# TD 2 : boucles et tableaux

Programmation en C (LC4)

Semaine du 5 février 2007

## 1 Boucles, tableaux à une ou deux dimensions

**Exercice 1** Écrivez une fonction `void somme(int n, int tab[])` qui prend en argument un tableau d'entiers `tab` et sa taille `n` et qui affiche la somme des nombres qu'il contient. Résolvez ce problème :
- en utilisant `while`,
- en utilisant `do - while`,
- en utilisant `for`.

**Exercice 2** Écrivez une fonction `int inverse(int nb)` qui vérifie que `nb` est strictement compris entre 0 et 10000 et, le cas échéant, affiche `nb` à rebours. Par exemple, si `nb`=1234, la fonction affichera 4321 (pensez à utiliser la fonction `modulo`).

**Exercice 3** Écrivez une fonction `tasse(int n, int tab[])` qui efface toutes les occurences du chiffre 0 dans le tableau `tab` et tasse les éléments restants (on remplira les cases vides en fin de tableau par des zéros).

**Exercice 4** Écrivez une fonction `inverse(int n,int tab[])` qui range les éléments du tableau `tab` dans l'ordre inverse sans utiliser de tableau supplémentaire.

**Exercice 5** Écrivez une fonction `fusion(int n, int tabA[], int m, int tabB[], int fus[])` qui prend en argument les tableaux `tabA` (de taille `n`) et `tabB` (de taille `m`) triés par ordre croissant et remplit le tableau `fus` (de taille `n+m`) avec les éléments de `tabA` et `tabB` triés par ordre croissant.

**Exercice 6** Écrivez une fonction `matrice2ligne(int x, int y, int** matrice, int ligne[])` qui écrit le contenu d'un tableau à deux dimensions de taille `ixj` dans un tableau en ligne de taille `i*j`. On écrira les lignes de `matrice[i][j]` à la suite dans le tableau `ligne[i*j]`.

```
a b c d
e f g h   ==>    a b c d e f g h i j k l
i j k l
```

**Exercice 7** Écrivez une fonction `insere(int n, int tab[], int val)` qui prend en argument un tableau `tab` de taille `n`, trié par ordre croissant dont la valeur de la dernière case est indéterminée et qui y insère l'entier `val` de manière à ce que le tableau reste trié.

**Exercice 8** Écrivez la fonction `factorielle_boucle(int n)` qui calcule `n!` en utilisant une boucle. Écrivez ensuite la fonction `factorielle_recursive(int n)` qui fait la même chose récursivement.

## 2 Reprise du TP1

**Exercice 9** Comment coder un pixel couleur dans un entier de 32 bits ? Écrivez une fonction `couleur_en_entier(int rouge, int vert, int bleu)` qui effectue un tel codage.

**Exercice 10** En utilisant les mêmes procédés, écrivez trois fonctions `donne_rouge(int couleur)`, `donne_vert(int couleur)` et `donne_bleu(int couleur)` qui prennent un entier codant les trois couleurs et renvoient un entier codant une couleur spécifique.