Chapitre 3
Les structures conditionnelles et les structures itératives

I. Les structures conditionnelles

1. Structure conditionnelle à un choix

- Définition :

Pour cette structure, si la condition logique est satisfaite (vraie), alors le traitement sera exécuté. Dans le cas contraire, rien ne se passe.

- Syntaxe :

```
Si (Condition) Alors
        <Traitement >
Fin Si
```

- Remarques :

- Une condition est une expression logique qui peut être présentée sous forme de comparaison en utilisant les opérateurs : <, <=, >, >=, =, <>.
- La condition peut être simple ou composée.

Pour la condition composée, on doit utiliser les opérateurs logiques (ET, OU et NON).

Exemple (condition simple) :

```
Si (Age>18) Alors
        Ecrire (Moyenne)
Fin Si
```

Exemple (condition composée) :

```
Si ((x<0 ET y<0) OU (x>0 ET y>0)) Alors
        Ecrire ("Le produit de x ey y est positif")
Fin Si
```

2. Structure conditionnelle à deux choix

- Définition :

Pour cette structure, si la condition logique est satisfaite (vraie), alors le traitement 1 sera exécuté, sinon le traitement 2 sera exécuté.

- Syntaxe :

```
Si (Condition) Alors
        <Traitement 1>
Sinon
        <Traitement 2>
Fin Si
```

Exercice : Ecrire un algorithme qui lit un nombre entier et affiche sa valeur absolue.

```
Algorithme Absolue
Var : x, a : entier
Début
Ecrire ("Donner un entier") Lire (x)
Si (x>0) Alors
        a ← x
Sinon
        a ← -x
Fin Si
Ecrire ("La valeur absolue de", x, "est : ", a)
Fin
```

3. Structure conditionnelle imbriquée

- Définition :

Cette structure fait appel à plusieurs traitement (plus que deux traitements).

- Syntaxe :

```
Si (Condition) Alors
        <Traitement 1>
Sinon
        Si (Condition 2) Alors
                <Traitement 2>
        Sinon
        ⋮
                Si (Condition N) Alors
                        <Traitement N>
                Sinon
                        <Traitement N+1>
                Fin Si
                ⋮
        Fin Si
Fin Si
```

Exercice :
Ecrire un algorithme qui affiche la mention des étudiants admis.

```
Algorithme Mention
Var : M : réel
Début
Lire(M)
Si (M<12) Alors
        Ecrire("Passable")
Sinon
        Si (M<14) Alors
                Ecrire ("Assez Bien")
        Sinon
                Si (M<16) Alors
                        Ecrire ("Bien")
                Sinon
                        Si (M<18) Alors
```

```
                    Ecrire ("Très Bien")
                Sinon
                    Ecrire ("Excellent")
                Fin Si
            Fin Si
        Fin Si
Fin Si
Fin
```

4. Structure conditionnelle à choix multiple

- Définition :

Cette structure est une représentation simplifiée des conditions imbriquées. Elle permet d'exécuter un bloc d'actions parmi plusieurs selon des conditions. Le choix du bloc d'actions se fait suivant la valeur d'un sélecteur.

- Syntaxe :

```
Selon Sélecteur Faire
    Valeur 1 : <Traitement 1>
    ⋮
    Valeur N : <Traitement N>
    Sinon :  <Traitement erreur>
Fin Selon
```

- Remarques :

- Le sélecteur ne peut pas être une chaîne.
- Si la valeur du sélecteur est différente des valeurs proposées (Valeur i), alors le <Traitement Sinon> de selon est exécuté.

Exercice 1:

C'est le même exercice précédent mais en utilisant Selon.

```
Algorithme Mention2
Var :
M : réel
Début
Lire (M)
Selon M Faire
    1..11.99 : Ecrire ("Passable")
    12..13.99 : Ecrire ("Assez Bien")
    14..15.99 : Ecrire ("Bien")
    16..17.99 : Ecrire ("Très Bien")
    18..20 : Ecrire ("Excellent")
    Sinon :  Ecrire ("Non admis")
Fin Selon
Fin
```

Exercice 2:

Ecrire un algorithme qui lit deux réels et un opérateur arithmétique (+, -, ×) et affiche le résultat de l'opération.

```
Algorithme Arithmétique
Var :
Nb1,Nb2 : réel
Op : caractère
Début
Lire (Op)
Selon Op Faire
        '+' : Ecrire (Nb1+Nb2)
        '-' : Ecrire (Nb1-Nb2)
        '×' : Ecrire (Nb1×Nb2)
Fin Selon
Fin
```

II. Les structures répétitives

Les structures répétitives ou itératives permettent d'exécuter plusieurs fois une même série d'instructions en fonction d'une condition donnée.

