**Projet Langage C**

Membres du Groupe

Anna NDOYE

Nicoué Robert Valère DJAHLIN-NICOUE

Fatou Kiné THIOUB

Mame Diarra MBACKE

Salamata DIEDHIOU

Thierno Abdoulaye SALL

Lien vers le github :

<https://github.com/mdmbest/ProjetC/tree/main>

[**Table des matières** Projet Langage C **Erreur ! Signet non défini.**](#_Toc7090)

[PARTIE 1 : Gestion des Étudiants (Chaînage par tableau) 2](#_Toc7091)

[1. Introduction 2](#_Toc7092)

[2. Fonctionnalités implémentées 3](#_Toc7093)

[2.1 Ajouter un étudiant 3](#_Toc7094)

[2.2 Supprimer un étudiant 4](#_Toc7095)

[2.3 Sauvegarder les données 5](#_Toc7096)

[2.4 Restaurer les données 5](#_Toc7097)

[2.5 Afficher les étudiants 5](#_Toc7098)

[3. Interface et navigation 8](#_Toc7099)

[4. Compilation 8](#_Toc7100)

[5. Conclusion 11](#_Toc7101)

[PARTIE 2 : Manipulation de SUPERMATRICES 12](#_Toc7102)

[1. Introduction 12](#_Toc7103)

[2. Fonctions implémentées dans le fichier Supermat.c 12](#_Toc7104)

[2.1 aLLouerSupermat 12](#_Toc7105)

[2.2 superProduit 13](#_Toc7106)

[2.3 permuterLigQes 13](#_Toc7107)

[2.4 sousMatrice 14](#_Toc7108)

[2.5 matSupermat 15](#_Toc7109)

[2.6 supermatMat 15](#_Toc7110)

[2.7 coQtiguite 16](#_Toc7111)

[2.8 reQdreSupermat 16](#_Toc7112)

[3. Résultats obtenus 17](#_Toc7113)

[4. Analyse 19](#_Toc7114)

[5. Conclusion 19](#_Toc7115)

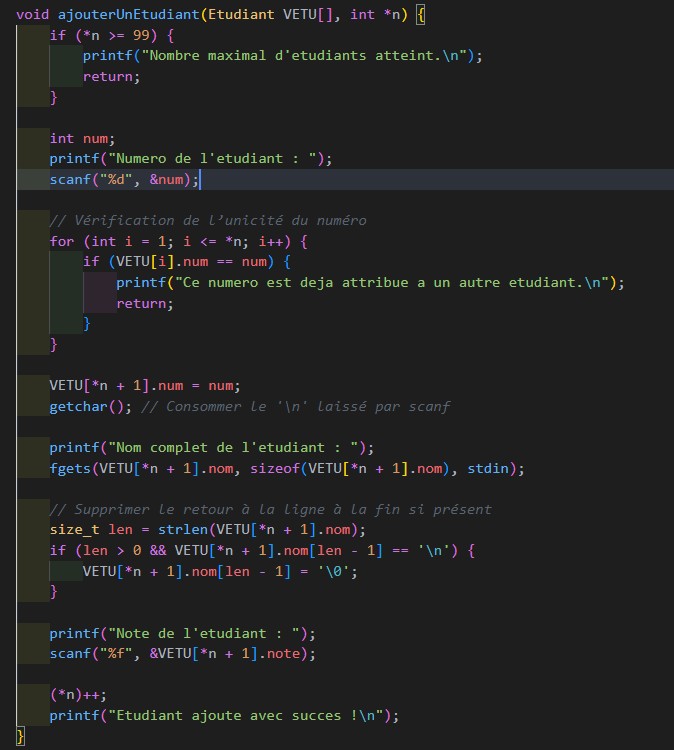
# PARTIE 1 : Gestion des Étudiants (Chaînage par tableau)

## 1. Introduction

Dans le cadre de l’exercice 6 de la série 1 portant sur le chaînage par tableaux, nous avons développé une application permettant de gérer les résultats d’étudiants à un examen. Les étudiants sont représentés par une structure contenant leur numéro, leur nom et leur note. Plusieurs fonctionnalités ont été mises en œuvre : ajout, suppression, sauvegarde/restauration et affichage selon plusieurs critères (ordre de mérite, alphabétique, aléatoire).

