

ALGORITHMIQUE ET PROGRAMMATION INFORMATIQUE

Chapitre 3 : Les structures de contrôle

I. Programmation structurée

1. Définition

La programmation structurée est un style de programmation dans lequel la structure du programme est construite de la façon la plus claire possible. Les idées de base sont les suivantes :

- La programmation peut se faire en utilisant un nombre réduit de structures de programmes (structures de contrôles, sous-programmes).
- La présentation du programme doit refléter la structure du programme.
- Les tâches doivent être éclatées en sous-tâches (raffinement).
- Le raffinement doit être fait en tenant compte des données manipulées par les tâches. En particulier, les données manipulées par une tâche doivent être déclarées près de la tâche.
- Chaque sous-tâche doit avoir un point d'entrée unique et un point de sortie unique.

2. Objectifs

Les principaux objectifs de la programmation structurée sont les suivants :

- Le développement modulaire du programme;
- La facilité du codage;
- La rapidité du codage;
- La facilité du débogage;
- La facilité de la maintenance.

3. Structures de contrôles

Les diverses instructions élémentaires doivent être organisées afin que leur exécution puisse concourir au résultat souhaité. Les consignes par lesquelles on indique comment les actions commandées doivent s'organiser s'appellent **structures de contrôle**.

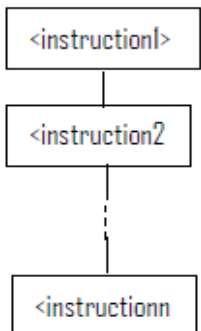
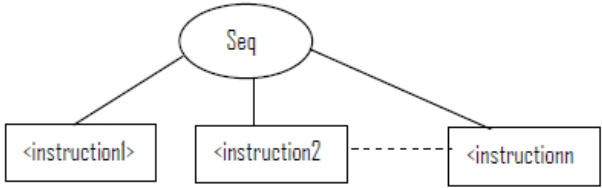
Une structure de contrôle va donc permettre d'enchaîner l'exécution de plusieurs instructions en déterminant comment et dans quel ordre elles doivent s'exécuter. En programmation structurée, toute logique de traitement est exprimée par trois structures de contrôle: la séquence, la conditionnelle et la répétitive.

II. Séquence

1. Définition

La séquence (ou structure séquentielle ou encore structure linéaire) indique que les instructions doivent être exécutées inconditionnellement les unes après les autres dans l'ordre de leurs apparitions. La séquence est la structure de contrôle par défaut.

2. Notation

En LDA	En algorithme	Arbre programmatique
<p><instruction1> <instruction2> ----- <instruction n></p>		

3. Fonctionnement

L'instruction <instruction 1> est exécutée d'abord, puis l'instruction <instruction 2>, ainsi de suite jusqu'à l'exécution de l'instruction <instruction n>.

4. Bloc d'instructions

Un bloc d'instructions est une séquence d'instructions regroupées pour ne former qu'une seule instruction.

Un bloc se note en **LDA** :

DEBUT

<instruction 1>
<instruction 2>
- - -
<instruction n>

FIN

III. Les Structures de contrôle

Dans le déroulement d'un algorithme, il arrive souvent que l'on ait à choisir une instruction ou entre deux ou plusieurs instructions suivant une condition portant sur la valeur de certaines données. Les structures conditionnelles ou structures de choix ou encore de sélection permettent cela.

1. Condition simple (si)

La sélection simple permet d'exécuter une instruction ou une suite d'instructions seulement si une condition est vraie et de ne rien faire de particulier si la condition est fausse.

a. Syntaxe :

```
SI <condition> ALORS  
    <suite d'instructions>  
FINSI
```

b. Fonctionnement

Premièrement : La condition <condition> est évaluée ;

Deuxièmement : Si la condition a la valeur « vrai », on exécute la suite d'instructions <suite d'instructions>.

Si la condition a la valeur « faux », on sort de la structure pour reprendre le cours normal du déroulement de l'algorithme.

