UNIVERSIDAD DE CANTABRIA

Seminario de Programación en Ada

Profesor: Héctor Pérez Tijero

Sistemas de Tiempo Real (Ingeniería Informática)

Seminario de Programación en Ada

UNIVERSIDAD DE CANTABRIA

Bloque I

- **Elementos básicos del lenguaje**

Introducción

UNIVERSIDAD DE CANTABRIA

Motivación para el lenguaje: "Crisis del software" de mediados de los años 70

- promovido por el Departamento de Defensa de EEUU
- gran número de lenguajes
- necesidad de introducir técnicas de ingeniería de software
  - fiabilidad
  - modularidad
  - programación orientada a objetos (comenzaba en ese momento)
- Ada en honor de Lady Augusta ADA Byron
- inspirado en PASCAL
- el objetivo era conseguir un lenguaje con posibilidades de convertirse en un estándar universal

Introducción (cont.)

UNIVERSIDAD DE CANTABRIA

Versiones:

- Ada 83: estándar de 1983
  - para propósito general
  - incluye sistemas de tiempo real
- Ada 95:
  - versión mejorada incluyendo programación orientada a objetos
- Ada 2005: revisión aprobada en 2006
  - añade mejoras en el uso de objetos
  - nuevas características para la programación de tiempo real

Principios de diseño del lenguaje Ada

UNIVERSIDAD DE CANTABRIA

- Fiabilidad
  - legibilidad
  - tipificación estricta
  - excepciones
- Modularidad
- Abstracción de datos y tipos
- Compilación separada
- Concurrencia
- Tiempo Real
- Estandarizado

Estructura de un programa

```
with Nombres_de_Otros_Modulos_de_Programa;
procedure Nombre_Programa is
   declaraciones;
begin
   instrucciones;
end Nombre_programa;
```

Las declaraciones son:

- de *datos*: constantes, variables, tipos
- de *fragmentos de programa*: procedimientos, funciones, paquetes

Ejemplo

UNIVERSIDAD DE CANTABRIA

```
with Ada.Text_IO;
procedure Sencillo is
   -- sin declaraciones
begin
   Ada.Text_IO.Put("Esto es un programa sencillo");
end Sencillo;
```

Comentarios sobre el ejemplo

UNIVERSIDAD DE CANTABRIA

- Concepto de sangrado:
  - legibilidad
  - muestra visualmente la estructura del código
- Comentarios: comienzan por `--` y acaban al final de la línea
- Las instrucciones y declaraciones acaban en "`;`". El programa también
- Los nombres de identificadores:
  - comienzan por letra
  - siguen letras, números y "_"
  - no se distinguen mayúsculas de minúsculas
  - no es aconsejable usar acentos y *ñ*
  - hay un conjunto de nombres reservados

Variables, constantes, y tipos simples

UNIVERSIDAD DE CANTABRIA

La información se guarda en casillas de memoria

- ***variables***: el contenido puede variar
  - algunas tienen nombre
  - otras se crean dinámicamente sin nombre (se verán más adelante)
- ***constantes***: el contenido no puede variar
  - las hay con nombre
  - y sin nombre (***literales***): se pone directamente el valor

Todos los datos tienen siempre un tipo asociado:

- tipos *predefinidos*
- tipos *definidos por el usuario*

Tipos predefinidos

UNIVERSIDAD DE CANTABRIA

Tipo	Valores
Integer	entero de 16 bits mínimo [-2**15..2**15-1]
Float	real de unos 6 dígitos como mínimo
Character	caracter de 8 bits (**Wide_Character** para 16 bits)
String (1..n)	texto o secuencia de caracteres
Boolean	**True** o **False**
Duration	número real en segundos

Atributos de los tipos predefinidos:

Tipo	Atributo	Descripción
enteros, reales y enumerados	**tipo'FIRST** **tipo'LAST** **tipo'IMAGE(numero)** **tipo'VALUE(string)**	**Primero valor** **Ultimo valor** **Conversión a String** **Conversión String a número**
reales	**tipo'DIGITS** **tipo'SMALL**	**Número de dígitos** **Menor valor positivo**
discretos (enteros, caracteres y enumerados)	**tipo'POS(valor)** **tipo'VAL(numero)**	**Código numérico de un valor** **Conversión de código a valor**
strings	**s'LENGTH (s es un string concreto)**	**Número de caracteres del string**

