Übungsblatt 10 zu “Programmiersprachen”

Berthold Hoffmann, Studiengang Informatik (hof@informatik.uni-bremen.de)

Besprechung am 25. Mai 2010

Einfache Pakete in C++

Realisieren Sie folgende Pakete in C++ – soweit möglich, bzw. so genau wie möglich.

Ein einfaches Paket

```
package Earth is
   type Continent is (Africa, Antarctica, Asia, Australia, Europe, NAmerica, SAmerica);
   radius: constant Float := 6.4e3;
   area: constant array (Continent) of Float :=
                        (30.3e6, 13.0e6, 43.3e6, 7.7e6, 10.4e6, 24.9e6, 17.8e6);
   population: array (Continent) of Integer;
end Earth;
```

Ein Paket mit Schnittstelle

```
package Trig is
   function sin (x: Float) return Float;
   function cos (x: Float) return Float;
end Trig;

package body Trig is
   twice_pi: constant Float := 6.2832;
   function norm (x: Float) return Float is
   begin
     . . . -- code to compute x modulo twice_pi
   end;

   function sin (x: Float) return Float is
   begin
     . . . -- code to compute the sine of norm(x)
   end;
   function cos (x: Float) return Float is
   begin
     . . . -- code to compute the cosine of norm(x)
   end;
end Trig;
```

Ein Paket mit gekapselter Variable

```
package The_Dictionary is
   procedure add (wd: in Word);
   -- Add word wd to the dictionary if it is not already there.
   function contains (wd: Word) return Boolean;
   -- Return true if and only if word wd is in the dictionary.
end The_Dictionary;

package body The_Dictionary is
   maxsize: constant := 1000;
   size : Integer := 0;
   words: array (1 .. maxsize) of Word;
   -- The dictionary is represented as follows: size contains the number of
   -- words, and words(1..size) contains the words themselves, in no
   -- particular order.
   procedure add (wd: in Word) is
   begin
     if not contains(wd) then
         size := size + 1;
         words(size) := wd;
     end if;
   end;
   function contains (wd: Word) return Boolean is
   begin
     for i in 1 .. size loop
         if wd = words(i) then
             return true;
         end if;
     end loop;
     return false;
   end;
end The_Dictionary;
```

Ein Paket mit einem konkreten Datentyp

```
package Dictionaries is
  maxsize: constant := 1000;
  type Dictionary is
       record
            size : Integer;
            words: array (1 .. maxsize) of Word;
       end record;
  procedure clear (dict: in out Dictionary);
  -- Make dictionary dict empty.
  procedure add (dict: in out Dictionary;
                              wd: in Word);
  -- Add word wd to dictionary dict if it is not already there.
  function contains (dict: Dictionary; wd: Word)
                   return Boolean;
  -- Return true if and only if word wd is in dictionary dict.
end Dictionaries;

package body Dictionaries is
  procedure clear (dict: in out Dictionary) is
  begin
     dict.size := 0;
  end;
  procedure add (dict: in out Dictionary;
                   wd: in Word) is
  begin
     if not contains(dict, wd) then
        dict.size := dict.size + 1;
        dict.words(dict.size) := wd;
     end if;
  end;
  function contains (dict: Dictionary; wd: Word)
              return Boolean is
  begin
     for i in 1 .. dict.size loop
        if wd = dict.words(i) then
           return true;
        end if;
     end loop;
     return false;
  end;
end Dictionaries;
```

Ein Paket mit einem abstrakten Datentyp

```
package Dictionaries is
  type Dictionary is limited private;
  -- A Dictionary value represents a set of words.
  procedure clear (dict: in out Dictionary);
  -- Make dictionary dict empty.
  procedure add (dict: in out Dictionary;
                          wd: in Word);
  -- Add word wd to dictionary dict if it is not already there.
  function contains (dict: Dictionary; wd: Word)
                  return Boolean;
  -- Return true if and only if word wd is in dictionary dict.
private
   maxsize: constant Integer := 1000;
   type Dictionary is
         record
            size : Integer;
          words: array (1 .. maxsize) of Word;
        end record;
end Dictionaries;

function "=" (dict1, dict2: Dictionary)
           return Boolean is
begin
   if dict1.size /= dict2.size then
      return false;
   end if;
   for i in 1 .. dict2.size loop
     if not contains(dict1, dict2.words(i)) then
        return false;
     end if;
   end loop;
   return true;
end;
```