).

Константы

```
const
      i2 = 45; { целое число }
      pi = 3.14; { вещественное  число }
```

целая и дробная часть отделяются точкой

```
      qq = 'Вася'; { строка символов }
```

можно использовать русские буквы!

```
      L  = True; { логическая величина }
```

может принимать два значения:

- True (истина, "да")
- False (ложь, "нет")

Переменные

Переменная – это величина, имеющая имя, тип и значение. Значение переменной можно изменять во время работы программы.

Типы переменных:

- integer { целая }
- real { вещественная }
- char { один символ }
- string { символьная строка }
- boolean { логическая }

Объявление переменных (выделение памяти):

```
var a, b: integer;
    Q: real;
    s1, s2: string;
```

Вопрос 2

Типы данных

Простые типы данных

В языке Pascal используются различные **типы данных**. Мы будем пользоваться только некоторыми из них – простыми типами данных.

Название	Обозначение	Допустимые значения	Область памяти
Целочисленный	integer	- 32 768 ... 32 767	2 байта со знаком
Вещественный	real	$\pm(2.9 * 10^{-39} \ldots 1.7 * 10^{+38})$	6 байтов
Символьный	char	Произвольный символ алфавита	1 байт
Строковый	string	Последовательность символов длиной меньше 255	1 байт на символ
Логический	boolean	True и False	1 байт
Целочисленный	Byte	0...255	1 байт

Типы данных

- byte { целые 0..255 }
- shortint { целые -128..128 }
- word { целые 0..65535 }
- longint { целые -2147483648..2147483647 }
- single { вещественная, 4 байта }
- real { вещественная, 6 байта }
- double { вещественная, 8 байтов }
- extended { вещественная, 10 байтов }
- boolean { логическая, 1 байт }
- char { символ, 1 байт }
- string { символьная строка }

Символьный тип

Множество значений *символьного типа есть* множество символов, упорядоченных в соответствии с их **ASCII**-кодами.

Любое значение символьного типа может быть получено с помощью стандартной функции ***Chr*** *из его кода ASCII.*

Пример:

```
var ch: Char;
...
ch:= 'A';
ch:= Chr(32);{ ch:= ' ';}
```

Строковые величины

Строка – это последовательность символов кодовой таблице.

Длина строки (количества символов) может лежать в диапазоне 0..255

Для определения длины данных строкового типа используется идентификатор string, за которым следует максимальное значение длины строки данного типа.

```
var s: string[20];
```

В программе значения переменных и констант типа char (символьный) заключается в апострофы.

Например, st:=’река’

Операции со строками

```
var s, s1, s2: string;
```

Запись нового значения:

```
s := 'Вася';
```

Объединение: добавить одну строку в конец другой.

```
s1 := 'Привет';
s2 := 'Вася';
s := s1 + ', ' + s2 + '!';
```

`'Привет, Вася!'`

Подстрока: выделить часть строки в другую строку.

```
s := '123456789';

s1 := Copy ( s, 3, 6 );
s2 := Copy ( s1, 2, 3 );
```