Componentes de los strings

UNIVERSIDAD DE CANTABRIA

Caracteres individuales

```
s(i)
```

Rodajas de string (también son strings)

```
s(i..j)
```

Constantes sin nombre o literales

UNIVERSIDAD DE CANTABRIA

Números enteros

```
13   0   -12   1E7    13_842_234
16#3F8#
```

Números reales

```
13.0    0.0    -12.0    1.0E7
```

Caracteres

```
'a'    'Z'
```

Strings

```
"texto"
```

Booleanos

```
True    False
```

Variables y constantes con nombre

UNIVERSIDAD DE CANTABRIA

Características

- Ocupan un lugar en la memoria
- tienen un tipo
- tienen un nombre : identificador
- las constantes no pueden cambiar de valor

Declaración de variables

```
identificador : tipo;
identificador : tipo := valor_inicial;
```

Declaración de constantes:

```
identificador : constant := valor_inicial;
identificador : constant tipo := valor_inicial;
```

Ejemplos

```
Numero_De_Cosas  : Integer;
Temperatura      : Float:=37.0;
Direccion        : String(1..30);
Esta_Activo      : Boolean;
Simbolo          : Character:='a';
A,B,C            : Integer;
Max_Num          : constant Integer:=500;
Pi               : constant:=3.1416;

Bytes_Per_Page   : constant := 512;
Pages_Per_Buffer : constant := 10;
Buffer_Size      : constant := Pages_Per_Buffer * Bytes_Per_Page;

Buffer_Size      : constant := 5_120;   -- ten pages
```

Ejemplo de un programa con datos

UNIVERSIDAD DE CANTABRIA

```
with Ada.Text_Io;

procedure Nombre is

   Tu_Nombre,Tu_Padre : String (1..20);
   N_Nombre,N_Padre   : Integer;
   -- Ejemplo de uso de strings de tamaño variable

begin
   Ada.Text_Io.Put("Cual es tu nombre?: ");
   Ada.Text_Io.Get_Line(Tu_Nombre,N_Nombre);
   Ada.Text_Io.Put("Como se llama tu padre?: ");
   Ada.Text_Io.Get_Line(Tu_Padre,N_Padre);
   Ada.Text_Io.Put_Line("El padre de "&Tu_Nombre(1..N_Nombre)&
                        " es "&Tu_Padre(1..N_Padre));
end Nombre;
```

El mismo ejemplo, con cláusula "use"

UNIVERSIDAD DE CANTABRIA

```
with Ada.Text_Io;
use Ada.Text_Io;
procedure Nombre_Con_Use is

   Tu_Nombre,Tu_Padre : String (1..20);
   N_Nombre,N_Padre   : Integer;
   -- Ejemplo de uso de strings de tamaño variable

begin
   Put("Cual es tu nombre?: ");
   Get_Line(Tu_Nombre,N_Nombre);
   Put("Como se llama tu padre?: ");
   Get_Line(Tu_Padre,N_Padre);
   Put_Line("El padre de "&Tu_Nombre(1..N_Nombre)&
            " es "&Tu_Padre(1..N_Padre));
end Nombre_Con_Use;
```

A observar

UNIVERSIDAD DE CANTABRIA

- Para strings de longitud variable
  - Usamos un string grande y de él sólo una parte
  - una variable entera nos dice cuántos caracteres útiles hay
- Uso de rodajas de string
  - para coger la parte útil del string
- Text_IO: `Put`, `Get_Line`, `Put_Line`, `New_Line`
- Cláusula `use`
- Uso de variables

Expresiones

UNIVERSIDAD DE CANTABRIA

Permiten transformar datos para obtener un resultado

Se construyen con

- operadores
  - dependen del tipo de dato
  - se pueden definir por el usuario
  - binarios (dos operandos) y unarios (un operando)
- operandos
  - variables
  - constantes
  - funciones

Precedencia de los operadores

UNIVERSIDAD DE CANTABRIA

La precedencia puede modificarse con el uso de paréntesis.