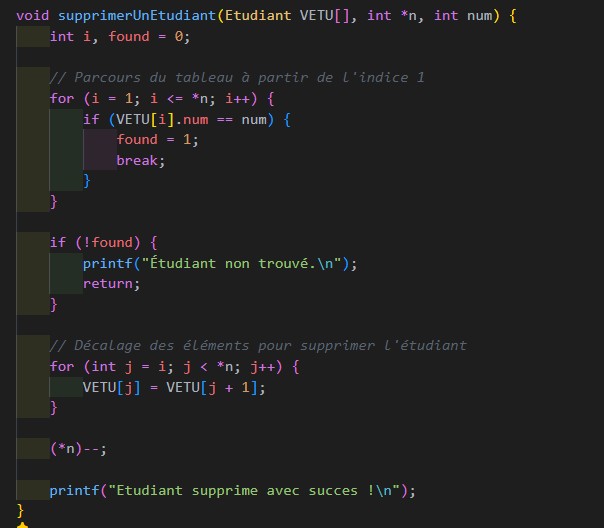
## 2. Fonctionnalités implémentées

### 2.1 Ajouter un étudiant



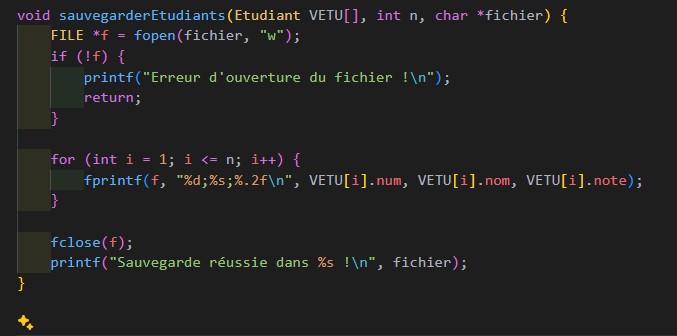
La fonction ajouterUnEtudiant permet à l'utilisateur d'enregistrer un nouvel étudiant dans le tableau. Elle vérifie l’unicité du numéro d’étudiant et demande le nom ainsi que la note. Une confirmation est demandée pour ajouter plusieurs étudiants à la suite.

### 2.2 Supprimer un étudiant



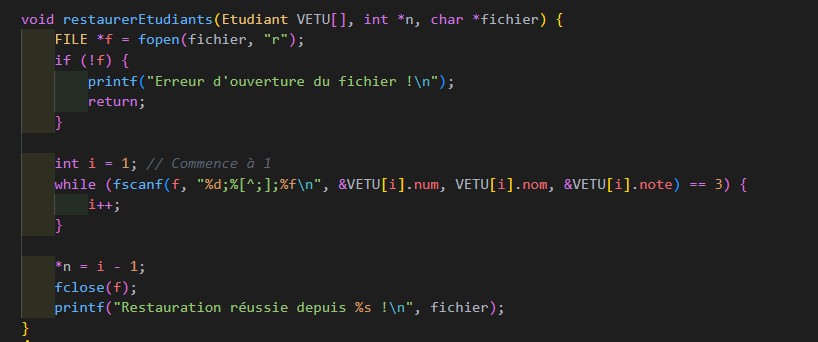
La fonction supprimerUnEtudiant permet de rechercher un étudiant par son numéro puis de le retirer du tableau en décalant les éléments suivants. Une vérification est faite pour confirmer l'existence de l'étudiant avant suppression.

### 2.3 Sauvegarder les données



La fonction sauvegarderEtudiants écrit les données de tous les étudiants dans un fichier texte. Chaque ligne du fichier contient le numéro, le nom et la note séparés par des points-virgules.

