

Programmation - Ada

La documentation

Les Entrées-Sorties simples

[Exemple 4 : Programme simple documenté.](#)

[Exemple 5 : Les E/S simples - Les nombres entiers](#)

[Exemple 6 : Les E/S simples - Les nombres réels](#)

[Exemple 7 : Les E/S simples - Les booléens](#)

=====

Programmation - Ada

La documentation

Les Entrées-Sorties simples

Jusqu'à présent nous avons écrit quelques programmes simples en mettant en évidence la nécessité de mettre des messages avant une saisie au clavier mais il manque :

- La documentation au niveau du programme source,
- Des messages au niveau de l'exécution du programme.

Exemple 4 : Programme simple documenté.

Sujet : Programme affichant un message. (Le "Hello" ou le "Bonjour" ...)

Démarche et pseudo-code :

Ce programme a une **structure de bloc**. Dans ce cas le pseudo-code est inutile. La codification en Ada se fera directement.

Codification en Ada et commentaires :

```
-- *****  
-- Nom Prog      : Exemple_4.adb  
-- Type          : Application  
-- Sujet         : Modele de programme  
--  
-- Auteur        : R. VANDAELE  
-- Version       : 1.1  
-- Creation      : 27/10/2004  
-- Dern. Modif  : 27/10/2004  
--  
-- Compilateur  : GNAT 3.12p  
-- Remarques    : Environnement Windows 98 SE  
-- *****  
  
-- Paquetage(s) utilise(s)  
  
with Ada.Text_IO;  
use  Ada.Text_IO;  
  
procedure Exemple_4 is  
  -- partie declarative  
  -- Variables, Procedure, Fonctions, ...
```

```

begin
  -- Debut du corps de la procedure

  -- Message de debut d'execution
  Put_Line ("Debut - Execution Exemple_4");
  New_Line;
  Put_Line ("Modele de procedure GNAT 3.12p          ");
  Put_Line ("EnvironmentWindows 32 bits (95 a XP)");
  Put_Line ("-----");
  New_Line;

  -- Procedure proprement dite
  -- Suite d'instructions
  Put_Line ("Ce programme affiche un message");

  -- Message de fin d'execution
  New_Line;
  Put_Line ("Fin   - Execution Exemple_4");

end Exemple_4;

```

La première partie du programme source, l'en-tête donne :

- Le nom du programme source
- Le sujet traité
- Des renseignements divers (l'auteur, les dates de création et de modification, l'environnement de travail)

Ce programme permet d'identifier les différentes parties d'une procédure écrite en Ada.

- La partie déclarative,
- Le corps proprement dit.

Le résultat de l'exécution est :

```

Debut - Execution Exemple_4

Modele de procedure GNAT 3.12p
EnvironmentWindows 32 bits (95 a XP)
-----

Ce programme affiche un message

Fin   - Execution Exemple_4

```

Exemple 5 : Les E/S simples - Les nombres entiers

EXEMPLE 5a : Un oubli

Sujet : Saisir un nombre entier et visualiser son carré.

Démarche et pseudo-code :

Ce programme a une **structure de bloc**. Dans ce cas le pseudo-code est inutile. La codification en Ada se fera directement.

Codification en Ada et commentaires :

```
-- *****
-- Nom Prog      : Exemple_5a.adb
-- Type         : Application
-- Sujet        : Les E/S simples (1)
--
-- Auteur       : R. VANDAELE
-- Version      : 1.1
-- Creation     : 27/10/2004
-- Dern. Modif  : 27/10/2004
--
-- Compilateur  : GNAT 3.12p
-- Remarques    : Environnement Windows 98 SE
-- *****

-- Paquetage(s) utilise(s)

with Ada.Text_IO;
use  Ada.Text_IO;

procedure Exemple_5a is

    Nombre_Entier : Integer;

begin

    Put_Line ("Debut - Execution Exemple_5a");
    New_Line;
    Put_Line ("Les E/S simples (1)");
    Put_Line ("-----");
    New_Line;

    Put ("Saisir un nombre entier : ");
    Get (Nombre_Entier);
    Put ("Son carre est egal a : ");
    Put (Nombre_Entier * Nombre_Entier);
    New_Line;

    -- Message de fin d'execution
    New_Line;
    Put_Line ("Fin   - Execution Exemple_5a");
```

```
end Exemple_5a;
```

