Seminario de Programación en Ada

UNIVERSIDAD DE CANTABRIA

Anexo

- **Entrada/salida con ficheros**

Introducción

UNIVERSIDAD DE CANTABRIA

Hay dos tipos de entrada/salida según el tipo de información:

- de *texto*
  - pensada para los “humanos”
  - restringida a datos que se puedan convertir a texto (strings, caracteres, números, enumerados)
  - basada en caracteres
- *binaria*
  - pensada para las “máquinas”
  - para cualquier tipo de dato, excepto punteros
  - las estructuras con punteros deben ser “serializadas”
  - basadas en secuencias de bytes

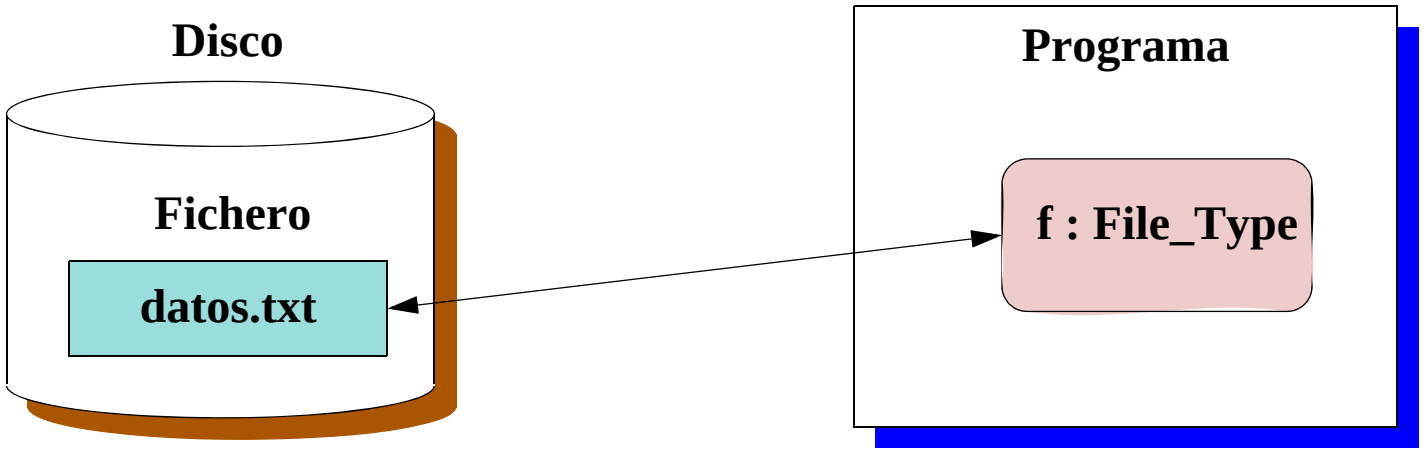
Ficheros

Representan una estructura de datos almacenada en memoria secundaria o un dispositivo de entrada/salida

- **Si es memoria secundaria, los datos son *no volátiles***

Se identifican en el sistema operativo mediante un *nombre*

Se representan en el programa mediante una estructura de datos de un tipo `File_Type`



Uso de ficheros

UNIVERSIDAD DE CANTABRIA

El uso siempre es así:

- ***Abrir*** **el fichero para establecer una asociación entre la estructura de datos y el fichero**
- ***Leer*** **y/o** ***escribir***
- ***Cerrar*** **el fichero para desvincular la asociación y dejar los datos en un estado consistente**

Hay unos ficheros de texto ya abiertos, para uso habitual

- **entrada estándar (generalmente el teclado)**
- **salida estándar (generalmente la pantalla)**
- **salida de error (generalmente la pantalla)**

Se pueden modificar haciendo que sean otros ficheros

Interfaz para los ficheros

UNIVERSIDAD DE CANTABRIA

Abrir

- 2 modalidades:
  - `Create`: para crear un fichero nuevo (se borra el viejo)
  - `Open`: para ficheros existentes (si no existe, falla)
- requiere indicar el modo (del tipo `File_Mode`)
  - `In_File` (leer), `Out_File` (escribir), `Append_file` (añadir al final), `Inout_File` (leer y escribir)
- también requiere la variable del tipo `File_Type` y el nombre del fichero (string)

