Seminario de Programación en Ada

UNIVERSIDAD DE CANTABRIA

Anexo

- **Librerías predefinidas**

Librería de Servicios Numéricos

UNIVERSIDAD DE CANTABRIA

```
package Ada.Numerics is
pragma Pure (Numerics);

   Argument_Error : exception;

   Pi : constant :=
          3.14159_26535_89793_23846_26433_
            83279_50288_41971_69399_37511;

   e : constant :=
          2.71828_18284_59045_23536_02874_
            71352_66249_77572_47093_69996;

end Ada.Numerics;
```

Librería estándar de funciones matemáticas

UNIVERSIDAD DE CANTABRIA

```
package Ada.Numerics.Elementary_Functions is

   function Sqrt     (X           : Float) return Float;
   function Log      (X           : Float) return Float;
   function Log      (X, Base     : Float) return Float;
   function Exp      (X           : Float) return Float;
   function "**"     (Left, Right : Float) return Float;

   function Sin      (X           : Float) return Float;
   function Sin      (X, Cycle    : Float) return Float;
   function Cos      (X           : Float) return Float;
   function Cos      (X, Cycle    : Float) return Float;
   function Tan      (X           : Float) return Float;
   function Tan      (X, Cycle    : Float) return Float;
   function Cot      (X           : Float) return Float;
   function Cot      (X, Cycle    : Float) return Float;
```

Librería estándar de funciones matemáticas (cont.)

UNIVERSIDAD DE CANTABRIA

```
function Arcsin   (X           : Float) return Float;
function Arcsin   (X, Cycle    : Float) return Float;
function Arccos   (X           : Float) return Float;
function Arccos   (X, Cycle    : Float) return Float;

function Arctan
  (Y : Float; X : Float := 1.0) return Float;

function Arctan
  (Y : Float; X : Float := 1.0; Cycle : Float) return  Float;

function Arccot
  (X : Float; Y : Float := 1.0) return Float;

function Arccot
  (X : Float; Y : Float := 1.0; Cycle : Float) return  Float;
```

Librería estándar de funciones matemáticas (cont.)

UNIVERSIDAD DE CANTABRIA

```
   function Sinh     (X : Float) return Float;
   function Cosh     (X : Float) return Float;
   function Tanh     (X : Float) return Float;
   function Coth     (X : Float) return Float;
   function Arcsinh  (X : Float) return Float;
   function Arccosh  (X : Float) return Float;
   function Arctanh  (X : Float) return Float;
   function Arccoth  (X : Float) return Float;

end Ada.Numerics.Elementary_Functions;
```

Ejemplos de uso de las funciones matemáticas

UNIVERSIDAD DE CANTABRIA

```
with Ada.Numerics.Elementary_Functions;
use  Ada.Numerics.Elementary_Functions;

procedure Prueba is

   A,B,C : Float;

begin
   A:=Sqrt(2.13);
   B:=Log(A);
   C:=Sin(B);
   A:=Sin(B,360.0);
   B:=A**C;
end Prueba;
```

Paquetes de librerías estándares: Números aleatorios

UNIVERSIDAD DE CANTABRIA

```
package Ada.Numerics.Float_Random is

   --  Basic facilities

   type Generator is limited private;

   subtype Uniformly_Distributed is
         Float range 0.0 .. 1.0;

   function Random (Gen : Generator)
         return Uniformly_Distributed;

   procedure Reset   (Gen       : Generator);
   procedure Reset   (Gen       : Generator;
                      Initiator : Integer);

   --  Advanced facilities
   ...
end Ada.Numerics.Float_Random;
```

Ejemplo de uso de números aleatorios

UNIVERSIDAD DE CANTABRIA

```
with Ada.Numerics.Float_Random, Text_Io, Ada.Integer_Text_Io;
use Ada.Numerics.Float_Random, Text_Io, Ada.Integer_Text_Io;
procedure Dado is

   Gen : Generator;
   Pulsada : Boolean;
   C : Character;

begin
   Put_Line("Pulsa una tecla para parar");
   Reset(Gen);
   loop
      Put(Integer(6.0*Random(Gen)+0.5));
      New_Line;
      delay 0.5;
      Get_Immediate(C,Pulsada);
      exit when Pulsada;
   end loop;
end Dado;
```