Résultat de la compilation dans le fichier Exemple_5a.lsb

Pour obtenir le résultat de la compilation dans un fichier, voici la séquence à utiliser :

- Run
- Compile to Listing

```
GNAT 3.12p (19990629) Copyright 1992-1999 Free Software
Foundation, Inc.

Compiling: d:\mes_pr~1\ada_ead\exemple_5a.adb (source file time
stamp: 2004-10-29 22:15:32)

  1. --
*****
  2. -- Nom Prog      : Exemple_5a.adb
  3. -- Type         : Application
  4. -- Sujet        : Les E/S simples (1)
  5. --
  6. -- Auteur       : R. VANDAELE
  7. -- Version      : 1.1
  8. -- Creation     : 27/10/2004
  9. -- Dern. Modif : 27/10/2004
 10. --
 11. -- Compilateur  : GNAT 3.12p
 12. -- Remarques   : Environnement Windows 98 SE
 13. --
*****
 14.
 15. -- Paquetage(s) utilise(s)
 16.
 17. with Ada.Text_Io;
 18. use  Ada.Text_Io;
 19.
 20. procedure Exemple_5a is
 21.
 22.     Nombre_Entier : Integer;
 23.
 24. begin
 25.
 26.     Put_Line ("Debut - Execution Exemple_5a");
 27.     New_Line;
 28.     Put_Line ("Les E/S simples (1)");
 29.     Put_Line ("-----");
 30.     New_Line;
 31.
 32.     Put ("Saisir un nombre entier : ");
 33.     Get (Nombre_Entier);
    |
    >>> invalid parameter list in call (use -gnatf for
details)
```

```

34.   Put ("Son carre est egal a : ");
35.   Put (Nombre_Entier * Nombre_Entier);
    |
    >>> invalid parameter list in call (use -gnatf for
details)
    >>> possible missing instantiation of Text_IO.Integer_IO

36.   New_Line;
37.
38.   -- Message de fin d'execution
39.   New_Line;
40.   Put_Line ("Fin   - Execution Exemple_5a");
41.
42. end Exemple_5a;

42 lines: 3 errors

```

Résultat de la compilation dans le fichier Exemple 5a.lsb

A la ligne 33 : le paramètre Get est invalide.

A la ligne 35 : le paramètre Put est invalide et l'instanciation de Text_IO.Integer_Io est omise.

Il existe trois méthodes pour corriger ces erreurs.

L'exécution du programme est impossible

EXEMPLE 5b

Première Méthode : Paquetage Ada.Integer Text Io

Utilisation du paquetage Ada.Integer Text Io fourni dans la version Ada95

Il existe trois méthodes pour corriger ces erreurs. Le compilateur GNAT contient des paquetages, en particulier le paquetage **Ada.Integer Text Io**, il permet de gérer les entrées-sorties des entiers.

La référence à ce paquetage est faite au niveau des "instructions" **with** et **use**

Codification en Ada et commentaires :

```

-- *****
-- Nom Prog      : Exemple_5b.adb
-- Type         : Application
-- Sujet        : Les E/S simples (2)
--
-- Auteur       : R. VANDAELE
-- Version      : 1.1
-- Creation     : 27/10/2004
-- Dern. Modif  : 27/10/2004
--

```

```

-- Compilateur : GNAT 3.12b
-- Remarques   : Environnement Windows 98 SE
-- *****

-- Paquetage(s) utilise(s)

with Ada.Text_Io, Ada.Integer_Text_Io;
use   Ada.Text_Io, Ada.Integer_Text_Io;

procedure Exemple_5b is

    Nombre_Entier : Integer;

begin

    Put_Line ("Debut - Execution Exemple_5b");
    New_Line;
    Put_Line ("Les E/S simples (2)");
    Put_Line ("-----");
    New_Line;

    Put ("Saisir un nombre entier : ");
    Get (Nombre_Entier);
    New_Line;

    Put ("Son carre est egal a : ");
    Put (Nombre_Entier * Nombre_Entier);
    New_Line;

    -- Message de fin d'execution
    New_Line;
    Put_Line ("Fin   - Execution Exemple_5b");

end Exemple_5b;

```

Cette fois la compilation se fait sans erreur.

