



TP3 : Développement d'un Système de Gestion de Scolarité

Contexte

Vous êtes chargé de développer en C++ un système de gestion de scolarité pour SupNum. Le programme doit permettre de gérer les informations des étudiants, leurs notes, et produire divers rapports.

Description :

- ✓ Initialisation des données
Dans votre fonction principale main, déclarez les constantes et les tableaux nécessaires pour stocker les informations des étudiants : Déclarez les constantes et tableaux nécessaires :
- MAX_ETUDIANTS = 5 (nombre maximum d'étudiants)
- NB_MATIERES = 4 (nombre de matières par semestre)
- NB_FILIERES = 5 (nombre de filières, fixe)
- Les tableaux : noms, prénoms, genres, filières, notes, moyennes, résultats

```
const int MAX_ETUDIANTS = 5;  
const int NB_MATIERES= 4;  
const int NB_FILIERES = 5;
```

```
// ----- Déclaration des tableaux -----  
string noms[MAX_ETUDIANTS];  
string prenomes[MAX_ETUDIANTS];  
char genres[MAX_ETUDIANTS];  
string filieres[MAX_ETUDIANTS];  
float notes[MAX_ETUDIANTS][NB_MATIERES ];  
float moyennes[MAX_ETUDIANTS];  
string resultats[MAX_ETUDIANTS];
```

```
string FILIERES[NB_FILIERES] = {"DSI", "RSS", "DWM", "IDS", "ISCA"};
```



Exemples de données pour les Tableaux

L'utilisation de tableaux parallèles signifie que l'indice **[i]** correspond au même étudiant dans tous les tableaux.

Les tableaux stockent l'identité et le profil de base.

- **noms[**MAX_ETUDIANTS**]** : Stocke le nom de famille en majuscules.
 - *Exemple* : {"SIDI", "FATI", "AICHA", "BAH", "DEMBA"}
- **prenoms[**MAX_ETUDIANTS**]** : Stocke le prénom tel quel.
 - *Exemple* : {"Med", "Bah", "Oumar", "Ahmed", "Samba"}
- **genres[**MAX_ETUDIANTS**]** : Stocke 'M' pour masculin ou 'F' pour féminin.
 - *Exemple* : {'M', 'F', 'F', 'M', 'M'}
- **filieres[**MAX_ETUDIANTS**]** : Stocke la filière choisie parmi la liste FILIERES (DSI, RSS, DWM, IDS, ISCA).
 - *Exemple* : {"DSI", "RSS", "RSS", "DWM", "ISCA"}
- **notes[**MAX_ETUDIANTS**][**NB_MATIERES**]** : Un tableau à deux dimensions. Chaque ligne représente un étudiant, chaque colonne une matière (0 à 20).
 - *Exemple (pour l'étudiant 0 - Sidi)* : {11.0, 12.5, 13.0, 10.0}
 - *Exemple (pour l'étudiant 1 - Fati)* : {10.5, 5.0, 3.0, 11.0}

