Grado en Ingeniería de Computadores

Laboratorio de Sistemas en Tiempo Real

L1 - El lenguaje de programación Ada

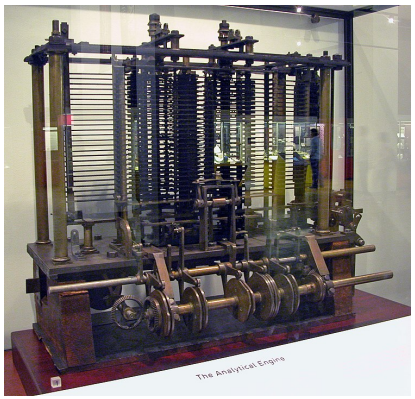
Departamento de Automática

Universidad de Alcalá

23 de septiembre del 2020

Origen

- Ada es un lenguaje de programación diseñado por Jean Ichbiah de CII Honeywell Bull por encargo del Departamento de Defensa de los Estados Unidos en 1980
- Esta propuesta era un sucesor de un lenguaje anterior llamado LIS y desarrollado durante los años 1970
- El nombre se eligió en conmemoración de lady Augusta Ada Byron (1815-1852) Condesa de Lovelace, considerada la primera programadora de la Historia



Características

- Fuertemente tipado
- Orientado a objetos
- Concurrente

Objetivo

- Ada se usa principalmente en entornos en los que se necesita una **gran seguridad** y **fiabilidad** como la defensa, la aeronáutica, la gestión del tráfico aéreo y la industria aeroespacial entre otros.

AIRBUS

Indra

Compiladores

- El compilador de Ada que utilizaremos en la asignatura es **GNAT**
- Originalmente desarrollado por la Universidad de Nueva York bajo el patrocinio del Departamento de Defensa de USA
- En realidad es un **compilador cruzado** basado en gcc y, por lo tanto **software libre**
- Actualmente está mantenido por AdaCore, una empresa que ofrece soporte y servicios sobre el compilador

Instalación

- Para su instalación en linux únicamente hay que instalar el paquete gnat. Por ejemplo: apt install gnat-7

Entornos de desarrollo integrado (IDEs)

- Aunque existem varios IDEs que soportan Ada como Eclipse o Netbeans, en el laboratorio utilizaremos editores de texto avanzados como Kate o Sublime

Compilación

- La compilación implica varios pasos que resuelven las dependencias, enlazan, traducen a C y de ahí a código máquina ejecutable. Veremos un ejemplo del proceso en la primera práctica.
- De forma general la compilación se hace con la orden: **gnatmake archivo_principal.adb**
- gnatmake resolverá las dependencias y llevará a cabo todas las compilaciones adicionales necesarias, generando el archivo ejecutable **archivo_principal**

Escribiendo un Hola_mundo.adb en Ada

- El "Hola mundo" en Ada quedaría así

```
with Ada.Text_IO;    -- Incluimos el paquete Text_IO de Ada
use Ada.Text_IO;     -- Es similar a namespace en C++

procedure Hola_Mundo is
begin
    Put_Line ("Hola mundo!");  -- Put_Line esta en Text_IO
end Hola_Mundo;
```

Compilando Hola_mundo.adb

- El "Hola mundo" se compilaría con gnatmake Hola_mundo.adb
- Fíjate, en Ada el **procedimiento principal** (lo que es el main en C, el **punto de entrada al programa**) se llama normalmente igual que el archivo principal. Si no es así el compilador dará un warning

✎ Escribe, compila y ejecuta el "Hola Mundo"

- Comprueba que todo funciona correctamente
- Comenta la línea del **use** poniendo −− al comienzo ¿qué ocurre? ¿cómo lo solucionarías?
- Cambia el nombre del archivo y vuelve a compilarlo ¿qué ocurre?

