Level Up! Delphi

Занятие №4

Массивы

Метод бинарного поиска

На практике довольно часто производится поиск в массиве, элементы которого упорядочены по некоторому критерию (такие массивы называются упорядоченными). Например, массив фамилий, как правило, упорядочен по алфавиту, массив данных о погоде — по датам наблюдений. В случае, если массив упорядочен, то применяют другие, более эффективные по сравнению с методом простого перебора алгоритмы, один из которых — *метод бинарного поиска*.

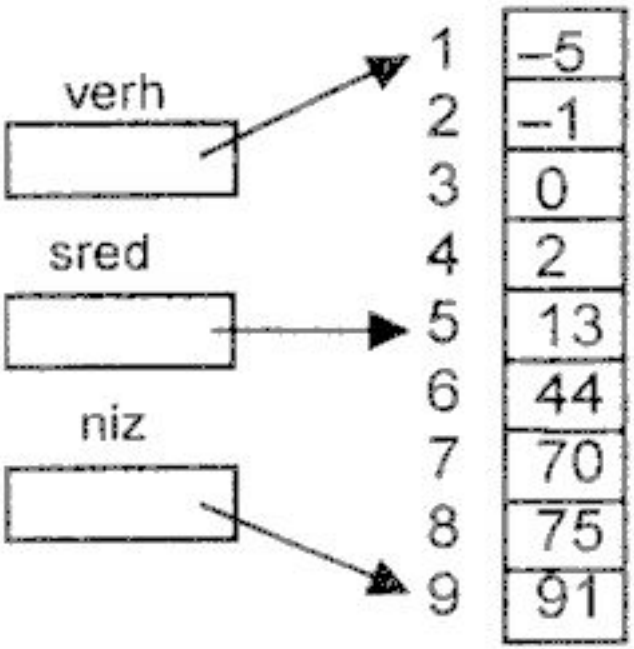
Пусть есть упорядоченный по возрастанию массив целых чисел. Нужно определить, содержит ли этот массив некоторое число (образец).

Массивы

Метод (алгоритм) бинарного поиска реализуется следующим образом:

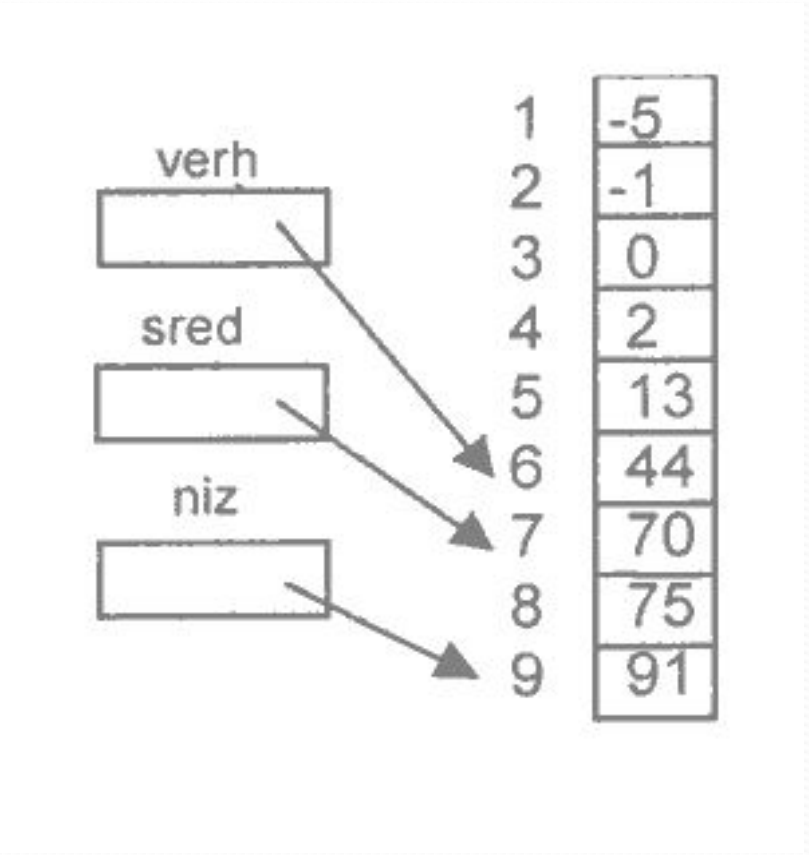
1. Сначала образец сравнивается со средним (по номеру) элементом массива.

