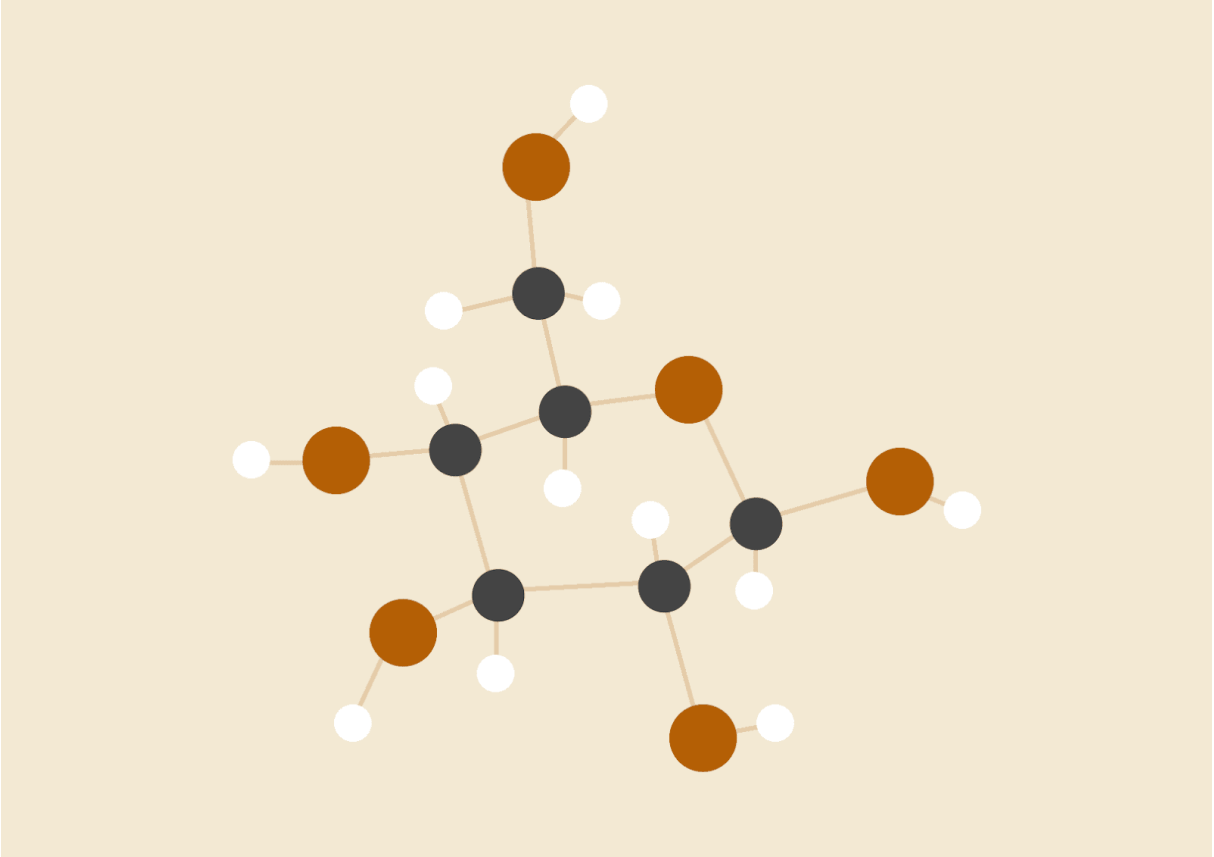
DOSSIER DE DÉVELOPPEMENT LOGICIEL

SAE S1.01



Ariane Luc - Romain Millet

11/11/2022

1 ère année BUT Informatique – 104

Table des matières

Présentation

Le rôle fonctionnel de l'application :

L'objectif de notre interpréteur de commande est de mettre à disposition de l'utilisateur un outil qui lui permet de gérer une formation universitaire. C'est à travers les différentes commandes, que l'utilisateur peut insérer et traiter des données dans les structures Formation et Étudiants.

Le niveau présentation

La présentation est très rudimentaire, en effet il n'y a rien d'affiché au préalable sur notre cmd, l'utilisateur se contente seulement d'écrire les commandes ainsi que les données qu'il souhaite introduire dans la machine ou bien à l'aide d'un fichier in préalablement rempli par les commandes et donnés souhaitées.

Insérer des données

C'est dans nos structures formation et étudiants que l'on stock les données qui seront insérées par l'utilisateur et réutilisées pour effectuer des tâches.

Tâches

Certaines commandes vont permettre de réaliser des vérifications tel que les coefficients ou même d'attribuer une décision de jury face au bulletin d'un étudiant.

Les entrées et les sorties de l'application

Les entrées

L'utilisateur peut affecter, le nombre de semestre, les matières, les épreuves ainsi que les coefficients à la formation. De plus, il peut aussi introduire des notes qui seront affectées à un élève préalablement enregistré avec son prénom.

