TD n°2 : Itération - Les boucles tant que, Répéter et Pour

Exercice 1 : Complétez les algorithmes suivants afin qu'ils effectuent le traitement itératif décrit. Justifiez clairement votre choix de boucle.

a) Algorithme Factorielle

```
        //Cet algorithme calcule la factorielle d'un nombre fourni par l'utilisateur
var val, fact: entier
début
        ecrire( 'Entrez un nombre entier ')
        Lire(val)
        ...
        //affichage du résultat
        ecrire( ' La factorielle de ', val, ' est ', fact)
fin
```

b) Algorithme Bonjour

```
//Cet algorithme simule un bonjour poli et personnalisé}
        variables réponse : entier
                début
                // Demander à l'utilisateur de taper 1 si elle est une étudiante,
                //et 2 si il est un étudiant, et réitérer la demande tant que les consignes ne sont pas suivies
                        ...
                //affichage du résultat}
                si réponse = 1
                        alors ecrire ('Bonjour Mademoiselle ')
                        sinon ecrire (' Bonjour Monsieur ')
                finsi
        fin
```

c) Algorithme AffichePhrase

```
 //Cet algorithme saisit et affiche au fur et à mesure les caractères fournis par l'utilisateur jusqu'au //premier
point rencontré. En fin de traitement, il affiche le nombre de caractères saisis.
        constante (STOP : caractère) = '.'
        var unCar : caractère
        début
                ...
        //présentation des résultats
        ecrire ('Le nombre de caractère saisis est ' , cpt)
        fin
```

Attention : le point final est la valeur d'arrêt et ne doit donc pas être affiché, ni comptabilisé.

Exercice 2 : Modifiez l'algorithme AffichePhrase pour :

a) qu'il n'affiche pas les blancs (espaces). L'algorithme devra afficher, en fin de traitement, le nombre de caractères blancs éliminés.

b) et qu'il saisisse au maximum 100 caractères (pensez aux constantes).

L'algorithme devra également indiquer, en fin de traitement, la raison de la sortie de boucle : a-t-on saisi la valeur d'arrêt ou bien a-t-on dépassé le maximum de valeurs autorisé ? Justifiez la boucle choisie.

Exercice 3: On veut trouver la plus grande valeur dans une suite de valeurs saisies par l'utilisateur.

1) Ecrire un algorithme qui
- demande à l'utilisateur le nombre nbVal de valeurs entières à saisir,
- saisit les nbVal valeurs,
- affiche la plus grande valeur ainsi que son rang dans la suite.

Par exemple, si la suite de valeurs est 45 –6 35 11 –345 462 –24
alors le programme affichera : La plus grande valeur est 462, c'est la 6 ème valeur de la suite.
Justifiez la boucle choisie.

Exercice 4
La bibliothèque « **stdlib.h** » offre une fonction permettant d'utiliser le générateur pseudo-aléatoire : « **int rand()** » qui retourne une valeur entière. En langage algorithmique, on notera cette fonction « **random()** ».

En langage C, pour utiliser cette fonction sans avoir toujours la même liste de nombres "aléatoires", il faut initialiser le générateur avec l'instruction « **srand(time(NULL));** » que vous devez ajouter au tout d\ébut de la fonction main. Cette instruction utilise la bibliothèque « time.h ».

1. Ecrire un algorithme qui tire aléatoirement *N* entiers compris entre *MIN* et *MAX* (inclus) et les affiche. Les valeurs associées à *N*, *MIN* et *MAX* seront définies comme des constantes. Justifiez le type de boucle que vous avez choisi.

2. Ecrire un algorithme qui tire des entiers compris entre *MIN* et *MAX* jusqu'à ce que la valeur *BUT* soit tirée (la valeur associée à *BUT* sera définie comme une constante). Votre algorithme devra afficher les valeurs tirées et le nombre de tirages effectués avant le tirage de la valeur *BUT*. Justifiez le type de boucle que vous avez choisi.

3. Nous souhaitons maintenant limiter le nombre de tirages effectués. Le nombre maximum de tirages possibles est représenté par la valeur *NBCOUPS* définie comme une constante. Modifiez votre algorithme pour qu'au maximum *NBCOUPS* tirages soient effectués et que s'affiche :

- *GAGNE* si la valeur *BUT* a été trouvée en *NBCOUPS* tirages au maximum ;
- *PERDU* si la valeur *BUT* n'a pas été trouvée après *NBCOUPS* tirages.

Justifiez le type de boucle que vous avez choisi.

Exercice 5 : On veut saisir une valeur dans un intervalle donné. Pour cela :

1. Ecrivez un algorithme qui saisit une valeur de référence val_début, et puis saisit une suite de nombres et s'arrête dès que l'on saisit une valeur qui soit inférieure à val_debut. A l'issue du traitement, l'algorithme doit afficher la valeur qui a provoqué l'arrêt, ainsi que son rang. Justifiez la boucle choisie.

2. Modifiez votre algorithme pour qu'il saisisse deux entiers référence val_début et val_fin délimitant un intervalle donné (vous supposerez que val_début est plus petit que val_fin), et qu'il arrête la boucle de saisie dès que l'on saisit une valeur qui soit dans cet intervalle. A l'issue du traitement, l'algorithme doit afficher la valeur qui a provoqué l'arrêt ainsi que son rang.

3. On ne suppose plus que val_début est plus petit que val_fin. Améliorez votre algorithme pour permettre à l'utilisateur de ressaisir les deux valeurs de référence tant que la première n'est pas inférieure à la deuxième.

Exercice 6 : Multiplication égyptienne

Soient X et Y, deux nombres entiers. Ecrire un algorithme qui doit calculer le produit des deux en n'utilisant que l'addition**,** la soustraction et la division par 2. On remarque que:

Si X est pair, XY =(X/2) (2Y).

Si X est impair, XY = (X-l)Y +Y.

D'où par exemple :
15x 53 = 14x53+53

= 7*106+53

= 6*106+106+53

= 3*212+159

= 2*212+212 +159= 2*212+371

= 1*424+371 = **795**

Remarque: Il est souhaitable que X soit inférieur à Y. Le programme agira en conséquence.

Exercice 7
Écrire un algorithme *pyramide_Descendante* permettant d'afficher le triangle suivant, le nombre de lignes étant donné par l'utilisateur.

```
0
01
012
0123
01234
012345
0123456
01234567
012345678
```

Exercice 8
Écrire un algorithme *pyramid_Ascendente* permettant d'afficher le triangle suivant, le nombre de lignes étant donné par l'utilisateur.

```
        0
       01
      012
     0123
    01234
   012345
  0123456
 01234567
012345678
```

Exercice 9
Écrire un algorithme pyramide permettant d'afficher le triangle suivant, le nombre de lignes étant donné par l'utilisateur.

```
         0
        012
       01234
      0123456
     012345678
    01234567890
0123456789012
```