IFT 159 – Devoir 6

Classes et récursivité

Objectifs :

À la fin de ce devoir, vous aurez

1. implémenté et utilisé des classes ;
2. utilisé la récursivité pour résoudre un problème de recherche;
3. complété l'analyse d'un problème;
4. appliqué les principes de conception modulaire;
5. complété la conception d'un programme déjà entamé (diagramme structurel, algorithmes, diagramme de classes UML);
6. complété la mise en œuvre d'un programme en C++;
7. documenté du code au moyen de commentaires et des normes de programmation;
8. documenté des types de données.

Instructions :

Ce devoir est à faire en équipe de deux **obligatoirement**. Vous devez lier les comptes turnin web des deux membres du binôme.

Vous devez comprendre et compléter, en C++, un jeu de démineur. **Aucune analyse/conception** entamée ne vous est fournie avec cet énoncé.

Vous aurez une classe ainsi que trois fonctions à concevoir.

Vous devriez compléter les éléments demandés dans la conception **avant** de commencer à écrire votre code C++. Vous devez fournir deux fichiers textes de carte, nommés respectivement `carte1` et `carte2`. Vous devez aussi fournir deux fichiers de tests (`test1` et `test2`), un pour chacune des cartes. Chaque fichier de test contient les informations pour déminer une carte. La première entrée de chacun de ces fichiers devrait donc être le nom de la carte à utiliser. Les entrées suivantes seront les positions à ouvrir. Vous avez **un exemple d'une carte** avec un **fichier de test**.

Vous pouvez trouver aussi, des exécutables déjà compilés (le vôtre devrait fonctionner comme ceux fournis), pour Windows, pour Mac OSX ou pour Ubuntu, ainsi que des exemples illustrés de scénarios.

Travail à faire : (tenu en compte lors de la correction et l'évaluation)

1. Pairez votre équipe sur Turnin.
2. Lisez et comprenez la description du problème (fichier `Démineur-Description.pdf`.
3. Les modules à concevoir sont des méthodes de la classe `Carte` dont les signatures sont :
    - `bool essaieCase(Position);`
    - `void compteMinesAdjParCase();`
    - `Compteur getNbMinesAdjacentes(Position e_pos)`.
4. Concevez la classe `Position`
    1. Dessiner le diagramme de classes : vous pouvez vous inspirer (et compléter) le diagramme de classes fourni.
    2. Concevez la classe (voyez le document **`D6-classePosition.pdf`**);
    3. Reportez dans le document à remettre (`aRemettre.pdf`) le diagramme de classes et les analyses des méthodes de la classe.
5. Récupérez **la base de code** et codez en C++ une version bidon de la classe (pour assurer la compilation, à ce stade)
    1. Définition bidon de la classe dans le fichier `.h` : seulement définir les méthodes qui sont appelées dans les autres fichiers;
    2. Méthodes bidons qui sont définies, dans le fichier `.cpp`;
    3. Compiler le code qui est fourni; si vous avez bien fait votre codage de classe bidon, cela devrait compiler;
6. Complétez le codage de votre classe selon la conception;
    - Testez les différentes méthodes, le plus tôt possible;
7. Concevez deux cartes différentes (voir **fichier exemple**) que vous nommerez `carte1` et `carte2`. **Les deux cartes doivent avoir des dimensions différentes et contenir au moins 10 mines**.
8. Concevez deux tests (un pour chaque carte, voir **fichier exemple**) que vous nommerez `test1` et `test2`. Un test devrait se solder par une explosion et l'autre par un déminage complet. Chaque test devrait comporter au moins trois essais de case.
9. Écrivez votre code-source pour les 3 méthodes de `Carte` à compléter pour le jeu. Les définitions des méthodes sont dans deux fichiers séparés, `carte.cpp` et `carte_.cpp`. Les trois méthodes que vous devez implémenter sont dans `carte_.cpp`. Vous n'avez pas à modifier ce qui est dans `carte.cpp`.
10. Testez avec les jeux d'essais planifiés et corrigez, s'il y a lieu, votre code;

11. Faites la remise de tout ce qui est demandé sur turnin;

    À remettre (sur **turnin** web) : la remise s'effectue dans le projet `Devoir6`. Respectez le nom des fichiers demandés :

    1. Fichier `aRemettre.pdf`

    2. Fichiers de code :

        - `carte_.cpp`
        - `position.h`
        - `position.cpp`

    3. Deux (2) fichiers de cartes et leurs fichiers de test correspondants

        - `carte1`
        - `test1`
        - `carte2`
        - `test2`