Если образец равен среднему элементу,
то задача решена.



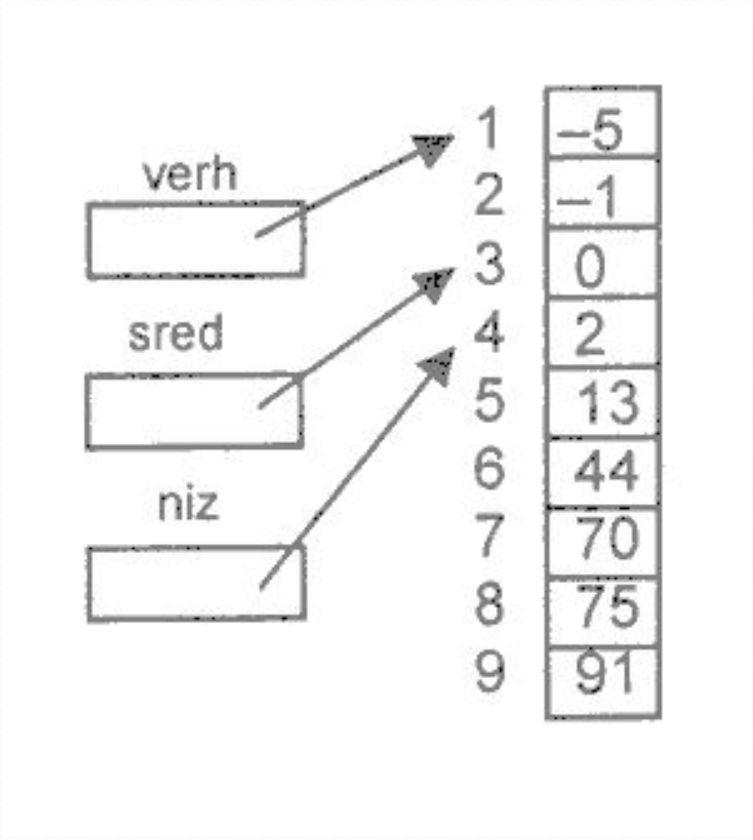
Массивы

Если образец больше среднего элемента, то это значит, что искомый элемент расположен ниже среднего элемента (между элементами с номерами sred+1 и niz), и за новое значение verh принимается sred+i, а значение niz не меняется.



