

Programmation - Ada

Les bases de la programmation structurée

La programmation structurée a été introduite vers la fin des années 1960 et le début des années 1970 par DIJKSTRA (1965 - 1968) et WIRTH (1971). DIJKSTRA a démontré mathématiquement que tous les programmes séquentiels peuvent s'écrire avec les trois structures suivantes :

La structure de **bloc**

La structure **alternative**

La structure de **boucle**

En quelques mots, c'est la fin de l'instruction GO TO utilisée dans les langages COBOL, BASIC,

Le pseudo-code, en abrégé **P-code**, autorise aussi la **déclaration des variables**.

Il n'est pas question ici de faire un cours sur la programmation structurée, il existe des bibliographies très bien faites et il y a des cours ou des informations que vous trouverez sur Internet.

Vous pouvez trouver tout cela en lançant une recherche : programmation structurée sur un moteur comme Google ou un métamoteur comme Copernic ou tout autre outil de votre choix.

Les principes fondamentaux

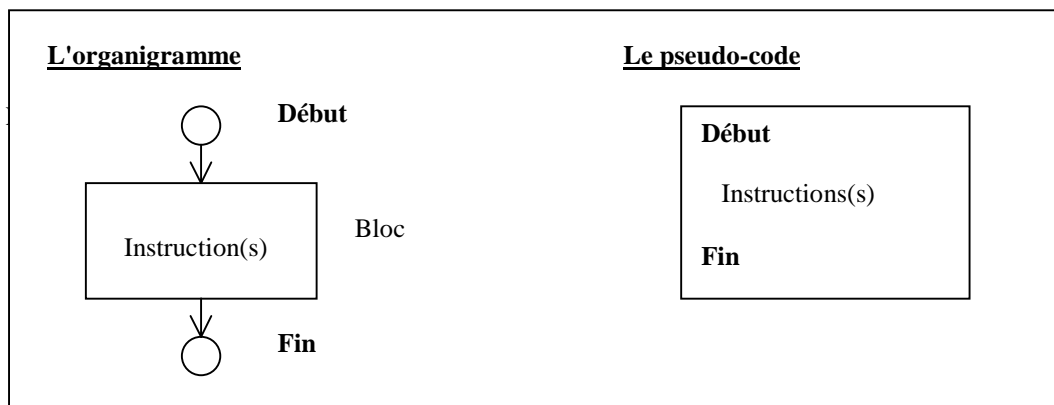
Une programme écrit en utilisant les principes de la programmation structurée possède **une entrée** et **une sortie**.

Comme il n'y a pas d'instruction GO TO (GOTO), il n'y a pas de code mort en programmation structurée.

Cette présentation est basée sur les **D-schémas**.

Les structures de base

La structure de bloc : Début Fin



Remarques :le bloc contient une instruction ou une suite d'instructions.

En Ada le bloc doit contenir au moins une instruction. Si cette instruction ne fait rien !! On utilise alors l'instruction "null".

Au niveau de l'écriture, l'**indentation** (décalage vers la droite) est utilisée afin de faciliter la lecture et la compréhension du programme source.

La structure alternative :