Les sorties

Nous aurons deux types de sorties, les sorties qui sont de type erreur ou réussite.

En effet s'il y a des erreurs dans les commandes introduites, en sortie ses erreurs seront prévenues par l'invité de commande qui affichera un message particulier suivant l'erreur introduite. Cela permet au client de comprendre son erreur et de la modifier.

Exemple de sortie d'erreur : "Le numero de semestre est incorrect"

Par ailleurs, il y a aussi des sorties de réussite qui permettent de prévenir que la commande est correcte et à bien été prise en compte. Prenons l'exemple de : "Matiere ajoutee a la formation" ou "Etudiant ajoute a la formation".

Fonctions

Nous avons créé une fonction initialisation, afin d'initialiser les tableaux de nos structures à 0. Il y a seulement le tableau des notes qui a été initialisé à -1, car nous voulions rechercher si un étudiant possédait déjà ou non une note. Sachant qu'une note peut être égale à 0, on a préféré l'initialiser à -1 pour la vérification.

De plus, nous avons créé une petite fonction nommée arrondi, qui arrondi en supprimant ce qui se trouve après le premier chiffre après la virgule et non en arrondissant au chiffre le plus près.

Validation

L'organisation des tests de l'application et le bilan de validation des différents sprints que vous avez développés.

Pour tester le programme, nous avons aussi personnaliser nos fichiers in, par exemple pour tester notre fonction "coefficient" ou bien pour tester les sorties de nos commandes, par exemple s'il manquait une note dans la formation etc.

Jeux de tests de l'application

Difficultés rencontrés :

1. Les printfs ne coïncidaient pas avec le fichier out du sprint. En effet, il nous est arrivé d'avoir mal copié le message d'erreur demandé par le sujet. Ou bien l'ordre d'affichage des sorties n'était pas respecté. Exemple au lieu d'afficher :

   Matiere ajoutee

   Epreuve ajoutee

   notre programme affichait :

   Epreuve ajoutee

   Matiere ajoutee

2. Les alignements :

   Même si les alignements ne devaient pas obligatoirement être respectés, les respecter nous donnait une satisfaction supplémentaire. C'est pourquoi nous avons fait en sorte de les respecter ce qui n'a pas été facile dès le début.

Validation en TP

Pour la validation de nos sprints nous avons préféré commencer de manière crescendo. Nous avons validé notre sprint 1, 2 et 3 sans soucis. Puis nous avons eu un problème avec le sprint n°4 pour la commande décision. En effet nous avons remarqué qu'il y avait un problème d'arrondi avec les moyennes annuelles. En effet, il manquait de temps en temps un centième qui faussait notre résultat. Pour résoudre ce problème, premièrement nous avons enregistré la vrai valeur de la moyenne des épreuves par UE, et affiché dans le printf la valeur arrondi. Puis pour la moyenne annuelle, nous avons utilisé la moyenne précédente qui n'était pas arrondi, puis calculer la moyenne annuelle et nous l'avons affiché avec arrondi.

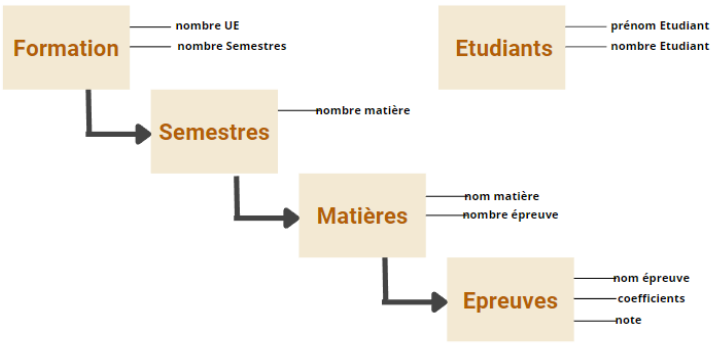
Bilan

Difficultés rencontrées :

Première difficulté :

Notre première difficulté a été la commande 3 car nous n'avions pas les connaissances nécessaires pour la réussir. En effet, on ne voyait pas comment visualiser et parcourir cette structure. Par notre manque de connaissance, c'est la commande sur laquelle nous avons passé le plus de temps (environ 3 semaines). Après avoir demandé à nos paires ainsi que notre professeur de TD sur la manière dont on pouvait parcourir une structure de données, nous avons pu comprendre et finir cette commande. Et aller plus vite sur les autres commandes.