Résultat de l'exécution

```

Debut - Execution Exemple_5b

Les E/S simples (2)
-----

Saisir un nombre entier : 12

Son carre est egal a :      144

Fin   - Execution Exemple_5b

```

EXEMPLE 5c

Deuxième Méthode : Instanciation interne de Integer Io

Instanciation de Integer Io dans la partie déclarative du programme

On donnera un nom à ce paquetage interne. En Ada 83, le paquetage Ada . Integer_Text_Io n'était pas implémenté. Le nom du paquetage interne (ou externe voir troisième méthode) est très souvent appelé Iio pour Integer I/O.

Codification en Ada et commentaires :

```
-- *****
-- Nom Prog      : Exemple_5c.adb
-- Type         : Application
-- Sujet        : Les E/S simples (3)
--
-- Auteur       : R. VANDAELE
-- Version      : 1.1
-- Creation     : 27/10/2004
-- Dern. Modif  : 27/10/2004
--
-- Compilateur  : GNAT 3.12p
-- Remarques    : Environnement Windows 98 SE
-- *****

-- Paquetage(s) utilise(s)

with Ada.Text_Io;
use  Ada.Text_Io;

procedure Exemple_5c is

  Nombre_Entier : Integer;

  package Iio is new Integer_Io(Integer);
  use Iio;

begin

  Put_Line ("Debut - Execution Exemple_5c");
  New_Line;
  Put_Line ("Les E/S simples (3)");
  Put_Line ("-----");
  New_Line;

  Put ("Saisir un nombre entier : ");
  Get (Nombre_Entier);
  New_Line;
```

```

Put ("Son carre est egal a : ");
Put (Nombre_Entier * Nombre_Entier);
New_Line;

-- Message de fin d'execution
New_Line;
Put_Line ("Fin - Execution Exemple_5c");

end Exemple_5c;

```

Résultat de l'exécution : Identique à l'Exemple 5b

EXEMPLE 5d

Troisième Méthode : Création du paquetage externe Iio

Pour cela nous allons construire notre premier paquetage externe.

Un paquetage est constitué de deux parties :

- La partie spécification du paquetage.
- La partie corps du paquetage.

Comme il s'agit d'une instantiation d'un paquetage-fils (Integer_Io) du paquetage Ada.Text_Io, seule la partie spécification doit implémentée.

Partie spécification Programme source Iio.ads

Cette extension est particulière, c'est une spécification. C'est la raison pour laquelle elle se nomme .ads.

Codification en Ada et commentaires :

```

-- *****
-- creation du paquetage externe IIO
-- Robert VANDAELE le 30/10/2004
-- *****

with Ada.Text_Io;
use Ada.Text_Io;
package Iio is new Integer_Io (Integer);

```

On peut aussi écrire le paquetage de la façon suivante :

```

with Ada.Text_Io;

package Iio is new Ada.Text_Io.Integer_Io (Integer);

```

On ne met pas la clause use et on met comme préfixe le paquetage Ada.Text_IO.

Ces deux écriture sont équivalentes.

Codification en Ada et commentaires : (Exemple 5d)

```
-- *****
-- Nom Prog      : Exemple_5d.adb
-- Type          : Application
-- Sujet         : Les E/S simples (4)
--
-- Auteur        : R. VANDAELE
-- Version       : 1.1
-- Creation      : 27/10/2004
-- Dern. Modif  : 27/10/2004
--
-- Compilateur  : GNAT 3.12p
-- Remarques    : Environnement Windows 98 SE
-- *****

-- Paquetage(s) utilise(s)

with Ada.Text_IO, Iio;
use  Ada.Text_IO, Iio;

procedure Exemple_5d is

    Nombre_Entier : Integer;

begin

    Put_Line ("Debut - Execution Exemple_5d");
    New_Line;
    Put_Line ("Les E/S simples (4)");
    Put_Line ("-----");
    New_Line;

    Put ("Saisir un nombre entier : ");
    Get (Nombre_Entier);
    New_Line;

    Put ("Son carre est egal a : ");
    Put (Nombre_Entier * Nombre_Entier);
    New_Line;

    -- Message de fin d'execution
    New_Line;
    Put_Line ("Fin   - Execution Exemple_5d");

end Exemple_5d;
```

Cette fois le paquetage IIO est ajouté dans les clauses **with** et **use**.