2. Alternative

L'alternative permet de choisir, suivant la valeur d'une condition, entre deux suites d'instructions celle qui doit être exécutée.

a. Syntaxe :

```
SI <condition> ALORS  
    <bloc1>  
SINON  
    <bloc2>  
FINSI
```

b. Fonctionnement

Premièrement : La condition <condition> est évaluée ;

Deuxièmement : Si la condition a la valeur « vrai », seule la suite d'instructions <bloc1> est exécutée ;

Si la condition a la valeur « faux », seule la suite d'instructions <bloc2> est exécutée.

3. Sélection multiple

a. Imbrication de *Si*

Il arrive que l'on ait à opter un choix exclusif entre plusieurs suites d'instructions ; on procède alors par élimination en imbriquant les *Si*.

i. Syntaxe :

SI <condition> **ALORS**

 <Bloc 1>

SINON

SI <condition> ***ALORS***

 <Bloc 2>

SINON

 <Bloc 3>

FINSI

FINSI

ii. Exemple :

Ecrire un algorithme qui compare deux nombres.

ALGORITHME Comparaison

 a, b : REEL ;

DEBUT

 ECRIRE ("Entrer les nombres a comparer :") ;

 LIRE (a, b) ;

 SI (a = b) ALORS

 ECRIRE ("Les deux nombres sont égaux") ;

 SINON

 SI (a > b) ALORS

 ECRIRE (a, "est plus grand que", b);

 SINON

```
        ECRIRE (a, "est plus petit que", b);
    FINSI
FINSI
FIN
```

b. Choix multiple

L'imbrication des *Si* devient fastidieuse lorsque le nombre d'alternatives est élevé ; aussi on préfère utiliser la structure de choix multiple ou sélection multiple lorsque le choix se fait en fonction de la valeur d'une expression.

i. Syntaxe :

```
selon <identificateur>
    (liste de) valeur(s) : instructions
    (liste de) valeur(s) : instructions
    ...
    [autres: instructions]
```

Ou

```
SELON (<expr>) FAIRE
    <val1> : <bloc1>s
    <val2> : <bloc2>
    - - -
    <valn> : <blocn>
    [AUTRES : <bloc n+1>]
```

ii. Fonctionnement

Premièrement : Evaluation de l'expression <expr> qui doit être de type scalaire.

Deuxièmement : Exécution de la suite d'instructions <bloc i> si le résultat est <val i> (i = 1, ..., n).

Troisièmement : Exécution de la suite d'instructions <bloc i+1> si le résultat ne correspond à aucune des valeurs <val i> (i = 1, ..., n).

iii. Exemple :

selon abreviation

"M" : afficher(" Monsieur ")

"Mme" :afficher(" Madame ")

"Mlle" : afficher(" Mademoiselle ")

autres :afficher(" Monsieur, Madame ")

L'équivalent de cet exemple avec instruction Conditionnelle est

si abreviation = "M "

alors afficher("Monsieur")

sinon si abreviation = « Mlle »

alors afficher("Mademoiselle")

sinon si abreviation = "Mme"

alors afficher("Madame")

sinon afficher("Monsieur,Madame ")

finsi

finsi

finsi

iv. Remarque :

- Les conditions des alternatives doivent être mutuellement exclusives, c'est-à-dire pour une valeur de l'expression, on ne doit pouvoir exécuter qu'une seule des alternatives.
- Les conditions du choix multiple peuvent être :
 - un test d'égalité entre l'expression <expr> et une valeur
 - un test d'appartenance de l'expression à un ensemble de valeurs : <val i0>, <val i1>, ..., <val ip> : <bloc>
 - un test d'appartenance à un intervalle de valeurs : <val j0>...<val jm> :<bloc>

4. Les opérateurs logiques

Les conditions intervenant dans les structures conditionnelles sont des expressions logiques qui peuvent être complexes c'est-à-dire formées de conditions simples reliées entre elles par des opérateurs logiques ET, OU ou NON.

