

Программирование на языке Паскаль

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

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**).

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, %, &)

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, longint { вещественная }
- char { один символ }
- string { символьная строка }
- boolean { логическая }

Объявление переменных (выделение памяти):

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

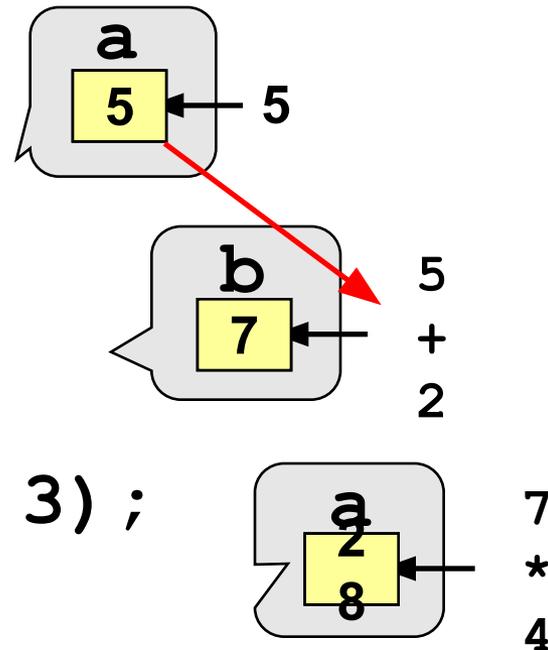
Изменение значений переменной

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:

+ - * / div mod

умножение

деление

деление
нацело

остаток от
деления

- 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

Арифметические функции

Функции	Назначение	Тип результата
Abs	абсолютное значение аргумента	совпадает с типом X
Arctan (X)	арктангенс аргумента	веществ.
Cos (X)	косинус аргумента	веществ.
Exp (X)	e	веществ.
Frac (X)	дробная часть числа	веществ.
Ln (X)	натуральный логарифм	веществ.
Pi (X)	значение величины $\pi = 3,14159265358979932385$	веществ.
Sin (x)	синус аргумента	веществ.
Sqr (X)	квадрат аргумента	совпадает с типом X
Sqrt (X)	квадратный корень аргумента	веществ.

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 вывода

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

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

`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

Задания для самостоятельной работы

1. Составить программу, переводящую введенные с клавиатуры мили в километры(1 миля =1,852 км)

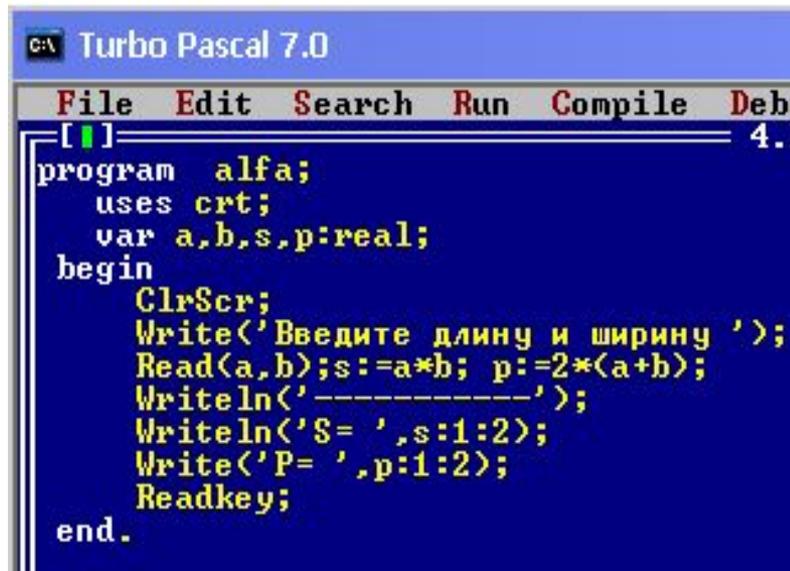
```
1.PAS
program alfa;
uses crt;
var a,b:real;
begin
  ClrScr;
  Write('Введите мили ');Read(a);
  b:=a*1.852;
  Write(a:1:0,' миль= ',b:1:3,' км');_
  Readkey;
end.
```

2. Составить программу, запрашивающую 2 числа и выдающую их сумму.

```
2.P
program alfa;
uses crt;
var a,b,c:integer;
begin
  ClrScr;
  Write('Введите 2 числа ');Read(a,b);
  c:=a+b;
  Write(a,' + ',b,' = ',c);
  Readkey;
end.
```

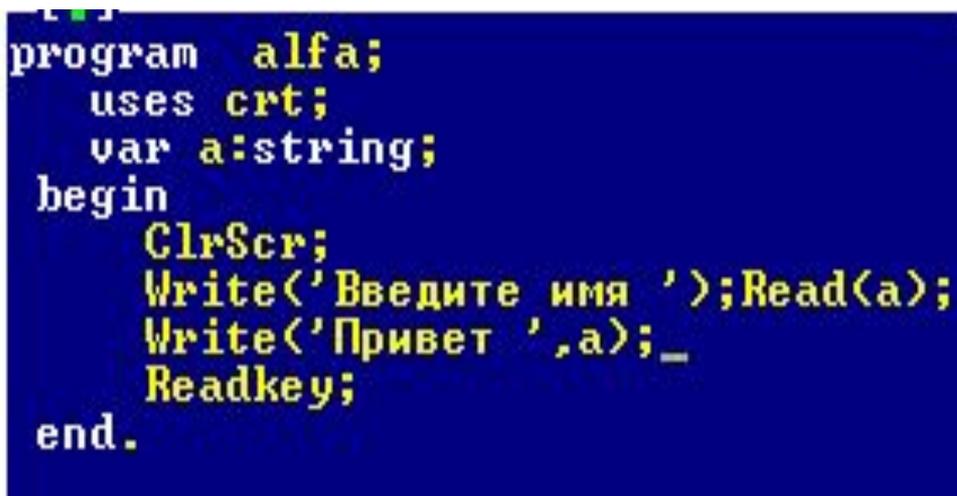
Задания для самостоятельной работы

3. Вычислить периметр и площадь прямоугольника, по введенным с клавиатуры сторонам.



```
 Turbo Pascal 7.0
File Edit Search Run Compile Debu
[ ] 4.1
program alfa;
uses crt;
var a,b,s,p:real;
begin
  ClrScr;
  Write('Введите длину и ширину ');
  Read(a,b);s:=a*b; p:=2*(a+b);
  Writeln('-----');
  Writeln('S= ',s:1:2);
  Write('P= ',p:1:2);
  Readkey;
end.
```

4. Составить программу, запрашивающую имя и приветствующую по этому имени.



```
 Turbo Pascal 7.0
File Edit Search Run Compile Debu
[ ] 4.1
program alfa;
uses crt;
var a:string;
begin
  ClrScr;
  Write('Введите имя ');Read(a);
  Write('Привет ',a);_
  Readkey;
end.
```

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:

