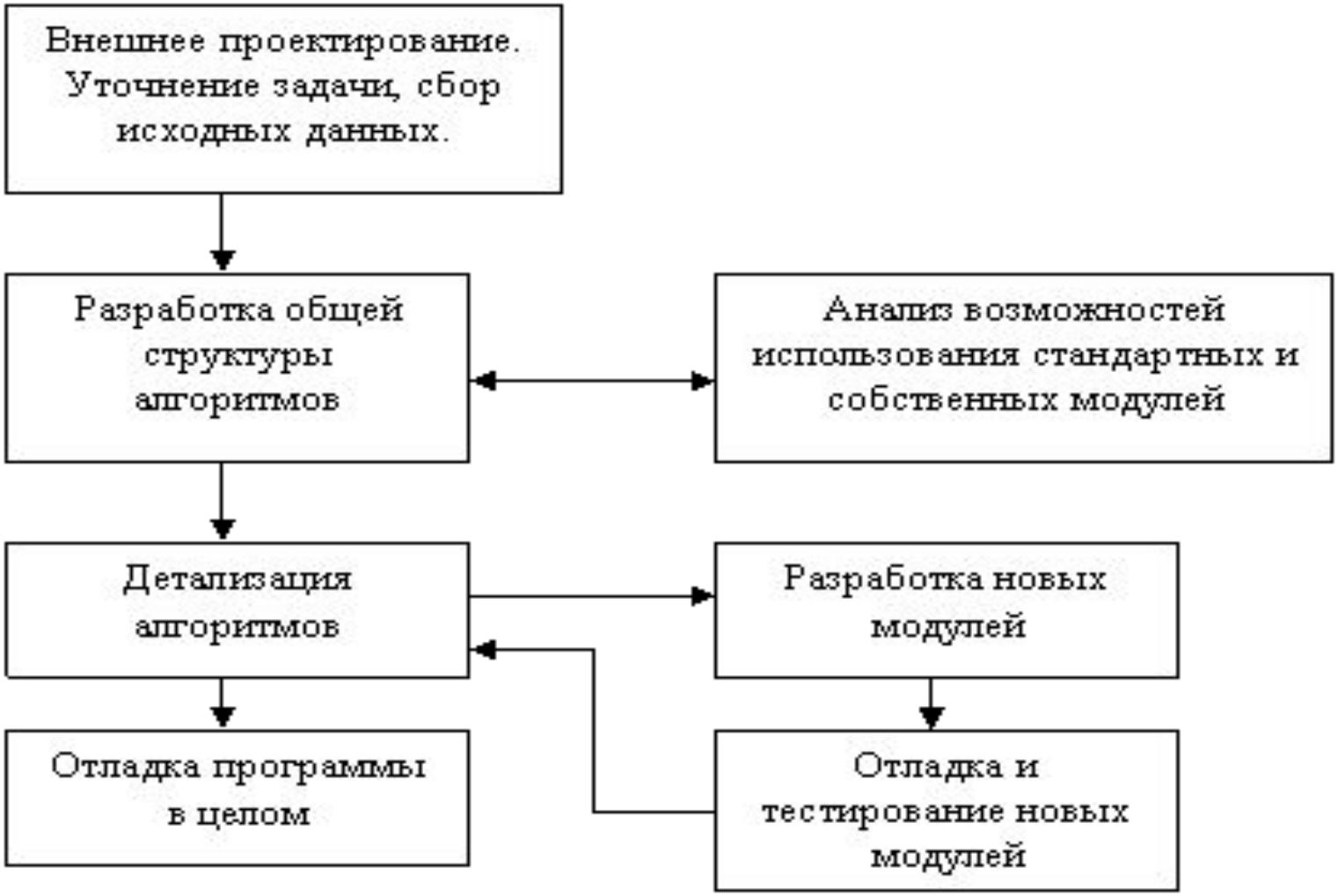
Лекция 7 (часть 4)

Модули

***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.

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.



Значение модулей для технологии разработки программного проекта может быть продемонстрировано диаграммой на рис. 1.

Модули представляют собой прекрасный инструмент для разработки библиотек прикладных программ и мощное средство модульного программирования. **Важная особенность** модулей заключается в том, что компилятор размещает их программный код в отдельном сегменте памяти. **Длина сегмента** не может превышать **64 Кбайт**, однако количество одновременно используемых модулей ограничивается лишь доступной памятью, что позволяет создавать большие программы.

Структура модулей Паскаля

Всякий модуль Паскаля имеет следующую структуру:

Unit <имя_модуля>;
interface <интерфейсная часть>;
implementation < исполняемая часть >;
begin
<инициирующая часть>;
end .

