

## TP 6 : Fonctions, tableaux, tableaux à deux dimensions

Rappel : nous utilisons dans ce cours certaines fonctionnalités de C++ qui ont été définies dans le standard ISO C++ de 2011. `CodeBlocks` doit donc être configuré pour utiliser ce standard ; si ce n'est pas encore fait, voir l'exercice 1 de la fiche de TP 5 pour les instructions.

### Exercice 1 (Tableaux à deux dimensions).

- (1) Écrire un programme qui affiche les quatre tableaux suivants en utilisant la fonction de l'exercice 3 (3) du TD. Documenter cette fonction.

```
vector<vector<int> > tabVide    = { };
vector<vector<int> > tabCarre  = { { 1,  2, 3 },
                                   { 4, 11, 6 },
                                   { 9, 12, 7 } };
vector<vector<int> > tabBizarre = { { 1,  2, 3 },
                                   { 4,  5 },
                                   { 6,  7, 8, 10 } };
```

- (2) Implanter, documenter et tester les fonctions `nombreDeLignes`, `nombreDeColonnes`, `contient`, `comptePlusGrandQueDix` et `estSymmetrique` de l'exercice 3 du TD. On pourra partir du fichier fourni `tableaux2D.cpp`, ou s'en inspirer.
- (3) ♣ La fonction pour afficher un tableau est une fonction qui agit par effet de bord et non pas qui calcule. Il n'est donc pas possible de la tester automatiquement avec `ASSERT`. Comment pourrait-on changer ses spécifications pour qu'il soit possible de la tester automatiquement ?

### Exercice 2 (Types de base ; moins de 20 minutes).

Consulter les programmes suivants donnés dans les exemples du cours : `bool.cpp`, `int.cpp`, `long.cpp`, `char.cpp`, `string.cpp`, `float.cpp`. Prédire ce que chacun d'entre eux affiche *puis seulement* l'exécuter pour contrôler votre prédiction.

Comme d'habitude, les programmes vus en cours sont sur la page web du cours.

Vous pouvez aussi accéder à chacun d'entre eux individuellement en suivant le lien dans les notes de cours en ligne.

**Exercice ♣ 3.**

L'objectif de cet exercice est de réaliser le jeu du « démineur ».

- (1) Implanter les algorithmes de l'exercice 4 du TD dans le squelette fourni dans l'archive (voir `demineur.cpp` et `aleatoire.cpp`, attention à bien désarchiver les fichiers avant de les ouvrir). On complètera au fur et à mesure la documentation et les tests.
- (2) Ajouter une fonction `partie(n,m)` qui tire une grille au hasard de taille  $n \times m$ , l'affiche, demande à l'utilisateur le coup qu'il souhaite jouer (sous la forme : « m 2 4 » pour marquer la case de coordonnées (2,4) et « d 2 4 » pour démasquer ladite case) et recommence jusqu'à la fin de la partie.
- (3) (♣) Pour éviter la frustration du joueur qui clique directement sur une mine, écrire un programme qui attende le premier démasquage du joueur, puis génère une grille de telle sorte que le premier coup ne soit jamais perdant.
- (4) (♣♣♣) En supposant (3), une grille est dite *résoluble* si l'on peut démasquer toutes les cases non minées, une à une, sans jamais avoir recours à de l'aléatoire (c'est à dire qu'on est toujours certain qu'une case ne contienne pas de mine avant de la démasquer).

Écrire un programme qui prenne une grille en entrée, et réponde si oui ou non elle est résoluble.