с 3-его символа

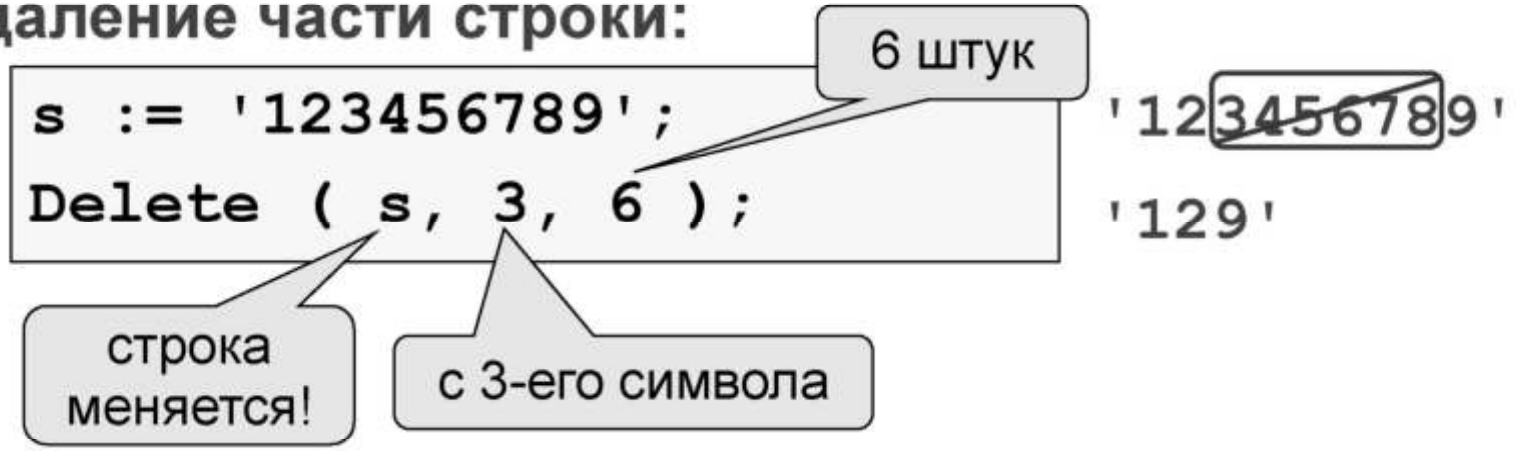
6 штук

`'345678'`

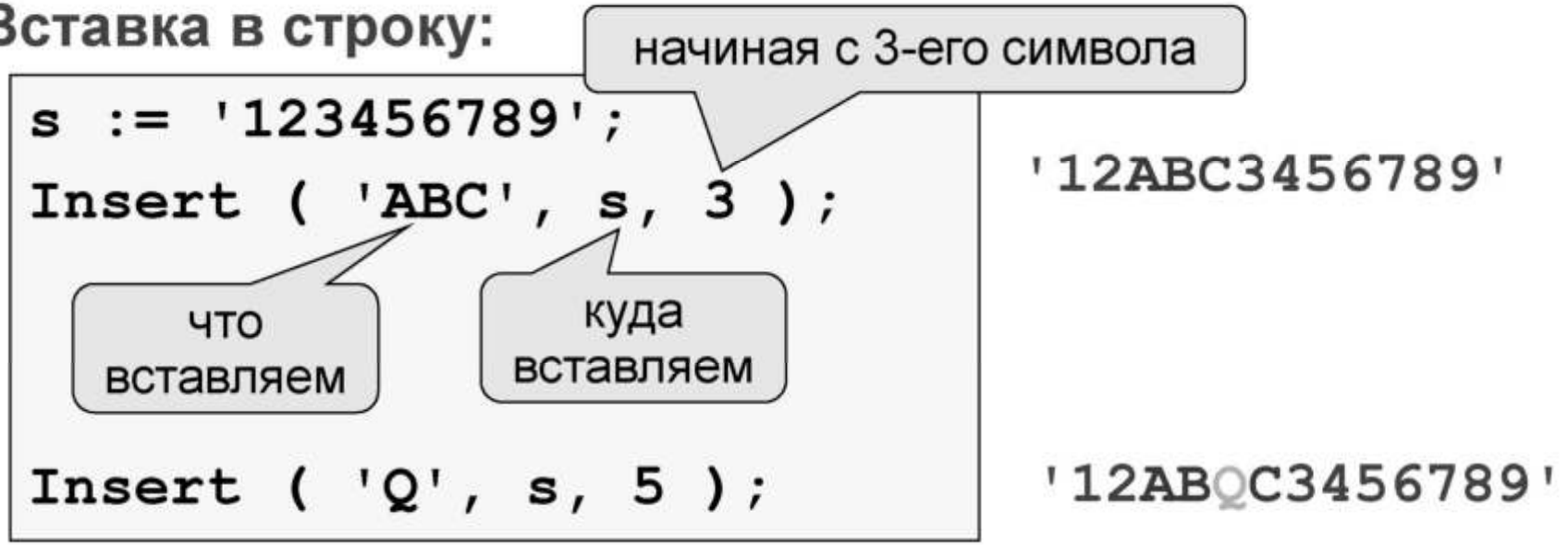
`'456'`

Удаление и вставка

Удаление части строки:



Вставка в строку:



Массивы

Массив – это группа однотипных элементов, имеющих общее имя и расположенных в памяти рядом.

Особенности:

- все элементы имеют **один тип**
- весь массив имеет **одно имя**
- все элементы расположены в памяти **рядом**

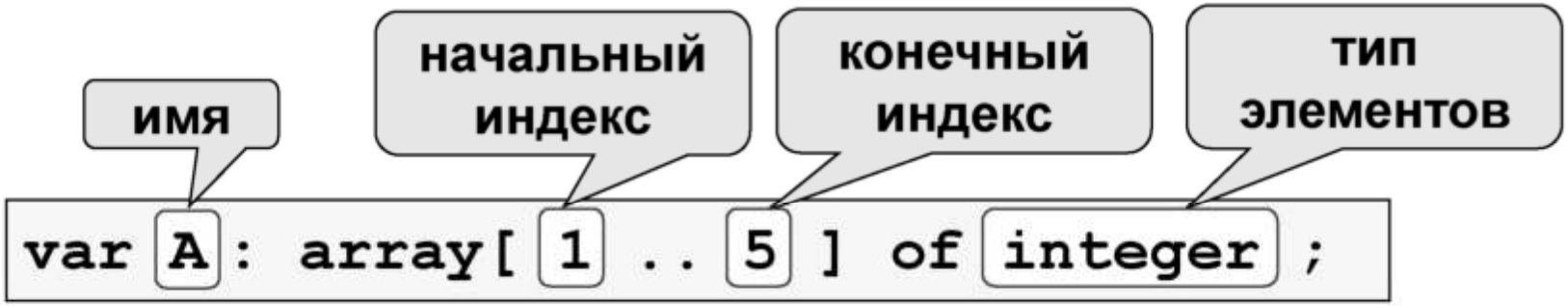
Примеры:

- список учеников в классе
- квартиры в доме
- школы в городе
- данные о температуре воздуха за год

Объявление массивов

- определить **имя** массива
- определить **тип** массива
- определить **число элементов**
- выделить **место в памяти**

Массив целых чисел:



Размер через константу:

```
const N=5;
var A: array[1..N] of integer;
```

Массивы

Объявление:

```
const N = 5;
var a: array[1..N] of integer;
    i: integer;
```

Ввод с клавиатуры:

```
for i:=1 to N do begin
  write('a[', i, ']=');
  read ( a[i] );
end;
```

```
a[1] = 5
a[2] = 12
a[3] = 34
a[4] = 56
a[5] = 13
```

Поэлементные операции:

```
for i:=1 to N do a[i]:=a[i]+1;
```

Вывод на экран:

```
writeln('Массив A:');
for i:=1 to N do
  write(a[i]:4);
```

```
Массив A:
   6   13   35   57   14
```

Массивы

Объявление:

```
const N = 5;
var a: array[1..N] of integer;
    i: integer;
```

Ввод с клавиатуры:

```
for i:=1 to N do begin
  write('a[', i, ']=');
  read ( a[i] );
end;
```

Поэлементные операции:

```
for i:=1 to N do a[i]:=a[i]*2;
```

Запись

Запись — структурированный тип данных. Записи являются неоднородными неупорядоченными структурами с прямым доступом к компонентам. Компоненты записи называют **полями записи**

```
Type
TDate=record
        Day: 1..31;
        Month: 1..12;
        Year: Word;
    end;

pol=(m,w);
people=record
          fam:string[20];
          BDay: TDate;
          case mw:pol of
             m: ( voen: boolean; spec: string[15]);
            w: ( merry: boolean; child: byte)
          end;
end;
```

```
Var
 p1, p2: people;

P1.mw:=m;
p1.voen:=true;
p2.child:=2;

With p1.Bday do
     begin
       Day:=12;
       Month: =1;
       Year:=1994;
     end;
```

Структурированные типы в Pascal

Pascal | **Записи** | Pascal

Действия над записями

- **Значения переменных типа запись можно присваивать другим переменным того же типа (A := B).**

С записями целиком допустима только операция присваивания, все остальные действия выполняются с отдельными полями записи.

- **Есть два способа доступа к полю записи:**

1. **С помощью составного имени, его формат:**

<имя_записи>.<имя_поля>

Например: A.year := 1994; D.year := 1947;

2. **С помощью оператора присоединения with. Его формат:**

with <переменная> do <оператор>

где **переменная** – *имя переменной типа запись, за которым может следовать список вложенных полей,*

with, do – *ключевые слова,*

оператор – *любой оператор Pascal.*

Например:

```
with A do begin
  year := 1994;
  day := 8;
  month := 11
end;
```

Оператор With удобно использовать, когда требуется обратиться к нескольким полям одной и той же записи.

