Файл Модуля

```
    unit Unit1;
interface //открытый интерфейс модуля
 {список подключаемых модулей}
uses
 Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;
    Стандартные модули, которые используются в данном модуле.

{объявление класса формы}
type TForm1 = class(TForm)
 Private //закрытый раздел класса
 { Private declarations }

{сюда могут помещаться объявления переменных, функций и процедур,
    включаемых в класс формы, но не доступных для других модулей}
```

Продолжение файла Модуля

```
public //открытый раздел класса
 { Public declarations }
{сюда могут помещаться объявления переменных, функций и процедур,
    включаемых в класс формы и доступных для других модулей }
 end;
var
 Form1: TForm1;
{сюда могут помещаться объявления типов, констант, переменных,
    функций и процедур, к которым будет доступ из других модулей, но
    которые не включаются в класс формы}
implementation
{$R *.DFM}
{сюда могут помещаться предложения uses, объявления типов, констант,
    переменных, к которым не будет доступа из других модулей. Тут же
    должны быть реализации всех объявленных в разделе interface
    функций и процедур, а также могут быть реализации любых
    дополнительных, не объявленных ранее функций и процедур}
end.
```

МОДУЛЬ

Заголовок программы, процедуры, функции

procedure <Объявление программной единицы>

{Раздел описаний}

Описания всех идентификаторов, используемых в разделе исполняемых операторов

BEGIN

{Раздел исполняемых операторов}

END;

Раздел описаний может включать объявления меток(**LABEL**), типов(**TYPE**), констант (**CONST**), переменных(**VAR**), процедур (**PROCEDURE**) и функций(**FUNCTION**).

Слова: **unit, uses, var, begin, end -** это так называемые ключевые слова языка.

Преобразование целых чисел в строку и обратно.

Для преобразования необходимо воспользоваться спец. функцией ***IntToStr*.** У неё только один аргумент – целое число, а на выходе она возвращает строку символов.

label1.Caption:=inttostr(r);

где **r** – целое число.

Для обратного преобразования строки в число используется функция ***StrToInt***.

При преобразовании строки в число, необходимо быть уверенным в том, что строка содержит число. Если в строке будет хоть один символ не относящейся к цифре, то во время преобразования произойдёт ошибка.

Преобразование вещественных чисел

Для преобразования вещественного числа в строку есть функция ***FloatToStr***, которой надо передать дробное число и получить строку.

Точно так же есть и обратное преобразование ***StrToFloat***, где передается строка, а получается вещественное число.

Форматировании строк возможно с помощью функции:

Format(const Format: string; const Args: array of string): string;

Функция из модуля **SysUtils** форматирует элементы открытого массива **Args** . В качестве результата функция возвращает отформатированную строку.

В качестве одного из параметров, в данных подпрограммах используется строка форматирования **Format.**

Общий вид спецификатора можно представить в следующем виде:

"%« [width] [".« prec] type

Спецификатор начинается с символа %.
За ним следуют:

необязательный параметр [width], задающий минимальную длину результирующей строки;

необязательный параметр ["." prec], задающий точность;

символ преобразования типа, type.

Идентификатор type может иметь одно из значений представленных в таблице:

d	Десятичный формат. Аргумент должен иметь целочисленное значение, которое будет преобразовано в строку символов десятичных цифр.
u	Десятичный беззнаковый формат. Форматируется аналогично параметру d, но знак числа не выводится.
e	Научный формат. Аргумент должен представлять собой число с плавающей запятой. Значение будет преобразовано в строку формата "-d.ddd...E+ddd".
f	Фиксированный формат. Аргумент должен быть числом с фиксированной десятичной точкой. Значение аргумента будет преобразовано в строку формата "-ddd.ddd...".
s	Строковый формат. Аргумент должен представлять собою символ, строку типа **string** или *PChar*.

Format('%9.2f', [12345.6789]);

Результирующей строкой будет

‘ 12345.68‘