Ada95 / GNAT introduction

General

- Ada compiler: gnatmake
- Ada specification files: package_name.ads
- Ada implementation files: package_name.adb

Useful Packages

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

Offers: `Put('String'); Put_Line('String');` for printing characters on the screen.

To print numbers, they can be converted to a String using `Integer'Image(number)`, or `Float'Image(number)`. Strings can be concatenated using the "&" symbol: `'string1' & 'string 2'```

Statements

```ada
<statements>:
    loop | while <condition> loop | for I in (3..6) loop
        <statements>
        [exit;]
    end loop;

    if <condition> then
        <statements>
    elsif <condition> then
        <statements>
    else
        <statements>
    end if;

    Variable:=1;
    Procedure_Name(Argument1, Argument2);
```
Package structure

Specification

package Example_Package is
   <declarations>
end Example_Package;

<declarations>:
   <procedure_declaration>  |
   <function_declaration>  |
   <typeDeclaration>  |
   <task_declaration>  |
   <protected_declaration>

<procedure_declaration>:
   procedure Proc1 (Param1: in Integer; Param2: out Integer);

<function_declaration>:
   function Func1 (Param1: in Integer) return Float;

<typeDeclaration>:
   type Enumeration_Type is (one, two, three, something_else);
   type Modular_Type is mod 15;
   subtype Marker2 is Integer range 5..12;
   type List2 is array (Enumeration_Type) of Element;
   type Record_Type is record
      <typeDeclaration>*
   end record;

<task_declaration>:
   task [type] Task_Type is
      entry Entry1(Argument: in Integer);
   end Task_Type;

<protected_declaration>:
   protected [type] Protected_Type is
      entry Entry1(Argument: in Modular_Type);
      private:

   end Protected_Type;

<condition>:
   <condition> and <condition>  |  <condition> or <condition>  |  not <condition>  |
   <literal> = <literal>  |  <literal> /= <literal>  |
   <number> > <number>  |  <number> < <number>  |  <number> >= <number>  |  <number> <number>
Implementation

package body Example_Package is
  <declarations>
  <implementations>
begin
  <Statements>
end Example_Package;

<implementations>:
  <procedure_implementation> | 
  <function_implementation> | 
  <task_implementation>    | 
  <protected_implementation>

<procedure_implementation>:
  procedure Proc1 (Param1: in Integer; Param2: out Integer) is
  <declarations>
begin
  <Statements>
end Proc1;

<function_implementation>:
  function Func1 (Param1: in Integer) return Float is
  <declarations>
begin
  <Statements>
end Func1;

<task_implementation>:
  task body Task_Type is
  <declarations>
begin
  accept Entry1(Argument: in Integer) do
    <Statements>
  end;
  <Statements>
exception
  when others => Put_Line('Error');
end Task_Type;

<protected_implementation>:
  protected body Protected_Type is
    entry Entry1(Argument: in Modular_Type) when <condition> is
    <declarations>
begin
    <Statements>
end Entry1;
end Protected_Type;