

# Лекция 13

## **1. Процедуры и функции**

- 1. Формальные и фактические параметры**
- 2. Глобальные и локальные переменные**
- 3. Параметры-значения и параметры-переменные.**
- 4. Использование рекурсии**

## **2. Модули**

## **3. Типы переменных**

## **4. Записи**

# Вопросы к экзамену

## 1. Информатика

1. Предмет и задачи информатики.
2. Информация. Различные определения. Количество информации.
3. Информационные системы. Информационные технологии.
4. Системы счисления. Непозиционные и позиционные системы. Двоичная и шестнадцатеричная системы счисления. Перевод из одной системы счисления в другую.
5. Арифметические основы работы ЭВМ.
6. Логические основы работы ЭВМ. Операции отрицания, дизъюнкции, конъюнкции, эквиваленции, импликации.
7. Базовая аппаратная конфигурация персонального компьютера. Системный блок. Монитор. Клавиатура. Мышь. Дополнительные устройства.
8. Внутренние устройства персонального компьютера. Блок питания. Материнская плата. Жесткий диск. Дисковод компакт-дисков. Видеокарта (видеоадаптер). Звуковая карта.
9. Прерывания. Виды прерываний.
10. Технология Plug and Play.
11. BIOS. Назначение.
12. **Операционные системы. Основные функции. Загрузка ОС.**
13. **Принципы внедрения и связывания объектов.**

## 2. MS Word

1. Назначение MS Word. Форматы поддерживаемых файлов. Возможности программы.
2. Создание стиля.
3. Использование шаблонов.
4. Использование макросов.
5. Создание вычисляемых и отображаемых формул.
6. Колонтитулы и нумерация страниц.
7. Графические объекты. Вставка изображений в документ. Объекты WordArt.
8. **Технология OLE.**

# Вопросы к экзамену. Delphi

1. **Алгоритмы и блок схемы.**
2. Языки программирования. Компиляторы. Интерпретаторы. Уровни языков.
3. Интегрированная среда Delphi.
4. Характеристика проекта в Delphi. Формы, модули.
5. **Структура программы в Delphi.**
6. Типы данных в Delphi. Простые типы: целочисленные, вещественные, логические, литерные.
7. Типы данных в Delphi. Массивы. Статические и динамические массивы
8. Типы данных в Delphi. Записи.
9. Типы данных в Delphi. Переменные файлового типа.
10. Типы данных в Delphi. Указатели.
11. **Линейные и разветвляющиеся алгоритмы: 1) оператор if, 2) оператор case.**
12. **Операторы для организации циклов. 1) for; 2) while; 3) repeat.**
13. **Процедуры и функции. Формальные и фактические параметры. Глобальные и локальные переменные. Параметры-значения и параметры-переменные.**
14. **Функции преобразования типов.**
15. Модули. Структура модуля.
16. Основные концепции объектно-ориентированного программирования. Инкапсуляция. Наследование. Полиморфизм.
17. Классы и объекты. Описание объектов в Delphi. Поля, свойства, методы .
18. События в Delphi.

# Procedures and functions

**Подпрограммой** называется именованная логически законченная группа операторов языка, которую можно вызвать для выполнения по имени любое количество раз из различных мест программы.

```
function < имя функции>(параметры):< тип результата>;  
    < раздел описаний>  
begin  
    < раздел операторов>  
end;
```

```
Function del(a,b,c:real):real;  
Var d,e:real;  
begin  
d:=- $(c-a)*(c-a)/(2.0*b*b)$ ;  
e:=20.0+d;  
if e<=0.0 then del:=0.0  
    else del:=exp(d)  
end;
```

```
var Fun,x,y,z:real ;  
  
Begin  
    x:=1; y:=2; z:=3;  
    Fun:=del (x,y,z);  
End;
```

```
Procedure < имя процедуры>(параметры);      <
раздел описаний>
    begin
        < раздел операторов>
    end;
```