Массивы

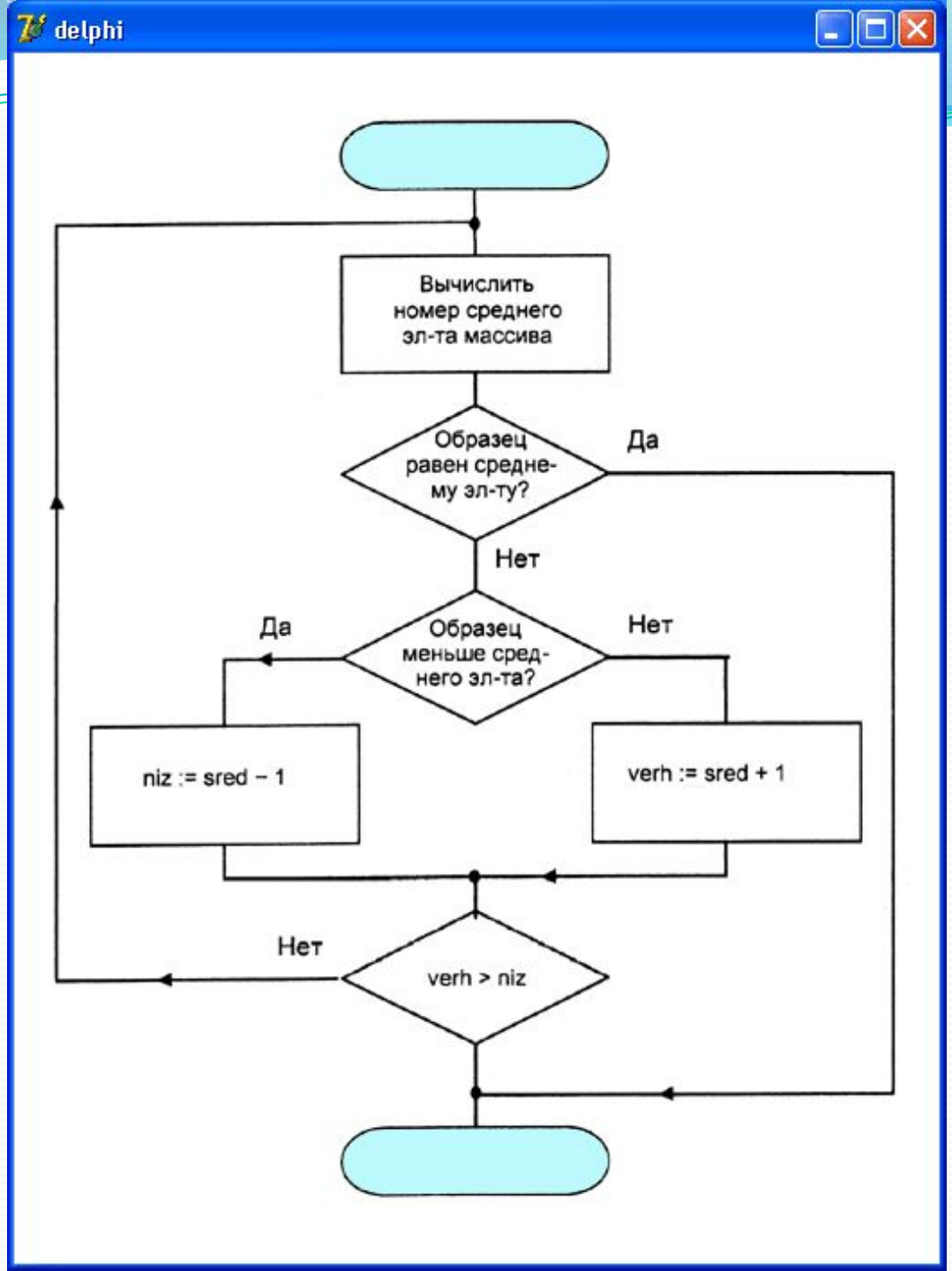
Если образец меньше среднего элемента, то это значит, что искомый элемент расположен выше среднего элемента (между элементами с номерами verh и sred-1), и за новое значение niz принимается sred-1, а значение verh не меняется.



Массивы

2. После того как определена часть массива, в которой может находиться искомый элемент, по формуле (niz-verh) /2+verh вычисляется новое значение sred и поиск продолжается.

