IFT1166 – TRAVAIL PRATIQUE #4 – 12 novembre 2005

JEU de MINI-MONOPOLY

Mohamed Lokbani

Équipes: Le travail en équipe de deux est permis. Vous ne remettez alors qu'un travail par équipe.
Remise : Deux remises à effectuer : électronique et papier le jeudi 1 décembre, 20h30 au plus tard, sans possibilité de retard. Vous pouvez remettre votre copie à vos démonstrateurs sinon le cours (durant la pause) du 29 novembre, sinon sous la porte de mon bureau P-A.A (3225).
Conseil: N'attendez pas le dernier jour avant la remise pour commencer, vous n'aurez pas le temps.

But : Ce TP porte sur le passage par référence, l'allocation dynamique, l'utilisation des classes de base et la manipulation de fichiers.

Énoncé : Dans ce TP, vous devez écrire un programme C++ qui simule une partie de « mini-monopoly ».

Description du jeu : Le « mini-monopoly » consiste en une planche carrée de cases (situées à la bordure) et un groupe de joueurs. Chaque joueur possède de l'argent et avance à son tour sur la planche un nombre de cases, déterminé par les dés. La première case représente le « GO » et la dernière représente les « TAXES ». Toutes les autres cases représentent des propriétés. Initialement, toutes les propriétés appartiennent à la banque.

CAPITAL : Initialement, chaque joueur dispose de la somme de 1500$.

QUAND LE JOUEUR TOMBE SUR UNE PROPRIETE SANS PROPRIETAIRE : Si le joueur possède assez d'argent pour l'acheter (prix d'achat <= son portefeuille), alors il doit l'acheter de la banque, sinon il y reste tranquillement.

QUAND LE JOUEUR TOMBE SUR UNE PROPRETE QUI A DEJA UN PROPRIETAIRE : Alors le propriétaire perçoit du joueur un loyer. Si le joueur n'a pas assez d'argent, il remet quand même tout ce qu'il a comme argent au propriétaire, puis il se place dans la situation « Fin de partie » décrite plus bas.

SALAIRE : La première case de la planche est un peu spéciale. En effet, ce n'est pas une propriété à vendre, mais la case « GO ». Tout joueur qui y passe (sans nécessairement s'y arrêter), réclame 200$ de la banque.

TAXES : La dernière case de la planche non plus n'est pas une propriété à vendre. Tout joueur qui s'y arrête, doit payer 1500$ à la banque.

DES DOUBLES : Si un joueur obtient un doublet en jetant les dés, il avance son jeton comme d'habitude et bénéficie de tous les privilèges ou doit subir toutes les pénalités relatifs à la case atteinte. Si au tour courant, il n'a pas été éliminé, il garde le cornet, lance les dés de nouveau et continue à joueur tant et aussi longtemps qu'il n'obtient pas de doublets ou n'est pas éliminé.

FIN DE LA PARTIE : Dès qu'un joueur n'a plus d'argent, il est automatiquement éliminé de la partie. Ses propriétés sont immédiatement saisies par la banque et remises sur le marché. La partie continue avec les joueurs restants et se termine lorsqu'il ne reste plus qu'un joueur possédant de l'argent. Celui-ci est alors déclaré le vainqueur.

Description du travail : Votre programme doit lire les spécifications du « mini-monopoly » (voir « description de l'entrée »), simuler une partie et afficher chaque action des joueurs (voir "description de la sortie").

Description de l'entrée : Votre programme doit lire **l'entrée du clavier**. L'entrée de votre programme consiste en 5 parties:

1. la dimension "n" de la planche

La dimension de la planche n'est pas fixe. Elle est représentée par un entier n où $1 < n < 10$. N représente la taille de chaque côté de la planche et non le nombre de cases. Vous pouvez supposer que l'utilisateur entrera un entier, mais vous devez vérifier qu'il soit entre les bonnes bornes, et, le cas échéant, afficher un message d'erreur.

2. le nombre "j" de joueurs

Le « mini-monopoly », se joue avec j joueurs, où $2 < j < 9$. Vous pouvez supposer que l'utilisateur entrera un entier, mais vous devez vérifier qu'il soit entre les bonnes bornes, et, le cas échéant, afficher un message d'erreur.

3. la description des "p" propriétés

Pour une planche de dimension n, il y aura p propriétés où $p = (n-2)*4 +2$. Chaque propriété sera décrite de la façon suivante:
<le nom de la propriété> <le prix d'achat> <le prix de location>

La taille du nom d'une propriété n'est pas limitée mais elle est sous la forme d'un seul mot (pas de blancs). Vous pouvez supposer que ce format sera respecté et que « p » propriétés seront décrites.