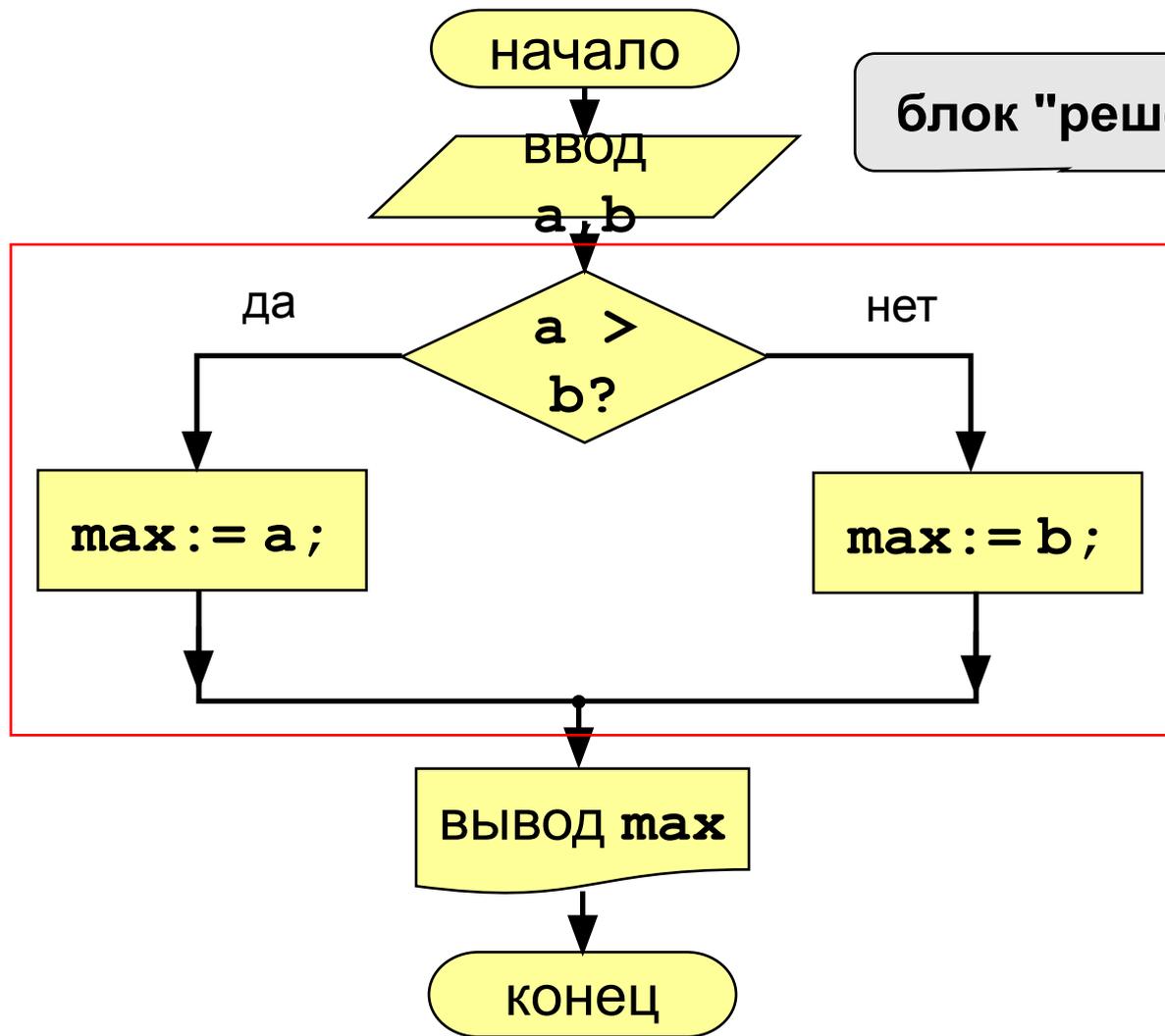
- перед ***else*** **НЕ** ставится точка с запятой
- вторая часть (***else*** ...) может отсутствовать (неполная форма)
- если в блоке один оператор, можно убрать слова ***begin*** и ***end***

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.

Блок-схема



блок "решение"

полная форма
ветвления

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.
```

полная форма
условного
оператора

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)

<

<=

>

>=

=

<>

равно

не равно

Сложные условия

Порядок выполнения

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

Особенность – каждое из простых условий обязательно заключать в скобки.

Пример

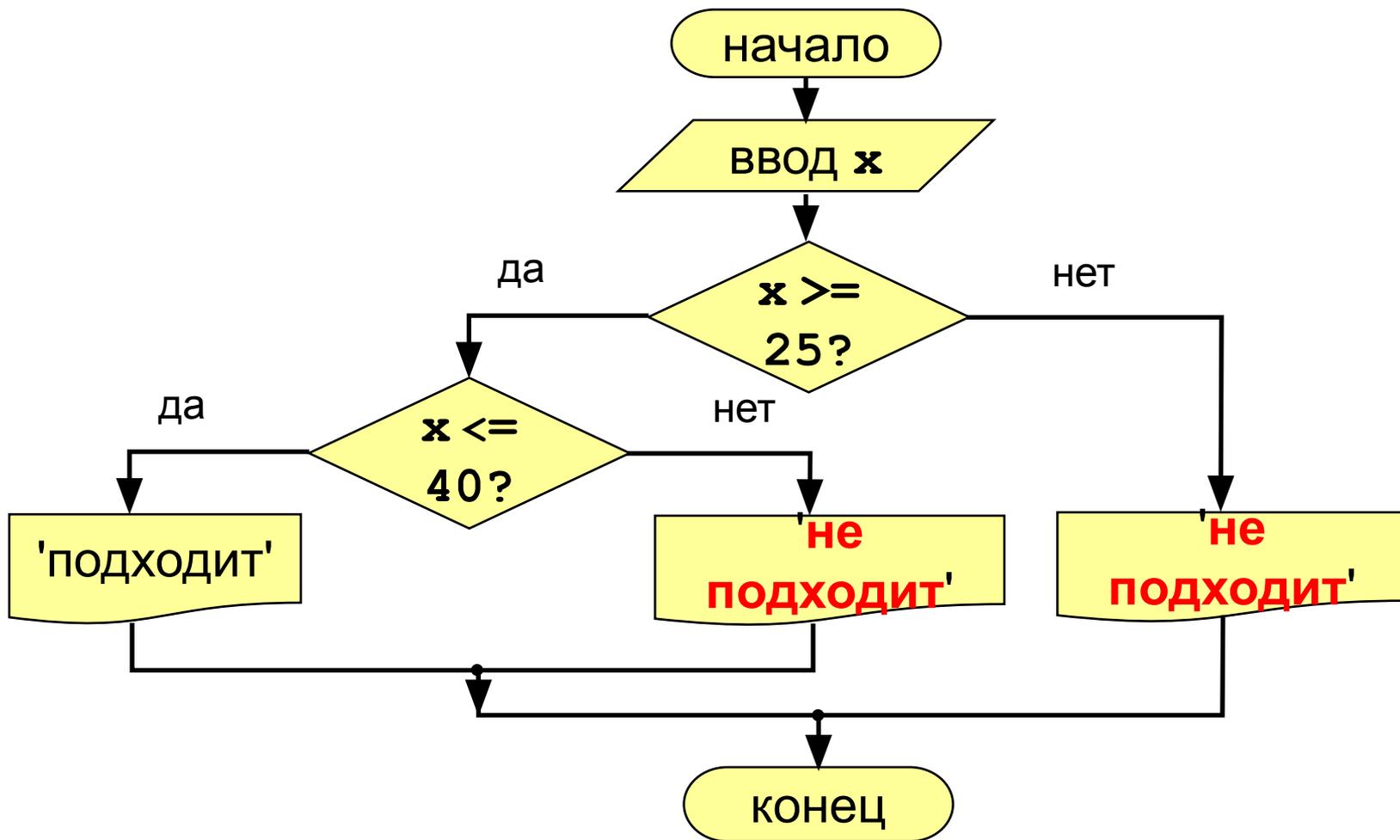
```
      4      1      6      2      5      3  
if not (a > b) or (c <> d) and (b <> a)  
then begin  
    . . .  
end
```

Сложные условия

Задача. Фирма набирает сотрудников от 25 до 40 лет включительно. Ввести возраст человека и определить, подходит ли он фирме (вывести ответ "подходит" или "не подходит").

Особенность: надо проверить, выполняются ли два условия одновременно.

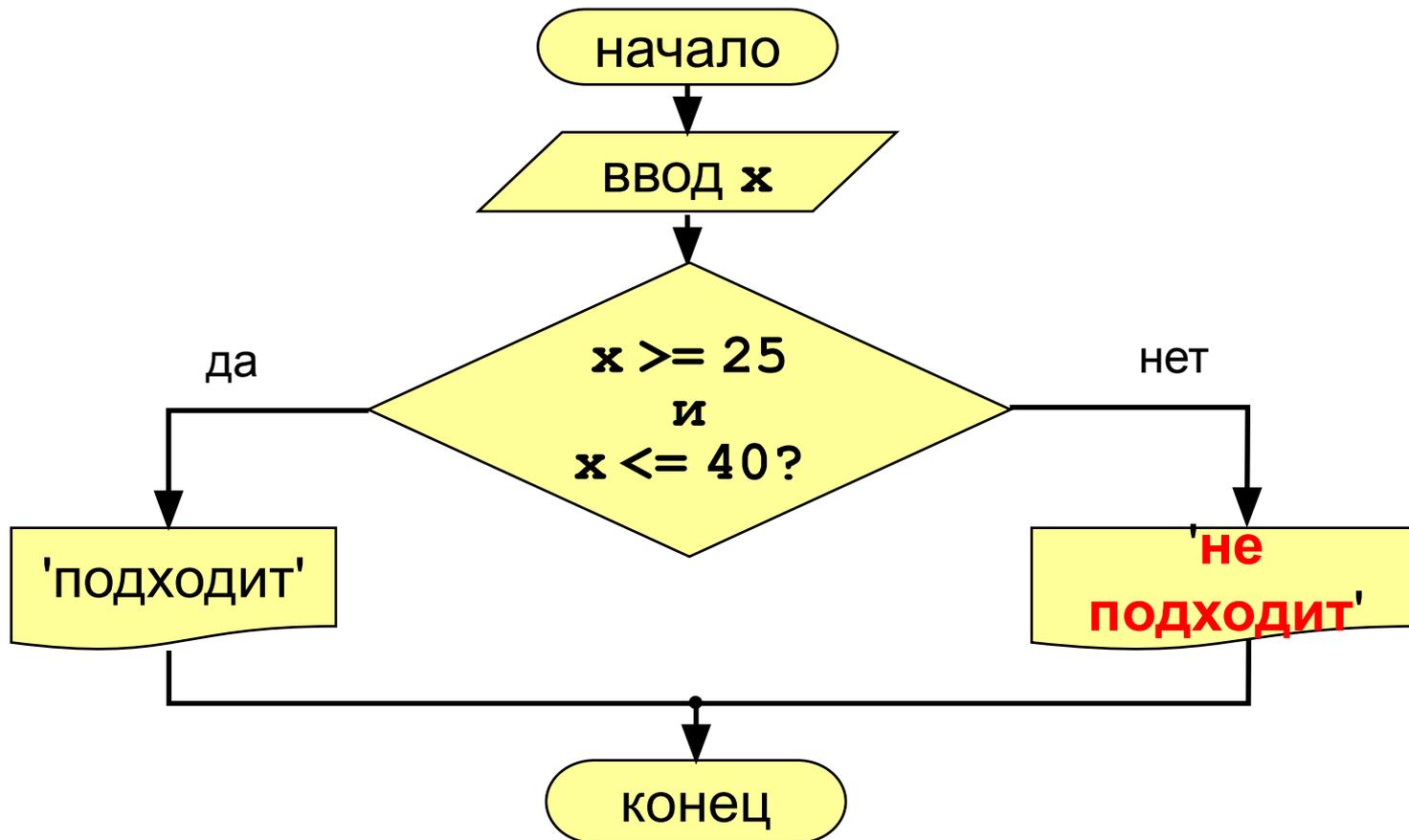
Вариант 1. Алгоритм



Вариант 1. Программа

