

# Programmation - Ada

## Les Instructions de base - Quelques exemples

Reprenons les mots clés de la programmation structurée et associons les instructions Ada correspondantes.

Ceci devrait permettre de codifier ces exemples en Ada.

### Les mots clés du pseudo-code et le langage Ada

Les mots clés décalés vers la droite sont des synonymes

#### Les entrées - sorties

| <u>Pseudo Code</u>               | <b>Ada</b>      |
|----------------------------------|-----------------|
| LIRE                             | <b>GET</b>      |
| ECRIRE<br>AFFICHER<br>VISUALISER | <b>PUT</b>      |
| RETOUR_LIGNE                     | <b>NEW_LINE</b> |

#### L'affectation

| <u>Pseudo Code</u> | <b>Ada</b>    |
|--------------------|---------------|
| A ← B              | <b>A := B</b> |

#### Les opérateurs logiques

| <u>Pseudo Code</u> | <b>Ada</b>     |
|--------------------|----------------|
| ET                 | <b>AND</b>     |
| OU                 | <b>OR</b>      |
| NON                | <b>NOT</b>     |
| NON-ET             | <b>NOT AND</b> |

#### Les commentaires

| <u>Pseudo Code</u> | <b>Ada</b> |
|--------------------|------------|
|--------------------|------------|

|                             |   |
|-----------------------------|---|
| - Ceci est un commentaire - | -- En Ada un commentaire commence deux tirets |
|-----------------------------|---|

### Les tests

| <u>Pseudo Code</u> | Ada            |
|--------------------|----------------|
| < <= > >=          | mêmes symboles |
| <>                 | /= différent   |

### Les structures de contrôle

| <u>Pseudo Code</u>             | Ada                                      |
|--------------------------------|--|
| DEBUT FIN                      | <b>BEGIN END</b>                         |
| SI ALORS FIN SI                | <b>IF THEN END IF</b>                    |
| SI ALORS SINON FIN SI          | <b>IF THEN ELSE END IF</b>               |
| SI ALORS SINON_SI SINON FIN SI | <b>IF THEN ELSIF ELSE END IF</b>         |
| TANT QUE ...FIN TANT QUE       | <b>WHILE LOOP END LOOP</b>               |
| POUR VAR DE N1 A N2 FIN POUR   | <b>FOR VAR IN N1 .. N2 LOOP END LOOP</b> |
| CAS PARMY FIN CAS              | <b>CASE END CASE</b>                     |

### Les déclarations de variables

Exemples :

| <u>Pseudo Code</u> | Ada                |
|--------------------|--------------------|
| N DE TYPE ENTIER   | <b>N : INTEGER</b> |
| R DE TYPE REEL     | <b>R : FLOAT</b>   |
| B DE TYPE BOOLEEN  | <b>B : BOOLEAN</b> |

**Remarques** : Les autres déclarations et instructions seront données au fur et à mesure de cette présentation en fonction des besoins.

### Exemple 1 : Pluspetit, plus grand ou égal.

**Sujet** : Saisir deux nombres entiers (A et B) et afficher selon le cas :

A est égal à B

A est plus petit que B

A est plus grand que B

### **Démarche et pseudo-code :**

Il faut saisir les deux nombres entiers A et B et faire trois tests (  $A = B$  ,  $A < B$  et  $A > B$  ). Les saisies et les tests ne s'exécutent qu'une seule fois. Le programme a **une structure d'alternative.**

#### **Structure générale du programme.**

**Le programme ne s'exécute qu'une seule fois. C'est une structure d'alternative. (Si Alors Sinon Fin Si)**

### **Pseudo-code simplifié : Solution 1**

A DE TYPE ENTIER

B DE TYPE ENTIER

On peut écrire les deux lignes précédentes sous la forme

A, B DE TYPE ENTIER

DEBUT

LIRE A

LIRE B

SI A = B ALORS

    ECRIRE "A est égal à B"

FIN SI

SI A < B ALORS

    ECRIRE "A est plus petit que B"

FIN SI

SI A > B ALORS

    ECRIRE "A est plus grand que B"

FIN SI

FIN

**Remarque** : La séquence des trois tests répond au problème posé mais cette solution n'est pas très élégante !

La codification suivante (Exemple\_1b) utilise le "Si imbriqué".