Алгоритм бинарного поиска заканчивает свою работу, если искомый элемент найден или если перед выполнением очередного цикла поиска обнаруживается, что значение verh больше, чем niz.


delphi
Вычислить номер среднего эл-та массива
Образец равен средне-му эл-ту?
Да
Нет
Образец меньше сред-него эл-та?
Да
Нет
niz := sred – 1
verh := sred + 1
verh > niz
Нет

Двумерные массивы

Представьте себе таблицу, состоящую из нескольких строк. Каждая строка состоит из нескольких ячеек. Тогда для точного определения положения ячейки нам потребуется знать не одно число (как в случае таблицы линейной), а два: номер строки и номер столбца. Структура данных в языке Delphi для хранения такой таблицы называется двумерным массивом. Описать такой массив можно двумя способами:

```
Var
  A : Array [1..20] Of Array [1..30] Of Integer;

  A : Array [1..20,1..30] Of Integer;
```

Двумерные массивы

```
Var
   A : Array[1..10,1..10] Of Integer;
   I, J : Byte;
   Sum : Integer;
begin
   Sum := 0;
   For I :=1 To 10 Do
      Begin
        For J :=1 To 10 Do
          Begin
            A[I,J] := Trunc(Random * 100) + 1;
            Write(A[I,J]:6);
            If (J > I) Then Sum := Sum + A[I,J]
          End;
        Writeln
      End;
  Writeln('Сумма элементов выше гл. диагонали равна ', Sum)
end;
```

Типы данных, определяемые программистом

Типы данных, определяемые программистом

Объявляемый программистом новый тип данных базируется на стандартных типах или на типах, созданных программистом ранее. Тип, определенный программистом, может быть отнесен к:

- перечисляемому;
- интервальному;
- составному типу данных (записи).

Типы данных, определяемые программистом

Перечисляемый тип

Определить перечисляемый тип — это значит перечислить все значения, которые может принимать переменная, относящаяся к данному типу.

В общем виде объявление перечисляемого типа выглядит так:

Тип =(Значение1, Значение2, … Значение i)

Примеры:

TDayOfWeek = (MON,TUE,WED,THU,FRI,SAT,SUN);
TColor = (Red,Yellow,Green);

Типы данных, определяемые программистом

```
type
TDayOfWeek = (MON,TUE,WED,THU, FRI,SAT,SUN) ;

var
   ThisDay, LastDay: TDayOfWeek;
```

{MON < TUE < WED < THU < FRI < SAT < SUN}

Типы данных, определяемые программистом

```
if (Day = SAT) or (Day = SUN) then
begin

{ действия, если день — суббота или воскресенье }

end;
```

Типы данных, определяемые программистом

Интервальный тип

При объявлении интервального типа указываются нижняя и верхняя границы интервала, т. е. наименьшее и наибольшее значение, которое может принимать переменная объявляемого типа. В общем виде объявление интервального типа выглядит так:

Тип = НижняяГраница .. ВерхняяГраница;

Примеры:

```
TIndex = 0 .. 100;
TRusChar = 'А' .. 'я';
```

Типы данных, определяемые программистом

```
const
    HBOUND = 100;
type
    Tindex = l .. HBOUND;
```

```
type
    TIndex = 1 .. 100;
var
  tabl : array[TIndex] of integer;
  i: TIndex;
```

Процедуры и функции

Процедуры и функции

Подпрограмма — это небольшая программа, которая решает часть общей задачи. В языке Delphi есть два вида подпрограмм — процедура и функция.

У каждой подпрограммы есть имя, которое используется в программе для вызова подпрограммы (процедуры).

Отличие функции от процедуры состоит в том, что с именем функции связано значение, поэтому функцию можно использовать в качестве операнда выражения, например, инструкции присваивания.

Процедуры и функции

Как правило, подпрограмма имеет параметры. Различают формальные и фактические параметры.

Параметры, которые указываются в объявлении функции, называются *формальными*. Параметры, которые указываются в инструкции вызова процедуры, называются *фактическими*.

Параметры используются:

- для передачи данных в подпрограмму;
- для получения из результата подпрограммы.

В общем случае в качестве фактического параметра процедуры можно использовать выражение, тип которого должен совпадать с типом соответствующего формального параметра.

