Introduction to Computers and Programming

Prof. I. K. Lundqvist

Reading: FK pp. 175-182, 193-194, 347-353

Lecture 10
Sept 24 2003

Recap

- User defined types
  - Scalar types
    - Ordered → relational operators are defined
    - Each value of a discrete type has a position number
  - Operations on scalar types
  - Sub-types
  - Enumeration types
  - Derived types
- Packages
  - Procedures
  - Functions
  - Formal vs. actual parameters

Scope of Declarations

- Where does a given declaration apply?
- What declarations apply at a given point?
- Scope of a declaration
  - From where it is made, to the end of the subprogram that contains it

Visibility

A declaration can be hidden from direct visibility, but not hidden from all visibility, and can be accessed using selector syntax:

```
procedure P is
   X : Integer;
   procedure Q is
      X : Integer; -- hides outer declaration
   begin
      X := 2;  -- local decl. directly visible
      P.X :=3; -- global decl. Visible,
               -- but not directly
   end Q;
begin
   Q;
end;
```

Visibility

Some declarations are hidden from all visibility, in particular once an inherited declaration is overridden, there is no way to name it:

```
type T is (A, B, C, D);
procedure P (X : T );
type T1 is new T;
-- inherited P is visible
procedure P (X : T1 );
-- inherited P is hidden from all visibility
```

Example

```
1 with TEXT_IO; use TEXT_IO;
2
3 procedure main is
4
5 length : constant := 4;
6 str : string (1..length);
7 num : INTEGER;
8
9
10 procedure one (num, len : in INTEGER) is
11
12 str : string (1..10);
13
14 begin -- one
15 ... ... ... ...
16 end one;
17
18
```

```
19 X, Y : FLOAT;
20
21
22 procedure two (len : in INTEGER) is
23
24 X : INTEGER;
25
26 begin -- two
27 ... ... ... ...
28 end two;
29
30
31 begin -- main
32 ... ... ... ...
33 end main;
```

Visibility of each name

Name	Line	one	two	main
PUT, etc	1			
length	5			
str	6			
num	7			
num	10			
len	10			
str	12			
X	19			
Y	19			
len	22			
X	24			

Visibility of each name

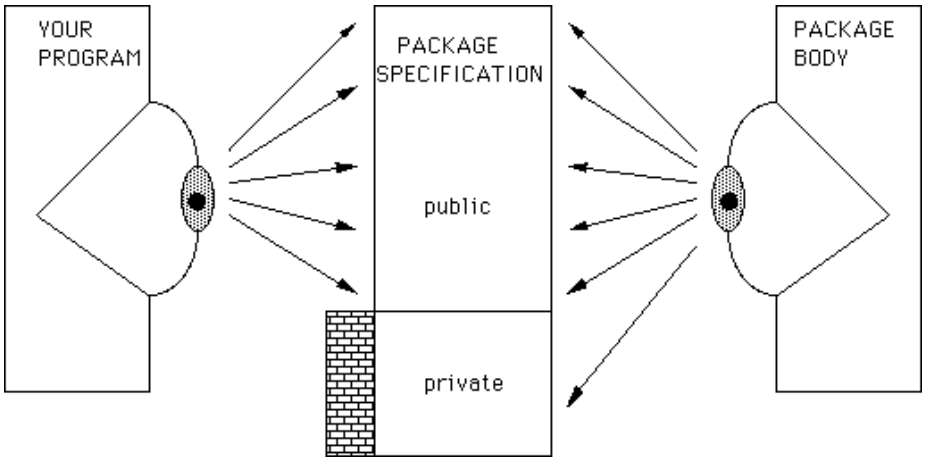
Name	Line	one	two	main
PUT, etc	1	Y	Y	Y
length	5	Y	Y	Y
str	6	N	Y	Y
num	7	N	Y	Y
num	10	Y	N	N
len	10	Y	N	N
str	12	Y	N	N
X	19	N	N	Y
Y	19	N	Y	Y
len	22	N	Y	N
X	24	N	Y	N

Packages

- Collection of resources
- Encapsulated in one unit
- Single library unit
  - Free-standing unit
    - Must contain its own declarations for everything it needs
  - Compiled on its own
    - Incorporated in other programs via ‘with’
    - Compilation order:
      - Library unit
      - Procedures that use it

Package Organization

- Package **specification** show "what" it provides
- Package **body** defines "how" it is implemented
- Both are separate from the user's program that uses the package



Courtesy of Chris Lokan. Used with permission.