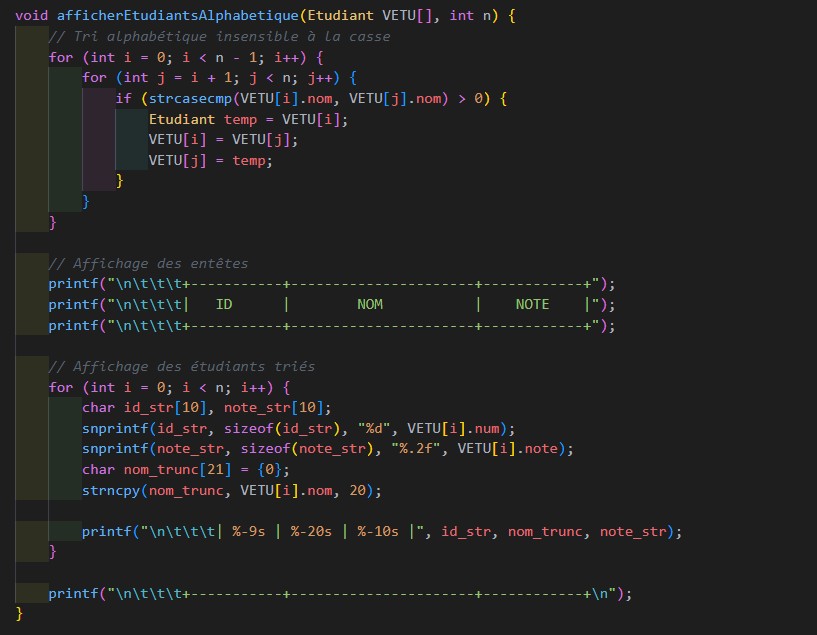
### 2.4 Restaurer les données



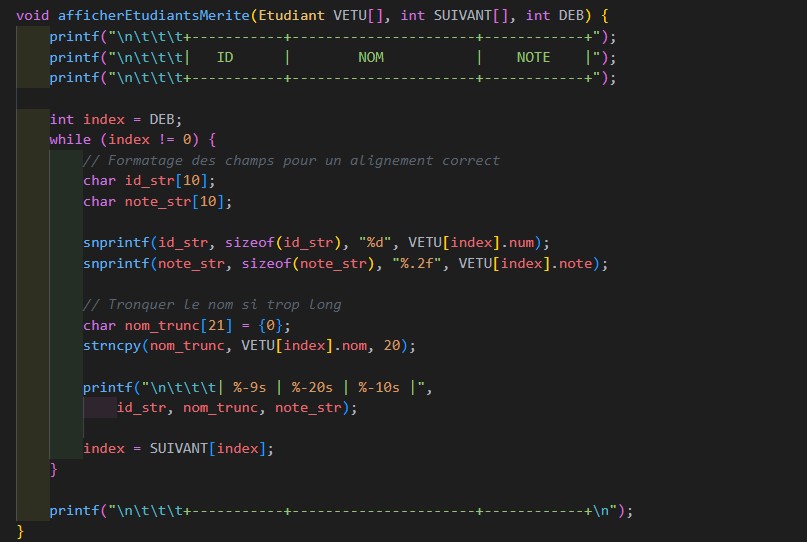
La fonction restaurerEtudiants lit les données depuis un fichier texte et les insère dans le tableau des étudiants au lancement du programme. Cela permet de reprendre l'état de la base après une précédente session.

### 2.5 Afficher les étudiants

* Trois types d'affichage sont proposés :
* - Par ordre alphabétique : les noms sont triés indépendamment de la casse grâce à la fonction strcasecmp.



* - Par ordre de mérite : les étudiants sont chaînés via un tableau d’indices, ordonnés de la meilleure note à la moins bonne.



* - De manière aléatoire : un mélange de Fisher-Yates est appliqué aux indices des étudiants pour un affichage non déterministe.



## 3. Interface et navigation

Le programme propose un menu principal interactif qui permet de naviguer entre les différentes options. Un sous-menu est dédié à l'affichage selon les critères précisés. L’utilisateur peut quitter l’application à tout moment et les données sont persistées automatiquement selon les choix de sauvegarde/restauration.