```
uses crt;
var x: integer;
begin
  clrscr;
  writeln('Введите возраст');
  read ( x );
  if x >= 25 then
    if x <= 40 then
      writeln ('Подходит')
    else writeln ('Не подходит')
  else
    writeln ('Не подходит');
  readkey;
end.
```

Вариант 2. Алгоритм



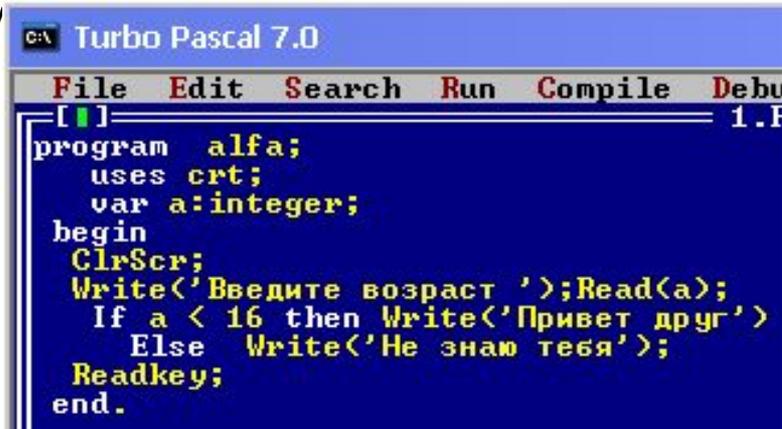
Вариант 2. Программа

```
uses crt;  
var x: integer;  
begin  
clrscr;  
  writeln('Введите возраст');  
  read ( x );  
  if (x >= 25) and (x <= 40) then  
    writeln ('Подходит')  
  else writeln ('Не подходит')  
readkey;  
end.
```

сложное
условие

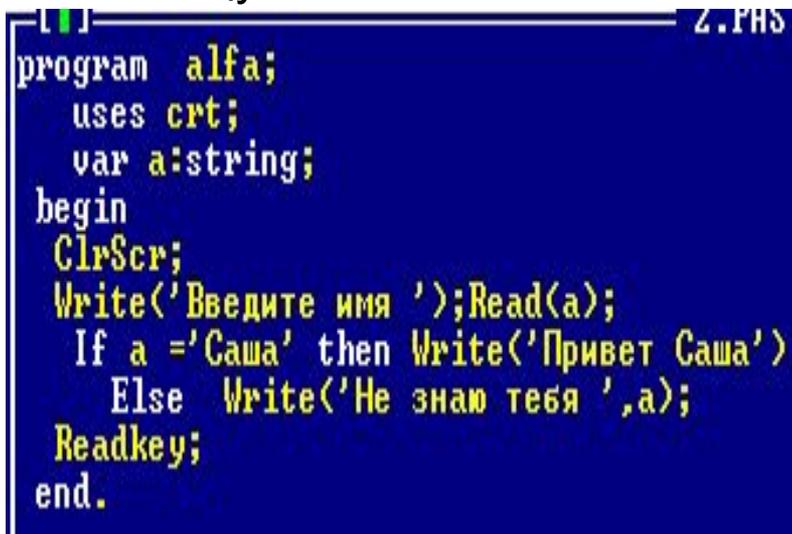
Задания для самостоятельной работы

1. Составить программу, запрашивающую возраст и отвечающую «Привет друг» если возраст введен меньше 16 лет и отвечающую «Не знаю тебя» в противном



```
c:\ Turbo Pascal 7.0
File Edit Search Run Compile Debu
[ ] 1.F
program alfa;
  uses crt;
  var a:integer;
begin
  ClrScr;
  Write('Введите возраст ');Read(a);
  If a < 16 then Write('Привет друг')
  Else Write('Не знаю тебя');
  Readkey;
end.
```

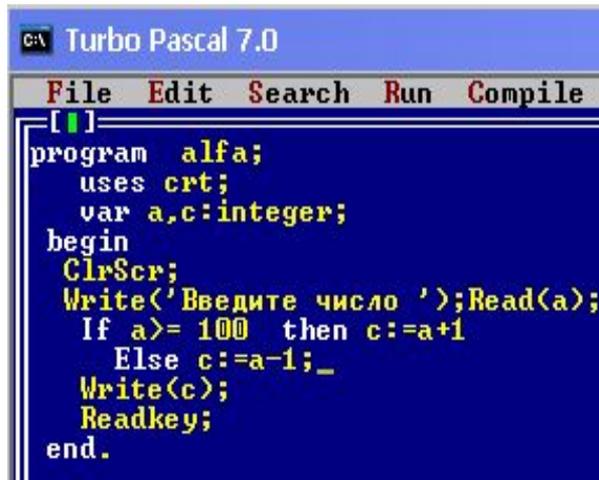
2. Составить программу - отзывающуюся только на имя «Саша»



```
[ ] 2.PAS
program alfa;
  uses crt;
  var a:string;
begin
  ClrScr;
  Write('Введите имя ');Read(a);
  If a ='Саша' then Write('Привет Саша')
  Else Write('Не знаю тебя ',a);
  Readkey;
end.
```

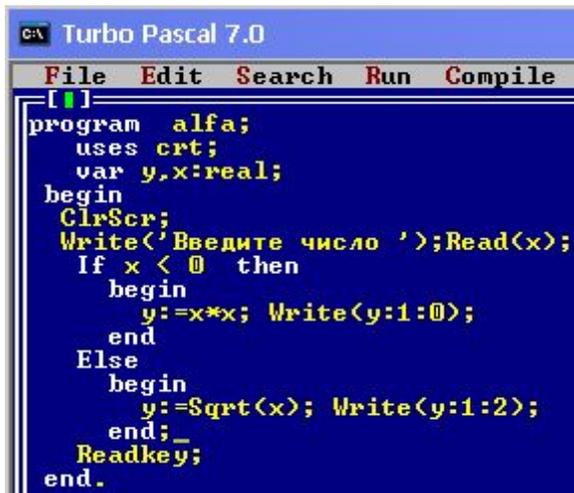
Задания для самостоятельной работы

3. Составьте программу, увеличивающую введенное число на 1, если оно больше или равно 100 и уменьшающее число на 1 в противном случае.



