Модули

Стандартный Паскаль не предусматривает механизмов раздельной компиляции частей программы с последующей их сборкой перед выполнением. Вполне понятно стремление разработчиков коммерческих компиляторов Паскаля включать в язык средства, повышающие его **модульность**.

Модуль Паскаля – это автономно компилируемая программная единица, включающая в себя различные компоненты раздела описаний (типы, константы, переменные, процедуры и функции) и, возможно, некоторые исполняемые операторы инициирующей части.

Основным принципом **модульного программирования** является принцип «разделяй и властвуй».**Модульное программирование** – это организация программы как совокупности небольших независимых блоков, называемых **модулями**, структура и поведение которых подчиняются определенным правилам.

Использование модульного программирования позволяет упростить тестирование программы и обнаружение ошибок. Аппаратно-зависимые подзадачи могут быть строго отделены от других подзадач, что улучшает мобильность создаваемых программ.

Термин «модуль» в программировании начал использоваться в связи с внедрением модульных принципов при создании программ. В 70-х годах под модулем понимали какую-либо процедуру или функцию, написанную в соответствии с определенными правилами. Например: «Модуль должен быть простым, замкнутым (независимым), обозримым (от 50 до 100 строк), реализующим только одну функцию задачи, имеющим одну входную и одну выходную точку».

Модули предназначены для поддержки принципов модульного программирования при разработке программ, основным из которых является принцип скрытия информации. Согласно этому принципу влияние логически независимых фрагментов программы должно быть сведено к минимуму. Принцип скрытия информации, поддерживаемый модулями, позволяет создавать надежно работающие и легко модифицируемые программы. Модули применяются преимущественно для создания библиотеки процедур, функций и объектов, которые затем могут использоваться в программах, разрабатываемых пользователем.
Используя модули, важно правильно указывать их имена. При подключении стандартного модуля достаточно корректно записать его идентификатор в предложении USES

При разработке собственных модулей необходимо помнить о следующих правилах:
1) Не допускается одновременное использование модулей с одинаковыми именами.
2) Модуль - программная единица, текст которой компилируется автономно, благодаря этому время компиляции для больших программ существенно сокращается.
3) Имя модуля должно совпадать с именем файла, в котором он хранится, поэтому имя модуля не может состоять более чем из восьми символов.

unit имя модуля;
{интерфейсный раздел}
interface -{ начало раздела объявлений. Здесь описывается взаимодействие данного модуля с другими стандартными и пользовательскими модулями, а также с главной программой. }
{ список импорта интерфейсного раздела}
USES-В этом списке через запятую перечисляют идентификаторы модулей, информация интерфейсных частей которых должна быть доступна в этом модуле. Здесь целесообразно описывать идентификаторы только тех модулей, информация из которых используется в описании раздела interface данного модуля.

{список экспорта интерфейсного раздела}

const

type

var

procedure

function

все эти позиции определены в данном модуле, но могут быть использованы в других программных модулях, включающих имя данного модуля в своей строке ***USES.***

Для процедур и функций здесь описываются только заголовки, но обязательно с полным описанием формальных параметров

{раздел реализации}

implementation – здесь указывается реализационная часть (личная) данного модуля, которая недоступна для других модулей и программ.

{список импорта раздела реализации}

uses – в этом списке через запятую перечисляются идентификаторы модулей, информация интерфейсных частей которых должна быть доступна в данном модуле. Здесь целесообразно описывать идентификаторы всех необходимых модулей, информация из которых не используется в интерфейсном разделе и об использовании которых не должен знать ни один другой модуль.

{раздел описаний, внутренний для модуля}
label
const
type
var
procedure
function
Все эти позиции для внутреннего пользования. Они недоступны никакому другому модулю или программе. Заголовки процедур и функций допускается указывать без параметров, если параметры указаны, то они должны быть идентичны параметрам в интерфейсной части соответствующих процедур.

{раздел инициализации}

begin **-** здесь указываются все начальные установки для корректной работы модуля. Выполняется этот раздел при запуске модуля раньше всех остальных. Используется для открытия файлов и формирования структур данных. Если ничего этого делать не нужно, то модуль просто заканчивают оператором ***end.*** В этом случае begin не пишут.

их отличие от процедур и функций.
Традиционные правила действия локальных и глобальных переменных для модулей не работают. Языковая конструкция «модуль» была выбрана специально, чтобы исключить влияние глобальных переменных, объявленных в данной программе на внутреннее описание модуля. Поэтому, если возникает необходимость ввести какие-либо доступные для всех блоков программы описания (т.е. глобальные описания), то это можно сделать только одним способом – создать модуль глобальных описаний и включить его в список импорта всех остальных модулей

Компиляция модулей Паскаля

В среде Турбо Паскаль имеются средства, управляющие способом компиляции модулей и облегчающие разработку больших программ. Определены три режима компиляции: **COMPILE , MAKE , BUILD**. Режимы отличаются способом связи компилируемого модуля или основной программы с другими модулями, объявленными в предложении USES .

При компиляции модуля или основной программы в режиме **COMPILE** все, упоминаемые в предложении USES модули, должны быть предварительно откомпилированы, и результаты компиляции должны быть помещены в одноименные файлы с расширением TPU (от англ. Turbo Pascal Unit). Файл с расширением TPU создается автоматически при компиляции модуля Паскаля.