### **EXEMPLE 1A**

#### **Codification en Ada et commentaires :**

```
with Ada.Text_IO; use Ada.Text_IO;
with Ada.Integer_Text_IO; use Ada.Integer_Text_IO;
procedure Exemple_1a is
  A : Integer;
  B : Integer;
begin
```

```

Get (A);
Get(B);
if A = B then
    Put("A est egal à B");
end if;
if A < B then
    Put("A est plus petit que B");
end if;
if A > B then
    Put("A est plus grand que B");
end if;
end Exemple_1a;

```

Les deux premières lignes contiennent des références à deux paquetages :

Ada.Text\_Io

Ada.Integer\_Text\_Io

Le premier Ada.Text\_Io permet de gérer les entrées sorties des caractères et des chaînes de caractères.

Le second Ada.Integer\_Text\_Io permet de gérer les entrées sorties des entiers.

Le reste du programme doit se lire assez facilement.

### **Le résultat de l'exécution est :**

```

5
2
A est plus grand que B

```

L'exécution correspond bien à l'algorithme mais aucun message n'apparaît à l'écran. (voir le programme Exemple\_1c).

### **Pseudo-code simplifié : Solution 2**

A, B DE TYPE ENTIER

DEBUT

LIRE A

LIRE B

SI A = B ALORS

    ECRIRE "A est égal à B"

SINON

    SI A < B ALORS

        ECRIRE "A est plus petit que B"

    SINON

        ECRIRE "A est plus grand que B"

    FIN SI

FIN SI

FIN

## **EXEMPLE 1B**

### **Codification en Ada et commentaires :**

```
with Ada.Text_IO; use Ada.Text_IO;
with Ada.Integer_Text_IO; use Ada.Integer_Text_IO;
procedure Exemple_1b is
  A : Integer;
  B : Integer;
begin
  Get (A);
  Get(B);
  if A = B then
    Put("A est egal à B");
  else
    if A < B then
      Put("A est plus petit que B");
    else
      Put("A est plus grand que B");
    end if;
  end if;
end Exemple_1b;
```

Ce programme contient une structure de if imbriqué.

### **Le résultat de l'exécution est :**

```
5
2
A est plus grand que B
```

## **EXEMPLE 1C**

### **Pseudo-code**

A, B DE TYPE ENTIER

DEBUT

AFFICHER "Début Exemple\_1c"

AFFICHER "Rentrer un nombre entier A : "

LIRE A

AFFICHER "Rentrer un nombre entier B : "

LIRE B

SI A = B ALORS

    ECRIRE "A est égal à B"

SINON

```

SI A < B ALORS
    ECRIRE "A est plus petit que B"
SINON
    ECRIRE "A est plus grand que B"
FIN SI
FIN SI
AFFICHER "Fin Exemple_1c"
FIN

```

### **Codification en Ada et commentaires :**

```

with Ada.Text_IO; use Ada.Text_IO;
with Ada.Integer_Text_IO; use Ada.Integer_Text_IO;
procedure Exemple_1c is
  A : Integer;
  B : Integer;
begin
  Put_Line("Debut Exemple_1c");
  Put("Rentrer un nombre entier A : ");
  Get (A);
  Put("Rentrer un nombre entier B : ");
  Get(B);
  if A = B then
    Put("A est egal a B");
  else
    if A < B then
      Put("A est plus petit que B");
    else
      Put("A est plus grand que B");
    end if;
  end if;
  New_Line;
  Put_Line("Fin Exemple_1c");
end Exemple_1c;

```

**Commentaire** : Ce programme met en évidence qu'il est préférable de mettre des messages avant une saisie.

Le **Put\_line** permet d'afficher un caractère ou une chaîne de caractères et de placer le curseur au début de la ligne suivante (retour chariot et interligne).