**Глобальные** - константы, типы, переменные – это те, которые объявлены в головной программе.

**Локальные** – это константы, типы и переменные, существующие только внутри подпрограммы и объявленные либо в списке параметров, либо в соответствующих разделах блока описаний этой подпрограммы. При совпадении имен локальной и глобальной переменной сильнее оказывается локальное имя.

```
Function del(a,b,c:real):real;  
  Var x,y:real;  
  begin  
    x:=-(c-a)*(c-a)/(2.0*b*b);  
    y:=20.0;  
    if y<=0.0 then del:=0.0  
      else del:=exp(x)  
    end;  
end;
```

x,y,z – фактические  
параметры  
a,b,c – формальные  
параметры

```
var Fun,x,y,z:real ;  
  
Begin  
  x:=1; y:=2; z:=3;  
Fun:=del (x,y,z);  
End;
```

```
program test;  
  var Zn1,Zn2,Zn3,Zn4:real;  
  
  procedure Sum(a,b:real; var c:real);  
    begin c:=a+b; end;  
  
  function Proiz(a,b:real):real;  
  begin  
    Proiz:=a*b;  
  end;  
  
  Begin  
    Zn1:=5; Zn2:=7;  
    Sum(Zn1,Zn2,Zn3);  
    Zn4:=Proiz(Zn1,Zn2);  
  end.
```

Zn1,Zn2,Zn3,Zn4 – глобальные  
переменные  
a,b,c – локальные переменные

a,b,c – формальные параметры  
Zn1,Zn2,Zn3 – фактические  
параметры.

c – параметр - значение  
a,b – параметр-переменная

## Некоторые арифметические и математические функции

<code>Abs (x)</code>	вычисление абсолютной величины (модуля) числа $x$
<code>arctan (x)</code>	вычисление угла, тангенс которого равен $x$
<code>cos (x) , sin (x)</code>	вычисление косинуса и синуса $x$
<code>Exp (x)</code>	вычисление экспоненциальной функции
<code>frac (x)</code>	вычисление дробной части числа $x$
<code>int (x)</code>	вычисление целой части числа $x$
<code>ln (x)</code>	вычисление натурального логарифма $x$
<code>odd (I)</code>	возвращает <code>true</code> , если аргумент нечетное число
<code>pi</code>	возвращает значение числа
<code>random</code>	генерирует случайное число из диапазона 0..0.99. Тип результата вещественный
<code>Random (I)</code>	генерирует значение случайного числа из диапазона 0 . . I
<code>randomize</code>	процедура для загрузки новой базы в генератор случайных чисел
<code>Sqr (x)</code>	возведение в квадрат значения $x$
<code>sqrt (x)</code>	вычисление квадратного корня из $x$



## Функции преобразования типов

function <b>StrToInt</b> (s:string) :integer	преобразует строку <i>s</i> в целое число
function <b>IntToStr</b> (I:integer) :string	преобразует значение целочисленного выражения <i>I</i> в строку
function <b>StrToFloat</b> (s:string) :extended	преобразует строку <i>s</i> в вещественное число
function <b>FloatToStr</b> (x:extended) :string	преобразует значение вещественного выражения <i>x</i> в строку
function <b>FloatToStrF</b> (Value:Extended; Format: TFloatFormat; Precision, Digits:Integer) :string	преобразует значение вещественного выражения <i>x</i> в строку с учетом параметров <i>Precision</i> и <i>Digits</i>

<b>Format</b>	<b>форматы изображения числа</b>
ffExponent	научный формат
ffFixed	формат с десятичной точкой
ffGeneral	общий цифровой формат

## Процедуры и функции для работы со строковыми переменными

<pre>function <b>Copy</b>(s:string; index, count: integer):string</pre>	выделяет из строки <i>s</i> подстроку длиной <i>count</i> , начиная с символа в позиции <i>index</i>
<pre>function <b>Length</b>(s:string):integer</pre>	возвращает текущую длину строки <i>s</i>
<pre>function <b>Concat</b>(s1,s2,...,sn:string):string</pre>	возвращает строку, представляющую собой сцепление строк <i>s1, s2, ..., sn</i>
<pre>function <b>Pos</b>(s1,s2:string):integer</pre>	определяет первое появление в строке <i>s2</i> подстроки <i>s1</i> . Результат равен номеру позиции
<pre>procedure <b>Delete</b>(s:string;          poz, n:integer)</pre>	удаляет <i>n</i> символов строки <i>s</i> начиная с позиции <i>poz</i>
<pre>procedure <b>Insert</b>(s1,s2:string; poz:integer)</pre>	вставляет строку <i>s1</i> в строку <i>s2</i> , начиная с позиции <i>poz</i>