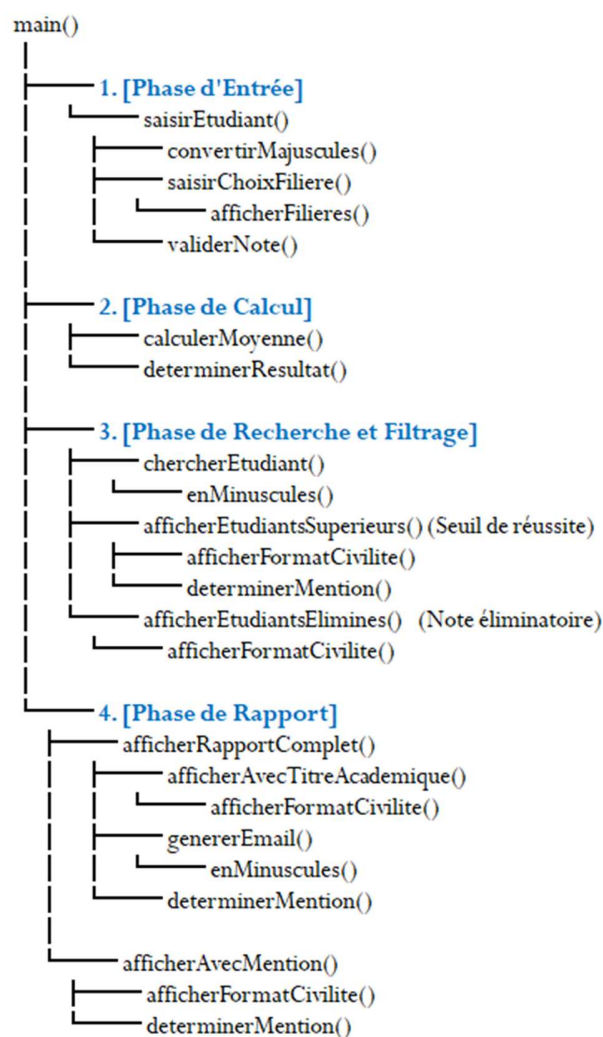
Ces 2 tableaux sont remplis après les calculs basés sur les notes.

- **moyennes[**MAX_ETUDIANTS**]** : Moyenne arithmétique des 4 notes.
 - *Exemple* : {11.62, 7.38, 14.0, 10.5, 16.5}
- **resultats[**MAX_ETUDIANTS**]** : Statut selon la moyenne (Admis, Rattrapage, Ajourné).
 - *Exemple* : {"Admis", "Ajourné", "Admis", "Admis", "Admis"}

✓ Hiérarchie des appels des fonctions

Pour vous aider, on vous donne ci-dessous la hiérarchie et les relations entre les appels des fonctions. On vous conseille fortement de développer les fonctions une par une. Testez la fonction développée avant de passer à la fonction suivante.

✓ Vous avez un scénario d'exécution du programme à la fin.





Questions :

1. Fonction de conversion

Écrivez une fonction *convertirMajuscules* qui prend une chaîne de caractères en paramètre et la convertit en majuscules. Cette fonction modifie directement la chaîne passée en paramètre. Ecrire une autre fonction *enMinuscules* qui retourne une chaîne en minuscules (utile pour les questions ci-dessous sur les emails et la recherche).

```
void convertirMajuscules(string &chaine) {  
  
}
```

```
string enMinuscules(string chaine) {  
  
}
```

2. Affichage des filières

Écrivez une fonction *afficherFilières* qui affiche la liste des 5 filières disponibles : DSI, RSS, DWM, IDS, ISCA.

```
void afficherFilières(string FILIERES[]) {  
  
}
```

3. Saisie contrôlée

Écrivez une fonction

```
int saisirChoixFiliere(string FILIERES[]) {  
  
}
```

qui :

- Affiche les filières disponibles en appelant la fonction *afficherFilières*

- Demande à l'utilisateur de choisir un nombre entre 0 et 4
- Retourne l'indice correspondant (0 pour DSI, ..., 4 pour ISCA)
- Redemande en cas de saisie invalide

4. Validation des notes

Écrivez une fonction qui force la saisie d'une note entre 0 et 20.

```
float validerNote() {  
  
}
```

qui :

- Demande à l'utilisateur de saisir une note entre 0 et 20
- Redemande la saisie tant que la note n'est pas valide
- Retourne la note validée

5. Saisie d'un étudiant

Écrivez une fonction

```
void saisirEtudiant(int indice, string FILIERES) {  
  
}
```

qui saisit et valide toutes les informations d'un étudiant `indice`:

- Nom (converti en majuscules)
- Prénom
- Genre (M ou F, en majuscule avec validation)
- Filière (en utilisant la fonction de la question 4)
- Notes des 4 matières (en utilisant la fonction de la question 5)

6. Calcul de moyenne

Écrivez une fonction

```
float calculerMoyenne(int indice, float notes[][NB_MATIERES]) {  
}
```

qui :

