COMPTE RENDU TRAVAUX PRATIQUES DE LANGAGE C

TP1 : Manipulation de vecteur

Etudiant : Assane Thiao

Formation : Télécommunications et Réseaux

Niveau : ING1

Année : 2024/2025

Tables des matières

1. Introduction

L'objectif de ce TP est de revoir les bases de la programmation en C en manipulant des vecteurs. Nous avons implémenté plusieurs fonctions permettant de manipuler des vecteurs de taille fixée, notamment leur affichage, leur copie, le calcul du produit scalaire ainsi que diverses mesures statistiques.

2. Description des Fonctions Implémentées

2.1 Affichage d'un vecteur

La fonction `printVector(double v[])` permet d'afficher un vecteur à l'écran sous forme de liste de valeurs séparées par des espaces.

2.2 Calcul du produit scalaire

La fonction `dot(double v1[], double v2[])` calcule le produit scalaire de deux vecteurs en effectuant la somme des produits des éléments correspondants.

2.3 Copie d'un vecteur

La fonction `copyVector(double dest[], double src[])` copie les valeurs d'un vecteur source `src` vers un vecteur destination `dest`.

2.4 Calculs statistiques

Nous avons implémenté plusieurs fonctions pour effectuer des calculs statistiques sur un vecteur :

- `moyenne(double v[])` : Calcule la moyenne des éléments du vecteur.
- `standardDeviation(double v[])` : Calcule l'écart type des valeurs du vecteur.
- `minValue(double v[])` : Trouve la plus petite valeur du vecteur.
- `maxValue(double v[])` : Trouve la plus grande valeur du vecteur.
- `median(double v[])` : Calcule la médiane du vecteur en utilisant `qsort` pour trier les valeurs.