4. la description des "j" joueurs

Chaque joueur aura un nom. La taille du nom d'un joueur n'est pas limitée mais elle est sous la forme, elle aussi, d'un seul mot (pas de blancs). Vous pouvez supposer que ce format sera respecté et que j joueurs seront décrits.

5. la valeur des dés

La valeur de chaque dé sera lue en entrée. Vous pouvez supposer que la valeur des dés sera bien entre 1 et 6. Il serait bien sûr plus intéressant d'utiliser une fonction aléatoire pour générer la valeur d'un dé... c'est pour une autre fois !.

Exemple d'entrée

Pour un exemple d'entrée, consultez le fichier « entree.txt » disponible sur la page Web.

Description de la sortie : L'affichage en sortie doit se faire dans un fichier dont le nom est lu comme seul argument de la ligne de commandes. Chaque action d'un joueur doit être décrite sur une ligne à part. Le tableau suivant vous indique les actions à décrire et le message à afficher.

Action	Correspondance
Lancement des dés	"<nom du joueur> lance les des, il a D1 et D2." (où D1 et D2 sont les valeurs des dés)
Avancement sur la planche	"<nom du joueur> avance de N cases." (où N est la somme des dés)
Passage par GO	"<nom du joueur> passe par GO, il reclame 200$."
Arrêt sur une case	"<nom du joueur> s'arrete sur <nom de la case>."
Achat d'une propriété	"<nom du joueur> achete <nom de la propriété>."
Paiement de location	"<nom du joueur> paie la location de <nom de la propriété>."
Contenu du portefeuille: après une transaction (paiement ou réclamation), le contenu du portefeuille du joueur doit être annoncé	"<nom du joueur> possede X$." (où X est le montant de son portefeuille)
Élimination d'un joueur	"<nom du joueur> est elimine."
Doublet	"<nom du joueur> a un doublet, il joue de nouveau."
Vainqueur de la partie	"<nom du joueur> gagne la partie."

Exemple de sortie

Pour un exemple de sortie, consultez le fichier « sortie.txt » disponible sur la page Web.

Erreurs possibles : S'il y a des erreurs dans l'entrée, votre programme doit les détecter, afficher un message d'erreur et arrêter son exécution. Les messages d'erreurs doivent être renvoyés sur la sortie des erreurs. Les erreurs possibles sont:

Nature de l'erreur	Message à afficher
Une planche trop petite ou trop grande	"Planche trop petite." "Planche trop grande."
Pas assez de joueurs ou trop de joueurs	"Il manque des joueurs." "Il y a trop de joueurs."
Le fichier de sortie n'est pas disponible sur la ligne de commandes, ou il y a une erreur d'ouverture du fichier de sortie.	"La ligne de commandes est incomplete. Il manque le nom du fichier de sortie. La ligne de commande doit etre : nom_du_programme fic_sortie" "impossible d'ouvrir le fichier de sortie."

Tester votre programme : Pour tester votre programme avec un fichier d'entrée (plutôt que de retaper l'entrée au clavier à chaque exécution), vous pouvez écrire un fichier contenant les entrées (comme le fichier « entree.txt » disponible sur le Web) et utiliser la redirection Unix pour lire l'entrée de ce fichier et écrire les résultats dans un fichier.

Par exemple, si votre programme s'appelle « tp4.cpp » et dont l'exécutable est « tp4.exe ». Alors:
`tp4.exe < entree.txt sortie.txt`
Il va lire l'entrée à partir du fichier « entree.txt » suite à une redirection et va écrire les résultats dans « sortie.txt »

Remise : Il est important de noter que votre TP sera compilé avec gcc3.2.4. Si par choix vous décidez d'utiliser un autre compilateur, vérifiez que le code que vous avez produit (qui normalement fonctionne correctement chez vous) fonctionne bien sur les ordinateurs du DESI. Pour avoir la version du compilateur utiliser la commande : "**gcc –v**" devra donnée le numéro de version "**3.2.4**".
Vous devez remettre **2 fichiers** : « tp4.cpp » et le « rapport ».

Le rapport doit être au format texte ou pdf. Voir la rubrique « Foire Aux Questions » sur la page web du cours, sur la procédure à suivre pour produire des pdf.

