

## Application de Gestion des Étudiants en Langage C

Sciences Mathématiques et Informatiques  
(SMI)

**Faire une application de gestion des étudiants  
En langage C avancée**

**Réalisé par :**

- AICH MOHAMMED
- TRISSI AHMED

**Encadré par :**

- Prof : R. HANNANE

Écrit le 22/04/2024 en FSSM

## Tableau de matières :

I.	Introduction.....	4
II.	Résumé.....	4
III.	Conception et Implémentation .....	4
IV.	Fonctionnalités.....	6
1.	Ajout.....	6
2.	Consultation .....	6
•	Liste de Tous les étudiants .....	6
•	Liste des étudiants Admis.....	6
•	Liste des étudiants par Filière .....	6
3.	Recherche d'étudiants.....	6
•	Par Numéro Apogée .....	6
•	Par Nom, Prénom ou Date d'Inscription .....	6
4.	Modification des Informations.....	6
5.	Tri des étudiants.....	6
6.	Suppression d'étudiants.....	7
7.	Enregistrer les modifications.....	7
V.	Explication de fonctionnalité.....	7
1.	Main.....	7
i.	Inclusions de Fichiers d'En-tête .....	7
ii.	Déclaration des Variables .....	7
iii.	Chargement des Étudiants depuis un Fichier.....	7
iv.	Affichage du Menu Principal.....	7
v.	Enregistrement des Étudiants dans un Fichier.....	7
vi.	Fin de l'Exécution .....	7
2.	Ajout d'étudiants .....	8
3.	Consultation .....	9
4.	Recherche d'étudiants.....	9
5.	Modification des informations.....	10
6.	Trie des étudiants.....	11
7.	Suppression d'étudiants.....	12
8.	Sortie .....	12
VI.	Conclusion .....	13

## Les figure :

Figure 1 : Structure des étudiants.....	5
Figure 2 : Menu principale.....	6
Figure 3 : Main.....	8
Figure 4 : Affichage .....	9
Figure 5 : Recherche .....	10
Figure 6 : Modification .....	11
Figure 7 : trie .....	12
Figure 8 : Menu de sortie .....	13

## I. Introduction :

Ce projet de développement d'une application de gestion des étudiants en langage C a été une expérience enrichissante dans l'apprentissage de la programmation avancée. L'objectif principal était de concevoir une application console permettant de simplifier la gestion des informations des étudiants, en mettant en œuvre des concepts tels que la programmation modulaire, les structures de données et la manipulation de fichiers.

L'application offre un ensemble de fonctionnalités essentielles, notamment l'ajout, la modification, la consultation et la suppression des données des étudiants. Elle permet également de trier les étudiants par différents critères et de rechercher des étudiants spécifiques. L'utilisation de fichiers pour stocker les données garantit la persistance des informations entre les différentes sessions d'exécution de l'application.

Ce projet a permis d'approfondir notre compréhension de la programmation en langage C, en mettant en pratique les concepts théoriques enseignés en cours. Il a également renforcé nos compétences en matière de conception logicielle, de développement modulaire et de tests unitaires. En conclusion, ce projet a été une étape importante dans notre apprentissage de la programmation et a contribué à notre développement en tant que programmeurs.

## II. Résumé :

L'objectif principal de ce projet est de concevoir et de développer une application de gestion des étudiants en langage C, en mettant en pratique les concepts et les techniques appris dans le cadre du module de Programmation 2(Programmation C Avancée). Les objectifs spécifiques incluent :

- Mettre en œuvre la programmation modulaire pour organiser efficacement le code source de l'application.
- Utiliser des structures de données pour représenter les informations des étudiants de manière structurée et efficace.
- Implémenter un ensemble de fonctionnalités permettant d'ajouter, de modifier, de consulter et de supprimer des étudiants.
- Utiliser des fichiers pour stocker et récupérer les données des étudiants, assurant ainsi la persistance des données entre les différentes exécutions de l'application.
- Fournir une interface utilisateur claire et intuitive, permettant aux utilisateurs d'interagir facilement avec l'application.
- Tester rigoureusement l'application pour garantir son bon fonctionnement et sa fiabilité dans différents scénarios d'utilisation.