Cerrar: `Close`

Leer y *escribir*: depende del tipo de fichero

Entrada/salida de texto

UNIVERSIDAD DE CANTABRIA

Está en el paquete `Ada.Text_IO`

Ya la conocemos para entrada/salida estándar

- ej.: operaciones `Get`, `Put`, `Get_Line`, `Put_Line`

Existen versiones de estas operaciones para otros ficheros

- el primer parámetro es la variable `File_Type`

Excepciones en la entrada/salida

UNIVERSIDAD DE CANTABRIA

Excepción	Causa
`Status_Error`	Abrir un fichero ya abierto, cerrar uno ya cerrado, usar uno no abierto
`Mode_Error`	Leer si se ha abierto para escribir, escribir si se ha abierto para leer
`Name_Error`	Al hacer `Open`, el fichero no existe
`Use_Error`	Uso del fichero incompatible con las restricciones del sistema operativo (p.e., usar sin permiso)
`Device_Error`	Error en el dispositivo (p.e., disquete malo)
`End_Error`	Intentar leer pasado el final del fichero
`Data_Error`	Formato de los datos incorrecto (p.e, encontrar letras al leer un número)
`Layout_error`	Error con el formato de líneas (líneas demasiado largas)

Tipos de entrada/salida binaria

UNIVERSIDAD DE CANTABRIA

3 modalidades

- casillas uniformes (siempre del mismo tipo)
  - entrada/salida *secuencial* (`Ada.Sequential_IO`)
  - entrada/salida *directa*, con acceso aleatorio (similar a un array, pero en disco, en `Ada.Direct_IO`)
- casillas de tipos diferentes
  - entrada/salida de *streams*
  - es secuencial
  - es fácil equivocarse, si se leen datos en orden distinto a como se han escrito

Paquete Ada.Sequential_IO (1/3)

UNIVERSIDAD DE CANTABRIA

```
with Ada.IO_Exceptions;
generic
   type Element_Type (<>) is private;

package Ada.Sequential_IO is

   type File_Type is limited private;

   type File_Mode is (In_File, Out_File, Append_File);

   -- File management
   procedure Create (File : in out File_Type;
                     Mode : in File_Mode := Out_File;
                     Name : in String := "";
                     Form : in String := "");
   procedure Open   (File : in out File_Type;
                     Mode : in File_Mode;
                     Name : in String;
                     Form : in String := "");
```

Paquete Ada.Sequential_IO (2/3)

```
procedure Close  (File : in out File_Type);
procedure Delete (File : in out File_Type);
procedure Reset  (File : in out File_Type; Mode : in File_Mode);
procedure Reset  (File : in out File_Type);

function Mode    (File : in File_Type) return File_Mode;
function Name    (File : in File_Type) return String;
function Form    (File : in File_Type) return String;

function Is_Open (File : in File_Type) return Boolean;

-- Input and output operations

procedure Read  (File : in File_Type; Item : out Element_Type);
procedure Write (File : in File_Type; Item : in Element_Type);

function End_Of_File (File : in File_Type) return Boolean;
```

Paquete Ada.Sequential_IO (3/3)

UNIVERSIDAD DE CANTABRIA

```
   -- Exceptions

   Status_Error : exception renames IO_Exceptions.Status_Error;
   Mode_Error   : exception renames IO_Exceptions.Mode_Error;
   Name_Error   : exception renames IO_Exceptions.Name_Error;
   Use_Error    : exception renames IO_Exceptions.Use_Error;
   Device_Error : exception renames IO_Exceptions.Device_Error;
   End_Error    : exception renames IO_Exceptions.End_Error;
   Data_Error   : exception renames IO_Exceptions.Data_Error;

private
   ...
end Ada.Sequential_IO;
```

Ejemplo de entrada/salida secuencial (1/4)

UNIVERSIDAD DE CANTABRIA

```
with Alumnos;

package Clases is

   ...

   procedure Escribe
     (La_Clase   : in Clase;
      En         : in String);
   procedure Lee
     (La_Clase   : out Clase;
      En         : in  String);

private
   ...
end Clases;
```

Ejemplo de entrada/salida secuencial (2/4)