Le chapitre sur la *Logique propositionnelle* nous en donne plus de détails.

TD2 :

Exercice 1 : On s'intéresse au bloc d'instructions suivant:

```
1: ...  
2: Si  $(C - B) = B$  Alors default  
3:    $A \leftarrow A + 1$   
4:    $C \leftarrow C + B$   
5:    $B \leftarrow A$   
6: Sinon default  
7:    $B \leftarrow A$   
8:    $A \leftarrow A - 1$   
9:    $C \leftarrow C \times B$   
10: ...
```

Donner les valeurs des variable A, B et C \blacklozenge la sortie de ce bloc d'instructions:

- pour $A \leftarrow 2, B \leftarrow 3,$ $C \leftarrow A \times B$
- pour $A \leftarrow 1, B \leftarrow 5, C \leftarrow 3$
- pour $A \leftarrow -3, B \leftarrow A \times A, C \leftarrow B - 5$
- pour $A \leftarrow 8, B \leftarrow 3,$ $C \leftarrow A - 2$
- $A \leftarrow 10, B \leftarrow 1,$ $C \leftarrow -B + A^2$

Exercice 2 :

Dans un programme de calcul d'une facture, on veut effectuer une remise de 1% si le montant de la facture dépasse 1000F.

Ecrire l'algorithme qui affiche le montant payé de la facture à partir du prix unitaire, de la quantité et du taux de la tva.

Exercice 3:

1° Ecrire un algorithme qui mémorise et affiche la somme ou le produit de deux nombres selon le choix de l'utilisateur.

2° Résoudre l'équation $ax+b=0$

Exercice 4 :

Ecrire un algorithme qui calcule et affiche la circonférence des cercles dont on lui fournit le rayon ; pour arrêter l'utilisateur doit entrer un nombre négatif ou nul.

Exercice 5 :

Algorithme qui permet à l'utilisateur de calculer la circonférence de plusieurs cercles à partir du rayon jusqu'à ce qu'il décide d'arrêter.

Exercice 6 :

Ecrire un algorithme qui compare deux nombres.

Exercice 7

Ecrire un algorithme qui demande à l'utilisateur d'entrer deux nombres et afficher le plus grand.

Exercice 8

Ecrire un algorithme qui demande à l'utilisateur d'entrer 3 nombres et afficher le plus grand.

Exercice 9

Ecrire un algorithme qui demande à l'utilisateur les données suivantes

PHTU

Nombre d'article

TVA

Et calculer le PTTC selon la règle suivante telle que :

- Remise = 10% si nombre d'article > 10
- Remise = 0.75% si $5 < \text{nombre d'article} \leq 10$
- Remise = 0.50% si nombre d'article ≤ 5 .

Exercice 10

Ecrire un algorithme qui demande à l'utilisateur la température de l'eau et afficher son état (solide, liquide, vapeur).

Exercice 11

Soit l'équation au 1er ordre : $ax+b=0$

Ecrire un algorithme qui demande à l'utilisateur a et b afin de calculer la solution de l'équation.

Exercice 12

Ecrire un algorithme pour résoudre une équation au 2ème ordre sous la forme $ax^2+bx+c=0$

NB : traiter tous les cas possibles.

Exercice 13

Ecrire un algorithme qui demande à l'utilisateur d'entrer la note et qui affiche la mention comme suite :

« Faible » si $\text{note} < 10$

« Passable » si $10 \leq \text{note} < 12$

« A. Bien » si $12 \leq \text{note} < 14$

« Bien » si $14 \leq \text{note} < 16$

« T. Bien » si $16 \leq \text{note} < 18$

« Excellent » si $18 \leq \text{note} < 20$

Exercice 14

Ecrire un algorithme qui demande l'âge de l'enfant en suit il informe ça catégorie

« Poussin » de 6 ans à 7 ans

« Papille » de 8 ans à 9 ans

« Minime » de 10 ans à 11 ans

« Cadet » après 12 ans