En réalisant ce projet, l'objectif est d'acquérir une expérience pratique significative dans le développement logiciel en langage C, tout en renforçant notre compréhension des concepts fondamentaux de la programmation.

### III. Conception et Implémentation :

Le projet de gestion des étudiants a été conçu avec une approche modulaire afin d'assurer une organisation efficace du code source et une meilleure maintenabilité du projet. Voici un aperçu de la conception et de l'implémentation de chaque module :

**Module Etudiant** : Ce module est responsable de la gestion des opérations sur les étudiants, y compris l'ajout, la modification, la recherche, la suppression, la consultation et le tri. Il contient la structure **Etudiant** qui représente les informations d'un étudiant, ainsi que les fonctions associées pour effectuer ces opérations.

```
struct Etudiant {  
    int num_apo;  
    char nom[MAX_STRINGS];  
    char prenom[MAX_STRINGS];  
    Date date_naissance;  
    Filiere filiere;  
    Module modules[MAX_MODULES];  
    int nbr_modules;  
    float moyenne;  
    Date date_inscription;  
};
```

*Figure 1 : Structure des étudiants.*

**Module Menu** : Ce module gère l'affichage du menu principal de l'application et la récupération des choix de l'utilisateur. Il offre une interface conviviale pour permettre à l'utilisateur d'interagir avec les fonctionnalités de l'application.

Ce module utilise des boucles de menu pour afficher les options disponibles et récupérer le choix de l'utilisateur.

Prototype : menu (etudiants, &nbr\_etudiants\_actu) ;

```
Menu:
1. Ajouter un etudiant
2. Consulter les etudiants
3. Rechercher un etudiant
4. Modifier un etudiant
5. Trier les etudiants
6. Supprimer un etudiant
7. Quitter
Votre choix: |
```

*Figure 2 : Menu principale.*

Cette conception modulaire permet une séparation claire des responsabilités et facilite la collaboration entre les membres de l'équipe de développement. Chaque module est indépendant et peut être testé individuellement, ce qui améliore la qualité du code et la fiabilité de l'application dans son ensemble.

## IV. Fonctionnalités :

Les fonctionnalités principales de l'application de gestion des étudiants en langage C :

1. Ajout d'étudiants : Permet d'ajouter de nouveaux étudiants à la base de données en fournissant leurs informations telles que le nom, le prénom, le numéro apogée, la filière, etc.
2. Consultation :
  - Liste de Tous les étudiants : Affiche la liste complète de tous les étudiants enregistrés dans la base de données.
  - Liste des étudiants Admis : Affiche uniquement les étudiants ayant une moyenne supérieure à un seuil défini, indiquant ainsi ceux qui sont admis.
  - Liste des étudiants par Filière : Permet de consulter les étudiants appartenant à une filière spécifique, filtrant ainsi la liste par filière.
3. Recherche d'étudiants :
  - Par Numéro Apogée : Permet de rechercher un étudiant spécifique en saisissant son numéro apogée.
  - Par Nom, Prénom ou Date d'Inscription : Permet de trouver tous les étudiants ayant un nom, un prénom ou une date d'inscription spécifiés.
4. Modification des Informations : Permet de modifier les informations d'un étudiant existant en fonction de son numéro apogée. L'utilisateur peut mettre à jour les détails tels que le nom, le prénom, la filière, les notes.
5. Tri des étudiants : Permet de trier les étudiants selon différents critères tels que le numéro apogée, la moyenne ou la date d'inscription. Cette fonctionnalité facilite la recherche et l'organisation des données.

8. Suppression d'étudiants : Permet de supprimer un étudiant de la base de données en fonction de son numéro apogée. Cette fonctionnalité offre la possibilité de gérer efficacement la liste des étudiants enregistrés.
9. Enregistrer les modifications : Pour la deuxième partie qui contient un fichier texte pour l'enregistrement des informations des étudiants, elle contient un autre fichier nommé fichiers.c Ce module est chargé de la lecture et de l'écriture des données des étudiants à partir et vers un fichier. Il garantit la persistance des données entre les différentes exécutions de l'application.

