Travaux Pratiques de programmation n°6

Cours de programmation impérative

—Licence MPI L1 S2 - Info 121—

Types tableau de produit et produit de tableau

Cette séance de travaux pratiques est dédiée à l'écriture et l'utilisation de programmes sur les tableaux et produits.

▸ **Exercice 1. (État civil)**

On veut coder une application qui traite des informations sur l'état civil des personnes. Une personne est décrite par : son nom avec 20 caractères maximum, son genre (masculin ou féminin), les indices (entiers) de son conjoint éventuel et de ses parents. On suppose qu'on a au maximum 50 personnes à enregistrer. Enfin on interdira toute homonymie dans le système.

L'information n'étant pas toujours complète, on peut ne pas connaître l'un ou l'autre ou les deux parents d'une personne. Le statut marital des personnes peut aussi évoluer dans le temps.

1. On vous donne le type `Personne` ainsi que la structure pour représenter les informations de 50 personnes :

```
#include <string>
using namespace std;

const int MAXPERSONNES = 50;

enum Genre { M, F };

struct Personne {
  string nom;
  Genre genre;
  int indConjoint, indParent1, indParent2;
};

struct EtatCivil {
   Personne tableP[MAXPERSONNES];
   int nbP;
};
```

2. Coder une procédure `AjoutePersonne` qui ajoute une nouvelle personne dans l'état civil. Cette fonction prendra en paramètre le nom de cette personne, son genre et les deux indices de ses parents (les indices peuvent être -1 s'ils sont inconnus). On suppose qu'au moment de cet ajout, la personne n'a pas de conjoint. La procédure doit afficher un message d'erreur si elle ne peut pas ajouter la personne.
3. Coder une procédure `AffichePersonne` qui affiche les informations d'une personne donnée par son indice dans la base de l'état civil. Si la personne n'existe pas la procédure doit afficher un message d'erreur.
4. Coder une procédure qui affiche toutes les personnes de l'état civil.

5. Coder un programme principal qui appelle la procédure `RemplitEtatCivil` (permettant de remplir l'état civil avec 20 personnes) et utilise les fonctions et procédures précédentes afin de les tester.
6. Coder et tester une fonction qui recherche une personne donnée par son nom dans l'état civil et renvoie son indice si elle le trouve, ou -1 si la personne est non existante.
7. Coder et tester une fonction qui enregistre, dans l'état civil, le mariage de deux personnes données par leurs noms. La fonction renvoie `true` si le mariage est possible et `false` sinon. Un mariage entre deux personnes est possible si les deux personnes existent et si elles ne sont pas déjà mariées.
8. Coder une procédure permettant d'afficher, pour un individu donné, son arbre généalogique sous la forme :

```
Individu
    Mère
        Grand-mère maternelle
            ...
        Grand-père maternel
            ...
    Père
        Grand-mère paternelle
            ...
        Grand-père paternel
            ...
```

Voici par exemple l'affichage de l'arbre généalogique de l'individu 9 :

```
Individu 9
   Individu 5
      Individu inconnu
      Individu 1
         Individu inconnu
         Individu inconnu
   Individu 10
      Individu 12
         Individu inconnu
         Individu inconnu
      Individu 11
         Individu inconnu
         Individu inconnu
```

▸ **Exercice 2. (Jeu de cartes)**

L'objectif de cet exercice est de programmer un jeu de cartes simple : la bataille. Les cartes sont représentées par leur valeur de 1 à 10 et par leur couleur Pique, Cœur, Trèfle, Carreau. On vous donne le type `Couleur`, celui d'une carte ainsi que celui d'un paquet de cartes :

```
enum Couleur { P, Co, T, Ca };

struct Carte {
  int valeur;
  Couleur couleur;
```

```
};

struct Paquet {
  int taille;
  Carte carte[40];
};
```

La procédure `initPaquet` prend en paramètre un paquet de cartes et le remplit avec les 40 cartes possibles.

1. Coder une procédure qui affiche un paquet de cartes dans l'ordre.
2. Coder une procédure qui mélange le paquet de cartes. L'idée est de tirer au hasard deux nombres entre 0 et 39 et de permuter les cartes à ces positions. Vous répéterez cette procédure suffisamment pour mélanger le paquet.
3. Quand deux joueurs s'affrontent, chacun tire une carte du dessus de son paquet et celui qui a la plus forte valeur prend les deux cartes et les met à la fin de son paquet. Quand les deux cartes sont de valeurs égales, c'est la couleur qui permet de décider le vainqueur, les couleurs étant ordonnées de la plus forte à la plus faible ainsi : Pique, Cœur, Trèfle, Carreau. Coder une procédure qui prend en paramètre deux paquets et les modifie de façon à représenter un tour de jeu.
4. Le jeu se termine quand un joueur gagne toutes les cartes. Dans le `main`, répartir en deux paquets de 20 cartes le paquet initial puis afficher quel joueur gagne la partie (et éventuellement les étapes de cette partie).