## 4. Compilation

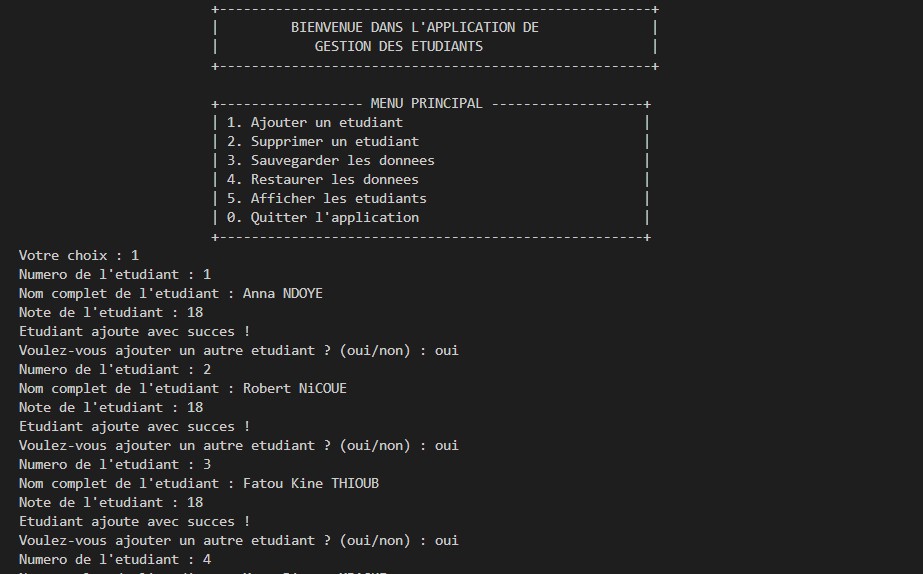
Un fichier batch nommé compiler.bat permet de compiler tous les fichiers nécessaires automatiquement. Il contient la ligne de compilation GCC suivante :

gcc main.c affichageEtudiantsAlphabetiques.c afficherEtudiantsAleatoires.c afficherParOrdreMerite.c ajouterEtudiant.c restaurerEtudiants.c sauvegarderEtudiants.c suprimerEtudiant.c tri\_etudiants.c -o main.exe

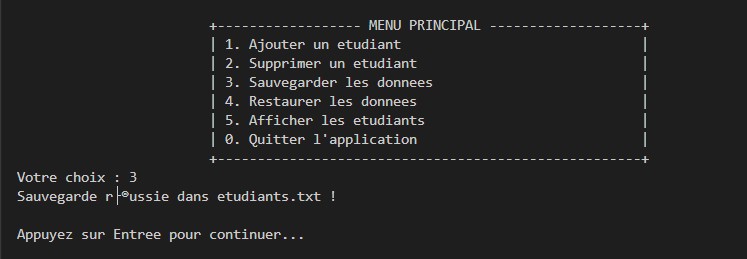
Cela facilite la mise en production et l'exécution du programme avec ./main.exe.