UNIVERSIDAD DE CANTABRIA

```
package body Clases is

   ...

   procedure Escribe (La_Clase   : in Clase;
                      En         : in String)
   is
     package Alumno_Io is new Ada.Sequential_Io(Alumnos.Alumno);
      Fichero : Alumno_Io.File_Type;
   begin
      Alumno_Io.Create(File => Fichero,
                       Name => En);
      for I in 1..La_Clase.Num loop
         Alumno_Io.Write(Fichero,La_Clase.Alumnos(I));
      end loop;
      Alumno_Io.Close(Fichero);
   end Escribe;
```

Ejemplo de entrada/salida secuencial (3/4)

UNIVERSIDAD DE CANTABRIA

```
   procedure Lee (La_Clase   : out Clase;
                  En         : in String)
   is
      package Alumno_Io is new Ada.Sequential_Io(Alumnos.Alumno);
      Fichero : Alumno_Io.File_Type;
   begin
      La_Clase.Num:=0;
      Alumno_Io.Open(File => Fichero,
                     Mode => Alumno_Io.In_File,
                     Name => En);
      while not Alumno_Io.End_Of_File(Fichero) loop
         La_Clase.Num:=La_Clase.Num+1;
         Alumno_Io.Read(Fichero,La_Clase.Alumnos(La_Clase.Num));
      end loop;
      Alumno_Io.Close(Fichero);
   exception
      when Alumno_Io.Name_Error =>null;
   end Lee;

end Clases;
```

Ejemplo de entrada/salida secuencial (4/4)

UNIVERSIDAD DE CANTABRIA

```
with Menu,Alumnos,Clases;
procedure Lista_Alumnos is

   Tercero_B  : Clases.Clase;
   Eleccion   : Menu.Opcion;
   Alu : Alumnos.Alumno;
   Num : Clases.Num_Alumno;

begin
   Clases.Lee(Tercero_B,"datos.dat");
   loop
      Menu.Pide_Opcion (Eleccion);
      case Eleccion is
         when Menu.Insertar => ...
         when Menu.Mirar => ...
         when Menu.Salir => exit;
      end case;
   end loop;
   Clases.Escribe(Tercero_B,"datos.dat");
end Lista_Alumnos;
```

Paquete Ada.Direct_IO (1/4)

```
with Ada.IO_Exceptions;
generic
   type Element_Type is private;
package Ada.Direct_IO is

   type File_Type is limited private;
   type File_Mode is (In_File, Inout_File, Out_File);
   type Count is new System.Direct_IO.Count;
   subtype Positive_Count is Count range 1 .. Count'Last;

    -- File Management --
   procedure Create (File : in out File_Type;
                     Mode : in File_Mode := Inout_File;
                     Name : in String := "";
                     Form : in String := "");
   procedure Open   (File : in out File_Type;
                     Mode : in File_Mode;
                     Name : in String;
                     Form : in String := "");
```

Paquete Ada.Direct_IO (2/4)

```
procedure Close  (File : in out File_Type);
procedure Delete (File : in out File_Type);
procedure Reset  (File : in out File_Type; Mode : in File_Mode);
procedure Reset  (File : in out File_Type);

function Mode (File : in File_Type) return File_Mode;
function Name (File : in File_Type) return String;
function Form (File : in File_Type) return String;

function Is_Open (File : in File_Type) return Boolean;

-- Input and Output Operations

procedure Read (File : in File_Type;
                Item : out Element_Type;
                From : in Positive_Count);

procedure Read (File : in File_Type;
                Item : out Element_Type);
```

Paquete Ada.Direct_IO (3/4)

```
procedure Write (File : in File_Type;
                 Item : in Element_Type;
                 To   : in Positive_Count);

procedure Write (File : in File_Type;
                 Item : in Element_Type);

procedure Set_Index (File : in File_Type;
                     To   : in Positive_Count);

function Index (File : in File_Type) return Positive_Count;
function Size  (File : in File_Type) return Count;

function End_Of_File (File : in File_Type) return Boolean;
```

Paquete Ada.Direct_IO (4/4)

```
   -- Exceptions

   Status_Error : exception renames IO_Exceptions.Status_Error;
   Mode_Error   : exception renames IO_Exceptions.Mode_Error;
   Name_Error   : exception renames IO_Exceptions.Name_Error;
   Use_Error    : exception renames IO_Exceptions.Use_Error;
   Device_Error : exception renames IO_Exceptions.Device_Error;
   End_Error    : exception renames IO_Exceptions.End_Error;
   Data_Error   : exception renames IO_Exceptions.Data_Error;

private
   ...
end Ada.Direct_IO;
```