```
 Turbo Pascal 7.0
File Edit Search Run Compile
[ ]
program alfa;
uses crt;
var a,c:integer;
begin
  ClrScr;
  Write('Введите число ');Read(a);
  If a>= 100 then c:=a+1
  Else c:=a-1;_
  Write(c);
  Readkey;
end.
```

4. Составить программу, вычисляющую функцию: $y = \begin{cases} x^2, & \text{если } x < 0 \\ \sqrt{x}, & \text{иначе} \end{cases}$



```
 Turbo Pascal 7.0
File Edit Search Run Compile
[ ]
program alfa;
uses crt;
var y,x:real;
begin
  ClrScr;
  Write('Введите число ');Read(x);
  If x < 0 then
  begin
    y:=x*x; Write(y:1:0);
  end
  Else
  begin
    y:=sqrt(x); Write(y:1:2);
  end;_
  Readkey;
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;
```

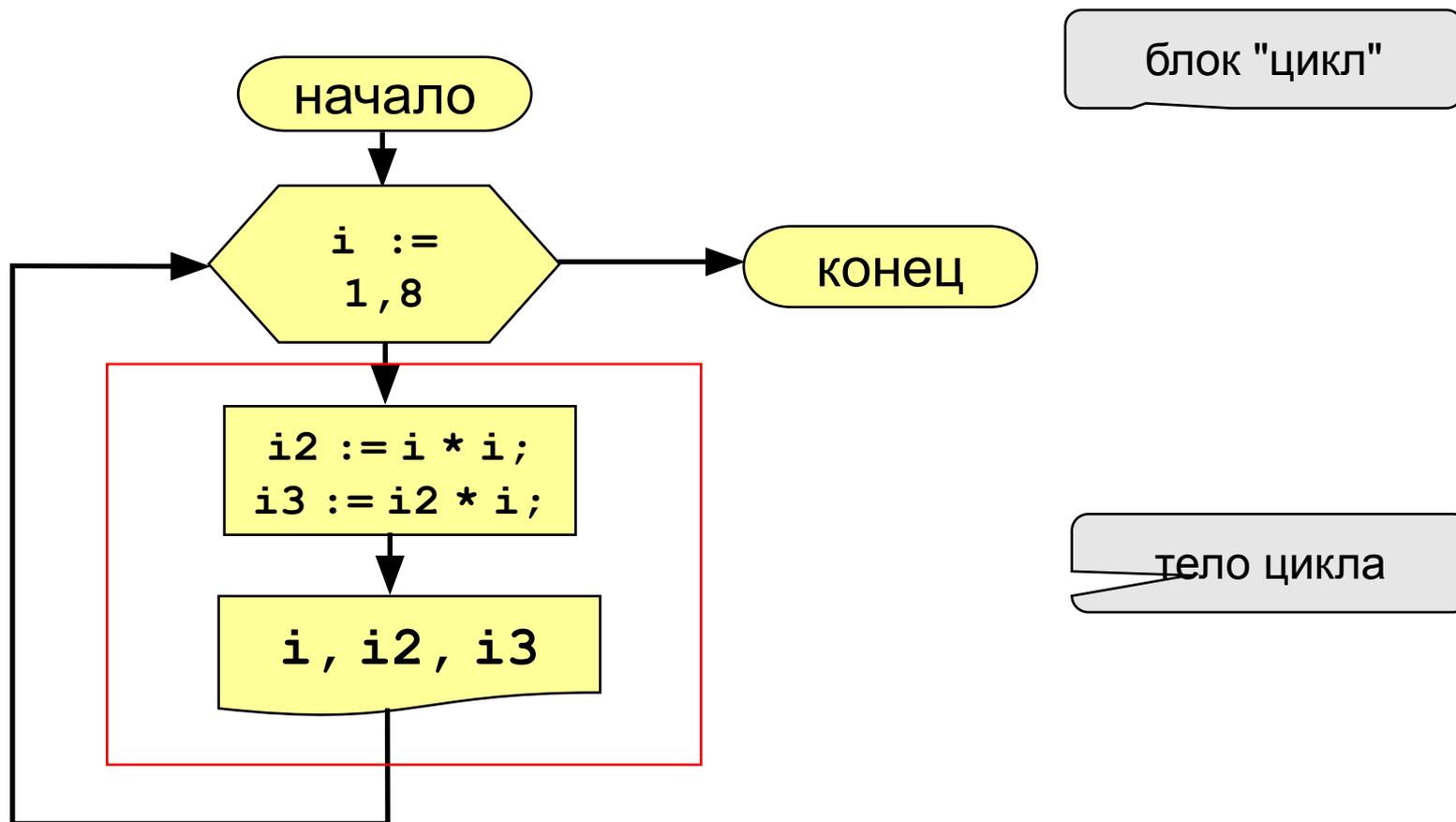
Уменьшение переменной на 1:

```
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;
```

переменная цикла

начальное значение

конечное значение

```
    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;  
...
```

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;
```

Сколько раз выполняется цикл?

```
a := 4; b := 6;  
while a < b do a := a + 1;
```

2 раза

a = 6

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

1 раз

a = 10

```
a := 4; b := 6;  
while a > b do a := a + 1;
```

0 раз

a = 4

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

1 раз

b = -2

```
a := 4; b := 6;  
while a < b do a := a - 1;
```

зацикливание

Цикл WHILE

Задача: Ввести целое число (<2000000) и определить число цифр в нем.

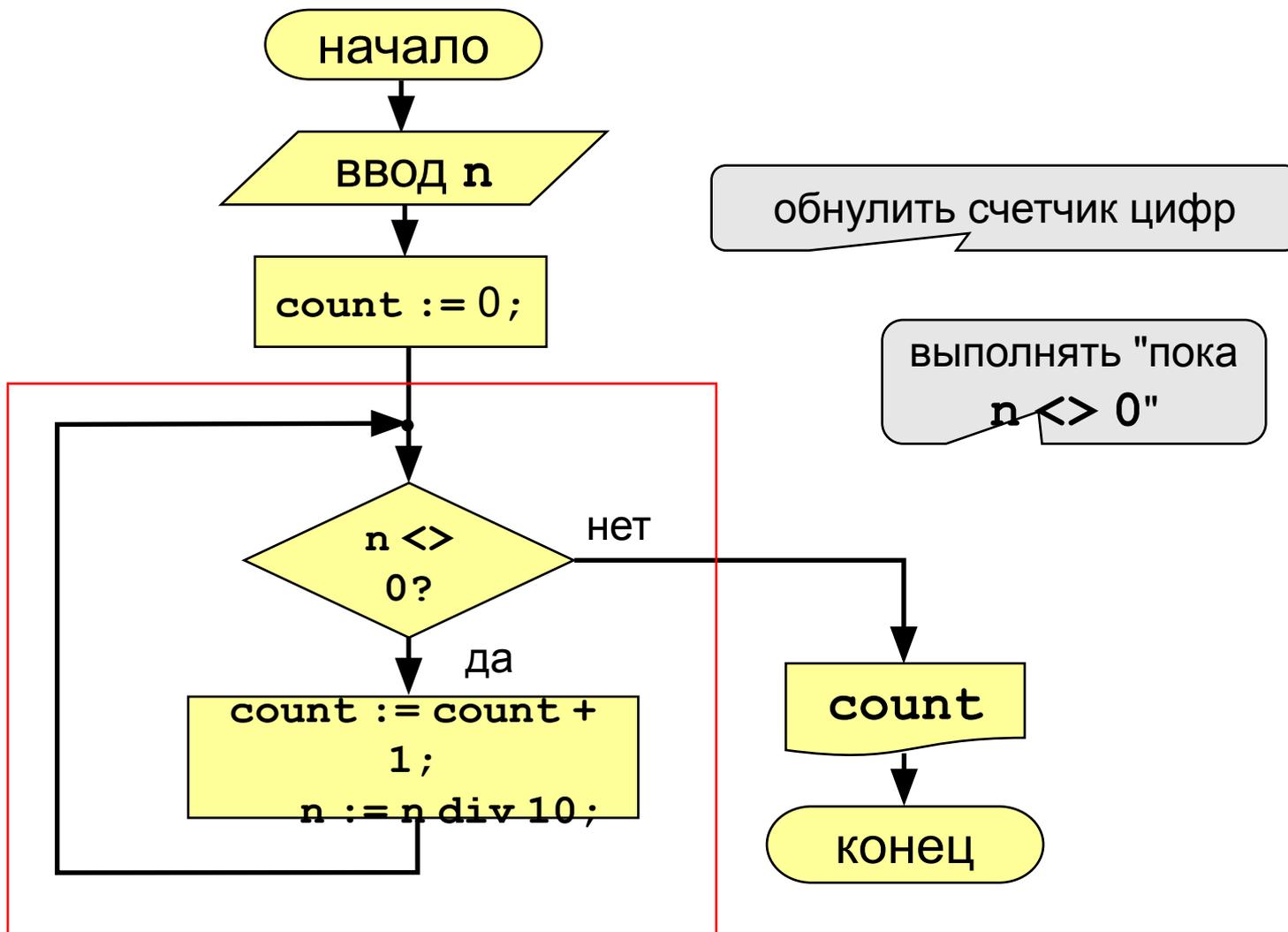
Идея решения: Отсекаем последовательно последнюю цифру, увеличиваем счетчик.

n	count
123	0
12	1
1	2
0	3

Проблема: Неизвестно, сколько шагов надо сделать.

