Bases de C
TP 1 - Bases

Ce TP a pour objectif de vous faire découvrir le langage C et les outils nécessaires pour le faire fonctionner. Vous démarrerez depuis l'écriture de quelques lignes de code que vous compilerez, pour finalement réaliser un petit programme de statistiques. Durant tout l'exercice, vous utiliserez au fur et à mesure quelques notions essentielles du langage C.

1 Programmes simples

1) Écrivez un programme en C calculant le produit de deux nombres (positifs ou négatifs) en effectuant des additions. Vous afficherez l'état des deux paramètres au fur et à mesure ligne après ligne.

Pour vous aider, voici le prototype attendu pour la fonction :

```
int Multiplication(int x, int y)
```

2) Écrivez un programme en C calculant la puissance de x^n (pour n positif ou nul) en effectuant des multiplications. Vous afficherez l'évolution des deux paramètres au fur et à mesure.

3) Écrivez un programme en C testant la parité d'un nombre n.

Essayez de renvoyer directement le test que vous feriez dans un **`if`** afin de mieux comprendre ce que vérifie réellement une condition en C. Testez votre code avec plusieurs cas.

4) Écrivez un programme en C calculant la factorielle d'un nombre, et affichant les valeurs successives.

5) Écrivez un programme en C affichant un carré de caractères #. La taille du carré sera donnée en paramètre. Le carré sera rempli de vide (seule la bordure sera constituée de caractères).

6) Écrivez une fonction transformant un format horaire en un format uniquement composé de secondes. Cette fonction prendra 3 entiers en paramètre (les heures, les minutes, et les secondes) et les convertira en secondes. Par exemple, 1h 23m 45s deviendra 5025 secondes.

7) Écrivez une autre fonction de conversion horaire qui prend cette fois un unique entier qui respecte un format précis (hhmmss) pour le convertir en secondes. Par exemple le paramètre 153042 signifie 15h 30m 42s qu'il faut convertir en secondes.

8) Écrivez une fonction qui transforme un nombre en son miroir. Dans une première version, vous ferez une procédure qui affiche simplement les caractères au fur et à mesure, et dans une deuxième version, vous ferez une fonction qui construit un entier que vous retournerez. Par exemple, pour 12034, son miroir sera 43021. Autre exemple : 2000 aura comme miroir 0002, c'est-à-dire 2. Attention aux nombres composés d'un nombre pair/impair de chiffres.

9) Écrivez une fonction détectant si un nombre est un palindrome. Un palindrome est simplement un mot ou un nombre composé des mêmes caractères ou chiffres sur sa première moitié par rapport à sa deuxième moitié. Par exemple, 27972 est un palindrome. 1331 est également un palindrome, mais 1664 n'en est pas un.

2 Conversions et autres usages

10) Écrivez une fonction convertissant les prix en euros en francs (rappel : le taux de conversion est de $1 € = 6,55957$ F.

11) Écrivez une fonction convertissant les prix en francs en euros.

12) Écrivez une fonction testant si un nombre est premier ou non.

13) Écrivez une fonction affichant la table de multiplication d'un nombre donné en paramètre.

14) Écrivez une procédure affichant la table ASCII avec les équivalences décimales, héxadécimales, puis le caractère. (Vous pouvez vous contenter des valeurs entre 32 et 126)

15) Écrivez une fonction qui transforme un entier en un tableau de 6 caractères interprétable comme une chaîne de caractères. (Vous pouvez faire une meilleure gestion de l'espace mémoire si vous vous en sentez le courage)

16) Écrivez une fonction demandant un code PIN à l'utilisateur (vous pourrez utiliser la fonction **`scanf(3)`** pour cela). La fonction doit afficher "Déverrouillage" si le code est trouvé, sinon, la fonction doit re-demander à l'utilisateur d'entrer un code PIN.

17) Écrivez une procédure qui contient une variable de type **`char`** que vous initialiserez. Affichez son contenu, sa taille, et son adresse en mémoire.

18) Copiez la procédure précédente et cette fois, n'initialisez pas son contenu, mais affichez les mêmes informations que précédemment.

19) Copiez la procédure précédente encore une fois, transformez la variable de type **`char`** en **`char*`**, puis allouez une seule case mémoire pouvant contenir un caractère avec **`malloc(3)`**. Affichez d'abord le contenu de ce pointeur, puis, allouez une valeur dans la case mémoire, et re-affichez les informations concernant le caractère.