Chapitre 02 : Les structures alternatives et répétitives

I. Les structures alternatives

I.1. Introduction

Contrairement au traitement séquentiel, La structure alternative ou conditionnelle permet d'exécuter ou non une série d'instructions selon la valeur d'une condition.

I.2. La structure Si ... Alors ... Sinon ... FinSi

Syntaxe :

Si condition **Alors**
    Instruction(s)1
Sinon
    Instruction(s)2
FinSi

Une condition est une expression logique ou une variable logique évaluée à ***Vrai*** ou ***Faux.***

La condition est évaluée. Si elle est vraie, la série d'instructions1 est

exécutée et l'ensemble d'instructions2 est ignoré, la machine sautera directement à la première instruction située après le ***FinSi***.

De même, au cas où la condition était fausse (avait comme valeur Faux), la machine saute directement à la première ligne située après le ***Sinon*** et exécute l'ensemble d'instructions2.

Exercice :

Ecrire un algorithme qui affiche si un nombre entier saisi au clavier est pair ou impair.

Solution :

On dit qu'un nombre entier n est pair si le reste r de la division entière de n par 2 est égale à 0. Sinon il est impair.

```
Algorithme Parité
// déclaration des variables
Variables n, r : entiers
Début
   Ecrire ("Entrez la valeur de n : ")
   Lire (n)
   // r est le reste de la division entière de n par 2
   r ← n Mod 2
   Si r = 1 Alors
      Ecrire (n, " est impair")
   Sinon
      Ecrire (n, " est pair ")
   FinSi
Fin
```

Remarque : Présentation de l'algorithme ou du programme

Certaines parties de l'algorithme ou du code sont en retrait par rapport à d'autres, c'est ce qu'on appelle ***l'indentation***. Celle-ci est très importante pour la lisibilité de l'algorithme ou du programme. Elle montre rapidement le début et la fin de chaque instruction ***alternative*** ou ***répétitive*** ainsi le début et la fin de l'algorithme ou du programme.

De plus, pour faciliter la compréhension, toutes vos instructions

Prof. Omar El kharki & Mme Jamila Mechbouh / elkharki@gmail.com

doivent être commentées. Un développeur est souvent amené à modifier un code, par conséquent, des commentaires de qualité rendent cette tâche plus facile et plus rapide.

Note 1 :

Les deux blocs 1 et 2 d'instruction sont équivalents.

Bloc 1		**Bloc 2**

Si condition Alors
  Instructions(1)
Sinon
  Instructions(2)
FinSI
	Le bloc 1 est équivalent au bloc 2	Si ***Non*** (condition) Alors
		  Instructions(2)
		Sinon
		  Instructions(1)
		FinSI

Exemple :

Bloc 1		**Bloc 2**

Si note > = 10 Alors
  Ecrire("Note acceptable")
Sinon
  Ecrire("Mauvaise Note")
FinSI
	Le bloc 1 est équivalent au bloc 2	Si note < 10 Alors
		  Ecrire("Mauvaise Note")
		Sinon
		  Ecrire("Note acceptable")
		FinSI

Note 2 : Conditions composées

Certains problèmes exigent de formuler des conditions qui ne peuvent pas être exprimées sous la forme simple exposée ci-dessus.
Par exemple : la condition "x est inclus dans l'intervalle]10, 20 [" est composée de deux conditions simples qui sont : "x est supérieur à 10" et "x est inférieur à 20" reliées par l'opérateur logique ***Et***.

Exemple :

Ecrire un algorithme qui teste si une note saisie au clavier est comprise entre 0 et 20.

Solution :

Prof. Omar El kharki & Mme Jamila Mechbouh / elkharki@gmail.com

```
Algorithme TestNote
Variables Note : réel
          Message : Chaîne
Début
   Ecrire ("Entrer la note : ")
   Lire (Note)
   Si Note >= 0 Et Note <= 20 Alors
      Message ← "La note " & Note & " est correcte"
   Sinon
      Message ← "La note " & Note & " est incorrecte"
   Fin Si
   Écrire (Message)
Fin
```

Exercice :

Ecrire un algorithme qui demande deux nombres m et n à l'utilisateur et l'informe ensuite si le produit est négatif ou positif. On inclut dans l'algorithme le cas où le produit peut être nul.

