Модульное программирование

Модули

Модуль - **автономно компилируемая программная единица, включающая, компоненты раздела описаний и, возможно некоторые исполняемые операторы в инициирующей части.**

интерфейсная часть

UNIT <ИмяМодуля>; **{файл MODUL1.PAS}**
INTERFACE
<интерфейсная часть> {раздел объявлений}
IMPLEMENTATION
<исполняемая часть> {описание подпрограмм}
BEGIN
<инициирующая часть> {фрагмент программы}
END. {признак конца модуля}

Раздел **объявлений** **INTERFACE**
Uses после **INTERFACE**

```
Unit Имя модуля;
INTERFACE                    {Начало раздела объявлений}
  Uses   Имя1, Имя2, ..;    {Используемые при объявлениях модули}
  Const                      {Блок объявления библиотечных констант}
  Type                       {Блок объявления библиотечных типов }
  Var                        {Блок объявления библиотечных
     переменных}
Заголовки библиотечных процедур и (или) функций

      Раздел реализации IMPLEMENTATION
                        (Forward)
IMPLEMENTATION
Uses   Имя101, Имя102, ..;  {Используемые при реализации модули}
  Const                     {Блок объявления внутренних констант}
  Type                      {Блок объявления внутренних типов }
  Var                       {Блок объявления внутренних переменных}
  Label                     {Блок описания меток блока инициализации }
Описание библиотечных процедур и (или) функций
          Раздел инициализации
     Begin
          Блок инициализации модуля
     End.
```

USES <список имен модулей через запятую>;
например: USES MODUL1, CRT, GRAPH;
подключает три модуля.

Пример 1. Для задания лаб.раб. 14_1 (по лр1) программа с дополнительным модулем (для вычисления суммы чисел) имеет вид:

```
Program mod14a; {модуль aa подключается} UNIT aa;
uses aa;
var
  a,b,s:integer;
begin
       writeln('Введи 2 целых числа');
    read(a,b);
      writeln('s=',ma(a,b));
       writeln('s=',ma(a,a));
     readln;
end.
```

```
INTERFACE
function ma(x,y:integer):integer;
IMPLEMENTATION
function ma(x,y:integer):integer;
    begin
          ma:=x+y;
    end;
begin
END.
```

Компиляция модулей

Три режима компиляции COMPILE, MAKE, BUILD.

USES

В режиме **COMPILE** USES .TPU

В режиме **MAKE** TPU-файлов .PAS

и если в PAS-файл внесены изменения, то перекомпиляция

В режиме **BUILD** TPU-файлы игнорируются, отыскивается PAS-файл

Достоинства применения модулей.

1) Построение собственных библиотек

2) Возможность создавать программы практически любого размера. (Один модуль - не более 64К, сумма ограничена емкостью ОЗУ ПК).

Пример2. Найти сумму максимальных элементов массивов A и B. Подпрограмму скрыть в модуле.

```
Program mod17_; { подключается модуль rr }
  uses crt,rrr;
   var   a,b:mass;
         i,s:integer;
     begin
       clrscr;
        writeln('a');
        for i:=1 to 8 do
            read(a[i]);
         writeln('b');

            read(b[i]);
    s:=maxim(a)+maxim(b);
        writeln('Sum=',s);
           readln;
                end.
```

```
Unit rrr;
  INTERFACE
 type mass=array[1..8] of integer;
function maxim(d:mass):integer;
  IMPLEMENTATION
 function maxim(d:mass):integer;
 var mx,i:integer;
      begin
    {MaxLongInt=2147483647}
 mx:=-MaxInt;
    {MaxInt=32767}
          for i:=1 to 8 do
               if mx<d[i] then
            mx:=d[i];
maxim:=mx;
      end;
      BEGIN {можно опустить}
       END.
```

Дома: для лаб.раб.14_1(по лр1 см.пр.2), для 14_2 и 14_3 в 8_3, 9_2 скрыть подпрограммы в модулях

СТАНДАРТНЫЕ БИБЛИОТЕЧНЫЕ МОДУЛИ *(*ТП *7.0)*

SYSTEM (подключается автоматически);

ADDR(x):pointer – возвращает адрес заданного объекта

ChDir(S: string) Устанавливает текущий каталог

DOS позволяет открыть доступ к средствам дисковой операционной системы MSDOS;

