AMICHI Lounès – TP B2
15 Octobre

IAP - TP N°2

Objectifs :

- Manipulation des tableaux monodimensionnels et multidimensionnels
- Connaissance des méthodes de tri simples

Partie 1 – Transcription d’une analyse en pseudo-code en code C

Exercice n°1

```c
#include <stdio.h>
#define TAILLE 20

int main() {
   int T[TAILLE];
   int nb;
   int error = 0;
   int i = 0;
   int comp;

   // Saisie de la taille du vecteur
   do {
      printf("Combien de valeurs voulez-vous saisir? (Entre 1 et %d)\n", TAILLE);
      scanf("%d", &nb);
      error++;
      if (error == 3) {
         printf("3 saisies erronées, fin du programme. (Erreur 1)\n");
         return 1;
      }
   } while (nb < 1 || nb > TAILLE);

   // Saisie des composantes du vecteur
   for (i; i < nb; i++) {
      printf("Saisir la composante %d du vecteur:\n", i);
      scanf("%d", &comp);
      T[i] = comp;
   }

   // Affichage du vecteur
   for (i = 0; i <nb; i++) {
      printf("V[%d] = %d\n", i, T[i]);
   }
   return 0;
}
```

```
Combien de valeurs voulez-vous saisir? (Entre 1 et 20)
0
Combien de valeurs voulez-vous saisir? (Entre 1 et 20)
21
Combien de valeurs voulez-vous saisir? (Entre 1 et 20)
-3
3 saisies erronées, fin du programme. (Erreur 1)
lamichi@IG2I-2025-74:~/Documents/GitHub/cours/intro. à la prog.
 structurée/TP2$ ./exo1
Combien de valeurs voulez-vous saisir? (Entre 1 et 20)
4
Saisir la composante 0 du vecteur:
12
Saisir la composante 1 du vecteur:
34
Saisir la composante 2 du vecteur:
30
Saisir la composante 3 du vecteur:
-22
V[0] = 12
V[1] = 34
V[2] = 30
V[3] = -22
```

Exercice n°2

```c
#include <stdio.h>
#define TAILLE 20

int main() {
   int T[TAILLE][TAILLE];
   int nby = 0;
   int nbx = 0;
   int error = 0;
   int i;
   int j;
   int comp;

   // Saisie des lignes
   do {
      printf("Combien de lignes voulez-vous saisir? (Entre 1 et %d)\n", TAILLE);
      scanf("%d", &nby);
      if (error == 3) return 1;
      error++;
   } while (nby < 1 || nby > TAILLE);
   error = 0;
   do {
      printf("Combien de colonnes voulez-vous saisir? (Entre 1 et %d)\n", TAILLE);
      scanf("%d", &nbx);
      if (error == 3) return 1;
      error++;
   } while (nbx < 1 || nbx > TAILLE);

   // Saisie des valeurs du tableau
   for (i = 0; i < nby; i++) {
      for (j = 0; j < nbx; j++) {
         printf("Saisir la valeur de la matrice à (%d,%d):\n", j, i);
         scanf("%d", &comp);
         T[i][j] = comp;
      }
   }

   // Affichage du vecteur
   for (i = 0; i <nby; i++) {
      printf("V[%d]: ", i);
      for (j = 0; j<nbx; j++) {
         printf("%d     ", T[i][j]);
      }
      printf("\n");
   }
   return 0;
}
```

```
Combien de lignes voulez-vous saisir? (Entre 1 et 20)
0
Combien de lignes voulez-vous saisir? (Entre 1 et 20)
21
Combien de lignes voulez-vous saisir? (Entre 1 et 20)
-5
3 saisies erronées, fin du programme. (Erreur 1)
lamichi@IG2I-2025-74:~/Documents/GitHub/cours/intro. à la prog.
 structurée/TP2$ ./exo2
Combien de lignes voulez-vous saisir? (Entre 1 et 20)
2
Combien de colonnes voulez-vous saisir? (Entre 1 et 20)
2
Saisir la valeur de la matrice à (0,0):
25
Saisir la valeur de la matrice à (1,0):
-25
Saisir la valeur de la matrice à (0,1):
12
Saisir la valeur de la matrice à (1,1):
12
V[0]: 25     -25
V[1]: 12     12
```

Partie II – Implémenter les méthodes de tris dans le programme fourni en tant que squelette.