**Conditions de tests** : Afin de vérifier complètement l'algorithme il faudra exécuter le programme trois fois avec le jeu d'essai suivant :

| Exécution | Variable A | Variable B |
|-----------|------------|------------|
| 1         | 2          | 2          |
| 2         | 2          | 5          |

|   |   |   |
|---|---|---|
| 3 | 5 | 2 |
|---|---|---|

### **Résultat de l'exécution 1 :**

```
Debut Exemple_1c
Rentrer un nombre entier A : 2
Rentrer un nombre entier B : 2
A est egal a B
Fin Exemple_1c
```

### **Résultat de l'exécution 2 :**

```
Debut Exemple_1c
Rentrer un nombre entier A : 2
Rentrer un nombre entier B : 5
A est plus petit que B
Fin Exemple_1c
```

### **Résultat de l'exécution 3 :**

```
Debut Exemple_1c
Rentrer un nombre entier A : 5
Rentrer un nombre entier B : 2
A est plus grand que B
Fin Exemple_1c
```

Ces trois tests permettent de tester toutes les "branches" de votre programme.

Vous aurez souvent à le faire afin de vérifier que votre programme teste toutes les combinaisons de saisie.

Les messages de début et de fin indiquent clairement que le programme a démarré mais surtout qu'il se termine correctement.

Ces notions (passage dans toutes les branches et marquage du début et de la fin) introduisent la **notion de preuve de programme.**

## **EXEMPLE 1D**

### **Pseudo-code**

A, B DE TYPE ENTIER

DEBUT

AFFICHER "Début Exemple\_1d

AFFICHER "Rentrer un nombre entier A : "

LIRE A

AFFICHER "Rentrer un nombre entier B : "

LIRE B

```

SI A = B ALORS
    ECRIRE "A est égal à B"
SINON SI A < B ALORS
    ECRIRE "A est plus petit que B"
SINON
    ECRIRE "A est plus grand que B"
FIN SI
AFFICHER "Fin Exemple_1d
FIN

```

### **Codification en Ada et commentaires :**

```

with Ada.Text_IO; use Ada.Text_IO;
with Ada.Integer_Text_IO; use Ada.Integer_Text_IO;
procedure Exemple_1dis
  A : Integer;
  B : Integer;
begin
  Put_Line("Debut Exemple_1d");
  Put("Rentrer un nombre entier A : ");
  Get (A);
  Put("Rentrer un nombre entier B : ");
  Get(B);
  if A = B then
    Put("A est egal a B");
  elsif A < B then
    Put("A est plus petit que B");
  else
    Put("A est plus grand que B");
  end if;
  New_Line;
  Put_Line("Fin Exemple_1d");
end Exemple_1d;

```

**Commentaire :** Ce programme utilise le Si Alors Sinon Si Sinon Fin Si. Il peut y avoir, si nécessaire, plusieurs Sinon Si à la suite.

### **Résultat de l'exécution :**

Les résultats des exécutions sont identiques à celles de l'Exemple\_1c.

### **Exemple 2 : Une ligne d'étoiles**

**Sujet :** Saisir un nombre entier (Largeur) et afficher une ligne de "Largeur fois étoiles".

Si le nombre entier Largeur est égal à 15, la ligne sera de 15 étoiles.

Dans un premier temps le nombre entier Largeur sera compris entre 1 et 20 sans contrôle.

## Démarche et pseudo-code :

Il faut saisir un nombre entier Largeur. Ensuite il faut tracer une ligne contenant un nombre d'étoiles égal à Largeur. Cette fois le programme est constitué d'une structure de boucle.

### EXEMPLE 2A

## Utilisation d'une boucle Tant Que

### Pseudo-code simplifié : Solution 1 (avec messages)

LARGEUR DE TYPE ENTIER

N DE TYPE ENTIER

DEBUT

AFFICHER "Début Exemple\_2a"

AFFICHER "Rentrer un nombre entier LARGEUR ( entre 1 et 20) : "

LIRE LARGEUR

$N \leftarrow 1$

TANT QUE  $N \leq$  LARGEUR

    VISUALISER '\*'

$N \leftarrow N + 1$

FIN TANT QUE

RETOUR\_LIGNE

AFFICHER "Fin Exemple\_2a"

