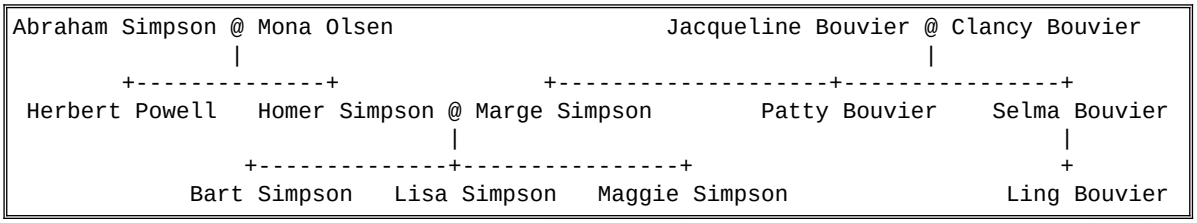
Programmation 1
2022-2023

TM11

Généalogie : Des tableaux et des structures

On souhaite représenter l'arbre généalogique d'une famille à l'aide des structures. Par exemple, l'arbre de la famille Simpson sur plusieurs générations est le suivant.



Pour ce faire on utilisera une structure `Individu` comportant quatre champs : `nom` et `prenom` de type chaîne de caractères, ainsi que `pere` et `mere` de type `Individu`. La famille entière sera représentée par un tableau d'`Individu`.

Question 1.1:

Dessiner sur une feuille le contenu de la mémoire permettant de représenter la famille Simpson. Pour limiter la taille du dessin, on se contentera de Marge, Homer et de leurs enfants.

Question 1.2:

Écrire
(1) une fonction `creerFamilleSimpson` qui crée et retourne un tableau correspondant à la famille entière ;
(2) une procédure `afficherIndividu` qui reçoit en paramètre un `Individu` et affiche son prénom, son nom et les prénoms et noms de son père et de sa mère ;
(3) une procédure `afficherFamille` qui reçoit en paramètre un tableau représentant une famille et en affiche tous les membres en utilisant la fonction `afficherIndividu`;
(4) une fonction principale `main` qui teste les trois fonctions précédentes.

Question 1.3:

Définir une fonction `rechercherMembre` qui prend en paramètre un tableau d'`Individu` ainsi q'un prénom et un nom et retourne le membre de famille correspondant. Lorsque la fonction ne trouve pas dans le tableau le membre de famille demandé, elle renvoie une référence `null`.

Exercice 2: Quelques pas vers le haut ♫

Question 2.1:

Écrire une fonction `parents` qui prend en paramètre un individu et retourne un tableau de deux cases contenant son père et sa mère. On laissera les cases à `null` si nécessaire. *N'hésitez pas à enrichir votre famille de test pour tester raisonnablement votre fonction.*

Question 2.2:

De façon similaire à la question précédente, écrire une fonction `grandParents`, qui retourne cette fois-ci un tableau de quatre cases. Ne pas oublier le cas où le père et/ou la mère sont `null`.

Question 2.3:

De façon similaire aux deux questions précédentes, écrire une fonction `mereDeMere4` qui retourne cette fois-ci un tableau de quatre cases contenant la mère, la grand-mère maternelle, la mère de la grand-mère maternelle et la grand-mère maternelle de la grand-mère maternelle. Ne pas oublier le cas où l'une de ces individus est laissé à `null`.

Exercice 3: Quelques pas vers le bas ♫

Question 3.1:

Écrire une fonction `rechercherEnfants` qui prend en paramètre un individu et un tableau d'individu qui représente sa famille, et qui retourne un tableau de la bonne taille contenant les enfants de cet individu. On pourra faire une première recherche pour calculer le nombre d'enfants, puis créer le tableau de la bonne taille et refaire une recherche pour remplir ce tableau.

Question 3.2:

Afin d'éviter de recalculer en permanence les enfants, ce qui demande une recherche dans toute la famille à chaque fois, on souhaite calculer les enfants de tout le monde et les mémoriser dans la structure `Individu`. Cela nous permettra de calculer les frères et les oncles de façon plus efficace. Ajouter un cinquième champ `enfants` à votre structure et écrire une fonction `initEnfants` qui prend en paramètre une famille et qui remplit le champ `enfants` de chaque membre en utilisant la fonction `rechercheEnfants` de la question précédente.

Question 3.3:

On peut enfin écrire les fonctions `freresEtSoeurs`, `onclesEtTantes`, `cousinsEtCousines` et `petitsEnfants`. Chacune de ces fonctions prend en paramètre un `Individu` et retourne un tableau d'`Individu`. Comme précédement, vous ne connaissez pas à l'avance le nombre d'individus à retourner ; vous pouvez le

faire en deux étapes, la première consiste à compter et la seconde à collecter les individus.

Question 3.4: (bonus)

Écrivez une fonction qui affiche la descendance d'un membre donné d'une famille donné sous la forme suivante:

```
+ individu
+-+ enfant1
| +-- petitenfant1
| \-- petitenfant2
+-+ enfant2
| \-- petitenfant3
\-+ enfant3
  +-- petitenfant4
  +-- petitenfant5
  \-- petitenfant6
```