Características generales

- La sintaxis intenta que el lenguaje sea **legible**
- No distingue entre mayúsculas y minúsculas (aunque Linux sí)
- Un programa es un único procedimiento, que puede contener subprogramas (procedimientos y funciones)
- Se distingue entre **procedimientos** (subrutinas que **no devuelven ningún valor**, pero **pueden modificar sus parámetros**) y **funciones** (subrutinas que **devuelven un valor** y **no modifican los parámetros**). Esto es un mecanismo adicional de seguridad
- El operador de **asignación** es **:=**, el de **igualdad** es **=**

Partes de un programa

- Un programa simple en Ada (con un procedimiento principal) suele tener tres partes: La zona donde se incluyen los paquetes o librerías, la parte declarativa y el cuerpo

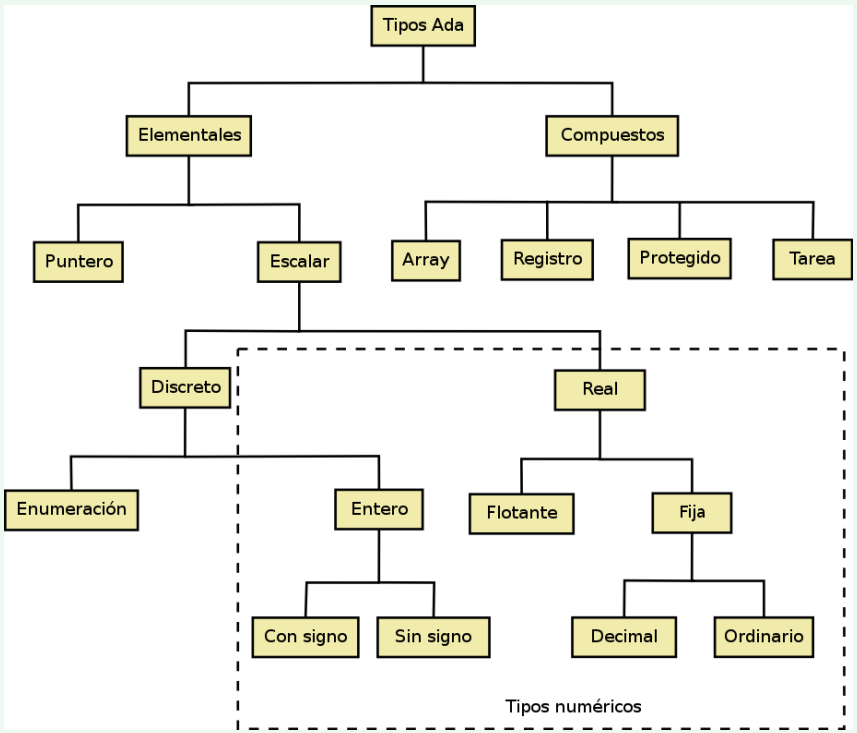
```
with Ada.Text_IO;    -- Primero se incluyen los paquetes

procedure Hola_Mundo is    -- Procedimiento principal
    -- Segundo se declaran variables del procedimiento
begin
    -- Cuerpo donde estan las sentencias ejecutables
    Text_IO.Put_Line ("Hola mundo!");
end Hola_Mundo;
```

Tipos como Objetos

En Ada los tipos se caracterizan por:

- Un conjunto de valores válidos (Fuertemente tipado)
- Un conjunto de operaciones que se pueden aplicar al tipo



Declaración de variables básicas

En Ada las variables se **declaran** como "nombre_de_variable : tipo":

```
x : Integer;              -- una variable de tipo integer llamada x
a, b, c : Integer;        -- Tres variables de tipo integer
d, e, f : Integer := 0;   -- Podemos inicializar las variables
M: Float:=5.5;            -- Declaramos una variable de tipo Float
```

Paquetes genéricos Ada.Text_IO [1]

- Se utilizan a modo de plantilla
- Tienen que ser **instanciados** por el usuario

