Unité fondamentale :……..

Année d'étude : Master 01 géomatique

Cours N° 03 : *les structures de contrôle*

Réalisé par : Hadjira Lyes

C'est quoi les structures de contrôle ?

Reprenons l'exemple d'un chauffeur d'ambulance que vous allez guidé à destination; Pour cela vous lui direz par exemple :

« *Suivez cette route jusqu 'au carrefour, ensuite vous regardez à droite si vous ne voyez pas d'embouteillage prenez là, et à environ 100 mètres prenez à gauche, et votre adresse est enfin de rue à gauche. Sinon prenez à gauche puis prenez la prochaine rue à droite et votre adresse est en fin de rue à droite.* »

Nous voyons bien que notre chauffeur doit prendre des décisions en fonction de deux situations :

- Prendre à droite dans le cas où la route est libre ou
- Prendre gauche dans le cas où la route est encombrée.

Ainsi, l'algorithme se traduit par les instructions suivantes :

- Suivre la route jusqu'au carrefour
- Si pas d'embouteillage alors
- Prenez à droite
- Roulez environ 100 mètres
- Prenez à gauche
- S'arrêter en fin de rue à gauche
    - Sinon
- Prenez à gauche
- Prenez la prochaine rue à droite S'arrêter en fin de rue à droite.

Ce que vous venez de faire !, vous avez introduit une structure de contrôle dans votre algorithme :

```
Si <condition> alors traitementl
Sinon traitement2
```

Nous verrons dans ce qui suit les différentes structures qu'on peut avoir dans un algorithme.

Une structure de contrôle sert à contrôler le déroulement d'un traitement.

Un traitement peut s'exécuter de différentes manières :

1. Séquentiellement,
2. Alternativement
3. Répétitivement.

1. Le traitement séquentiel

Une séquence est une suite d'instructions qui s'exécutent l'une à la suite de l'autre.

```
Début
 Instruction 1
 Instruction 2
 ….
 Instruction N Fin.
```

Séquence d'instructions

Exemple

Soit à additionner deux nombres A et B, les opérations à effectuer se présentent ainsi :

- Introduire la valeur de A
- Introduire la valeur de B
- Additionner A et B (A+B)
- Afficher le résultat

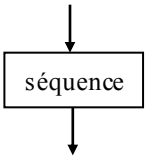
Les opérations de 1 à 4 s'exécutent séquentiellement, c'est-à-dire l'une à la suite de l'autre.

Suite d'instructions s'exécutant séquentiellement

Algorithme

```
Début
      Lire(A)
      Lire(B)
      S <- A+B
      Écrire(S)
Fin.
```

Organigramme d'une séquence :



2. Le traitement alternatif

Le traitement alternatif repose sur un test, ce dernier est exprimé avec une ou plusieurs conditions.

Si « Condition_Vérifié » Alors Actions

- La condition peut prendre une valeur logique Soit «Faux», soit «Vrai»
- Une condition peut être composée, c'est-à-dire, qu'on peut avoir une ou plusieurs conditions à vérifier en même temps ; pour cela on utilise les opérateurs logiques ET, OU, NON et XOU

***Si** Condition 1 _Vérifiée **ET** Condition2_Vérifiée Alors Actions*

a- L'alternative simple : (si .. alors)

- **Formalisme**

Si <condition> alors
<Séquence>
Souvent dans « Séquence » on peut avoir plusieurs instructions, alors il faut regrouper les instructions dans un bloc délimité par Début et Fin.

```
Si  <condition>  alors
Début
     Instruction1
     Instruction2
     InstructionN
Fin
```

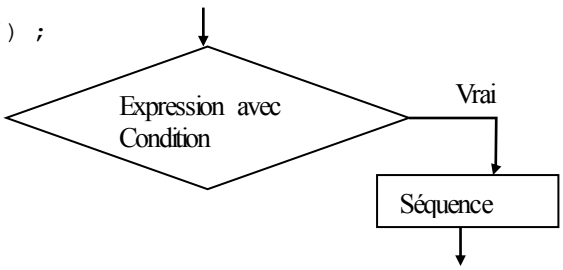
Principe: Si la condition est vraie, alors la séquence d'instructions s'exécute, Sinon ne rien faire.