Solution :

```
Algorithme Signe_Produit
Variables m, n : entiers
Début
   Ecrire(Entrez deux nombres m et n : )"
   Lire(m, n)
   Si m = 0 Ou n = 0 Alors
      Ecrire "Le produit est nul"
   SinonSi (m < 0 Et n < 0) Ou (m > 0 Et n > 0) Alors
      Ecrire("Le produit est positif")
   Sinon
      Ecrire("Le produit est négatif")
   Finsi
Fin
```

Remarque :

- Dans une condition composée employant à la fois des opérateurs ***Et***

Prof. Omar El kharki & Mme Jamila Mechbouh / elkharki@gmail.com

et des opérateurs ***Ou***, la présence de parenthèses possède une influence sur le résultat, tout comme dans le cas d'une expression numérique comportant des opérateurs numériques (exemple : des multiplications et des additions).

- Toute structure de test requérant une condition composée faisant intervenir l'opérateur ***Et*** peut être exprimée de manière équivalente avec un opérateur ***Ou***, et réciproquement (Lois de Morgan).

I.3. La structure Si ... Alors ... FinSi

Cette structure est utilisée si on veut exécuter une instruction seulement si une condition est vraie et ne rien faire si la condition est fausse.

Syntaxe :

Si condition **Alors**
  Instruction(s)
FinSi

Exemple :

Une grande surface accorde à ses clients, une réduction de 2% pour les montants d'achat supérieurs à 1500,00dh.

Ecrire un algorithme permettant de saisir le prix total HT (PTHT) et de calculer le montant TTC (PTTC) en prenant en compte la remise et la TVA=20%.

Solution :

Prof. Omar El kharki & Mme Jamila Mechbouh / elkharki@gmail.com

```
Algorithme      Calcul_PTTC
Constante TVA = 0.2
Variables   PTHT, PTTC : réel
Début
   Ecrire ("Entrez le prix total hors taxe : ")
   Lire (PTHT)
   Si PTHT > 1500 Alors
      PTHT ← PTHT * 0.98
   FinSi
   PTTC ← PTHT + (PTHT * TVA)
   Ecrire ("Le prix TTC est ", PTTC)
Fin
```

I.4. La Structure SI ... Alors ... SinonSI ... Sinon ... FinSi

Syntaxe :

Si condition1 **alors**
    Instructions1
SinonSi condition2 **alors**
    Instructions2
SinonSi condition3 **alors**
    Instructions3
    …..
Sinon
    Instructions
FinSi

L'instruction qui sera exécutée est l'instruction dont la condition est vraie. Si aucune condition n'a la valeur vraie, c'est l'instruction qui suit le ***Sinon*** qui sera exécutée.

Exemple :

Ecrire un algorithme qui permet d'afficher le maximum parmi deux nombres saisis au clavier.

Solution :

```
Algorithme Max
Variables A, B : réel
Début
   Ecrire ("Entrez la valeur de A : ")
   Lire (A)
   Ecrire ("Entrez la valeur de B : ")
   Lire (B)
   Si A > B Alors
      Ecrire ("Le maximum est ", A)
   SinonSI A = B alors
      Ecrire ("égalité")
   Sinon
      Ecrire ("Le maximum est ", B)
   FinSi
Fin
```

Remarque :

Une structure conditionnelle peut contenir à son tour une autre structure conditionnelle, c'est-à-dire l'une incluse dans l'autre. Dans ce cas, les instructions à exécuter après **Alors** et/ou **Sinon** sont, à leur tour, des instructions **Si...Alors...Sinon**.