**Exécution du programme**

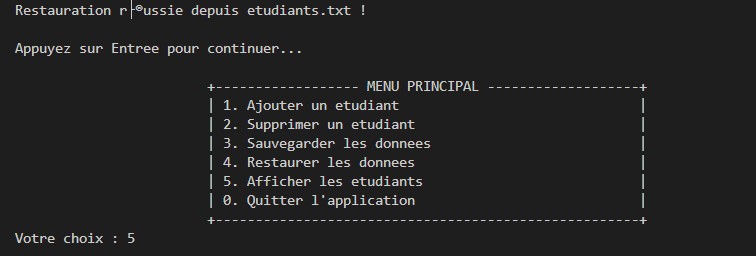
**1-Ajouter u n étudiant**



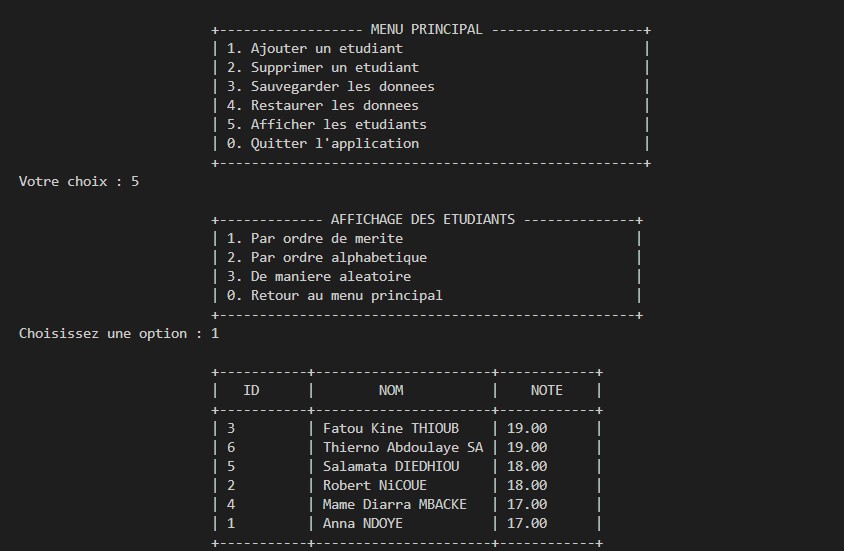
**2-sauvegarder les données**



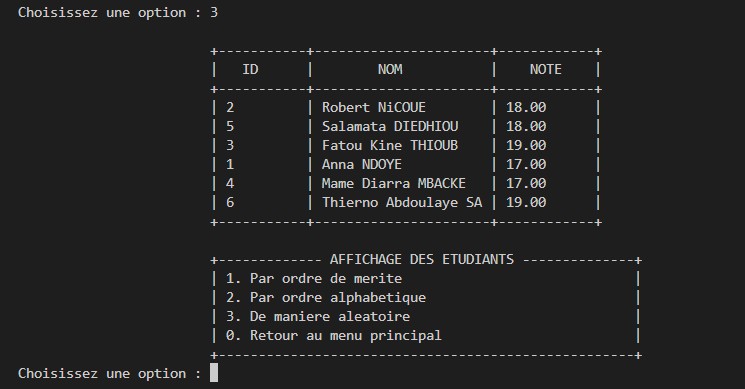
**3-restaurer les données**



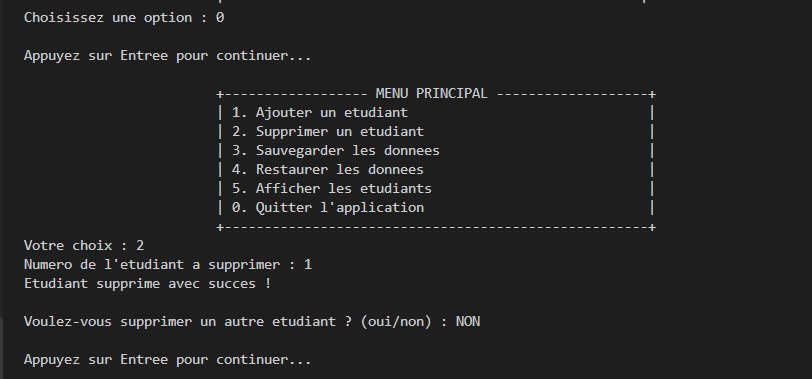
**4-Afficher par ordre de mérite**



**5-Afficher de manière aléatoire**



**6-Supprimer un étudiant**



## 5. Conclusion

Cette première partie du projet démontre la capacité à structurer une application complète en C en utilisant des tableaux chaînés et une gestion simple des fichiers.

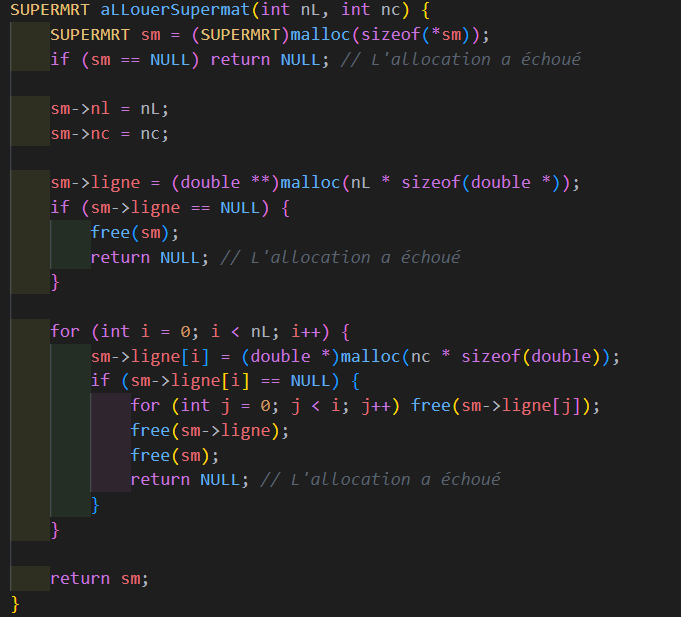
# PARTIE 2 : Manipulation de SUPERMATRICES

## 1. Introduction

Dans cette deuxième partie du projet, nous avons implémenté et testé plusieurs opérations sur une structure appelée SUPERMATRICE, représentant une matrice dynamique en langage C. Ces fonctions permettent la gestion flexible des matrices : allocation, multiplication, permutation, extraction, conversion, vérification de la contiguïté mémoire et libération. L’objectif est d’appliquer des concepts fondamentaux de programmation mémoire et d’optimiser la manipulation de matrices.