Exemple1 : Si demain il pleut alors j'annulerai ma visite.

Exemple2

```
Si  A>B    alors
écrire   ( ' Choisir  la  valeur  de  A') ;
```

Organigramme d'une alternative simple



b-L'alternative composée (Si ..alors.. Sinon)

» Formalisme

Si < condition > alors

< Séquence 1>

Sinon < Séquence2 >

> Principe

Si la condition est vérifiée (Vrai), alors Séquence 1 sera exécutée, sinon, c'est Séquence2 qui le sera.

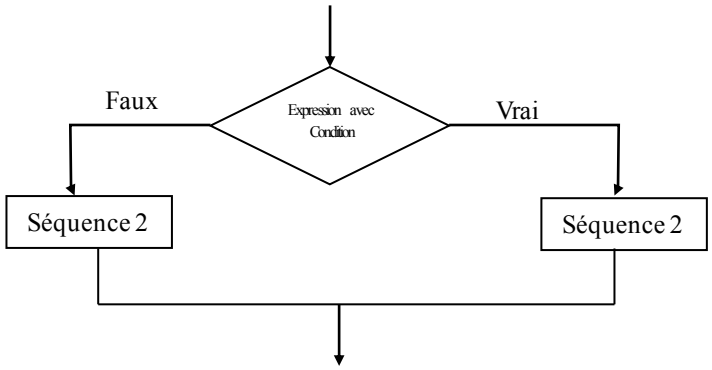
Exemple 1: Si demain il pleut alors j'annulerai ma visite. Sinon je visiterai mes parents à la campagne.

Exemple2

```
Si A>B alors
Écrire ('Choisir la valeur de A')    «1»
Sinon
Écrire   ('Choisir  la valeur de B') «2»
```

Remarque

« 1 » et «2 » ne peuvent jamais s'exécuter ensemble, c'est l'une ou l'autre



c- L'alternative multiple (choix): (Choix <Variable> de « valeurs i»)

Formalisme

Choix <variable de choix> de
choix 1 : Séquence_1
choix 2 : Séquence_2
choix n : Séquence_N
Sinon
Séquence
Fin;

Remarques

La variable de choix doit être de type ***Entier positif ou*** de type ***caractère.***

« Séquence_i » peut être remplacé par un bloc d'instruction délimité par Début et Fin.

```
Choix i : Début
              Instructionl
              Instruction2
              …
              instructionN
              Fin
```

- **Principe**

Cette structure de contrôle présente plusieurs cas d'exécution du traitement, mais un seul sera exécuté.

Remarque

La structuré Choix…de peut ne pas avoir de sinon.

Exemple 1

Selon le cas qui se présente je prendrai la décision qu'il faut :

Cas l : si mon chiffre d'affaires est inférieur à 600000 DA, alors aucun investissement n'est à envisager.

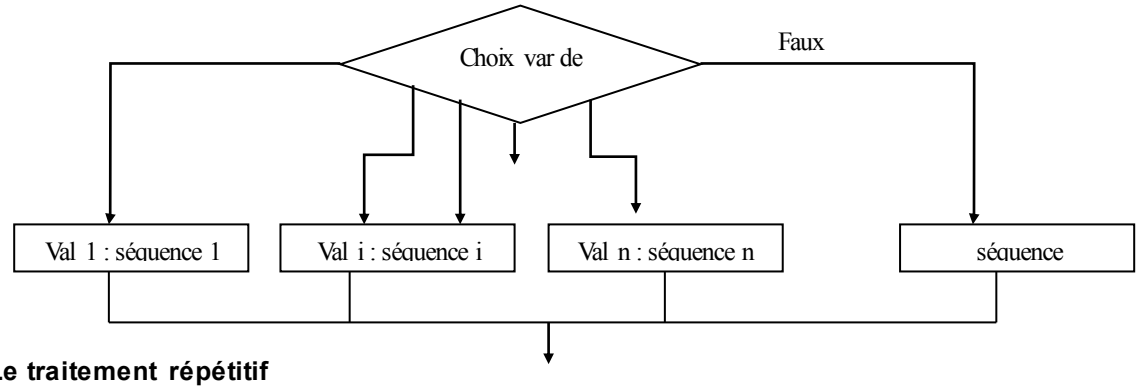
Cas 2 : si mon chiffre d'affaires atteint les 1000000 DA, alors j'achèterai un véhicule utilitaire.

