3. Structures de contrôles

3.1. Introduction

- Les ***structures de contrôle*** encore appelées ***instructions (actions) structurées*.**
- Elles permettent d'exprimer la façon dont l'enchaînent d'exécution des instructions d'un algorithme.
- Il existe trois structures fondamentales :
  - ✓ Actions séquentielles,
  - ✓ Actions conditionnelles,
  - ✓ Actions de répétition.

3.2. Organigramme d'un algorithme

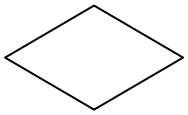
- Pour décrire les structures de contrôles précédentes, on utilise une notation textuelle (***algorithme***) et une notation graphique (***organigramme***).
- Dans un organigramme on utilise les symboles suivants :


Représente le début et la Fin de l'organigramme


Entrées / Sorties : Lecture des données et écriture des résultats.


Représente les Actions (traitement)

Représente l'ordre d'exécution des opérations(Enchaînement)


Représenter les conditions (Tests et décision)

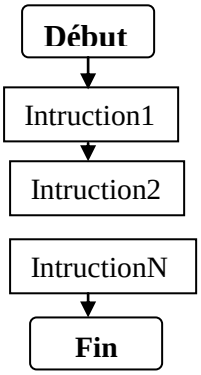
3.3. Actions séquentielles :

- Les Actions séquentielles se présentent sous forme d'une suite ordonnée d'instructions regroupée en un bloc.

Syntaxe :

```
Début
    Intruction1 ;
    Intruction2 ;
     ……
    Instruction N ;
Fin.
```

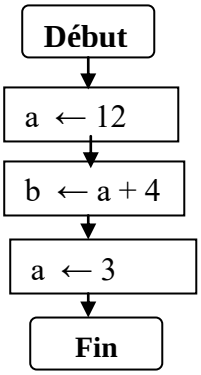
>>>>>>>>>



Exemple :

```
Algorithme exp_sequ
        a, b : entier;
Début
   a← 12 ;      (Instruction 1)
   b←a + 4 ;   (Instruction 2)
   a ← 3 ;       (Instruction 3)
Fin.
```

>>>>>>>>>



Trace d'exécution :

	a	b
Au début	?	?
Instruction 1	12	?
Instruction 2	12	16
Instruction 3	3	16

3.4. Action conditionnelle

3.4.1. Action conditionnelle simple

- Elle est composée de deux parties : ***condition*** et ***action***.
  - ✓ La partie ***(condition)*** décrit un état qui peut être vrai ou faux (expression de type Booléen).
  - ✓ La partie **< *Bloc d'actions* >** représente un morceau d'un algorithme (une ou plusieurs instructions).

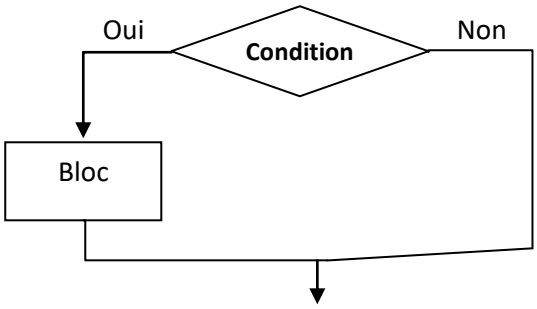
Syntaxe :

…….
Si (condition) **alors**

< Bloc d'actions (instructions)>

FinSi

……

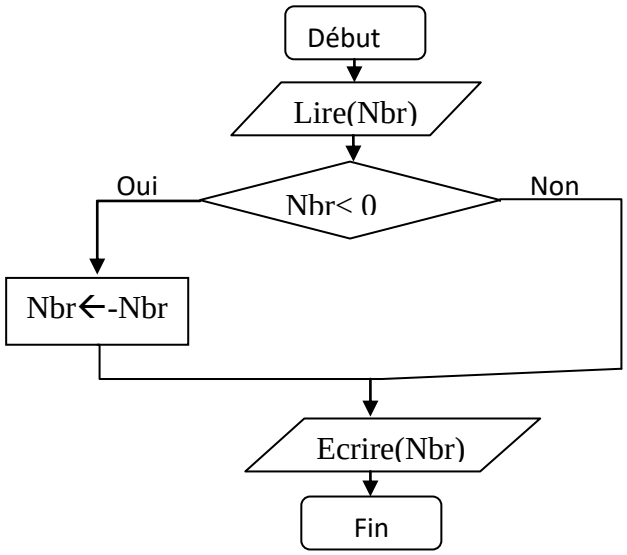


Exécution de l'action conditionnelle :

- Si la condition est ***vérifiée (vrai)***, les instructions du < Bloc d'actions > sont exécutées et on continue l'exécution des actions (instructions) situées après le Finsi.
- Si la condition n'est ***pas vérifiée (faux)***, la partie < Bloc d'actions > à l'intérieur du Si n'est pas exécutée et on poursuit l'exécution de l'algorithme directement à partir de l'instruction qui suit le FinSi.