Operador	Operación	Operandos(s)	Resultado

`and`
`or`
`xor`
	y
	o inclusivo
	o exclusivo
		Boolean
		Boolean
		Boolean
			Boolean
			Boolean
			Boolean

`=`
`/=`
`<`
`<=`
`>`
`>=`
	igual a
	distinto de
	menor que
	menor o igual que
	mayor que
	mayor o igual que
		cualquiera no limitado
		cualquiera no limitado
		escalar o string
		escalar o string
		escalar o string
		escalar o string
			Boolean
			Boolean
			Boolean
			Boolean
			Boolean
			Boolean

`&`	Concatenación	Strings,string y carácter	String
`+` `-`	suma resta	numérico numérico	el mismo el mismo

Precedencia de los operadores (cont.)

Operador	Operación	Operandos(s)	Resultado
`+`	identidad	numérico	el mismo
`-`	negación	numérico	el mismo
`*`	multiplicación	Entero	Entero
		Real	Real
`/`	división	Entero	Entero
		Real	Real
`mod`	módulo	Entero	Entero
`rem`	resto	Entero	Entero
`**`	exponenciación	Entero, Entero no negativo	Entero
`**`	"	Real, Entero	Real
`not`	negación	Boolean	Boolean
`abs`	valor absoluto	numérico	el mismo

Ejemplos de expresiones

UNIVERSIDAD DE CANTABRIA

aritméticas

17-4*A2**

YN+8.0**

relacionales

X>3.0

N=28 -- compara N con 28. No confundir con la asignación

booleanas

X>3.0 and x<8.0 -- true si X es mayor que 3 y menor que 8

inclusión

A in 1..20 -- true o false según A esté o no entre 1 y 20

concatenación

Mi_Nombre&" texto añadido" -- el resultado es un nuevo string

Tipificación estricta

UNIVERSIDAD DE CANTABRIA

No podemos mezclar datos de distinto tipo:

```
I, J : Integer;
X, Y : Float;
I+3*J -- expresión entera
I+X   -- expresión ilegal
```

Conversión de tipos:

```
Tipo(valor)

I+Integer(X)   -- expresión entera
Float(I)+X     -- expresión real
```

Ojo con las operaciones de división

```
3/10      -- vale cero (división entera)
3.0/10.0  -- vale 0.3
```

Ejemplo de un programa con expresiones

UNIVERSIDAD DE CANTABRIA

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

procedure Nota_Media is
   Nota1,Nota2,Nota3,Nota_Media : Integer;
begin
   Put("Nota del primer trimestre: ");
   Get(Nota1); Skip_Line;
   Put("Nota del segundo trimestre: ");
   Get(Nota2); Skip_Line;
   Put("Nota del tercer trimestre: ");
   Get(Nota3); Skip_Line;
   Nota_Media := (Nota1+Nota2+Nota3)/3;
   Put("Nota Media : ");
   Put(Nota_Media); New_Line;
   Put("Nota Media (otra) : ");
   Put(Float(Nota1+Nota2+Nota3)/3.0);
   New_Line;
end Nota_Media;
```

Mismo ejemplo con el atributo 'IMAGE o 'IMG

UNIVERSIDAD DE CANTABRIA

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

procedure Nota_Media is

   Nota1,Nota2,Nota3,Nota_Media : Integer;

begin
   Put("Nota del primer trimestre: ");
   Get(Nota1); Skip_Line;
   Put("Nota del segundo trimestre: ");
   Get(Nota2); Skip_Line;
   Put("Nota del tercer trimestre: ");
   Get(Nota3); Skip_Line;
   Nota_Media := (Nota1+Nota2+Nota3)/3;
   Put_Line("Nota Media : " & Nota_Media'Img));
   Put_Line("Nota Media (otra) : "&
            Float'Image(Float(Nota1+Nota2+Nota3)/3.0));
end Nota_Media;
```

