|  |
| --- |
|  |
| Mini-Projet Programmation C |
| SUJET : JEU DE SUDOKU |
|  |
| **Rapport élaboré par : Bouaouina Omar & Louati Oussema** |
| **2017-2018** |

|  |  |
| --- | --- |
| Mini-Projet Programmation C | **2017-2018** |

**Remerciements**

**Nous tenons à remercier notre encadreur monsieur Mohamed Ramzi Haddad pour ses recommandations le long de ce projet afin de le réaliser.**

**Nous tenons également à remercier l'ENIT pour la formation que l'école nous a assuré durant la première semestre et qui nous a aidé à la réalisation du projet.**

|  |
| --- |
|  |

|  |  |
| --- | --- |
| Mini-Projet Programmation C | **2017-2018** |

**Sommaire**

* **Liste des figures Page 4**
* **Introduction Générale Page 5**
* **Partie I : Spécification**

1. **Introduction Page 6**
2. **Spécification Page 6**
3. **Conclusion Page 6**

* **Partie II : Conception**

1. **Introduction Page 7**
2. **Conception Page 7**
3. **Conclusion Page 21**

* **Partie III : Réalisation**

1. **Introduction Page 22**
2. **Implémentation Page 22**
3. **Tests de quelques cas Page 22**
4. **Conclusion Page 22**

* **Conclusion Générale Page 23**

|  |  |
| --- | --- |
| Mini-Projet Programmation C | **2017-2018** |

**Liste des figures**

* **Diagramme de Flux Page 8**
* **Figure 1 : Menu Principal Page 9**
* **Figure 2 : Quittance du jeu Page 9**
* **Figure 3 : Menu de « Créer une nouvelle partie » Page 10**
* **Figure 4 : Grille générée avec les méthodes de Page 11**

**remplissage possibles**

* **Figure 5 : Saisie de la grille Page 11**
* **Figure 6 : Grille saisie avec les méthodes de Page 12**

**remplissage possibles**

* **Figure 7 : Menu de « Reprendre une partie » Page 12**
* **Figure 8 : Un fichier texte comportant une grille Page 13**
* **Figure 9 : Reprise d’une partie en indiquant Page 13**

**le chemin du fichier texte**

* **Figure 10 : Grille obtenue à partir du fichier Page 14**

**texte avec les méthodes de remplissage**

**possibles**

* **Figure 11 : La console de la phase de remplissage Page 15**
* **Figure 12 : La sauvegarde et la quittance Page 16**

**de la partie**

* **Figure 13 : Saisie d’un coup et l’insertion de la Page 17**

**valeur dans la grille**

* **Figure 14 : Saisie d’un coup non possible Page 18**
* **Figure 15 : Remplissage incorrecte d’une case Page 19**
* **Figure 16 : Remplissage totale et correcte Page 20**
* **Figure 17 : Remplissage automatique Page 21**

**et les coordonnées des cases remplissent**

|  |  |
| --- | --- |
| Mini-Projet Programmation C | **2017-2018** |

**Introduction Générale**

Le sujet de notre mini-projet consiste à créer un jeu de SUDOKU qui est en forme de grille, formée par 9 lignes et 9 colonnes.

Le but du jeu est de remplir les cases vides de cette grille avec des chiffres, allant de 1 jusqu’à 9, tels qu’il y’en a jamais répétition d’un chiffre dans une même ligne, dans une même colonne ou dans une sous-grille qui est de la forme d’un carré 3x3 (appelée aussi bloc), donc dans une grille, on a un nombre total de 9 blocs.

La grille est initialement, partiellement remplisse pour pouvoir la résoudre.

Notre rapport se décompose en trois parties :

Partie I (Spécification) : qui comporte une description bref et générale des besoins fonctionnels et ceux non fonctionnels du cahier de charge.

Partie II (Conception) : qui comporte une explication plus profond des fonctions et les structures utilisés pour la création du jeu, ainsi qu’un diagramme montrons tous les cas possibles qu’on peut obtenir dès l’exécution du jeu.

Partie III (Réalisation) : qui comporte une description concernant l’implémentation du jeu, ainsi qu’un aperçu des tests de tous les cas mentionnés dans la deuxième partie.

Enfin, on terminera par une conclusion générale sur ce projet.

|  |  |
| --- | --- |
| Mini-Projet Programmation C | **2017-2018** |

**PARTIE I : Spécification**

I.1 Introduction :

Dans cette partie, on présente le cahier de charge de notre projet accompagné des descriptions des besoins fonctionnels et non fonctionnels.