Prototype : `lireEtudiantsDeFichier(etudiants, &nbr_etudiants_actu);`

Ces fonctionnalités offrent à l'utilisateur un ensemble complet d'outils pour gérer efficacement les informations des étudiants, simplifiant ainsi les tâches administratives liées à la gestion des étudiants.

## V. Explication de fonctionnalité :

### 1. Main :

Ce fichier main du projet est le point d'entrée de l'application. Voici une description détaillée de son fonctionnement :

- i. Inclusions de Fichiers d'En-tête : Le programme commence par inclure les fichiers d'en-tête nécessaires. Ces fichiers contiennent les déclarations de fonctions, les définitions de structures et les constantes utilisées dans le programme. Dans cet exemple, les fichiers inclus sont main-header.h, ajout.h, menu.h et fichiers.h.
- ii. Déclaration des Variables : Le main commence par déclarer quelques variables :
  - Sauvegarde : Une variable pour suivre si une sauvegarde des données des étudiants est nécessaire après l'exécution du programme.
  - etudiants: Un tableau de structures Etudiant pour stocker les données des étudiants.
  - nbr\_etudiants\_actu: Un entier pour suivre le nombre d'étudiants actuellement enregistrés.
- iii. Chargement des Étudiants depuis un Fichier : La fonction `lireEtudiantsDeFichier` est appelée pour charger les étudiants à partir d'un fichier dans le tableau `etudiants`. Le nombre d'étudiants chargés est mis à jour dans `nbr_etudiants_actu`.
- iv. Affichage du Menu Principal : La fonction `menu` est appelée pour afficher le menu principal de l'application et permettre à l'utilisateur d'interagir avec les différentes fonctionnalités.
- v. Enregistrement des Étudiants dans un Fichier : Après que l'utilisateur ait terminé ses actions dans le menu et que la variable sauvegarde a été mise à 1 (indiquant que des modifications ont été apportées aux données des étudiants), la fonction `enregistrerEtudiantsDansFichier` est appelée pour sauvegarder les données des étudiants dans un fichier.
- vi. Fin de l'Exécution : Une fois que toutes les opérations sont terminées, le programme se termine avec `return 0`.

En résumé, ce fichier main orchestre le fonctionnement de l'application en chargeant les données des étudiants, en affichant le menu principal, en permettant à l'utilisateur d'effectuer des actions, en sauvegardant les modifications et en terminant l'exécution.

```
#include "main-header.h"
#include "ajout.h"
#include "menu.h"
#include "fichiers.h"

int main() {
    int sauvegarde = 0;
    Etudiant etudiants[MAX_ETUDIANTS];
    int nbr_etudiants_actu = 0;
    lireEtudiantsDeFichier(etudiants, &nbr_etudiants_actu);
    sauvegarde= menu(etudiants, &nbr_etudiants_actu);

    if (sauvegarde == 1)enregistrerEtudiantsDansFichier(etudiants, nbr_etudiants_actu);

    return 0;
}
```

Figure 3 : Main

## 2. Ajout d'étudiants

Pour développer la fonctionnalité d'ajout d'étudiants en fournissant leurs informations telles que le nom, le prénom, le numéro apogée, la filière, etc., nous allons créer une fonction `ajouter_etudiant` qui permettra à l'utilisateur de saisir ces informations et d'ajouter un nouvel étudiant à la base de données.

Prototype : `Etudiant ajout_etudiant()` ;

Et : `void ajout_n_etudiants(Etudiant etudiants[], int *nbr_etudiants_actu)`

Dans cette implémentation :

- La fonction `ajouter_n_etudiant` permet à l'utilisateur de saisir les informations des étudiant à ajouter.
- Elle vérifie d'abord si le nombre maximal d'étudiants est atteint.
- Ensuite, elle saisit les informations de l'étudiant à partir de l'entrée utilisateur.
- Elle ajoute ensuite cet étudiant à la liste et met à jour le nombre total d'étudiants.

Dans la fonction `main`, nous créons un tableau d'étudiants pour stocker les étudiants et appelons la fonction `ajouter_etudiant` pour ajouter un nouvel étudiant à la liste.



### 3. Consultation :

Pour développer la fonctionnalité de consultation des étudiants, nous devons créer trois fonctions distinctes pour chaque sous-fonctionnalité spécifiée :

- ❖ Liste de Tous les Étudiants : Cette fonction affichera simplement la liste complète de tous les étudiants enregistrés dans la base de données. Elle parcourra la liste des étudiants et affichera les détails de chaque étudiant.