Voici un schéma que nous avons fait pour comprendre les structures :



Deuxième difficulté :

Le deuxième problème auquel nous avons pu faire face est la vérification des coefficients dans les commandes 4, 7 et 8. En effet dans un premier temps nous pensions que la vérification devait s'effectuer de cette manière :

Vérification horizontale ❌

Epreuves	UE 1	UE 2	UE 3

Programmation	0	0	0
SGBD	1	2	0.5

Tandis qu'au final il fallait effectuer une vérification verticale car si on à tous les coefficients d'une UE qui sont égal à 0 lors du calcul de la moyenne pondérés le dénominateur sera égal à 0 donc calcul impossible.

Vérification verticale ✔

Epreuves	UE 1	UE 2	UE 3
Programmation	0	2	0.5
SGBD	0	0.5	1

Ce qui est réussi

Appart quelques améliorations que nous allons citer juste après, nous pensons avoir majoritairement tout réussi, vu que nos jeux de tests ont plutôt bien marché.

Quelques précisions : On a fait en sorte que les étudiants ainsi que les notes ne s'enregistrent pas tant que toutes les conditions n'ont pas été respectées.

Amélioration

Dans notre programme, plusieurs améliorations sont à envisager. Comme par exemple l'optimisation de ce dernier.*

1. Fonctions

En effet on pourrait créer plus de fonctions, qui permettraient d'éviter de copier des bouts de codes entre les commandes.

2. Switch

Utiliser un switch dans la fonction main aurait été plus optimum

3. Complexité mathématique inférieur

A l'aide des fonctions ont aurait pu faire de boucles moins complexe pour l'optimisation

Annexe

Sources utilisés

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#pragma warning (disable:4996)

enum {
	nb_semestre = 2,
	UE_min = 3,
	UE_max = 6,
	matiere_max = 10,
	epreuves_max = 5,
	etudiants_max = 100,
	char_max = 30
};

const float MIN_NOTE = 0.f, MAX_NOTE = 20.f;

typedef char Nom[char_max];

//structure étudiante
typedef struct {
	Nom etu[etudiants_max];
	int nbEtu;
}Etudiants;


//structure formation
typedef struct {
	Nom nom;
	float coef[UE_max];
	float note[etudiants_max];
} Epreuve;

typedef struct {
	char nom[char_max];
	unsigned int nbEpreuves;
	Epreuve epreuves[epreuves_max];
} Matiere;

typedef struct {
```

```
        unsigned int nbMatiere;
        Matiere matiere[matiere_max];
} Semestre;


typedef struct {
        unsigned int nbUE;
        Semestre semestre[nb_semestre];
} Formation;

/*********************************************************  INITIALISATION
*******************************************************************************/

void initial(Formation* f, Etudiants* e) {
        f->nbUE = 0;
        e->nbEtu = 0;
        //initialisation nombre de matières
        for (int i = 0; i < nb_semestre; i++) {
                f->semestre[i].nbMatiere = 0;
                //initialisation nombre d'epreuves
                for (int j = 0; j < matiere_max; j++) {
                        f->semestre[i].matiere[j].nbEpreuves = 0;
                        //initialisation note
                        for (int k = 0; k < epreuves_max; k++) {
                                for (int p = 0; p < etudiants_max; p++) {
                                        f-
>semestre[i].matiere[j].epreuves[k].note[p] = -1;
                                }
                        }

                }

        }

}
/*********************************************************  ARRONDI
*****************************************/

float arrondi(float a) {
        float tmp = a * 10;
        int tmp2 = (int)tmp;
        return (tmp2 / 10.0);
}

/*********************************************************    COMMANDE 2
*******************************************************************************/
void formation(Formation* f) {
        unsigned int nbue;
        //verification des nombres de UE
        if (f->nbUE >= 3 && f->nbUE <= 6) {
                printf("Le nombre d'UE est deja defini\n");
        }
        else {
                scanf("%u", &nbue);
                if (nbue >= 3 && nbue <= 6) {
                        f->nbUE = nbue;
                        printf("Le nombre d'UE est defini\n");
                }
```

```
                else {
                        printf("Le nombre d'UE est incorrect\n");
                }
        }
}

/******************************************************** COMMANDE 3
*****************************************************************************************/
void epreuve(Formation* f) {
        int nsem;
        char matiere[char_max];
        char epreuve[char_max];
        scanf("%d", &nsem);

        //vérification du numero de semestre
        if (nsem == 1 || nsem == 2) {
                int cmtmatiere = 0;
                int cmtepreuve = 0;

                //On cherche si la matiere existe deja
                scanf("%s", matiere);

                while (cmtmatiere < f->semestre[nsem - 1].nbMatiere &&
strcmp(matiere, f->semestre[nsem - 1].matiere[cmtmatiere].nom) != 0)
                        ++cmtmatiere;

                //On cherche si l'epreuve existe deja
                scanf("%s", epreuve);//nom de l'epreuve
                while (cmtepreuve < f->semestre[nsem -
1].matiere[cmtmatiere].nbEpreuves && strcmp(epreuve, f->semestre[nsem -
1].matiere[cmtmatiere].epreuves[cmtepreuve].nom) != 0)
                        ++cmtepreuve;