La remise comprend **3** (**TROIS**) choses (**avant le jeudi 1 décembre à 20h30**) :

1. Envoyez votre fichier « tp4.cpp », ainsi que le rapport par la procédure de remise électronique habituelle (Pour obtenir de l'aide sur cette commande, tapez dans un xterm : man remise). Respecter les noms des fichiers.

   remise ift1166 tp4 tp4.cpp rapport

2. Vérifier que la remise s'est effectuée correctement.

   remise –v ift1166 tp4

3. Remettez une **copie papier** de votre programme (« tp4c.cpp ») ainsi que de votre rapport.

Barème : Ce TP4 est noté sur 13 points.

Compilation et respect des spécifications	**1**
Codage, commentaires, style de programmation etc.	**4**
Rapport	**3**
Tests (fournis et non fournis)	**5**

En plus du précédent barème, vous risquez de perdre des points dans les cas suivants ….

- La non remise électronique (volontaire ou par erreur) est sanctionnée par la note 0.
- La non remise papier vous pénalise de 1 point.
- Les programmes ne contenant pas d'en-tête, -1 point.
- Un programme qui ne compile pas : 0/13.
- Un programme qui compile mais ne fait les choses prévues dans la spécification : 0/13.
- Les avertissements (warnings) non corrigés : cela dépend de la quantité! À partir de -0.25 et plus.
- Le nom respect du nom de fichier, donc forcement il ne va pas compiler et un des points de la spécification n'a pas été respecté : 0/13.
- Aberration dans le codage : Même si tous les chemins mènent à Rome, faites l'effort nécessaire pour éviter de prendre le plus long!

Des questions à propos de ce TP?

Une seule adresse : pift1166@iro.umontreal.ca

Pour faciliter le traitement de votre requête, inclure dans le sujet de votre email, au moins la chaîne: [IFT1166]

Mise à jour

12-11-2005 diffusion

Annexes

Nous avons fourni plusieurs programmes supplémentaires qui traitent sur les questions suivantes :

- Comment connaître le nombre d'arguments et comment manipuler la liste des arguments : « **argcv911.cpp** »
- Comment ouvrir un fichier, lire son contenu et écrire dans un fichier : « **fichier911.cpp** »
- Comment déclarer une variable du type « string » et les méthodes associées au type « **string911.cpp** ».

Démarches

Plusieurs façons d'aborder ce travail. Voici comment, nous percevons les choses, vous pouvez procéder autrement, mais dans tous les cas, il est préférable d'aller par étapes et éviter donc de tester qu'après avoir écrit tout le code ! Le débogage risque de vous faire passer des nuits blanches …

- Étudier d'abord les différents exemples fournis « argvc911.cpp », « fichier911.cpp » et « string911.cpp ».
- Essayer de vous faire la main avec un petit programme de rien du tout, avec la commande (**tp4.exe < entree.txt sortie.txt**). Il faudra donc créer d'abord le fichier « entree.txt » et y écrire des données quelconques. Par la suite dans un programme lire la ligne de commandes et tester si l'argument (fichier de sortie) est présent. Si c'est le cas l'ouvrir puis le fermer. Dans une seconde démarche lire le contenu de la redirection en entrée (i.e. le contenu du fichier « entree.txt ») comme vous l'avez déjà fait dans les précédents travaux, et écrire les données lues dans le fichier de sortie. Si c'est ok, prenez comme exemple le fichier « entree.txt ». À la fin de cette étape, vous aurez compris comment manipuler les entrées et sorties.
- Passer maintenant au travail demandé.
- Prendre un papier et un crayon et dérouler un simple jeu avec peu de joueurs et de cases pour déjà comprendre le mécanisme du jeu. Vous pouvez utiliser le fichier « planche.pdf » comme exemple.
- Réfléchir comment introduire la notion d'objets dans le problème. Par exemple, vous pouvez concevoir deux classes, une pour contenir les propriétés du joueur et l'autre les propriétés de la case.
- Lire les données du fichier « entree.txt » et les associer aux classes définies. Vous pouvez ajouter des affichages en sortie pour tester que cette étape s'est faite correctement.
- Écrire les fonctionnalités qui relient la case au joueur. Si par exemple un joueur tombe sur la case « Go » que dois-je faire ? Si un joueur tombe sur une case qui a un propriétaire, que dois-je faire ? etc.
- Compléter le scénario au complet.
- Pour tester, prenez le précédent exemple, celui déroulé sur papier, et examiner comment il se comporte avec votre programme.