Процедуры и функции

6. Способы передачи данных в подпрограмму

Если параметр определен как параметр-значение, то при вызове подпрограммы автоматически создается локальная переменная, куда копируется значение фактического параметра (переменной, константы или результата вычисления выражения). После этого фактический параметр становится недоступным из подпрограммы. Подпрограмма работает с созданными локальными переменными. Таким образом, фактические и формальные параметры при передаче параметров-значений должны быть совместимыми по присваиванию. Изменения параметра-значения в теле подпрограммы не приведут к изменениям значения переменной головной программы, соответствующей фактическому параметру (рис. 4).

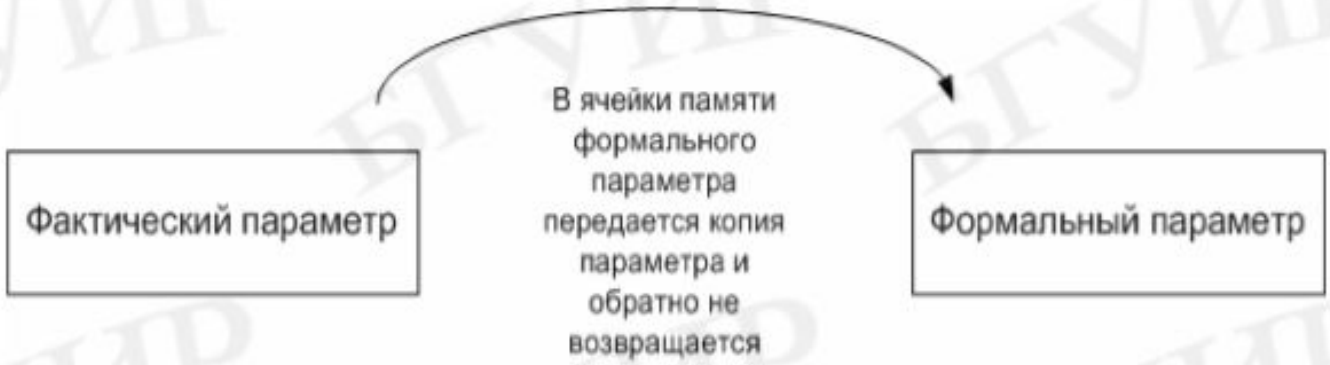


Рис. 4. Механизм передачи параметра-значения

Процедуры и функции

Если параметр определен как параметр-константа, то при вызове подпрограммы данному параметру может соответствовать фактический параметр в виде константы, переменной нужного типа или выражения, результат вычисления которого имеет нужный тип. При передаче параметра-константы в подпрограмму передается адрес области памяти, в которой располагается фактический параметр. Однако любое изменение значения параметра-константы в подпрограмме невозможно (рис. 5).

