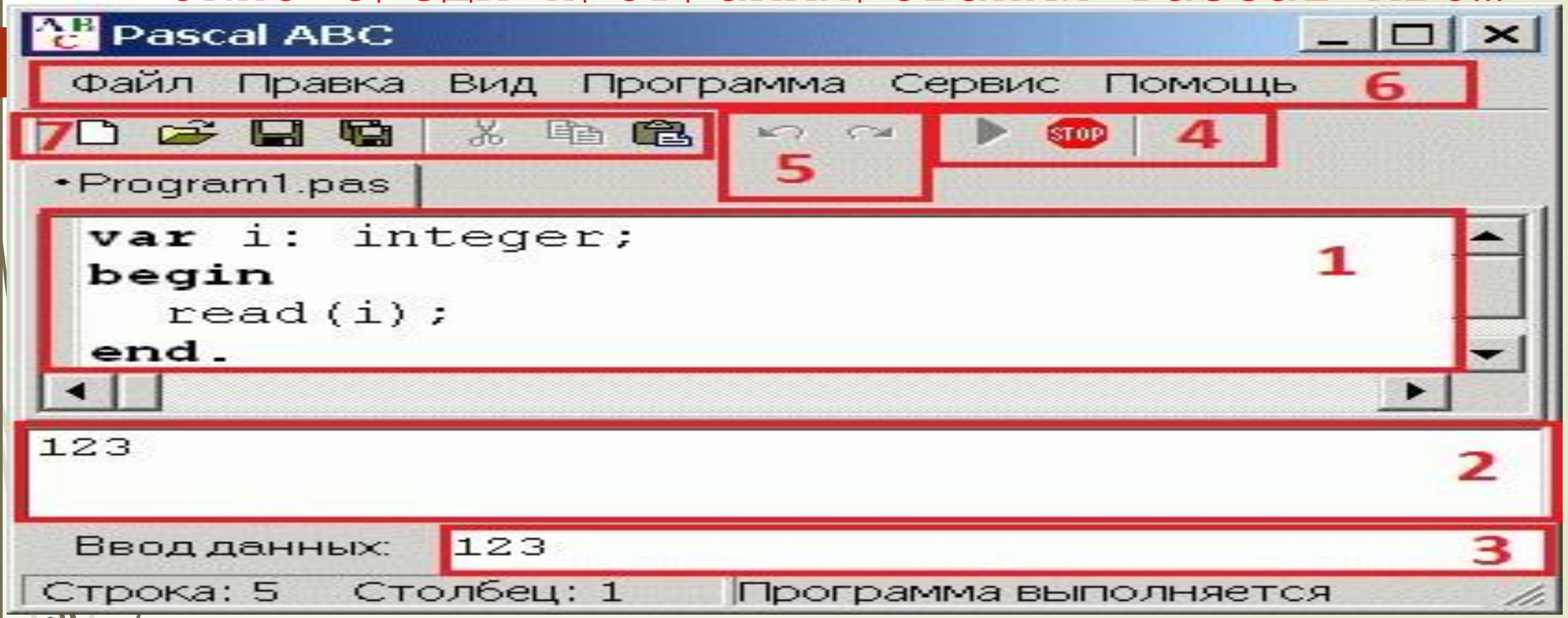
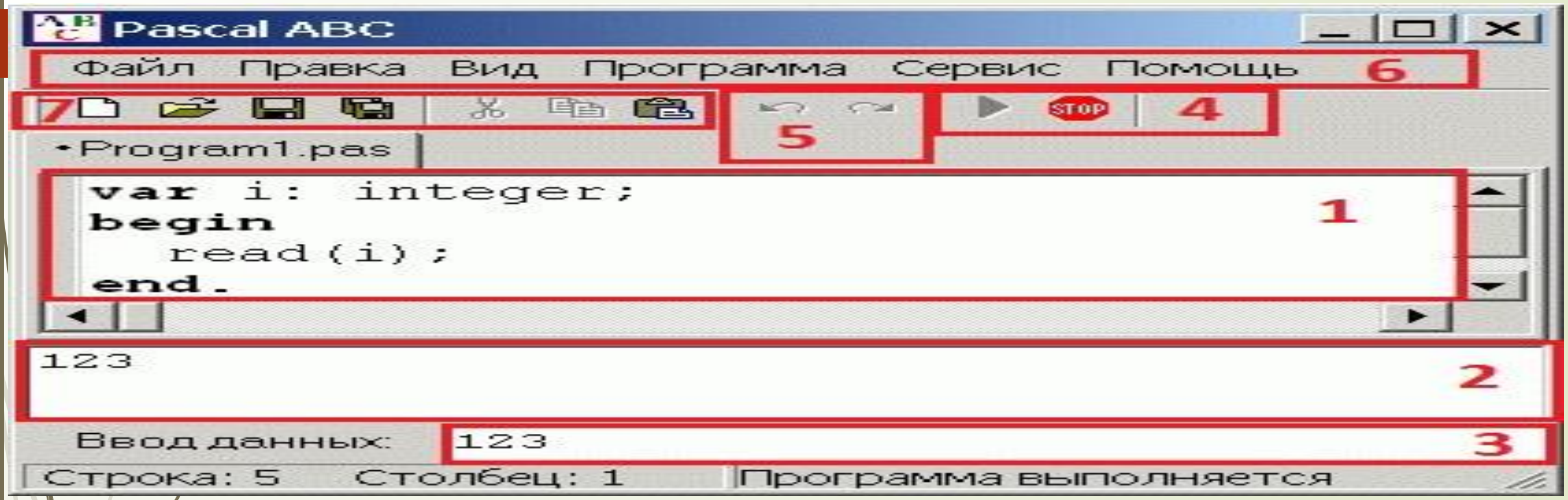
Pascal ABC – это просто!

