

L'objectif de ce TP est de gérer une liste de scores composée d'un pseudo du joueur et du score réalisé. Cette liste sera sauvegardée dans un fichier "score.txt". Ce sera l'occasion de réviser ou vous familiariser avec les notions de Listes Chaînées, de manipulation de chaînes de caractères, de Compilation séparée, d'Ecriture et d'Utilisation d'un Makefile, de Gestion de fichiers, d'Algorithme de tris et enfin d'une première prise en main de la bibliothèque graphique proposée.

Important : Il est impératif de tester chaque fonction écrite dans la fonction main() pour s'assurer de son bon fonctionnement. Il n'est pas raisonnable d'écrire tout l'exercice et de se lancer dans l'exécution à la fin. On valide chaque brique séparément et on avance pas à pas.

PREPARATION

- Avant de venir à ce TP, il est obligatoire d'**avoir révisé les supports de cours en langage C** sur les bases de la syntaxe, l'allocation dynamique, la manipulation des chaînes de caractères, les listes chaînées, ect. ainsi qu'**avoir lu les annexes de ce document**.

RAPPEL

- *A propos des chaînes de caractères* : Comme vu dans l'annexe B, la fonction standard scanf(char *format, ...) permet de saisir des chaînes de caractères. Mais attention, pour cette fonction le caractère blanc (espace) est également un caractère séparateur. scanf() ne peut donc saisir une chaîne comportant plusieurs mots séparés en une seule fois. Une autre fonction de la librairie stdio.h permet de saisir une chaîne composée de plusieurs mots : gets(char *). Cette fonction cependant est d'un emploi dangereux car aucun contrôle de longueur de la chaîne saisie ne peut être fait. On lui préférera donc une troisième fonction de la même librairie, fgets(char*, int, FILE*), mais qui sera expliquée plus tard avec les fonctions de manipulation de fichiers. Voici un exemple d'utilisation de cette fonction :

```
#include <stdio.h>
void main(void)
{
    char chaine[21];
    printf("saisir une chaine d'au plus 20 caractères : ");
    fgets(chaine, 20, stdin);
    printf("chaine saisie = %s\n", chaine);
}
```

Si la longueur de la chaîne tapée par l'utilisateur devait dépasser 20, seuls les 20 premiers caractères seraient pris en compte. Aucune erreur de dépassement de tableau n'est donc possible.

TRAVAIL A REALISER

Liste Chaînée et manipulation de chaînes de caractères

Soit la structure suivante :

```
typedef struct liste
{
    int score;
    char *pseudo;
    struct liste *suivant;
}LISTE;
```

- Ecrire la fonction **LISTE *creer_maillon()** qui alloue de la mémoire pour un élément de type LISTE et retourne son adresse. On initialisera impérativement suivant à NULL. Question : Pourquoi retourne-t-on l'adresse de l'élément alloué ? aurais-t-on pu déclarer tout simplement une variable dans la fonction et retourner son adresse ? pourquoi ?. Pourquoi initialiser le champ suivant à NULL?
- Ecrire la fonction **VOID saisir_maillon(LISTE*)** qui demande un pseudo et un score à l'utilisateur, réalise la saisie de ces deux informations et remplit les champs correspondant dans le maillon passé en argument d'entrée. Attention à allouer le pointeur `char* pseudo`.
- Ecrire la fonction **void afficher_maillon(LISTE*)** qui affiche le contenu d'un élément de type LISTE passé par adresse et ne retourne rien. Question : Pourquoi ne retourne-t-on rien ?
- Ecrire la fonction **LISTE * chainage_avant(LISTE*,LISTE*)** qui reçoit la tête de la liste chaînée et l'adresse du maillon à insérer et procède à son insertion à la première position. On retourne l'adresse du premier maillon de la liste chaînée. Question : pourquoi retourne-t-on l'adresse de la liste chaînée ?
- Ecrire la fonction **LISTE * chainage_arriere(LISTE*,LISTE*)** qui reçoit la tête de la liste chaînée et l'adresse du maillon à insérer et procède à son insertion à la dernière position. On retourne l'adresse du premier maillon de la liste chaînée.
- Ecrire la fonction **void afficher_liste(LISTE*)** qui reçoit la tête de la liste chaînée et affiche le contenu du champ nb de tous les maillons.
- Ecrire la fonction **void supprimer_liste(LISTE*)** qui reçoit la tête de la liste chaînée et procède à la libération mémoire de tous ses maillons.
- Ecrire la fonction **main()** qui permet à l'utilisateur de créer une liste chaînée de dix éléments. Afficher le contenu de la nouvelle liste puis supprimer tous les maillons alloués.

