Projet de Programmation
Sudoku Solver

Section Génie Mécanique

Balthazar Bolze SCIPER: 361471
Pierre Louis Peter SCIPER: 363161
Cecile Caen SCIPER: 361560

Décembre 2023

1 Introduction

1.1 Matériels Informatiques

- OS : sti windows 10
- Plateforme de test : Machine Virtuelle epfl
- Compilateur C : Visual Studio Code puis conversion sur Visual Studio
- Matlab : MATLABR2023a
- Labview : NI LabVIEW 2023 Q3

1.2 Fichiers auxiliaires :

- OCR.exe (éxecutable)
- sX.jpg (photo en format JPG des 13 sudokus)
- s8.pdf (sudoku résolu avec grille s8)
- CellValue.txt
- Cell.bin
- rapport.pdf

2 Partie C :

2.1 Listes des Fichiers :

- OCR.c

2.2 Informations :

- Entrée : Exécutable + Chemin courant pour accéder au fichier Cell.bin + seuils maximums correspondant aux différents chiffres.
- Sortie : Fichier texte CellVallue (contient numéro de la case + pourcentage)
- Détail : La fonction main va gérer la gestion d'erreurs, ainsi que lire les cellules du sudoku.
  - Entrée : Exécutable + Chemin courant pour accéder au fichier Cell.bin + seuils maximums correspondant aux différents chiffres.
  - Sortie : Fichier texte CellVallue (contient numero de la case + pourcentage)

- Detail : La fonction va gérer la lecture des cases du sudoku ainsi que la gestion d'erreurs liées à cette lecture.
  - if il y a le bon nombre d'arguments, c'est-à-dire 12. (ligne 66)
  - boucle parcourant les ratios donnés en argument, tout en vérifiant s'ils sont compris entre 0 et 100. (ligne 72)
  - f_in et f_out, ouverture des fichiers d'entrée et de sortie. Ensuite, vérification s'ils ne sont pas vides. (ligne 82)
  - initialisation des variables L, H, a et b. Puis lecture de la longueur et de la largeur du fichier en entrée. (ligne 89)
  - if la largeur et la longueur ont été bien lues. (ligne 97 & 102)
  - if la largeur est comprise entre 16 et 100. (ligne 107)
  - if la hauteur est comprise entre 24 et 100 (ligne 111)
  - on attribue à cell une mémoire, par la fonction malloc. (ligne 117)
  - stockage des élèments de f_in dans cell. (ligne 126)
  - if il y a suffisament d'élèments dans cell. (ligne 129)
  - if il y n'a pas trop d'élèments dans cell. (ligne 134)
  - double boucle parcourant les pixels de cell. Appel de la fonction GetCellBit pour connaître la ressemblance entre la case donnée et le BitMap vide. Incrémentation de la valeur check. (ligne 147)
  - calcul du ratio de ressemblance. (ligne 158)
  - if ratio calculé est plus grand que ratio minimal alors cell est considérée vide. (ligne 161)
  - quintuple boucle for, parcourant les 9 Bitmap, leurs colonnes puis leurs lignes. Puis les colonnes de la case regardée et enfin ses lignes. Appel de GetDigitBitmapBit et de GetCellBit, pour savoir si le Bitmap et la case sont compatibles. Vérification si le ratio est plus grand que le ratio du bitmap. (ligne 174 - 211)
  - écriture dans le fichier de sortie de la valeur du chiffre correspondant à celui de la cellule , ainsi que son ratio de compatibilité. (ligne 215)
  - libération de mémoire. (ligne 217)
  - fermeture des fichiers d'entrée et de sortie. (ligne 218)
  - return 0. (ligne 224)

2.3 Différentes fonctions :

2.3.1 GetDigitBitmapBit

- Entrée : tableau digitBitmap (chiffre entre 1 et 9), l (ligne), c (colonne).
- Sortie : boolean (0 ou 1)
- Detail : La fonction lit le digitBitmap du chiffre concerné à la ligne l et va effectuer une opération binaire pour connaître l'état au pixel lu, 0 ou 1.