1. Boucle Pour

Cette boucle itère le même traitement pour une plage de valeurs entières comprises entre une borne inférieure et une borne supérieure.
L'arrêt du traitement de la boucle se réalise lorsqu'on dépasse l'une des bornes.
La boucle Pour est conseillée si le nombre d'itérations à faire est connu à l'avance.

- Syntaxe :

```
Pour i de val_initiale à val_finale (Pas=Val_pas) Faire
        <Traitement>
Fin Pour
```

i : variable de type entier (en général) servant d'un compteur
Val_initiale : Valeur initiale de i
Val_finale : Valeur finale de i
Pas : contient une valeur entière indiquant la valeur e l'incrémentation de i.

Exercice :
Ecrire un algorithme qui permet de saisir les moyennes de N matières et de calculer la moyenne générale d'un étudiant, sachant que les coefficients de toutes matières égales à 1.

```
Algorithme CalculMoyG
Var :N, i : entier
      Moy, MoyG : réel
Début
Ecrire ("Donner le nombre de matières d'un étudiant")
Lire (N)
MoyG ← 0
Pour i de 1 à N Faire
        Ecrire ("Donner la moyenne de la matière n°", i) Lire (Moy)
        MoyG ← MoyG + Moy
Fin Pour
MoyG ← MoyG / N
Ecrire ("La moyenne générale est : ",MoyG)
Fin
```

2. Boucle Répéter

Cette boucle permet de répéter un traitement un ou plusieurs fois quelle que soit la valeur de la condition d'arrêt.
Lorsque la condition d'arrêt est atteinte, on sort de la boucle.
Il faut mettre à jour les variables de la condition d'arrêt à la fin de chaque itération (pour pouvoir reboucler).

- Syntaxe :

Répéter
    <Traitement>
    <Mise à jour de condition d'arrêt>
Jusqu'à (Condition d'arrêt atteinte)

Exercice :
Ecrire un algorithme qui lit un nombre entier inférieur à 100.
On utilise la boucle Répéter qui répète le traitement de saisie jusqu'à le nombre donné soit inférieur à 100.

```
Algorithme Lecture Entier
Var :
X : entier
Début
Répéter
    Ecrire ("Donner une valeur de X inférieure à 100") Lire (X)
Jusqu'à (X<100)
Fin
```

2. Boucle Tant que

Cette boucle permet de répéter le traitement lorsque la condition d'exécution est vérifiée (Donc on a 0 ou plusieurs répétitions).
Tant que la condition est vraie, le traitement est exécuté et on revient pour répéter la même chose. Dès que la condition n'est plus vérifiée, l'exécution s'arrête.

- Syntaxe :

Tant que (Condition d'exécution vérifiée) Faire
    <Traitement>
    <Mise à jour de la condition>
Fin Tant que

Exercice :
Ecrire un algorithme qui oblige l'utilisateur de saisir un entier positif.

```
Algorithme SaisiePositif
Var :
n : entier
Début
n ← -1
tant que (n<0)
    Ecrire ("Donner une valeur positive de n") Lire (n)
Fin Tant que
Ecrire ("L'entier positif n= ", n)
Fin
```

- Remarques générales :

La différence entre la boucle Répéter et la boucle Tant que est que le traitement est exécuté au moins une fois dans le premier cas et peut être exécuté 0 ou plusieurs fois dans le deuxième cas.
La condition d'arrêt de la boucle Répéter est l'inverse de la condition d'exécution de la boucle Tant que.

Exercice récapitulatif :
En utilisant les 3 boucles, écrire les algorithmes de calcul de factoriel de N.
Exemple : N ! = 1*2*3* … *N et 0 ! = 1

```
Algorithme Fact1
Var :
N, F, i : entier
Début
Lire (N)
F ← 1
Pour i de 1 à N Faire
        F← F × i
Fin Pour
Fin
```

```
Algorithme Fact2
Var :
N, F, i : entier
Début
Lire (N)
F ← 1
i ← 1
Répéter
        F← F × i
        i← i +1
Jusqu'à (i>N)
Fin
```

```
Algorithme Fact3
Var :
N, F, i : entier
Début
Lire (N)
F ← 1
i ← 1
Tant que (i<=N) Faire
        F← F × i
        i← i +1
Tant que
Fin
```