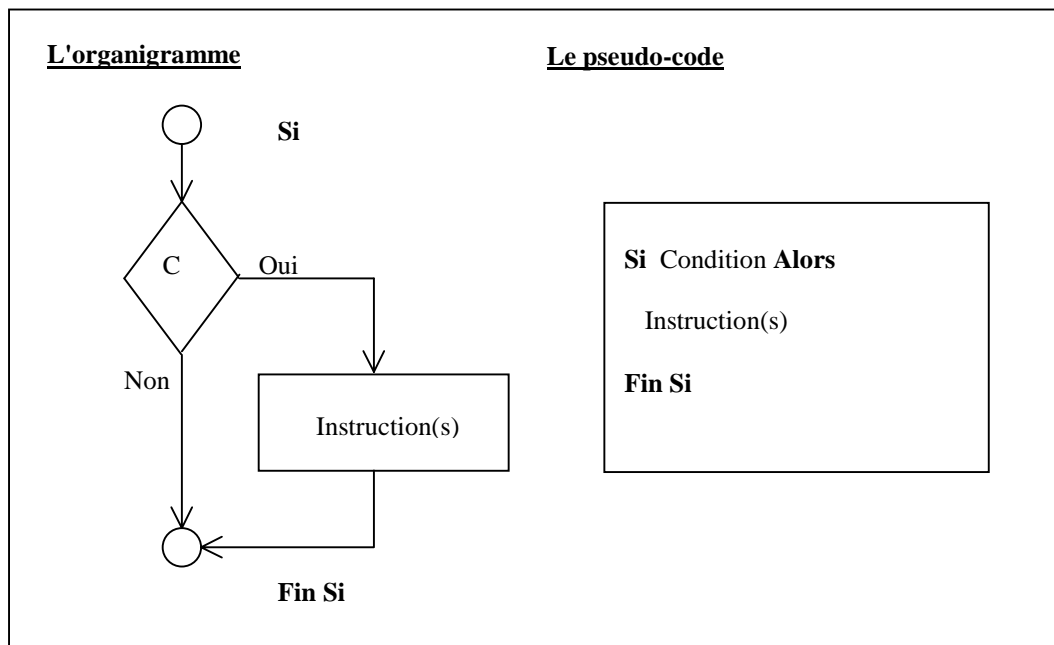
On peut distinguer plusieurs alternatives :

- l'alternative simplifiée (le **Si Alors Fin Si**)
- l'alternative (le **Si Alors Sinon Fin Si**)
- l'alternative imbriquée ne sera pas abordé dans cette présentation.

Le Si imbriqué (alternative imbriquée) sera abordé dans les exemples et les exercices mais la structure complémentaire **Cas** lui est préférable.

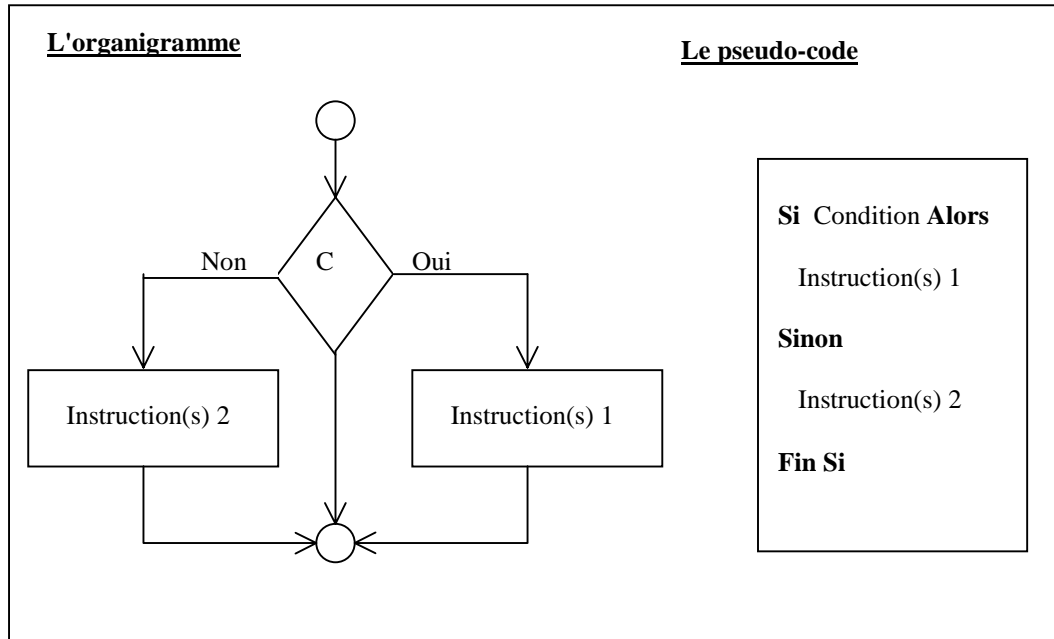
Important : L'alternative est une structure qui s'exécute une seule fois.

