d_byte v0.2

Paquete para el manejo de **b**ytes en Ada95

Documentación

1- Objetivos v0.2

- Paquete de implementación de un TAD para el manejo de bytes y bits
- Conversión entre bytes, bits, enteros cortos y caracteres.
- Operaciones de lógica booleana
- Manejo de ficheros binarios directos y secuenciales.
- Implementación de operaciones de entrada y salida para ficheros binarios.
- Soporte de excepciones.

2- Especificación del TAD

- Conjuntos de valores

Bit – Conjunto característico formado por un 1 o un 0.
Byte – Conjunto formado por la agrupación de 8 bits, uno detrás de otro.
Position – Conjunto que representa las posiciones de los bits dentro de un byte. Va de 1 a 8.
T_Binary_File – Conjunto que representa el símbolo lógico de un archivo formado por bytes.
T_Mode – Conjunto formado por las diferentes posibilidades de apertura o creación de un fichero binario T_Binary_file. Está formado por In_File, Inout_File, Out_File, cuyo significado es el mismo que el Mode de Ada.direct_IO.

- Conjuntos de valores externos que participan en el TAD

Boolean; **Integer**; **Character**; **String**; **Natural**; **Positive**.

Nota: *Position* se puede convertir directamente a entero o un tipo similar con la operación "Integer()" o "Positive()"

- Constructores

Init_byte: -> Byte
Init_byte devuelve un byte formado por 8 bits de valor 0: '00000000'

Zero: -> Bit
Zero devuelve un bit de valor '0'

One: -> Bit
One devuelve un bit de valor '1'

- Operaciones

Create_byte: Bit x Bit x Bit x Bit x Bit x Bit x Bit x Bit-> Byte
Create_byte(B0,B1,B2,B3,B4,B5,B6,B7) devuelve un byte de valor 'B0B1B2B3B4B5B6B7'.
Si B0..B7=0, Create_byte(B0,B1,B2,B3,B4,B5,B6,B7)=Init_byte

Look_bit: Byte x Position -> Bit
Look_bit(B,P) devuelve el valor del bit cuya posición en B es P. Si P=8, devuelve el valor del bit menos significativo.

Change_bit: Byte x Bit x Position -> Byte
Change_bit(B,X,P) modifica el valor del bit de B cuya posición es P por el valor X.

Is_Zero: Bit -> Boolean
Is_zero(B) devuelve TRUE si B=0, sino devuelve FALSE.

Is_One: Bit -> Boolean
Is_One(B) devuelve False si B=1, sino devuelve FALSE.

- *Operaciones de conversión*

To_byte: Integer -> Byte
To_byte(I) devuelve el byte que corresponde al entero I. Por ejemplo si I=4, devuelve '00000100'. Si I vale más de 255 o menos de 0 (es decir, si no es un número binario natural sin signo de 8 bits), el resultado es indefinido.

To_Integer: Byte -> Integer
To_Integer(B) devuelve el entero representado por el byte B en BNSS (binario natural sin signo). Por ejemplo, B='00001111' la función devolvería '15'.

To_byte: Character -> Byte
To_byte(C) devuelve el byte que corresponde al caracter C según el ASCII.

To_Character: Byte -> Character
To_Character(B) devuelve el caracter ASCII que corresponde al byte B.

To_bit: Integer -> bit
To_bit(I) devuelve el bit '0' si I=0, cualquier otro valor para I devuelve el bit '1'.

To_Integer: bit -> Integer
To_Integer(B) devuelve el valor del bit en forma entera. Si el bit B es '1', devuelve el entero '1'. Si el bit B es '0', devuelve el entero '0'

- *Operaciones de lógica booleana*

Not: Bit -> Bit
Not B devuelve el otro valor posible de bit: si B=0 devuelve 1, sino devuelve 0. Not Zero=One y Not One=Zero.

Or: Bit x Bit -> Bit
A or B devuelve el resultado de aplicar un Or entre A y B según la lógica booleana.
Si A y/o B es 1, esta operación devuelve un uno. Solo devuelve cero cuando A y B son 0.

And: Bit x Bit -> Bit
A And B devuelve el resultado de aplicar un And entre A y B según la lógica booleana.
Esta operación devuelve un uno solamente cuando A y B valgan 1. Si A y/o B valen 0, devuelve un 0.

Xor: Bit x Bit -> Bit
A xor B devuelve el resultado de aplicar un Xor entre A y B según la lógica booleana.
Si A o B es 1, esta operación devuelve un uno. Solo devuelve cero cuando A y B valen lo mismo (es decir, cuando tanto A como B valen uno o cuando tanto A como B valen 0).

Not: Byte -> Byte
Not B devuelve un byte cuyos bits han sido invertidos.
Si B='10101100', Not B='01010011'.

Or: Byte x Byte -> Byte
A or B hace un OR bit a bit de los bits de A con los bits de B. Por ejemplo, si A='10000100' y B='00000101' esta operación devolvería '10000101'.

And: Byte x Byte -> Byte
A And B hace un AND bit a bit de los bits de A con los bits de B. Por ejemplo, si A='10000100' y B='00000101' esta operación devolvería '00000100'.

Xor: Byte x Byte -> Byte
A xor B hace un XOR bit a bit de los bits de A con los bits de B. Por ejemplo, si A='10000100' y B='00000101' esta operación devolvería '10000001'.

– *Operaciones de entrada y salida*

Las operaciones de manejo de ficheros y de entrada/salida son las equivalentes a las operaciones del paquete Direct_IO, adaptadas para ser usadas directamente con el tipo "Fichero Binario (T_Binary_File)"

Se enumeran a continuación tal como figuran en **d_byte.ads**:

```
----------------
-- Ficheros --
----------------

procedure Create (
      File : in out T_Binary_file;
      Mode : in     T_Mode  := Inout_File;
      Name : in     String  := ""          );

procedure Open (
      File : in out T_Binary_file;
      Mode : in     T_Mode;
      Name : in     String   );

procedure Close (
      File : in out T_Binary_file );

procedure Delete (
      File : in out T_Binary_file );

procedure Reset (
      File : in out T_Binary_file;
      Mode : in     T_Mode   );

procedure Reset (
      File : in out T_Binary_file );

function Mode (
      File : in     T_Binary_file )
  return T_Mode;

function Name (
      File : in     T_Binary_file )
  return String;

function Is_Open (
      File : in     T_Binary_file )
  return Boolean;
```

```
-----------------------
-- Entrada / Salida--
-----------------------

procedure Read (
      File : in     T_Binary_file;
      Item :    out Byte;
      From : in     Positive );

procedure Read (
      File : in     T_Binary_file;
      Item :    out Byte     );

procedure Write (
      File : in     T_Binary_file;
      Item : in     Byte;
      To   : in     Positive );

procedure Write (
      File : in     T_Binary_file;
      Item : in     Byte     );

procedure Set_Index (
      File : in     T_Binary_file;
      To   : in     Positive );

function Index (
      File : in     T_Binary_file )
  return Positive;

function Size (
      File : in     T_Binary_file )
  return Natural;

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