GetDate – возвращает текущую дату

GetTime - возвращает текущее время

SetDate - устанавливает текущую дату

SetTime – устанавливает текущее время

SetFTime – устанавливает время и дату последнего обновления

DiskFree - возвращает число свободных байт на указанном диске

DiskSize - возвращает число полный объем указанного диске в байтах

Exec(Name;PathStr, CmdLine:string) - выполняет заданную программу ***Name*** с указанной командной строкой ***CmdLine***

PRINTER содержит переменную lst;

OVERLAY для организации оверлейных программ;

CRT работы с клавиатурой и дисплеем;

Graph графические процедуры и функции;

Turbo3 и Graph3 для связи с программами и графикой TP 7.0 _

новый тип *PChar*, определяемый как указатель на символ **PChar =^Char;**

STRINGs содержит функции обработки строк (только в версии 7.0);

Некоторые ***STRINGS функции:***

StrComp(S1, S2:Pchar *):integer* - сравнение двух строк;

StrlComp(S1 , S2:Pchar):integer - сравнение 2-х строк без различия между прописной и строчной латинскими буквами;

StrLComp(s1,s2:Pchar; MaxLen:word): Pchar - сравнение заданного числа символов 2-х строк;

StrLower(S:Pchar): Pchar - преобразование в строке прописных латинских букв в строчные;

StrNew(S:Pchar): Pchar размещение строки в динамической области. Возвращает указатель на строку;

StrDispose(Str: PChar); удаляет динамич. Строку из кучи

Для реализации операций над *ASCIIZ*— строками в язык введен новый тип *PChar*, определяемый как указатель на символ **PChar =^Char;**

StrPas(S:Pchar): String - преобразование ASCIIZ строки в строку типа string;

StrPCopy(D:Pchar, S: String) - процедура преобразования строки S в ASCIIZ - строку D;

Объединяет строки. Function **StrCat(D, S: PChar): PChar;**

Function StrCopy (D, S: PChar) : PChar; Копирует строку *S* в строку *D* и возвращает указатель на *D*

и др. См. Фаронов В.В. Турбо Паскаль 7.0.

WinDos содержит :

1) Процедуры и функции управления вычислительным процессом
2) **Intr(IntNo:byte; var Regs:TRegisters**) - программное прерывание

GetArgStr - получение параметра командной строки,

а также

2) Процедуры и функции работы с каталогами;.
3) Процедуры и функции работы с файлами и дисками;
4) Процедуры и функции работы с датой и временем;
5) Разные.

Процедурные типы

*пример:*TYPE func = Funtion (x:real):real;

```
        Proc1=Procedure(a,b,c:real;var d:real);
        Proc2=Procedure(var a,b:byte);
        Proc3=Procedure;
        Func1=Function:string;
        Func2=Function(s:string):real;
 VAR
   p:proc1;
  f1:func2;
```

директива FAR

пример:

```
 TYPE
   Func=Function(x:real):real;
   Function F(x:real):real;FAR;   {Далеко}
      Begin
   . . .
     end;
 Procedure FF(n:byte;F:Func);
   Var ...;
     Begin
  . . .
     end;
      Begin
  . . .
        FF(1,F1);...FF(5,F1);
      End.
Пр.    FUNCTION  SINUS (x:real):real;
          FAR;
             Begin
                 SINUS:=sin(x)
             end;
```

Пример:

```
Program typefun;
  var x,y:real;
  type fun=function(x,y,n,t:real):real;
Function fe(x1,y1,n,t:real):real;far;
  begin
        fe:=n*x1+t*y1
  End;
Function sim1(var f:fun;x,y,n,t:real):real;
  begin
           sim1:=f(x,y,n,t)*x+f(x,y,n,t)/y
  end;
  function func(fe:fun;t,n:real);
  var x1,y1:real;
  begin
     x1:=1; y1:=2;
     func:=2*sim1(fe,x1,y1,t,n)
  end;
  begin
     read(x,y);
     writeln('t=',func(fe,x,y):8:3)
  End.
```

```
11
9
t=87
50
30
t=330.00
```

Результаты:

```
3
4
t=33.00
11
9
t=87
50
30
t=330.00
```