                //Condition de l'epreuve
                if (cmtepreuve < f->semestre[nsem -
1].matiere[cmtmatiere].nbEpreuves) {//epreuve existe deja
                        printf("Une meme epreuve existe deja\n");
                        return;
                }
                else {
                        float coefe[UE_max];
                        float lecoef;
                        int coefnul = 0;
                        //Récupération de coef en fonction du nombre d'UE
                        for (int i = 0; i < f->nbUE; i++) {
                                scanf("%f", &lecoef);
                                if (lecoef < 0) {
                                        printf("Au moins un des
coefficients est incorrect\n");
                                        return;
                                }
                                else {
                                        coefe[i] = lecoef;
                                        if (lecoef == 0)
                                                coefnul++;
                                }
                                //Vérification des coef
                        }if (coefnul == f->nbUE) {
```

```
                                printf("Au moins un des coefficients est
incorrect\n");
                                return;
                        }
                        else {

                                //Ajout des coefficients
                                for (int i = 0; i < f->nbUE; i++)
                                        f->semestre[nsem -
1].matiere[cmtmatiere].epreuves[cmtepreuve].coef[i] = coefe[i];

                                //Ajout de 1 au nombre de matière
                                if (cmtmatiere == f->semestre[nsem -
1].nbMatiere) {
                                        printf("Matiere ajoutee a la
formation\n");
                                        f->semestre[nsem - 1].nbMatiere +=
1;
                                }
                                //Ajout de l'epreuve
                                strcpy(f->semestre[nsem -
1].matiere[cmtmatiere].epreuves[cmtepreuve].nom, epreuve);//
                                printf("Epreuve ajoutee a la
formation\n");

                                //ajout de la matiere
                                strcpy(f->semestre[nsem -
1].matiere[cmtmatiere].nom, matiere);
                                //ajout de 1 au nombre d'epreuves
                                f->semestre[nsem -
1].matiere[cmtmatiere].nbEpreuves += 1;

                        }
                }
        }
        else
                printf("Le numero de semestre est incorrect\n");
}
/*************************************************** COMMANDE 4
*********************************************************************************************/
void verifcoef(Formation* f) {
        //variables locales
        unsigned int nsemestre;
        unsigned int totalepreuve;
        int cmtnul;

        scanf("%d", &nsemestre);

        //vérification du numero de semestres
        if (nsemestre == 1 || nsemestre == 2) {
                for (int i = 0; i < f->nbUE; i++) {
                        totalepreuve = 0;
                        //Vérif existence des épreuves (si aucune matiere
alors aucune epreuve)
                        if (f->semestre[nsemestre - 1].nbMatiere == 0) {
                                printf("Le semestre ne contient aucune
epreuve\n");
                                return;
```

```
                        }
                        cmtnul = 0;
                        //Vérif existence des épreuves avec le nombre
d'epreuves
                        for (int j = 0; j < f->semestre[nsemestre -
1].nbMatiere; j++) {
                                if (f->semestre[nsemestre -
1].matiere[j].nbEpreuves == 0) {
                                        printf("Le semestre ne contient
aucune epreuve\n");
                                        return;
                                }
                                //vérification des coefficients
                                for (int k = 0; k < f->semestre[nsemestre
- 1].matiere[j].nbEpreuves; k++) {
                                        ++totalepreuve;
                                        if (f->semestre[nsemestre -
1].matiere[j].epreuves[k].coef[i] == 0) {
                                                ++cmtnul;
                                        }
                                }
                        }
                        if (cmtnul == totalepreuve) {
                                printf("Les coefficients d'au moins une UE
de ce semestre sont tous nuls\n");
                                return;
                        }
                }
                printf("Coefficients corrects\n");
        }
        else {
                printf("Le numero de semestre est incorrect\n");
                return;
        }
}
/************************************************** COMMANDE 5
**************************************************************************************
***********/

void note(Formation* f, Etudiants* e) {

        //variables
        int nsem = 0;
        char nommatiere[char_max];
        char nomepreuve[char_max];
        float note;
        char prenom[char_max];

        //compteurs
        int cmtmatiere = 0;
        int cmtepreuve = 0;
        int cmtetudiant = 0;

        //scanf
        scanf("%d", &nsem);
        scanf("%s", &prenom);
        scanf("%s", &nommatiere);
        scanf("%s", &nomepreuve);
```

```
        scanf("%f", &note);