FIN

## Codification en Ada et commentaires :

```
with Ada.Text_IO; use Ada.Text_IO;
with Ada.Integer_Text_IO; use Ada.Integer_Text_IO;
procedure Exemple_2a is
  Largeur : Integer;
  N : Integer;
begin
  Put_Line("Debut Exemple_2a");
  Put("Rentrer un nombre entier LARGEUR (entre 1 et 20) : ");
  Get (Largeur);
  N := 1;
  while N <= Largeur loop
    Put('*');
    N := N + 1;
  end loop;
  New_Line;
  Put_Line("Fin Exemple_2a");
end Exemple_2a;
```

## Résultat de l'exécution

```
Debut Exemple_2a
Rentrer un nombre entier LARGEUR (entre 1 et 20) : 15
*****
Fin Exemple_2a
```

## **EXEMPLE 2B**

### **Utilisation d'une boucle Pour (avec messages)**

#### **Pseudo-code simplifié : Solution 2**

LARGEUR DE TYPE ENTIER

DEBUT

AFFICHER "Début Exemple\_2b"

AFFICHER "Rentrer un nombre entier LARGEUR (entre 1 et 20) : "

LIRE LARGEUR

POUR I de 1 à LARGEUR

    VISUALISER '\*'

FIN POUR

RETOUR\_LIGNE

AFFICHER "Fin Exemple\_2b"

FIN

#### **Codification en Ada et commentaires :**

```
with Ada.Text_IO; use Ada.Text_IO;
with Ada.Integer_Text_IO; use Ada.Integer_Text_IO;
procedure Exemple_2b is
  Largeur : Integer;
begin
  Put_Line("Debut Exemple_2b");
  Put("Rentrer un nombre entier LARGEUR (entre 1 et 20) : ");
  Get (Largeur);
  for I in 1 .. Largeur loop
    Put('*');
  end loop;
  New_Line;
  Put_Line("Fin Exemple_2b");
end Exemple_2b;
```

#### **Résultat de l'exécution**

```
Debut Exemple_2b
Rentrer un nombre entier LARGEUR (entre 1 et 20) : 15
*****
Fin Exemple_2b
```

### **Exemple 3 : Rectangle plein.**

**Sujet :** Saisir deux nombres entiers (Largeur et Hauteur) et afficher un rectangle plein, le caractère de remplissage sera l'étoile. Dans un premier temps les nombres entiers Largeur et Hauteur seront pris entre 1 et 20 pour Largeur et entre 1 et 10 pour Hauteur sans contrôle dans le programme.

#### **Démarche et pseudo-code :**

Il faut saisir les deux nombres entiers Largeur et Hauteur. Ensuite il faut tracer des ligne contenant un nombre d'étoiles égal à Largeur. Le nombre de lignes est égal à Hauteur.

Ce programme est constitué des **deux boucles imbriqués**

#### **Pseudo-code (avec messages) :**

```
LARGEUR, HAUTEUR DE TYPE ENTIER
DEBUT
    AFFICHER "Rentrer un nombre entier LARGEUR (de 1 à 20)
    LIRE LARGEUR
    AFFICHER "Rentrer un nombre entier HAUTEUR (de 1 à 10)
    LIRE HAUTEUR
    POUR I de 1 à HAUTEUR
        POUR J DE 1 à LARGEUR
            VISUALISER '*'
        FIN POUR
    RETOUR_LIGNE
FIN POUR
RETOUR_LIGNE
FIN
```

#### **Codification en Ada et commentaires :**

```
with Ada.Text_IO; use Ada.Text_IO;
with Ada.Integer_Text_IO; use Ada.Integer_Text_IO;
procedure Exemple_3 is
    Largeur : Integer;
    Hauteur : Integer;
begin
    Put_Line("Debut Exemple_3");
    Put("Rentrer un nombre entier LARGEUR (entre 1 et 20) : ");
    Get (Largeur);
    Put("Rentrer un nombre entier HAUTEUR (entre 1 et 10) : ");
    Get (Hauteur);
    for I in 1 .. Hauteur loop
        for J in 1 .. Largeur loop
            Put('*');
        end loop;
    end loop;
```