Exemple : Ecrire un algorithme qui permet de lire un nombre réel identifier par 'Nbr', puis donne sa valeur absolue.

```
Algorithme val_abs
        Nbr : réel ;
Début
         Lire(Nbr) ;
        Si (Nbr< 0) alors
                Nbr← - Nbr ;
        FinSi
        Ecrire (Ndr) ;
Fin.
```



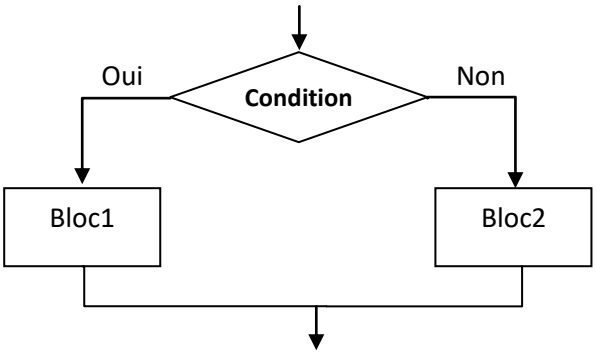
Remarque :

- la condition est une ***expression*** de type ***Booléen*** donc elle doit comporter au moins un opérateur de comparaison (<, >, =,!=,…etc.) ou une ***variable Booléen***.

3.4.2. Action alternative

Syntaxe :

……
Si (condition) **alors**
< Bloc d'actions**1** (instructions) >
Sinon
< Bloc d'actions**2** (instructions) >
FinSi
……



Exemple : Ecrire un algorithme qui permet de lire deux nombres réels, puis détermine le plus grand entre eux ?

```
Algorithme PlusGrNbr
        x, y, Max : réel ;
Début
   Lire(x) ;
   Lire(y) ;
   Si (x > y) alors
        Max← x ;
   Sinon
        Max← y ;
   FinSi
   Ecrire ('Le plus grand nombre est :', Max);
Fin.
```

Remarque:

- On peut voir une structure de contrôle en entier comme une seule action (< Bloc d'actions >) donc il peut y a avoir plusieurs ***structures de contrôle imbriquées***.

Exemple : Ecrire un algorithme qui permet de lire deux nombres réels, puis détermine le plus grand entre eux ?

```
Algorithme plus_G
        x, y, Max : réel ;
Début
  Lire(x, y) ;
  Si (x = y) alors
     Ecrire (''Les deux sont égaux'');
  Sinon   // x ≠ y
     Si (x > y) alors
       Max ← x ;
       Ecrire (''Le plus grand nombre est :'', Max);
     Sinon // x < y
       Max ← y ;
       Ecrire (''Le plus grand nombre est :'', Max);
      FinSI
   FinSi
Fin.
```

3.3.2. Action de choix multiples :

- Elle permet de distinguer plusieurs cas selon les ***valeurs*** d'une ***expression***.
- Le « **si** » permet de distinguer deux cas alors que le « **Selon cas »** permet de distinguer un grand nombre de cas.

Syntaxe :