2.3.2 GetCellBit

- Entrée : cell (array de la cellule), Width, line, col.
- Sortie : boolean (0 ou 1)
- Detail : GetCellBit lit la cellule sous forme binaire. La cellule a été transformée d'un format 44 par 44, à un format où la largeur vaut 44*44 et la hauteur 1. Donc la fonction va parcourir la cellule pour connaître l'état d'un pixel, 0 ou 1.

2.4 Gestion des erreurs :

2.4.1 Erreurs :

Toutes les erreurs print un message et arrêtent le programme :

- **Nombres d'arguments différents de 12**
  => return -1 & print : "arguments that you have entered are wrong ! try again !"
- **Ratio argument n'appartient pas a l'enssemble [0,100]**
  => return -1 & print : "Valeurs des ratios non-contenues entre 0 et 100 !"
- **Fichier n'arrive pas à etre ouvert/créé/...**
  => return -1 & print : "perror"
- **Nombre d'éléments pour la largeur lu dans fin (fichier in) est différent de 1**
  => return -1 & print : "erreur dans la lecture de la largeur"
- **Nombre d'éléments pour la hauteur lu dans fin (fichier in) est différent de 1**
  => return -1 & print : "erreur dans la lecture de la hauteur"
- **Largeur non comprise entre dans l'intervalle [16,100]**
  => return -1 & print : "L n'est pas compris entre 16 et 100"
- **Hauteur non comprise entre dans l'intervalle [24,100]**
  => return -1 & print : "H n'est pas compris entre 24 et 100"
- **Fichier Cell vide**
  => return -1 & print : "erreur de memoire"
- **Nombre d'éléments (pixels) lu dans le fichier insuffisant**
  => return -1 & perror : "erreur dans la lecture des éléments"

2.4.2 Warning :

Un warning print un message de prévention et continue le programme !

- **Nombre d'éléments (pixels) lu dans le fichier en excès**
  => return -1 & print : "warning : il y a trop d'éléments"

3 Partie Matlab :

3.1 Listes des Fichiers :

- checkSolved.m
- dispSudoku.m
- FillHypothesis.m
- getChoices.m
- solveSudoku.m
- Solve.m

3.2 Differentes fonctions :

3.2.1 dispSudoku

- Entrée : Initial, Solution, file, NBiterations, recLevel, duration, solvability
- Sortie : Affichage de la grille du sudoku résolue en fichier PDF.
- Detail : La fonction DispSudoku permet à partir d'une matrice initiale et d'une matrice solution de remplir une grille créée à l'aide de la fonction meshgrid. Cette grille indiquera en gras dans les cases

concernées les chiffres présents dans Initial, tandis que les chiffres trouvés par le programme seront en italique. Les bordures du sudoku sont également affichées en différentes teintes.

3.2.2 FillHypothesis

- Entrée : Matrice initiale M
- Sortie : Génère la matrice A 9x9x9 comportant des 0 et des 1
- Detail : La fonction FillHypothesis prend en entrée la matrice M 9x9, la grille de sudoku initial. Elle génère la matrice A 9x9x9, où chacune des couches correspond aux possibilités d'avoir un chiffre spécifique dans une case spécifique du sudoku.

3.2.3 getChoices

- Entrée : Matrices A et M
- Sortie : x,y,choices
- Detail : La function getChoices retourne le nombre de choix possibles par case et les coordonnées de celle-ci, à partir de la matrice A 9x9x9, en sommant sa profondeur. La case ayant le plus petit nombre de choix est sélectionnée.

3.2.4 checkSolved

- Entrée : Matrice A
- Sortie : solvability
- Detail : La function checkSolved vérifie si le sudoku est solvable ou non. La somme de la matrice A 9x9x9 efectuée dans la fonction getChoices est utilisée, donnant la matrice R. Si toute les cases de R contiennent 1 alors le sudoku est résolu, si une seule contient un 0, alors il est insolvable, sinon il n'est pas encore résolu.

3.2.5 solveSudoku

- Entrée : M,rlevel
- Sortie : M, solvability
- Detail : SolveSudoku est une fonction récursive, elle s'appelle elle-même afin de résoudre le sudoku. En son script, elle appelle également FillHypothesis, checkSolved et getChoices. La function va par récursivité tester tous les choix possibles et donc résoudre le sudoku s'il est solvable.