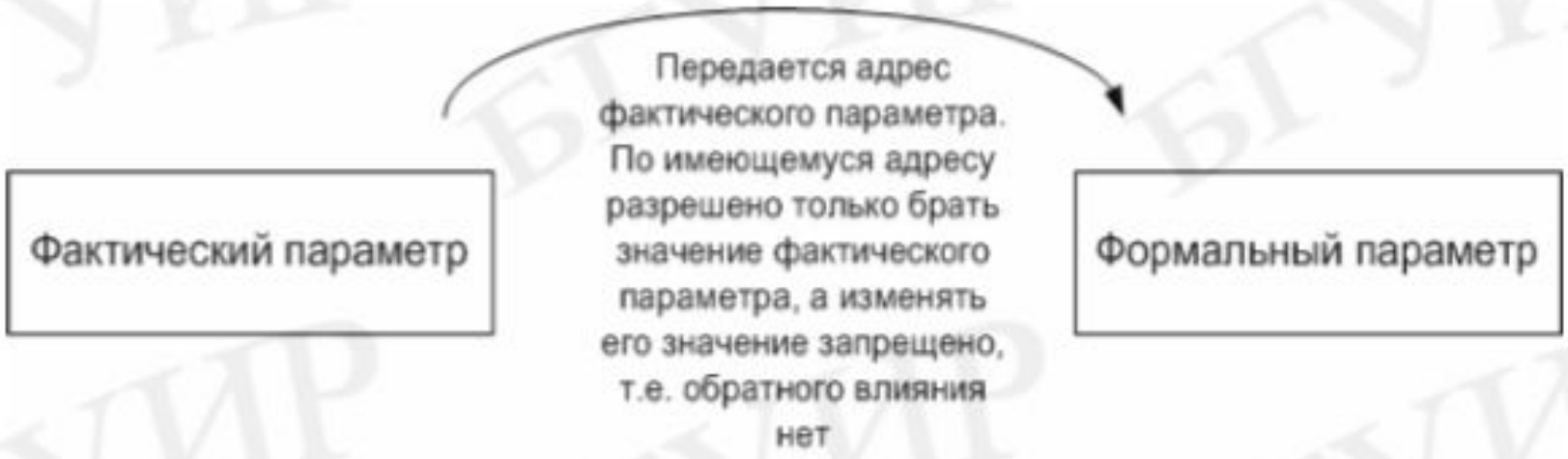


Рис. 5. Механизм передачи параметра-константы

Процедуры и функции

Если параметр определен как параметр-переменная, то при вызове подпрограммы в нее передается адрес фактического параметра. Поэтому фактическим параметром в данном случае может быть только переменная того же типа, что и тип формального параметра. Любое изменение в теле подпрограммы формального параметра ведет к изменению в головной программе фактического параметра. Поэтому результаты работы подпрограммы могут быть переданы в вызывающую программу, используя параметры-переменные (рис. 6).

Рис. 6. Механизм передачи параметра-переменной

Процедуры и функции

Процедуры — это конструкции программного кода, которые позволяют создавать в программном коде некоторые подпрограммы, которые выполняют определенные операции независимо от остального программного кода.

Функции — это по сути те же процедуры, но только функции могут возвращать результат, т.е. какое-либо значение.

Процедуры и функции

```
procedure shownumbers(n:integer);
var i:integer;
begin
for i:=1 to n do
showmessage(inttostr(i));
end;

function calc(a,b,c:integer; d:string): integer;
begin
result:=length(d);
inc(a,5);
dec(b,2);
inc(result, a+b-c);
end;
```

Процедуры и функции

Процедура — это разновидность подпрограммы. Обычно подпрограмма реализуется как процедура в двух случаях:

- когда подпрограмма не возвращает в основную программу никаких данных. Например, вычерчивает график в диалоговом окне;
- когда подпрограмма возвращает в вызвавшую ее программу больше чем одно значение. Например, подпрограмма, которая решает квадратное уравнение, должна вернуть в вызвавшую ее программу два дробных числа — корни уравнения.

Процедуры и функции

Объявление процедуры.

В общем виде объявление процедуры выглядит так:
procedure *Имя* (var *параметр1: тип1; …* var *параметрК:типК) ;*
var
// *здесь объявление локальных переменных*
begin

// здесь *инструкции процедуры*

end;

Процедуры и функции

Список параметров — это механизм передачи значений процедурам (равно как и функциям). Список параметров может содержать один или более параметров. Если список содержит более одного параметра, они разделяются точкой с запятой.

procedure DisplayString(s: string);

Процедуры и функции

Delphi обладает огромным множеством стандартных функций, которые можно использовать в приложениях. Две из этих процедур действительно просты, очень полезны и часто используются в целях оптимизации кода. Это процедуры **Inc** и **Dec**. Процедура **Inc** служит для увеличения, а процедура **Dec** — для уменьшения перечислимого значения на единицу или более.

Заголовки этих процедур показаны ниже:

procedure Inc(var X [; N: Longint]);

procedure Dec(var X [; N : Longint]);

Процедуры и функции

inc(a;integer;b:integer) и dec(a:integer;b:integer)

Первая увеличивает целочисленное число «a» на «b» единиц.

