***Модульное программирование***

Модуль (unit) представляет собой набор констант, типов данных, переменных, процедур и функций. Каждый модуль аналогичен отдельной программе на Паскале: он может иметь основное тело, которое вызывается перед запуском Вашей программы и осуществляет необходимую инициализацию. Короче говоря, модуль представляет собой библиотеку описаний, которую можно вставить в свою программу и которая позволит разбить программу на части, компилируемые отдельно.

**Структура модуля**

Модуль обеспечивает набор средств благодаря процедурам и функциям при поддержке констант, типов данных и переменных, однако действительная реализация этих средств скрыта в силу того, что модуль разделен на две секции: интерфейса и реализации. Если программа использует модуль, то все описания модуля становятся доступными этой программе, как если бы они были определены в ней самой.

Структура модуля аналогична структуре программы, однако есть несколько существенных различий. Например, рассмотрим модуль:

**unit** <идентификатор>;

**interface**

**uses** <список модулей>; *{Необязательный}*

*{открытые описания}*

**implementation**

*{закрытые описания процедуры и функции }*

**begin**

*{код инициализации }*

**end**.

Заголовок модуля начинается зарезервированным словом unit, за которым следует имя модуля (идентификатор) точно так же, как и в случае имени программы. Следующим элементом в модуле является ключевое слово interface. Оно обозначает начало секции интерфейса модуля - секции, видимой всем другим модулям или программам, в которых он используется.

Модуль может использовать другие модули, для этого они определяются в предложении uses. Предложение uses, если имеет место, то следует сразу после ключевого слова interface. Отметим, что здесь выполняется общее правило использования предложения uses: если модуль, имя которого указано в предложении uses, использует другие модули, то имена этих модулей также должны быть указаны в предложении uses, причем до того, как они будут использованы.

**Интерфейсная часть**

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

**Пример фрагмента программы**

**Unit** complexn;  
**Interface** **Type** Complex= **record**  
   Re, im: real;  
**End**;  
**Procedure** **AddC**(x,y: complex, **var** z: complex);  
**Procedure** **MulC** (x,y: complex, **var** z: complex);

Если теперь в основной программе написать предложение

**Uses** complexn ;

то в программе станут доступными тип cmplex и две процедуры – AddC и MulC из модуля complexn .

Отметим, что объявление подпрограмм в интерфейсной части автоматически сопровождается их компиляцией с использованием дальней модели памяти. Таким образом, обеспечивается доступ к подпрограммам из основной программы и других модулей Паскаля.

Следует учесть, что все константы и переменные, объявленные в интерфейсной части модуля Паскаля, равно как и глобальные константы и переменные основной программы, помещаются компилятором Турбо Паскаля в общий сегмент данных (максимальная длина сегмента 65536 байт).

Порядок появления различных разделов объявлений и их количество может быть произвольным. Если в интерфейсной части объявляются внешние подпрограммы или подпрограммы в машинных кодах, их тела (т.е. зарезервированное слово EXTERNAL , в первом случае, и машинные коды вместе со словом INLINE – во втором) должны следовать сразу за их заголовками в исполняемой части модуля (не в интерфейсной!). В интерфейсной части модулей Паскаля нельзя использовать опережающее описание.

**Исполняемая часть модуля Паскаля**

Исполняемая часть модуля Паскаля 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 блоки, а также метки.

Описанию подпрограммы, объявленной в интерфейсной части модуля Паскаля, в исполняемой части должен предшествовать заголовок, в котором можно опустить 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 параметров и объявлением типа результата для функции, то он должен полностью совпадать с заголовком подпрограммы в интерфейсной части, например:

**Пример модуля Паскаля**

**Unit** complexn;  
{--------------------------------}  
**Interface**  
**Type**  
Complex= **record**  
   Re, im: real;  
**End**;  
**Procedure** **AddC**(x,y: complex, **var** z: complex);  
{---------------------------------}  
**Implementation**  
   **Procedure** **AddC**;  
      z.re:= x.re + y.re;  
      z.im:= x.im + y.im;  
   **end** ;  
**end** .

**Написание собственных модулей**

Допустим, Вы написали модуль IntLib, записали его в файл INTLIВ.PAS и оттранслировали на диск; получившийся в результате код находится в файле INTLIВ.ТРU. Для использования этого модуля в программе необходимо включить в нее оператор uses, указывающий компилятору, какой модуль используется. Ваша программа может выглядеть следующим образом:

**program** MyProg;

**uses** IntLib;

Отметим, что Турбо-Паскаль предполагает, что файл, в котором находится модуль, имеет такое же имя, что и сам модуль. Если имя Вашего модуля МyUtilities, то Турбо-Паскаль будет искать файл с именем МYUTILIТ.PAS.

Компиляция модуля

Модуль компилируется точно так же, как компилируется программа: он создается с помощью редактора, а затем вызывается команда Соmpile/Соmpile (Компилировать/ Компилировать) (или нажимаются клавиши Аlt-С). Однако, вместо файла с расширением .ЕХЕ Турбо-Паскаль создает файл с расширением .ТРU (Turbо Раscal Unit - модуль Турбо-Паскаля). После этого Вы можете оставить этот файл как есть или же вставить его в ТURВО.TPL с помощью TPUMOVER.ЕХЕ.

В любом случае имеет смысл переслать файлы с расширением \*.ТРU (вместе с исходными файлами) в каталог модулей, который определен с помощью команды О/D/Unit directories (Каталоги модулей). В одном исходном файле может находиться только один модуль, поскольку компиляция прекращается, как только обнаружен завершающий оператор end.

Пример:

Напишем небольшой модуль. Назовем его IntLib и вставим в него две простые подпрограммы для целых чисел - процедуру и функцию:

**unit** IntLib;

**interface**

**procedure** ISwap(**var** I,J: **integer**);

**function** IMax(I,J: **integer**): **integer**;

**implementation**

**procedure** ISwap;

**var**

Temp: **integer**;

**begin**

Temp:=I;

I:=J;

J:=Temp

**end**; *{конец процедуры ISwap }*

**function** IMax;

**begin**

**if** I > J **then** IMax:=I **else** IMax:=J

**end**; *{конец функции IMax }*

**end**. *{конец модуля IntLib }*

Введем эту подпрограмму, запишем ее в файл INTLIВ.PAS, а затем оттранслируем на диск. В результате получим код модуля в файле INTLIВ.ТРU. Перешлем его в каталог модулей. Следующая программа использует модуль IntLib:

**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 }*

Все описания внутри модуля связаны друг с другом. Например, модуль Crt содержит все описания, необходимые для подпрограмм работы с экраном на Вашем персональном компьютере.

Список используемой литературы

1. <https://itteach.ru/pascal/pascal-modul-unit>
2. <http://www.pascal.helpov.net/index/pascal_modules_programming>