



COMPTE RENDU TRAVAUX PRATIQUES DE LANGAGE C

TP0

Etudiant : Assane Thiao

Formation : Télécommunications et Réseaux

Niveau : ING1

Année : 2024/2025

Tables des matières

Tables des matières.....	2
1. Objectif du code	3
2. Explication des fonctionnalités	3
3. Erreurs et problèmes dans le code	3
✖ Problèmes critiques.....	3
⚠ Problèmes potentiels.....	4
4. Code corrigé.....	6
5. Test.....	7
6. Conclusion.....	7

1. Objectif du code

Ce programme en C génère un tableau de nombres aléatoires, l'affiche, le trie en ordre croissant, puis l'affiche à nouveau avant d'attendre que l'utilisateur appuie sur **Entrée** pour quitter.

2. Explication des fonctionnalités

- **initialiser(int *tab)**
→ Remplit un tableau avec des nombres aléatoires compris entre **0 et 999**.
- **trier(int *tab)**
→ Trie le tableau avec **l'algorithme du tri à bulles** (*Bubble Sort*).
- **afficher(int tab[])**
→ Affiche le contenu du tableau.
- **main(int argc, char *argv[])**
→ Vérifie l'usage correct du programme, initialise le tableau, l'affiche, le trie et le réaffiche.

3. Erreurs et problèmes dans le code

✖ Problèmes critiques

1. Définition manquante de SIZE

- a. SIZE n'est jamais défini dans le code, ce qui entraîne une erreur de compilation.
- b. ♦ **Solution** : Ajouter `#define SIZE 10` (ou une autre valeur) au début du code.

```
...{  
...    int tab[SIZE];  
  
#define SIZE 100 /* Déclaration de SIZE en tant que macro pour fixer la taille d'un tableau */
```

2. Erreur de syntaxe dans trier()

- a. `int tmp = tab[j]` → Il manque un point-virgule ; après `tmp = tab[j]`.

```
...     int tmp = tab[j]
...     tab[j] = tab[j+1];
```

- b. `for (i = 0; i < SIZE; i++)` → Il manque la déclaration de **j** dans la boucle for.

```
...     int tab[SIZE];
```

3. Erreur de syntaxe dans afficher()

- a. `for (i = 0; i <= SIZE; i++)` →
i. `i+` doit être `i++`.
ii. `<= SIZE` doit être `< SIZE` (sinon dépassement du tableau).

```
...     for (i = 0; i <= SIZE; i++)
...         printf ("%d", tab[i]);
```

⚠ Problèmes potentiels

4. Utilisation incorrecte de getchar()

- a. Le retour de `getchar()` est ignoré, ce qui peut causer un avertissement.
b. ♦ **Solution**: `int c = getchar(); (void)c;`

5. Absence de `stdlib.h` et `stdio.h` dans les includes

- a. `stdio.h` manque pour `printf()` et `fprintf()`.
b. ♦ **Solution**: Ajouter `#include <stdio.h>`

```
/*#include <time.h> ..... pour time() et
#include <stdlib.h> ..... pour srand() et rand()
ne sont pas des bibliotheque utilise dans le code donc ce n'est pas la peine de les importer*
```

```
...
{
    fprintf (stderr, "Utilisation: %s\n"
    return 1;
```

```
|---fprintf (stderr, "Utilisation: %s\n", argv[0]);  
|---return 1;  
|
```

4. Code corrigé

```
/* test.c - 13/03/2013 - Romain Dubessy Exemple de programme. */
#include <stdio.h>           /*bibliotheque standard des entrees et sorties*/

#define SIZE 10             /*fixer la taille du tableau*/

void initialiser(int* tab);
void trier(int* tab);
void afficher(int tab[]);
int main(int argc, char* argv[])
{
    if (argc != 1)
    {
        fprintf(stderr, "Utilisation: %s\n", argv[0]);
        return 1;
    }
    else
    {
        int tab[SIZE];
        initialiser(tab);
        afficher(tab);
        trier(tab);
        afficher(tab);
        printf("Taper entrée pour quitter!\n");
        getchar();
    }
    return 0;
}
void initialiser(int* tab)
{
    int i;
    srand(time(NULL));
    for (i = 0; i < SIZE; i++)
        tab[i] = rand() % 1000;
}
void trier(int* tab)
{
    int i;

    int j; /*declaration de j*/

    for (i = 0; i < SIZE; i++)
    {
        for (j = 0; j < SIZE - 1 - i; j++)
        {
            if (tab[j] > tab[j + 1])
            {
                int tmp = tab[j];      /*imission du point virgule*/
                tab[j] = tab[j + 1];
                tab[j + 1] = tmp;
            }
        }
    }
}
void afficher(int tab[])
{
    int i;
    printf("Tableau:");
    for (i = 0; i <= SIZE; i++)      /*mission du + on fait i++ au lieu de i+=1*/
        printf(" %d", tab[i]);
    printf("\n");
}
/* test.c */
```

5. Test

```
C:\Users\e12410656\source\repos\partieTest\Debug\partieTest.exe
Tableau: 931 499 675 705 196 780 142 607 416 512 -858993460
Tableau: 142 196 416 499 512 607 675 705 780 931 -858993460
Taper entrée pour quitter!
```

6. Conclusion

Le code est fonctionnel mais contenait plusieurs erreurs syntaxiques et de logique qui empêchaient la compilation et l'exécution correcte. Après correction, il fonctionne correctement et trie un tableau de 10 nombres aléatoires en ordre croissant.