

Chapitre 3 : **LES STRUCTURES ITERATIVES**

Introduction

Les structures itératives sont aussi appelées structures répétitives ou boucles. Elles permettent de répéter l'exécution d'un ou de plusieurs traitements. Les boucles sont classées en 3 catégories et chacune d'elles a sa syntaxe et son contexte d'utilisation. Les boucles sont :

- la boucle **Pour**
- la boucle **Repeter**
- la boucle **Tantque**

I. La Boucle Pour

1. Contexte d'utilisation

La boucle Pour permet de répéter l'exécution d'un ou de plusieurs traitements. Elle peut être utilisée que si le nombre d'itérations est connu c'est-à-dire le nombre de fois que les actions de la boucle seront exécutées.

2. Syntaxe

Pour <indice> allant de <valeur initiale> à <valeur finale> **Par Pas de** <valeur pas> **Faire**
 <Action 1>
 <Action n>

FinPour

Remarque :

L'indice de parcours de la boucle doit toujours être une variable de type **entier** ou **caractere**. L'expression **Par Pas de** <valeur pas> est facultative si le pas d'incrément est 1. Si le pas d'incrément est différent de 1 alors c'est une obligation de mettre l'expression spécifiant la valeur du pas d'incrément ou de décrétement. L'indice peut être utilisé pour des parcours croissants ou décroissants. Tout dépend de la relation d'ordre qui existe entre la valeur initiale et la valeur finale.

Exercice d'application 1

Ecrire un programme qui demande à l'utilisateur de saisir une valeur entière positive N. le programme affiche les nombres pairs compris entre 1 et N ainsi que la somme de ces nombres pairs.

Exercice d'application 2

Ecrire un programme qui demande à l'utilisateur de saisir un nombre entier positif. Le programme détermine et affiche la table de multiplication de l'entier saisi. La limite de la table est fixée à 12.

II. La Boucle Repeter

1. Contexte d'utilisation

La boucle Repeter est une boucle de contrôle et elle permet aussi de répéter l'exécution d'un ou de plusieurs traitements. Sa particularité est que les actions sont exécutées au moins une fois.

2. Syntaxe

Repeter

<Action 1>

<Action n>

Jusqua(condition(s))

Remarque :

Parmi les actions de la boucle Repeter, il en faut une qui permet la condition de pouvoir évoluer.

Exercice d'application

Ecrire un programme qui permet de contrôler la saisie d'une valeur entière positive. Le programme détermine et affiche le factoriel de la valeur saisie.

III. La Boucle Tantque

1. Contexte d'utilisation

La boucle Tantque permet de contrôler des saisis mais aussi de répéter l'exécution d'un ou de plusieurs traitements. Sa particularité est que ses actions ne sont exécutées que si la condition est vérifiée.

2. Syntaxe

Tantque(condition(s)) Faire

<Action 1>

<Action n>

FinTantque

Remarque :

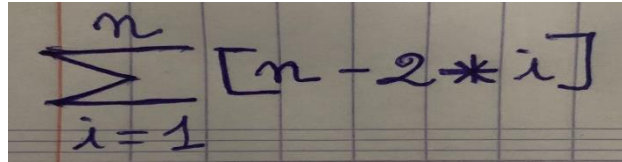
Pour éviter une boucle infinie, il faut toujours que nous ayons au moins une action de la boucle qui permet à la condition de pouvoir évoluer. La boucle Tantque peut être utilisée dans tous les cas.

Exercice d'application 1

*Ecrire un programme qui demande à l'utilisateur de saisir un nombre entier positif. Le programme détermine et affiche si le nombre saisi est un nombre premier ou pas. **Un nombre est premier s'il n'a que deux diviseurs qui sont 1 et le nombre lui-même.***

Exercice d'application 2

Ecrire un programme qui demande à l'utilisateur de saisir une valeur entière positive n. le programme calcule et affiche le résultat de l'expression suivante :



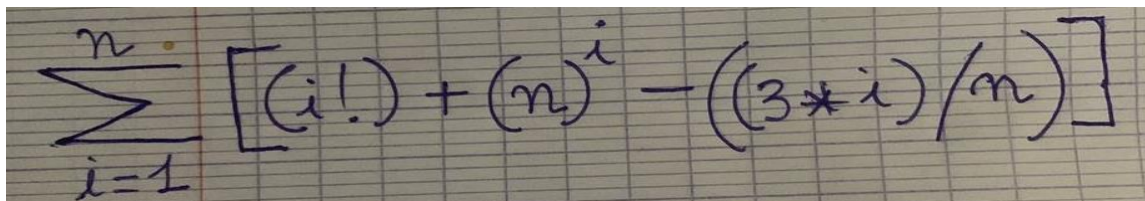
$$\sum_{i=1}^n [n - 2 * i]$$

IV. Imbrication des structures répétitives

Dans un programme, il est fréquent qu'une boucle soit définie à l'intérieur d'une autre boucle alors on parle d'imbrication de boucles. Dans ce cas, la boucle la plus interne est fermée avant celle la plus externe.

Exercice d'application 1

Ecrire un programme qui permet de contrôler la saisie d'une valeur entière positive n . Le programme détermine et affiche le résultat de l'expression suivante :



$$\sum_{i=1}^n [(i!) + (n)^i - ((3 * i) / n)]$$

Exercice d'application 2

Ecrire un programme qui permet de contrôler la saisie d'une valeur entière positive N . Le programme détermine et affiche les tables de multiplication des nombres pairs compris entre 1 et N . La limite de chaque table est fixée à 12.

Exercice d'application 3

Ecrire un programme qui permet de saisir une série de N valeurs entières positives. Le programme détermine et affiche la moyenne des nombres pairs ainsi que le pourcentage de présence des nombres impairs divisibles par 5.

Moyenne des nombres pairs = Somme des nombres pairs / Nombre de nombres pairs

*% de présence des nombres impairs = (Nombre de nombres impairs * 100) / N*