Окно среды программирования Pascal ABC…



1. **Окно редактора** – окно для редактирования программного кода
2. **Окно вывода** – в нем происходит вывод данных программой

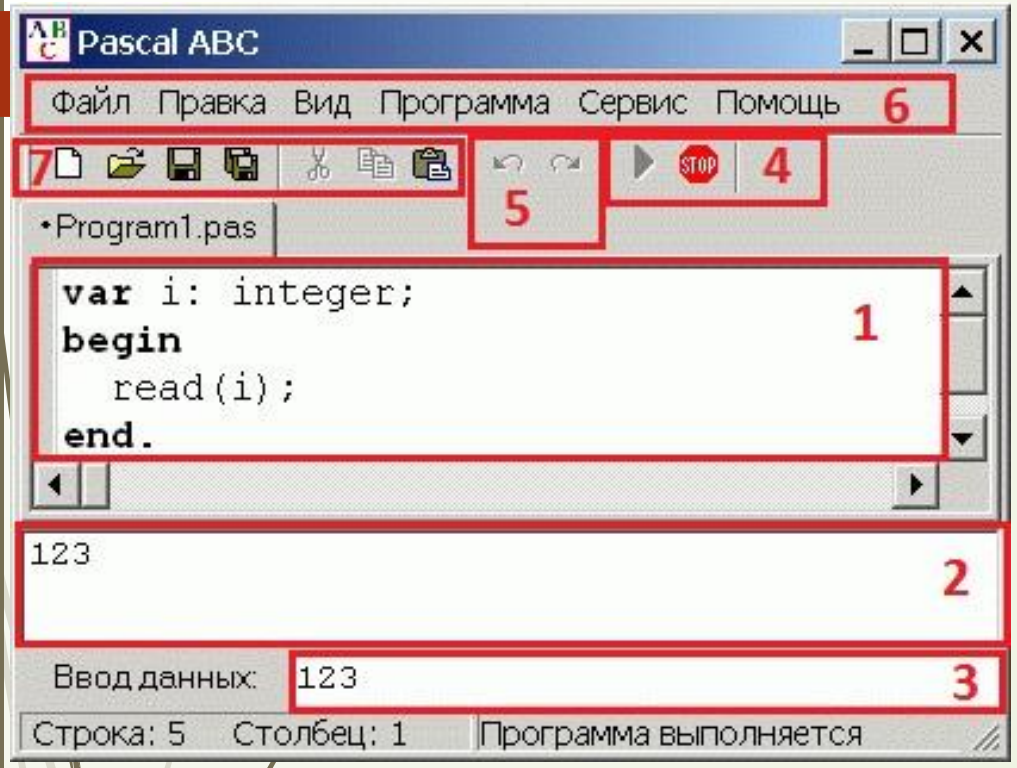
Окно среды программирования Pascal ABC…



3. **Окно ввода** – служит для передачи программе каких-либо данных

4. **Кнопки управления** – предназначены для запуска, а так-же прерывания программы (**F5**-выполнить, **Ctrl+F2** -завершить)