Записи (Record) в Паскале

Описание анкеты студента в Паскале будет выглядеть так:

```
Type anketa=record
    fio: string[45];
    pol: char;
    dat_r: string[8];
    adres: string[50];
    curs: 1..5;
    grupp: string[3];
  end;
```

Такая запись Паскаля, так же как и соответствующее ей дерево, называется двухуровневой.

Файлы

Файл – это область на диске, имеющая имя.

Текстовые

только текст без оформления, не содержат управляющих символов (с кодами < 32)

ACSII (1 байт на символ)

UNICODE (2 байта на символ)

`*.txt, *.log,`

`*.htm, *.html`

Двоичные

могут содержать любые символы кодовой таблицы

`*.doc, *.exe,`

`*.bmp, *.jpg,`

`*.wav, *.mp3,`

`*.avi, *.mpg`

Папки (каталоги)

Принцип сэндвича

Переменная типа "текстовый файл":
`var f: text;`

I этап. открыть файл :

- связать переменную **f** с файлом

```
assign(f, 'qq.dat');
```

- открыть файл (сделать его активным, приготовить к работе)

```
reset(f); {для чтения}
```

```
rewrite(f); {для записи}
```

```
append(f); {дописывать данные}
```

II этап: работа с файлом

```
read ( f, n );  { ввести значение n }
```

```
write ( f, n );  { записать значение n }
writeln ( f, n );{с переходом на нов.строку }
```

III этап: закрыть файл

```
close(f);
```

Работа с файлами

Особенности:

- имя файла упоминается только в команде **`assign`**, обращение к файлу идет через файловую переменную
- файл, который открывается на чтение, должен **существовать**
- если файл, который открывается на запись, существует, старое содержимое **уничтожается**
- данные записываются в файл в текстовом виде
- при завершении программы все файлы закрываются автоматически
- после закрытия файла переменную **`f`** можно использовать еще раз для работы с другим файлом

Программа

```
program qq;
var s, x: integer;
    f: text;
begin
   assign(f, 'input.txt');
   reset(f);
   s := 0;
   while not eof(f) do begin
      readln(f, x);
      s := s + x;
   end;
   close(f);
   assign(f, 'output.txt');
   rewrite(f);
   writeln(f, 'Сумма чисел ', s);
   close(f);
end.
```

логическая функция, возвращает **True**, если достигнут конец файла

запись результата в файл **output.txt**

Обработка текстовых данных

Алгоритм:

пока не кончились данные

1. Прочитать строку из файла (`readln`).
2. Удалить все сочетания ", *короче,*" (`Pos, Delete`).
3. Перейти к шагу 1.

Обработка строки `s`:

искать ", короче,"

удалить 9 символов

```
repeat
  i := Pos(', короче,', s);
  if i <> 0 then Delete(s, i, 9);
until i = 0;
```

Особенность:

надо одновременно держать открытыми два файла (один в режиме чтения, второй – в режиме записи).

Работа с файлами

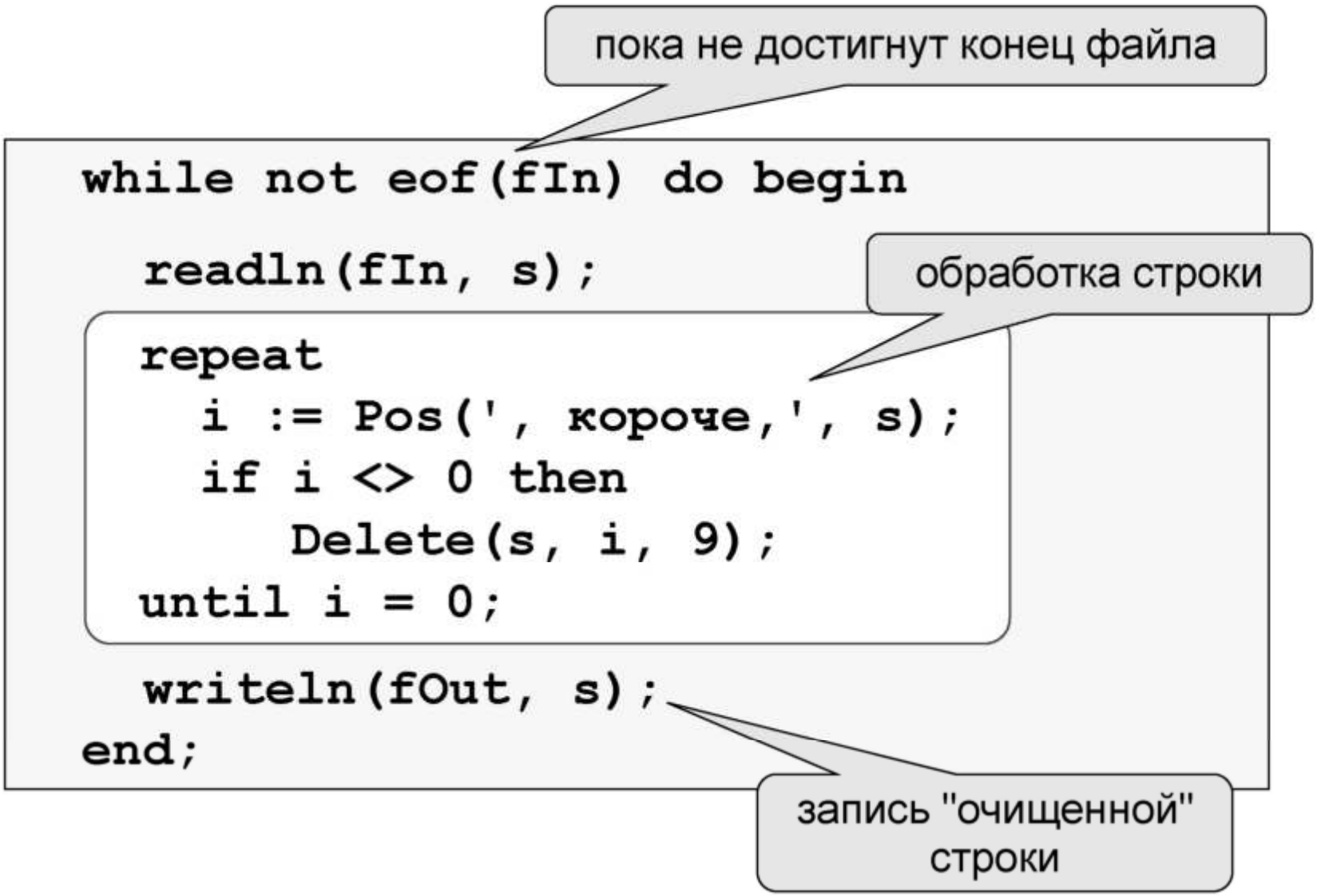
```
program qq;
var s: string;
    i: integer;
    fIn, fOut: text;
begin
  assign(fIn, 'instr.txt');
  reset(fIn);
  assign(fOut, 'outstr.txt');
  rewrite(fOut);
  ... { обработать файл }
  close(fIn);
  close(fOut);
end.
```

