Библиотеки (модули unit)

Сколь ни хорошо унаследовать библиотеку, еще лучше собрать ее самому.

О. Биррель

Компьютерные науки © М.Л. Цымблер

Содержание

- Модуль unit
- Структура модуля unit
- Использование модулей unit
- Стандартные модули unit
- Разработка собственных модулей unit

Библиотеки © М.Л. Цымблер 2

Модуль unit

- *Модуль* **unit** – отдельно хранимая и независимо компилируемая программная единица, содержащая определения подпрограмм и других программных объектов.
- В отличие от программной единицы **program**, модуль **unit** *не является выполняемой программной единицей*. Однако все программные объекты модуля доступны для использования в программной единице **program** и других модулях **unit**.

Библиотеки © М.Л. Цымблер 3

Структура модуля **unit**

```
unit  <Имя модуля>;                          { Заголовок }
interface                                    { Интерфейс }
      [ uses <Список используемых модулей unit >; ]
      <Описания видимых программных объектов>
      <Заголовки экспортируемых подпрограмм>
implementation                               { Реализация }
      [ uses <Список используемых модулей>; ]
      <Описания скрытых программных объектов>
      <Реализации экспортируемых подпрограмм>
[ begin                                      { Инициализация }
      <Операторы инициализации> ]
end.
```


Секция **interface**

- *Интерфейсная секция* содержит объявления программных объектов, *видимых* в любой программе/модуле, использующим данный.
- Интерфейсная секция может содержать подключения других модулей unit, а также объявления констант, типов, переменных, подпрограмм.
- При этом для подпрограмм описываются *только заголовки*. Полные описания подпрограмм помещаются в секцию реализации модуля.

Секция **implementation**

- *Секция реализации* содержит полные описания подпрограмм из интерфейсной секции.
- Реализация модуля может содержать также объявления внутренних программных объектов, *не видимых* в других программах и модулях.
- Все программные объекты, объявленные в интерфейсной секции модуля, *видимы* в секции его реализации.

Секция инициализации

- Модуль **unit** может содержать *секцию инициализации*, которая обычно содержит операторы инициализации данных этого модуля.
- При запуске программы, использующей модули, перед выполнением ее тела выполняются операторы из секций инициализации всех используемых ею модулей.

Библиотеки © М.Л. Цымблер 7

Использование модулей **unit**

- Модули **unit** компилируются *независимо* друг от друга и от программы и сохраняются как машинные коды в файлах с расширением **.tpu**.
- Интерфейсная часть модуля **unit** также сохраняется в файле **.tpu** в виде *таблицы символов*, используемой для связывания данного модуля с другими модулями.
- Во время компоновки к машинному коду программы добавляются машинные коды всех модулей, прямо или косвенно используемых в данной программе.

Библиотеки © М.Л. Цымблер 8

Пример использования модуля **unit**

```
{ myprog3.pas, 18-мар-05
  Иванов И.И., МП-101
  Пример программы с модулем unit. }
Program MyProg3;
uses Ucalc; { Подключение модуля unit }
var
   A, B, C: Integer;
   R: Real;
procedure InpData(var A, B, C: Integer);
{ Осуществляет ввод исходных данных }
begin
  ...
end;
procedure OutData(Res: Real);
{ Осуществляет вывод результатов}
begin
  ...
end;
begin
  InpData(A,B,C);
  Calculation(A,B,C, R);
  OutData(R);
end.
```

```
{ ucalc.pas, 18-мар-05
  Петров П.П., МП-102
  Пример модуля unit. }
unit Ucalc;
interface
procedure Calculation(A, B, C: Integer; var
 Result: Real);
{ Выполняет некоторые вычисления с
исходными данными }
implementation
procedure Calculation(A, B, C: Integer; var
Result: Real);
{ Выполняет некоторые вычисления с
исходными данными }
begin
  ...
end;
procedure CalcX(A, B, C: Integer; var X: Real);
{ Выполняет вспомогательные вычисления}
begin
  ...
end;
  ...
  ...
end.
```

Библиотеки © М.Л. Цымблер 9

Предотвращение конфликтов

- Недопустимо использование модулей **unit** с *циклическими ссылками*. Например:

```
unit A;
uses B;
...
```

```
unit B;
uses A;
...
```

```
unit A;
uses B;
...
```

```
unit B;
uses C;
...
```

```
unit C;
uses A;
...
```

- При совпадении идентификаторов в различных модулях **unit**, используемых в одной программе, необходимо использовать *квалификацию идентификаторов*. Например:
  Res := UnitA.Limit + UnitB.Limit(L, 5);

Библиотеки © М.Л. Цымблер 10

Преимущества модулей **unit**

- *Независимая компиляция и кодирование* подсистем большой программной системы (особенно важно в случае коллективной разработки).
- *Инкапсуляция (скрытие) деталей реализации*. Возможно использование модуля **unit** как .tpu файла и текстового файла с интерфейсной секцией.

Библиотеки © М.Л. Цымблер 11

Стандартные модули unit

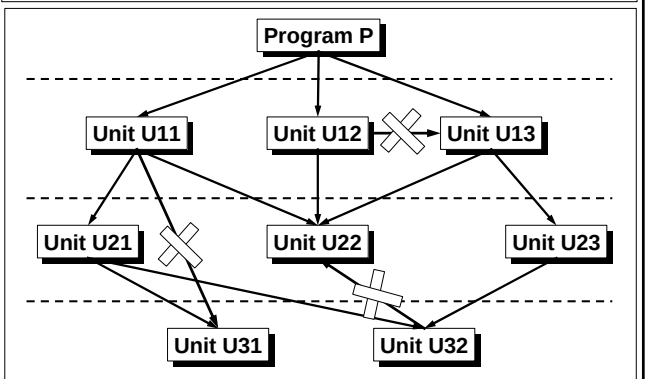
- Коды модулей System, Crt, Overlay, Dos, Printer составляют *библиотеку времени выполнения* и содержатся в файле turbo.tpl. Модуль System подключается к программе автоматически, остальные нужно указывать в предложении **uses**.
- Не включенные в turbo.tpl модули: Graph, Turbo3, Graph3.

Библиотеки © М.Л. Цымблер 12

Разработка собственных модулей **unit**

- Модуль **unit** оформляется в виде отдельного текстового **.pas** файла. Имя файла должно совпадать с именем модуля. В одном файле может быть только один модуль **unit**.
- Способы компиляции модуля **unit** в Turbo-оболочке (команды меню **Compile**): Compile – создание файла .**tpu**, Make и Build – автоматическая компиляция модуля при компиляции программы, использующей этот модуль.
- Большую программную систему необходимо строить как *иерархию модулей*.

Иерархия модулей



Вопросы для обсуждения

- Модули unit vs {$i myprog.pas}
- Недостатки модулей unit: тестирование подпрограмм из секции implementation и др.