Модули

Pascal

Pascal

Модуль – это подключаемая к программе библиотека ресурсов. Он может содержать описания типов, констант, переменных и подпрограмм (одна или несколько процедур или функций). В модуле обычно соединяют связанные между собой ресурсы. Модуль – это инструмент для разработки библиотек прикладных программ и средство модульного программирования.

Модуль – автономно компилируемая программная единица размещаемая в отдельном сегменте памяти (максимальная длина сегмента для одного модуля – 64К, но количество одновременно используемых в одной программе модулей ограничено только доступной памятью; таким образом можно создавать большие программы). Исходный текст модуля размещается в отдельном файле <имя>.pas. Результат компиляции модуля храниться на диске в отдельном файле <имя>.tpu. В оболочке Borland Pascal результат компиляции модуля по умолчанию размещается на диске в той же папке (директории) где находится исходный модуль. Компиляция модуля в Borland Pascal может осуществляться в одном из трёх режимов: Compile, Make, Build. Режимы отличаются подходом к отбору типов файлов и способом связи компилируемого модуля или основной программы с другими модулями, объявленными в Uses. В режиме Compile программист сам следит за наличием необходимых TPU файлов; в режиме Make проверяет наличие объявленных модулей, а также ищет исходные тексты модулей и если они изменились, компилирует их и подключает новые версии; в режиме Build существующие TPU файлы игнорируются и система ищет и компилирует соответствующий PAS файл для каждого объявленного в Uses модуля.

Для подключения модулей к программе используется объявление:

Uses <модуль1>[, <модуль2>, …,<модульN>];

Оно размещается в начале раздела описаний основной программы. Uses также может использоваться внутри модуля для подключения к нему других модулей.

В Borland Pascal файлы *.tpu будут доступны, если путь к их размещению указан в настойках directiries.

Модули

Pascal

Pascal

Структура (формат описания) модуля:

```
Unit <имя_модуля >        { Заголовок модуля }
     { Имя_модуля должно совпадать с именем файла, содержащего модуль }
Interface                 { Интерфейсная секция  модуля }
     { содержит описание глобальных элементов модуля (видимых извне) }
  Implementation          { Секция реализации модуля (исполняемая часть) }
     { Описание локальных (внутренних) элементов модуля }
  Begin                   { Начало секции инициализации }
     { Описание элементов инициализации }
End.                      { Конец модуля }
```

Модуль, как видно, состоит и заголовка и трёх составных частей, любая может быть пустой, *но ключевые слова – присутствуют*.

Interface – определяет глобальные константы, типы данных, переменные, заголовки процедур и функций. В начале секции должно быть Uses (если требуется), затем типы данных, константы, переменные (причем не только те, которые используются в этом модуле – память для них выделяется в общем сегменте данных -64К- с глобальными данными основной программы), затем заголовки процедур и функций, описанных в этом модуле в секции реализации.

Implementation – описывает подпрограммы, заголовки которых объявлены в секции interface, список формальных параметров и тип (для функции) можно опускать (т.к. они уже описаны выше). В начале этой секции (до подпрограмм) могут быть описания Uses, а также объявления локальных для модуля объектов (вспомогательные типы, константы, переменные и блоки, метки (если они исп. в инициирующем сегменте). Память для всех локальных данных выделяется в общем сегменте памяти.

Begin – используется для присваивания начальных значений. Операторы, расположенные в этой секции, выполняются перед операторами основной программы (когда объявлено несколько модулей – секции инициализации выполняются в порядке их описания в Uses). В секции инициализации считаются известными внешние объявления интерфейсной секции, внутренние объявления секции реализации и внешние объявления всех модулей из Uses этого модуля.

Модули

Пример простого модуля

Pascal

Pascal

```
unit ModulEx;
(* Простой пример модуля Pascal.
        Программа вычисления среднего арифметического значения элементов массива *)

interface
    const n = 10; (*число элементов массива*)
  type mas = array[1..n] of real;       (* тип массива *)
  procedure average (x: mas; var av: real);
implementation
   procedure average (x: mas; var av: real)
    var i: integer;   (* счетчик цикла for  *)
    begin
       av := 0; (* инициализация перемен-ной результата *)
       for i:=1 to n do av:=av+x[i];   (*  цикл суммирования элементов массива *)
  av := av / n; (* вычисление среднего *)
    end;
end.
```

```
Program ExampleModul;
(* Пример использования модуля ModulEx. Программа находит разность средних арифметических значений двух массивов  *)

uses ModulEx;

var a, b : mas;   (* имена массивов *)
   i : integer;   (* переменная циклов *)
   dif, av_a, av_b : real; (* dif - результат, av_a - среднее массива a,  av_b - среднее массива b *)
Begin
WriteLn (' Введите 10 значений массива A, разделяя их Enter ');
  for i:=1 to n do read (a[i]);
WriteLn (' Введите 10 значений массива B, разделяя их Enter ');
  for i:=1 to n do read (b[i]);
   average (a, av_a);  (* Вычисление среднего значения элементов массива a *)
   average (b, av_b);   (* Вычисление среднего значения элементов массива b *)
  dif := av_a - av_b;  (* Вычисление разности средних двух массивов *)
  WriteLn (' Разность значений ', dif:6:2);
End.
```

Pascal

Модули

Пример более сложного модуля:

