# INF203 - Exercices semaine 5

## C : types, tests, boucles, parcours de tableaux

**Exercice 1 :**

Soit le programme suivant :

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

int main() {
    double a=0.5, b=0.25, c=0.1;
    printf("a = %.60f\n", a);
    printf("b = %.60f\n", b);
    printf("c = %.60f\n", c);
    printf("a+b = %f\n", a+b);
    printf("b+c = %f\n", b+c);
    double epsilon = 1 - (a+c+c+c+c+c);
    printf("epsilon = %.60f", epsilon);
    if (fabs(epsilon) <= 0.00001)
        printf (" vaut 0\n");
    else
        printf(" ne vaut pas 0\n");
    return 0;
}
```

1. Qu'affiche-t-il ? pourquoi ?
2. Comment pouvez vous faire pour que l'affichage soit plus pertinent ?
3. Comment pouvez vous faire pour que les calculs soient exacts ?

**Exercice 2 :**

Étant donnée la fonction suivante :

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

// Genere un entier entre 0 et borne-1
long generer_entier(long borne) {
    static int seme = 0;
    if (!seme) {
        srandom(getpid());
        seme = 1;
    }
    return random() % borne;
}
```

on peut écrire le programme suivant :

```
#include <stdio.h>
#include "generer_entier.c"

int main() {
    long x = generer_entier(100);
    printf("J'ai genere l'entier %ld\n", x);
    return 0;
}
```

1. Écrivez un programme qui génère deux entiers, les affiche et affiche le maximum des deux en utilisant une fonction `max2` que vous écrirez pour l'occasion.
2. Écrivez un programme qui génère trois entiers, les affiche et affiche le maximum des trois
   (a) d'abord en utilisant `max2` ;
   (b) ensuite sans l'utiliser.
3. Même question avec $n$ entiers, où $n$ est un entier défini au début du programme.

**Exercice 3 :**

1. Répétez la génération d'un nouvel entier jusqu'à ce que celui-ci soit égal à 42.
2. Répétez l'exercice précédent 100 fois (pas à la main !) et calculez le nombre moyen de tirages qu'il a fallu faire pour atteindre 42.

**Exercice 4 :**

Écrire un programme en C qui définit un entier `n` et qui affiche la forme suivante, constituée de `n` lignes (ci-dessous exemple avec `n = 5`). Pour le a) par exemple, on affiche `n` lignes, dans chacune desquelles on compte de `1` à `n`.

```
a. 12345   b.  1       c.      1   d.  12345   e.  12345   f.      1
   12345       12             12       1234        1234         121
   12345       123           123       123          123        12321
   12345       1234         1234       12            12       1234321
   12345       12345       12345       1              1      123454321
```

**Exercice 5 :**

1. Utilisez le générateur fourni pour remplir un tableau de 10 entiers avec des valeurs aléatoires comprises entre 0 et 99 et l'afficher.
2. Écrivez une fonction pour calculer la moyenne des éléments d'un tableau et complétez le programme précédent pour afficher la moyenne des éléments d'un tableau généré.
3. Écrivez une fonction permettant de chercher un élément de valeur donnée dans un tableau donné. Utilisez-la dans le programme précédent pour indiquer si le tableau contient 0, 42 ou 99.

**Exercice 6 :**

On peut déterminer si un entier $n$ est un nombre premier en déterminant s'il est divisible par l'un des nombres premiers compris dans l'intervalle $[0; n-1]$. On peut énumérer les nombres premiers compris entre 2 et $N$ de la manière suivante :

— on considère une liste $L$, initialement vide
— pour chaque entier $n$ compris entre 2 et $N$, dans l'ordre :
  — si $n$ est divisible par l'un des entiers de $L$ alors il n'est pas premier
  — sinon, ajouter $n$ à $L$

Cet algorithme s'appelle le crible d'Érastosthène. La liste $L$ peut être stockée dans un tableau de taille suffisante en conservant sa taille dans une variable entière. On peut tester si un nombre $x$ divise un nombre $n$ avec l'opérateur `%` (modulo) : `n%x` est le reste de la division entière de $n$ par $x$ ; s'il vaut 0, $x$ divise $n$. En utilisant cet algorithme, énumérez les nombres premiers compris entre 2 et 100.

**Exercice 7 :**

Qu'affiche le programme suivant ?

```
#include <stdio.h>
int main() {
    int qd=0b101010;
    printf("Voici un entier :\n");
    printf("En base 8 : %o, en base 10 : %d, en base 16 : %x\n", qd, qd, qd);
```

```
    return 0; }
```