Ejemplo de Entrada/Salida Directa (1/7)

UNIVERSIDAD DE CANTABRIA

Hacer un programa para anotar en un fichero las reservas de unas aulas

- el aula se identifica por su número
  - será el índice de la casilla del fichero
- la reserva se representa por el nombre de la persona

Daremos las opciones de consultar, hacer reserva y salir, usando el menú genérico:

```
generic
   type Opcion is (<>);
function Menu_Generico return Opcion;
```

Ejemplo de Entrada/Salida Directa (2/7)

UNIVERSIDAD DE CANTABRIA

```
with Ada.Text_IO, Ada.Integer_Text_IO;
use Ada.Text_IO, Ada.Integer_Text_IO;
function Menu_Generico return Opcion is
   Num_Opcion : Integer;
begin
   -- Poner el menú en pantalla
   New_Line(2);
   Put_Line("-------------Menu------------");
   for Op in Opcion loop
      Put_Line(Integer'Image(Opcion'Pos(Op)+1)&" - "&
               Opcion'Image(Op));
   end loop;
```

Ejemplo de Entrada/Salida Directa (3/7)

UNIVERSIDAD DE CANTABRIA

```
   -- pedir y retornar la opcion deseada
   loop
      begin
         New_Line;
         Put("Introduce opcion deseada : ");
         Get(Num_Opcion); Skip_Line;
         return Opcion'Val(Num_Opcion-1);
      exception
         when Data_Error|Constraint_Error =>
            Skip_Line;
            Put_Line("Error al leer la opcion");
      end;
   end loop;
end Menu_Generico;
```

Ejemplo de Entrada/Salida Directa (4/7)

UNIVERSIDAD DE CANTABRIA

```
with Ada.Direct_Io,Ada.Text_Io, Ada.Integer_Text_IO,
  Var_Strings, Menu_Generico;
use Ada.Text_Io, Ada.Integer_Text_IO, Var_Strings;

procedure Reserva is

   Max_Aulas : constant Integer:=25;

   subtype Aula is Integer range 1..Max_Aulas;
   subtype Persona is Var_String;

   package Reserva_IO is new
     Ada.Direct_IO(Persona);

   procedure Modifica
     (A : Aula; P : Persona; F : in out Reserva_IO.File_Type)
   is
   begin
      Reserva_IO.Write(F,P,Reserva_IO.Count(A));
   end Modifica;
```

Ejemplo de Entrada/Salida Directa (5/7)

UNIVERSIDAD DE CANTABRIA

```
procedure Lee
  (A : Aula; P : out Persona; F : in out Reserva_IO.File_Type) is
begin
   Reserva_IO.Read(F,P,Reserva_IO.Count(A));
end Lee;

procedure Lee_Aula(A : out Aula) is
begin
   loop
      begin
         Put("Aula: ");
         Get(A); Skip_Line;
         exit;
      exception
         when Data_Error|Constraint_Error =>
            Skip_Line;
            Put_Line("Error al leer el aula");
      end;
   end loop;
end Lee_Aula;
```

Ejemplo de Entrada/Salida Directa (6/7)

UNIVERSIDAD DE CANTABRIA

```
   type Opcion is (Mirar_Reserva, Hacer_Reserva, Salir);

   function Pide_Opcion is new Menu_Generico(Opcion);

   -- variables del programa

   F : Reserva_IO.File_Type;
   A : Aula;
   P : Persona;

begin
   begin
      Reserva_IO.Open (F,Reserva_IO.Inout_File,"reserva.dat");
   exception
      when Reserva_IO.Name_Error =>
         Reserva_IO.Create (F,Reserva_IO.Inout_File,"reserva.dat");
         for A in Aula loop
            Modifica(A,To_Var_String("Nadie"),F);
         end loop;
   end;
```

Ejemplo de Entrada/Salida Directa (7/7)

UNIVERSIDAD DE CANTABRIA

```
   loop
      case Pide_Opcion is
         when Mirar_Reserva =>
            Lee_Aula(A);
            Lee(A,P,F);
            Put_Line("La persona es : "&P);
         when Hacer_Reserva =>
            Lee_Aula(A);
            Put("Introduce nombre persona : ");
            Get_Line(P);
            Modifica(A,P,F);
         when Salir =>
           exit;
      end case;
   end loop;
   Reserva_IO.Close(F);
end Reserva;
```