        //vérification du numero de semestre
        if (nsem == 1 || nsem == 2) {

                //****Chercher les matières****//
                while (cmtmatiere < f->semestre[nsem - 1].nbMatiere &&
strcmp(nommatiere, f->semestre[nsem - 1].matiere[cmtmatiere].nom) != 0)
                        ++cmtmatiere;
                if (cmtmatiere != f->semestre[nsem - 1].nbMatiere)
{//matiere existe deja
                        //cherche existence epreuve
                        while (cmtepreuve < f->semestre[nsem -
1].matiere[cmtmatiere].nbEpreuves && strcmp(nomepreuve, f-
>semestre[nsem - 1].matiere[cmtmatiere].epreuves[cmtepreuve].nom) != 0)
                                ++cmtepreuve;
                        if (cmtepreuve != f->semestre[nsem -
1].matiere[cmtmatiere].nbEpreuves) {//epreuve existe déjà
                                //parcourir les notes
                                if (note >= 0 && note <= 20) {
                                        //On cherche parmis les etudiants
connus
                                        while (cmtetudiant < e->nbEtu &&
strcmp(prenom, e->etu[cmtetudiant]) != 0)
                                                ++cmtetudiant;
                                        //Vérif si la note existe
                                        if (f->semestre[nsem -
1].matiere[cmtmatiere].epreuves[cmtepreuve].note[cmtetudiant] == -1) {
//tableau préalablement initalisé à -1
                                                //enregistre la note
                                                f->semestre[nsem -
1].matiere[cmtmatiere].epreuves[cmtepreuve].note[cmtetudiant] = note;
                                                if (cmtetudiant == e->nbEtu)
{ //pas trouvé d'étudiants donc enregistre prenom
                                                        strcpy(e-
>etu[cmtetudiant], prenom);
                                                        ++(e->nbEtu);//un de
plus
                                                        printf("Etudiant
ajoute a la formation\n");


                                                }
                                                else {
                                                }
                                                printf("Note ajoutee a
l'etudiant\n");

                                        }
                                        else {
                                                printf("Une note est deja
definie pour cet etudiant\n");
                                        }

                                }
                                else {
                                        printf("Note incorrect\n");
                                }
```

```
                        }
                        else {//epreuve existe pas
                                printf("Epreuve inconnue\n");
                        }
                }
                else { //matiere n'existe pas
                        printf("Matiere inconnue\n");
                }
        }
        else {
                printf("Le numero de semestre est incorrect\n");
        }
}
/********************************************** COMMANDE 6
**********************************************************************************************
*****************************************************************/
void notes(Formation* f, Etudiants* e) {
        //variables
        int nsem = 0;
        char prenom[char_max];
        //compteurs
        int cmtetudiant = 0;

        //scanf
        scanf("%d", &nsem);
        scanf("%s", &prenom);

        //vérif du numero de semestre
        if (nsem == 1 || nsem == 2) {
                //vérifier si l'étudiant existe déjà
                while (cmtetudiant < e->nbEtu && strcmp(prenom, e-
>etu[cmtetudiant]) != 0)
                        ++cmtetudiant;
                if (cmtetudiant < e->nbEtu) { //trouvé

                        //rechercher s'il manque des notes dans chaque
épreuve il faut vérifier
                        for (int i = 0; i < f->semestre[nsem -
1].nbMatiere; i++) {
                                for (int j = 0; j < f->semestre[nsem -
1].matiere[i].nbEpreuves; j++) {
                                        if (f->semestre[nsem -
1].matiere[i].epreuves[j].note[cmtetudiant] == -1) {//note inexistante
car == -1
                                                printf("Il manque au moins
une note pour cet etudiant\n");
                                                return;
                                        }

                                }

                        }
                        //note existante
                        printf("Notes correctes\n");


                }
                else {//pas trouvé donc enregistre
```

```
                        printf("Etudiant inconnu\n");
                }
        }
        else {
                printf("Le numero de semestre est incorrect\n");
        }
}

void releve(Formation* f, Etudiants* e) {
        //variables
        int nsem = 0;
        char prenom[char_max];
        //compteurs
        int cmtetudiant = 0;
        int strmax = 0;

        //scanf
        scanf("%d", &nsem);
        scanf("%s", &prenom);

        //vérification du numero de semestre
        if (nsem == 1 || nsem == 2) {
                //vérifier si l'étudiant existe déjà
                while (cmtetudiant < e->nbEtu && strcmp(prenom, e-
>etu[cmtetudiant]) != 0)
                        ++cmtetudiant;
                if (cmtetudiant < e->nbEtu) { //trouvé
                        //Vérifier les coefficients du semestre :
                        for (int i = 0; i < f->nbUE; i++) {
                                int totalepreuve = 0;
                                int cmtnul = 0;
                                for (int j = 0; j < f->semestre[nsem -
1].nbMatiere; j++) {
                                        if (strlen(f->semestre[nsem -
1].matiere[j].nom) > strmax) {
                                                strmax = strlen(f-
>semestre[nsem - 1].matiere[j].nom);
                                        }
                                        for (int k = 0; k < f-
>semestre[nsem - 1].matiere[j].nbEpreuves; k++) {
                                                ++totalepreuve;
                                                if (f->semestre[nsem -
1].matiere[j].epreuves[k].coef[i] == 0) {
                                                        ++cmtnul;
                                                }
                                        }
                                }
                                if (cmtnul == totalepreuve) {
                                        printf("Les coefficients de ce
semestre sont incorrects\n");
                                        return;
                                }
                        }