Окно программы Pascal ABC…



6.Панель управления – нужна, для создания нового кода, сохранения проекта, поиска ошибок, получения справки, загрузи примеров кода и т.д.

7.Панель быстрого доступа – предназначена для сохранения кода **(Ctrl+S)**, создание нового **(Ctrl+N)**, копирования, открытия существующего проекта **(Ctrl+O)**.

Типы и описания констант и переменных.

Особенностями языка являются строгая типизация.
Перед тем как писать программу, сначала нужно определиться с типами данных , которые нам понадобятся…
Итак!

Основные типы переменных и констант –

Целые числа

1. **INTEGER** диапазон значений -2147483648.. 2147483647
2. **BYTE** диапазон значений 0 .. 255
3. **WORD** диапазон значений 0 .. 65 535

Символьный тип: CHAR Значения этого типа занимают 1 байт и представляют собой символы кодовой таблицы в кодировке Windows

Типы и описания констант и переменных.

Действительные числа

1. **REAL** Значения вещественного типа занимают 8 байт, содержат 15-16 значащих цифр и по модулю не могут превосходить величины $1.7 \cdot 10^{308}$.

2. Текст: формат STRING диапазон значений 0-255 символов

Формат BOOLEAN (логический): диапазон значений true или false

Стандартные процедуры и функции

Read(a,b..) – вводит значение с клавиатуры в переменные
Write(a,b..) – выводит значение переменных в окно вывода
Readln(a,b..) – считывает данные с последующим переносом строки
Writeln(a,b..) – выводит значение с последующим переносом строки
Abs(x) – возвращает модуль переменной
Sqr(x) – возвращает квадрат переменной
Sqrt(x) – возвращает квадратный корень из переменной
Random(x) функция от x - типа integer возвращает случайное целое в диапазоне от 0 до x-1

Стандартные процедуры и функции

Ln(x) – возвращает натуральный логарифм
Exp(x) – возвращает **e** в степени **x**
Sin(x) – возвращает значение *sin X* переменной (x задается в радианах)
Cos(x) – возвращает значение *cos X* переменной
Arctan(x) – возвращает значение *arctg X* переменной

Структура программы

Программа на языке Pascal ABC имеет следующий вид:

program имя программы;
раздел подключения модулей
раздел описаний
Begin
  операторы
End.

Раздел описаний может включать описания: переменных, констант, типов, процедур и функций, 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".

Описание переменных

Переменные могут быть описаны в разделе описаний, а также непосредственно внутри описания подпрограмм.
Раздел описания переменных начинается со служебного слова **var**, после которого следуют элементы описания вида

список имен: тип;

или

имя: тип := выражение;

Описание переменных

Имена однотипных переменных в списке перечисляются через запятую. Например:

```
var
  a,b,c: integer;
  d: real := 3.7;
```

Оператор присваивания

Оператор присваивания имеет вид:

переменная:= выражение

Например:

a: = g*sqr(t);

Выражение должно иметь тип, либо совпадающий с типом переменной, либо к нему приводящийся.

Условный оператор

Условный оператор
имеет *полную* и *краткую* формы.
Полная форма условного оператора выглядит следующим образом:
if *условие* **then** *оператор1*
else *оператор2*
В качестве условия указывается некоторое логическое выражение. Если условие оказывается истинным, то выполняется *оператор1*, в противном случае выполняется *оператор2*.

Условный оператор

Пример полной формы :

```
k:=0; p:=1;
if k < p then
  k := p;
else
  p := k;
```

Если k < p, тогда **k := p,** иначе **p := k;**

Здесь k получит значение 1

Условный оператор

Краткая форма условного оператора имеет вид:
if *условие* **then** *оператор*
Если условие оказывается истинным, то выполняется *оператор*, в противном случае происходит переход к следующему оператору программы.

