РАЗДЕЛ 2

МОДУЛИ

УЧЕБНИК ПАВЛОВСКАЯ

ТЕМА МОДУЛЬНОЕ ПРОГРАММИРОВАНИЕ

Модуль – это подключаемая к программе библиотека ресурсов (описания типов, констант, переменных и подпрограмм. Обычно это связанные между собой ресурсы.)

Виды модулей:

1. Стандартные

2. Пользовательские (модуль программиста)

TPU - файл

ОСОБЕННОСТИ:

1. Для использования достаточно знать интерфейс (?), детали реализации скрыты от пользователя.
2. Программа становится более понятной
3. Возрастает скорость компиляции (?) (т.к. модули уже скомпилированы)
4. Преодолевается ограничение на объем кода исполняемой программы в один сегмент (?) памяти (код содержится в отдельном сегменте)

ОПИСАНИЕ МОДУЛЯ

UNIT имя; {заголовок модуля}

{интерфейсная секция}

INTERFACE {описание глобальных элементов модуля (видимых извне)}

{секция реализации}

IMPLEMENTATION {описание локальных (внутренних) элементов}

{секция инициализации}

Begin

End.

Исходный текст модуля хранится в отдельном файле с расширением **.pas**

Имя файла, в котором хранится модуль, должно совпадать с именем, заданным после ключевого слова unit.

НАЗНАЧЕНИЕ РАЗДЕЛОВ МОДУЛЯ

Модуль может использовать другие модули, для этого их надо перечислить в операторе uses, который может находиться только непосредственно после ключевых слов interface или implementation.

Интерфейсная секция:

- определяют константы, типы данных, переменные, а также **заголовки** процедур и функций.

Секция реализации

- описываются подпрограммы, заголовки которых приведены в интерфейсной части. Заголовок подпрограммы должен быть или идентичным указанному в секции интерфейса, или состоять только из ключевого слова procedure или function и имени подпрограммы. Для функции также указывается ее тип.
- также можно определять константы, типы данных, переменные и внутренние подпрограммы.

Секция инициализации предназначена для присваивания начальных значений переменным, которые используются в модуле. Операторы, расположенные в секции инициализации модуля, выполняются перед операторами основной программы.

ПРИМЕР МОДУЛЬ **INTLIB**

```
unit IntLib;
interface
        procedure ISwap(var i,J: integer);
        function IMax(I,J: integer): integer;
implementation
            procedure ISwap (var i,J: integer);
            var
            Temp: integer;
            begin
            Temp:=i;
            I:=J;
            J:=Temp
            end; {конец процедуры ISwap }
 function IMax(I,J: integer):
integer;
            begin
            if I > J then IMax:=I else
IMax:=J
            end; {конец функции
IMax }
End;
Begin
Writeln(“ПРИВЕТ!”)
end.

 {конец модуля IntLib }
```

ПРИМЕР

ОСНОВНАЯ ПРОГРАММА

```
program IntTest;
uses IntLib;
var
A,B: integer;
begin
    Write('Введите два целочисленных значения: ');
    Readln(A,B);
    ISwap(A,B);
    Writeln('A=',A,' B=',B);
    Writeln('Максимальное значение равно ',IMax(A,B));
end. {конец программы IntTest }
```

Для сохранения скомпилированного модуля на диске требуется установить значение пункта Destination меню Compile в значение Disk.

Компилятор создаст файл с расширением .tpu., который надо переместить в специальный каталог, путь к которому указан в пункте Directories в поле Unit Directories.

ТЕХНОЛОГИЯ СОЗДАНИЯ МОДУЛЯ

1. Создать файл модуля
2. Откомпилировать его (Compile – Destination установить Disk)
3. Подключить в нужной программе (Uses имя модуля)

ПРИМЕР МОДУЛЯ (ОПИСАНИЕ)


```
Unit Average;
Interface
    Const N=10;
    Type mas = array [1..N] of real;
    Procedure Average(X : mas, var av: real);
Implementation
    Procedure Average(X : mas, var av: real);
        Var i : integer;
            begin
             av := 0;
             for i:=1 to n do av := av+x[i];
             av:=av/n;
            end;
end.
```