                        //rechercher s'il manque des notes dans chaque
épreuve il faut vérifier
                        for (int i = 0; i < f->semestre[nsem -
1].nbMatiere; i++) {
```

```
                                for (int j = 0; j < f->semestre[nsem -
1].matiere[i].nbEpreuves; j++) {
                                        if (f->semestre[nsem -
1].matiere[i].epreuves[j].note[cmtetudiant] == -1) { //note inexistante
car === -1
                                                printf("Il manque au moins
une note pour cet etudiant\n");
                                                return;
                                        }
                                }
                        }
                }
                else {
                        printf("Etudiant inconnu\n");
                        return;
                }
        }
        else {
                printf("Le numero de semestre est incorrect\n");
                return;
        }

        /***Début de l'affichage du bulletin***/

                //Affichage des UE
        printf("%-*s", strmax + 1, "");
        for (int i = 1; i <= f->nbUE; ++i) {
                printf(" UE%-2d", i);
        }

        printf("\n");
        float moyennes[UE_max];
        float coef[UE_max];

        //initialisation des tableaux moyennes et coef
        for (int i = 0; i < UE_max; ++i) {
                moyennes[i] = 0;
                coef[i] = 0;
        }

        //Calculer la moyenne des UE
        for (int i = 0; i < f->semestre[nsem - 1].nbMatiere; i++) {
                printf("%-*s", strmax, f->semestre[nsem -
1].matiere[i].nom);
                for (int u = 0; u < f->nbUE; ++u) {

                        float nxc = 0; //variable produit entre coef et
note
                        float sc = 0;  //variable somme des coef
                        float sue;     //division entre le produit et la
somme des coef

                        for (int j = 0; j < f->semestre[nsem -
1].matiere[i].nbEpreuves; j++) {
                                nxc += (f->semestre[nsem -
1].matiere[i].epreuves[j].coef[u]) * (f->semestre[nsem -
1].matiere[i].epreuves[j].note[cmtetudiant]);
```

```
                                sc += f->semestre[nsem -
1].matiere[i].epreuves[j].coef[u];
                                coef[u] += f->semestre[nsem -
1].matiere[i].epreuves[j].coef[u];
                        }
                        moyennes[u] += nxc;

                        //Si coef = 0 note Non Défini
                        if (sc == 0) {
                                printf(" %4s", "ND");
                        }
                        else {
                                sue = (nxc / sc);
                                printf(" %4.1f", arrondi(sue));
                        }
                }
                printf(" \n");
        }
        printf("--\n");
        printf("%-*s", strmax, "Moyennes");

        //Affichage de la moyenne pondéré par UE
        for (int i = 0; i < f->nbUE; ++i) {
                printf("%5.1f", arrondi(moyennes[i] / coef[i]));
        }
        printf(" \n");
}


void decision(Formation* f, Etudiants* e) {
        //Variables
        char prenom[char_max];
        int nsem = 0;
        int cmtcoef = 0;
        float sommenotesem[UE_max]; //tableau pour la moyenne des 2
semestres

        //Compteurs
        int cmtetudiant = 0;
        int cmtmatiere = 0;
        int cmtepreuve = 0;

        //scanf
        scanf("%s", prenom);

        //Rechercher si l'étudiant existe :
        while (cmtetudiant < e->nbEtu && strcmp(prenom, e-
>etu[cmtetudiant]) != 0)
                ++cmtetudiant;

        //Trouvé
        if (cmtetudiant < e->nbEtu) {
                //Vérifier les coefficients des semestres :
                for (nsem; nsem < nb_semestre; ++nsem) {
                        for (int i = 0; i < f->nbUE; i++) {
                                int cmtnul = 0;
                                int totalepreuve = 0;
```

```
                                for (int j = 0; j < f-
>semestre[nsem].nbMatiere; j++) {
                                        for (int k = 0; k < f-
>semestre[nsem].matiere[j].nbEpreuves; k++) {
                                                ++totalepreuve;
                                                if (f-
>semestre[nsem].matiere[j].epreuves[k].coef[i] == 0) {
                                                        ++cmtnul;
                                                }
                                        }
                                }
                                if (cmtnul == totalepreuve) {
                                        printf("Les coefficients d'au moins
un semestre sont incorrects\n");
                                        return;
                                }
                        }