Условный оператор

Пример краткой формы :

```
k:=1; p:=0;
if k < p then
  k := p;
```

Если k < p, тогда **k := p**.

Если условие оказывается ложным, то *оператор после* **then** не выполняется, т.е. **k** остается равным **1**

Составной оператор (блок)

Составной оператор предназначен для объединения нескольких операторов в один. Он имеет вид:

```
begin
  операторы
end
```

Например:

```
 D:=b*b-4*a*c;
if D>=0 then
  begin
    x1:=(-b-sqrt(D))/(2*a);
    x2:=(-b+sqrt(D))/(2*a);
  end
```

Условный оператор

В случае конструкции вида
if *условие1* **then**
  if *условие2* **then** *оператор1*
  else *оператор2*
else всегда относится к ближайшему предыдущему оператору **if**, для которого ветка **else** еще не указана.

Условный оператор

Если в предыдущем примере требуется, чтобы **else** относилась к первому оператору **if**, то необходимо использовать составной оператор:

```
if условие1 then
begin
  if условие2 then оператор1
end
else оператор2
```

Условный оператор

Пример :

```
k:=11; p:=7; m:=5; h:=8;
if k < p then
  if h = p then  p := m
  else p := h
else
  m := k;
WriteIn(p,h,m);
```

Если k < p, тогда проверяем, если h = p, тогда p := m, иначе p := h.

Если условие k < p – ложно, тогда m := k

Циклы

Цикл – группа операторов, которая может быть выполнена много раз подряд.

В PABC различают 3 основных операторов цикла

1) **For**
2) **While**
3) **Repeat**

Цикл for

Оператор цикла **for** предполагает заранее определенное количество итераций и имеет одну из двух форм:

for *переменная := начальное значение* **to** *конечное значение* **do**
оператор

или

for *переменная := начальное значение* **downto** *конечное значение* **do**
оператор

Цикл for

Кроме того, переменную можно описать непосредственно в заголовке цикла:

for *переменная: тип := начальное значение* **to** *или* **downto** *конечное значение* **do**
оператор

или

for var *переменная := начальное значение* **to** *или* **downto** *конечное значение* **do**
оператор

Цикл for

Текст от слова **for** до слова **do** включительно называется **заголовком цикла**, а оператор после **do** - **телом цикла**. Переменная после слова **for** называется **параметром цикла**. Для первой формы цикла с ключевым словом **to** параметр цикла меняется от начального значения до конечного значения, увеличиваясь всякий раз на единицу, а для второй формы ключевым словом **downto** - уменьшаясь на единицу.

Цикл for

Для каждого значения переменной-параметра выполняется тело цикла. Однократное повторение тела цикла называется **итерацией цикла**. Значение параметра цикла после завершения цикла считается неопределенным.

Цикл for

Пример : создать таблицу вычисления квадратов натуральных чисел от 1 до 20.

```
m:=20;
for var i := 0 to m do
  writeln(i, '² = ', i*i);
```

Цикл while

Оператор цикла **while** имеет следующую форму:
while *условие* **do**
оператор
Условие представляет собой выражение логического типа, а оператор после **do** называется *телом цикла*. Перед каждой итерацией цикла условие вычисляется, и если оно истинно, то выполняется тело цикла, в противном случае происходит выход из цикла.

Цикл while

Теперь опишем пример вывода таблицы, как в цикле for, чтобы увидеть отличия данных операторов.

Пример : создать таблицу вычисления квадратов натуральных чисел от 1 до 20.

```
k:=1; m:=2;
while k < m do
  begin
    writeln(k, '² = ', k*k);
    m:=21;
    k:=k+1
  end
```

Цикл repeat

Оператор цикла **repeat** имеет следующую форму:

repeat
  операторы
until *условие*

В отличие от цикла **while**, условие вычисляется после очередной итерации цикла, и если оно истинно, то происходит выход из цикла. Таким образом, операторы, образующие тело цикла оператора **repeat**, выполняются по крайней мере один раз.