Prototype : `void afficherNEtudiants(Etudiant etudiants[], int nbr_etudiants_actu) ;`

- ❖ Liste des Étudiants Admis : Cette fonction filtrera la liste des étudiants pour n'afficher que ceux ayant une moyenne supérieure à un seuil défini, indiquant ainsi ceux qui sont admis. Elle comparera la moyenne de chaque étudiant avec le seuil spécifié et n'affichera que ceux qui la dépassent.

Prototype : `void afficherEtudiantsAdmisSeulement(Etudiant etudiants[], int nbr_etudiants_actu) ;`

- ❖ Liste des Étudiants par Filière : Cette fonction permettra à l'utilisateur de spécifier une filière, puis elle affichera uniquement les étudiants appartenant à cette filière. Elle parcourra la liste des étudiants et n'affichera que ceux appartenant à la filière spécifiée.

Prototype : `void AfficherEtudiantsFiliere(Etudiant etudiants[], int nbr_etudiants_actu) ;`

Ces fonctions peuvent être mises en œuvre de manière modulaire, en séparant la logique de traitement des données de la logique d'affichage. Une fois que les données pertinentes sont récupérées, elles peuvent être affichées dans la fonction principale de consultation.

```
1. Consulter tous les etudiants
2. Consulter les etudiants admis seulement
3. Consulter les etudiants par filière
Votre choix:
```

Figure 4 : Affichage

### 4. Recherche d'étudiants

Pour développer la fonctionnalité de recherche d'étudiants, nous devons créer des fonctions permettant de rechercher des étudiants en fonction de différents critères.

- ❖ Recherche par Numéro Apogée : Cette fonctionnalité permettra à l'utilisateur de rechercher un étudiant spécifique en saisissant son numéro apogée. La fonction vérifiera chaque étudiant dans la base de données et renverra celui correspondant au numéro apogée spécifié, le cas échéant.

Prototype : `int rechercherEtudiantApogee(Etudiant etudiants[], int nbr_etudiants_actu, int num_apo) ;`

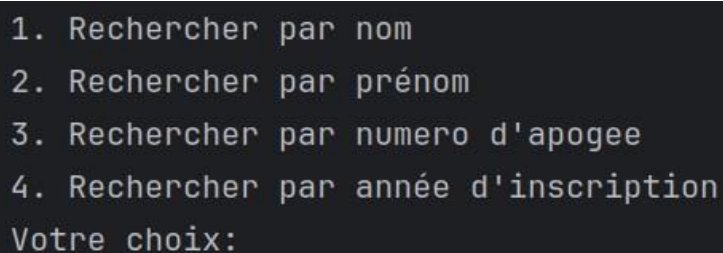
- ❖ Recherche par Nom, Prénom ou Date d'Inscription : Cette fonctionnalité permettra à l'utilisateur de rechercher tous les étudiants ayant un nom, un prénom ou une date d'inscription spécifiée. La fonction parcourra la liste des étudiants et renverra tous ceux correspondant au critère de recherche.

Prototype pour nom : `void rechercherEtudiantNom(Etudiant etudiants[], int nbr_etudiants_actu) ;`

Prototype pour prénom : `void rechercherEtudiantPrenom(Etudiant etudiants[], int nbr_etudiants_actu) ;`

Prototype pour date d'inscription : `void rechercherEtudiantAnneeInscription(Etudiant etudiants[], int nbr_etudiants_actu) ;`

Chaque fonction de recherche devrait être conçue de manière à parcourir efficacement la liste des étudiants et à renvoyer les résultats correspondants. Elle devrait également être accompagnée d'une fonction d'affichage appropriée pour présenter les résultats de manière claire à l'utilisateur.



```
1. Rechercher par nom
2. Rechercher par prénom
3. Rechercher par numero d'apogee
4. Rechercher par année d'inscription
Votre choix:
```

*Figure 5 : Recherche*

## 5. Modification des Informations

La fonctionnalité de modification des informations des étudiants permettra à l'utilisateur de mettre à jour les détails d'un étudiant existant dans la base de données.