Paquete Ada.Exceptions (1/2)

UNIVERSIDAD DE CANTABRIA

```
package Ada.Exceptions is

   type Exception_Id is private;
   Null_Id : constant Exception_Id;

   function Exception_Name
        (X : Exception_Id) return String;

   type Exception_Occurrence is
         limited private;

   function  Exception_Message
        (X : Exception_Occurrence)
         return String;

   function Exception_Identity
        (X : Exception_Occurrence)
         return Exception_Id;
```

Paquete Ada.Exceptions (2/2)

UNIVERSIDAD DE CANTABRIA

```
   function Exception_Name
         (X : Exception_Occurrence)
          return String;

   function Exception_Information
         (X : Exception_Occurrence)
          return String;

   ...

private
   ...
end Ada.Exceptions;
```

Paquete Ada.Calendar

UNIVERSIDAD DE CANTABRIA

```
package Ada.Calendar is

   type Time is private;
   subtype Year_Number  is Integer range 1901 .. 2099;
   subtype Month_Number is Integer range 1 .. 12;
   subtype Day_Number   is Integer range 1 .. 31;
   subtype Day_Duration is Duration range 0.0 .. 86_400.0;

   function Clock return Time;
   function Year    (Date : Time) return Year_Number;
   function Month   (Date : Time) return Month_Number;
   function Day     (Date : Time) return Day_Number;
   function Seconds (Date : Time) return Day_Duration;
   function Time_Of
      (Year    : Year_Number;
       Month   : Month_Number;
       Day     : Day_Number;
       Seconds : Day_Duration := 0.0)
       return Time;
```

Paquete Ada.Calendar

```
   function "+" (Left : Time;     Right : Duration) return Time;
   function "+" (Left : Duration; Right : Time)     return Time;
   function "-" (Left : Time;     Right : Duration) return Time;
   function "-" (Left : Time;     Right : Time)     return Duration;
   ...

end Ada.Calendar;
```

Ejemplo

UNIVERSIDAD DE CANTABRIA

```
-- Imprimir en pantalla el día, mes y año actuales,
-- y luego las horas y minutos

with Ada.Calendar, Ada.Text_Io;
use Ada.Calendar; use Ada.Text_Io;
procedure Muestra_Dia_Y_Hora is
   Instante : Time := Clock;
   Hora     : Integer := Integer(Seconds(Instante))/3600;
   Minuto   : Integer := (Integer(Seconds(Instante))-Hora*3600)/60;
begin
   Put_Line("Hoy es "&Integer'Image(Day(Instante))&" del "&
            Integer'Image(Month(Instante))&" de "&
            Integer'Image(Year(Instante)));
   Put_Line("La hora es : "&Integer'Image(Hora)&
            ":"&Integer'Image(Minuto));
end Muestra_Dia_Y_Hora;
```

Paquete Ada.Real_Time

UNIVERSIDAD DE CANTABRIA

```
package Ada.Real_Time is
   type Time is private;
   Time_First : constant Time;
   Time_Last  : constant Time;
   Time_Unit  : constant := -- real number;
   type Time_Span is private;
   Time_Span_First : constant Time_Span;
   Time_Span_Last  : constant Time_Span;
   ...
   function Clock return Time;

   function "+" (Left : Time; Right : Time_Span) return Time;
   function "+" (Left : Time_Span; Right : Time) return Time;
   function "-" (Left : Time; Right : Time_Span) return Time;
   function "-" (Left : Time; Right : Time) return Time_Span;
   ...
   function To_Duration (TS : Time_Span) return Duration;
   function To_Time_Span (D : Duration) return Time_Span;

   function Nanoseconds (NS : integer) return Time_Span;
   function Microseconds (US : integer) return Time_Span;
   function Milliseconds (MS : integer) return Time_Span;
end Ada.Real_Time;
```