Selon cas (expression**)**

Cas 1 : <bloc actions 1>

Cas 2 : <bloc actions 2>

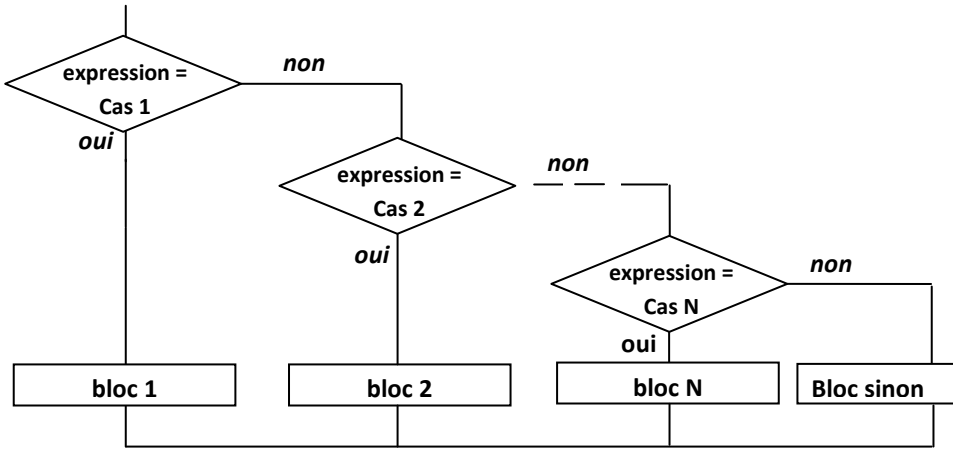
……..

Cas N : <bloc actions N>

Sinon // Le sinon est optionnel

<bloc action Sinon>

Fin cas



Exemple : Ecrire l'algorithme qui permet de lire un chiffre (0, 1, 2, 3,4) puis donne son non (zéro, un, deux, trois, quatre) ?

Action alternative imbriquée	**Action à choix multiple**
Algorithme Nom_chif n : entier ; **Début** Ecrire ("donnez votre chiffre entre 0 et 4 : ") ; Lire (n) ; Si (n=0) Alors Ecrire ("Zéro") ; Sinon Si (n=1) Alors Ecrire ("Un") ; Sinon Si (n=2) Alors Ecrire ("Deux") ; Sinon Si (n=3) Alors Ecrire ('Trois') ; Sinon Si (n=4) Alors Ecrire ("Quatre") ; Sinon Ecrire ("erreur de la saisie ") ; Fin si Fin si Fin si Fin si Fin si **Fin.**	**Algorithme** Nom_chif n : entier ; **Début** Ecrire ("donnez votre chiffre entre 0 et 4 : ') ; Lire (n) ; **Selon Cas (**n) 0 : Ecrire (" Zéro') ; 1 : Ecrire ("Un') ; 2 : Ecrire ("Deux") ; 3 : Ecrire ("Trois") ; 4 : Ecrire ("Quatre") ; **Sinon** Ecrire ("erreur de la saisie") ; **Fin cas** **Fin.**

3.4. Actions de répétition (actions itératives ou boucles)

- Les itérations (boucles) permettent de répéter plusieurs fois une même série d'instructions.

3.4.1. Action itérative « Tantque » (while) :

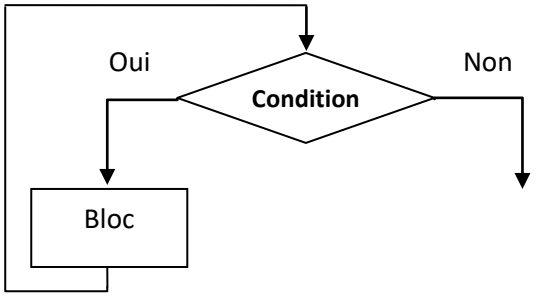
- Elle permet la répétition d'une ou plusieurs actions ***tant que la condition reste vérifiée.***

Syntaxe :

Tant que (condition) **Faire**

< Bloc d'actions >

Fin tant que



- Il s'agit de répéter l'exécution des instructions du < Bloc d'actions > tant que la condition est vérifiée et arrêter leur exécution dès que la condition devient non vérifiée.

Exemple :

Ecrire un algorithme qui calcule la somme des n premiers nombres entiers positifs ?

```
Algorithme som-ent1
        n ,i,som: entier;
Début
    Lire(n) ;
    Som← 0 ;            //la somme est initialisée à 0
    i←0 ;
    Tantque (i≤ n) Faire
        Som←Som + i ;
        i←i + 1;                    // incrimination de i
    Fin tan que
    Ecrire(som) ;
Fin.
```

Remarque :

- L'action répétitive **Tantque** est utilisée quand on ne connaît pas à priori le nombre de répétitions et lorsqu'il est possible de ne pas du tout exécuter les actions du bloc.