Exemple :

Ecrire un algorithme permettant de calculer le montant des allocations familiales sachant que le montant dépend du nombre d'enfants :
- Si nombre d'enfants est inférieur ou égale à trois alors les allocations sont de 150 dh par enfant.
- Si le nombre d'enfants est strictement supérieur à trois et inférieur ou égale à six alors les allocations sont de :
    - 150 dh par enfant pour les trois premiers enfants
    - 38 dh par enfant pour les suivants
- Si le nombre d'enfants est strictement supérieur à six alors les allocations sont de :
    - 150dh par enfant pour les trois premiers enfants
    - 38 dh par enfant pour les 3 suivants

Prof. Omar El kharki & Mme Jamila Mechbouh / elkharki@gmail.com

- 0 dh par enfant pour les suivants.

Solution :

```
Algorithme      Allocation_familiale
Variables   NE, M : entier
/* NE : le nombre d'enfants
M : montant des allocations familiales */
Début
   Ecrire ("Entrez le nombre d'enfants : ")
   Lire (NE)
   Si NE <= 3 Alors
      M ← NE * 150
   Sinon
      Si NE <=6 alors
         M ← 450 + (NE – 3) * 38
      Sinon
         M ← 564
      FinSi
   FinSi
   Ecrire ("Le montant d'allocation familiale est ", M)
Fin
```

I.5. Structure à choix multiples

Cette structure conditionnelle permet de choisir le traitement à effectuer en fonction de la valeur ou de l'intervalle de valeurs d'une variable ou d'une expression.

Syntaxe :

```
Selon sélecteur faire
   Valeur1 : action(s)1
   Valeur2 : action(s)2
   Valeur3 : action(s)3
   ....
   Valeurn : action(s)n
Sinon
   action(s)
```


Prof. Omar El kharki & Mme Jamila Mechbouh / elkharki@gmail.com

FinSelon

Lorsque l'ordinateur rencontre cette instruction, il vérifie la valeur de la variable de sélection (sélecteur) et il la compare aux différentes valeurs.

Les valeurs sont évaluées dans l'ordre, les unes après les autres, et dès qu'une de celle-ci est vérifiée l'action associée est exécutée. On peut utiliser une instruction **Sinon** (facultative), dont l'action sera exécutée si aucune des valeurs évaluées n'a été remplie.

Exemple :

Ecrire un algorithme permettant d'afficher le mois en toute lettre selon son numéro saisi au clavier.

Solution :

Prof. Omar El kharki & Mme Jamila Mechbouh / elkharki@gmail.com

```
Algorithme Mois
Variables   N : Entier
Début
   Ecrire ("Donner le numéro du mois: ")
   Lire (N)
   Selon N faire
      1 : Ecrire ("Janvier")
      2 : Ecrire ("Février")
      3 : Ecrire ("Mars")
      4 : Ecrire ("Avril")
      5 : Ecrire ("Mai ")
      6 : Ecrire ("Juin")
      7 : Ecrire ("Juillet")
      8 : Ecrire ("Août")
      9 : Ecrire ("Septembre")
      10 : Ecrire ("Octobre")
      11 : Ecrire ("Novembre")
      12 : Ecrire ("Décembre")
   Sinon
      Ecrire ("Le numéro saisi est incorrecte: ")
   FinSelon
Fin
```

II. Les structures répétitives

II.1. Introduction

Prenons le cas d'une saisie au clavier, par exemple, on pose une question à laquelle l'utilisateur doit répondre par O (Oui) ou N (Non).

L'utilisateur risque de taper autre chose (une autre lettre), le programme peut soit planter par une erreur d'exécution soit se dérouler normalement jusqu'au bout, mais en produisant des résultats fantaisistes.

Pour remédier à ce problème, on peut mettre en place un contrôle de saisie pour vérifier que les données entrées au clavier correspondent

bien à celles attendues par l'algorithme.