## Использование справки

### uses

```
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs, StdCtrls, ExtCtrls;
```

### begin

```
if (Edit1.Text <> '')  
  begin  
    a:=StrToInt(Edit1.Text);  
    b:=StrToInt(Edit2.Text);  
    Edit3.Text:='';  
    Case RadioGroup1.Position of  
      0: c:=a+b;  
      1: c:=a-b;  
      2: c:=b*a;  
      3: if b=0 then begin showMessage('На ноль делить нельзя!');  
          exit; end  
          else c:=b/a;  
    end;  
  end;
```

### begin

```
a:=StrToInt(Edit1.Text);
```

```
b:=StrToInt(Edit2.Text);
```

```
Edi func SysUtils.StrToInt: function(const S: String): Integer - sysutils.pas (4606)
```

```
Case RadioGroup1.Position of
```

```
0: c:=a+b;
```

```
begin showMessage('На ноль делить нельзя!');
```

```
exit; proc Dialogs.ShowMessage: procedure(const Msg: String) - Dialogs.pas (2122)
```

```
else c:=b/a;
```

# Рекурсия: Вызов подпрограммы самой подпрограммой

```
Function Fact(n:integer):integer;  
Begin  
  if n>=0 then Fact:=1 else Fact:=n*Fact(n-1);  
End;
```

При каждом вызове подпрограммы система сохраняет некоторые значения в **стеке** (**стек** – упорядоченный список, в котором элементы добавляются и удаляются с одного и того же конца списка). Если рекурсивная процедура вызывается много раз, она может заполнить весь стек и вызвать ошибку переполнения стека.

## **Для правильного определения области действия идентификаторов (переменных) необходимо придерживаться следующих правил:**

- ✓ каждая переменная должна быть описана перед тем, как она будет использована;
- ✓ областью действия переменной является та подпрограмма, в которой она будет описана;
- ✓ все переменные в подпрограммах должны быть уникальными;
- ✓ одна и та же переменная может быть по-разному определена в каждой из подпрограмм;
- ✓ если имя подпрограммы совпадает с названием стандартной подпрограммы, то последняя игнорируется, а выполняется подпрограмма пользователя;
- ✓ если внутри какой-либо процедуры встречается переменная с таким же именем, что и глобальная переменная, то внутри процедуры будет действовать локальное описание;
- ✓ каждая подпрограмма может изменить значение глобальной переменной.

# Модули

**unit** <имя модуля>;

**interface** {раздел интерфейса}

uses <список модулей>;

const <список констант>;

type <описание типов>;

var <объявление переменных>;

< заголовки процедур >

< заголовки функций >

**implementation** {раздел реализации}

uses <список модулей>

const <список констант>

type <описание типов>

var <объявление переменных>

< описание процедур >

< описание функций >

**initialization** {раздел инициализации}

<операторы>

**finalization** {раздел деинициализации}

<операторы>

**End.**

Модуль содержит переменные и подпрограммы, которые могут использоваться в других модулях и подпрограммах.

Есть пользовательские и встроенные модули.

Для каждой формы приложения создается отдельный модуль.

При компиляции модуля создается файл с расширением \*.DCU

В разделе **interface** размещаются описания идентификаторов, которые должны быть доступны всем модулям или программам, использующим данный модуль.

В разделе **implementation** располагается код подпрограмм, заголовки которых приведены в разделе **interface** . Можно указывать только названия подпрограмм, поскольку список параметров и тип результата функции указаны в разделе **interface** . В разделе **implementation** описываются типы и объявляются переменные. Которые используются только в данном модуле.

В разделе **initialization** располагаются инструкции, выполняемые в начале работы программы, которая подключает данный модуль. раздел **finalization** содержит

Delphi 6 - ProjectKP2\_1

File Edit Search View Project Run Component Database Tools Window Help <None>

Standard Additional Win32 System Data Access Data Controls dbExpress DataSnap BDE ADO InterBase WebServices InternetExpress Inte

Object TreeView

- Form1
  - Button1
  - Edit1
  - Edit2
  - Edit3
  - Label1
  - Label2
  - Label3

Mozilla Мой Stivenson... Interest

KP2\_1.pas KP2\_1

TFForm1  
Variables/Const  
Uses

KP2\_1

Значение x:   
 Значение y:   
 Счет:   
 Результат:

ants, Classes, Graphics, Contro

```

Edit1: TEdit;
Edit2: TEdit;
Edit3: TEdit;
Button1: TButton;
procedure Button1Click(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
end;

var
  Form1: TForm1;
  