A destacar

UNIVERSIDAD DE CANTABRIA

Uso de `Put` y `Get` para enteros y reales

Uso del `Skip_Line` detrás de cada "`Get`" en el teclado (pero no detrás de `Get_Line`)

- **Aunque a veces no es necesario, otras sí (p.e., cuando después hay un Get_Line)**
- **Es conveniente acostumbrarse a ponerlo siempre.**

También es cómodo usar el atributo:

- **`'Image` al escribir cuando se usa el tipo de dato**
- **`'Img` al escribir cuando se usa el identificador**
- **`'Value` al leer**

Instrucciones de control

UNIVERSIDAD DE CANTABRIA

Permiten modificar el flujo de ejecución del programa

Son:

- **Instrucciones condicionales**
  - **`if` (simple, doble, múltiple)**
  - **`case`: condición discreta (múltiple)**
- **Instrucciones de lazo (`loop`)**
  - **`for`: número de veces conocido**
  - **`while`: condición de salida al principio**
  - **otras condiciones de salida (`exit`)**

Instrucción condicional lógica (if)

UNIVERSIDAD DE CANTABRIA

Forma simple:

```
if exp_logica then
   instrucciones;
end if;
```

Forma doble:

```
if exp_logica then
   instrucciones;
else
   instrucciones;
end if;
```

Forma múltiple:

```
if exp_logica then
   instrucciones;
elsif exp_logica then
   instrucciones;
elsif exp_logica then
   instrucciones
...
else
   instrucciones;
end if;
```

Ejemplo: cálculo del máximo de A,B y C:

UNIVERSIDAD DE CANTABRIA

```
if A>B then
   max:=A;
else
   max:=B;
end if;
if max<C then
   max:=C;
end if;
```

Nota: Evaluación condicional de expresiones booleanas

UNIVERSIDAD DE CANTABRIA

Las expresiones lógicas normales evalúan las dos partes de la expresión

```
if j>0 and i/j>k then ...
```

Posibilidad de error si j=0. Solución para evaluar la segunda parte sólo si se cumple la primera:

```
if j>0 and then i/j>k then ...
```

Lo mismo se puede hacer con la operación "or":

```
if j>0 or else abs(j)<3 then ...
```

La evaluación condicional es más eficiente

Instrucción condicional discreta (case)

UNIVERSIDAD DE CANTABRIA

Para decisiones que dependen de una expresión discreta se usa la instrucción case

- **más elegante**
- **posiblemente más eficiente**

Las expresiones discretas son

- **enteros**
- **caracteres**
- **booleanos**
- **enumerados**

pero no los números reales.

Instrucción case

UNIVERSIDAD DE CANTABRIA

```
case exp_discreta is
   when valor1 =>
        instrucciones;
   when valor2 =>
        instrucciones;
   when valor3 | valor4 | valor5 =>
        instrucciones;
   when valor6..valor7 =>
        instrucciones;
   when others =>
        instrucciones;
end case;
```

Requisitos:

- La cláusula `others` es opcional, pero si no aparece, es obligatorio cubrir todos los posibles valores
- Si no se desea hacer nada poner la instrucción `null`

Ejemplo: Poner la nota media con letra

UNIVERSIDAD DE CANTABRIA

```
Nota_Media : Integer:=...;

case Nota_Media is
   when 0..4   => Put_Line("Suspenso");
   when 5..6   => Put_Line("Aprobado");
   when 7..8   => Put_Line("Notable");
   when 9..10  => Put_Line("Sobresaliente");
   when others => Put_Line("Error");
end case;
```

El mismo ejemplo si la nota es un real

UNIVERSIDAD DE CANTABRIA

```
Nota_Media : Float:=...;

if Nota_Media<0.0 then
   Put_Line("Error");
elsif Nota_Media<5.0 then
   Put_Line("Suspenso");
elsif Nota_Media<7.0 then
   Put_Line("Aprobado");
elsif Nota_Media<9.0 then
   Put_Line("Notable");
elsif Nota_Media<=10.0 then
   Put_Line("Sobresaliente");
else
   Put_Line("Error");
end if;
```

