IMA 3ème année

Programmation Avancé

TP2 Listes contiguës, Fichiers

1 Objectifs

— Savoir utiliser les listes contiguës en C.
— Utiliser les opérations de lecture et d'écriture sur les fichiers en C.
— Utiliser les arguments de la ligne de commande.

Contexte et préparation : Le contexte de ce TP est le même que celui du TP1, à savoir une application de gestion d'annuaire.

IMPORTANT !
Ce TP utilise les types de données et les fonctions du TP1.

Avant de commencer

— Créer un répertoire TP2 dans votre répertoire de PA et y copier les fichiers `entiers.txt`, `annu.txt` et votre `.c` du TP précédent (renommez ce dernier en `tp2.c`).
— Récupérer sur le compte de Walter Rudametkin, les fichiers `moyenne_fichier.c` et `ldc.c` présents à l'adresse : `~wrudamet/public/IMA3/TP2/`

2 Questions du TP (à faire impérativement)

2.1 Chargement à partir d'un fichier

La version demandée dans le précédent TP ne permet pas les opérations interactives sur l'annuaire, l'entrée standard étant redirigée pour le chargement des données. Pour libérer l'entrée standard, nous allons donc manipuler explicitement le fichier de données en C.

1. En vous inspirant de l'exemple `moyenne_fichier.c`, écrire une nouvelle fonction de chargement de l'annuaire qui permet de lire les données à partir d'un fichier texte dont le pointeur est passé en paramètre. Pour cela, on commencera par modifier la fonction de lecture d'une personne. Tester à partir du `main`, en ouvrant le fichier avec :
```
FILE *fp ;
fp = fopen("annu.txt", "r") ;
```
2. Utiliser les paramètres de la fonction `main` pour vérifier que l'argument a été correctement fourni au lancement et récupérer le nom du fichier de données passé en argument (voir l'exemple `ldc.c`). À ce stade, le lancement de votre programme annuaire doit se faire de la façon suivante : `./annuaire annu.txt`

2.2 Recherche dans l'annuaire

Dans cette partie, il faut que l'annuaire soit trié selon les noms des personnes. Si vous n'avez pas écrit de fonction de tri dans le premier TP, modifiez le fichier `annu.txt` pour qu'il soit trié par nom.

Remarque : La recherche dichotomique dans un tableau trié est un algorithme classique vu en TD.

1. Écrire une fonction `rech_dicho` qui prend en paramètre un nom de personne et un annuaire, effectue une recherche dichotomique de cette personne dans l'annuaire et retourne l'indice de rangement de la personne dans la liste contiguë si elle est présente, `-1` sinon.
2. Tester la fonction `rech_dicho` en demandant un nom de personne à l'utilisateur et en affichant après le retour de la fonction les informations trouvées (nom, prénom, date de naissance, numéro de téléphone) ou *inconnu* sinon.

2.3 Modification de l'annuaire et sauvegarde dans un fichier

1. Écrire une fonction qui donne la possibilité à l'utilisateur de modifier le numéro de téléphone d'une personne recherchée dans l'annuaire comme précédemment. Tester la fonction.
2. L'annuaire pouvant être modifié, écrire une fonction permettant de sauvegarder l'annuaire dans un (autre) fichier texte, selon le même format que celui du fichier `annu.txt` donnée en argument.

3 Questions s'il vous reste du temps

On va faire un joli menu pour l'application annuaire :

1. Écrire une fonction menu qui propose à l'utilisateur les différentes fonctionnalités possibles de l'application (affichage de l'annuaire, recherche d'une personne, modification du numéro de téléphone d'une personne donnée, quitter l'application avec ou sans sauvegarde de l'annuaire) et qui retourne le choix de l'utilisateur.
2. Modifier la fonction `main` de l'application pour afficher le menu utilisateur et effectuer le traitement correspondant au choix de l'utilisateur.
3. N'oubliez pas de boucler sur le menu jusqu'à ce que l'utilisateur demande de quitter l'application.

4 Annexes

```c
/* Source Code From Laure Gonnord, modif par Walter Rudametkin */

/* À tester avec :
   ./ldc
   ./ldc toto
   ./ldc toto 3
   ...
 */

#include <stdio.h>

int main(int argc, char* argv[])
{
  int i=0;
  printf("Bonjour, il y a %d argument(s) à cette commande! \n",argc);

  while(i<argc)
  {
    printf("argument %d : %s\n",i,argv[i]);
    i++;
  }

  printf("c'est fini !\n");

  return 0;
}
```

ldc.c

```c
/* Source code from B. Carre, modif by L. Gonnord and Walter Rudametkin */

/* Calcule la moyenne des entiers lus dans le fichier "entiers.txt"
 *
 * Compilation : gcc -o moyenne_fichier moyenne_fichier.c -Wall -Wextra -pedantic
 *
 * Utilisation : ./moyenne_fichier
 */

#include <stdio.h>

int main(void)
{
    int i;          /* nbre lu */
    float somme=0.0;/* la somme des nombres lus */
    int nbre=0;     /* le nombre d'entiers lus */

    FILE* fd ;
    fd = fopen("entiers.txt","r"); //ouverture en lecture
    if (fd!=NULL)
    {
        //fichier ouvert avec succes !
        fscanf(fd,"%d", &i);
        while (!feof(fd))
        {
            //feof APRES la premiere lecture.
            nbre=nbre+1;
            somme=somme+i;
            fscanf(fd,"%d", &i);
        }
        //impression de la moyenne
        printf("la moyenne est : %f\n", somme/nbre);
        fclose(fd);//fermeture
    }

    return 0;
}
```

moyenne_fichier.c