Cas 3 : si mon chiffre d'affaires dépasse les 1500000 DA, alors j'achèterai un véhicule utilitaire et un autre de service.

Exemple2: Afficher les noms des jours de la semaine ;
j est de type entier

```
Lire (j)
 Choix j  de
         1 : Écrire ('C'est Samedi ')
         2 :  Écrire ('C'est Dimanche')
         3 : Écrire ('C'est Lundi')
         4 : Écrire ('C'est Mardi')
         5 :  Écrire ('C'est Mercredi')
         6 : Écrire ('C'est Jeudi')
         7 :  Écrire ('C'est Vendredi')
Sinon
Écrire ('ce n'est pas une journée')
Fin
```

Dans cet exemple J est de type entier positif il prend la valeur 1 ou 2 ou 3 ou 4 ou 5 ou 6 ou 7.



3. Le traitement répétitif

On dispose de trois (03) structures différentes pour contrôler un traitement répétitif :

- La boucle Tant que ..... faire.
- La boucle Pour ..... faire.
- La boucle Répéter ..... jusqu'à.

a-La boucle Tant que : (Tant que ... Faire)

- **Formalisme**

Tant que < condition vérifiée > faire

< Séquence >

- **Principe**

Tant que la condition est vérifiée, la séquence s'exécute, elle ne s'arrêtera que lorsque la condition n'est pas vérifiée.

Exemple 1

Tant que le clou n'est pas totalement enfoncé, frapper avec le marteau.

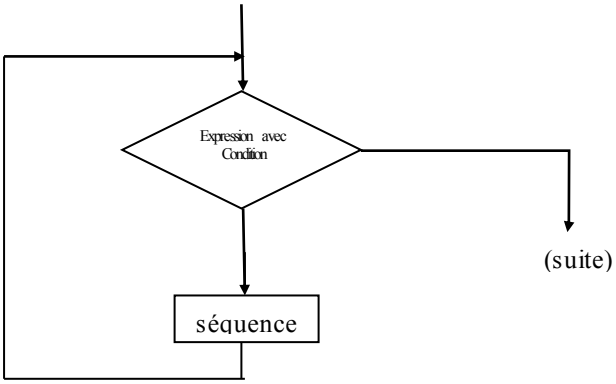
- *Opération répétée :* Frapper avec le marteau.
- *Condition d'arrêt :* Clou totalement enfoncé.

Exemple2 : Afficher les 100 premiers nombres entiers positifs.

```
I <- 1
Tant que I < 100 faire
Début
I <- I+1
Écrire (I)
Fin.
```

NB : I doit être de type entier positif ; (I : Entier)

Organigramme de la boucle Tant que



Éléments de la boucle Tant que

*** La condition d'arrêt**

Sans condition d'arrêt la boucle devient infinie (cas à éviter). Dans l'exemple 1 (I>100) est la condition d'arrêt.

» Le compteur de la boucle Il sert à faire avancer la boucle (Incrémentation).

Dans l'exemple 1 (`I <- 1+1`) est l'instruction qui permet d'incrémenter la boucle.

*** Initialisation du compteur**

La condition d'arrêt porte sur le compteur de la boucle I, pour cela, la boucle doit commencer par une valeur initiale (I<-0).