Здесь UNIT – зарезервированное слово (единица); начинает заголовок модуля;

<имя_модуля> - имя модуля (правильный идентификатор);

INTERFACE – зарезервированное слово (интерфейс); начинает интерфейсную часть модуля;

IMPLEMENTATION – зарезервированное слово (выполнение); начинает исполняемую часть модуля;

BEGIN – зарезервированное слово; начинает инициирующую часть модуля; причем конструкция begin <инициирующая часть> необязательна;

END – зарезервированное слово – признак конца модуля.

Таким образом, модуль Паскаля состоит из заголовка и трех составных частей, любая из которых может быть пустой.

Заголовок модуля Паскаля и связь модулей друг с другом

Заголовок модуля Паскаля состоит из зарезервированного слова **unit** и следующего за ним **имени модуля**. Для правильной работы среды Турбо Паскаля и возможности подключения средств, облегчающих разработку больших программ, **имя модуля** Паскаля должно совпадать с **именем дискового файла**, в который помещается исходный текст модуля. Если, например, имеем заголовок модуля Паскаля

Unit primer ;

то исходный текст этого модуля должен размещаться на диске в файле **primer .pas .**

Имя модуля Паскаля служит для его связи с другими модулями и основной программой. Эта связь устанавливается специальным предложением:

uses<список модулей>

Здесь **uses** – зарезервированное слово (использует);

<список модулей> - список модулей, с которыми устанавливается связь; элементы списка – имена модулей через запятую.

Если в Паскале модули используются, то предложение **uses <список модулей>** должно стоять сразу после заголовка программы , т.е. должно открывать раздел описаний основной программы. В модулях Паскаля могут использоваться другие модули. В модулях предложение **uses <список модулей>** может стоять сразу после слова **interface** или сразу после слова **implementation** . Допускается и два предложения **uses** , т.е. оно может стоять и там, и там.

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 **complex** и две процедуры – **AddC** и **MulC** из модуля **complexn** .

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 .
```

Инициирующая часть модуля Паскаля

Инициирующая часть завершает модуль Паскаля. Она может отсутствовать вместе с начинающим ее словом **BEGIN** или быть пустой – тогда вслед за **BEGIN** сразу следует признак конца модуля.

В **инициирующей части** размещаются исполняемые операторы, содержащие некоторый фрагмент программы. Эти операторы исполняются до передачи управления основной программе и обычно используются для подготовки ее работы. Например, в инициирующей части могут инициироваться переменные, открываться файлы, устанавливаться связи с другими компьютерами и т.п.:

Пример модуля Паскаля

```
Unit fileText;
    {-----------------------------}
    Interface
       Procedure print(s: string);
    {-----------------------------}
    implementation
    var f: text;
    const
       name= ‘output.txt’;
    procedure print;
    begin
       writeln(f, s)
    end ;
    {-------------------------------}
    {начало инициирующей части}
    begin
       assign(f, name);
       rewrite ( f );
    {конец инициирующей части}
    end .
```

Не рекомендуется делать инициирующую часть пустой, лучше ее опустить.

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.

В режиме **MAKE** компилятор проверяет наличие **TPU** -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 автоматически.

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.

Пример ошибок модуля Паскаля

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

Это ограничение можно обойти, если «спрятать» предложение **USES** в исполняемые части зависимых модулей:

Пример исправленных ошибок модуля Паскаля

Unit A; Unit B;

interface interface

……. ……

implementation implementation

uses B;Uses A;

…….. ……

end. end.

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

Доступ к объявленным в модуле Паскаля объектам

Пусть, например, мы создаем модуль Паскаля, реализующий сложение и вычитание комплексных чисел с помощью процедур:

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;
    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}
    end .
```

Текст этого модуля следует поместить в файл complexn . pas . Вы можете его откомпилировать, создав TPU -файл.

В следующей программе осуществляются арифметические операции над комплексными числами:

Арифметические операции над комплексными числами

```
Program primer ;
    Uses complexn;
    Var
      a,b,c: coplex;
    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** . При необходимости можно переопределить любой из этих объектов, как произошло, например, с типизированной константой ***c*** , объявленной в модуле Паскаля. Переопределение объекта означает, что вновь объявленный объект «закрывает» ранее определенный в модуле одноименный объект. Чтобы получить доступ к «закрытому» объекту, нужно воспользоваться составным именем: перед именем объекта поставить имя модуля и точку. Например :

Writeln (complexn.c.re: 5: 1, complexn.c.im: 5: 1);

Этот оператор выведет на экран содержимое «закрытой» типизированной константы, объявленной в модуле Паскаля из предыдущего примера.

Стандартные модули Паскаля

В Турбо 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 матричный принтер. В нем определяется файловая переменная **LST** типа **TEXT** , которая связывается с логическим устройством **PRN**.

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. Турбо 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 системы Турбо Паскаль.