Résultat de l'exécution : Identique à l'Exemple 5b et 5c

Exemple 6 : Les E/S simples - Les nombres réels

En cas d'oubli du paquetage concernant les entrées sorties des nombres réels, les messages sont similaires aux messages en cas d'oubli pour les nombres entiers.

Sujet pour l'ensemble des exemples 6 : Saisir un nombre réel et visualiser sa racine carrée (format sans formatage et avec formatage).

Démarche et pseudo-code :

Ce programme a une **structure de bloc**. Dans ce cas le pseudo-code est inutile. La codification en Ada se fera directement.

EXEMPLE 6a

Première Méthode : Paquetage Ada.Float Text Io

Utilisation du paquetage Ada.Float Text Io fourni dans la version Ada95

Le compilateur GNAT contient des paquetages, en particulier le paquetage **Ada.Float Text Io**, il permet de gérer les entrées-sorties des nombres réels.

La référence à ce paquetage est faite au niveau des "clauses" **with** et **use**

Codification en Ada et commentaires :

```
-- *****
-- Nom Prog      : Exemple_6a.adb
-- Type         : Application
-- Sujet        : Les E/S simples (5)
--
-- Auteur       : R. VANDAELE
-- Version      : 1.1
-- Creation     : 31/10/2004
-- Dern. Modif  : 31/10/2004
--
-- Compilateur  : GNAT 3.12p
-- Remarques    : Environnement Windows 98 SE
-- *****

-- Paquetage(s) utilise(s)

with Ada.Text_Io, Ada.Float_Text_Io;
use  Ada.Text_Io, Ada.Float_Text_Io;

-- paquetages a ajouter pour les fonctions SQRT, ...
with Ada.Numerics.Elementary_Functions;
```

```

use Ada.Numerics.Elementary_Functions;

procedure Exemple_6a is

    Nombre_Reel : Float;

begin

    Put_Line ("Debut - Execution Exemple_6a");
    New_Line;
    Put_Line ("Les E/S simples (5)");
    Put_Line ("-----");
    New_Line;

    Put ("Saisir un nombre reel          : ");
    Get (Nombre_reel);
    New_Line;

    -- Racine carree sans format de sortie
    Put ("Sa racine carree est egale a : ");
    Put (sqrt(Nombre_reel));
    New_Line (2);

    -- racine carree avec format de sortie
    Put ("Sa racine carree est egale a : ");
    Put (sqrt(Nombre_reel), fore => 4, aft => 5, exp => 0);
    New_Line;

    -- Message de fin d'execution
    New_Line;
    Put_Line ("Fin   - Execution Exemple_6a");

end Exemple_6a;

```

La compilation se fait sans erreur.

Résultat de l'exécution

```

Debut - Execution Exemple_6a

Les E/S simples (5)
-----

Saisir un nombre reel          : 2.0

Sa racine carree est egale a   : 1.41421E+00

Sa racine carree est egale a   : 1.41421

Fin   - Execution Exemple_6a

```

EXEMPLE 6b

Deuxième Méthode : Instanciation interne de Float Io

Instanciation de Float Io dans la partie déclarative du programme

On donnera un nom à ce paquetage interne. En Ada 83, le paquetage `Ada.Float_Text_Io` n'était pas implémenté. Le nom du paquetage interne (ou externe voir troisième méthode) est très souvent appelé **Fio** pour Float I/O.

Codification en Ada et commentaires :

```
-- *****
-- Nom Prog      : Exemple_6b.adb
-- Type          : Application
-- Sujet         : Les E/S simples (6)
--
-- Auteur        : R. VANDAELE
-- Version       : 1.1
-- Creation      : 31/10/2004
-- Dern. Modif  : 31/10/2004
--
-- Compilateur  : GNAT 3.12p
-- Remarques    : Environnement Windows 98 SE
-- *****

-- Paquetage(s) utilise(s)

with Ada.Text_Io;
use  Ada.Text_Io;

-- paquetages a ajouter pour les fonctions SQRT, ...
with Ada.Numerics.Elementary_Functions;
use  Ada.Numerics.Elementary_Functions;

procedure Exemple_6b is

    Nombre_Reel : Float;

    -- instanciation de FIO
    package Fio is new Float_Io(Float);
    use Fio;

begin

    Put_Line ("Debut - Execution Exemple_6b");
    New_Line;
    Put_Line ("Les E/S simples (6)");
    Put_Line ("-----");
    New_Line;
```

```

Put ("Saisir un nombre reel      : ");
Get (Nombre_Reel);
New_Line;

-- Racine carree sans format de sortie
Put ("Sa racine carree est egale a : ");
Put (Sqrt(Nombre_Reel));
New_Line (2);

-- racine carree avec format de sortie
Put ("Sa racine carree est egale a : ");
Put (Sqrt(Nombre_Reel), Fore => 4, Aft => 5, Exp => 0);
New_Line;

-- Message de fin d'execution
New_Line;
Put_Line ("Fin - Execution Exemple_6b");

end Exemple_6b;