On pourrait essayer avec un SI.

```
Algorithme  contrôle_de_saisie
Variable   Rep : caractère
Début
   Ecrire "Voulez vous une copie de ce cours ? (O/N)"
   Lire (Rep)
   Si Rep ≠ 'O' ET Rep ≠ 'N' Alors
      Ecrire ("Erreur de saisie. Recommencez")
      Lire (Rep)
   FinSi
Fin
```

L'algorithme ci-dessus résout le problème si on se trompe qu'une seule fois, et on fait entrer une valeur correcte à la deuxième demande. Sinon en cas de deuxième erreur, il faudrait rajouter un SI. Et ainsi de suite, on peut rajouter des centaines de SI.

La solution à ce problème consiste à utiliser une structure répétitive.

Une structure répétitive, encore appelée boucle, est utilisée quand une instruction ou une liste d'instructions, doit être répétée plusieurs fois. La répétition est soumise à une condition.

II.2. La boucle TantQue ... Faire

La boucle TantQue permet de répéter un traitement tant que la condition est vraie.

Syntaxe :

TantQue condition **faire**
    Instruction(s)
FinTantQue

L'exécution de la boucle dépend de la valeur de la condition. Si elle est vraie, le programme exécute les instructions qui suivent, jusqu'à ce qu'il rencontre la ligne FinTantQue. Il retourne ensuite sur la ligne du

Prof. Omar El kharki & Mme Jamila Mechbouh / elkharki@gmail.com

TantQue, procède au même examen, et ainsi de suite.

La boucle ne s'arrête que lorsque la condition prend la valeur fausse, et dans ce cas le programme poursuit son exécution après FinTantQue.

Exemple :

```
Algorithme  contrôle_de_saisie
Variable   Rep : caractère
Début
   Ecrire ("Voulez vous une copie de ce cours ? (O/N)" )
   Lire (Rep)
   TantQue Rep ≠ 'O' ET Rep ≠ 'N' faire
      Ecrire (" Erreur de saisie")
      Ecrire ("Voulez vous une copie de ce cours ? (O/N)" )
      Lire (Rep)
   FinTantQue
Fin
```

Remarques :

- Etant donnée que la condition est évaluée avant la mise en oeuvre des instructions, ***ce qui est une sécurité***, il est possible que celles-ci ne soient jamais exécutées.
- Si une structure TantQue dans laquelle la condition ne devient jamais fausse. Le programme tourne dans une boucle infinie et n'en sort plus.

Exemple : Boucle infinie

Dans l'exemple ci-dessous nous avons une boucle infinie, l'ordinateur ne s'arrêtera jamais d'afficher le message **Bonjour** car la variable **I** qui est testée dans la condition n'est jamais incrémentée :

```
I ← 1
Tant que I < = 10 Faire
   Ecrire(" bonjour")
FinTantQue
```


Prof. Omar El kharki & Mme Jamila Mechbouh / elkharki@gmail.com

Exercice :

Ecrire un algorithme qui calcule S = 1 + 2 + 3 + 4 + 5 + ... + N.

Solution :

```
Algorithme  Somme
Variables   S, I, N : Entier
Début
   Ecrire ("Entrer la valeur de N : ")
   Lire (N)
   S ← 0
   I ← 1
   TantQue  I <= N faire
      S ← S + I
      I  ← I +1
   FinTantQue
   Ecrire("La somme des ", N, " premiers entiers est : ",  S)
Fin
```

II.3. La boucle Pour ... Faire

La boucle **Pour … Faire** permet de répéter une liste d'instructions un nombre connu de fois.

Syntaxe :

Pour compteur ← valeur_initiale **jusqu'à** valeur_finale **Faire**
    Instruction(s)
FinPour

La variable compteur est de type entier. Elle est initialisée à la valeur initiale. Le compteur augmente sa valeur de **un** (1) **automatiquement** à chaque tour de boucle jusqu'à la valeur finale.

Pour chaque valeur prise par la variable compteur, la liste des instructions est exécutée.

Lorsque la variable compteur vaut la valeur finale, le traitement est exécuté une dernière fois puis le programme sort de la boucle.

Prof. Omar El kharki & Mme Jamila Mechbouh / elkharki@gmail.com

Exemple :

Les instructions suivantes :

```
Pour k ← 0 jusqu'à 10 faire
   Ecrire( " 7 * ", k, " = ", 7*k)
   Ecrire("\")
FinPour
```

affichent à l'écran la table de multiplication de 7.