Решение: Надо остановиться, когда $n = 0$, т.е. надо делать "пока $n \neq 0$ ".

Алгоритм



Программа

```
uses crt;
var n, count: , n1: integer;
Begin
  clrscr;
  writeln('Введите целое число');
  read(n); n1 := n;
  count := 0;
  while n <> 0 do begin
    count := count + 1;
    n := n div 10;
  end;
  writeln('В числе ', n1, ' нашли ',
    count, ' цифр');
  readkey;
end.
```

выполнять "пока
n <> 0"



Какая ошибка?

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;
```

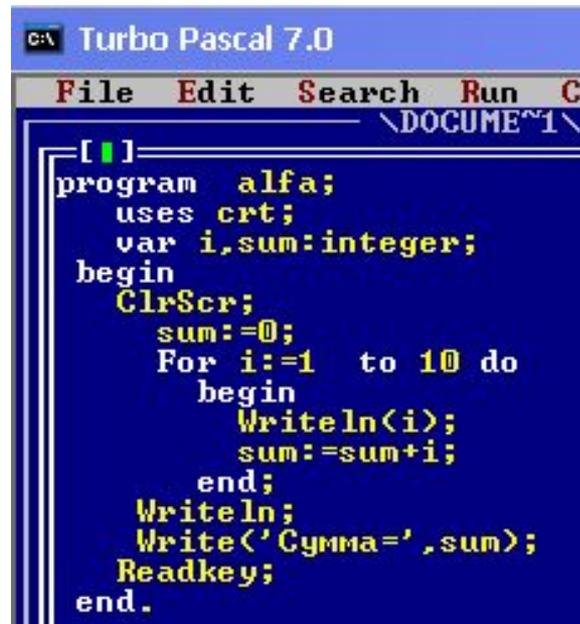
```
i := a;
while i >= b do begin
  {тело цикла}
  i := i - 1;
end;
```

Замена цикла **for** на **while** возможна **всегда**.

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

Задания для самостоятельной работы

1. Найти сумму чисел от 1 до 10



```

C:\ Turbo Pascal 7.0
File Edit Search Run C
\DOCUME~1\
[ ]
program alfa;
uses crt;
var i,sum:integer;
begin
  ClrScr;
  sum:=0;
  For i:=1 to 10 do
  begin
    Writeln(i);
    sum:=sum+i;
  end;
  Writeln;
  Write('Сумма=',sum);
  Readkey;
end.

```

2. Напечатать таблицу умножения на

введенное с клавиатуры число, например ввели 7

7x1=7

7x2=14

.....

7x9=63



```

C:\ Turbo Pascal 7.0
File Edit Search Run Compile Debug
\DOCUME~1\BA22~1\0016~1\A
7.PAS
[ ]
program alfa;
uses crt;
var a,i: integer;
begin
  ClrScr;
  Write('Введите число ');Read(a);
  For i:=1 to 9 do
    Writeln(a,' x ',i,' = ',a*i);
  Readkey;
end.

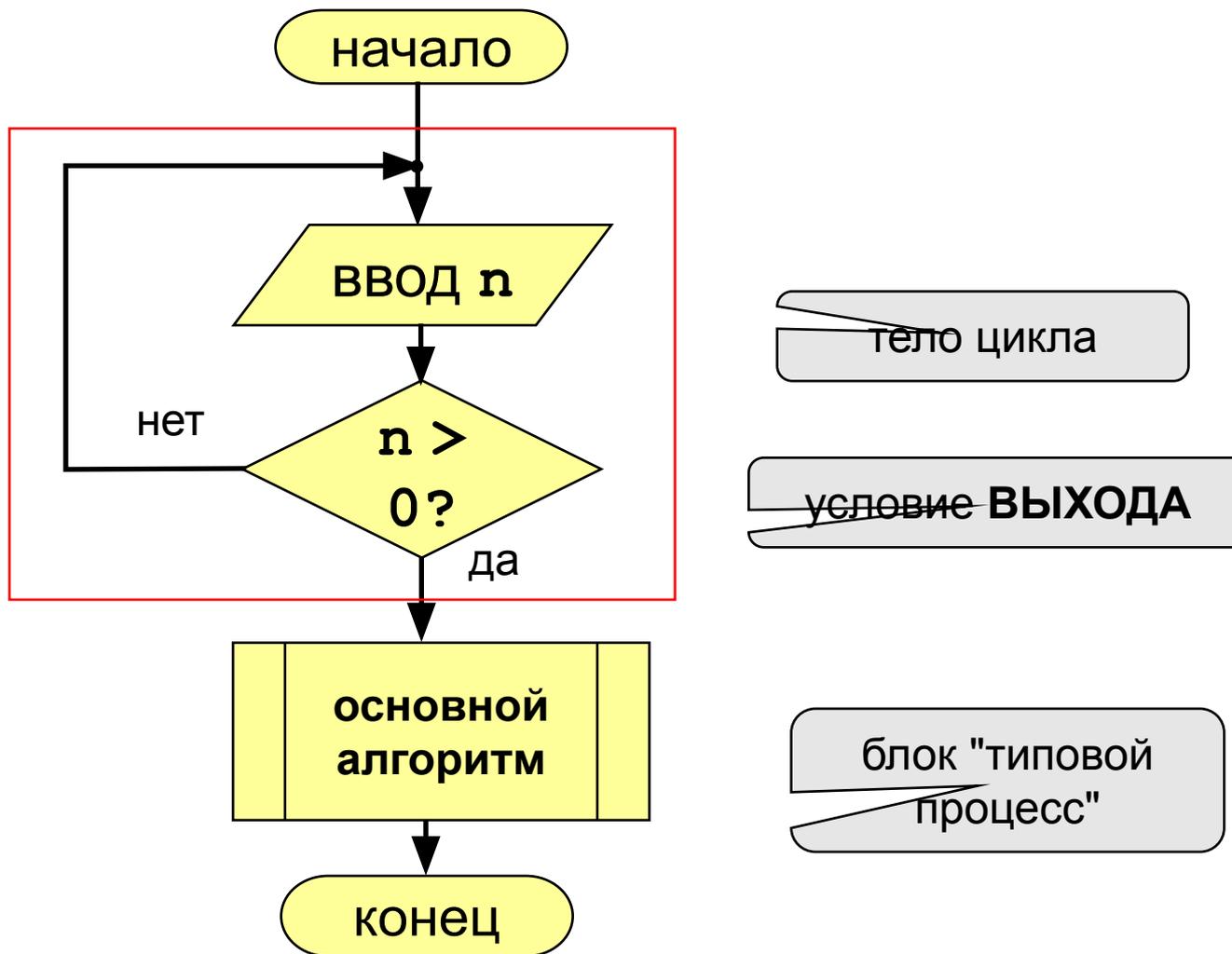
```

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: алгоритм



Программа