L'alternative simplifiée : Si Alors Fin Si



Remarque : S'il y a une réponse Non à la condition, ce qui signifie que la condition est "fausse", on va directement au Fin Si

L'alternative : Si Alors Sinon Fin Si

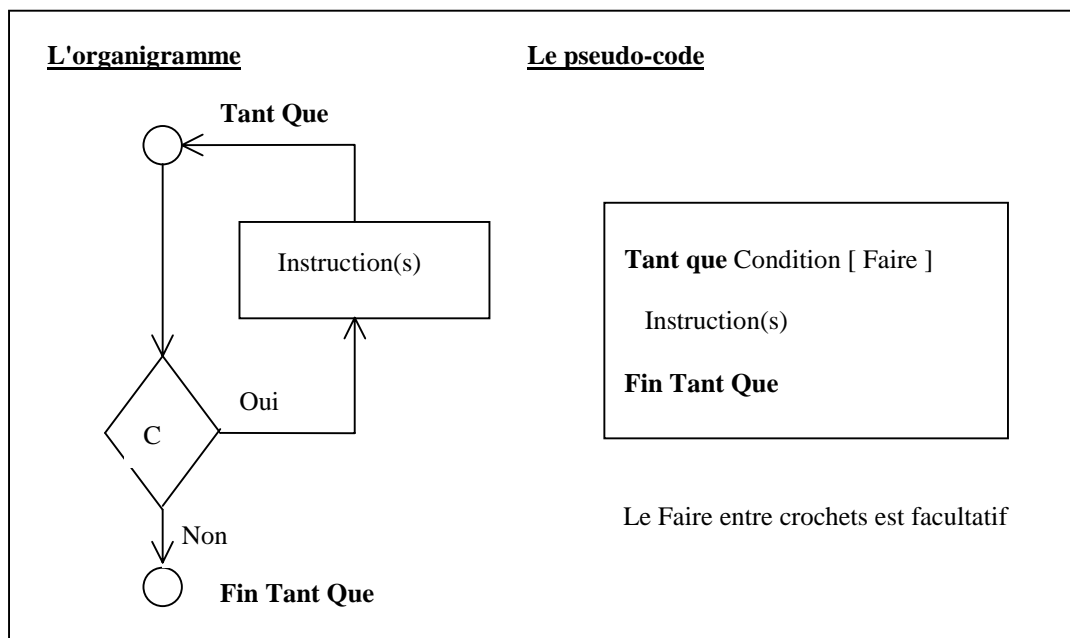


Fonctionnement : Si la **condition** est **vraie** alors on exécute la séquence d'**Instruction(s) 1**.

Si la **condition** est **fausse** (Sinon) on exécute la séquence d'**Instruction(s) 2**.

La structure de boucle

La boucle Tant Que Fin Tant Que



Dans cette présentation la boucle **Tant Que Fin Tant Que** est commentée.

Dans la boucle **Tant Que**, il y a toujours une **initialisation** de la variable de boucle avant de "rentrer" dans la boucle proprement dite.

Dans les exemples la boucle **Pour Fin Pour** sera expliquée.

Par contre la boucle **Répéter jusqu'à** ne sera pas utilisée. En effet cette boucle n'existe pas dans tous les langages (en Ada entre autre) et son comportement n'est pas identique dans tous les langages.

Remarque : La représentation sous la forme d'organigramme au niveau des structures fondamentales a été donnée à titre documentaire et ne sera pas utilisée.

Par contre le pseudo code sera toujours utilisé.

Les mots clés du pseudo-code (Rappels et compléments)

Les mots clés décalés vers la droite sont des synonymes

Les entrées - sorties

LIRE

SAISIR

ECRIRE

AFFICHER

VISUALISER

IMPRIMER

RETOUR_LIGNE

L'affectation

A ← B

Les opérateurs logiques

ET, OU, NON, NON-ET

Les commentaires

- Ceci est un commentaire -

Les tests

< <= > >= <> (≠)

Le symbole <> (différent) est utilisé en pseudo-code,. On peut aussi utiliser le symbole ≠ .

Les structures de contrôle

DEBUT FIN

SI ALORS FIN SI

SI ALORS SINON FIN SI

SI ALORS SINON SI SINON FIN SI
TANT QUE FIN TANT QUE
POUR VAR DE Val_Deb A Val_Fin FIN POUR
CAS PARMY FIN CAS

Les abréviations tolérées

FIN SI	FSI
TANT QUE	TQ
FIN TANT QUE	FTQ

Les déclarations de variables

Exemples :

N DE TYPE ENTIER

R DE TYPE REEL

B DE TYPE BOOLEEN

Dans cette présentation ces trois types seront les types les plus couramment utilisés.

