Section 02 - Subprograms
Subprograms 01

Pedro Fernando Flores Palmeros

1 INTRODUCTION

In the previous section we have used procedures, mostly to have a main body of code to execute.

There are two kinds of subprograms in Ada, *functions* and *procedures*. The distinction between the two is that a **function returns a value** and **returns does not return anything**.

In the next two code snippets is shown a function can be declared and implemented, the declaration is the function signature as it is shown in Code 1, and the implementation of the function body is presented at Code 2

Code 1: increment.ads

```
1 function Increment (I : Integer) return Integer;
```

Code 2: increment.adb

```
1 function Increment (I : Integer) return Integer i
2
3 begin
4    return I + 1;
5 end Increment;
```

Subprograms in Ada can have parameters. Also paratmeters can have default values. When calling the subprogram you then omit parameters if they have a default value. Unlike C/C++, a call to a subprogram without parameters does not include parentheses.

Code 3: increment.adb

```
function Increment_By
   (
   I    : Integer := 0;
   Incr : Integer := 1) return Integer;
   )
```

Code 4: increment.adb

```
function Increment_By
   (
   I    : Integer := 0;
   Incr : Integer := 1
   ) return Integer is
begin
   return I + Incr;
end Increment_By
```

The GNAT toolchain, requires the following file name scheme:

- files with the `.ads` extension contain the specification, while
- files with the `.adb` extension contain the implementation.

Therefore, in the GNAT toolchain, the specification fo the Incrementfunction must be soted in the `increment.ads` file, whicle its implementation must be stored in the `increment.adb` file. this rule always applies to packages.

In the next Code.(5) the main objective is to present different ways of invoking the subprograms.

Code 5: main.adb

```

with Ada.Text_IO; use Ada.Text_IO;
with Increment_By;

procedure Show_Increment is
   A, B, C : Integer;
begin
   C:= Increment_By; -- Parameterless call
   -- Value of I is 0, and Incr is 1

   Put_Line ("Using defaults for Increment_By is"
           & Integer'Image(C));
   A := 10;
   B := 3;
   C := Increment_By(A, B); -- Regular parameter passing

```

```
17    Put_Line("Increment of " & Integer'Image(A) & " with " & Integer'Image(B) &
18          " is " & Integer'Image(C));
19
20    A := 20;
21    B := 5;
22    C := Increment_By( I => A, Incr => B); -- Named parameter passing
23
24    Put_Line("Increment of "
25          & Integer'Image(A)
26          & " with "
27          & Integer'Image(B)
28          & " is "
29          & Integer'Image(C));
30 end Show_Increment;
```

In the previous code there are some interesting lines that needs our attention, in line 8 the subprogram is called with no parameters, hence in this case, the values for the execution might be the default values defined in the supbrogram definition.

In line 15 the subprogram is executed by sending two arguments and they have to be sorted depending on the subprogram definition.

In line 22 the subprogram is executed using two arguments, observe that in this case variable assignation is being performed in the subprogram called, hence in the order of the parameters is not needed.