```

12: 44 Insert \Code \Diagram

Object Inspector

Form1 TForm1

Properties Events

Action

ActiveContro

Align alNone

AlphaBlend False

AlphaBlend 255

▣ Anchors [akLeftakT

AutoScroll False

AutoSize False

BiDiMode bdLeftToRi

▣ BorderIcons [biSystemM

BorderStyle bsSingle

BorderWidth 0

Caption KP2\_1

ClientHeight 229

ClientWidth 342

Color  clBtnFac

▣ Constraints (TSizeCons

Ct3D True

Cursor crDefault

DefaultMoni dmActiveFc

DockSite False

All shown

old connections

Текстовый докумен...

unit KP2\_1;

## interface

### uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs, StdCtrls;

### type

```
TForm1 = class(TForm)
  Label1: TLabel;
  Label2: TLabel;
  Label3: TLabel;
  Edit1: TEdit;
  Edit2: TEdit;
  Edit3: TEdit;
  Button1: TButton;
  procedure Button1Click(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
end;
```



# Типы данных

Простые	Целочисленные, Литерные (символьные), Логические(булевы), Вещественные
структурные	Строки, Массивы, Множества, <b>Записи</b> , Файлы, строки
указатели	
процедурные	
вариантные	

# Переменные типа запись

**Записи** объединяют фиксированное число элементов данных других типов. Отдельные элементы записи имеют имена и называются **полями**. Различают фиксированные и варианты записи.

**Фиксированная запись** состоит из конечного числа полей

```
type <имя_типа>=record
  <идентификатор_поля_1>:<тип_поля_1>;
  ...
  <идентификатор_поля_n>:<тип_поля_n>;
end;
```

```
var <идентификатор>:<имя_типа>;
```

# Примеры записей

<pre>type Elements=record   N:integer;   KoordX:real;   KoordY:real;   KoordZ:real; end;  var Element:Elements;</pre>	<pre>Type Mans=record   Name:string;   Salary:real;   Bonus: real; End;  Var man1,man2,people:Mans</pre>
	<pre>Var man1:record   Name:string;   Salary:real;   Bonus: real; End;</pre>
<pre>... Element.N:=5; Element.KoordY:=10; Element.KoordX:=Element.KoordY;</pre>	<pre>... man1.Name:='Ivanov P.I.' man2.Salary:=2700000; people.Bonus:=3000000;</pre>

```
with <переменная_типа_запись> do
  begin
    <операторы>;
  end;
```

**Пример:** with Element do

```
  begin
    N:=5;
    KoordY:=10;
    KoordX:=KoordY;
  end;
```

**Вариантная запись**, так же как и фиксированная, имеет конечное число полей, однако позволяет по-разному интерпретировать области памяти, занимаемые полями.

```
type <имя_типа>=record
  <идентификатор_поля>:<тип_поля>;
case <поле_признака>:
  <имя_типа> of <Вариант_1>:(поле_1:тип_1);
               <Вариант_2>:(поле_2:тип_2);
end;var <идентификатор>:<имя_типа>;

var <идентификатор>:<имя_типа>;
```

```
type Elements=record
  N:integer
  case Flag:boolean of
    true:(usel1,usel2,usel3:integer);
    false:(usel1,usel2,usel3,usel4:real);
end;

var Element:Elements;
```

...

```
Element.Flag:=true;
  with element do begin
    usel1:=3;
    usel2:=4;
    usel3:=5
  end;
```

...

```
Element.Flag:=false;
  with element do begin
    usel1:=3.8; usel2:=4.2;
    usel3:=5.7; usel4:=5.7
  end;
```

...

```
Element.Flag:=true;
  Element.usel1:=3;
  Element.usel2:=4;
  Element.usel3:=5
```

...

```
Element.Flag:=false;
  Element.usel1:=3.8;
  Element.usel2:=4.2;
  Element.usel3:=5.7;
  Element.usel4:=5.7
```

<http://ge.tt/5mGYDhr>

<https://www.dropbox.com/sh/ex26edcc026h23c/AADkuFomJzyAYKceibd4o-5Aa?dl=0>