Introduction à la Programmation

—

Examen partiel sur machine

—

G1: monasse*(at)*imagine.enpc.fr G2: facciolg*(at)*imagine.enpc.fr
G3: alexandre.boulch*(at)*onera.fr G4: theophile.dalens*(at)*inria.fr
G5: bourkia(at)imagine.enpc.fr G6: fernandl*(at)*imagine.enpc.fr

—

18/11/15

—

1 Enoncé

1.1 Sudoku

On va programmer un jeu de Sudoku, qui permet à la fois de jouer et de donner la solution. Voici les règles :

— On a 3×3 blocs, chaque bloc étant une grille 3×3 de cases.
— Certaines cases (les indices) sont fixées avec une valeur entière entre 1 et 9, les autres sont vides.
— Il faut complèter les cases vides avec des valeurs entières entre 1 et 9 de telle sorte qu'aucun chiffre ne se retrouve deux fois dans la même ligne, la même colonne ou le même bloc.

1.2 Travail demandé

Il est plus important de livrer un code clair (commenté et indenté) et qui compile sans warning, même s'il ne répond pas à toutes les questions. Pour cela, vérifiez à chaque étape que votre programme compile et se lance correctement.

Créez un projet Imagine++ avec un fichier contenant le `main`. Prenez le `CMakeLists.txt` d'un projet existant ou celui de `[IMAGINEPP_ROOT]/test/Graphics/` et adaptez-le[1].

1.2.1 Affichage

1. On utilise la structure suivante (dans un fichier `grille.h`) pour stocker la grille :

```
struct Grille {
    int valeur[9][9];
    bool indice[9][9];
};
```

où on met l'entier entre 1 et 9 de la case (i, j) dans `valeur[i][j]` et 0 si la case n'a pas de valeur, et `indice[i][j]` indique si la case est un indice. Écrire une fonction `init_grille` pour construire une grille vide.

2. Écrire une fonction `affiche_grille` qui affiche à l'écran une grille. On pourra mettre en rouge les valeurs d'indices, en noir les valeurs normales et ne rien afficher dans les cases de valeur 0. Tester avec une grille comportant un indice et une valeur normale non indice. Pour afficher un nombre, utiliser la fonction `drawString` qui écrit dans un rectangle dont le coin bas-gauche est donné en argument de la fonction. Pour convertir un caractère en `string`, on peut faire : `string s="X"; char c='2'; s[0]=c;`.

1. `[IMAGINEPP_ROOT]` est le chemin où Imagine++ est installé, typiquement `/usr/share/Imagine++` sous Mac et Linux, et `C:/Program Files (x86)/Imagine++` sous Windows

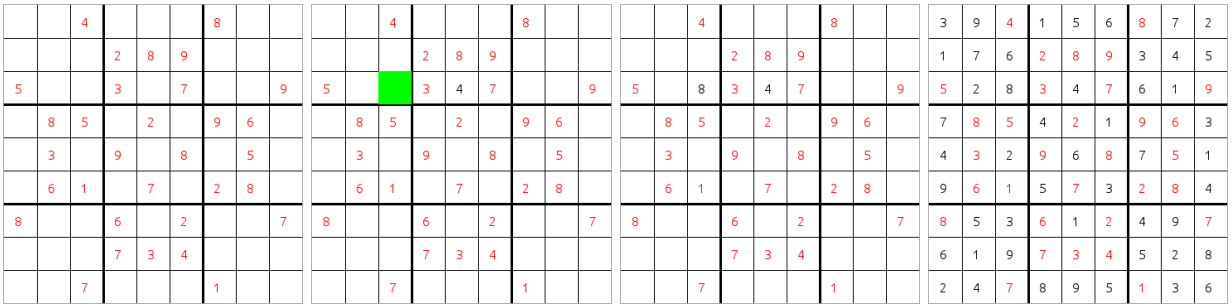

FIGURE 1 – De gauche à droite : grille avec indices, après clic dans `entre_case`, après entrée de la valeur 8 dans `entre_case`, sudoku résolu automatiquement.

3. Écrire une fonction `entre_case` qui attend un clic de l'utilisateur sur une case (`getMouse`) puis que l'utilisateur entre un chiffre entre 1 et 9 (`getKey` renvoie un code de la touche, donc pour le chiffre 0 renvoie '0', c'est-à-dire 48, et non 0). Cette fonction renvoie le booléen `false` si le clic est avec le bouton à droite de la souris (auquel cas l'utilisateur n'entre pas de valeur numérique). Sinon, renvoie dans des arguments de la fonction le numéro de case et la valeur. Quand on attend la valeur numérique, on pourra afficher le fond de la case dans une couleur différente.
4. Écrire une fonction `entre_grille` qui renvoie une grille après avoir interagi avec l'utilisateur par `entre_case` jusqu'à avoir un clic droit.
5. Écrire une fonction `verifie_case` qui indique si la valeur en (i, j) (arguments de la fonction) n'entre pas en conflit avec le reste de la grille suivant les règles rappelées en introduction.
6. Appeler cette fonction dans `entre_grille` de manière à afficher un fond rouge pendant une seconde si l'utilisateur entre une valeur interdite.

1.2.2 Résolution

7. Définir une fonction `bool case_suivante(Grille g, int& i, int& j)` qui passe d'une case (i, j) à la suivante dans l'ordre de lecture, de gauche à droite et de haut en bas, en sautant les cases d'indice. Renvoie `false` s'il n'y a pas de case suivante.
8. Écrire une fonction récursive `bool resous_grille(Grille& g, int i, int j)` qui :
   — donne successivement à (i, j) toutes les valeurs possibles,
   — teste à chaque fois si la valeur est licite (pas de conflit),
   — et s'appelle récursivement sur la case suivante si c'est le cas.
   Cette fonction renvoie `true` si la grille est soluble et dans ce cas renvoie la grille remplie.
9. Écrire une fonction `bool resous_grille(Grille& g)` qui appelle la précédente en partant de la bonne position de départ.
10. Tester le programme : l'utilisateur entre d'abord les indices puis essaie de remplir sa grille. Dans cette dernière phase, s'il fait un clic droit on lui donne la solution.

Important : Quand vous avez terminé, créez une archive du projet *à votre nom ou au nom du binôme* en ZIP, RAR, TGZ ou 7z. N'incluez que les fichiers source et le CMakeLists.txt, pas les fichiers binaires créés par Cmake. Ne partez pas avant que le surveillant ne soit passé vous voir pour copier cette archive sur clé USB.