файловые переменные

открыть файл для чтения

открыть файл для записи

Полный цикл обработки файла



Вопрос 3

Операторы языка Паскаль

Операторы языка Паскаль

- **Простые операторы** (оператор присваивания, оператор безусловного перехода Goto, пустой оператор)
- **Структурированные операторы** (составной оператор, условный оператор IF, условный оператор CASE, оператор цикла REPEAT, оператор цикла WHILE, оператор цикла FOR)

Условный оператор IF

```
if <условие> then begin
   {что делать, если условие верно}
end
else begin
   {что делать, если условие неверно}
end;
```

Особенности:

- перед ***else*** НЕ ставится точка с запятой
- вторая часть (***else*** …) может отсутствовать (неполная форма)
- если в блоке один оператор, можно убрать слова ***begin*** и ***end***

Разветвляющиеся алгоритмы

Алгоритмы, в которых последовательность шагов зависит от выполнения некоторых условий, называются **разветвляющимися.**

Задача. Ввести два целых числа и вывести на экран наибольшее из них.

Программа

```
uses crt;
var a, b, max: integer;
begin
  clrscr;
  writeln('Введите два целых числа');
  read ( a, b );
  if a > b then begin
    max := a;
  end
  else begin
    max := b;
  end;
  writeln ('Наибольшее число ', max);
readkey;
end.
```

Сложные условия

Сложное условие – это условие, состоящее из нескольких простых условий (отношений), связанных с помощью **логических операций**:

- **`not`** – НЕ (отрицание, инверсия)
- **`and`** – И (логическое умножение, конъюнкция, одновременное выполнение условий)
- **`or`** – ИЛИ (логическое сложение, дизъюнкция, выполнение хотя бы одного из условий)
- **`xor`** – исключающее ИЛИ (выполнение только одного из двух условий, но не обоих)

Простые условия (отношения)

`<` `<=` `>` `>=` `=` `<>`

Сложные условия

Порядок выполнения

- выражения в скобках
- **not**
- **and**
- **or, xor**
- **<, <=, >, >=, =, <>**

Особенность – каждое из простых условий обязательно заключать в скобки.

Пример

```
      4      1      6     2       5      3
if not (a > b) or (c <> d) and (b <> a)
then begin
   ...
end
```

Операторы циклы

Цикл – это многократное выполнение одинаковой последовательности действий.

- цикл с **известным** числом шагов
- цикл с **неизвестным** числом шагов (цикл с условием)

Оператор цикла FOR

Увеличение переменной на 1:

```
for <переменная> := <начальное значение> to
    <конечное значение> do begin
  {тело цикла}
end;
```

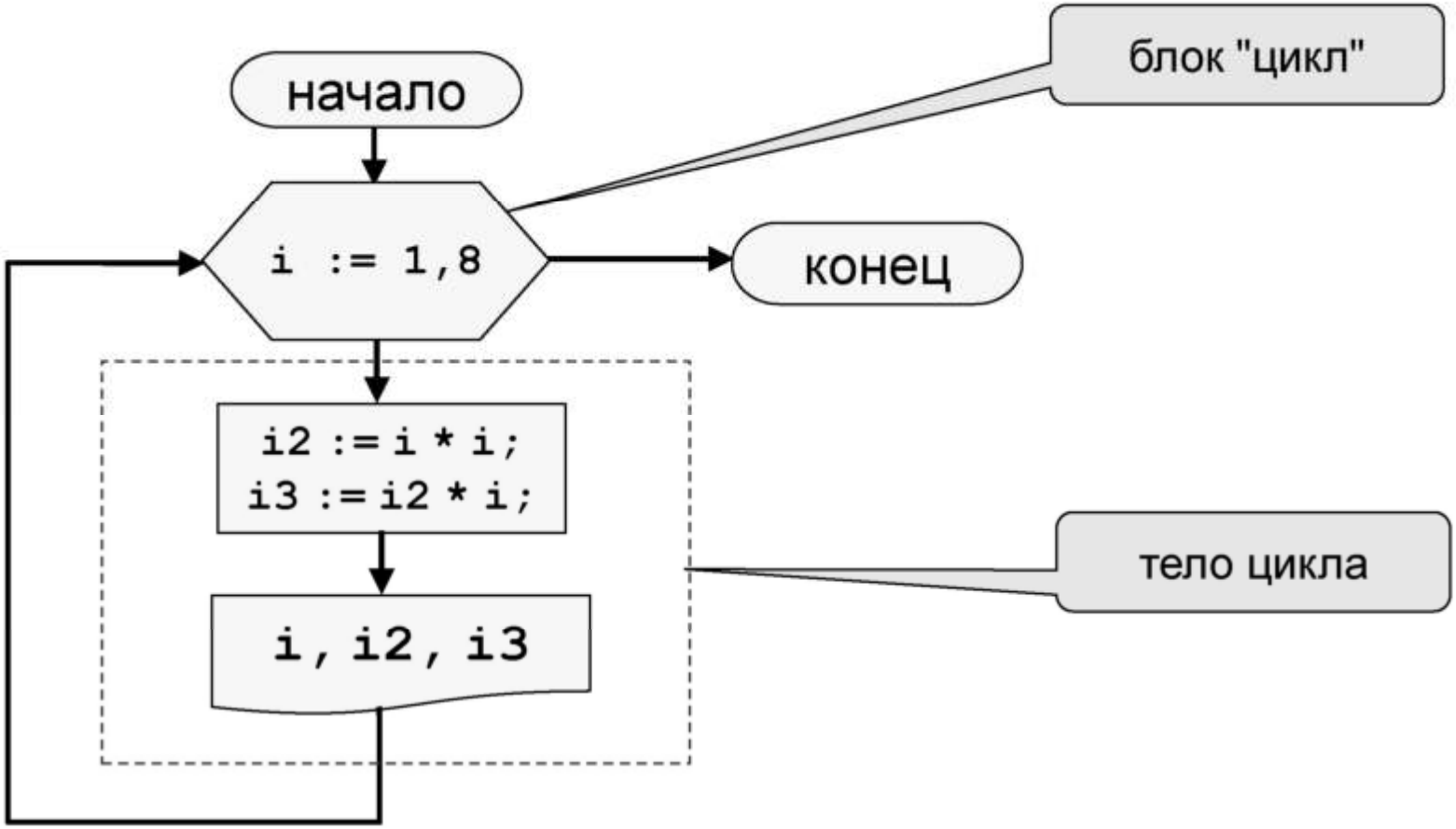
Уменьшение переменной на 1:

```
for <переменная> := <начальное значение>
          downto
    <конечное значение> do begin
  {тело цикла}
end;
```