```

Résultat de l'exécution : Identique à l'Exemple 6a

EXEMPLE 6c

Troisième Méthode : Création du paquetage externe Fio

L'écriture est analogue au paquetage Iio

Partie spécification Programme source Fio.ads

Codification en Ada et commentaires :

```

-- *****
-- creation du paquetage externe FIO
-- Robert VANDAELE le 30/10/2004
-- *****

with Ada.Text_IO;
use Ada.Text_IO;
package Fio is new Float_IO (Float);

```

On peut aussi écrire le paquetage de la façon suivante :

```

with Ada.Text_IO;

package Fio is new Ada.Text_IO.Float_IO (Float);

```

On ne met pas la clause use et on met comme préfixe le paquetage Ada.Text_IO.

Ces deux écriture sont équivalentes.

Codification en Ada et commentaires : (Exemple 6c)

```
-- *****
-- Nom Prog      : Exemple_6c.adb
-- Type          : Application
-- Sujet         : Les E/S simples (7)
--
-- Auteur        : R. VANDAELE
-- Version       : 1.1
-- Creation      : 31/10/2004
-- Dern. Modif  : 31/10/2004
--
-- Compilateur  : GNAT 3.12p
-- Remarques    : Environnement Windows 98 SE
-- *****

-- Paquetage(s) utilise(s)

with Ada.Text_IO, Fio;
use  Ada.Text_IO, Fio;

-- paquetages a ajouter pour les fonctions SQRT, ...
with Ada.Numerics.Elementary_Functions;
use  Ada.Numerics.Elementary_Functions;

procedure Exemple_6c is

    Nombre_Reel : Float;

begin

    Put_Line ("Debut - Execution Exemple_6c");
    New_Line;
    Put_Line ("Les E/S simples (7)");
    Put_Line ("-----");
    New_Line;

    Put ("Saisir un nombre reel      : ");
    Get (Nombre_Reel);
    New_Line;

    -- Racine carree sans format de sortie
    Put ("Sa racine carree est egale a : ");
    Put (Sqrt(Nombre_Reel));
    New_Line (2);

    -- racine carree avec format de sortie
```

```

Put ("Sa racine carree est egale a  : ");
Put (Sqrt(Nombre_Reel), Fore => 4, Aft => 5, Exp => 0);
New_Line;

-- Message de fin d'execution
New_Line;
Put_Line ("Fin   - Execution Exemple_6c");

end Exemple_6c;

```

Cette fois le paquetage FIO est ajouté dans les clauses **with** et **use**.

Résultat de l'exécution : Identique à l'Exemple 6a et 6b

Exemple 7 : Les E/S simples - Les booléens

Rien n'est prévu au niveau du compilateur GNAT pour les entrées-sorties des booléens.

Il faut donc instancier un paquetage interne au programme, dans sa partie déclarative.

Le nom du paquetage sera cette fois **Bio**, pour Boolean I/O.

Sujet pour l'ensemble des exemples 7 : Saisir deux nombres entiers, afficher TRUE si ces deux nombres sont égaux, afficher FALSE si c'est deux nombres sont différents.

Démarche et pseudo-code :

Ce programme a une **structure de bloc** contenant une alternative simple. Le pseudo-code est dans ce cas inutile.

La codification en Ada se fera directement.

