IMA 3ème année

Programmation Avancé

TP1 Structures, Listes contiguës, Redirections

1 Objectifs

— Savoir déclarer un type "structure" en C, déclarer des variables de ce nouveau type et utiliser ces variables dans des fonctions C.
— Savoir implémenter les listes contiguës en C en utilisant les structures, écrire les fonctions de base pour ajouter et retirer des éléments.
— Utiliser les redirections de l'entrée standard pour automatiser certains traitements de données.

Remarque : Cet énoncé s'appuie sur les énoncés des TD1 et TD2 de PA, certains des algorithmes ont été décrits en pseudo-code.

Contexte Nous travaillerons sur un exemple simple de gestion d'un annuaire de personnes. Chaque personne est représentée par une structure contenant les informations suivantes :
— son nom, prénom et numéro de téléphone sous la forme de chaînes de caractères *de taille statiquement fixée*.
— sa date de naissance sous la forme d'une structure jour, mois, année (entiers). Un annuaire de personnes sera représenté par une liste contiguë de `MAX_PERSONNES` (constante valant 100, par exemple).

2 Questions du TP (à faire impérativement)

IMPORTANT !
Toutes les questions de cette partie sont nécessaires pour le TP2. Si vous ne finissez pas le TP1 dans la séance, il faudra le finir chez vous, sous peine de ne pouvoir faire le TP2.

Toutes les questions seront testées au fur et à mesure (appels dans le main), *comme d'habitude* ! Avant de commencer, créer un répertoire de travail pour la Programmation Avancée, et créer un sous-répertoire TP1 *à la ligne de commande*.

2.1 Type `Date`

Dans un fichier `annu1.c` :

1. Définir le type `Date` permettant de représenter une date de naissance.
2. Écrire une fonction `lire_date` qui permet de lire une date au clavier et retourne une valeur de type `Date`.
3. Écrire une fonction `affiche_date` qui affiche une date passée en paramètre. Passer à la suite seulement si ces fonctions sont testées et fonctionnent (écrire donc un `main`).

2.2 Types `Personne` et `Annuaire`

1. Définir le type `Personne`.
2. Écrire une fonction `lire_personne` qui permet de lire les données d'une personne au clavier.
   La fonction aura le prototype suivant :
   ```
   int lire_personne(Personne* ptr_pers)
   ```
   Après récupération des informations de `Personne` (nom, prénom,...) à l'aide de la fonction `scanf`, cette fonction retournera la valeur entière `0`.
3. Écrire une fonction `affiche_personne` qui affiche une personne `p` passée en paramètre. Tester.
4. Définir le type `Annuaire`, et déclarer un annuaire dans le `main`.

2.3 Initialisation de la structure `Annuaire`, et utilisation

Pour initialiser la structure `Annuaire`, et afin d'éviter les longues saisies à la main à chaque lancement de notre programme, nous allons écrire des informations dans un fichier texte, en mettant une information par ligne. Dans ce TP, la lecture de ces informations sera faite à partir de `scanf` (lecture sur le terminal), mais ces informations seront écrites sur le terminal à travers une redirection de l'entrée standard vers un fichier fourni (cf TP5 de Programmation Structurée).

Comprendre les redirections (révision) Récupérer sur le compte de Walter Rudametkin les fichiers `redirect.c` et `entiers.txt` disponibles à l'adresse : `~wrudamet/public/IMA3/TP1/`
Regarder le code source de `redirect.c`, compiler, comprendre le code, le tester sans redirection (arrêter en faisant control+d), puis en lançant `./nomdubinaire < entiers.txt`.

Récupération des données de l'annuaire

1. Récupérer l'annuaire `annu.txt` à la même adresse que précédemment.
2. En s'inspirant de `redirect.c`, écrire une fonction qui lit une suite de personnes (par redirection de l'entrée standard sur le fichier texte `annu.txt` précédemment récupéré) et les range dans un annuaire par ajout en queue. On écrira une fonction `construire_annuaire` qui effectuera des appels à `lire_personne`. La fonction `lire_personne` doit donc être modifiée pour pouvoir retourner `1` (et terminer) si la fin de fichier est atteinte lors de l'appel à `scanf` pour un nom de personne.
3. Écrire une fonction qui affiche toutes les personnes d'un annuaire donné avec leur indice de rangement.
4. Tester !

3 Questions s'il vous reste du temps

Nous allons trier l'annuaire :

1. Écrire une fonction `compare_dates` qui compare 2 dates `d1` et `d2` données et retourne :
$$\texttt{compare\_dates(d1,d2)} = \begin{cases} 0, & \text{si } \texttt{d1=d2} \\ 1, & \text{si } \texttt{d1>d2} \\ -1, & \text{si } \texttt{d1<d2} \end{cases}$$
2. Écrire une fonction qui trie un annuaire selon la date par la méthode du tri bulle, telle que vue en TD.
3. Écrire une fonction qui trie un annuaire selon le nom.
4. Tester : trier l'annuaire obtenu précédemment et affichez le (avant et après) !

4 Annexes

```c
#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 */
  while (scanf("%d", &i) != EOF) {
    nbre = nbre+1;
    somme = somme+i;
  }
  /* impression de la moyenne */
  printf("la moyenne est : %.2f\n", somme/nbre);
  return 0;
}
```

redirect.c

```
12
20
8
6
16
```

entiers.txt

```
Zebulon
Michel
03
04
1980
0321420156
Alibaba
Arthur
10
12
1980
0452136521
Martin
Martine
25
03
1985
0215329545
```

annu.txt