```
program qq;  
var n: integer;  
begin
```

```
  repeat  
    writeln('Введите положительное число');  
    read(n);
```

```
  until n > 0;
```

условие **ВЫХОДА**

```
  ... { основной алгоритм }
```

```
end.
```

Особенности:

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

Задания для самостоятельной работы

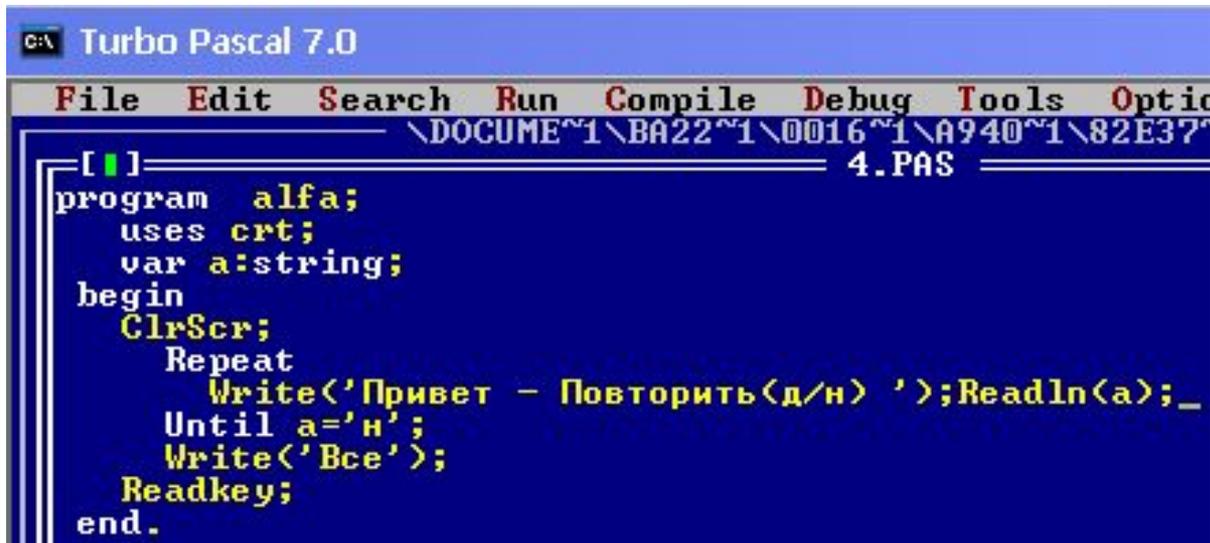
1. Составить программу, запрашивающую числа до тех пор, пока не ввели числа большего 50.



```

Turbo Pascal 7.0
File Edit Search Run Compile Debug
\DOCUME~1\BA22~1\0016~1\A
[ ] 3.PAS
program alfa;
uses crt;
var a:integer;
begin
  ClrScr;
  Repeat
    Write('Введите число ');Read(a);
  Until a>50;
  Write('Все');
  Readkey;
end.
```

2. Составить программу, печатающую слово «Привет» и запрашивающую «Повторить Д/Н?». Программа завершает выполнение в случае нажатия Н, если Д, то повторяет.



```

Turbo Pascal 7.0
File Edit Search Run Compile Debug Tools Optic
\DOCUME~1\BA22~1\0016~1\A940~1\82E37
[ ] 4.PAS
program alfa;
uses crt;
var a:string;
begin
  ClrScr;
  Repeat
    Write('Привет - Повторить(д/н) ');Readln(a);_
  Until a='н';
  Write('Все');
  Readkey;
end.
```

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
  'a': writeln('Антилопа');
  'б': writeln('Барсук');
  else writeln('Не знаю');
end;
```

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

перечисление

диапазон

смесь

```
case i of
  1:           a := b;
  2, 4, 6:    a := c;
  10..15:     a := d;
  20, 21, 25..30: a := e;
  else writeln('Ошибка');
end;
```

Оператор выбора CASE

Задача: Ввести номер месяца и вывести количество дней в этом месяце.

Решение: Число дней по месяцам:

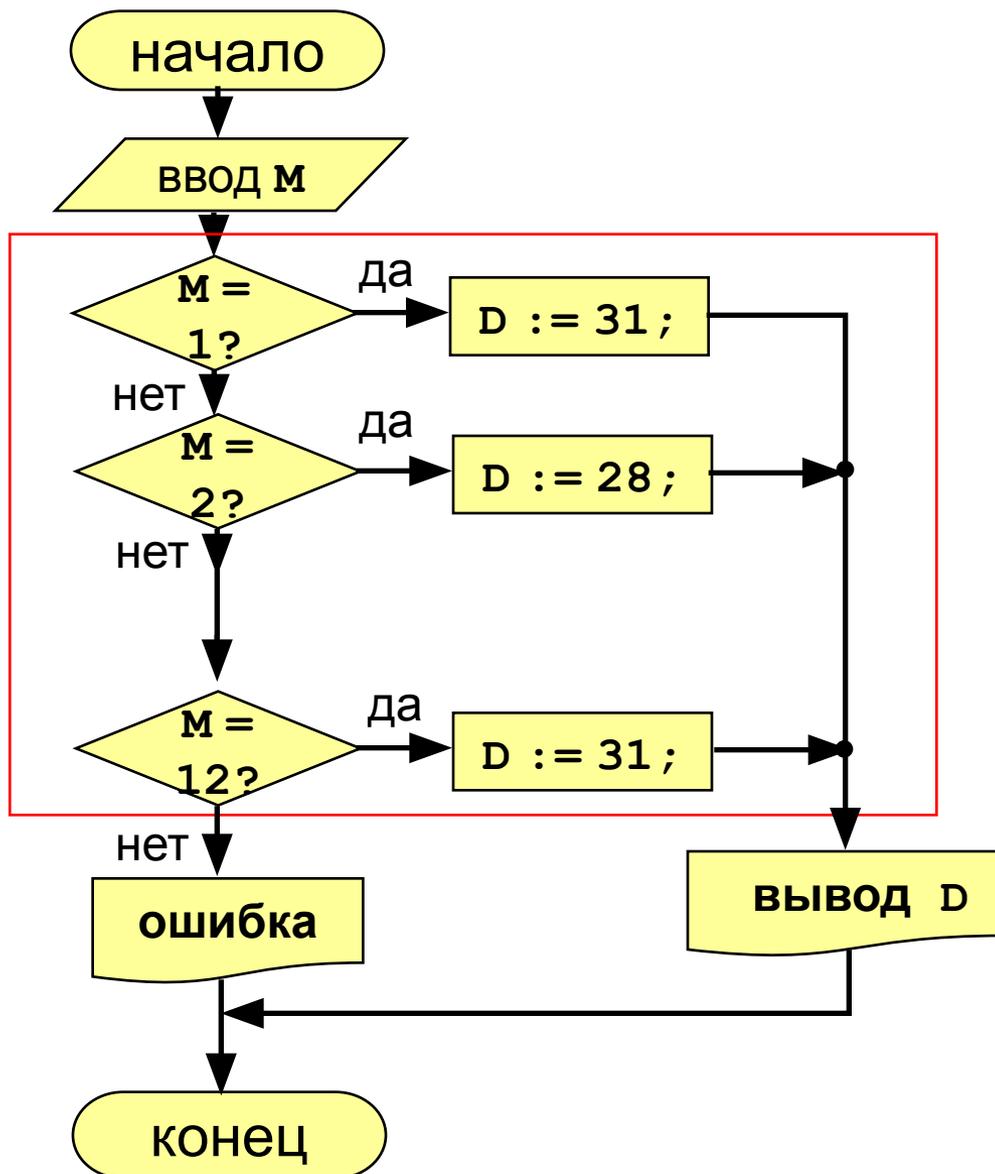
28 дней – 2 (февраль)

30 дней – 4 (апрель), 6 (июнь), 9 (сентябрь), 11 (ноябрь)

31 день – 1 (январь), 3 (март), 5 (май), 7 (июль),
8 (август), 10 (октябрь), 12 (декабрь)

Особенность: Выбор не из двух, а из нескольких вариантов в зависимости от номера месяца.