ПРИМЕР МОДУЛЯ (ИСПОЛЬЗОВАНИЕ)

```
Program Div_Average;
Uses Average;
Var  a,b :mas;
       i:integer;
Dif, av_a, av_b : real;
Begin
     for i:=1 to n do read (a[i]);
     for i:=1 to n do read (b[i]);
     average(a, av_a);
     average(b, av_b);
      dif:= av_a- av_b;
Writeln (‘Разность значений ’, dif:6:2)
end.
```

После этого все описания, расположенные в интерфейсных секциях модулей, становятся доступными в программе и ими можно пользоваться так же, как и величинами, определенными в ней непосредственною

Поиск модуля:

1. В библиотеке исполняющей среды;
2. В текущем каталоге;
3. В каталогах заданных в **Option/Directories**

Если описаны две величины с одинаковыми именами (в модуле и в программе), то обращение к той что из модуля: имя модуля.имя величины

СТАНДАРТНЫЕ МОДУЛИ

System содержит все стандартные и встроенные процедуры и функции. Любая подпрограмма, не являющаяся частью стандартного Паскаля и не находящаяся ни в каком другом модуле, содержится в System. Этот модуль присоединяется ко всем программам.

CRT обеспечивает набор специфичных для описаний констант, переменных и программ для операций ввода/вывода. Последние можно использовать для работы с экраном (задание окон, непосредственное управление курсором, цвет текста и фона). Можно осуществлять "необработанный" ввод с клавиатуры и управлять платой генерации звукового сигнала персонального компьютера. Этот модуль обеспечивает множество подпрограмм, которые были стандартными в версии 3.0.

DOS определяет многочисленные паскалевские процедуры и функции, которые эквивалентны наиболее часто используемым вызовам DOS, как например, GetTime, SetTime, DiskSize и так далее. Кроме того, он определяет две программы низкого уровня MsDos и Intr, которые позволяют активизировать любой вызов MS-DOS или системное прерывание. Registers представляет собой тип данных для параметра в MsDos и Intr. Кроме того, определяются некоторые другие константы и типы данных.

Graph обеспечивает набор быстродействующих, эффективных графических подпрограмм, которые позволяют использовать в полной мере графические возможности компьютера.

Strings позволяет работать с ASCIIZ-строками (последний байт строки содержит символ с кодом 0). Введение таких строк связано с необходимостью совместить программы, написанные в Turbo Pascal, с программами, использующими среду Windows, а также для установления соответствия с другими языками (например, Си, ассемблер и т. д.). Подпрограммы этого модуля позволяют манипулировать с такими строками, а также преобразовывать их в строки типа string, и наоборот.

ПРИМЕР ИСПОЛЬЗОВАНИЯ МОДУЛЯ **CRT**

```
Программа "Угадай число"
program luck;
uses crt;
const max = 10;
var i, k, n : integer;
begin
   clrscr;                          { очистить экран }
   randomize;
   i := random(max);                { загадать число }
   window(20, 5, 60, 20);           { определить окно }
   TextBackGround(Blue);            { цвет фона – синий }
   clrscr;                          { залить окно фоном }
```

ПРИМЕР ИСПОЛЬЗОВАНИЯ МОДУЛЯ **CRT**

```
Программа "Угадай число"
TextColor(LightGray);                { цвет символов – серый}
   k := –1;                          { счетчик попыток }
   GotoXY(12, 5); writeln(' Введите число : ');
   repeat                            { цикл ввода ответа }
      GotoXY(20, 9);                    { установить курсор }
      readln(n);                        { ввести число }
      inc(k);
   until i = n;
```

ПРИМЕР ИСПОЛЬЗОВАНИЯ МОДУЛЯ **CRT**

```
Программа "Угадай число“_продолжение
window(20, 22, 60, 24);          { определить окно результата }
   TextAttr := 2 shl 4 + 14;     { желтые символы за зеленом фоне }
   clrscr;                        { залить окно фоном }
   GotoXY(6, 2);                      { установить курсор }
   writeln(' Коэффициент невезучести : ', k / max :5:1);
   readkey;                     { ждать нажатия любой клавиши }
   TextAttr := 15;               { белые символы на черном фоне }
   clrscr;                     { очистить после себя экран }
end.
```