Remarques : Les autres mots clés seront donnés au fur et à mesure de cette présentation en fonction des besoins.

On reprendra les mêmes exemples que dans le document concernant les instructions de base.

Ces exemples vont mettre en évidence la structure du programme :

La **structure de bloc** (assez rare),
La **structure alternative** (une seule exécution),
La **structure itérative** ou **la structure de boucle(s)**.
A ce stade les **structures de données** seront simples.

Exemple 1 : Plus petit, 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**.

Pseudo-code simplifié : Solution 1 (Trois SI successifs)

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 élégante ! La codification suivante utilise le "Si imbriqué".

Pseudo-code simplifié : Solution 2 (Utilisation du SI Imbriqué)

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

```

Pseudo-code simplifié : Solution 3 (Utilisation du SINON SI)

```

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

```

Ces exemples seront repris ultérieurement et seront "codés" en Ada.

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**.

Pseudo-code simplifié : Solution 1

LARGEUR DE TYPE ENTIER

N DE TYPE ENTIER

DEBUT

LIRE LARGEUR

$N \leftarrow 1$

TANT QUE $N \leq$ LARGEUR

VISUALISER '*'

$N \leftarrow N + 1$

FIN TANT QUE

RETOUR_LIGNE

FIN

Comme la boucle démarre à 1 et finit à Largeur, on pourra utiliser une boucle POUR. On utilisera une variable de boucle I. Cette variable n'est pas à déclarer dans le programme. En Ada le pas de la boucle est de toujours de 1 (ou de - 1).

Ces notions seront revues et commentées dans les instructions de base.

Pseudo-code simplifié : Solution 2

LARGEUR DE TYPE ENTIER

DEBUT

LIRE LARGEUR

POUR I de 1 A LARGEUR

VISUALISER '*'

FIN POUR

RETOUR_LIGNE

FIN

Déroulement du pseudo-code simplifié : Solution 2

On suppose que LARGEUR est égal à 3.

Mécanisme de la boucle POUR :

A l'entrée de la boucle POUR la variable de boucle I est initialisé à 1.

A chaque passage sur le POUR (y compris le premier) la comparaison est effectuée.

A chaque passage sur le FIN POUR La variable de boucle est incrémentée de 1 ($I \leftarrow I + 1$)

Numérotation des instructions

LIRE LARGEUR	1
POUR I de 1 à LARGEUR	2
VISUALISER '*'	3
FIN POUR	4
RETOUR_LIGNE	5

Déroulement du programme

Numéro Instruction	Action	Commentaire
1	Largueur \leftarrow 3	
2	$I \leftarrow 1$ Comparaison I à LARGEUR	Initialisation
3	"*"	
4	$I \leftarrow 2$	
2	Comparaison I à LARGEUR	
3	"*"	
4	$I \leftarrow 3$	
2	Comparaison I à LARGEUR	
3	"*"	
4	$I \leftarrow 4$	
2	Comparaison I à LARGEUR	$I > LARGEUR$ Sortie de boucle
5	RETOUR_LIGNE	

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 lignes contenant un nombre d'étoiles égal à Largeur.

Le nombre de lignes est égal à Hauteur.

Ce programme est constitué des **deux boucles imbriquées** : Une boucle externe ou "englobante" pour les lignes (Hauteur) et une boucle interne pour les colonnes (Largeur).

Pseudo-code simplifié :

LARGEUR, HAUTEUR DE TYPE ENTIER

DEBUT

 LIRE LARGEUR

 LIRE HAUTEUR

 POUR I de 1 à HAUTEUR

 POUR J DE 1 à LARGEUR

 VISUALISER '*'

 FIN POUR

 RETOUR_LIGNE

 FIN POUR

 RETOUR_LIGNE

FIN

Exercice (à faire)

Sujet : Dans l'exemple 3, donner les valeurs suivantes :

 Largeur : 3

 Hauteur : 2

Numéroter les instructions.

Donner le déroulement du programme sous forme de tableau.

Remarque : Si un lecteur souhaite proposer une solution satisfaisante, envoyez-moi un mail et s'il le souhaite elle sera mise en ligne à titre de correction. Je ne souhaite pas donner de façon systématique des éléments de correction. Les corrections systématiques inhibent le travail personnel.

=====