S3E3C

Listes doublement chaînées

Licence Informatique — Programmation Nécessaire

Stéfane

Table des Matières

1 Introduction

Lors de ce TP on va s'intéresser aux réseaux sociaux grâce auxquels "on est plus jamais seul.e". Ces réseaux sont décrits par des graphes d'amitiés qui relient des personnes. Un tel graphe a pour sommets les personnes et pour arcs les relations d'amitié.

Chaque personne est décrite par :

- son nom,
- son prénom,
- sa date de naissance au format `JJ/MM/AAAA`,
- la liste des personnes avec qui elle est amie.

Déclaration du TA `person_t` et ses fonctions membres.

```c
/*
   Modélisation statique des chaînes de caractères
   afin d'utiliser au mieux les fichiers binaires
*/
#define LG 40

typedef struct {
    char name[LG];
    char forename[LG];
    struct {
        int day;
        int month;
        int year;
    } birth_date;
    list_t * friends;
} person_t;

person_t * new_person ();
person_t * scan_person ();
void free_person ( person_t ** ptrP, bool del_friends );

int cmp_person (person_t * P1, person_t * P2 );
void print_person ( person_t * P );

person_t * stream_2_person ( FILE * stream, stream_mode_t mode );
// void person_2_stream ( person_t * P, FILE * fd, stream_mode_t mode );
```

Chaque amitié est décrite par :

- Une première personne `A`,
- Une seconde personne `B`.

Déclaration du TA `friendship_t` et ses fonctions membres.

```
typedef struct {
    person_t * A;
    person_t * B;
} friendship_t;

friendship_t * new_friendship ();
friendship_t * scan_friendship ();
void free_friendship ( friendship_t ** ptrFriend, bool del_persons );

int cmp_friendship ( friendship_t * F1, friendship_t * F2 );
void print_friendship ( friendship_t * F );

friendship_t * stream_2_friendship ( FILE * stream, stream_mode_t mode, list_t * Lpers, list_t * Lfriends
    );
// void friendship_2_stream ( FILE * stream, stream_mode_t mode, list_t * Lfriends );
```

2 Listes génériques homogènes doublement chaînées

Depuis le dernier TP, vous disposez des outils de manipulation des listes génériques homogènes simplement chaînées que vous allez étendre au doublement chaînées.

Vous allez lire deux fichiers :

- Le premier fichier contient toutes les personnes du réseau considéré. Il va vous servir à construire la liste des personnes ;
- Le second fichier contient toutes les amitiés ayant cours dans ce réseau. Il va vous servir à construire la liste des amitiés ainsi que la liste des amitiés de chaque personne.

3 Liste de personnes

Grâce au premier fichier vous construisez la liste des personnes sans pour autant connaître leurs amitiés. Ces amitiés vont être enregistrées lors de la lecture du second fichier.

Chaque personne est décrite dans le fichier par son nom, son prénom et sa date de naissance au format `jj/mm/aaaa`. Remarquez que les amitiés n'y sont pas présentes. C'est le rôle du second fichier.

4 Liste des amitiés

À la lecture du fichier des amitiés, chaque ligne représente une amitiés entre deux personnes `A` et `B` grâce à leurs noms et prénoms. Vous construisez alors la liste des amitiés en lisant ligne par ligne le fichier et en recherchant dans la liste des personnes les personnes concernées. Une amitié sera un couple de références (pointeurs) sur les personnes concernées.

4.1 Mise à jour des personnes

Une fois la liste des amitiés construite ou en même temps que celle-ci se construit, vous devez enregistrer les amitiés dans les listes des amitiés de chaque personnes. De cette manière, vous construisez un graphe accessible depuis la liste des personnes (les sommets du graphe) ou depuis la liste des amitiés (les arcs et chemins du graphe)

5 L'algorithme principal

Faites les essais comme indiqués dans le code.

```c
#include <stdio.h>
#include <ctype.h>

#include "list.h"
#include "person.h"
#include "friendship.h"
#include "io.h"

int main(int argc, char *argv[]){
  #ifdef _DEBUG_0
    printf ( "main starts with argv[1] = %s and argv[2] = %s\n", argv[1], argv[2] );
  #endif
  /******
    VERSION LECTURE D'UN FICHIER DE PERSONNES
    À FAIRE UNE FOIS LA VERSION CLAVIER FONCTIONNANT

  list_t * Lpers = stream_2_persons ( argv[1], TEXT );
  ******/
  /* VERSION LECTURE AU CLAVIER */
  list_t * Lpers = new_list ();
  int OK = 1;
  while ( OK ) {
    char choix;
    person_t * P = scan_person ();
    print_person (P);
    insert_ordered ( Lpers, P, &cmp_person );
    do {
      printf ( "Une aute personne ? (O/N) "), scanf ( " %c", &choix);
      choix = toupper (choix);
    } while (choix != 'N' && choix != 'O' );
    OK = choix == 'O';
  }
  view_list ( Lpers, &print_person, "_______Liste des personnes_______" );

  /******
    VERSION LECTURE D'UN FICHIER DE PERSONNES
    À FAIRE UNE FOIS LA VERSION CLAVIER FONCTIONNANT

  list_t * Lfriends = stream_2_friendships ( argv[2], TEXT, Lpers );
  ******/
  /* VERSION LECTURE AU CLAVIER */
  list_t * Lfriends = new_list ();
  int OK = 1;
  while ( OK ) {
    char choix;
    friendship_t * F = scan_friendship ();
    print_friendship ( F, &print_person );
    queue ( Lfriends, F );
    do {
      printf ( "Une aute amitié ? (O/N) "), scanf ( " %c", &choix);
      choix = toupper (choix);
    } while (choix != 'N' && choix != 'O' );
    OK = choix == 'O';
  }
  view_list ( Lfriends, &print_friendship, "_______Liste des amitiés_______" );
  return 0;
}
```