# Corrigé série 5.3 : Les boucles

## Exercice 1 :

Expliquer ce que font les boucles suivantes :

a)

```
 n = 1 ;
while (n < 40)  {
  printf("valeur %d\n", n);
  n *= 2;
}
```

```
Valeur 1
Valeur 2
Valeur 4
Valeur 8
Valeur 16
Valeur 32
```

b)

```
int  compte = 0;

do
{
   printf("%d ", compte);
   ++compte;
}
while (compte % 8 != 0);
```

```
0
1
2
3
4
5
6
7
```

c)

```
while (n == 0);
```

Cette boucle ne fait rien ';' Soit elle n'est jamais exécutée, si n est différent de 0, soit elle est infinie (le programme se bloque là). (utilité éventuelle avec le mot-clef volatile)

d)

```
for (x = 0; x < 1000; ++x)
   printf("%d ", x);
```

```
0 1 2 3 4 5 6 7 8 9
10 11 12 13 14 15 16
17
         ...

994 995 996 997 998
999
```

e)

```
for (argent = 100; argent < 10000;
    argent *= 2.0)
printf("Fortune = %g\n", argent);
```

```
Fortune = 100
Fortune = 200
Fortune = 400
Fortune = 800
Fortune = 1600
Fortune = 3200
Fortune = 6400
```

**Exercice 2**

a) 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. Utilisez la boucle la mieux appropriée.

```c
#include <stdio.h>
int main()
{
 int n;
 int nombre;   /* nombre courant    */
 int somme;    /* la somme   des nombres entrés */
 int produit;  /* le produit des nombres entrés */

 printf("Nombre de données : ");
 scanf("%d", &n);

 somme=0;
 produit=1;
 for (i=1 ; i<=n ; i++)
 {
     printf("%d. nombre : ", i);
     scanf("%d", &nombre);
     somme   += nombre;
     produit *= nombre;
 }
 printf("Somme   des %d nombres est %d\n",   n, somme);
 printf("Produit des %d nombres est %d\n",   n, produit);
 printf("Moyenne des %d nombres est %.4f\n", n, (float)somme/n);
 return 0 ;
}
```

b) Répétez l'introduction du nombre N jusqu'à ce que N ait une valeur entre 1 et 15.

Remplacer les lignes

```c
   printf("Nombre de données : ");
   scanf("%d", &n);
```

par :

```c
do
{
  printf("%d. nombre (min:1 max:15): ", i);
  statut=scanf("%d", &nombreLu);       // statut vaut 0 en cas d'erreur
  fflush(stdin);         // instruction vidant le tampon mémoire du clavier
}
while( (statut!=1) || (nombreLu<1) || (nombreLu>15)   ) ;
```

Quelle structure répétitive utilisez-vous ? Pourquoi ?

`while` ou `do..while`, car on ne sait pas le nombre de fois que sera effectué la lecture

`do..while`, car on doit passer au moins une fois dans cette boucle

Note: la condition du while utilise ici l'évaluation conditionnelle de l'opérateur || en C (*short-circuit*)

**Exercice 3**

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

```c
#include <stdio.h>
#include <stdlib.h>

/// EX 5.3.3 Calcul de la factorielle de N, N!
int main()
{
    int  N;
    double factorielle; // Type double a cause de la grandeur du résultat!

    do
    {
        printf("Entrez un entier naturel N: ");
        scanf("%d", &N);
    }
    while (N<0);

    factorielle = 1.0 ; // Car si N=0, 0! = 1.0 par definition
    for (int i=1 ; i<=N ; i++)
    {
        factorielle*=i;
    }
    printf ("%d! = %.0f\n", N, factorielle);

    system("pause");
    return 0;
}
```

Quelle structure répétitive utilisez-vous ? Pourquoi ?

`for`, car on sait combien de fois sera exécuté la boucle.

## **Exercice 4** Avancé facultatif

a) Calculez la racine carrée *x* d'un nombre réel positif A par approximations successives en utilisant la relation de récurrence suivante :

$$x_{j+1} = (x_j + A/x_j) / 2 \quad \text{avec :} \quad x_1 = A$$

L'utilisateur doit entrer la précision ε du calcul souhaitée. Le programme doit s'arrêter lorsque $| X_{J+1} - X_J | < \varepsilon$

b) Assurez-vous lors de l'introduction des données que la valeur pour A et $\varepsilon$ sont des réels positifs.

c) Affichez lors du calcul toutes les approximations calculées :

```
La 1ère approximation de la racine carrée de ... est ...
La 2e   approximation de la racine carrée de ... est ...
La 3e   approximation de la racine carrée de ... est ...
  . . .
```

```c
#include <stdio.h>
#include <math.h>
int main()
{
   double a;       /* donnée */
   double x_cur;   /* racine carrée courante */
   double x_prev;  /* racine carrée précédente */
   double epsilon; /* précision voulue de l'approximation     */
   int j;
   do {
      printf("Entrer le réel positif a : ");
      scanf("%lf", &a);
   } while(a<0);

   do {
      printf("Précision de l'approximation [0...1E-1]: ? ");
      scanf("%lf", &epsilon);
   }
   while(epsilon<=0 || epsilon>=1E-1);

   j=1;
   x_cur=a;
   do {
      x_prev = x_cur;
      x_cur  = (x_prev + a/x_prev) / 2;
      printf("La %2d%s approximation de la racine carrée \\
              de %.2f est %.2f [epsilon:%lf]\n",         \\
              j, (j==1)?"ere":"e",
              a, x_cur, fabs(x_cur-x_prev) );
      j++;
    } while(  fabs(x_cur-x_prev)> epsilon  );

    return 0;
}
```

