# Организация библиотек. Стандартные библиотечные модули и модули пользователя

Лекция 11

# *Понятие модуля

*Библиотечный модуль это* отдельно компилируемая программная единица, содержащая различные элементы раздела описаний (типы, константы, переменные, процедуры и функции) и, возможно, некоторые операторы.

Хранится модуль как в исходном, так и в откомпилированном виде (файлы с расширениями pas и tpu соответственно).

В модуле явным образом выделяется некоторая «видимая» интерфейсная часть, в которой сконцентрированы описания глобальных типов, констант и переменных, а также приводятся заголовки глобальных процедур и функций. Появление объектов в интерфейсной части делает их доступными для других модулей и основной программы. Тела процедур и функций располагаются в исполняемой части модуля, которая может быть скрыта от пользователя.

По ходу работы любой программист обычно накапливает для себя целую коллекцию таких полезных модулей — свою личную библиотеку, это позволяет ему писать гораздо меньше кода для новых программ, ведь он может *многократно использовать свои старые разработки*.

В этом и заключается один из наиболее фундаментальных принципов современного программирования — *принцип модульности*.

На практике реализация принципа модульности предполагает выполнение двух условий:

- при разработке программы необходимо выделять подпрограммы таким образом, чтобы можно было воспользоваться уже имеющимися модулями;
- если в личной библиотеке программиста не находится подходящих подпрограмм для решения какой-либо задачи, то разработку новых процедур или функций следует выполнять таким образом, чтобы полученные модули можно было использовать в качестве составных частей для решения других задач.

Таким образом, принцип модульности отлично дополняет структурный подход к разработке программ, позволяя *многократно* использовать разработанные и отлаженные модули.

# *Преимущества модульного программирования

- Благодаря использованию модулей можно *многократно использовать свои старые разработки*.
- Модули, в отличие от процедур и функций, включаемых в исходный код программы, могут храниться на диске в откомпилированном виде. В этом случае процесс подготовки программы к выполнению займет меньше времени, т. к. компилироваться будет только основная программа, а код из модулей будет подключаться на этапе компоновки.
- Еще одно немаловажное обстоятельство — при разработке больших программ отдельные модули могут разрабатываться различными программистами, т. к. это относительно автономные программные единицы.
- Для языка Pascal уже накоплено большое количество

# Структура модуля

**UNIT ИмяМодуля;**

**INTERFACE**

**{интерфейсная часть}**

**IMPLEMENTATION**

**{исполняемая часть}**

**BEGIN**

**{инициирующая часть}**

**END.**

Заголовок и интерфейсная часть задают название модуля и перечисление всех программных элементов, которые предоставляет данный модуль. Программный код располагается в исполняемой части, иногда в инициирующей части.

Напишем модуль, который будет содержать две простые математические функции $sign(x) = \begin{cases} 1, x > 0 \\ 0, x = 0 \\ -1, x < 0 \end{cases}$ и $(-1)^n$.

**Заголовок модуля** состоит из зарезервированного слова *UNIT* и следующего за ним имени модуля. **ИмяМодуля** должно совпадать с именем дискового файла, в который помещается исходный текст модуля.

```
unit raznoe;
```

**Интерфейсная часть** открывается зарезервированным словом *INTERFACE*. В этой части содержатся объявления всех глобальных объектов модуля (типов, констант, переменных и подпрограмм), которые должны стать доступными основной программе и другим модулям. При объявлении глобальных подпрограмм в интерфейсной части указывается только их заголовок.

```
interface
    function sign(x:real):integer;
    function minusvst(n:integer):integer;
```

**Исполняемая часть** начинается зарезервированным словом ***IMPLEMENTATION*** и содержит описания подпрограмм, объявленных в интерфейсной части. В ней могут объявляться локальные для модуля объекты - вспомогательные типы, константы, переменные и т.д.

Описанию подпрограммы, объявленной в интерфейсной части модуля, в исполняемой части должен предшествовать заголовок, в котором можно опускать список формальных параметров (и тип результата для функции), так как они уже описаны в интерфейсной части. Но если заголовок подпрограммы приводится в полном виде, т.е. со списком формальных параметров и результата, он должен совпадать с заголовком, объявленным в интерфейсной части, например:

```
implementation
  function sign;
    begin
      if x>0 then sign:=1
            else if x<0 then sign:=-1
                        else sign:=0;
    end;
  function minusvst(n:integer):integer;
    begin
      if n mod 2=0 then minusvst:=1
                   else minusvst:=-1;
  end;
```

**Инициирующая часть** завершает модуль. Она может отсутствовать вместе с начинающим ее словом *BEGIN* или быть пустой (лучше так не делать!).

