ATELIER: FOR – WHILE – DO ..WHILE

OBJECTIFS :

Manipuler les différentes structures répétitives disponibles en langage C et distinguer le cas d'utilisation de chaque structure.

Exercice 1:

```
Algorithme exercice1
Variables    i, k, N, P : entier
Début
    lire(N)
    i ←N
    P ←1
    Pour i de 1 à N faire
        k ← i
        P ← P*k
        k ← k-1
    finfaire
    écrire(P)
Fin
```

1. Donnez la trace d'exécution de cet algorithme pour **n = 5**
2. Que fait cet algorithme. Traduisez-le en C.

Exercice 2 :

Ecrivez un programme qui lit N nombres entiers au clavier et qui affiche leur somme, leur produit et leur moyenne. Choisissez un type approprié pour les valeurs à afficher. Le nombre N est à entrer au clavier. Résolvez ce problème,

a) En utilisant ***while***,
b) En utilisant ***do - while***,
c) En utilisant ***for***.
d) Laquelle des trois variantes est la plus naturelle pour ce problème?

Exercice 3 :

Répétez l'introduction du nombre N jusqu'à ce que N ait une valeur entre 1 et 15.
Quelle structure répétitive utilisez-vous? Pourquoi?

Exercice 4 :

Ecrire un programme qui demande un nombre de départ, et qui ensuite écrit la table de multiplication de ce nombre, présentée comme suit (cas où l'utilisateur entre le nombre 7) :

Table de 7 : 7 x 1 = 7
7 x 2 = 14
…
7 x 10 = 70

Exercice 5:

Calculez la factorielle **N! = 1*2*3*…*(N-1)*N** d'un entier naturel N. (0!=1).

a) Utilisez **while**,
b) Utilisez **for**.

Exercice 6:

Écrire un programme qui lit un réel **x** et un entier positif **p** et affiche **x** puissance **p**.

Exercice 7:

Écrire un programme qui permet de saisir un nombre entier n et d'afficher s'il est premier ou non. Un nombre premier est divisible uniquement par 1 et par lui-même.

Exercice 8:

Écrire un programme C qui détermine si un entier N est parfait ou non. Un entier est dit parfait s'il est égal à la somme de ses diviseurs stricts (Exemple: 6=3+2+1).

Exercice 9:

Etant donnée une classe composée de **N** élèves. Chaque élève a passé trois examens: *un test, un examen de travaux pratiques et un examen final*. Les coefficients respectifs sont ***C1***, ***C2*** et ***C3***.

Ecrire un programme permettant de lire les trois notes de chaque élève, calculer sa moyenne, d'écrire sa moyenne suivie d'une appréciation littérale :

- ***ECHEC*** si la moyenne est inférieure à 10,
- ***PASSABLE*** si elle est supérieure ou égale à 10 et inférieure à 12,
- ***ASSEZ_BIEN*** si elle est supérieure ou égale à 12 et inférieure à 14,
- ***BIEN*** si elle est comprise entre 14 et 16 et
- ***TRES_BIEN*** si elle est supérieure à 16.

Le programme doit bien entendu lire au préalable le nombre d'élèves et les coefficients. Il doit aussi vérifier la validité des données et refuser toute donnée non valide.

Annexe : LES STRUCTURES REPETITIVES

1. La structure while en C

```
while ( <expression> )
   <bloc d'instructions>
```

- Tant que l'<expression> fournit une valeur différente de zéro, le <bloc d'instructions> est exécuté.
- Si l'<expression> fournit la valeur zéro, l'exécution continue avec l'instruction qui suit le bloc d'instructions.
- Le <bloc d'instructions> est exécuté zéro ou plusieurs fois.
- La partie <expression> peut désigner une variable d'un type numérique ou une expression fournissant un résultat numérique.
- La partie <bloc d'instructions> peut désigner un bloc d'instructions compris entre accolades, ou une seule instruction terminée par un point-virgule.

2. La structure do - while en C

```
do
   <bloc d'instructions>
while ( <expression> );
```

La structure **do - while** est semblable à la structure **while**, avec la différence suivante :

- **while** évalue la condition ***avant*** d'exécuter le bloc d'instructions,
- **do - while** évalue la condition ***après*** avoir exécuté le bloc d'instructions. Ainsi le bloc d'instructions est exécuté au moins une fois.

3. La structure for en C

```
for ( <expr1> ; <expr2> ; <expr3> )
   <bloc d'instructions>
```

- **<expr1>** est évaluée une fois avant le passage de la boucle. Elle est utilisée pour initialiser les données de la boucle.
- **<expr2>** est évaluée avant chaque passage de la boucle. Elle est utilisée pour décider si la boucle est répétée ou non.
- **<expr3>** est évaluée à la fin de chaque passage de la boucle. Elle est utilisée pour réinitialiser les données de la boucle.
- La structure for en C est équivalent à :

```
<expr1>;
while ( <expr2> )
    {
       <bloc d'instructions>
       <expr3>;
     }
```