Bases de la Programmation
TP 1 – Bases du langage

Exercice 1 : Premier programme

Si vous ne l'avez pas déjà fait en lisant le cours, commencez par écrire, compiler et exécuter un programme affichant un simple message à l'écran.

Exercice 2 : Conditionnelles

1. Écrire un programme lisant un entier au clavier, puis affichant :
   - "`Gagné !`" si cet entier vaut `42`,
   - "`Perdu !`" sinon.
2. Écrire un programme lisant un entier au clavier, puis affichant
   - "`Un`" si cet entier vaut `1`,
   - "`Deux`" s'il vaut `2`,
   - "`Autre`" sinon
3. Écrire un programme lisant un entier au clavier puis affichant, parmi les trois messages ci-dessous, celui adapté à la valeur de l'entier lu :
   - "`négatif`"
   - "`entre 0 et 100`"
   - "`superieur a 100`"

Exercice 3 : Boucles simples

Écrire un programme lisant un entier `n` positif[1], au clavier, puis affichant successivement :

1. Une ligne de "`*`" de longueur `n`, un retour à la ligne.
2. Une colonne de "`*`" de hauteur `n`, un retour à la ligne.
3. Les carrés des nombres entre `1` et `n` inclus (`1 4 9 16`...), un retour à la ligne.

Exercice 4 : Boucles avec *flag*

Écrire un programme lisant un entier `n` positif au clavier, puis lisant successivement `n` entiers et enfin, indiquant à l'utilisateur si le nombre 6 se trouvait parmi ces `n` entiers.

1. Il faudrait en principe vérifier que la valeur lue satisfait cette condition, afficher sinon un message, interrompre le programme ou redemander une valeur, etc. Ne perdez pas de temps à écrire ce genre de choses dans de simples exercices, cela ne vous apprendra rien de plus.

Exercice 5 : Boucles avec accumulation

1. Écrire un programme lisant un entier `n` positif au clavier et affichant la somme des cubes des `n` premiers entiers naturels positifs. Par exemple, si l'utilisateur entre `5`, votre programme devra afficher `225` :

$$\begin{aligned}\sum_{k=1}^{5} k^3 &= 1^3 + 2^3 + 3^3 + 4^3 + 5^3 \\ &= 1 + 8 + 27 + 64 + 125 \\ &= 225\end{aligned}$$

2. Écrire un programme lisant un entier `n` positif au clavier, puis lisant successivement `n` entiers et enfin, affichant la somme (entière) et la moyenne (sous forme de `double`) des nombres lus.
3. Ecrire un programme lisant un entier `n` positif au clavier, puis affiche la *factorielle* de `n`. On a par définition $0! = 1$, et pour tout $n \geqslant 1$:

$$n! = 1 \times 2 \times \ldots \times n$$

   Modifier ce programme afin qu'il affiche les factorielles de chaque entier compris entre `1` et `n` :

```
1! = 1
2! = 2
3! = 6
4! = 24
5! = 120
6! = 720
...
```

Exercice 6 : Tirage de nombres aléatoires

L'usage des fonctionnalités décrites dans cet exercice n'est possible qu'en commençant le code-source par trois directives :

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

Le tirage de nombres aléatoires[2] est possible dans un programme. L'expression `rand()` s'évalue en un nombre tiré entre `0` et un "très grand" entier.

Seul problème : les évalutions de `rand()` produisent toujours la même suite de valeurs d'une exécution à l'autre, à moins que l'on écrive *une et une seule fois* au début de `main` :

```c
srand(time(NULL));
```

On peut ensuite écrire, n'importe où ailleurs dans le code, par exemple :

2. Tirés au hasard.

```
printf("%d\n", rand() % 10);
printf("%d\n", rand() % 10);
```

L'opérateur `%` calcule le reste de la division entière de son argument gauche par son argument droit. Les deux instructions ci-dessus affichent donc deux nombres aléatoires entre `0` inclus et `10` exclus.

1. Mettre en forme les éléments ci-dessus en un programme compilable et exécutable [3].
2. Modifier le programme afin qu'il affiche 10 nombres tirés au hasard entre 1 et 6, suivi d'un retour à la ligne :

   ```
   6 4 3 2 5 3 1 4 6 5
   ```

3. Modifier le programme afin qu'il affiche 100 nombres compris entre 1 et 6, sous forme de matrice 10×10 [4].

   ```
   6 4 3 2 5 3 1 4 6 5
   3 3 1 5 6 2 1 2 5 2
   ...
   ```

4. Compléter le programme pour qu'il affiche le nombre de fois où le nombre 6 a été tiré.

Exercice 7 : Suite de Fibonacci

La *suite de Fibonacci* (f_n) est définie par :

$$\begin{cases} f_0 &= 1 \\ f_1 &= 1 \\ f_n &= f_{n-2} + f_{n-1} \quad \text{pour tout } n \geq 2 \end{cases}$$

Ecrire un programme itératif [5] s'exécutant en espace constant [6] lisant un entier $n \geqslant 0$ au clavier, puis affichant les termes $f_0, \ldots, f_n$ de la suite de Fibonacci.

Exercice 8 : Boucles, *flags*, boucles imbriquées

On rappelle qu'un entier n est premier s'il est supérieur ou égal à 2, et si aucun nombre p entre 2 et $n-1$ ne divise n [7].

1. Ecrire un programme lisant un entier `n` au clavier, et affichant un message indiquant si ce nombre est premier ou non.
2. Modifiez votre programme de manière à afficher la suite des nombres premiers compris entre 1 et 100 [8] : 1, 3, 5, 7, 11, 13, etc.

3. Essayez `man srand` ou `man 3 rand`, ainsi que `man 2 time`.
4. Le reste de la division entière de `i` par 10 s'ecrit `i % 10`.
5. A l'aide d'une boucle.
6. Sans utiliser de tableaux.
7. C'est-à-dire si `n % p` est non nul pour tout `p` compris dans cet intervalle.
8. Cet algorithme n'est pas très efficace — nous verrons un algorithme meilleur lors du TP sur les tableaux.