Оператор цикла FOR

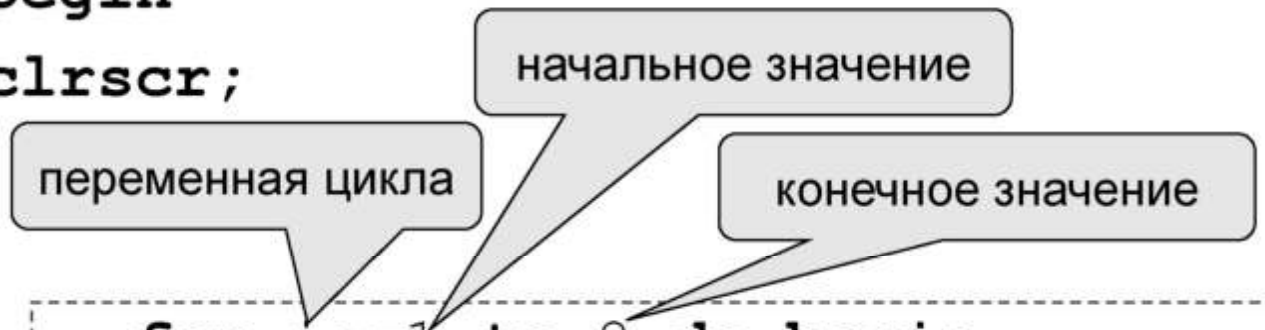
Задача. Вывести на экран квадраты и кубы целых чисел от 1 до 8 (от **a** до **b**).

Особенность: одинаковые действия выполняются 8 раз.



Программа

```
uses crt;
var i, i2, i3: integer;
begin
clrscr;
   for i:=1 to 8 do begin
   i2 := i*i;
   i3 := i2*i;
   writeln(i:4, i2:4, i3:4);
end;
readkey;
end.
```



Цикл FOR с уменьшением переменной

Задача. Вывести на экран квадраты и кубы целых чисел от 8 до 1 (в обратном порядке).

Особенность: переменная цикла должна уменьшаться.

Решение:

```
...
for i:=8 downto 1 do begin
  i2 := i*i;
  i3 := i2*i;
  writeln(i:4, i2:4, i3:4);
end;
...
```

III. Цикл с предусловием.

Может не выполнится не разу.

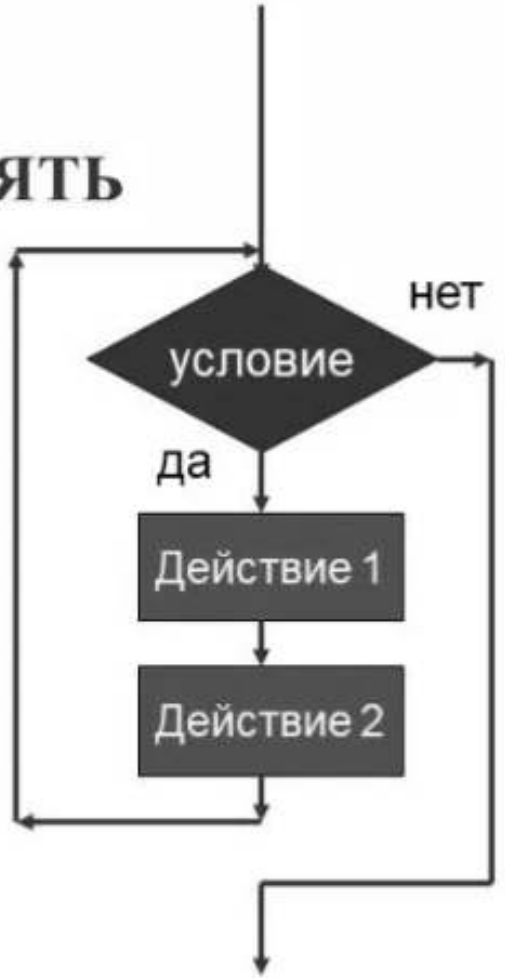
Выполняется пока условие истинно.

WHILE – пока

DO – выполнять

```
ПОКА (УСЛОВИЕ ИСТИННО) ВЫПОЛНЯТЬ
НАЧАЛО
   действие1;
   действие2;
КОНЕЦ;
```

```
WHILE (УСЛОВИЕ ИСТИННО) DO
BEGIN
   действие1;
   действие2;
END;
```



Оператор цикла WHILE

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

Особенности:

- можно использовать сложные условия:

```
while (a<b) and (b<c) do begin
   {тело цикла}
end;
```

- если в теле цикла только один оператор, слова **begin** и **end** можно не писать:

```
while a < b do
   a := a + 1;
```

Цикл с условием

Особенности:

- условие пересчитывается **каждый раз** при входе в цикл
- если условие на входе в цикл ложно, цикл не выполняется ни разу

```
a := 4; b := 6;
while a > b do
   a := a - b;
```

- если условие никогда не станет ложным, программа **зацикливается**

```
a := 4; b := 6;
while a < b do
   d := a + b;
```

Замена for на while и наоборот

```
for i:=1 to 10 do begin
   {тело цикла}
end;
```

```
i := 1;
while i <= 10 do begin
   {тело цикла}
   i := i + 1;
end;
```

```
for i:=a downto b do
   begin
   {тело цикла}
end;
```

```
i := a;
while i >= b do begin
   {тело цикла}
   i := i - 1;
end;
```

Замена цикла for на while возможна всегда.

Замена while на for возможна только тогда, когда можно заранее рассчитать число шагов цикла.