Le code des exercices 3 à 5 est directement extrait du programme squelette. Les mesures de complexité sont répertoriées dans un tableau.

Exercice n°3

```
for (i = 1; i < nb; i++) {
        min = i;
        for (j = i + 1; j<=nb; j++) {
                if (T2[j] < T2[min]) min = j;
        }
        aux = T2[min];
        T2[min] = T2[i];
        T2[i] = aux;
}
```

Exercice n°4

```
for (i = 0; i < nb; i++) {
        for (j = 0; j < nb - i; j++) {
                if (T2[j] > T2[j+1]) {
                        aux = T2[j];
                        T2[j] = T2[j+1];
                        T2[j+1] = aux;
                }
        }
}
```

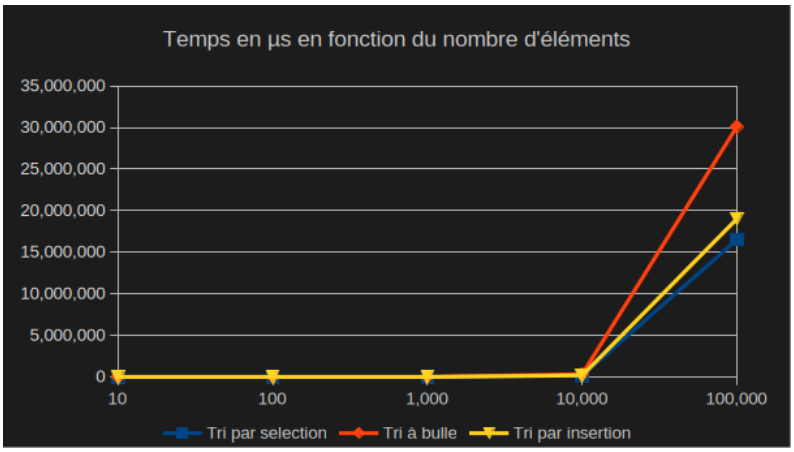
Exercice n°5

```
for (i = 2; i <= nb; i++) {
        aux = T2[i]; // on utilise aux pour économiser de la mémoire
        j = i-1;
        while (j>= 1 && T2[j] > aux) {
                T2[j+1] = T2[j];
                j--;
        }
                T2[j+1] = aux;
}
```

Étude de la complexité

Temps des tris

Format du tableau	Valeurs dans le tableau	10	100	1,000	10,000	100,000
Aléatoire	Tri par selection	1	59	4,347	344,257	38,002,873
	Tri à bulle	2	83	14,247	929,785	112,752,565
	Tri par insertion	1	51	2,218	248,368	19,854,611
Ordre croissant	Tri par selection	2	16	1,744	152,258	16,542,343
	Tri à bulle	2	18	1,642	187,953	19,504,351
	Tri par insertion	2	1	11	58	851
Ordre décroissant	Tri par selection	1	19	1,844	182,768	16,504,235
	Tri à bulle	1	32	3,645	343,276	30,120,351
	Tri par insertion	2	21	1,995	187,535	19,032,564



Cette courbe montre le temps pris par chaque tri en fonction du nombre d'éléments dans le pire des cas. On remarque que plus le nombre d'éléments est grand, plus le tri à bulle prend de temps. Les tris par insertion et sélection restent assez proches l'un de l'autre, mais on observe quand même une meilleure perfomance chez le tri par insertion.

Dans le pire des cas, les trois algorithmes de tri sont de complexité d'ordre $O(n^2)$. Les résultats mesurés ne correspondent pas à la complexité théorique.