Арифметика комплексных чисел

Pascal

```
unit ModCmplx;
(* Модуль арифметики комплексных чисел *)
              Interface
type  complex = record  re : real; im : real   end;
  procedure AddC (x,y: complex; var z: complex);
  procedure SubC (x,y: complex; var z: complex);
  procedure MulC (x,y: complex; var z: complex);
  procedure DivC (x,y: complex; var z: complex);
 const        c: complex = (re:0.1; im: -1);
                      Implementation
procedure AddC;
begin        z.re := x.re + y.re;   z.im := x.im + y.im;
    end;  {AddC}
{*****************************}
   procedure SubC;
begin        z.re := x.re - y.re;     z.im := x.im - y.im;
    end; {SubC}
{*****************************}
   procedure MulC;
begin
     z.re := x.re * y.re - x.im * y.im;
     z.im := x.re * y.im + x.im * y.re;
    end; {MulC}
{*****************************}
   procedure DivC;
var zz : real; begin
     zz := sqr (y.re) + sqr (y.im);
     z.re := (x.re * y.re + x.im * y.im) / zz;
     z.im := (x.re * y.im - x.im * y.re) / zz;
    end; {DivC}
end.
```

```
Program mCmplx;
 Uses ModCmplx;
var
 a,b,c : complex;

begin
   a.re := 1;   a.im := 1;
   b.re := 1;   b.im := 2;

 AddC (A,b,c);
writeln (' Сложение: ', c.re:5:1, c.im:5:1, 'i');
 SubC (a,b,c);
writeln (' Вычитание: ', c.re:5:1, c.im:5:1, 'i');
 MulC (a,b,c);
writeln (' Умножение: ', c.re:5:1, c.im:5:1, 'i');
 DivC (a,b,c);
writeln (' Деление : ', c.re:5:1, c.im:5:1, 'i');
end.
```

Pascal

Модули

Стандартные модули

Pascal

В Borlan Pascal есть стандартные модули, содержащие разнообразные типы, константы, процедуры и функции.

Это: System, Crt, Dos, WinDos, Printer, Graph, Overlay, Strings и для совместимости со старыми версиями Turbo Pascal - Turbo3, Grapf3.

Модули GRAPH, TURBO3, GRAPF3 выделены в отдельные TPU файлы, остальные входят в состав библиотечного файла TURBO.TPL.

Модуль SYSTEM подключается к основной программе автоматически, остальные надо подключать в Uses.

System – содержит базовые средства Pascal (ввод/вывод, работа со строками, с плавающей точкой, динамическое распределения памяти), а также те, которые не вошли в другие стандартные модули.

Crt – процедуры и функции управления текстовым режимом работы экрана, процедуры "слепого" чтения с клавиатуры и управление звуком.

Dos – процедуры и функции доступа к средствам работы с файлами и дисками MS DOS, системным временем, прерываниями и др.

WinDos – подпрограммы аналогичные модулю Dos, но для работы с ASCIIZ строками (строками с завершающим нулём).

Printer – делает доступным вывод на матричный принтер.

Graph –процедуры и функции для управления графическим режимом работы экрана (более 50 подпрограмм).

Overlay –процедуры и функции для организации работы с большими программами (с перекрытием).

Strings – процедуры и функции для работы со строками с завершающим нулём (ASCIIZ).

C/C++

Библиотеки С

C/C++

Библиотека С – это подключаемая к головной программе библиотека ресурсов в виде одного или нескольких **Заголовочных файлов**. Это файлы с расширением **.H**, которые включается в программу с помощью **директивы препроцессора #include**. **Заголовочные файлы представляют собой файлы в формате ASCII.**

В заголовочном файле могут содержаться:

- **Определения типов - struct point { int x, y; }**
- **Описания функций - extern int strlen(const char*);**
- **Определения inline-функций - inline char get() { return *p++; }**
- **Описания данных - extern int a;**
- **Определения констант - const float pi = 3.141593**
- **Перечисления - enum bool { false, true };**
- **Другие директивы include - #include**
- **Определения макросов - #define Case break; case**
- **Комментарии - /* проверка на конец файла */**
- **и др. элементы программ на С.**

Директива #include включает в программу содержимое указанного файла. Имя файла может быть указано двумя способами:

#include <some_file.h>
#include "my_file.h"

Если имя файла заключено в угловые скобки (<>), то это означает, что подключается стандартный заголовочный файл, и компилятор ищет этот файл в заданных в настройках местах.

Двойные кавычки (") означают, что заголовочный файл – пользовательский, и компилятор ищет его в том каталоге, где находится исходный текст программы.

Заголовочный файл также может содержать вложенные директивы #include.

Библиотеки С

C/C++

Файл LibEx.h

```
/* Простой пример пользовательского заголовочного файла С из 3-х функций */

#include <stdio.h>
#include <math.h>

int F1 (int a, int b, int c)
//  считает произведение трех целых чисел
{
  return (a*b*c);
}

float  F2 (int  d)
//  вычисляет корень  квадратный числа
{
return (pow(d,0.5));
}

void F3 (float e)
// выводит на экран вещественное число
{
printf ("\nчисло -> %6.3f\n", e);
}
```

```
/*  Программа последовательно вычисляет произведение 3-х целых чисел, затем корень квадратный из этого произведения и затем выводит на печать результат  */

#include <conio.h>
#include "LibEx.h"

main ()
{
   int a,b,c;   clrscr ();
          printf   ("Введите через пробел 3-и вещественных числа и нажмите  Enter\n");
   scanf ("%d %d %d", &a, &b, &c);

   F3(F2(F1(a,b,c)));
  getch ();
  return 0;
}
```