Цикл с постусловием (цикл REPEAT)

Цикл с постусловием – это цикл, в котором проверка условия выполняется в конце цикла.

Особенность: Один раз тело цикла надо сделать в любом случае => проверку условия цикла надо делать в конце цикла (цикл с **постусловием**).

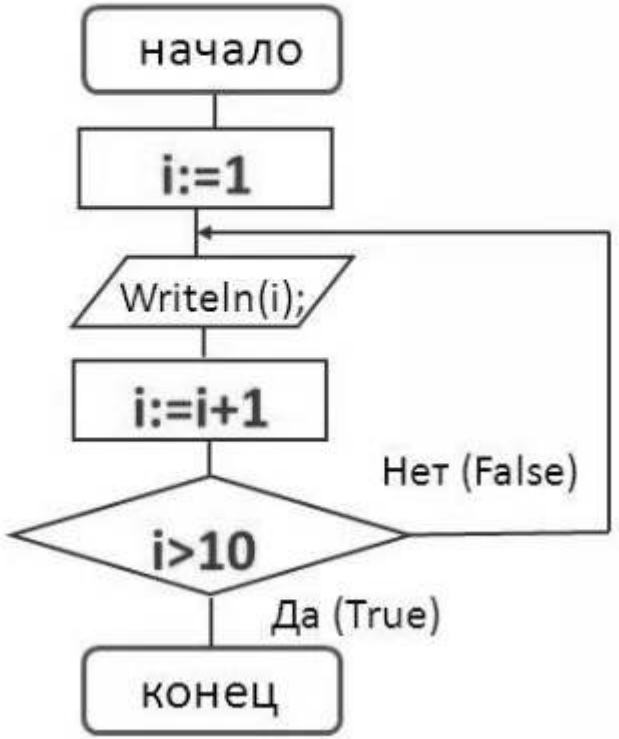
Цикл с постусловием **Repeat … until**

Решение задачи о выводе 10 целых чисел на экран с использованием цикла ***Repeat…until***:

Программа

```
var i: integer; {счетчик}
Begin
 i:  =1;  {начальное значение}

  Repeat
     Writeln(i);
      i:=i+1
   Until i>10
End.
```

Блок-схема алгоритма



Программа

```
program qq;
var n: integer;
begin
  repeat
    writeln('Введите положительное число');
    read(n);
  until n > 0;
  ... { основной алгоритм }
end.
```

условие **ВЫХОДА**

Особенности:

- тело цикла всегда выполняется хотя бы один раз
- после слова **`until`** ("до тех пор, пока не…") ставится условие **ВЫХОДА** из цикла

Оператор выбора CASE

Особенности:

- после **case** может быть имя переменной или арифметическое выражение целого типа (**integer**)

```
case i+3 of
 1: begin a := b; end;
 2: begin a := c; end;
end;
```

или символьного типа (**char**)

```
var c: char;
...
case c of
 'а': writeln('Антилопа');
 'б': writeln('Барсук');
 else writeln('Не знаю');
end;
```

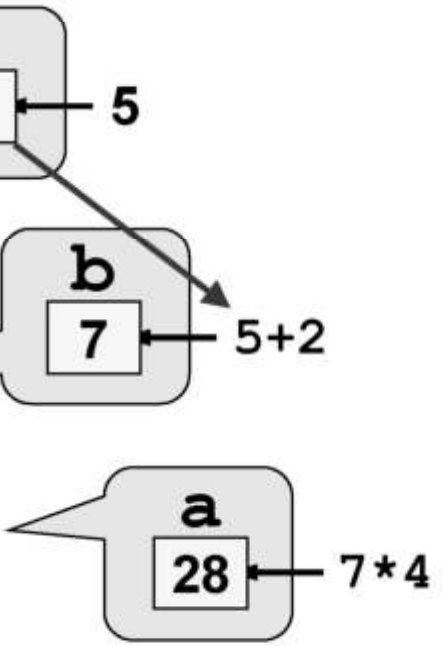
Изменение значений переменной

Оператор – это команда языка программирования высокого уровня.

Оператор присваивания служит для изменения значения переменной.

Пример:

```
program qq;
var a, b: integer;
begin
   a := 5;
   b := a + 2;
   a := (a + 2)*(b - 3);
end.
```



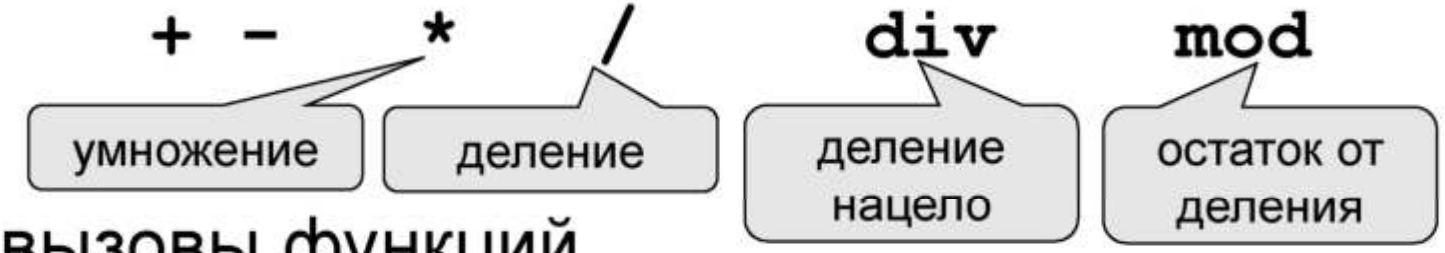
Оператор присваивания

Общая структура:

<имя переменной> **:=** *<выражение>*;

Арифметическое выражение может включать

- константы
- имена переменных
- знаки арифметических операций:



- вызовы функций
- круглые скобки ( )

Какие операторы неправильные?

```
program qq;
var a, b: integer;
    x, y: real;
begin
  a := 5;
  10 := x;
  y := 7,8;
  b := 2.5;
  x := 2*(a + y);
  a := b + x;
end.
```