EXEMPLE 7a

Première Méthode : Instanciation interne de Enumeration IO

Instanciation de Enumeration Io dans la partie déclarative du programme

Le type booléen est un type énuméré. On utilisera alors **Enumeration Io** pour le type booléen (boolean). Le paquetage interne sera nommé **Bio** pour Boolean I/O

Codification en Ada et commentaires :

```

-- *****
-- Nom Prog      : Exemple_7a.adb
-- Type         : Application
-- Sujet        : Les E/S simples (6)
--

```

```

-- Auteur      : R. VANDAELE
-- Version     : 1.1
-- Creation    : 31/10/2004
-- Dern. Modif : 31/10/2004
--
-- Compilateur : GNAT 3.12p
-- Remarques   : Environnement Windows 98 SE
-- *****

-- Paquetage(s) utilise(s)

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

procedure Exemple_7a is

    Nombre_1 : Integer;
    Nombre_2 : Integer;

    Egalite : Boolean;

    package Bio is new Enumeration_IO (Boolean);
    use Bio;

begin

    Put_Line ("Debut - Execution Exemple_7a");
    New_Line;
    Put_Line ("Les E/S simples (6)");
    Put_Line ("-----");
    New_Line;

    Put ("Saisir le premier nombre entier : ");
    Get (Nombre_1);
    New_Line;

    Put ("Saisir le second nombre entier : ");
    Get (Nombre_2);
    New_Line;

    Put("Resultat du test d'egalite : ");

    if Nombre_1 = Nombre_2 then
        Egalite := True;
    else
        Egalite := False;
    end if;

    Put(Egalite);
    New_Line;

    -- Message de fin d'execution

```

```
New Line;  
Put_Line ("Fin - Execution Exemple_7a");  
  
end Exemple_7a;
```

Résultat de l'exécution 1

```
Debut - Execution Exemple_7a  
  
Les E/S simples (6)  
-----  
  
Saisir le premier nombre entier : 29  
  
Saisir le second nombre entier : 29  
  
Resultat du test d'egalite : TRUE  
  
Fin - Execution Exemple_7a
```

Résultat de l'exécution 2

```
Debut - Execution Exemple_7a  
  
Les E/S simples (6)  
-----  
  
Saisir le premier nombre entier : 29  
  
Saisir le second nombre entier : 30  
  
Resultat du test d'egalite : FALSE  
  
Fin - Execution Exemple_7a
```

EXEMPLE 7b

Deuxième Méthode : Création du paquetage externe Bio

L'écriture est analogue aux paquetages Iio et Fio

Partie spécification Programme source Bio.ads

Codification en Ada et commentaires :

```
-- *****  
-- creation du paquetage externe BIO  
-- Robert VANDAELE le 31/10/2004
```

```

-- *****
with Ada.Text_IO;
use Ada.Text_IO;
package Bio is new Enumeration_IO (Boolean);

```

On peut aussi écrire le paquetage de la façon suivante :

```

with Ada.Text_IO;

package Bio is new Ada.Text_IO.Enumeration_IO (Boolean);

```

On ne met pas la clause use et on met comme préfixe le paquetage Ada.Text_IO.

Ces deux écriture sont équivalentes.

Codification en Ada et commentaires : (Exemple 7b)

```

-- *****
-- Nom Prog      : Exemple_7b.adb
-- Type          : Application
-- Sujet         : Les E/S simples (6)
--
-- Auteur        : R. VANDAELE
-- Version       : 1.1
-- Creation      : 31/10/2004
-- Dern. Modif  : 31/10/2004
--
-- Compilateur  : GNAT 3.12p
-- Remarques    : Environnement Windows 98 SE
-- *****

-- Paquetage(s) utilise(s)

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

procedure Exemple_7b is

    Nombre_1 : Integer;
    Nombre_2 : Integer;

    Egalite : Boolean;

begin

    Put_Line ("Debut - Execution Exemple_7b");
    New_Line;
    Put_Line ("Les E/S simples (6)");
    Put_Line ("-----");
    New_Line;

```

```

Put ("Saisir le premier nombre entier : ");
Get (Nombre_1);
New_Line;

Put ("Saisir le second nombre entier : ");
Get (Nombre_2);
New_Line;

Put("Resultat du test d'egalite : ");

if Nombre_1 = Nombre_2 then
    Egalite := True;
else
    Egalite := False;
end if;

Put(Egalite);
New_Line;

-- Message de fin d'execution
New_Line;
Put_Line ("Fin - Execution Exemple_7b");

end Exemple_7b;

```

Cette fois le paquetage BIO est ajouté dans les clauses with et use.

Résultat de l'exécution : Identique à l'Exemple 7a

=====