- ❖ Sélection de l'Étudiant : L'utilisateur devrait d'abord être en mesure de sélectionner l'étudiant dont il souhaite modifier les informations. Cela peut être fait en recherchant l'étudiant par son numéro apogée.
- ❖ Affichage des Informations Actuelles : Une fois que l'utilisateur a sélectionné l'étudiant à modifier, les informations actuelles de cet étudiant doivent être affichées à l'écran pour référence.
- ❖ Modification des Informations : L'utilisateur devrait pouvoir modifier les champs spécifiques de l'étudiant, tels que le nom, le prénom, la filière, les notes, etc. Chaque champ peut être traité individuellement, permettant à l'utilisateur de saisir les nouvelles valeurs pour chaque champ qu'il souhaite modifier.

- ❖ Validation des Modifications : Après avoir saisi les nouvelles valeurs, l'utilisateur devrait avoir la possibilité de vérifier les modifications apportées avant de les valider. Cela pourrait inclure l'affichage des nouvelles informations modifiées pour confirmation.
- ❖ Mise à Jour de la Base de Données : Une fois que l'utilisateur a confirmé les modifications, les informations de l'étudiant dans la base de données doivent être mises à jour avec les nouvelles valeurs fournies. (Ça concerne seulement la deuxième partie qui contient le fonctionnement sur fichiers).
- ❖ Notification de Succès : Enfin, l'utilisateur devrait recevoir une notification indiquant que les modifications ont été enregistrées avec succès dans la base de données.

Prototype : `void modifierEtudiant(Etudiant etudiants[], int nbr_etudiants_actu) ;`

Cette fonction utilise des boucles de menu pour afficher les options disponibles et récupérer le choix de l'utilisateur.

Cette fonctionnalité devrait être conçue de manière à assurer une interaction intuitive avec l'utilisateur et à garantir l'intégrité des données dans la base de données des étudiants.

```
Entrez le numéro d'apogée de l'étudiant à modifier: 100
1. Modifier le nom
2. Modifier le prénom
3. Modifier la filière
4. Modifier la date d'inscription
5. Modifier la date de naissance
6. Modifier les modules et les notes
7. Quitter
Votre choix: |
```

Figure 6 : Modification

## 6. Tri des étudiants

La fonctionnalité de tri des étudiants permettra à l'utilisateur de trier la liste des étudiants selon différents critères.

- ❖ Sélection du Critère de Tri : L'utilisateur devrait pouvoir choisir le critère selon lequel il souhaite trier les étudiants. Les critères de tri courants pourraient inclure le numéro apogée, la moyenne, la date d'inscription.
- ❖ Application du Tri : Une fois les options de tri sélectionnées, l'application devrait trier la liste des étudiants en fonction du critère choisi et de l'ordre spécifié.

Pour l'apogée :

Prototype : `void trierEtudiantApogee(Etudiant etudiants[], int nbr_etudiants_actu)`

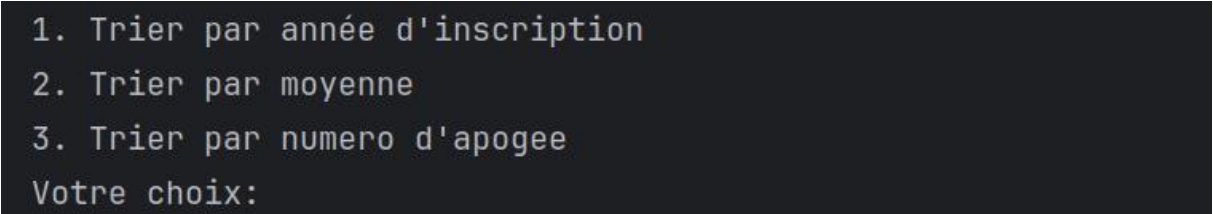
Pour la date d'inscription :

Prototype : `void trierEtudiantsInscription(Etudiant etudiants[], int nbr_etudiants_actu) ;`

Pour la moyenne :

Prototype : `void trierEtudiantMoyenne(Etudiant etudiants[], int nbr_etudiants_actu) ;`

Affichage des Résultats : Une fois le tri effectué, les étudiants triés devraient être affichés à l'utilisateur un message qui dit les étudiants sont triés par (le critère).



```
1. Trier par année d'inscription
2. Trier par moyenne
3. Trier par numero d'apogee
Votre choix:
```

Figure 7 : trie

## 7. Suppression d'étudiants

La fonctionnalité de suppression d'étudiants permettra à l'utilisateur de retirer un étudiant de la base de données.

