Initiation à la Programmation C
Travaux Pratiques - L2
Itérations, aiguillage, fonctions

Dans cette séance de travaux pratique, nous abordons les points suivants :

- les différentes formes d'itérations (`while`, `for` et `do ... while`) ;
- le formatage lors d'un appel à `scanf` ;
- la réalisation d'un menu ;
- le passage par adresse d'une valeur à une fonction.

▸ Exercice 1. Affichage de caractères et de leur code ASCII

Ecrire un programme qui affiche les caractères de `'0'` à `'9'`, puis de `'A'` à `'Z'`, suivis la valeur de leur code `ASCII`.

Exemple : `caractere A : code 65`

▸ Exercice 2. Devine un nombre

Ecrire un programme qui détermine au hasard un nombre entier entre `1` et `1000` puis laisse l'utilisateur le deviner en lui indiquant si ses propositions sont trop grande ou trop petite.
Lorsque le jeu se termine, le programme affiche le nombre d'essais `Bravo ! Gagne en ... essais.`.

Exemple :
```
J'ai choisi un nombre entre 1 et 1000.
A vous de le deviner.
Saisisez un nombre : 500
Trop petit
Saisisez un nombre : 750
Trop petit
Saisisez un nombre : 875
Trop grand
Saisisez un nombre : 811
Bravo ! Gagne en 4 essais
```

Indication :
Pour tirer un nombre au hasard, nous utiliserons la fonction `rand()` qui tire une nombre entre `0` et `RAND_MAX`. Pour en déduire un nombre entre `0` et `max - 1`, il suffit alors de faire `rand() % max`.
Enfin, pour initialiser la graine du générateur pseudo-aléatoire, en utilisant (presque) l'heure de lancement du processus, on utilisera l'instruction `srand(time(NULL))`.
Ces fonctions sont disponibles dans les bibliothèques `time.h` et `stdlib.h`.

▸ Exercice 3. Lecture de caractères

1. Écrire un programme qui lit deux caractères avec deux appels à `scanf`, puis qui les affiche sur deux lignes un caractère par ligne entouré du caractère `'*'`.

   Rappel : Le spécificateur de format `"%c"` indique à `scanf` qu'un caractère doit être lu. Avec ce formatage, les séparateurs sont pris en compte.

Testez le programme précédent en entrant les caractères de différentes manières :

- ⇝ un par ligne ;
- ⇝ deux sur la même ligne ;
- ⇝ en utilisant des espaces ;
- ⇝ ...

2. Si l'on veut que `scanf` ignore les séparateur, il faut en ajouter un `' '` dans le format.
   (a) Remplacer `scanf("%c")` par `scanf(" %c")` dans le programme précédent.
   (b) Refaire les tests pour comparer les comportements.

▸ **Exercice 4. Menu** On souhaite réaliser un menu pour permettre à l'utilisateur de choisir entre trois actions symbolisées par les couleurs `Rouge`, `Vert` et `Bleue`.

- Ecrire une fonction d'affichage du menu. Le choix des couleurs se fera à l'aide de valeurs alphabétiques ; la possibilité de sortir du programme se fera en rajoutant un choix de sortie au menu.
- Ecrire une fonction réalisant la saisie du choix par l'utilisateur. Un message d'erreur sera affiché si le choix n'appartient pas à l'ensemble des choix possibles. Sa valeur de retour sera `1` si l'utilisateur souhaite s'arrêter ; elle vaudra `0` sinon.
- Ecrire un programme réalisant le menu souhaité. Le programme affichera chaque couleur choisie par l'utilisateur, puis réaffichera le menu tant que le choix de sortie n'a pas été sélectionné.

▸ **Exercice 5. Passage par adresse**
Le but de l'exercice est de visualiser le fonctionnement du passage par adresse.

1. Ecrire une fonction `void ajouteDix(int * entier)` qui augmente `*entier` de `10`.

   Cette fonction affichera la valeur de `*entier` et l'adresse de `entier` avant et après l'appel.

2. Ecrire un `main` qui :
   (a) lit un entier `n` saisi par l'utilisateur, l'affiche puis affiche son adresse ;
   (b) appelle la fonction `ajouteDix` ;
   (c) réaffiche `n` ainsi que son adresse.
3. Faites le même test avec une fonction d'échange de deux variables.

▸ **Exercice 6. Papier, caillou, ciseaux**
On veut écrire un programme permettant de jouer au jeu "papier, caillou, ciseau".

On codera `caillou` par 0, `papier` par 1 et `ciseaux` par 2. Chaque joueur propose son pari en tapant 0, 1 ou 2. Pour déterminer lequel des deux joueurs a gagné, on utilise l'algorithme suivant, où $J1$ est le pari du joueur 1 et $J2$ celui du joueur 2 :

- Si $J1$ et $J2$ sont identiques le match est nul,
- $J1$ gagne si $J1 = (J2 + 1) \bmod 3$,

- $J2$ gagne dans le cas restant.

1. Pourquoi l'algorithme donné permet de décider du gagant ?
2. Ecrire une fonction `LireInf2` qui effectue la saisie contrôlée d'un entier entre 0 et 2.
3. Ecrire une fonction `arbitre` qui recoit les paris des deux joueurs et qui renvoie 0 si le match est nul, 1 si le joueur 1 a gagné et 2 si c'est le joueur 2.
4. Ecrire un programme qui arbitre 10 parties et qui affiche le score.

   On pourra encore découper le problème avec l'aide de plusieurs fonctions.