IN 101 - TD 02

23 septembre 2011

Énoncé

ENSTA ParisTech

Le but du TD est de bien comprendre la ligne de commande pour rentrer les paramètres, les tableaux et les boucles `for`.

Exercice 1 : La ligne de commande

Écrivez un programme qui additionne les nombres placés sur la ligne de commande, les affiche, affiche le nombre de paramètres et affiche la somme.

Exercice 2 : Les tableaux

Initialisez un tableau avec des nombres entrés sur la ligne de commande (au plus 10 entiers). Les questions suivantes seront faites dans le même fichier.

2.a] Affichez le tableau.

2.b] Comptez le nombre de valeurs supérieures à un seuil donné.

2.c] Affichez le tableau dans l'ordre inverse.

2.d] Inversez l'ordre des éléments du tableau et affichez le tableau.

2.e] Trouvez le maximum et le minimum du tableau.

Exercice 3 : Tableau aléatoirement initialisé

Pour utiliser le générateur de nombre aléatoire, on utilisera les fonctions `srand` et `rand`. Quel est l'include nécessaire pour les utiliser ? Quels sont les paramètres et les types de retour de ces fonctions ?

Ce type de générateur est une fonction déterministe qui utilise un germe. Si le germe n'est pas connu, la suite apparaît aléatoire. Pour initialiser ce germe, on utilise la fonction `srand` et ensuite à chaque fois que l'on veut une valeur aléatoire, on appelle la fonction `rand`. Comme `rand` renvoie des valeurs sur 32 bits, pour ne pas avoir des valeurs trop grandes, on évaluera `rand() % 100` pour obtenir un nombre entre 0 et 99.

Enfin, si on ne veut pas que l'utilisateur rentre la valeur du germe, on peut utiliser diverses techniques. Une solution est de prendre comme germe le numéro du processus. Sous Unix, on utilise la fonction `getpid`. Quels sont les includes nécessaires et les paramètres de cette fonction ?

3.a] Modifiez le programme de la question précédente pour qu'il utilise un tableau initialisé avec des valeurs aléatoires et vérifiez qu'il fonctionne toujours.

Exercice 4 : Tri à bulles

Le principe du tri à bulles est d'effectuer plusieurs passes sur le tableau, en faisant remonter les plus grands éléments du tableau vers la fin. À la première passe, le plus grand élément sera déplacé à la fin du tableau, à la i-ième passe, le tri metttra en place le i-ième plus grand élément. Chaque passe compare les éléments en position `tab[k]` et `tab[k+1]` et les échange si besoin, pour $k = 0$ à $N - 2$. Cette borne $N - 2$ pourra être modifiée une fois qu'on aura bien compris le fonctionnement de l'algorithme. En effectuant $N - 1$ passes, on est certain qu'à la fin tous les éléments seront à leur place.

4.a] Programmez le tri à bulles.

Exercice 5 : Calcul de Factoriel 100 !

On souhaite calculer avec de grands nombres qui dépassent 32 bits. On va stocker des entiers de 200 digits (chiffres décimaux) dans un tableau en base 100. Dans la case `i`, on mettra le $(2i)^{\text{ieme}}$ et $(2i+1)^{\text{ieme}}$ digits. Un tel entier sera de type : *`int`* `digit[100];`. Par exemple, l'entier `9876543201` sera représenté en stockant la longueur `5` dans `digit[0]` et le tableau contiendra `digit[1]=01`, `digit[2]=32`, `digit[3]=54`, `digit[4]=76`, `digit[5]=98`, et `digit[i]=0` pour $i = 6$ à 99.

5.a] Pourquoi les types *`int`* ou *`long long`* ne sont-ils pas suffisant ?

5.b] Écrivez une fonction qui affiche un tel nombre sans afficher les zéros en tête (faites attention, au cas où la case contient deux digits qui s'écrivent comme un seul : par exemple, la case `digit[1]` contient `01`). On considère que le grand entier est une variable globale au programme, donc la fonction ne prend pas d'argument.

5.c] Écrivez une fonction qui prend un entier < 100, qui initialise le grand entier en variable globale.

5.d] Écrivez une fonction qui effectue la multiplication d'un tel nombre par un entier ≤ 100 (n'oubliez pas la retenue).

5.e] Écrivez une fonction qui calcule factoriel 100.