INF 201 - TP 3 - Programmation C

T. Grandpierre 1/2009

Rapport :

Ce qu'il faut rendre :

A la fin du TP il faudra archiver et compresser l'ensemble de votre travail (y compris le rapport avec vos noms) avec la commande ***cd ~*** puis puis ***cd INF201*** puis ***tar zcvf INF201_TP3_VotreNom.tgz TPC3***

Envoyer ce fichier par mail à t.grandpierre@esiee.fr avec comme sujet : ***INF201 TP3 Nom_des_élèves_du_binomes*** (-3 points si les consignes ne sont pas respectées).

Au plus tard le jour de la rentrée vous rendrez un rapport imprimé finalisé. Il doit au minimum contenir :

- les réponses aux questions posées,
- le code source des programmes demandés (en annexe si ils sont trop longs),
- des commentaires sur ces programmes : ce qu'ils sont censés faire, leur limite, les solutions choisies quand il y avait un choix à faire,
- les traces d'exécution de chaque programme (capture écran par exemple).

Un seul des trois rapports sera noté, le même pour tous les groupes. Il y a bien 2 rapports à rendre : une version électronique incluse dans l'archive envoyée par mail le jour du TP et une version papier.

Exercice 1 : Permutation des contenus de structures

Sous Linux, dans votre racine **créez** un **sous-répertoire *TPC3 (cd INF201, mkdir TPC3)***, puis un sous-répertoire ***Exo1-struct (cd TPC2, mkdir Exo-struct)***.

Reprendre le code de l'exercice du TP 2 et écrire une fonction qui permute le contenu de 2 structures. Pour cela il faut permuter les champs des structures un à un.

Remarque : vous savez que pour accéder au un champs d'une structure on utilise l'opérateur '.' mais si on possède un pointeur sur une structure nous avons vu encore cours qu'il faut utiliser l'opérateur "->" (flèche composée du signe moins et du signe supérieur) pour accéder aux champs de cette structure.

Exemple :

```
struct cmpx {float Re; float Im;} var1;

struct cmpx *add1;  /*Déclaration d'une variable (pointeur) servant à stocker l'adresse d'une
                      variable de type structure cmpx */

add1=&vart1;  /* La variable add1 contient l'adresse de var1 : add1 pointe donc maintenant
                  sur la variable var1 */

add1->Re=10.0;      /*on écrit 10.0 dans le champs Re de la structure pointée par add1*/
```

Exercice 2 : Lecture de caractère depuis le clavier

Créez un sous-répertoire ***Exo2-Moyenne (cd TPC2, mkdir Exo2-Moyenne)***.

Ecrire un programme qui gère une liste d'élèves :

1. Ce programme stocke dans un premier tableau à 2 dimensions une liste de noms d'élèves qui seront saisis individuellement. La fonction *scanf* sera utilisée pour la saisie clavier. La saisie d'un nom égale à FIN indiquera la fin de la saisie. La fonction *strcpy* sera utilisée pour copier la chaîne de caractère saisie dans le tableau de noms.
2. Ensuite le programme demandera la note de chaque élève qu'il stockera dans un second tableau.
3. Le programme affichera enfin le nom et la note de chaque élève et terminera en affichant la moyenne de la classe.

Notes :

Trouvez le nom de la bibliothèque contenant le prototype de *scanf* à l'aide de son manuel (*man scanf*) de même pour *strcmp* et *strcpy*.

Nous avons vu en cours que comme la fonction *printf*, la fonction *scanf* repose sur le principe d'un format %. Dans le cas de *printf* le format sert à indiquer le type de représentation de la donnée à afficher. Dans le cas du *scanf* le format indique le type de donnée à lire sur l'entrée standard (l'entrée standard est par défaut le clavier). Ainsi *scanf(''%d'', &var)* indique que l'on attend la saisie d'un entier et qu'il sera stocké à l'adresse de var. En effet, *scanf* ne peut fonctionner que par adresse puisse qu'il doit changer le contenu d'une variable : il a donc besoin de l'adresse de cette variable (dans le cas d'un tableau de caractères le nom suffit puisque le nom d'un tableau c'est l'adresse de son premier élement).

Comme nous ne connaissons pas à priori le nombre d'élèves dans un promo., vous fixerez arbitrairement les tailles des tableaux à l'aide d'un symbole PROMO_MAX de 50 élèves *(#define PROMO_MAX 50).* Nous pourrions utiliser l'allocation dynamique pour ne pas être limité comme c'est le cas ici, mais faisons simple pour cette fois! Un second symbole permettra de définir la taille maximum des noms.

Squelette du programme (A compléter et rendre dans le rapport)

```
Déclarer :
        tableau de nom
        tableau de notes
        indice de parcours i

/*1ère étape : saisie des noms */
Faire  {
        saisir chaîne de caractères
        stocker chaîne dans tableau
        incrémenter indice
```

} Tant que chaîne différente de FIN ET que indice inférieure à PROMO_MAX

/*2ème étape : saisie des notes */
/* 3ème étape : calcul et affichage de la moyenne */

Remarque : pour un programme plus complet, l'usage d'une structure serait plus adapté, elle pourrait avoir des champs noms, prénoms, notes etc. Il suffirait ensuite de créer un tableau de structures.