Вторая процедура уменьшает целочисленное число «a» на «b» единиц.

Если переменной «b» не указать значение, то вместо числа «b» автоматически будет использоваться единица. Таким образом «inc(a);» — тоже самое что и «inc(a,1);». В качестве «b» могут выступать и отрицательные числа, что приведет к инверсии операции, т.е. «inc(a,-3);» — тоже самое что и «dec(a,3);». Процедуры инкремента и декремента использовать несколько удобнее и работают они относительно быстрее чем присвоение «a:=a+1;».

Процедуры и функции

```
var
   x: Integer;
   c: Char;
begin
   x := 10;
   Inc(x); { faster way of writing x := x + 1; }
   Dec(x); { faster way of writing x := x - 1; }
   Inc(x, 5); { faster way of writing x := x + 5; }
   Write(x);
   c := 'a';
   Inc(c);
   Write(c); { c := 'b'; }
   ReadLn;
end.
```

Процедуры и функции

```
program Project1;
{$APPTYPE CONSOLE}
uses
  SysUtils;
procedure Hello;
begin
  WriteLn('Level Up Delphi');
end;
begin
end.
```

Процедуры и функции

```
program Project1;
{$APPTYPE CONSOLE}
uses
SysUtils;
var
   i:  Integer;
procedure Hello;
begin
   WriteLn('Level Up Delphi');
end;
Begin
   for i := 1 to 20 do
      Hello;
End.
```

Процедуры и функции

В общем виде инструкция обращения к функции выглядит так:

Переменная := Функция (Параметры) ;

где:

Переменная — имя переменной, которой надо присвоить значение, вычисляемое функцией;

Функция — имя функции, значение которой надо присвоить переменной;

Параметры — список формальных параметров, которые применяются для вычисления значения функции. В качестве параметров обычно используют переменные или константы.

Процедуры и функции

Следует обратить внимание на то, что:

- каждая функция возвращает значение определенного типа, поэтому тип переменной, которой присваивается значение функции, должен соответствовать типу функции;

- тип и количество параметров для каждой конкретной функции строго определены.

Объявление функции в общем виде выглядит так:

function Имя (параметр1 : тип1, …, *параметрК : типК) : Тип;*
var
// здесь *объявления локальных переменных*
begin

// здесь инструкции *функции*

Имя := Выражение; // или Result := Выражение;
end;

Процедуры и функции

Во всех функциях присутствует зарезервированная переменная под названием «result», которая хранит в себе результат функции, т.е. ее значение. Тип переменной «result» указывается после перечисления параметров (после закрывающей скобки и двоеточия). Внутри программного кода функции мы также как и в процедуре можем производить любые операции над параметрами, а также и над переменной «result». Также как и в процедуре, в функции необязательны параметры. Тогда она может выглядеть так:

```
function example: string;
begin
result:='simple function';
end;
```

Процедуры и функции

```
function NOD(A, B: integer): integer;
begin
while (A<>0) and (B<>0) do
if (A>B) then A:=A mod B
        else B:=B mod A;
NOD:=A+B
end;
…
Begin
  k := NOD(25,105);
End.
```

Процедуры и функции

Функции и процедуры могут вызывать сами себя. Такой прием программирования называется рекурсией и используются чаще всего в реализации каких-либо алгоритмов.

Название процедур и функций может содержать только латинские буквы, цифры и знаки подчеркивания.

Процедуры и функции

Функции и процедуры с любыми различиями в параметрах считаются абсолютно разными и независимыми друг от друга, даже если у них одинаковые названия. Если существует две функции или процедуры с одинаковыми названиями, но разными параметрами, то при вызове одной из этих процедур или функций будет использоваться автоматически та, которая подходит по параметрам (по их типам и количеству).

Спасибо за внимание ☺️