- Prend l'indice du ième étudiant et les notes en paramètre

- Calcule et retourne la moyenne de ses 4 notes

7. Filtrage par note éliminatoire

Parcourt tous les étudiants et affiche ceux dont la moyenne est strictement inférieure à une note seuil (éliminatoire) définie par l'utilisateur.

```
void afficherEtudiantsElimines(float noteEliminatoire, int n,
float moyennes[], string noms[], string prenomes[]) {
}
```

8. Détermination du résultat

Écrivez une fonction

```
void determinerResultat(int indice, float moyennes[], string resultats[])
{
}
}
```

qui :

- Prend l'indice du ième étudiant et sa moyenne en entrée
- Détermine le résultat selon les règles suivantes :
 - Si Moyenne ≥ 10 alors "Admis"
 - Si $8 \leq$ Moyenne < 10 alors "Rattrapage"
 - Si Moyenne < 8 alors "Ajourné"
- Stocke le résultat dans le tableau correspondant, resultats

9. Détermination de la mention

Écrivez une fonction

```
string determinerMention(float moyenne) {
}
```

qui :

- Prend une moyenne en paramètre
- Retourne la mention selon ces règles :
 - moyenne ≥ 16 : "Très Bien"
 - moyenne ≥ 14 : "Bien"

- moyenne ≥ 12 : "Assez Bien"
- moyenne ≥ 10 : "Passable"
- moyenne < 10 : "Insuffisant"

10. Affichage formaté

Écrivez une fonction

```
void afficherEtudiant(int indice, string noms[], string prenomes[], char
    genres[], string filieres[], float moyennes[], string resultats[]) {
}
```

qui :

- Prend l'indice et tous les tableaux nécessaires d'un étudiant en paramètre
- Affiche toutes ses informations de manière claire et organisée

11. Format civil

Écrivez une fonction

```
void afficherFormatCivilite(int indice, string noms[],
    string prenomes[], char genres[]) {
}
```

qui :

- Prend l'indice d'un étudiant et les tableaux noms, prénoms et genre en paramètre
- Affiche "Mr." pour les hommes, "Mlle." pour les filles
- Suivi de la première lettre du prénom en majuscule suivi de '.' et du nom
- Exemple : "Mr. A. Sidi" ou "Mlle. F. Bah"

12. Titre académique

Écrivez une fonction

```
void afficherAvecTitreAcademique(int indice, string noms[],
    string prenomes[], char genres[]) {
}
```

qui :

- Prend l'indice de l'étudiant en paramètre
- Affiche "Étudiant" ou "Étudiante" selon le genre



- Suivi du format civil
- Exemple : "Étudiant Mr. A. Sidi"

13. Génération d'email

Écrivez une fonction

```
string genererEmail(int indice, string noms[], string prenomes[]) {  
  
}
```

qui :

- Prend l'indice d'un étudiant, son nom et son prénom en paramètre
- Génère une adresse email au format : `initiale.nom@supnum.mr`
- Initiale correspond à la première lettre du prénom en minuscule
- son nom en minuscules
- Exemple : "a.sidi@supnum.mr"

14. Affichage avec mention

Écrivez une fonction

```
void afficherAvecMention(int indice) {  
  
}
```

qui :

- Prend l'indice de l'étudiant en paramètre
- Affiche le format civilité, la filière, la moyenne, le résultat et la mention
- Exemple : "Mr. A. SIDI [DSI] Moyenne: 15.50 - Admis - Mention: Bien"

15. Chercher un étudiant par son nom

Écrivez une fonction

```
int chercherEtudiant(string tabNoms[], string nomAChercher, int n)  
{  
}
```

qui :

- Prend en paramètres : le tableau des noms, le nom à chercher, et le nombre d'étudiants
- Retourne l'indice de l'étudiant si trouvé
- Retourne -1 si non trouvé
- La recherche doit être insensible à la casse



16. Rapport complet

Écrivez une fonction

```
void afficherRapportComplet(int indice, string listeFilières[],
string noms[], string prenomes[], char genres[], string filieres[],
float notes[][NB_MATIERES], float moyennes[], string resultats[]) {

}
```

qui :