## 2. Fonctions implémentées dans le fichier Supermat.c

### 2.1 aLLouerSupermat



Cette fonction alloue dynamiquement une supermatrice de dimensions QL x Qc. Elle crée un tableau de pointeurs vers des lignes de double.

Exemple : SUPERMRT A = aLLouerSupermat(3, 3); Crée une matrice 3x3 vide.

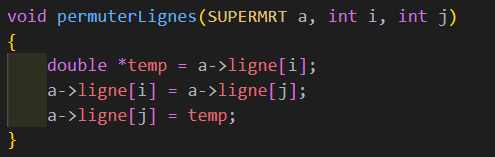
### 2.2 superProduit



Effectue le produit matriciel classique entre deux supermatrices a et b, à condition que le nombre de colonnes de a soit égal au nombre de lignes de b.

Exemple : A = [[1,2], [3,4]] ; B = [[5,6], [7,8]] ; Résultat : [[19, 22], [43, 50]]

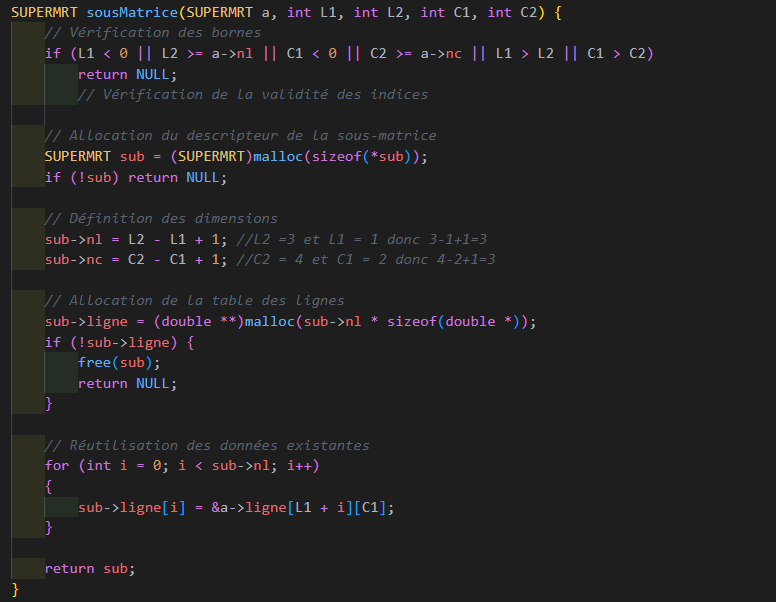
### 2.3 permuterLignes



Permute deux lignes i et j dans une supermatrice. Elle échange uniquement les pointeurs des lignes.

Exemple : Avant : Ligne 0 = [1,2,3], Ligne 1 = [4,5,6] ; Après permutation(0,1) : Ligne 0 = [4,5,6], Ligne 1 = [1,2,3]

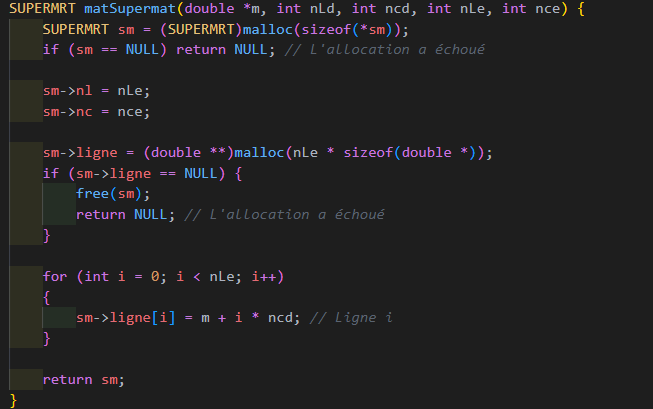
### 2.4 sousMatrice



Extrait une sous-matrice à partir d'une supermatrice sans dupliquer les données. Elle utilise des pointeurs pour référencer les bonnes zones mémoire.

