Université de Strasbourg
UFR de mathématique et d'informatique

L2 informatique
quinquennal 2018-2023

Structures de données et algorithmes

TP : Clans et listes

Clans et listes

Dans un jeu de rôles multi-joueurs, on veut modéliser la situation suivante. On considère une population de joueurs qui est organisée en clans : une personne donnée ne peut appartenir qu'à un clan et toute personne appartient à un clan (en mathématique, on dit qu'on a une partition de la population). Le "chef" du clan est aussi appelé son représentant. A priori, il ne change pas sauf lorsqu'il y a fusion de deux clans, alors l'un des deux représentants des anciens clans devient représentant du nouveau. Des personnes peuvent rejoindre ou quitter le jeu, elles peuvent aussi quitter ou rejoindre un clan et les clans peuvent fusionner, mais pas se couper en deux de manière simple.

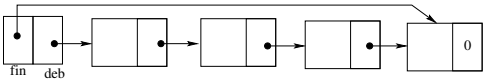
1. (Personne) Une personne est définie, pour simplifier, par un identifiant (un entier), un âge (un entier) et un genre (un caractère qui est soit 'h', soit 'f'). Il n'y a qu'une personne pour un identifiant donné. On traduit ceci par la spécification très simple :

```
Spec PERSONNE étend BASE *** on considère connus Nat, Bool, Car, Chaine ...
Sortes Genre Personne
Opérations                                          Axiomes
h f : -> Genre                                   Vars i a : Nat, g : Genre
per_mk : Nat Nat Genre -> Personne                    per_id(per_mk(i,a,g)) = i
per_id ; Personne -> Nat                              *** à compléter
per_age : Personne -> Nat
per_genre : Personne -> Genre
```

a) Complétez la spécification donnée en ajoutant les axiomes manquants.

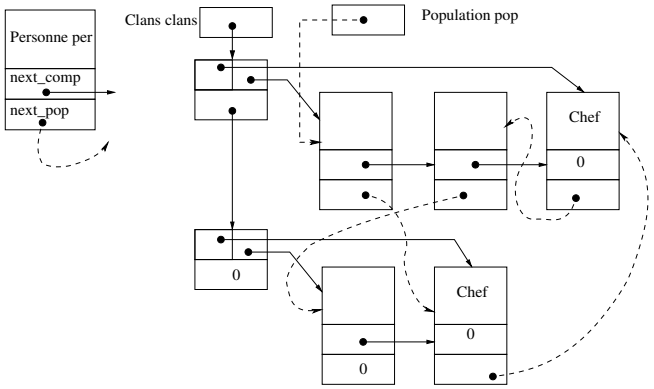
b) Définissez le type de données `Personne` et implantez les opérations de la spécification précédente.

2. (Clans)
En ce qui concerne les clans, deux opérations sont importantes : qui est le représentant du clan et la fusion de deux clans. En supposant qu'on ait une sorte `Clan`, donnez la signature de ces deux opérations. On choisit d'implanter la liste des membres du clan par chaînage simple avec un pointeur nul pour marquer la fin. L'entrée dans la liste est constitué de deux pointeurs l'un sur la tête et l'autre sur la fin ? Expliquer en quoi cette implantation est pratique.



3. (Population et clans)

Pour éviter la dispersion des données, on regroupe l'implantation de la population et des clans dans un même schéma : un clan sera un couple de pointeurs sur le premier et le dernier (le représentant) de la liste, la liste des clans sera une liste de pointeurs sur les clans. La population est une liste triée de toutes les personnes présentes dans le jeu (en pointillés sur le dessin). Une personne dans le jeu sera donc représentée comme à la figure de gauche et l'ensemble général est représenté à la figure de droite. Par ailleurs, on considérera les définitions `C` suivantes :




```
typedef struct stPer { Personne per ;              // les données de la personne
                       struct stPer *next_comp;    // le compagnon suivant dans le clan
                       struct stPer *next_pop;     // la personne suiv. dans la population
                     } *Population, Membre;        // plusieurs noms pour la lisibilité
typedef struct stClan { Membre * premier ;         // début de la liste du clan
                        Membre * dernier ;         // le dernier est le représentant
                        struct stClan *next_clan; // chaînage
                      } *Clans;
typedef struct { Population pop;                   // liste de la population entière
                 Clans clans;                      // liste des clans
               } Jeu;
```

Les questions suivantes sont données dans un ordre de difficulté plus ou moins croissant.

a) Au début d'un jeu, il n'y a aucun joueur et aucun clan. Programmez une fonction de nom `jeu-vide()` qui retourne un nouveau jeu.

b) Programmer une fonction de nom `Membre * jeu_adresse_chef(Jeu j, Membre *mclan )` qui donne l'adresse du chef d'un clan à partir de l'adresse d'un membre du clan.

c) Programmez deux fonctions de prototypes respectifs `void jeu_print_pop(Jeu j)` et `void jeu_print_clan(Jeu j, Membre *mclan)` qui affichent respectivement la liste des joueurs et la liste des membres d'un clan donné par l'adresse d'un membre.

d) La fusion de deux clans se fait par concaténation des listes correspondantes. Le deuxième clan de la fusion disparaît. Programmez cette opération de prototype `Jeu jeu_fusion(Jeu j, Clans clan1, Clans clan2)`.

e) Quand on ajoute une personne dans le jeu, elle constitue un clan à elle seule. Programmez la fonction de prototype `Jeu jeu_adj_joueur(Jeu j, Personne p)` qui réalise cette opération (attention, pour vérifier que l'identifiant n'est pas déjà utilisé, on utilise une insertion ordonnée dans le jeux. Si l'identifiant est déjà utilisé, on signale une erreur).

f) Programmez une fonction qui implante la sortie du jeu d'un joueur.