файлов для каждого объявленного модуля. Если какой-либо файл не найден, система ищет одноименный файл с расширением PAS , т.е. файл с исходным текстом модуля Паскаля. Если таковой файл найден, система приступает к его компиляции. Кроме того, в этом режиме система следит за возможными изменениями исходного текста любого используемого модуля. Если в PAS -файл внесены изменения, то независимо от того, есть ли в каталоге соответствующий TPU -файл или нет, система откомпилирует его перед компиляцией основной программы. Более того, если изменения внесены в интерфейсную часть, то будут откомпилированы все другие модули, обращающиеся к нему. Режим MAKE существенно облегчает процесс разработки крупных программ с множеством модулей Паскаля: программист избавляется от необходимости следить за соответствием TPU -файлов их исходному тексту, т.к. система делает это

В режиме **BUILD** существующие TPU -файлы игнорируются, система пытается отыскать и откомпилировать соответствующие PAS - файлы для каждого модуля Паскаля. После компиляции можно быть уверенным, что учтены все сделанные в текстах модулей Паскаля исправления и изменения. Подключение модулей Паскаля к основной программе и их компиляция происходит в порядке их объявления в предложении USES . При переходе к очередному модулю Паскаля система предварительно ищет все модули, на которые он ссылается. Ссылки модулей Паскаля друг на друга могут образовывать древовидную структуру любой сложности, однако запрещается явное или косвенное обращение модуля к самому себе. Например, недопустимы следующие объявления:

Пример ошибок модуля Паскаля

```
Unit A;                    Unit B;
interface                  interface
uses B;                    Uses A;
…….                         ……
implementation             implementation
…….                        . …….

end.                       end.
```

Пример исправленных ошибок модуля Паскаля

Unit A;	Unit B;
interface	interface
…….	……
implementation	implementation
uses B;	Uses A;
…….	. ……
end.	end.

Дело в том, что Паскаль разрешает ссылки на частично откомпилированные модули, что приблизительно соответствует опережающему описанию подпрограммы. Если интерфейсные части независимы (это обязательное условие!), Паскаль сможет идентифицировать все глобальные объекты в каждом модуле, после чего откомпилирует тела модулей обычным способом.

Доступ к объявленным в модуле Паскаля объектам

Пример модуля реализующий сложение и вычитание комплексных чисел

```
Unit complexn;
Interface
  type
  complex= record
  re, im: real;
  end;
procedure AddC (x, y: complex; var z: complex);
procedure SubC (x, y: complex; var z: complex);
const c: complex= (re: 0.1; im: -1);
implementation
procedure AddC(x, y: complex; var z: complex);
begin
  z.re:= x.re + y.re;
  z.im:= x.im + y.im;
end; {AddC}
procedure SubC(x, y: complex; var z: complex);
begin
  z.re:= x.re - y.re;
  z.im:= x.im - y.im;
end; {SubC}
end .
```

Текст этого модуля следует поместить в файл complexn . pas . Его можно откомпилировать, создав TPU -файл.

Арифметические операции над комплексными числами

```
Program primer ;
Uses complexn;
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’);
End.
```

После объявления Uses complexn программе стали доступны все объекты, объявленные в интерфейсной части модуля complexn .

Стандартные модули Паскаля

В Паскале имеется 8 стандартных модулей, в которых содержится множество различных типов, констант, процедур и функций. Этими модулями являются SYSTEM, PRINTER, DOS, CRT, GRAPH, OVERLAY, TURBO3, GRAPH3.

Модули Паскаля GRAPH , TURBO 3, GRAPH 3 выделены в отдельные TPU -файлы, а остальные входят в состав библиотечного файла TURBO . TPL . Лишь один модуль Паскаля SYSTEM подключается к любой программе автоматически, все остальные становятся доступны только после указания их имен в списке подключаемых модулей.

Модуль Паскаля SYSTEM. В него входят все процедуры и функции стандартного Паскаля, а также встроенные процедуры и функции, которые не вошли в другие стандартные модули (например, INC , DEC , GETDIR и т.п.). Модуль Паскаля SYSTEM подключается к любой программе независимо от того, объявлен ли он в предложении USES или нет, поэтому его глобальные константы, переменные, процедуры и функции считаются встроенными в Турбо Паскаль.

Модуль Паскаля PRINTER делает доступным вывод текстов на принтер. В нем определяется файловая переменная LST типа TEXT , которая связывается с логическим устройством PRN. После подключения данного модуля Паскаля можно выполнить, например, такое действие:

Пример стандартного модуля Паскаля

```
Uses printer;
Begin
  Writeln(lst, ‘ Турбо Паскаль ’);
End.
```

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 и выводить на экран надписи стандартными или созданными программистом шрифтами.
Модуль Паскаля DOS . В модуле собраны процедуры и функции, открывающие доступ к средствам дисковой операционной системы MS - DOS .
Модуль Паскаля OVERLAY . Данный модуль необходим при разработке громоздких программ с перекрытиями. Паскаль обеспечивает создание программ, длина которых ограничивается лишь основной оперативной памятью. Операционная система MS - DOS оставляет программе около 580 Кбайт основной памяти. Память такого размера достаточна для большинства исполняемых программ, тем не менее, использование программ с перекрытиями снимает это ограничение.
Модули Паскаля TURBO 3 и GRAPH 3 введены для обеспечения совместимости с ранней версией Паскаля.