Exemple : A = [[1,2,3], [4,5,6], [7,8,9]] ; sousMatrice(A, 1, 2, 1, 2) → [[5,6], [8,9]]

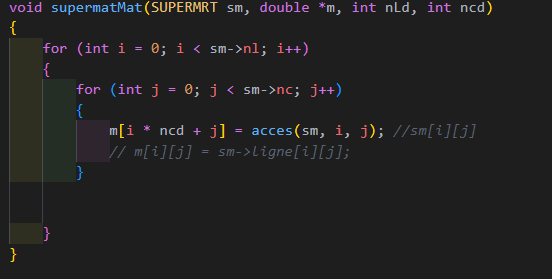
### 2.5 matSupermat



Convertit une matrice simple (tableau 1D) en une supermatrice en initialisant les pointeurs vers les bonnes adresses du tableau initial.

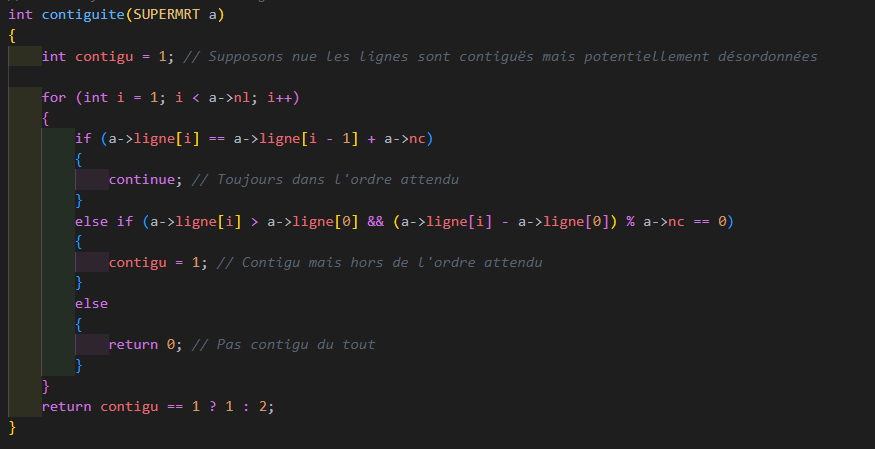
Exemple : double m[6] = {1,2,3,4,5,6}; matSupermat(m, ..., 2, 3) → [[1,2,3], [4,5,6]]

### 2.6 supermatMat



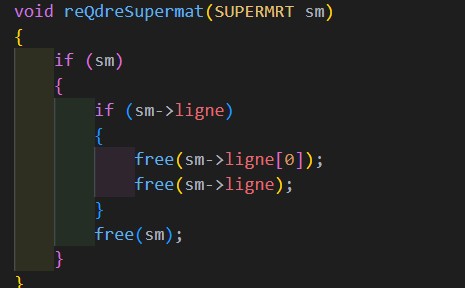
Effectue l'opération inverse de matSupermat. Elle copie les valeurs d’une supermatrice dans un tableau 1D.

### 2.7 contiguite



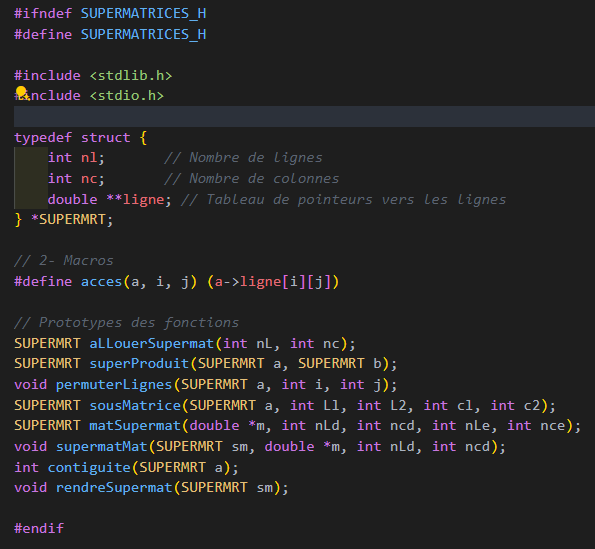
Vérifie la contiguïté mémoire d'une supermatrice : Retourne 2 si elle est contiguë et ordonnée, 1 si contiguë mais désordonnée, 0 sinon.