                        //rechercher s'il manque des notes dans chaque
épreuve il faut vérifier
                        for (int i = 0; i < f->semestre[nsem].nbMatiere;
i++) {
                                for (int j = 0; j < f-
>semestre[nsem].matiere[i].nbEpreuves; j++) {
                                        if (f-
>semestre[nsem].matiere[i].epreuves[j].note[cmtetudiant] == -1) {
//note inexistante
                                                printf("Il manque au moins
une note pour cet etudiant\n");
                                                return;
                                        }
                                }
                        }
                }
        }
        else { // Pas trouvé :
                printf("Etudiant inconnu\n");
                return;
        }


        //Calcul des moyennes de semestres
        printf("%-19s", "");
        for (int i = 1; i <= f->nbUE; ++i) {
                sommenotesem[i - 1] = 0.0;
                printf(" UE%-2d", i);
        }
        printf("\n");
        float moyennes[UE_max];
        float coef[UE_max];

        //calculer les moyennes des UE
        for (int sem = 0; sem < nb_semestre; ++sem) {
                for (int i = 0; i < f->nbUE; ++i) {
                        moyennes[i] = 0.0;
                        coef[i] = 0.0;
                }
                for (int i = 0; i < f->semestre[sem].nbMatiere; i++) {
```

```
                        for (int u = 0; u < f->nbUE; ++u) {
                                float nxc = 0; //produit des coef et notes
                                float sc = 0;  //somme coefficients
                                float sue;             //somme par UE
                                for (int j = 0; j < f-
>semestre[sem].matiere[i].nbEpreuves; j++) {
                                        nxc += (f-
>semestre[sem].matiere[i].epreuves[j].coef[u]) * (f-
>semestre[sem].matiere[i].epreuves[j].note[cmtetudiant]);
                                        sc += f-
>semestre[sem].matiere[i].epreuves[j].coef[u];
                                        coef[u] += f-
>semestre[sem].matiere[i].epreuves[j].coef[u];
                                }
                                moyennes[u] += nxc;
                                if (sc == 0) {
                                }
                                else {
                                        sue = (nxc / sc);
                                }
                        }
                }
                printf("S%-18d", sem + 1);
                for (int i = 0; i < f->nbUE; ++i) {
                        sommenotesem[i] += (moyennes[i] / coef[i]);
                        float m = arrondi(moyennes[i] / coef[i]);
                        printf("%4.1f ", m);
                }
                printf("\n");
        }
        printf("--\n");
        printf("%-19s", "Moyennes annuelles");
        for (int i = 0; i < f->nbUE; ++i) {
                float m = (sommenotesem[i] / nb_semestre);
                printf("%4.1f ", arrondi(m));
                sommenotesem[i] = m;  //à présent, notre tableau sera un
tableau de moyenne anuelle
        }
        printf("\n");
        printf("%-18s", "Acquisition");
        unsigned int compteurUE = 0;
        unsigned int indicesup = UE_max;
        //On cherche l'indice de la derniere UE validé pour le placement
de la virgule
        for (int i = 0; i < f->nbUE; ++i) {
                if (sommenotesem[i] >= 10) {
                        indicesup = i;
                }
        }
        //Si indice n'a pas été modifié soit = UE_max alors aucune
        if (indicesup == UE_max) {
                printf("Aucune");
        }
        //Sinon placement de la virgule
        else {
                for (int i = 0; i < f->nbUE; ++i) {
                        if (sommenotesem[i] >= 10) {
                                ++compteurUE;
```

```
                                if (i < indicesup) {
                                        printf(" UE%d%s", i + 1, ",");
                                }
                                else {
                                        printf(" UE%d", i + 1);
                                }
                        }
                }
        }

        printf("\n");
        //Passage lorsque le nombre d'UE validé est strictement
supérieur à la moitié du nombre d'UE
        if (compteurUE > f->nbUE / 2)
                printf("%-19s%s\n", "Devenir", "Passage");
        else
                printf("%-19s%s\n", "Devenir", "Redoublement");
}


int main() {
        //Structures
        Formation f;
        Etudiants e;
        //Initialisation des strucures
        initial(&f, &e);
        //Variables
        int nbmatiere = 0;
        int nbEpreuve = 0;
        char cmd[31] = "";
        do {
                scanf("%s", cmd);
                if (strcmp(cmd, "formation") == 0) {//C2
                        formation(&f);
                }
                else if (strcmp(cmd, "epreuve") == 0 && f.nbUE != 0)
{//C3
                        epreuve(&f);
                }
                else if (strcmp(cmd, "coefficients") == 0) {//C4
                        verifcoef(&f);
                }
                else if (strcmp(cmd, "note") == 0) {//C5
                        note(&f, &e);
                }
                else if (strcmp(cmd, "notes") == 0) { //C6
                        notes(&f, &e);

                }
                else if (strcmp(cmd, "releve") == 0) { //C7
                        releve(&f, &e);
                }
                else if (strcmp(cmd, "decision") == 0) { //C8
                        decision(&f, &e);
                }