End.

В инициирующей части размещаются исполняемые операторы, содержащие некоторый фрагмент программы. Это операторы передачи управления основной программе и обычно используются для подготовки ее работы. Например, в них могут инициироваться переменные, открываться нужные файлы и т.п.

Для того, чтобы подключить модуль к программе необходимо в разделе описания модулей записать его имя, если необходимо подключить несколько модулей, то их имена указываются через запятую.

```
uses raznoe;
```

Запишем модуль целиком и программу, его использующую.

```
unit raznoe;
 interface
   function sign(x:real):integer;
   function minusvst(n:integer):integer;
 implementation
   function sign(x:real):integer;
     begin
       if x>0 then sign:=1
                 else if x<0 then sign:=-1
                                  else sign:=0;
     end;
   function minusvst(n:integer):integer;
     begin
```

```
if n mod 2=0 then minusvst:=1
                  else minusvst:=-1;
    end;
  end.
Program uuu;
uses raznoe;
var x:real; k,s:integer;
begin
  writeln ('Vvedite x,n');
  readln (x,n);
  S:=sign(x);
  K:=minusvst(n);
  writeln(s:5,k:5);
  readln; end.
```

# *Стандартные модули

В Турбо Паскале имеется восемь стандартных модулей, в которых содержится большое число разнообразных типов, констант, процедур и функций. Этими модулями являются ***SYSTEM***, ***DOS***, ***CRT***, ***PRINTER***, ***GRAPH***, ***OVERLAY***, ***TURBO3*** и ***GRAPH3***.

Лишь один модуль ***SYSTEM*** подключается к любой программе автоматически, все остальные становятся доступны только после указания их имен в списке, следующем за словом ***USES***.

Модуль Паскаля PRINTER делает доступным вывод текстов на принтер.

Модуль Паскаля CRT. В нем сосредоточены процедуры и функции, обеспечивающие управление текстовым режимом работы экрана. С его помощью можно перемещать курсор в любую точку экрана, менять цвет выводимых символов и фона, создавать окна. Кроме того, в данный модуль включены также процедуры «слепого» чтения клавиатуры и управления звуком.

Модуль Паскаля GRAPH . Содержит набор типов, констант, процедур и функций для управления графическим режимом работы экрана. Этот модуль позволяет создавать различные графические изображения.

Модуль Паскаля DOS . В модуле собраны процедуры и функции, открывающие доступ к средствам операционной системы MS - DOS .

Модуль Паскаля OVERLAY . Данный модуль необходим при разработке громоздких программ с перекрытиями. Турбо Паскаль обеспечивает создание программ, длина которых ограничивается лишь основной оперативной памятью. Операционная система MS - DOS оставляет программе около 580 Кбайт основной памяти, использование программ с перекрытиями снимает это ограничение.

Модули Паскаля TURBO 3 и GRAPH 3 введены для обеспечения совместимости с ранней версией системы Турбо Паскаль.

# *Область видимости

Глобальные переменные, описанные в *исполняемой части* модуля, являются на самом деле не глобальными, а *статическими переменными*. Они видимы только *во всех подпрограммах данного модуля*, но *недоступны основной программе*.

Переменные, объявленными в интерфейсной части модуля — полноценные *глобальные переменные*, имеющие глобальную видимость и время жизни. Они располагаются в той же области памяти, где и глобальные переменные программы, и используются без всякого предварительного описания.

Если и в модуле, и в программе, к которой подключен модуль, объявить переменные или другие объекты с *одинаковыми именами*, то соответствующий объект из модуля станет невидим программе. Для обращения к объекту из модуля можно воспользоваться составным именем **ИмяМодуля.ИмяОбъекта.**

Глобальные переменные основной программы нельзя использовать в модулях, которые к ней подключены.

# *Домашнее задание

1. Составить опорный конспект лекции по теме «Организация библиотек. Стандартные библиотечные модули и модули пользователя, структура Unit'a» на основе презентации.

2. Программирование на языке Pascal. Рапаков Г. Г., Ржеуцкая С. Ю. СПб.: БХВ-Петербург, 2004, стр. 203-216.

3. Составить программы с использованием библиотечных модулей:

- Найти сумму ряда $S = 1 + \frac{2^n}{2!} + \frac{3^n}{3!} + \cdots + \frac{n^n}{n!}$.
- В исходном файле размерность одномерного массива и его целые элементы. Заменить все положительные элементы массива суммой их цифр.