Resumen de entrada/salida de Streams (1/3)

UNIVERSIDAD DE CANTABRIA

```
with Ada.IO_Exceptions;

package Ada.Streams.Stream_IO is

   type Stream_Access is access all Root_Stream_Type'Class;
   type File_Type is limited private;
   type File_Mode is (In_File, Out_File, Append_File);

   procedure Create
     (File : in out File_Type;
      Mode : in File_Mode := Out_File;
      Name : in String := "";
      Form : in String := "");

   procedure Open
     (File : in out File_Type;
      Mode : in File_Mode;
      Name : in String;
      Form : in String := "");
```

Resumen de entrada/salida de Streams (2/3)

UNIVERSIDAD DE CANTABRIA

```
   procedure Close  (File : in out File_Type);
   procedure Delete (File : in out File_Type);
   procedure Reset  (File : in out File_Type; Mode : in File_Mode);
   procedure Reset  (File : in out File_Type);

   function Mode (File : in File_Type) return File_Mode;
   function Name (File : in File_Type) return String;
   function Form (File : in File_Type) return String;

   function Is_Open     (File : in File_Type) return Boolean;
   function End_Of_File (File : in File_Type) return Boolean;

   function Stream (File : in File_Type) return Stream_Access;

private
   ...
end Ada.Streams.Stream_IO;
```

Resumen de entrada/salida de Streams (3/3)

UNIVERSIDAD DE CANTABRIA

Para leer datos de un tipo determinado:

Tipo'Input(Stream:Stream_Access, Dato:Tipo);

Para escribir datos de un tipo determinado:

Tipo'Output(Stream:Stream_Access, Dato:Tipo);

Para que la lectura sea correcta debe hacerse en el mismo orden que la escritura

- **Por ejemplo si hemos escrito datos de los tipos T1, T2, y T3, luego deben leerse datos de esos mismos tipos y en ese orden**

Ejemplo de entrada/salida de Streams (1/3)

UNIVERSIDAD DE CANTABRIA

```
package Datos_Prueba_Streams is

   type Registro is record
      A,B,C : Integer;
      X : Float;
   end record;

   type Vector_3d is array(1..3) of Float;

end Datos_Prueba_Streams;
```

Ejemplo de entrada/salida de Streams (2/3)

UNIVERSIDAD DE CANTABRIA

```
with Ada.Streams, Ada.Streams.Stream_IO; use Ada.Streams;
with Datos_Prueba_Streams; use Datos_Prueba_Streams;

procedure Prueba_Escribir_Stream is
   F : Stream_IO.File_Type;
   I : Integer:=12;
   X : Float:=32.0;
   R : Registro:=(1,2,3,3.0);
   V : Vector_3d:=(45.0,34.0,-34.0);
   Strm : Stream_IO.Stream_Access;

begin
   Stream_IO.Create(F,Stream_IO.Out_File,"stream.dat");
   Strm:=Stream_IO.Stream(F);
   Integer'Output  (Strm,I);
   Float'Output    (Strm,X);
   Registro'Output (Strm,R);
   Vector_3d'Output(Strm,V);
   Stream_IO.Close(F);
end Prueba_Escribir_Stream;
```

Ejemplo de entrada/salida de Streams (3/3)

UNIVERSIDAD DE CANTABRIA

```
with Ada.Streams, Ada.Streams.Stream_IO; use Ada.Streams;
with Datos_Prueba_Streams; use Datos_Prueba_Streams;

procedure Prueba_Leer_Stream is
   F : Stream_IO.File_Type;
   I : Integer;
   X : Float;
   R : Registro;
   V : Vector_3d;
   Strm : Stream_IO.Stream_Access;

begin
   Stream_IO.Open(F,Stream_IO.In_File,"stream.dat");
   Strm:=Stream_IO.Stream(F);
   I:=Integer'Input  (Strm);
   X:=Float'Input    (Strm);
   R:=Registro'Input (Strm);
   V:=Vector_3d'Input(Strm);
   Stream_IO.Close(F);
end Prueba_Leer_Stream;
```