Цикл repeat

Если условие всегда истинно, то может произойти *зацикливание:*

```
repeat
   write(1);
until 2=1;
```

Массивы

Массив – набор элементов одного типа, каждый из которых имеет свой номер, называемый *индексом* (индексов может быть несколько, тогда массив называется многомерным)

Массивы

Тип массива конструируется следующим образом:

array [*тип индекса**1***, …, *тип индекса**N***] **of** ***базовый тип***

Тип индекса обязательно представляет собой интервальный тип и обязательно должен задаваться в виде **a..b**, где **a** и **b** - константные выражения целого, символьного или перечислимого типа

Массивы

Тип массива конструируется следующим образом:

array [*тип индекса**1***, …, *тип индекса**N***] **of** ***базовый тип***

Тип индекса обязательно представляет собой интервальный тип и обязательно должен задаваться в виде **a..b**, где **a** и **b** - константные выражения целого, символьного или перечислимого типа

Массивы

Например:

```
var
  a1,a2: array [1..10] of integer;
  b: array ['a'..'z', ‘d'..‘g'] of string;
  c: array [1..10] of array [1..5] of real;
```

В последнем случае имеем массив массивов действительных чисел.

Массивы

К элементам массива обращаются при помощи переменных с индексами:

a1[3]:=**a2**[5];

b['f', ‘e']:='Hello';

c[3][4]:=3.14;

Описание процедур функций

Процедура или функция представляет собой последовательность операторов, которая имеет имя, список параметров и может быть вызвана из различных частей программ по имени. Функции, в отличие от процедур, в результате своего выполнения возвращают значение.

Процедурный тип

Переменные, предназначенные для хранения процедур и функций, называются *процедурными*.
Тип *процедурной* переменной имеет вид:

procedure (***список параметров***) или

function (***список параметров***) : ***тип возвращаемого значения***

Процедурный тип

Процедурной переменной можно присвоить *процедуру* или *функцию* с **совместным типом**:

```
procedure my(i: integer);
begin
  ...
end;
function f: integer;  begin
end;

...
p1:=my;
f1:=f;
```

Процедурный тип

Переменные могут описываться непосредственно внутри подпрограммы. Эти описания переменных имеют тот же вид, что и в разделе описаний основной программы

После этого можно вызвать процедуру или функцию через эту процедурную переменную, пользуясь обычным синтаксисом вызова:

```
p1(5);
write(f1);
```

Описание процедур функций

Описание процедуры имеет вид:

procedure имя(список формальных параметров);

раздел описаний

begin

операторы

end;

Описание процедур функций

Описание функции имеет вид:

function *имя*(*список формальных параметров*): *тип возвращаемого значения*;
раздел описаний
begin
операторы
end;

Описание процедур и функций

Пример описания процедуры :

```
procedure Reverse(var a: array [1..10] of integer; n: integer);
var i,v: integer;
begin
  for i:=1 to n div 2 do
  begin
    v:=a[i];
    a[i]:=a[n-i+1];
    a[n-i+1]:=v;
    Writeln(a[i]);
  end;
end;
```

Описание процедур функций

Пример вызова процедуры :

```
var i: integer;
    Pet: array [1..10] of integer;
begin
 for i:=1 to 10 do
  Readln(Pet[i]);
  Reverse(Pet, 10);

end.
```

Описание процедур функций

Пример описания функции:

```
function MinElement(var a: array [1..10] of real; n:
integer): real;
var i: integer;
begin
  Result:=a[1];
  for i:=1 to n do
    if a[i]<Result then Result:=a[i];
end;
```

Локальные и глобальные переменные

Переменные, описанные в разделе описаний подпрограммы, называются ее *локальными* переменными. Переменные же, описанные вне подпрограммы, называются *глобальными* по отношению к ней. Если имя локальной переменной совпадает с именем глобальной переменной, то локальная переменная скрывает глобальную, так что к глобальной переменной нельзя обратиться внутри подпрограммы

Локальные и глобальные переменные

Например:

```
var i: real;
procedure p;
  var i: integer;
   begin
     // к глобальной переменной i внутри процедуры нельзя обратиться
      i:=5; // присваивание локальной переменной i;
   end;
```