**IN 101 - TD 02**
23 septembre 2011

# Corrigé

ENSTA ParisTech

## Solution 1 : La ligne de commande

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(int argc, char* argv[]) {
5    int i;
6    int somme = 0;
7    printf("Le nombre de paramètres est %d\n",argc-1);
8
9    for (i=1; i<argc; i++){                  // boucle sur les arguments
10     somme = somme + atoi(argv[i]);
11     printf("%s\t",argv[i]);          // affiche l'argument i
12   }
13   printf("\nLa somme vaut %d\n", somme);
14   return 0;
15 }
```

Notez l'utilisation de `\t` à la ligne 11 qui permet d'afficher une tabulation pour "aligner" les éléments.

## Solution 2 : Les tableaux

Le programme suivant effectue toutes les opérations demandées, dans l'ordre.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #define N 10
4
5  int main(int argc, char* argv[]) {
6    int i, tab[N], n, seuil, compteur, tmp, max, min;
7    n=argc-1;
8    // on vérifie qu'il n'y a pas trop d'arguments
9    if (n > N) {
10     printf("Trop d'arguments !\n");
11     return 1;
12   }
13   // on initialise le tableau avec les arguments
14   for (i=0; i<n; i++) {
15     tab[i] = atoi(argv[i+1]);
16   }
17
18   // affichage du tableau
19   for (i=0; i<n; i++) {
20     printf("%d\t",tab[i]);
```

```
21    }
22    printf("\n");
23
24    // comptage des valeurs supérieures à un seuil
25    printf("Entrer un seuil: ");
26    scanf("%d", &seuil);
27    compteur = 0;
28    for (i=0; i<n; i++) {
29      if (tab[i] > seuil) {
30        compteur++;
31      }
32    }
33    printf("Le nombre de valeurs supérieures à %d est %d.\n", seuil, compteur);
34
35    // affichage en ordre inverse
36    for (i=n-1; i>=0; i--) {
37      printf("%d\t",tab[i]);
38    }
39    printf("\n");
40
41    // inverser l'ordre des éléments du tableau et l'afficher
42    for (i=0; i<n/2; i++) {
43      tmp = tab[i];        // tmp permet de stocker tab[i] pendant l'échange
44      tab[i] = tab[n-1-i];
45      tab[n-1-i] = tmp;
46    }
47    for (i=0; i<n; i++) {
48      printf("%d\t", tab[i]);
49    }
50    printf("\n");
51
52    // trouver le max et le min du tableau
53    max = tab[0];
54    min = tab[0];
55    for (i=1; i<n; i++) {
56      if(tab[i] > max) {
57        max = tab[i];
58      } else if(tab[i] < min) {
59        min = tab[i];
60      }
61    }
62    printf("Le maximum est %d et le minimum est %d.\n", max, min);
63    return 0;
64  }
```

## **Solution 3 :** Tableau aléatoirement initialisé

**3.a ]** Voici le début de la fonction main qui initialise avec des valeurs aléatoires, le reste du programme ne change pas.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#define N 10

int main(int argc, char* argv[]) {
  int i, tab[N], n, seuil, compteur, tmp, max, min;

  // on initialise la graine en fonction du PID
  srand(getpid());

  // on initialise tout le tableau avec rand
  for (i=0; i<N; i++) {
    tab[i] = rand() % 100;
  }
  ...
```

## **Solution 4 :** Tri à bulles

**4.a ]** À chaque passe, on arrète les comparaisons à $j = N-2-i$ (et donc $j+1 = N-1-i$) car les $i$ derniers éléments du tableaux sont déjà rangés à leur place au moment du $i$-ème passage.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#define N 10

int main() {
  int i, j, tab[N], tmp;
  // on initialise la graine en fonction du PID
  srand(getpid());

  // on initialise tout le tableau avec rand
  for (i=0; i<N; i++) {
    tab[i] = rand() % 100;
  }

  // on fait le tri à bulles du tableau
  for (i=0; i<N-1; i++) {       // on effectue N-1 passes
    for (j=0; j<N-1-i; j++) {   // les i derniers sont déjà en place
      if (tab[j] > tab[j+1]) {  // si les élements j et j+1 sont mal ordonnés
        tmp = tab[j];           // on les inverse
        tab[j] = tab[j+1];
        tab[j+1] = tmp;
      }
    }
  }
```

**Solution 5 :** Calcul de Factoriel 100 !

**5.a ]** Les types *int* et *long long* sont limités à 32 et 64 bits respectivement. Ce n'est pas assez pour stocker des nombres à 200 chiffres. Un *int* peut stocker jusque 2 milliards environ (4 milliards pour un *unsigned int*) et un *long long* une dizaine de milliards de milliards, ce qui n'est pas non plus assez...

**5.b ]** On peut afficher un tel chiffre avec la fonction suivante. Le `%02d` signifie d'afficher l'entier sur 2 caractères, en remplissant avec des 0.

```
#define N 100
int digit[N];
void affiche() {
  int i;
  printf("%d",digit[digit[0]]);  // affiche le premier chiffre
  for (i=digit[0]-1; i>0; i--) {
    printf("%02d", digit[i]);
  }
  printf("\n");
}
```

**5.c ]** Voici l'initialisation :

```
void init(int v) {
  int i;
  if (v >= 100) {
    printf("Valeur trop grande !\n");
    return;
  }
  digit[0] = 1;
  digit[1] = v;
  for (i=2; i<N; i++) {
    digit[i] = 0;
  }
}
```

**5.d ]** La multiplication est l'opération la plus compliquée ici, mais le programme est assez simple : `res` contient la retenue restant de la case précédente, et on y ajoute le résultat du produit de la case courante par `v`. On garde la partie plus petite que 100 pour la case courante et le reste va dans la retenue de la case suivante. La variable `last` sert à savoir quelle est la dernière case non nulle après le produit, pour mettre à jour la longueur.

```
void mult(int v) {
  int i, last, res = 0;
  for (i=1; i<=digit[0]+1; i++) {
    res += v*digit[i];
    digit[i] = res % 100;
    res /= 100;
    if (digit[i] != 0) {
      last = i;
    }
```

```
10   }
11   digit[0] = last;
12 }
```

**5.e ]** Le calcul du factoriel est maintenant assez simple :

```
1  int main() {
2    int i;
3    // on stocke 1 dans le grand entier
4    init(1);
5    for (i=2; i<=100; i++) {
6      mult(i);    // on multiplie par i
7    }
8    affiche();    // on affiche le résultat
9    return 0;
10 }
```