- Sélection de l'Étudiant à Supprimer : L'utilisateur devrait d'abord être en mesure de sélectionner l'étudiant qu'il souhaite supprimer. Cela pourrait être fait en recherchant l'étudiant par son numéro apogée.
- Suppression de l'Étudiant : Une fois que l'utilisateur a confirmé la suppression, l'étudiant sélectionné doit être retiré de la base de données des étudiants. (Ça concerne seulement la deuxième partie qui contient le fonctionnement sur fichiers).
- Notification de Succès : Enfin, l'utilisateur devrait recevoir une notification indiquant que l'étudiant a été supprimé avec succès de la base de données.

Prototype : `void supprimerEtudiantApogee(Etudiant etudiants[], int* nbr_etudiants_actu )`

## 8. Sortie :

La sortie du menu principal de l'application peut être définie avec les options suivantes :

- Sauvegarder les Modifications : Cette option permet à l'utilisateur de sauvegarder toutes les modifications apportées aux données des étudiants avant de quitter l'application. Cela garantit que toutes les données mises à jour sont enregistrées pour une utilisation ultérieure.
- Quitter sans Sauvegarder : Si l'utilisateur choisit cette option, l'application se ferme sans sauvegarder les modifications apportées aux données des étudiants. Cela peut être utile si l'utilisateur souhaite quitter rapidement l'application sans enregistrer les changements.

- Annuler : Cette option permet à l'utilisateur de revenir au menu principal et de continuer à utiliser l'application sans effectuer de sauvegarde ni de quitter. Cela offre une sortie alternative si l'utilisateur ne souhaite ni sauvegarder ni quitter immédiatement.

Ces trois options offrent à l'utilisateur la flexibilité de choisir la meilleure action en fonction de ses besoins. Elles peuvent être présentées dans le menu principal de l'application pour permettre à l'utilisateur de sélectionner son choix avant de quitter l'application.

```
1. Sauvegarder les modifications
2. Quitter sans sauvegarder
3. Annuler
Votre choix: |
```

*Figure 8 : Menu de sortie*

## VI. Conclusion

En conclusion, ce projet de gestion des étudiants en langage C a été une expérience enrichissante qui nous a permis d'appliquer et de consolider nos connaissances en programmation modulaire, manipulation de fichiers et gestion de données structurées. À travers ce projet, nous avons pu comprendre l'importance de la conception modulaire pour organiser efficacement le code source, faciliter la collaboration et améliorer la maintenabilité du logiciel.

Nous avons développé plusieurs fonctionnalités essentielles pour la gestion des étudiants, telles que l'ajout, la consultation, la modification, le tri et la suppression des données. Chaque fonctionnalité a été conçue avec une attention particulière à l'expérience utilisateur, en offrant une interface conviviale et des fonctionnalités robustes.

L'utilisation de directives au préprocesseur, de types de données personnalisés et de fonctions modulaires a permis de rendre le code plus clair, plus flexible et plus facile à comprendre et à maintenir. De plus, la manipulation de fichiers pour la persistance des données a assuré que les informations des étudiants sont conservées entre les différentes exécutions de l'application.

Bien que ce projet soit abouti dans sa version actuelle, il reste des possibilités d'amélioration et d'extension. Nous pourrions envisager d'ajouter des fonctionnalités supplémentaires, telles que la génération de rapports, la gestion des professeurs, ou l'intégration d'une interface graphique pour une meilleure expérience utilisateur. De plus, des tests unitaires pourraient être mis en place pour garantir la fiabilité et la qualité du code.

En somme, ce projet nous a permis de mettre en pratique nos compétences en programmation C avancée et de créer une application fonctionnelle pour la gestion des étudiants. Il représente une étape importante dans notre parcours d'apprentissage en informatique et nous sommes impatients de continuer à développer nos compétences dans des projets futurs.