Fiche n° 8 de TP

Tableaux (1)

Objectifs : manipulation de tableaux : initialisation, parcours.

Prérequis : notion de tableau : longueur, indice, opérateur [].

Travail minimum : exercices 1 à 5.

Le but de cette séance est de créer un petit logiciel permettant de déterminer si un étudiant, dont on connaît les notes dans une matière, est admis à celle-ci. Nous profiterons de l'occasion pour apprendre à écrire un fichier `Makefile`.

Exercice 1

Le logiciel fonctionnera de la façon suivante : les notes seront écrites dans un fichier texte appelé `notes.txt`. Ce dernier contient :

- une ligne de notes, finissant par `|`. Il s'agit de la liste des notes obtenues aux différents contrôles. Une note `-1` signifie une défaillance ;
- une ligne de coefficients, finissant par `|`. Il s'agit du coefficient des notes à chaque épreuve. Il doit y avoir le même nombre de valeurs sur cette ligne que sur la ligne précédente ;
- enfin, une ligne avec une seule note qui est celle de l'épreuve facultative. Un `-1` signifie que l'étudiant n'a pas passé cette épreuve.

Le programme sera appelé dans le terminal grâce à la ligne de commande `./bulletin < notes.txt`.
Rappelez ce que signifie cette syntaxe.

Exercice 2

Le « projet » est découpé en plusieurs fichiers :

1) un fichier `main.c` qui contient le programme principal. Remarquez qu'il contient une ligne `#define "ue.h"` ;

2) un fichier `ue.h` qui contient les prototypes d'un ensemble de fonctions qui seront utilisées dans `main.c` ;

3) deux fichiers `tab_double_io.c` et `tab_notes.c` qui contiendront les corps ce ces fonctions ;

4) un fichier `Makefile` qui permet de réaliser la compilation.

Ouvrez le fichier `Makefile`, il contient un certain nombre de lignes expliquant dans quel ordre compiler les programmes et expliquant quels sont les liens entre les différents fichiers.
La compilation se lance en tapant `make` dans le terminal. Que se passe-t-il ?
Pour corriger ce problème :
Ajoutez les lignes :

```
tab_double_io.o: tab_double_io.c ue.h
```

avant la ligne

```
main.o: main.c ue.h
```

et modifier la ligne

```
bulletin:  main.o
```

en

```
bulletin: main.o tab_double_io.o
```

Si maintenant vous écrivez `make`, vous ne devriez plus apercevoir de message d'erreur. Exécutez la ligne

```
./bulletin < notes.txt
```

et étudiez la structure du programme.

La note finale de contrôle continu se calcule de la façon suivante : la note de l'épreuve facultative remplace toutes les notes inférieures ou les absences à un contrôle. En cas d'absence à un contrôle, il est donc nécessaire que l'étudiant ait passé l'épreuve facultative (dans le cas contraire, il sera considéré comme défaillant).

Pour simuler ceci, nous disposerons de deux tableaux de `double` de même longueur, un pour les notes aux différents contrôles, que l'on notera `CC`, l'autre pour les coefficients, que l'on notera `coeff`. La longueur des tableaux

sera, évidement, égale au nombre de notes dans la matière, nous noterons ce nombre `nb_CC`. Dans le tableau des notes, les absences à un contrôle seront représentées par la macro-constante `DEF` (un nombre négatif). Si l'une des notes de la moyenne est `DEF` alors la moyenne est égale à `DEF` sinon elle est calculée par la formule

$$\texttt{moyenne} = \frac{\texttt{CC[0]} * \texttt{coeff[0]} + \texttt{CC[1]} * \texttt{coeff[1]} + \ldots + \texttt{CC[nb\_CC-1]} * \texttt{coeff[nb\_CC-1]}}{\texttt{coeff[0]} + \texttt{coeff[1]} + \ldots + \texttt{coeff[nb\_CC-1]}}$$

Les nombres minimaux et maximaux de notes par matière sont fixés par l'établissement, nous utiliseront les macro-constante `NB_MIN` et `NB_MAX` pour les représenter. Nous admettrons que le nombre `NB_MIN` est supérieur à 0, `NB_MIN < NB_MAX` et que le nombre `NB_MAX` n'est pas très grand.

Exercice 3

La fonction `est_defaillant` permet de décider si un étudiant est défaillant à partir des notes données dans le tableau qui est passé en paramètre. Un étudiant est défaillant si et seulement si l'une de ses notes vaut `DEF`.

1) Écrire la définition de la fonction dans le fichier `tab_notes.c`.

2) Rédigez correctement l'invariant de boucle.

3) Modifier le fichier `Makefile` afin que cette fonction soit utilisable dans le fichier `main.c`.

4) Modifier le fichier `main.c` pour tester cette fonction.

Exercice 4

1) Le numérateur de la formule s'appelle la somme pondérée des notes. Écrivez la définition de la fonction `somme_ponderee` dans le fichier `tab_notes.c` de façon à ce qu'il respecte les spécifications données dans le fichier `ue.h`.

2) Le dénominateur de la formule est la somme des coefficients. Écrivez la définition de la fonction `somme_coeff` dans le fichier `tab_notes.c` de façon à ce qu'il respecte les spécifications données dans le fichier `ue.h`.

3) Écrivez la définition de la fonction `moyenne_ponderee` dans le fichier `tab_notes.c` façon à ce qu'il respecte les spécifications données dans le fichier `ue.h`.

Exercice 5

Nous allons maintenant intégrer la note de l'épreuve facultative. Cette note vient remplacer toutes les notes qui lui sont inférieure. Rappelons qu'une note négative valant `DEF` signifie que l'étudiant est défaillant à une épreuve (n'importe quelle épreuve même la facultative).

1) Écrire le corps de la fonction `modif_note` qui permet d'intégrer la note de l'épreuve facultative en modifiant le tableau des notes passés en paramètre.

2) Modifier la fonction `main`, afin qu'elle affiche le résultat : `Défaillant`, `Ajourné` ou `Reçu`.

Exercice 6

Un étudiant, n'ayant pas encore passé son épreuve facultative, voudrait savoir la note minimale qu'il devrait obtenir à celle-ci pour valider la matière. Aidez-le en écrivant un programme.

Exercice 7

Nous n'avons pas pris en compte la note de TP dans nos calculs. Modifiez vos programmes pour la prendre en compte.