имя переменной должно быть слева от знака :=

целая и дробная часть отделяются **точкой**

нельзя записывать вещественное значение в целую переменную

Порядок выполнения операций

- вычисление выражений в скобках
- умножение, деление, **div**, **mod** слева направо
- сложение и вычитание слева направо

```
          2  3  5  4   1    7  8   6   9
z := (5*a*c+3*(c-d))/a*(b-c)/ b;
```

$$z = \frac{5ac + 3(c-d)}{ab}(b-c)$$

$$x = \frac{a^2 + 5c^2 - d(a+b)}{(c+d)(d-2a)}$$

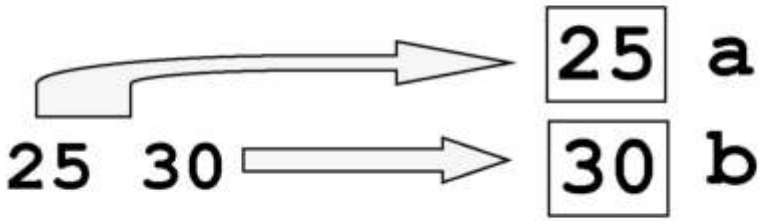
```
          2  6  3  4  7  5   1    12   8  11  10  9
x:=(a*a+5*c*c-d*(a+b))/((c+d)*(d-2*a));
```

Оператор ввода

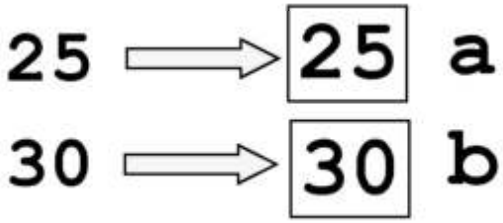
```
read ( a );        { ввод значения
                     переменной a}

read ( a, b );     { ввод значений
                     переменных a и b}
```

Как вводить два числа?

через пробел:



через *Enter*:



Оператор вывода

```
write ( a );     { вывод значения
                   переменной a}

writeln ( a ); { вывод значения
                   переменной a и переход
                   на новую строчку}

writeln ( 'Привет!' ); { вывод текста}

writeln ( 'Ответ: ', c );    { вывод
          текста и значения переменной c}

writeln ( a, '+', b, '=', c );
```

Форматы вывода

```
program qq;
var i: integer;
    x: real;
begin
  i := 15;
  writeln ( '>', i, '<' );
  writeln ( '>', i:5, '<' );
  x := 12.345678;
  writeln ( '>', x, '<' );
  writeln ( '>', x:10, '<' );
  writeln ( '>', x:7:2, '<' );
end.
```

всего символов

всего символов

в дробной части

Оператор выбора

Особенности:

- если нужно выполнить только один оператор, слова **begin** и **end** можно не писать

```
case i+3 of
 1: a := b;
 2: a := c;
end;
```

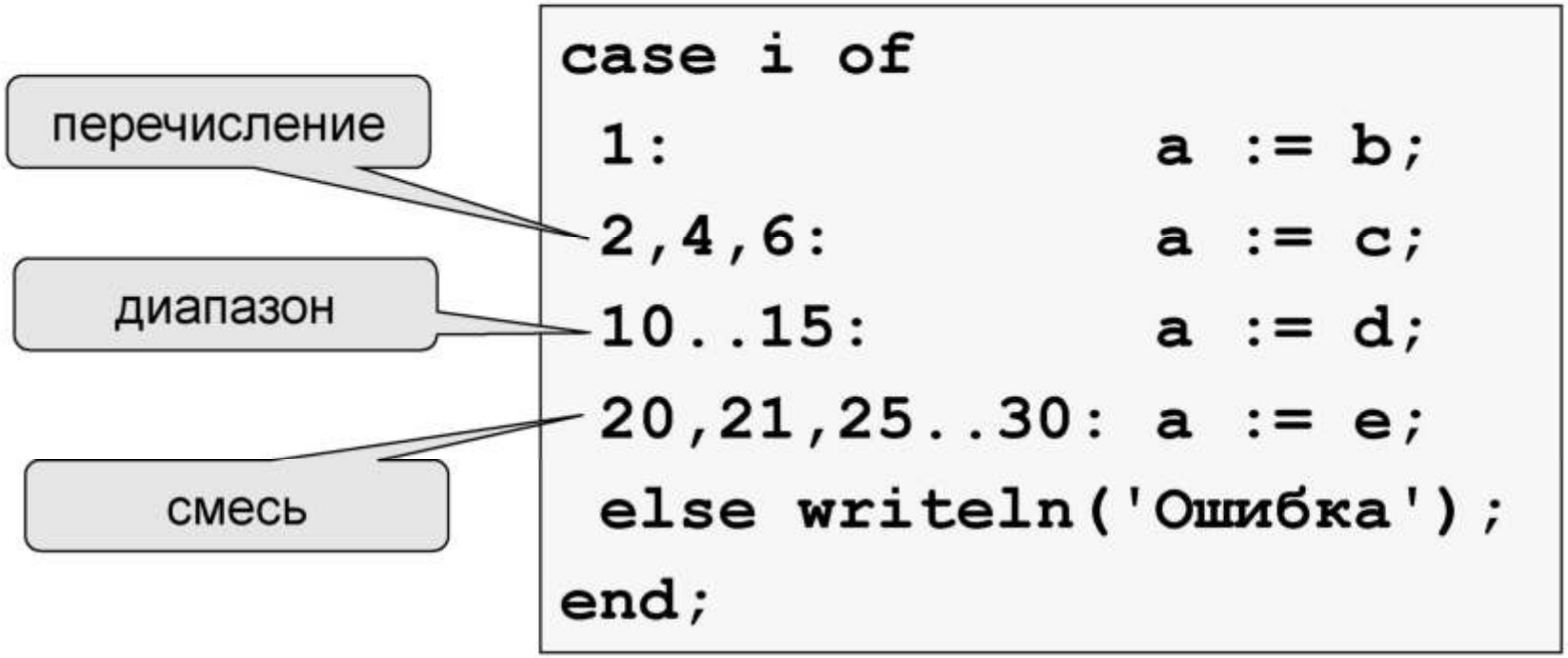
- нельзя ставить два одинаковых значения

```
case i+3 of
 1: a := b;
 1: a := c;
end;
```

Оператор выбора

Особенности:

