Модули

Наумов Д.А., доц. каф. КТ, ИТГД

Программирование и алгоритмические языки, 2020

Содержание лекции

Модульное программирование

организация программы как совокупности независимых блоков (модулей).

Использование модульного программирования:

1. упрощает разработку и тестирование программу;
2. дает возможность разрабатывать программу группе разработчиков;
3. позволяет проводить обновление (замену) модуля без изменения остальной части системы.

Основной принцип структурного программирования: декомпозиция сложных на более простые подззадачи.

Модуль

автономно компилируемая программная единица, реализующая определенную функциональность и предоставляющая интерфейс к ней.

В Pascal модуль представляющая собой библиотеку типов данных, констант, переменных, процедур и функций.

1. исходный текст программы: pas
2. результат компиляции программы: exe
3. исходный текст модуля: pas
4. результат компиляции модуля: tpu (Turbo pascal), dcu (Delphi), pcu (Pascal ABC).

Структура модуля

```
 1 unit <Имя модуля>;//Заголовок модуля
 2 //Интерфейсная секция (открытые описания)
 3   interface
 4     uses <СписокМодулей>;
 5     <РазделОткрытыхОписаний>
 6 //Секция реализации (закрытые описания)
 7   implementation
 8     uses <СписокМодулей>;
 9     <РазделЗакрытыхОписаний>
10 //Секция инициализациия
11   begin
12     <РазделИнициализации>
13   end.
```

Заголовок модуля является обязательным, имя модуля должно быть идентификатором языка Pascal и совпадать с именем файла модуля.

Для связи модулей друг с другом используется секция uses, где перечисляются имена подключаемых модулей.

1. порядок подключения модулей в современных реализациях языка Pascal не важен;
2. повторное подключение модуля в одной программе (модуле) приведет к ошибке компиляции.

```
//текст модуля
1 unit Complex; //заголовок модуля
2 end.

//текст программы
1 uses Complex; //подключаем модуль
2 begin
3 end.
```

Интерфейсная секция

Интерфейсная секция

содержит описания, которые будут доступны в других модулях и программах при подключении данного модуля.

Интерфейсная секция может содержать:

- раздел подключения модулей (чтобы использовать их в разделе описания данной секции);
- раздел описания открытых констант, переменных, типов данных;
- описание (сигнатуры, заголовки) открытых процедур и функций.

Фрагмент модуля работы с комплексными числами

```
1 unit Complex;
2 interface
3   type
4     TCElem = real;
5     TComplex = record Re, Im: TCElem; end;
6   const
7     C_ZERO: TComplex = (Re:0; Im:0);
8     C_ONE: TComplex = (Re:1; Im:0);
9     C_IM_ONE: TComplex = (Re:0; Im:1);
10  procedure Init(var A: TComplex; Re, Im: TCElem);
11  procedure Add(const A, B: TComplex; var C: TComplex);
12  procedure Sub(const A, B: TComplex; var C: TComplex);
13  function GetModule(const A: TComplex): TCElem;
14  function GetArgument(const A: TComplex): TCElem;
```

Секция реализации

Секция реализации должна содержать:

- определение (реализация) процедур и функций, описанных в интерфейсной секции;

При определении процедур и функций, описанных в интерфейсной части, можно опускать список формальных параметров.

Секция реализации может содержать:

- раздел подключения модулей (чтобы использовать их в данной секции и секции инициализации);
- раздел описания закрытых констант, переменных, типов данных;
- определение (реализация) закрытых процедур и функций;

Константы, переменные, типы данных, подпрограммы, описанные в секции реализации, являются локальными для модуля и не будут видны в программах и модулях, которые подключат данный модуль.

Фрагмент модуля работы с комплексными числами

```
15 implementation
16   const
17     EPS = 1e-6;
18   var
19     iCError: integer;
20   procedure Init(Re, Im: TCElem; var A: TComplex);
21  begin
22      A.Re := Re; A.Im := Im;
23  end;
24   procedure Add(const A, B: TComplex; var C: TComplex);
25   begin
26      C.Re := A.Re + B.Re;
27      C.Im := A.Im + B.Im;
28   end;
```

Секция инициализации

В секции инициализации
размещаются операторы, которые исполняются один раз в процессе запуска программы и используются для подготовки работы модуля (инициализация значений переменных и т.д.).

Запуск секций инициализации осуществляется в порядке следования имен модулей в секции uses.

```
29 begin
30   iCError := 0;
31 end;
```

Пример модуля: ComplexUnit.pas.
Пример использования модуля: ComplexProgram.pas.