7 * 0 = 0	7 * 4 = 28	7 * 8 = 56
7 * 1 = 7	7 * 5 = 35	7 * 9 = 63
7 * 2 = 14	7 * 6 = 42	7 * 10 = 70
7 * 3 = 21	7 * 7 = 49	

Exercice :

Ecrire un algorithme permettant de calculer la somme des dix premiers nombres entiers.

Solution :

```
Algorithme  Somme
Variables   S, I : Entier
Début
   S ← 0
   Pour I ← 1 jusqu'à 10 Faire
      S ← S + I
   FinPour
   Ecrire("La somme des dix premiers entiers est : ", S)
Fin
```

Remarques :

- Par défaut la variable compteur est incrémentée de 1 à chaque tour de boucle
- Pour modifier la valeur d'incrémentation, il suffit de rajouter le mot **Pas** (en anglais **step**) et la valeur de ce pas à la boucle **Pour**.

La syntaxe générale de la structure Pour est :

Prof. Omar El kharki & Mme Jamila Mechbouh / elkharki@gmail.com

Pour compteur ← VI **jusqu'à** VF **Pas** ValeurPas **Faire**
Instruction(s)
FinPour

VI c'est la valeur initiale
VF c'est la valeur finale.

Exercice :

Ecrire un algorithme qui permet de saisir un nombre entier et qui calcule la somme des entiers pairs jusqu'à ce nombre. Par exemple, si l'on saisi 10, le programme doit calculer :
0 + 2 + 4 + 6 + 8 + 10 = 30

Solution :

```
Algorithme  Somme
Variables   S, I, N : Entier
Début
   Ecrire ("Entrer la valeur de N : ")
   Lire (N)
   S ← 0
   Pour I ← 0 jusqu'à N Pas 2 Faire
      S ← S + I
   FinPour
   Ecrire("La somme des nombres pairs est : ", S)
Fin
```

II.4. La boucle Répéter … Jusqu'à

Cette boucle sert à répéter une instruction jusqu'à ce qu'une condition soit vraie.

Remarque :

Cette boucle ne s'utilise en général que pour des menus, elle est dangereuse car il n'y a pas de vérification de la condition avant d'y entrer !

Syntaxe :

Répéter

Prof. Omar El kharki & Mme Jamila Mechbouh / elkharki@gmail.com

Instruction(s)
Jusqu'à condition

La liste d'instructions est exécutée, puis la condition est évaluée. Si elle est fausse, le corps de la boucle est exécuté à nouveau puis la condition est réévaluée et si elle a la valeur vrai, le programme sort de la boucle et exécute l'instruction qui suit **Jusqu'à**.

Exemple :

En utilisant la boucle Répéter … Jusqu'à, écrire un algorithme qui calcule la somme des N premiers nombres entiers. On suppose que N est strictement positif.

Par exemple, si N = 6, le programme doit calculer : 1+2+3+4+5+6 =21

Solution :

```
Algorithme  Somme
Variables   S, I, N : Entier
Début
   Ecrire ("Entrer une valeur strictement positif : ")
   Lire (N)
   S ← 0
   I ← 1
   Répéter
      S ← S + I
      I ← I +1
   Jusqu'à  I >= N
   Ecrire("La somme des ",N," premiers entiers est : ", S)
Fin
```

Remarques :

- Les boucles « Répéter » et « TantQue » sont utilisées lorsqu'on ne sait pas au départ combien de fois il faudra exécuter ces boucles.
- A la différence de la boucle « TantQue », la boucle « Répéter » est exécutée au moins une fois.
- La condition d'arrêt de la boucle « Répéter » est la négation de

la condition de poursuite de la boucle « TantQue ».

- On utilise la boucle « Pour » quand l'on connaît le nombre d'itérations à l'avance.

Prof. Omar El kharki & Mme Jamila Mechbouh / elkharki@gmail.com