3.4.2. Action itérative « pour » (For) :

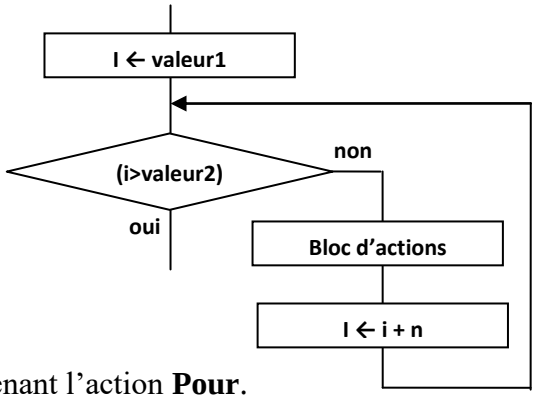
- Elle utilise une variable appelée variable de contrôle, ou encore ***compteur*** d'itérations pour contrôler le nombre d'itérations.

Syntaxe :

Pour i **allant de** Valeur1**à** Valeur 2(**pas**= n) **faire**

<Bloc d'action>

Fin Pour



Exemple :

Nous reprenons l'exemple précédent en utilisant maintenant l'action **Pour**.

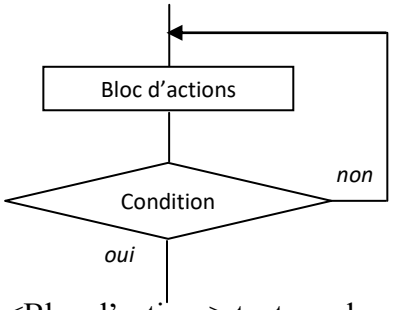
```
Algorithme som-ent
    n , i ,som;entier;
Début
  Lire(n) ;
  Som← 0 ;
  Pour i←1 à nfaire
       Som←Som + i ;
  Finpour
  Ecrire(som) ;
Fin.
```

Remarques :

- L'action **pour** est utilisée quand le nombre de fois **connu d'avance**. Pour cela, il faut préciser la valeur initiale et la valeur finale et éventuellement le pas.
- L'incrémentation du compteur se fait de manière automatique. Le pas, lorsqu'il n'est pas spécifié, est égal à 1.

3.4.3. Action itérative « Répéter » :

- Cette action exprime la répétition d'une ou plusieurs actions jusqu'à ce que la condition devienne vérifiée.



Syntaxe :

Répéter

<Bloc d'actions>

Jusqu'à (condition)

- Il s'agit de répéter l'exécution des instructions du bloc <Bloc d'actions> tant que la condition non vérifiée et arrêter leur exécution dès que la condition devient vérifiée.

Exemple :

Reprendre l'algorithme qui calcule la somme des N premiers nombres entiers positifs en utilisant l'action Répéter.

```
Algorithme som-ent4
        n , i ,som:entier;
Début
   Lire(n) ;
   Som← 0 ;    // la somme est initialisée à 0
    i← 0 ;       // l'initialisation de i
    Répéter
        Som← Som + i ;
        i←i + 1;     // incrémentation de i
   Jusqu'à (i> n)
   Ecrire (som) ;
Fin.
```

Remarque :

- L'action « ***Répéter »*** est utilisée quand on ne connaît pas à priori le nombre de répétitions et lorsque on doit exécuter les actions du bloc au moins une fois (car l'évaluation de la condition se trouve à la fin).

3.4.4. Structures de contrôle imbriquées

Exemples :

Ecrire un algorithme qui lit un nombre naturel ***N*** puis calcule et affiche la somme :

S = 1 – 2 + 3 - … N

Ecrire un algorithme qui lit un nombre naturel***N***calcule et affiche la somme :

S = (1 + 2 + 3 + · · · + N) + (2 + 3 + · · · + N) + · · · + (N-1 + N) + N.

```
Algorithme S2
        i, j, S, S1 : entiers ;
Début
   S ← 0 ;
   Pour i=1 `aNFaire
        S1 ← 0;
        Pour j allant de i a N Faire
           S1 ← S1 + i ;
        Fin Pour
        S ← S + S′
   Fin Pour
   Ecrire(S) ;
Fin.
```

Remarque :

- L'action ***Tantque*** est utilisée quand on ne connaît pas à priori le nombre de répétitions et lorsqu'il est possible de ne pas du tout exécuter les actions du bloc.
- L'action ***pour***est utilisée quandle nombre de fois **connu d'avance**. Pour cela, il faut préciser la valeur initiale et la valeur finale et éventuellement le pas.
- L'incrémentation du compteur se fait de manière automatique. Le pas, lorsqu'il n'est pas spécifié, est égal à 1.
- L'action ***Répéter*** est utilisée quand on ne connaît pas à priori le nombre de répétitions et lorsque on doit exécuter les actions du bloc au moins une fois (car l'évaluation de la condition se trouve à la fin).