**Exercice 5**

Affichez un triangle isocèle formé d'étoiles de N lignes (N est fourni au clavier) :

```
Nombre de lignes : 4

       *
      ***
     *****
    *******
```

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int lineNumber;
    do{
        printf("Nombre de lignes souhaite (min:1, max:20): ");
        scanf("%d", &lineNumber);
    }
    while (lineNumber<1 || lineNumber>20);

    Voir ci-dessous

    system("pause");
    return 0;
}
```

**Méthode 1**

| **Variable** | **starNumber** | **spaceNumber** |
|---|---|---|
| **Valeur initiale** | 1 | lineNumber - 1 |
| **Incrément** | +2 | -1 |

```
    int spaceNumber = lineNumber-1;
    int starNumber  = 1;
    for (int line=1 ; line<=lineNumber ; line++)
    {
        for (int column=1 ; column<=spaceNumber ; column++)
            putchar('.');
            for (int column=1 ; column<=starNumber  ; column++)
              putchar('*');
        putchar('\n');

        spaceNumber -= 1;
        starNumber  += 2;
    }
```

**Méthode 2**

lineNumber = 4

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | | | | * | | | |
| 2 | | | * | * | * | | |
| 3 | | * | * | * | * | * | |
| 4 | * | * | * | * | * | * | * |

| StarCount | spaceNumber | line |
|---|---|---|
| 1 | 3 | 1 |
| 3 | 2 | 2 |
| 5 | 1 | 3 |
| 7 | 0 | 4 |
| 2*line-1 | lineNumber-line | |

Formule :

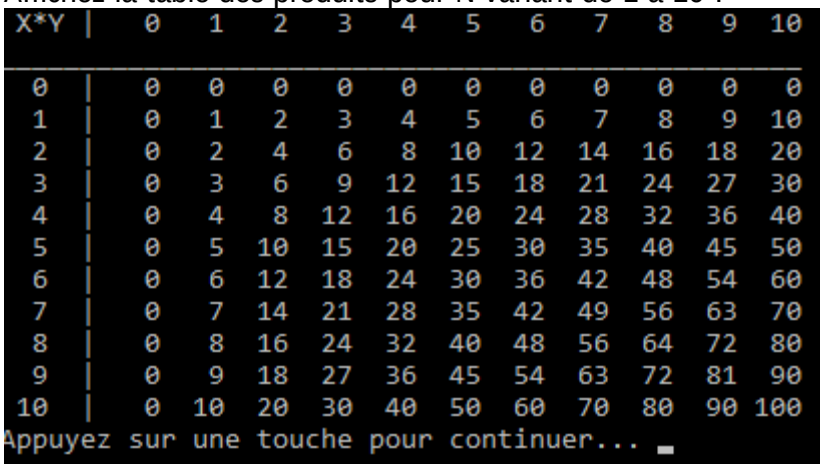
```
    // METHODE 1: Calcul direct du nomre d'espace et du nombe
                  d'étoiles
    for (int line=1; line<=lineNumber ; line++)
    {
        for (int column=1 ; column<=lineNumber-line ; column++)
            putchar('.');
        for (int column=1 ; column<=2*line-1        ; column++)
            putchar('*');
        putchar('\n');
 }
```

**Méthode 3**

```
    // METHODE 3: Moins performante, mais pouvant être utile dans
    //            d'autres contextes.
    // Parcours tous les points possibles et teste pour
    // chacun s'il faut afficher un espace (point) ou  une étoile
    for (line=0 ; line<lineNumber ; line++)
    {
        for (column=0 ; column<2*lineNumber ; column++)
        {
            if (column>=(-line+lineNumber-1)&&
                column<=( line+lineNumber-1))
               putchar('*');
            else
               putchar('.');
        }
        putchar('\n');
    }
```

**Exercice 6**

Affichez la table des produits pour N variant de 1 à 10 :




Les colonnes sont alignées grâce au format %3d.

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    const int MAX = 10; // nombre de lignes et de colonnes

    // Affichage de l'en-tete (premiere ligne)
    printf(" X*Y |");
    for (int j=0 ; j<=MAX ; j++)
        printf("%4d", j);   // gabarit de 4 cases
    printf("\n");

    // Affichage de la deuxieme ligne
    printf("______");
    for (int j=0 ; j<=MAX ; j++)
        printf("____");
    printf("\n");

    // Affichage du tableau
    for (int i=0 ; i<=MAX ; i++)
    {
        printf("%3d  |", i);
        for (int j=0 ; j<=MAX ; j++)
            printf("%4d", i*j);
        printf("\n");
    }
    system("pause");
    return 0 ;
}
```

**Exercice 7**

Faire un programme affichant le résultat d'une partie de ***FizzBuzz*** (sans erreur) pour les nombres de 1 à 100.

```
#include <stdio.h>
#include <stdlib.h>

/// EX 5.3.7 Jeu FizzBuzz
int main()
{
    const int NMax = 100;

    for (int number = 1; number <= NMax; number++)
    {
        if ( (0 == (number % 3)) && (0 == (number % 5)) )
            printf("Fizz Buzz,");
        else if ( 0 == (number % 3) )   // seulement un multiple de 3
            printf("Fizz,");
        else if ( 0 == (number % 5) )   // seulement un multiple de 5
            printf("Buzz,");
        else
            printf("%d,", number);
    }

    system("PAUSE");
    return 0;
}
```