Ejemplo

```
-- Creacion de ejemplares, para la entrada/salida de numeros,
-- a partir de los paquete genericos "Ada.Text_Io.Integer_Io"
-- y "Ada.Text_Io.Float_Io".
    package ES_int is new Ada.Text_Io.Integer_Io(Integer)
    package ES_float is new Ada.Text_Io.Float_Io(Float)

-- Creacion de ejemplares, para la entrada/salida de un
-- enumerado definido por nosotros
-- tipo definido por nosotros
    type dia_t is (LUN, MAR, MIE, JUE, VIE, SAB, DOM);
    package ES_dia_t is new Ada.Text_Io.Enumeration_Io(dia_t)
```

[1] https://es.wikibooks.org/wiki/Programación_en_Ada/Unidades_predefinidas/Ada.Text_IO

Paquetes genéricos Ada.Text_IO [2]

- Con el paquete de entrada salida de texto podemos leer por teclado e imprimir por pantalla **texto**
- Los **strings** se pueden concatenar con **&**

Ejemplo 1 – Manejando cadenas

```
-- Ejemplo de Ada.Text_IO con strings
with Ada.Text_IO; use Ada.Text_IO;

procedure ejemploString1 is
      s1: String(1 .. 2);
      s2: String(1 .. 3);
begin
      put("Introduce 5 caracteres y pulsa enter.. ");
   -- Al ser la cadena s1 de longitud 2 lee los 2 primeros
      get(s1);
      put_line("Primeros 2: " & s1);

   -- Al ser la cadena s2 de longitud 3 lee los 3 siguientes
      get(s2);
      put_line("Siguientes 3: " & s2);
      new_line;
end ejemploString1;
```

[1] https://www.radford.edu/~nokie/classes/320/stringio.html

Paquetes genéricos Ada.Text_IO [3]

- El primer ejemplo de instanciación de un paquete genérico es Ada.Text_IO para manejar diferentes tipos de variables en **texto**
- Nos permite leer por teclado y mostrar por pantalla

Ejemplo 2 – Manejando cadenas

```
-- Ejemplo de Ada.Text_IO con strings
with Ada.Text_IO; use Ada.Text_IO;

procedure ejemploString2 is
-- Creacion de ejemplares , para la entrada / salida de un
-- enumerado definido por nosotros
-- tipo definido por nosotros
    type dia_t is ( LUN , MAR , MIE , JUE , VIE , SAB , DOM );
    package ES_dia_t is new Ada.Text_Io.Enumeration_Io( dia_t );

    hoy: dia_t;
begin
       put("Introduce el dia de la semana : ");

-- Al ser la cadena s1 de longitud 2 lee los 2 primeros
       ES_dia_t.get(hoy);
       put("El dia de la semana es: ");
       ES_dia_t.put(hoy);

end ejemploString2;
```

[1] .../stringio.html

Paquetes genéricos Ada.Text_IO [4]

- Con el paquete de entrada salida de texto podemos leer por teclado e imprimir por pantalla **cualquier otro tipo que queramos**
- Para ello hay que especificar el tipo al/desde el que se convertirá el texto

Ejemplo 3 – Calculando el IVA

```
-- Ejemplo de Ada.Text_IO con Float
with Ada.Text_IO; use Ada.Text_IO;

procedure CalculaIVA is
-- Instanciamos un paquete a partir del generico Text_Io.Float_Io que maneja Floats
      package ES_Float is new Ada.Text_Io.Float_Io(Float);

      precio: Float;
      IVA: Float;
begin
      put("Introduce el precio con IVA: ");
-- Ahora el usuario introducira por teclado una cadena de texto
      ES_Float.get(precio);
-- El paquete ES_Float ha leido la cadena y la ha transformado a Float
      IVA := precio*(1.0 - 1.0 / 1.21);
      put_line("El IVA del producto es " & IVA'Image);
end CalculaIVA;
```

Grado en Ingeniería de Computadores

Laboratorio de Sistemas en Tiempo Real

L1 - El lenguaje de programación Ada

23 de septiembre del 2020