        } while (strcmp(cmd, "exit") != 0); //C1
}
```

Sprint du plus haut niveau validé

Plus haut sprint en TP

Sprint 4

```
C:\Users\Eleve\Desktop\demain>fc /w res-sp4.txt out-sp4-gr4.txt
Comparaison des fichiers res-sp4.txt et OUT-SP4-GR4.TXT
FC : aucune différence trouvée
```

Plus haut sprint avec ce dossier Jeu de tests (in/out)

Sprint 4

```
C:\Users\Ariane\source\repos\SAE Gestion de Formation Ariane et Romain\x64\Debug>Gestion_de_formation.exe < in-sp4-base.txt > res-sp4.txt

C:\Users\Ariane\source\repos\SAE Gestion de Formation Ariane et Romain\x64\Debug>fc /w  out-sp4-base.txt res-sp4.txt
Comparaison des fichiers out-sp4-base.txt et RES-SP4.TXT
FC : aucune différence trouvée
```

IN-OUT-SP4

formation 3

Le nombre d'UE est defini

epreuve 1 Programmation Projet 1 2 0

Matiere ajoutee a la formation

Epreuve ajoutee a la formation

epreuve 1 Programmation DST 2 3 0

Epreuve ajoutee a la formation

epreuve 1 SGBD Participation 0.5 0 0.5

Matiere ajoutee a la formation

Epreuve ajoutee a la formation

epreuve 1 SGBD Rapport 1.5 0 1.5

Epreuve ajoutee a la formation

epreuve 2 Architecture Interrogation 1 0 2

Matiere ajoutee a la formation

Epreuve ajoutee a la formation

epreuve 2 Architecture DST 0 1 4

Epreuve ajoutee a la formation

epreuve 2 Systeme QCM 2 3 0.5

Matiere ajoutee a la formation

Epreuve ajoutee a la formation

epreuve 2 Systeme Expose 3 2 0.5

Epreuve ajoutee a la formation

note 1 Paul Programmation Projet 12

Etudiant ajoute a la formation

Note ajoutee a l'etudiant

note 1 Paul Programmation DST 9

Note ajoutee a l'etudiant

note 1 Paul SGBD Participation 16

Note ajoutee a l'etudiant

note 1 Paul SGBD Rapport 12

Note ajoutee a l'etudiant

note 2 Paul Architecture Interrogation 18

Note ajoutee a l'etudiant

note 2 Paul Architecture DST 12

Note ajoutee a l'etudiant

note 2 Paul Systeme QCM 7

Note ajoutee a l'etudiant

note 2 Paul Systeme Expose 8

Note ajoutee a l'etudiant

note 1 Paule Programmation Projet 8

Etudiant ajoute a la formation

Note ajoutee a l'etudiant

note 1 Paule Programmation DST 11

Note ajoutee a l'etudiant

note 1 Paule SGBD Participation 20

Note ajoutee a l'etudiant

note 1 Paule SGBD Rapport 0

Note ajoutee a l'etudiant

note 2 Paule Architecture Interrogation 18

Note ajoutee a l'etudiant

note 2 Paule Architecture DST 12

Note ajoutee a l'etudiant

note 2 Paule Systeme QCM 7

Note ajoutee a l'etudiant

note 2 Paule Systeme Expose 8

Note ajoutee a l'etudiant

note 1 Paulo Programmation Projet 12

Etudiant ajoute a la formation

Note ajoutee a l'etudiant

note 1 Paulo Programmation DST 9

Note ajoutee a l'etudiant

note 1 Paulo SGBD Participation 16

Note ajoutee a l'etudiant

note 1 Paulo SGBD Rapport 12

Note ajoutee a l'etudiant

note 2 Paulo Architecture Interrogation 17

Note ajoutee a l'etudiant

note 2 Paulo Architecture DST 15

Note ajoutee a l'etudiant

note 2 Paulo Systeme QCM 16

Note ajoutee a l'etudiant

note 2 Paulo Systeme Expose 19

Note ajoutee a l'etudiant

decision Paul

	UE1	UE2	UE3
S1	11.2	10.2	13.0
S2	9.3	8.1	13.0

--

Moyennes annuelles 10.2 9.1 13.0

Acquisition UE1, UE3

Devenir Passage

decision Paule

	UE1	UE2	UE3
S1	8.0	9.8	5.0
S2	9.3	8.1	13.0

--

Moyennes annuelles 8.6 8.9 9.0

Acquisition Aucune

Devenir Redoublement

decision Paulo

	UE1	UE2	UE3
S1	11.2	10.2	13.0
S2	17.6	16.8	15.9

--

Moyennes annuelles 14.4 13.5 14.4

Acquisition UE1, UE2, UE3

Devenir Passage