2.5 Tri et calcul de la médiane

La fonction `median(double v[])` utilise `qsort` pour trier un vecteur avant de calculer sa médiane. Une fonction de comparaison `compare(const void* a, const void* b)` est utilisée pour le tri.

```c
/* Assane Thiao 28/02/2025 */
#include <stdio.h>   /* Pour l'affichage avec printf */
#include <stdlib.h>  /* Pour utiliser qsort */
#include <math.h>    /* Pour sqrt (racine carrée) */

#define SIZE 10  /* Taille des vecteurs fixée à 10 */

/* Déclaration des fonctions */
void printVector(double v[]);
double dot(double v1[], double v2[]);
void copyVector(double dest[], double src[]);
double moyenne(double v[]);
double standardDeviation(double v[]);
double minValue(double v[]);
double maxValue(double v[]);
double median(double v[]);
int compare(const void* a, const void* b);


int main() {
    /* Déclaration et initialisation de deux vecteurs */
    double v1[SIZE] = { -0.45,0.78,-0.12,0.99,-0.67,0.34,-0.89,0.21,-0.56,0.73 };
    double v2[SIZE] = { 0.62, -0.87, 0.15, -0.94, 0.48, -0.33, 0.79, -0.21, 0.56, -0.68 };
    double vCopy[SIZE];  /* Pour tester la copie de vecteur */

    /* Affichage des vecteurs */
    printf("Vecteur v1:\n");
    printVector(v1);
    printf("Vecteur v2:\n");
    printVector(v2);

    /* Calcul et affichage du produit scalaire */
    printf("Produit scalaire: %.2f\n", dot(v1, v2));

    /* Copie de v1 dans vCopy et affichage du résultat */
    copyVector(vCopy, v1);
    printf("Copie de v1 dans vCopy:\n");
    printVector(vCopy);

    /* Calcul et affichage des statistiques */
    printf("Moyenne de v1: %.2f\n", moyenne(v1));
    printf("Écart type de v1: %.2f\n", standardDeviation(v1));
    printf("Valeur minimale de v1: %.2f\n", minValue(v1));
    printf("Valeur maximale de v1: %.2f\n", maxValue(v1));
    printf("Médiane de v1: %.2f\n", median(v1));

    return 0;
}
```

```c
/* Fonction pour afficher un vecteur */
void printVector(double v[]) {
    for (int i = 0; i < SIZE; i++) {
        printf("%.2f ", v[i]);
    }
    printf("\n");
}

/* Fonction pour calculer le produit scalaire de deux vecteurs */
double dot(double v1[], double v2[]) {
    double resultat = 0.0;
    for (int i = 0; i < SIZE; i++) {
        resultat += v1[i] * v2[i];
    }
    return resultat;
}

/* Fonction pour copier un vecteur dans un autre */
void copyVector(double dest[], double src[]) {
    for (int i = 0; i < SIZE; i++) {
        dest[i] = src[i];
    }
}

/* Fonction pour calculer la moyenne d'un vecteur */
double moyenne(double v[]) {
    double somme = 0.0;
    for (int i = 0; i < SIZE; i++) {
        somme += v[i];
    }
    return somme / SIZE;
}

/* Fonction pour calculer la déviation standard, l'écart type d'un vecteur */
double standardDeviation(double v[]) {
    double moy = moyenne(v);
    double somme = 0.0;
    for (int i = 0; i < SIZE; i++) {
        somme += (v[i] - moy) * (v[i] - moy);
    }
    return sqrt(somme / SIZE);
}

/* Fonction pour trouver la plus petite valeur d'un vecteur */
double minValue(double v[]) {
    double min = v[0];
    for (int i = 1; i < SIZE; i++) {
        if (v[i] < min) min = v[i];
    }
    return min;
}
```

```c
/* Fonction pour trouver la plus grande valeur d'un vecteur */
double maxValue(double v[]) {
    double max = v[0];
    for (int i = 1; i < SIZE; i++) {
        if (v[i] > max) max = v[i];
    }
    return max;
}

/* Fonction pour calculer la médiane d'un vecteur */
double median(double v[]) {
    double aTrier[SIZE];
    copyVector(aTrier, v);  /* Copie du vecteur pour éviter de modifier l'original */
    qsort(aTrier, SIZE, sizeof(double), compare);  /* Tri du vecteur */
    if (SIZE % 2 == 0) {
        return (aTrier[SIZE / 2 - 1] + aTrier[SIZE / 2]) / 2.0;  /* Moyenne des deux valeurs centrales */
    }
    else {
        return aTrier[SIZE / 2];  /* Valeur centrale */
    }
}
/* Fonction de comparaison pour le tri */
int compare(const void* a, const void* b) {
    return (*(double*)a - *(double*)b);
}
```

3. Tests et Résultats

Pour tester les fonctions, nous avons déclaré et initialisé deux vecteurs v1 et v2, puis appelé les différentes fonctions pour vérifier leur bon fonctionnement. Voici les résultats obtenus :

1. **Affichage des vecteurs**
    a. `printVector(v1)`: Affichage correct du contenu du vecteur v1
    b. `printVector(v2)`: Affichage correct du contenu du vecteur v2
2. **Calcul du produit scalaire**
    a. `dot(v1, v2)`: Calcul correct du produit scalaire.
3. **Copie de vecteur**
    a. `copyVector(vCopy, v1)`: Vérification que vCopy est identique à v1.
4. **Calculs statistiques**
    a. `moyenne(v1)`: Valeur correcte.
    b. `standardDeviation(v1)`: Valeur correcte.
    c. `minValue(v1)`: Valeur correcte.
    d. `maxValue(v1)`: Valeur correcte.
    e. `median(v1)`: Valeur correcte.

4. Conclusion

Ce TP a permis de renforcer la maîtrise des tableaux en C et des fonctions associées. Nous avons appliqué des concepts fondamentaux tels que l'utilisation des boucles, le passage de tableaux en paramètre et l'utilisation de fonctions prédéfinies comme `qsort` et `sqrt`.

Les fonctions implémentées ont été testées et validées avec succès, garantissant leur bon fonctionnement.