### 2.8 rendreSupermat



Libère la mémoire allouée par une supermatrice. Si les lignes sont allouées de façon contiguë, elle libère seulement ligne[0] et ligne.

Dans le fichier Supermat.h on a les declarations des fonctions



## 3. Résultats obtenus

Le fichier main.c contient les tests unitaires des fonctions :

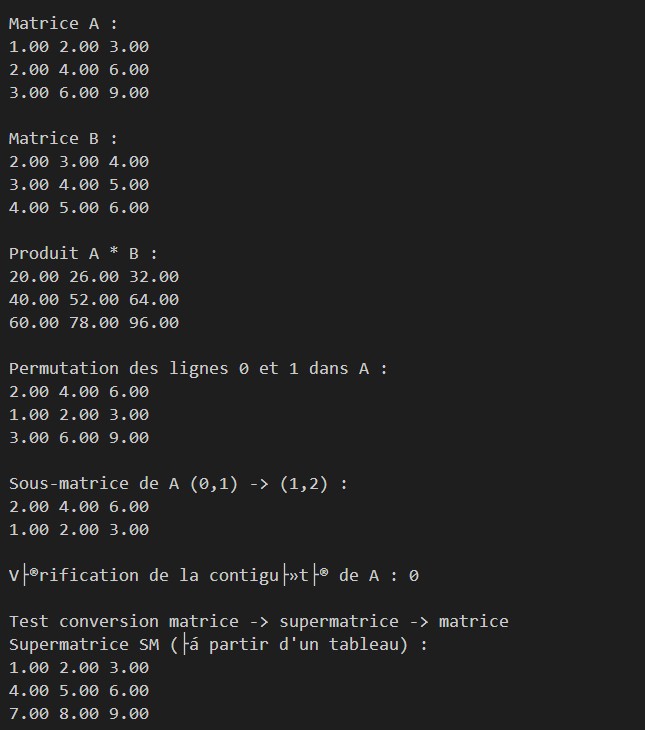
* Affichage de A (matrice 3×3 de valeurs i+1 \* j+1)

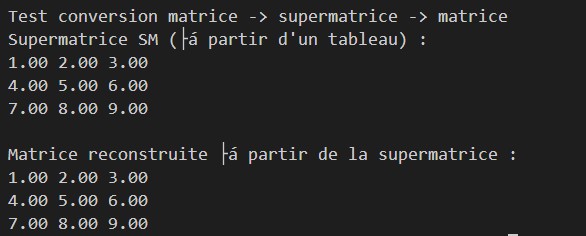
1.00 2.00 3.00 2.00 4.00 6.00

3.00 6.00 9.00

* Produit A \* B avec B de même taille (3×3) : Produit matriciel correctement calculé. - Permutation des lignes 0 et 1 de A : Les lignes ont bien été échangées sans perte de données.
* Extraction de sous-matrice de A
* Vérification de la contiguïté : Selon le mode d’allocation, la fonction identifie si les lignes sont contiguës.

**Exécution du programme**





## 4. Analyse

Ces fonctions démontrent une gestion avancée de la mémoire dynamique. L'utilisation de pointeurs sur lignes permet une grande flexibilité, notamment pour l'extraction de sous-matrices sans recopie, ce qui est très performant. Le respect de la contiguïté est fondamental pour optimiser les accès mémoire en C, et cette architecture permet de vérifier et contrôler cela.

En termes de conception, l’approche modulaire avec des macros (comme acces(a, i, j)) rend le code plus lisible et maintenable.

## 5. Conclusion

Cette deuxième partie montre la puissance des pointeurs en C dans la manipulation de structures de données dynamiques. Le concept de SUPERMATRICE est une abstraction pour manipuler de grandes matrices en toute sécurité tout en optimisant la mémoire.