UNIVERSITE LIBRE DE BRUXELLES 14 mars 2011
Faculté des Sciences Appliquées

INFO-H-100 : Introduction à la programmation
Projet 1 : Monty Hall

On vous demande de réaliser un programme C++ permettant à un utilisateur de jouer à une version inspirée du Monty Hall[1].

Le Monty Hall est un jeu télévisé diffusé aux Etats-Unis dans lequel un concurrent peut gagner une voiture s'il désigne la porte derrière laquelle elle se trouve. Le joueur est placé devant trois portes fermées, numérotées 1, 2 et 3. L'une d'entre elles cache une voiture. Les deux autres cachent une chèvre. Le joueur doit d'abord désigner une porte. Ensuite le présentateur, sachant où se trouve la voiture, ouvre une porte qui n'est ni celle choisie par le candidat, ni celle cachant la voiture. Le candidat a alors le choix d'ouvrir la porte choisie initialement ou bien d'ouvrir la troisième porte.

On vous demande de réaliser un programme permettant à l'utilisateur de jouer un certain nombre de fois au Monty Hall. A la fin de chaque partie, le joueur gagne 1 point s'il gagne la voiture, 0 sinon. A la fin du programme, son score final est affiché.

Voici un exemple complet d'exécution du jeu :

```
Combien de fois voulez-vous jouer ? : 3

Choisissez une porte entre 1 et 3 : 1
Monty ouvre la porte 2 qui cache une chèvre.
Choisissez la porte 1 ou 3 : 3
La porte 3 contenait une chèvre : vous avez perdu un point.

Choisissez une porte entre 1 et 3 : 2
Monty ouvre la porte 3 qui cache une chèvre.
Choisissez la porte 1 ou 2 : 2
La porte 2 contenait une voiture : vous avez gagné un point.

Choisissez une porte entre 1 et 3 : 2
Monty ouvre la porte 1 qui cache une chèvre.
Choisissez la porte 2 ou 3 : 3
La porte 3 contenait une voiture : vous avez gagné un point.

Le jeu est terminé. Votre score est 2 sur 3. A bientôt.
```

On vous demande de découper votre programme en fonctions dont voici quelques prototypes :

- `void playMontyHall()` qui lance le jeu. Ainsi, la fonction `main()` aura cette forme :

```
int main() {
  playMontyHall();
  return 0;
}
```

- `bool playGame()` qui effectue une partie et renvoie une valeur booléenne indiquant si la partie est gagnée.
- `int getMontyDoor(int carDoor, int playerDoor)` qui reçoit `carDoor`, le numéro de la porte derrière laquelle se trouve la voiture et `playerDoor`, celui que l'utilisateur a désigné dans son premier choix, et renvoie le numéro de la porte (déterminé pseudo-aléatoirement) que le présentateur choisit d'ouvrir.

On vous demande d'implémenter obligatoirement ces 3 fonctions et d'ajouter les autres fonctions nécessaires.

[1] `http://en.wikipedia.org/wiki/Monty_Hall_problem`

Génération de nombre pseudo-aléatoires

Voici un exemple complet de code `C++` affichant, tant que l'utilisateur appuie sur 'o', des entiers pseudo-aléatoires entre 0 et la constante `RAND_MAX` de la librairie `<cstdlib>`.

```
#include <iostream>
#include <cstdlib>
#include <ctime>

using namespace std;

int main() {
  srand(time(NULL));
  bool continuer = true;
  char c;

  while(continuer) {
    cout << rand() << endl;
    cout << "Continuer ? (o/n)" << endl;
    cin >> c;
    if(c != 'o')
      continuer = false;
  }
  return 0;
}
```

L'instruction `srand(time(NULL)) ;` initialise le générateur de nombres pseudo-aléatoires. Elle doit être exécutée une et une seule fois en début de programme.

La fonction `int rand()` renvoie pseudo-aléatoirement un entier entre 0 et `RAND_MAX`.

Consignes

Le projet est à réaliser **seul** et à remettre **au plus tard le lundi 28 mars à 12h30** au UB4.131 (à côté de la salle informatique Socrate). Les seules librairies autorisées sont `iostream`, `cstdlib` et `ctime`, les deux dernières ne pouvant être utilisées que dans le cadre de la génération de nombres pseudo-aléatoires. L'utilisation de tableaux est interdite.

Vous devez rendre un rapport dactylographié comprenant :

1. une introduction présentant succinctement les différentes fonctions de votre programme sur une page maximum,
2. une explication de votre algorithme permettant au présentateur de choisir aléatoirement la porte à ouvrir (un ou deux paragraphes).
3. une capture d'écran présentant une exécution de votre programme et
4. votre code complet intelligemment commenté.

Les critères d'évaluation sont les suivants :

- Le code : **résout exactement le problème demandé sans ajout ni apport personnel**, qualité de l'algorithmique, qualité et pertinence de la découpe en fonction, syntaxe correcte (doit compiler sans erreur ni avertissement), parfaitement indenté, les variables bien nommées, le minimum de commentaires utiles, respect des conventions et des règles de bonne pratique vues au cours, aux séances d'exercices et résumées sur la page web des projets (`http://cs.ulb.ac.be/public/teaching/infoh100/projets`)
- Le rapport : qualité du contenu, présentation, français correct.
- Le respect des consignes.

Les élèves assistants sont disponibles tous les jours à 12h30 dans la salle Socrate afin de répondre à vos questions.