Package Specification

- **package** package_name **is**
  declarations

  private
  type definitions
  end package_name;

  public portion

  private portion

- Public:
  - What you need to know to use the package
- Private:
  - Implementation of data types

Private Types

```
package accounts is
  type account is private; -- declaration comes later

  procedure withdraw(an_account : in out account;
                     amount     : in money);

  procedure deposit(an_account: in out account;
                    amount    : in money);

  function create(initial_balance : money) return account;

  function balance( an_account : account) return integer;

private -- this part of the package specification
        -- contains the full description.

  type account is
    record
      account_no :positive;
      balance :integer;
    end record;
end accounts;
```

Package Body

- Implementation of the resources provided by the package
- All a user of the package needs to know is what the package provides.
- The package is a "black box" to the user of the package.
- The package body is not visible to a package user.

Package Body

```
package body package_name is

  declarations

end package_name;
```

Package Example specification .ads

```
package PLANIMETRY is

  type length is digits 5 range 0.0 .. 1.0E10;
  type area is digits 5 range 0.0 .. 1.0E20;

  function area_rectangle (L,H : length) return area;
  function area_circle    (R : length)   return area;
  function area_triangle  (B,H : length) return area;
  function circumf_circle (R : length)   return length;

end PLANIMETRY;
```

Package Example
body .adb

```
package body PLANIMETRY is

   PI : constant := 3.1415926536;

   function area_rectangle (L,H : length) return area is
   begin
     return area(L) * area(H);
   end;

   function area_circle (R : length) return area is
   begin
     return PI * area(R) ** 2;
   end;
```

Package Example
body .adb

```
   function area_triangle (B,H : length) return area is
   begin
     return area(B) * area(H) / 2.0;
   end;

   function circumf_circle (R : length) return length is
   begin
     return 2.0 * PI * R;
   end;
 end PLANIMETRY;
```

Using Packages

- To use a package element
  - package.element
- Example
  - ```
    Ada.Text_Io.put (item => “abc”);
    Ada.Text_Io.new_line;
    int_io.put (mark, width => 1);
    planimetery.area_circle (2.0);
    ```
- USE allows package to be omitted
  - ```
    use Ada.Text_io, int_io, planimetry;
    …
    put (“abc”);
    new_line;
    put (mark, width => 1);
    area_circle (2.0);
    ```

User Program

```
with TEXT_IO, PLANIMETRY;

procedure main is

  use TEXT_IO;
  ... declarations

  L : PLANIMETRY.length;   -- length
  H : PLANIMETRY.length;   -- height
  A : PLANIMETRY.area;     -- area
  R : PLANIMETRY.length;   -- radius

begin
  R := ... ;
  A := PLANIMETRY.area_circle (R);
end main;
```

Case Statement

- Used for multiple selections
  - Alternative to multiple if
  - Used when we can explicitly list all alternatives for one selector

```
statement_before;

case selector is
   when value_list_1 =>
      statement(s)_1;
   when value_list_2 =>
      statement(s)_2;
      …
   when others =>
      statement(s)_n;
end case;

statement_after;
```

Selectors

- Variable or expression resulting in a discrete value
- Selector value_list may be:
  - A single constant value, e.g., ‘a’
  - A series of alternatives, e.g., ‘a’ | ‘b’ | ‘c’
  - A range of values e.g., ‘a’ .. ‘z’
  - Or any combination of the above

Restrictions on Case Statements

- A particular value may only occur **once** in a case statement
- All possible values of the selector **must** be supplied, either explicitly or using **when others**.
- **when others** indicates the action when none of the listed **when** alternatives are matched
  - it must be the last alternative
  - to specify no action, use the "`null;`" statement

Case vs Multiple if

- Case
  - Table of values and actions
  - Easy to operate specify range of selector values
  - Easy to specify alternative selector values
- Multiple if
  - Sequence of decisions and actions
  - Used when cannot specify range directly
    - Selector is not discrete
    - Choice depends on more than one selector