SWEN90010 lecture – Ada

Lecture outline

1. Obtain an initial understanding of the basic Ada language – enough to construct a small program.
2. To understand the properties of Ada that make it suitable for high integrity systems.

Lecture plan

1. History:
   (a) Introduced in 1977 when DoD became frustrated with supporting *hundreds* of programming languages (platform specific, obsolete, not re-usable). Wanted a *safe, modular* programming language.
   (b) Four contractors asked to put forward designs; CII Honeywell Bull's proposal called *Ada* was selected.
   (c) Designed for embedded, real-time systems, but adopted more generally.
   (d) Named after *Augusta Ada King, Countess of Lovelace*, (more generally known as *Ada Lovelace*) – thought to be the first computer programmer on Charles Babbage's *analytical engine* (calculating Bernoulli numbers).
2. Ada for software engineering:
   (a) A strong, static and safe type system; all type errors detected automatically at compile time.
   (b) Modularity. Ada modules (called "packages") compiled separately before implementation.
   (c) Information hiding.
   (d) Readability. Ada favours the *reader* over the *writer*.
   (e) Portability.
   (f) Standardisation. Ada is an ANSI and ISO standard.
3. hello_world.adb: compilation, procedure, "with", "begin..end", must be called "hello_world.adb"
4. greetings.ads/b: package specifications and bodies.
5. Types: major reason for Ada's success. Of important: types are equivalent by *name only*.
   (a) Integer, Float, Character, String, Boolean, etc.
   (b) Arrays: indices are not (just) integers (array_examples.adb)
   (c) New data types: dates.ads
6. Control structures: if-then, case, loops,
7. Procedures and functions.
   A procedure call is a statement and does not return any value and is considered a statement.
   A function returns a value and is itself an expression.

8. Parameter modes: in, out, in out, access (pointers)

9. Calling: no parameters, positional, named associations.

10. Tasks:

    (a) Show the "housekeeping" template, fill in the body, and run the tasks. Note the "null" main procedure.

    (b) Show the "entry call" example (without the `terminate`), which explains entry calls. Demonstrate the lack of termination. Add in termination condition to the select statement.

    (c) Insert the guard `Datum < 100` into the entry call example, and modify the main procedure to set `Datum` to be greater than 100. Re-run the example.

11. Subtyping: If time permits, lead into workshop 2 with an example of subtyping in `subtyping.adb`. Play around with the two sub-types, and show how the compiler detects out of bounds errors, but does not specify that the types are incompatible:

    Example of why subtyping is good: Mars Climate Orbiter. Disintegrated in 1999 when going towards Mars. Too close to the planet due to the ground-based software outputing non-SI pounds per second instead of the metric Netwons per second.

Listing 1: hello_world.adb

```
with Ada.Text_IO;

procedure Hello_World is
  begin
    Ada.Text_IO.Put_Line("Hello, world!");
end Hello_World;
```

Listing 2: greetings.ads

```
package Greetings is
   procedure Hello;
   procedure Goodbye;
   function Talk(Text : in String) return Integer;
end Greetings;
```

Listing 3: greetings.adb

```
with Ada.Text_IO; use Ada.Text_IO;

package body Greetings is
   procedure Hello is
   begin
      Put_Line("Hello, world!");
   end Hello;

   procedure Goodbye is
   begin
      Put_Line("Goodbye, world!");
   end Goodbye;

   function Talk(Text : in String) return Integer is
   begin
      Put_Line(Text);
      return 1;
   end Talk;
end Greetings;
```

Listing 4: array_examples.adb

```
with Ada.Text_IO; use Ada.Text_IO;
with Ada.Integer_Text_IO; use Ada.Integer_Text_IO;

procedure Array_Examples is
   Int_Index : array(Integer range 5..10) of Character :=
     ('a', 'b', 'c', 'd', 'e', 'f');
   Ch_Index : array(Character) of Character;
begin
   Put(Int_Index(5));  -- accessing using an integer index
   New_Line;

   Ch_Index('a') := 'z'; -- setting using a char index
   Put(Ch_Index('a')); -- accessing using a char index
   New_Line;

   -- setting and getting an array 'slice'
   Int_Index(6 .. 8) := ('X', 'Y', 'Z');

   -- array attributes "'First" and "'Last"
   Put(Ch_Index'First);
   New_Line;
   Put(Int_Index'Last);
   New_Line;
end Array_Examples;
```

Listing 5: date.ads

```
type Day_type   is range    1 .. 31;
type Month_type is range    1 .. 12;
type Year_type  is range 1800 .. 2100;
type Hours is mod 24;
type DayOfWeek is (Monday, Tuesday, Wednesday, Thursday,
                   Friday, Saturday, Sunday);

type Date is
   record
     Day   : Day_type;
     Month : Month_type;
     Year  : Year_type;
   end record;
```

Listing 6: calling_subprograms.adb

```
with Ada.Text_IO; use Ada.Text_IO;
with Ada.Integer_Text_IO; use Ada.Integer_Text_IO;

procedure Calling_Subprograms is
   An_Int : Integer := 50;
   Another_Int : Integer := 60;
begin
   Put(An_Int, 20);  -- positional parameter
   New_Line;         -- no parameters
   Put(Width => 20, Item => An_Int);  -- named associations
end Calling_Subprograms;
```

Listing 7: subtyping.adb

```
procedure Subtyping is

   subtype One is Integer range 0..100;
   subtype Two is Integer range 20..300;

   X : One := 0;
   Y : Two := 25;
begin
   X := X + Y;
end Subtyping;
```