Compilation séparée, Ecriture et Utilisation d'un Makefile

- Créer un fichier « `liste.h` » qui contiendra la déclaration de la structure LISTE ainsi que les déclarations des fonctions **xxx_liste()** gérant la liste que vous avez écrits précédemment.
- Créer un fichier « `liste.c` » qui contiendra toutes les définitions des fonctions **xxx_liste()**. Ne pas oublier la directive `#include "liste.h"` ainsi que celles des bibliothèques dont vous avez besoin (`stdio,...`).

- Créer un fichier « `main.c` » qui contiendra la fonction `main()`. Ne pas oublier la directive `#include "liste.h"` ainsi que celles des bibliothèques dont vous avez besoin (`stdio,...`).
- Créer le fichier « `Makefile` » correspondant à ce programme. Compiler avec l'aide de l'utilitaire `make`.

Gestion d'un fichier de score

- Créer un fichier « `scores.txt` » dans le même répertoire où se trouve votre programme contenant les lignes suivantes correspondantes aux *pseudos* et aux *scores* réalisés dans des parties précédentes :

```
Toto
12000
Alfred de la Montagne
11234
pipo
7601
nom prenom
6509
Newbie
12
```

- Ecrire la fonction **`LISTE *lecture_fichier_score(char*)`** qui reçoit une chaîne de caractères correspondant au chemin (relatif) et au nom du fichier « `scores.txt` », ouvre ce fichier, lit le fichier en créant et remplissant des maillons (contenant les informations *pseudo* et *score*) en les insérant à fur et à mesure dans une liste chaînée (que l'on renverra en sortie de la fonction). Si le fichier n'existe pas ou est vide, il conviendra de renvoyer la valeur `NULL`.
- Ecrire une fonction **`int ecriture_fichier_score(LISTE*, char*)`** qui reçoit une liste chaînée de scores ainsi que le chemin (relatif) et le nom d'un fichier (ex : « `scores.txt` ») et écrit dans ce fichier (s'il existe déjà, le fichier devra être écrasé) l'ensemble des scores contenus dans la liste chaînée. Si l'écriture s'est bien passée, la fonction renverra le nombre de scores enregistrés dans le fichier et -1 sinon.
- Réécrire la fonction **`main()`** de sorte à créer la liste chaînée en fonction du contenu du fichier « `scores.txt` », puis demander à l'utilisateur de saisir et ajouter un nouvel élément, d'afficher la nouvelle liste de scores et enfin d'enregistrer le nouvel ensemble de scores dans le fichier « `scores.txt` » avant de libérer la mémoire.
- Faire en sorte que la liste des scores soit triée dans l'ordre décroissant des scores. Le choix de la méthode de tri est laissé au choix du programmeur (Cependant, en supposant la liste déjà triée précédemment, il est plus judicieux d'utiliser un tri par insertion). Pour chaque nouveau score inséré, on supprimera la plus mauvais des scores (de sorte à conserver par exemple que le Top 5 des scores). Cela requerra de créer des fonctions de recherche selon un critère (inférieur au score du nouveau maillon), d'insertion d'un nouveau maillon dans la liste chaînée, de suppression d'un maillon,...

Prise en main de la bibliothèque graphique

- En vous aidant des premiers exemples de la bibliothèque graphique fournie, modifier l'affichage des scores de sorte que la liste soit affichée sur une fenêtre graphique X11.