3.2.6 Solve

- Entrée : pas d'entrée
- Sortie : pas de sortie
- Detail : Solve est l'équivalent d'une function main, c'est elle qui va faire appel à solveSudoku pour résoudre le Sudoku et qui va calculer le temps pris pour le résoudre. Cette function affiche également la solution en appelant la fucntion dispSudoku.

3.3 Gestion des erreures :

Il n'y pas de gestion d'erreur en Matlab !

4 Partie Labview :

4.1 Listes des Fichiers :

- Build_m_script.vi
- coordonnées_cellule.vi
- MP_LaunchMatlabScript4.vi
- read_sudoku.vi
- SolveSudoku_Labview.vi

4.2 Listes des SubVI :

- SolveSudoku_Labview est notre VI principale. C'est elle qui prend en entrée l'image du sudoku, qui appelle le programme C ainsi que le matlab, ce qui nous permet de résoudre le sudoku et nous le renvoie résolu sous forme de fichier pdf (que l'on decide d'ouvrir ou non).
- Build_m_script est une VI qui utilise des concatenate strings pour appeler le Matlab correctement. De plus c'est dans cette fonction que l'on fait le test pour savoir si l'on va ou non afficher le pdf du sudoku résolu.
- read_sudoku est une Vi utlisée au début du programme, elle permet de lire l'image entrée, de la convertir en noir et blanc et donne accès aux données de l'image.
- MP_LaunchMatlabScript4 est la VI qui nous a été fournie, elle permet de faire correctement appel au Matlab afin de résoudre le sudoku. Elle se situe dans unz case et n'est éffectuée que si le boolean correspondant est sur vrai.
- coordonnés_cellule est une VI qui calcule les corrdonnées de chacun des points de la cellule que nous voulons envoyer dans le C.

4.3 Informations :

- Entrée : Image du sudoku à résoudre en format JPG.
- Sortie : Le PDF du sudoku résolu.
- Detail : Labview correspond à l'interface utilisateur. Le sudoku est déposé dans le front panel sous format JPG. On lance ensuite le programme Labview qui va dans un premier temps lire l'image entrée, la convertir en noir et blanc et donner accès aux données de l'image. Ensuite, on entre dans la double boucle et l'on entre les coordonnées de chacun des coins de la cellule et ils sont convertis en un tableau de pixels. En parallèle, on crée le chemin complet du fichier cell.bin, on l'ouvre puis on écrit le tableau pixmap enfin on ferme le fichier. Après cela, on fait appel au programme C. On ouvre le ficher CellValue.txt que le C a créé, on lit les données, on le ferme. A ce stade on connait le chiffe inscrit dans la case en question. On veut alors appeler le matlab avec la matrice faite de toutes les cases lues par le C. Pour cela on utilise beaucoup de concatenate strings pour que le format soit compatible avec le Matlab. A la sortie de la double boucle, le boolean est effectué pour afficher le PDF. On entre enfin de la case qui est réglée par le boolean du front panel 'appel de Matlab'. Si le résultat est vrai on créée le dossier Solve.m que l'on ouvre et dans lequel on écrit. Dans le cas contraire le matlab n'est pas appelé.

4.4 Gestion des erreures :

- **Image avec une taille différente de 400*400**
  => return -1 & print : "Erreur, la taille de l'image insérée n'est pas bonne!"
- Pour la gestion des erreurs dans le LabVIEW nous avons simplement relié le fil d'erreur à nos fonctions et à nos noeuds. Ce fil d'erreur est utilisé pour transmettre des informations sur les erreurs qui se produisent pendant l'éxécution du programme LabVIEW. Chaque fois qu'une fonction ou un nœud dans LabVIEW rencontre une erreur, il peut transmettre cette erreur le long du fil d'erreur pour informer les autres

parties du programme de la nature et de la cause de l'erreur. Tout particulièrement ce fil d'erreur passe par la fonction d'appel au programme C. A ce moment deux cas d'erreurs peuvent se présenter à nous. Dans un cas le programme est mal appelé et ne compile donc pas ce qui crée une erreur directe, dans l'autre cas le programme C est lancé mais renvoie une erreur. Dans ce cas, l'erreur sort dans le Labview comme une standard error. On veut donc savoir si standard error est vide ou non. Si elle n'est pas vide cela veut dire qu'il y a une erreur et on relie cette dernière au fil d'erreur.