Exercice 3 : Création d'un fichier et écriture dans ce fichier

Dans l'exercice précédent des tableaux ont été créés et rempli, mais à l'issue du programme leur contenu ont été perdu. Dans cet exercice on va sauvegarder le contenu des tableaux dans un fichier. Ce fichier sera utilisé dans l'exercice 4 pour restaurer le contenu des tableaux.

Pour sauvegarder le contenu des tableaux il suffit de les parcourir et d'écrire chaque élément dans un fichier. Les éléments seront donc écrit les uns à la suite des autres dans le fichier.

On vous demande donc d'écrire une fonction *sauvegarde* qui reçoit les tableaux à sauvegarder en argument ainsi que le nombre d'éléments utiles de ces tableaux. Cette fonction retournera 0 en cas de succès et autre chose en cas d'échec (problème d'accès au fichier, voir ci-dessous).

1. Avant de pouvoir écrire dans un fichier il faut d'abord l'ouvrir (et si il n'existe pas le créer). La fonction *fopen* permet d'ouvrir et créer un fichier. Elle retourne un pointeur sur une structure de type *FILE*. Elle prend en argument le nom du fichier (cela peut inclure son chemin d'accès) et des droits d'accès au fichier (droit en lecture 'r' pour read, en écriture 'w' pour write… voir le détail avec le manuel de *fopen*.

Exemple :

```
File *descripteur_fichier1 ;
fichier1 = fopen("mon_fichier","w");  /* (w indique ouverture en écriture, si le fichier existe il est vidé, si il n'existe pas il est créé)*/
```

Attention en cas d'échec de *fopen* (i.e. plus de place sur le disque, nom de fichier invalide etc.) *fopen* retourne NULL : il faut donc TOUJOURS faire suivre *fopen* d'un test.

2. Une fois ouvert il est possible de lire ou écrire dans le fichier : la fonction *fprintf* effectue un *printf* dans le fichier ouvert : *fprintf* à donc un argument supplémentaire qui est le descripteur de fichier. Chaque appel à *fprintf* ajoute les éléments les uns à la suite des autres dans le fichier. Nous verrons la lecture dans l'exercice 3.

```
fprintf(fichier1, "%i", var1);
```

3. Quand il n'est plus nécessaire d'accéder au fichier, il faut fermer le fichier avec la fonction `fclose(fichier1);`

Remarque : le fichier ainsi créé peut être ouvert et lu avec un éditeur de texte standard car c'est un fichier ascii et non binaire.

Exercice 4 : lecture d'un fichier

En utilisant la fonction *fscanf* écrire un programme capable de relire et d'afficher le contenu du fichier.

N.B. : regardez bien dans le manuel de f*scanf* quels sont les caractères séparateurs de cette fonction.

Ajouter une fonction pour que le premier programme classe alphabétiquement le tableau de noms (en mettant à jour celui de notes). Pour mémoire, la fonction *strcmp* permet de comparer des chaînes de caractères.

Pour aller plus loin (pendant le TP si vous avez fini !) :

L'année prochaine, vous utiliserez les listes chaînées lors des cours d'algorithmique. N'ayant pas eu le temps de les traiter cette année, je vous conseil un très bon tutorial sur ce site : http://www.siteduzero.com/tutoriel-3-36245-les-listes-chainees.html

N'hésitez pas à me contacter si vous rencontrer des difficultés.

Mémo sur les pointeurs :

Nom_de_type ***X ;** /* **L'étoile** associée à la déclaration d'une variable déclare une variable (case mémoire X) qui contiendra l'adresse d'un élément de type "Nom_de_type" */

a = ***X ;** /* L'opérateur **étoile** quand il est à droite de l'égalité permet de récupérer ce qu'il y'a à l'adresse pointée par X (ce qu'il y a à l'adresse contenue dans X) */

***X**= b; /* L'opérateur **étoile** quand il est à gauche de l'égalité permet de modifier ce qu'il y'a à l'adresse pointée par X (ce qu'il y a à l'adresse contenue dans X) */

X=&c**;** /* L'opérateur & permet de récupérer l'adresse d'une variable, celle de c ici */

(et l'étoile précédée d'un / indique un début de commentaire, et l'étoile suivie d'un / la fin d'un commentaire !!)

Enfin ne pas confondre le prototype d'une fonction qui "attend" un pointeur avec l'appel de cette fonction, exemple pour permute que vous avez codé en TP2 :

Prototype : *void permute (int **a1, int * a2);* /* permute attend 2 adresses d'entiers a1, a2 */
Utilisation : *permute (&x, &y);* /* on appel permute en lui passant les adresses de x et y , ces adresses iront dans a1 et a2 */