Алгоритм



оператор выбора

ни один вариант не подошел

Программа

```
uses crt;
var M, D: integer;
begin
  clrscr;
  writeln('Введите номер месяца: ');
  read ( M );

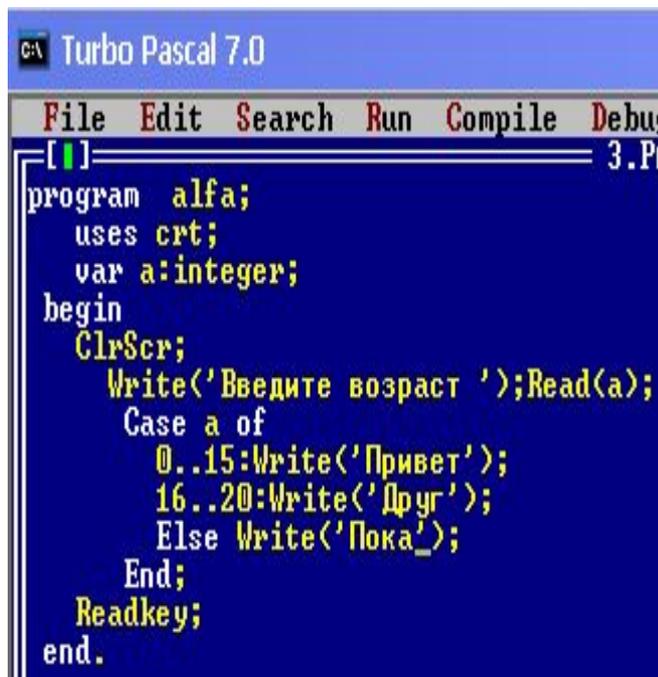
  case M of
    2:          begin D := 28; end;
    4,6,9,11:  begin D := 30; end;
    1,3,5,7,8,10,12: D := 31;
    else      D := -1;
  end;

  if D > 0 then
    writeln('В этом месяце ', D, ' дней.')
  else
    writeln('Неверный номер месяца');
  readkey;
end.
```

ни один вариант не подошел

Задания для самостоятельной работы

1. По введенному номеру от 1 до 7 определить день недели- например 1 – Понедельник и т.д.
2. Ставить программу , запрашивающую возраст и при введении до 15 включительно - говорит «Привет», от 16 до 20 говорит «друг», и говорит «Пока» в противном случае.



```
Turbo Pascal 7.0
File Edit Search Run Compile Debug
[ ] 3.PAS
program alfa;
uses crt;
var a:integer;
begin
  ClrScr;
  Write('Введите возраст ');Read(a);
  Case a of
    0..15:Write('Привет');
    16..20:Write('Друг');
    Else Write('Пока');
  End;
  Readkey;
end.
```



```
Turbo Pascal 7.0
File Edit Search Run Compile Debug Tools
[ ] 1.PAS
program alfa;
uses crt;
var a:integer;
begin
  ClrScr;
  Write('Введите число от 1 до 7 ');Read(a);
  Case a of
    1:Write('Понедельник');
    2:Write('Вторник');
    3:Write('Среда');
    4:Write('Четверг');
    5:Write('Пятница');
    6:Write('Суббота');
    7:Write('Воскресенье');
    Else Write('Нет такого дня');
  End;
  Readkey;
end.
```

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.
```

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**

- функция `function Max (a, b: integer): integer;` программы

Функции

Особенности:

- МОЖНО объявлять и использовать **локальные переменные**

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

- знач...
пере...
функции; объявлять ее **НЕ НАДО**:

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

Функции

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

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:



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

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;
```

По

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

Массивы

Задача: Заполнить массив из 5 элементов с клавиатуры и вывести на экран сумму 2 и 5 элементов

```
uses crt;  
    var i,sum: integer; a: array[1..5] of integer;  
begin  
    clrscr;  
    for i:=1 to 5 do  
        begin  
            write('Введите', i, 'элемент'); read (a[i]);  
        end;  
    sum:=a[2]+a[5];  
    write('Сумма 2 и 5 равна', sum);  
    readkey;  
end.
```

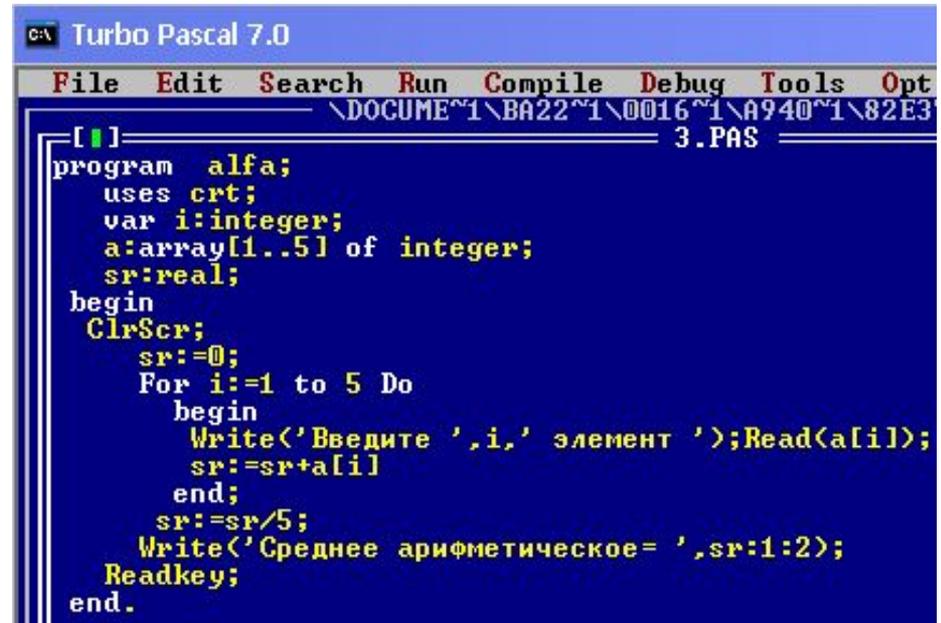
Задания для самостоятельной работы

1. Заполнить массив из 5 элементов и вывести на печать сначала все, ниже третий.



```
 Turbo Pascal 7.0
File Edit Search Run Compile Debug Tools
\DOCUME~1\BA22~1\0016~1\A940
[ ] 1.PAS
program alfa;
uses crt;
var i:integer; a:array[1..5] of integer;
begin
  ClrScr;
  a[1]:=3;a[2]:=7;a[3]:=6;a[4]:=9;a[5]:=2;
  For i:=1 to 5 Do
    Write(a[i], ' '); writeln;
    Write('3 элемент= ',a[3]);
  Readkey;
end.
```

2. Заполнить массив из 5 элементов с клавиатуры и найти их среднее арифметическое.



```
 Turbo Pascal 7.0
File Edit Search Run Compile Debug Tools Opt
\DOCUME~1\BA22~1\0016~1\A940~1\82E3
[ ] 3.PAS
program alfa;
uses crt;
var i:integer;
a:array[1..5] of integer;
sr:real;
begin
  ClrScr;
  sr:=0;
  For i:=1 to 5 Do
    begin
      Write('Введите ',i,' элемент ');Read(a[i]);
      sr:=sr+a[i]
    end;
  sr:=sr/5;
  Write('Среднее арифметическое= ',sr:1:2);
  Readkey;
end.
```

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'`

Символьные строки

Задача: ввести строку с клавиатуры и заменить все буквы "а" на буквы "б".

```
program qq;  
var s: string;  
    i: integer;  
begin  
    writeln('Введите строку');  
    readln(s);  
    for i:=1 to Length(s) do  
        if s[i] = 'a' then s[i] := 'б';  
    writeln(s);  
end.
```

ВВОД строки

длина строки

ВЫВОД строки

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:

```
s := '123456789';  
Delete ( s, 3, 6 );
```

6 штук

'12~~345678~~9'
'129'

строка
меняется!