Exemple 3 : Algorithme qui permet d'afficher les nombres négatifs.

```
Algorithme Négatif
 N : entier
 Bool: logique
Début
Bool <- faux
Tant que bool = faux faire
Début
      Lire (N)
      Si  N < 0 alors
        écrire (N , 'est un nombre négatif')
      Sinon
   Bool      vrai                    Valeur qui permet
   Fin                               de quitter la boucle
Fin.
```

- **Commentaires**

Tant que le nombre introduit est négatif, la boucle continue de s'exécuter, elle ne s'arrêtera que lorsqu'on introduit un nombre positif (bool← vrai) .

b-La boucle pour : (Pour ... Faire)

- **Formalisme**

Pour < Compteur > ← <V initiale> à <V finale> faire
<Séquence>

- **Principe**

Pour<Compteur>allant de<valeur_i > *à* <valeur_f>faire
<séquence>

Pour <compteur> allant de valeur initiale jusqu'à valeur finale, exécuter la séquence d'instructions.

Le nombre de fois que la séquence s'exécute est égal à (valeur finale - valeur initiale +1) ce qui veut dire que le nombre est connu d'avance.

Exemple 1 : Mettre des étiquettes pour 100 produits.

Pour les 100 produits en chaîne, mettre une étiquette pour chacun.

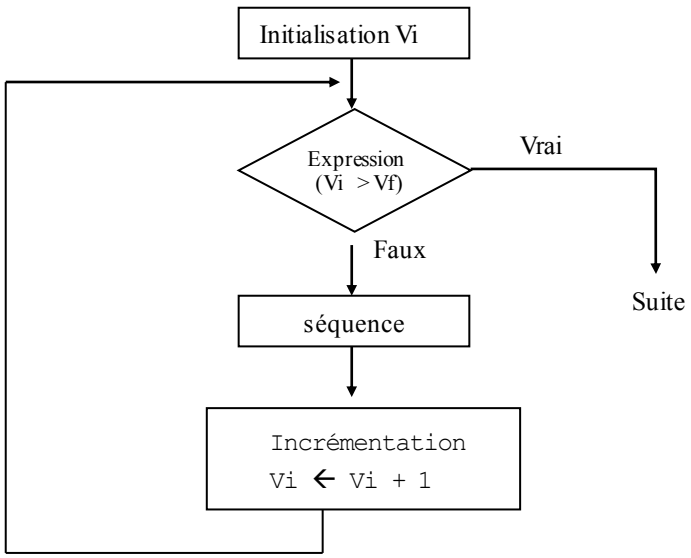
* *Opérations répétées :* Mettre une étiquette.

* *Condition d'arrêt :* 100 produits tous étiquetés.

Exemple2 : Afficher les 100 premiers nombres entiers positifs

```
I : entier
Début
Pour  I <- 1 à 100 faire
Écrire (I)
Fin.
```

Organigramme de la boucle Pour



Éléments de la boucle pour

• Valeur initiale/finale

Le compteur de la boucle doit commencer le comptage à partir de la valeur initiale et finir avec la valeur finale, (ici *Vi=1, Vf=100).*

Remarque

On voit clairement ici que la boucle Pour ne nécessite pas d'incrémentation ni d'initialisation au préalable.

c-La boucle Répéter : (répéter ... jusqu'à)

• Formalisme

Répéter

```
        <Séquence>
        jusqu'à  <Condition vérifiée>
```

• Principe

Répétez l'exécution de la séquence jusqu'à ce que la condition soit vérifiée.

Exemple1 Frapper avec le marteau, jusqu'à avoir le clou totalement enfoncé.

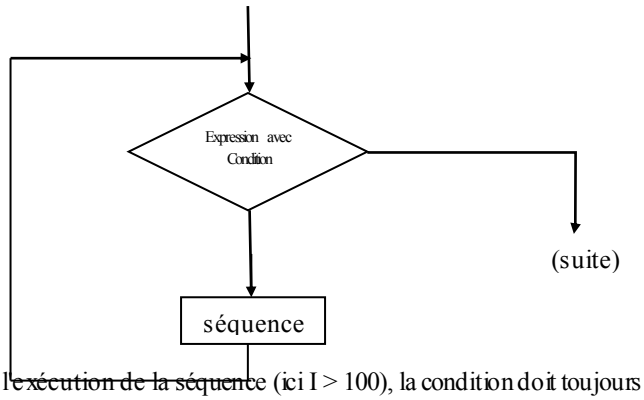
- *Opérations répétées :* **Frapper** avec **le marteau.**
- *Condition d'arrêt :* **Clou totalement enfoncé.**