- Affiche un rapport détaillé pour chaque étudiant d'indice `indice` incluant :
 - Titre académique (Etudiant ou Etudiante)
 - Email
 - Filière
 - Notes détaillées
 - Moyenne
 - Résultat
 - Mention
- Avec un formatage soigné et des séparateurs

17. Programme principal

Écrivez la fonction `main` qui :

1. Demande et valide le nombre d'étudiants
2. Saisit les informations de tous les étudiants (utilisez les fonctions précédentes)
3. Tester chaque question une à une



Un scénario d'exécution du programme

```
=== Gestion Scolarite SupNum ===
Nombre d'etudiants (1-5) : 3

--- Saisie Etudiant 1 ---
Nom : Sidi
Prenom : Med
Genre (M/F) : M
Filières disponibles :
0. DSI
1. RSS
2. DWM
3. IDS
4. ISCA

Votre choix (0-4) : 3
  Note Matiere 1 (0-20) : 10
  Note Matiere 2 (0-20) : 11.5
  Note Matiere 3 (0-20) : 12.0
  Note Matiere 4 (0-20) : 13
```

```
--- Saisie Etudiant 2 ---
Nom : Fati
Prenom : Oumar
Genre (M/F) : f
Filières disponibles :
0. DSI
1. RSS
2. DWM
3. IDS
4. ISCA

Votre choix (0-4) : 1
  Note Matiere 1 (0-20) : 11.5
  Note Matiere 2 (0-20) : 12.5
  Note Matiere 3 (0-20) : 5.5
  Note Matiere 4 (0-20) : 3.5
```

```
--- Saisie Etudiant 3 ---
Nom : Ahmed
Prenom : Ali
Genre (M/F) : m
Filières disponibles :
0. DSI
1. RSS
2. DWM
3. IDS
4. ISCA

Votre choix (0-4) : 0
  Note Matiere 1 (0-20) : 11.75
  Note Matiere 2 (0-20) : 5
  Note Matiere 3 (0-20) : 12.5
  Note Matiere 4 (0-20) : 14.5

--- MENU PRINCIPAL ---
1. Rapport complet
2. Chercher un etudiant
3. Liste Admis (Seuil)
4. Liste Elimines (Seuil)
5. Quitter
```

```
--- MENU PRINCIPAL ---
1. Rapport complet
2. Chercher un etudiant
3. Liste Admis (Seuil)
4. Liste Elimines (Seuil)
5. Quitter
Choix : 2
Nom : Fati

-----
Etudiante Mlle. O. FATI
Filiere : RSS | Moyenne : 8.25
Resultat : Rattrapage | Email : o.fati@supnum.mr
-----
```



```
--- MENU PRINCIPAL ---
1. Rapport complet
2. Chercher un etudiant
3. Liste Admis (Seuil)
4. Liste Elimines (Seuil)
5. Quitter
Choix : 4
Note eliminatoire : 6.0
Aucun etudiant elimine.

--- MENU PRINCIPAL ---
1. Rapport complet
2. Chercher un etudiant
3. Liste Admis (Seuil)
4. Liste Elimines (Seuil)
5. Quitter
Choix : 3
Seuil : 10.0
Mr. M. SIDI | Moyenne: 11.62 | Passable
Mr. A. AHMED | Moyenne: 10.94 | Passable
```

```
--- MENU PRINCIPAL ---
1. Rapport complet
2. Chercher un etudiant
3. Liste Admis (Seuil)
4. Liste Elimines (Seuil)
5. Quitter
Choix : 1

===== RAPPORT COMPLET =====
Etudiant Mr. M. SIDI | IDS | Moy: 11.62 | Res: Admis | Ment: Passable
Etudiante Mlle. O. FATI | RSS | Moy: 8.25 | Res: Rattrapage | Ment: Insuffisant
Etudiant Mr. A. AHMED | DSI | Moy: 10.94 | Res: Admis | Ment: Passable
```