Instrucciones de lazo

UNIVERSIDAD DE CANTABRIA

Hay varias instrucciones de lazo, según el número de veces que se quiera hacer el lazo:

- ***indefinido*: lazo infinito**
- ***número máximo de veces conocido* al ejecutar: lazo con variable de control**
- ***número máximo de veces no conocido*:**
  - **el lazo se hace *0 o más veces*: condición de permanencia al principio**
  - **el lazo se hace *1 o más veces*: condición de salida al final.**

Lazo infinito

UNIVERSIDAD DE CANTABRIA

Ejemplo:

```
loop
   Put_Line("No puedo parar");
   -- más instrucciones
end loop;
```

Lazo con variable de control (for)

UNIVERSIDAD DE CANTABRIA

Una variable discreta toma todos los valores de un rango:

```
for var in rango loop
  instrucciones;
end loop;
```

El rango se escribe de una de las siguientes maneras:

```
valor_inicial..valor_final
tipo
tipo range valor_inicial..valor_final
```

Comentarios:

- La variable se declara en la instrucción y sólo existe durante el lazo
- Toma los valores del rango: [inicial...final], uno por uno
- No se puede cambiar su valor

Lazo con variable de control (cont.)

UNIVERSIDAD DE CANTABRIA

Los valores se pueden recorrer en orden inverso:

```
for i in reverse rango loop
```

Ojo. No es lo mismo:

```
reverse 1..10
10..1        -- rango nulo !
```

Ejemplo: Suma de los 100 primeros números

```
Suma : Integer:=0;

for i in 1..100 loop
   Suma:=Suma+i;
end loop;
```

Lazo con condición de permanencia al principio (while)

UNIVERSIDAD DE CANTABRIA

Sintaxis:

```
while exp_logica loop
  instrucciones;
end loop;
```

Ejemplo: calcular cuántos números enteros pares hay que sumar, empezando en uno, para superar el valor 100.

```
J: Integer:=0;
Suma : Integer:=0;

while Suma<=100 loop
  J:=J+2;
  Suma:=Suma+J;
end loop;
```

Lazo con condición de salida en cualquier lugar

UNIVERSIDAD DE CANTABRIA

Instrucciones `exit`:

```
exit;
exit when condicion;
```

Ejemplo: Calcular la suma de la serie hasta que el término sumado sea menor que 10^{-6}, partiendo de un valor x=1.2

$$\frac{(x-1)}{x} + \frac{(x-1)}{x^2} + \frac{(x-1)}{x^3} + \ldots$$

Ejemplo de lazo con condición de salida al final

UNIVERSIDAD DE CANTABRIA

```
X : Float:=1.2;
Suma : Float:=0.0;     -- elemento neutro de la suma
Termino : Float;       -- no requiere valor inicial
Potencia : Float:=1.0; -- elemento neutro del producto

loop
   Potencia:=Potencia*X; -- es más eficiente "*" que "**"
   Termino:=(X-1.0)/Potencia;
   Suma:=Suma+Termino;
   exit when Termino<1.0E-6;
end loop;
```

Ejemplo con instrucciones condicionales y de lazo

UNIVERSIDAD DE CANTABRIA

Calcular el máximo de un conjunto de valores reales introducidos por teclado

Pseudocódigo:

```
Leer el número de valores
for i desde 1 hasta n
  leer num
  si num> maximo
     maximo=num
  fin si
fin lazo
```

Ejemplo con instrucciones condicionales y de lazo (cont.)

UNIVERSIDAD DE CANTABRIA

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

procedure Maximo is
   Maximo    : Float := Float'First;
   X         : Float;
   Num_Veces : Integer;
begin
   Put ("Numero de valores: ");
   Get (Num_Veces);
   Skip_Line;
   for I in 1..Num_Veces loop
      Put ("Valor : ");
      Get (X);
      Skip_Line;
      if X>Maximo then
         Maximo:= X;
      end if;
   end loop;
   Put ("El maximo es : ");
   Put (Maximo);
   New_Line;
end Maximo;
```