Exemple2

Afficher les 100 premiers nombres entiers positifs

```
I : entier
Début
I <-1
Répéter
 Écrire (I)
 I <- I + 1
Jusqu'à  I > 100
Fin.
```

Organigramme de la boucle répéter



Éléments de la boucle Répéter

• Condition d'arrêt

Celle-ci doit être définie d'une façon exacte pour arrêter l'exécution de la séquence (ici I > 100), la condition doit toujours porter sur le compteur de la boucle (dans le cas où le nombre d'itérations est connu).

• Incrémentation

Dans ce cas il y a nécessité d'incrémenter le compteur de boucle, (quand le nombre d'itérations est connu).

• Initialisation

Le compteur doit être initialisé à une valeur initiale pour débuter l'exécution.

Exemples

Algorithme qui introduit un entier, vérifie s'il est négatif et l'affiche.

```
Algorithme lecture
Bool : logique
N    : entier
Début
  Bool <- faux
  Répéter
      Lire (N)
      Si   N<0 alors
      Début
       écrire (N)
       Bool <- faux
      Fin
      Sinon
       Bool <- vrai
       Jusqu'à Bool  = Vrai
Fin.
```

Dans cet exemple, le nombre d'exécution de la boucle Répéter, n'est pas connu, donc il ya nécessité d'utiliser une variable logique (`bool`) pour contrôler la boucle.

Remarque

La condition d'arrêt (`bool = vrai`) signifie, que pour arrêter l'exécution de la boucle, il suffit d'introduire un nombre positif (N>0);

La boucle Répéter fonctionne de la même manière que la boucle Tant que, à une différence près.

Cas de la boucle infinie

Par principe les boucles infinies sont à éviter (sauf si cela est fait d'une manière volontaire « développeur de virus »).

On appelle boucle infinie une boucle dont la condition d'arrêt n'est jamais vérifiée, ce qui pousse la boucle à tourner indéfiniment jusqu'à ce que le programme se bloque.

Exemple

```
I ← 1
J ← 2
Tant que j > i faire
Début
  J ← j+1
Fin
```

Cette boucle ne s'arrêtera jamais puisque j est toujours supérieur à i.

Remarques générales sur les boucles

- La boucle «`Tant que`» peut ne pas s'exécuter du tout si la condition d'accès à la boucle n'est pas vérifiée.
- La boucle «`Répéter`» s'exécute au moins une fois.
- La boucle «`Pour`» s'exécute toujours, le nombre d'itérations est connu d'avance.
- La boucle «`Tant que`» et «`Répéter`» nécessitent toujours une initialisation, et une incrémentation du compteur de boucle (bien sûr, dans le cas où le nombre d'itérations est connu).
- Dans le cas où le nombre d'itérations n'est pas connu, la variable sur laquelle porte la condition doit être possible à atteindre, et doit être initialisée au préalable, autre cas, on risque de générer une boucle infinie.
- La boucle «`Pour`» est utilisée exclusivement dans le cas où le nombre d'itérations est connu, on utilisera un compteur entier.
- Si le nombre d'instructions du test ou de la boucle est supérieur à 1
  celles-ci devront être mises entre `Début et fin`. .
- Ceci n'est pas obligatoire dans le cas où il y'a une seule instruction.

Remarque

Certains ouvrages adoptent un formalisme particulier pour encadrer les séquences d'instructions à l'intérieur d'une structure de contrôle en notant à chaque fin de structure `FinSi`, `FinPour`, `FinTantque`, ou `FinRepeat` à la place de « `Début et Fin` ».

Alors ne soyer pas gêné si vous rencontrez de telles notations dans d'autres ouvrages.

Exemple

`Tantque <condition> faire`
`Instructionl`
`Instruction2`
`…`
`FinTanque`
	`Si <condition> Alors`
	`Intsructionl`
	`Instruction2`
	`…`
	`FinSi`