с 3-его символа

Вставка в строку:

```
s := '123456789';  
Insert ( 'ABC', s, 3 );
```

начиная с 3-его символа

'12ABC3456789'

что
вставляем

куда
вставляем

```
Insert ( 'Q', s, 5 );
```

'12ABQC3456789'

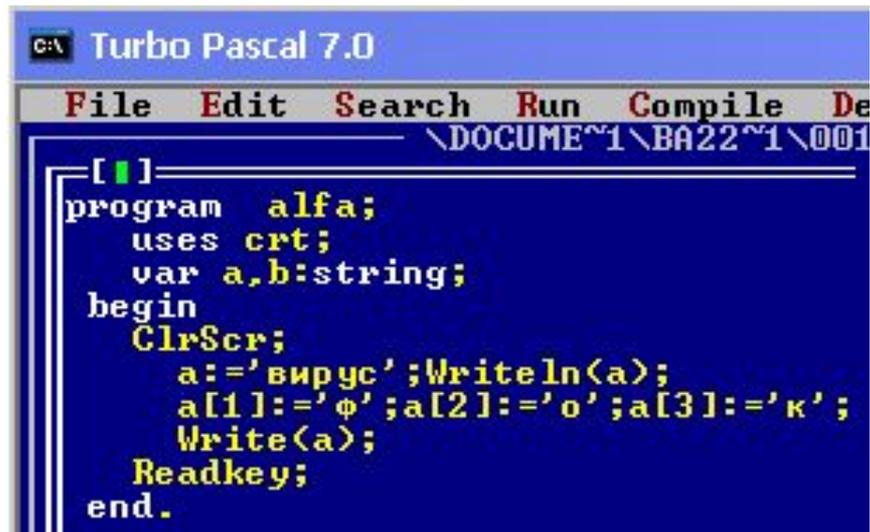
Задания для самостоятельной работы

1. Ввели слово, определить его длину и вывести 3 символ, а так же вывести вырезку с 2 символа , длиной 5 символов.



```
C:\ Turbo Pascal 7.0
File Edit Search Run Compile Debug
\DOCUME~1\BA22~1\0016~1\
1.Pa
[ ]
program alfa;
uses crt;
var a:string;
begin
  ClrScr;
  Write('Введите слово ');Read(a);
  Writeln(' Длина слова -',Length(a));
  Writeln(' Третий символ-',a[3]);
  Write(' Вырезка-',Copy(a,2,5));
  Readkey;
end.
```

2. Из слова «вирус» путем замены букв получите слово «фокус».



```
C:\ Turbo Pascal 7.0
File Edit Search Run Compile De
\DOCUME~1\BA22~1\001
[ ]
program alfa;
uses crt;
var a,b:string;
begin
  ClrScr;
  a:='вирус';Writeln(a);
  a[1]:='ф';a[2]:='о';a[3]:='к';
  Write(a);
  Readkey;
end.
```

Файлы

Файл – это область на диске, имеющая имя.

Файл

ы

Текстовые

е

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

ASCII (1 байт на символ)

UNICODE (2 байта на символ)

*.txt, *.log,

*.htm, *.html

Двоичные

могут содержать любые
символы кодовой таблицы

*.doc, *.exe,

*.bmp, *.jpg,

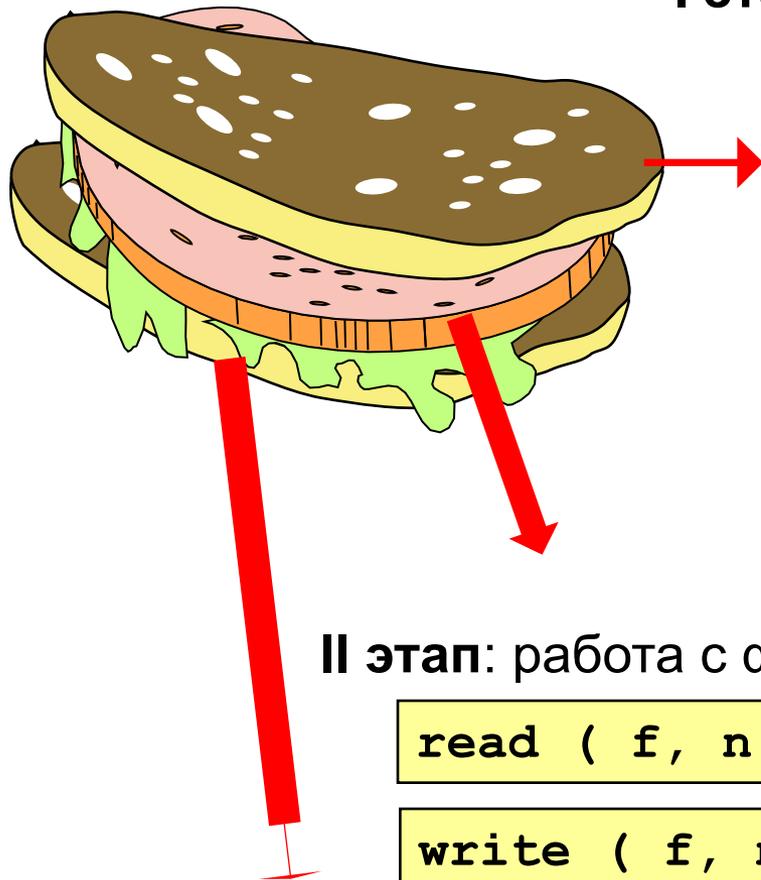
*.wav, *.mp3,

*.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

Пример

Задача: в файле `input.txt` записаны числа (в столбик), сколько их – неизвестно. Записать в файл `output.txt` их сумму.

Алгоритм:

1. Открыть файл `input.txt` для чтения.
2. $S := 0;$
3. Если чисел не осталось, перейти к шагу 7.
4. Прочитать очередное число в переменную x .
5. $S := S + x;$
6. Перейти к шагу 3.
7. Закрыть файл `input.txt`.
8. Открыть файл `output.txt` для записи.
9. Записать в файл значение S .
10. Закрыть файл `output.txt`.

цикл с условием
"пока есть данные"

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

Задача: в файле `input.txt` записаны строки, в которых есть слово-паразит "**короче**". Очистить текст от мусора и записать в файл `output.txt`.

Файл `input.txt` :

Мама, короче, мыла, короче, раму.

Декан, короче, пропил, короче, бутан.

А роза, короче, упала на лапу, короче, Азора.

Каждый, короче, охотник желает, короче, знать, где ...

Результат - файл `output.txt` :

Мама мыла раму.

Декан пропил бутан.

А роза упала на лапу Азора.

Каждый охотник желает знать, где сидит фазан.

Обработка текстовых данных

пока не кончились данные

Алгоритм:

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

Обработка строки `s`:

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

искать ", короче,"

удалить 9 символов

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

пока не достигнут конец файла

```
while not eof(fIn) do begin
```

```
  readln(fIn, s);
```

обработка строки

```
  repeat
```

```
    i := Pos(' , короче, ' , s);
```

```
    if i <> 0 then
```

```
      Delete(s, i, 9);
```

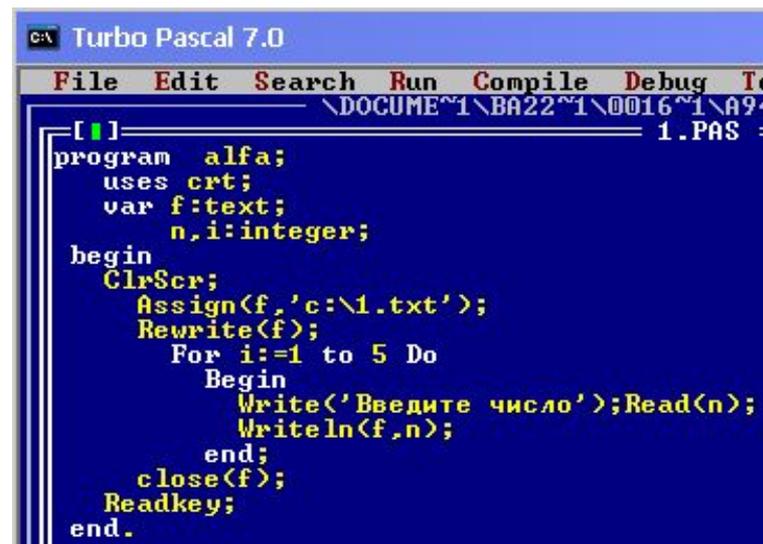
```
  until i = 0;
```

```
end;
```

запись "очищенной"
строки

Задания для самостоятельной работы

1. Создать на диске C файл 1.txt с 5 числами, введенными с клавиатуры.



```

Turbo Pascal 7.0
File Edit Search Run Compile Debug To
\DOCUME~1\BA22~1\0016~1\A94
1.PAS
[ ]
program alfa;
uses crt;
var f:text;
    n,i:integer;
begin
  ClrScr;
  Assign(f,'c:\1.txt');
  Rewrite(f);
  For i:=1 to 5 Do
  Begin
    Write('Введите число');Read(n);
    Writeln(f,n);
  end;
  close(f);
  Readkey;
end.

```

2. Дописать в существующий файл 1.txt данные, введенные с клавиатуры.



```

Turbo Pascal 7.0
File Edit Search Run Compile Debug
\DOCUME~1\BA22~1\0016~1\
5.PA
[ ]
program alfa;
uses crt;
var f:text;
    n,i:integer;
begin
  ClrScr;
  Assign(f,'c:\1.txt');
  Append(f);
  Write('Введите число');Read(n);
  Writeln(f,n);
  close(f);
  Readkey;
end.

```