- значения, при которых выполняются одинаковые действия, можно группировать

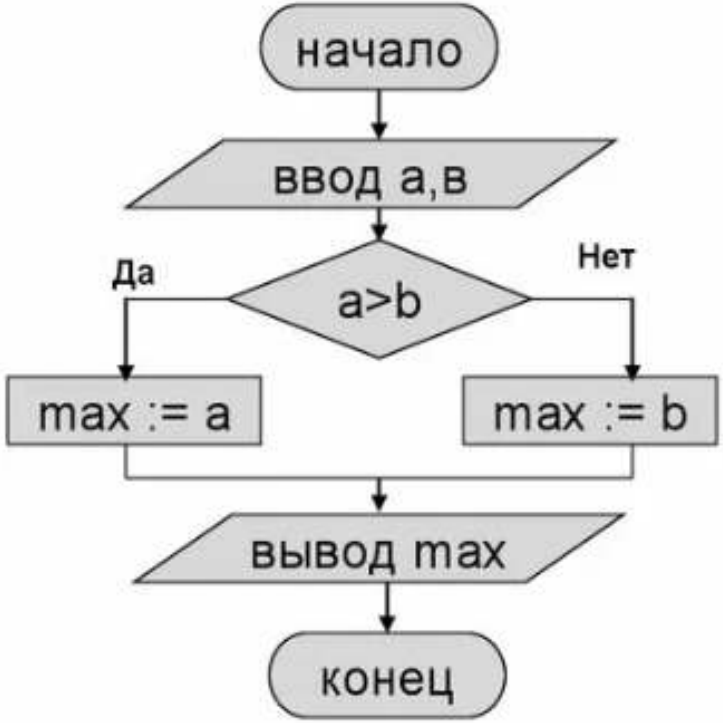


Программа

```
uses crt;
var M, D: integer;
begin
    clrscr;
    writeln('Введите номер месяца:');
    read ( M );
    case M of
     2:          begin D := 28; end;
     4,6,9,11: begin D := 30; end;
     1,3,5,7,8,10,12: D := 31;
     else                D := -1;
    end;
    if D > 0 then
       writeln('В этом месяце ', D, ' дней.')
    else
       writeln('Неверный номер месяца');
readkey;
end.
```

ни один вариант не подошел

Ветвление в Паскале




```
Program   Bid1;
Var
          a,b,max : real;
Begin
          write('a='); readln(a);
          write('b='); readln(b);
          If  a > b  then
              max := a
          else
              max := b;
          writeln('max=',max:7:2)

end.
```

Вопрос 4

Процедуры и функции

Числовые типы данных

Стандартные функции языка Паскаль:

Функция	Назначение	Тип аргумента	Тип результата
abs (x)	Модуль x	integer, real	Такой же, как у аргумента
sqr (x)	Квадрат x	integer, real	Такой же, как у аргумента
sqrt (x)	Квадратный корень из x	integer, real	real
round (x)	Округление x до ближайшего целого	real	integer
int (x)	Целая часть x	real	integer
frac (x)	Дробная часть x	real	real
random	Случайное число от 0 до 1	-	real

Процедуры

Процедура – это вспомогательный алгоритм, который предназначен для выполнения какой-то законченной последовательности действий.

- Для исполнения подпрограммы процедуры необходимо сначала описать ее, а потом к ней обращаться
- Описание процедуры включает заголовок (имя) и тело процедуры
- Заголовок состоит из зарезервированного слова procedure, имени процедуры и заключенных в скобки списка формальных параметров с указанием типа

Процедуры

Особенности:

- в процедуре можно объявлять дополнительные **локальные** переменные, остальные процедуры не имеют к ним доступа

```
program qq;
  procedure A(x, y: integer);
  var a, b: real;
  begin
    a := (x + y)/6;
    ...
  end;
begin
  ...
end.
```

локальные переменные

Процедуры

Задача: найти наибольшее из 4-х чисел, используя подпрограмму нахождения наибольшего из 2-х чисел

```
program max;
uses crt;
var a,b,c,d,p,q,m: integer;
  procedure bid(x,y: real; var z: real);
  begin
     if x>y then z:=x else z:=y
  end;
begin
clrscr;
    write('введите 4 числа:');
    readln (a,b,c,d);
    bid (a,b,p);
    bid (c,d,q);
    bid (p,q,m);
    writeln('наибольшее из 4-х чисел'; m);
    readkey;
end.
```

Процедура с параметрами

```
program binCode;
  procedure printBin(n: integer);
  var k: integer;
  begin
    k:=128;
    while k>0 do begin
      write(n div k);
      n:=n mod k;
      k:=k div 2
    end
  end;
begin
  printBin(99)
end.
```

локальная переменная

Параметры – данные, изменяющие работу процедуры.

значение параметра (**аргумент**)

Сложение чисел: полное решение

```
program Sum;
var a, b, c: integer;
begin
  writeln('Введите два целых числа');
  read ( a, b );
  c := a + b;
  writeln ( a, '+', b, '=', c );
end.
```

Протокол:

компьютер

```
Введите два целых числа
25 30
25+30=55
```

пользователь

Функции

Функция – это вспомогательный алгоритм (подпрограмма), результатом работы которого является некоторое значение.

Примеры:

- **вычисление** $\sin x$, $\cos x$, $\sqrt{x}$
- **расчет значений по сложным формулам**
- **ответ на вопрос (простое число или нет?)**

Отличия

- **в заголовке**
- **в теле функции: хотя бы раз имени функции должно быть присвоено значение**

Функции

Особенности:

- заголовок начинается словом **function**

```
function Max (a, b: integer): integer;
```

- формальные параметры описываются так же, как и для процедур

```
function qq( a, b: integer; x: real ): real;
```

- в конце заголовка через двоеточие указывается **тип результата**

```
function Max (a, b: integer): integer;
```

- функции располагаются **ВЫШЕ** основной программы

Функции

Особенности:

- можно объявлять и использовать **локальные переменные**

```
function qq (a, b: integer): float;
var x, y: float;
begin
   ...
end;
```

- значение, которое является результатом, записывается в переменную, **имя** которой совпадает с **названием функции**; объявлять ее **НЕ НАДО**:

```
function Max (a, b: integer): integer;
begin
   ...
   Max := a;
end;
```

СТАНДАРТНЫЕ ФУНКЦИИ ЯЗЫКА ПАСКАЛЬ

Функция	Назначение	Пример	Результат
Sqr	возведение в квадрат	x:=sqr(6)	36
Sqrt	извлечение корня квадр.	y:=sqrt(81)	9
Random(K)	Выбор случайного числа от 0 до K	x:=Random(10)	0..10
Round	Округление до ближайшего целого числа	y:=round(7.9) x:=round(-2.1)	8 -2

Возведение в произвольную степень $y=x^a$ **y:=exp(a*ln(x))**

Вычисление логарифма $y=\log_a x$ **y:=ln(x)/ln(a)**