I.2 Spécification :

Premièrement, comme besoins fonctionnels, le jeu de SUDOKU doit permettre à l’utilisateur de:

* Gérer des parties : précisément lui en permettre de créer une nouvelle partie (soit par saisir la grille manuellement ou soit par générer une), sauvegarder une partie, et reprendre une partie déjà sauvegardée.
* Résoudre des grilles (manuellement ou bien d’une manière automatique, c'est-à-dire par la machine).

Deuxièmement, pour ce qui concerne les besoins non fonctionnels du jeu, ils se résument dans les points suivants :

* Le code doit être clair pour permettre des futurs modifications et améliorations.
* L’ergonomie, c'est-à-dire qu’il doit offrir une interface facile à utiliser.

I.3 Conclusion :

Le jeu du SUDOKU offre à l’utilisateur une variété des choix qu’on va explorer dans les parties suivantes.

|  |  |
| --- | --- |
| Mini-Projet Programmation C | **2017-2018** |

**PARTIE II : Conception**

I.1 Introduction :

Dans cette partie, on présente la structure générale du programme en examinant les fonctions et les structures nécessaire pour son développement.

I.2 Conception :

* **Diagramme de Flux (voir page suivante) :**

|  |  |
| --- | --- |
| Mini-Projet Programmation C | **2017-2018** |

* **Démarrage du jeu :**

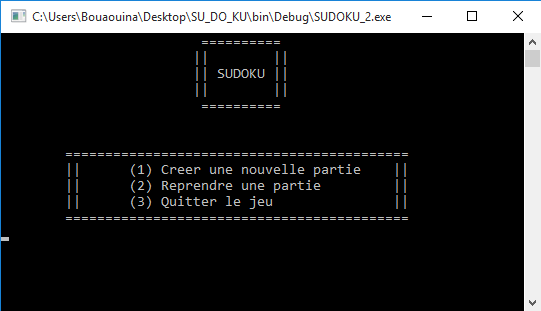
Le jeu démarre avec le menu principal :

FIGURE 1 : Menu Principal

Le programme permet à l’utilisateur de choisir entre la création d’une nouvelle partie, la reprise d’une partie, et la quittance du jeu.

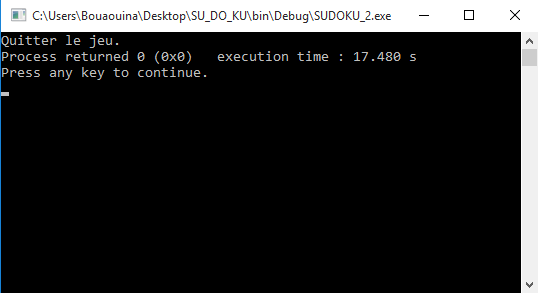
Commençons par le choix le plus simple, c’est de quitter le jeu. En prenons ce choix, cette nouvelle console apparait pour l’utilisateur :

FIGURE 2 : Quittance du jeu

|  |  |
| --- | --- |
| Mini-Projet Programmation C | **2017-2018** |

Apercevons maintenant les deux autres choix :

* **Création d’une nouvelle partie :**

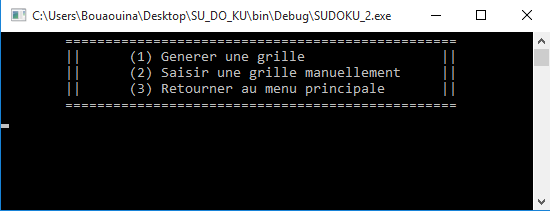
Si l’utilisateur opte pour le premier choix lors du démarrage du jeu, une nouvelle console apparait afin de lui indiquer les deux méthodes d’obtention d’une grille :

FIGURE 3 : Menu de « Créer une nouvelle partie »

1. **Génération d’une grille :**

En choisissons le premier choix, le programme fait appel à la fonction « void nouvellePartieGeneree(uneCase grille[9][9]) » . Cette fonction prend comme paramètre une grille 9x9 du type uneCase, un type qu’on a crée comme une structure contenant les deux champs « valeurCase » du type entier et « typeCase » du type « TYPEVALEUR », une énumération qu’on a crée comportant les constantes « ORIGINAL » assignée pour les valeurs fixes, « NOUVELLE » assignée pour les valeurs insérées par l’utilisateur et « VIDE » assignée pour les cases vides. En faisant appel à la fonction, elle remplit la grille avec des nombres entre 1 et 9 en les assignant le type de case ORIGINAL et en assurant qu’ils satisfaites les conditions d’une grille de SUDOKU à l’aide de la fonction « BOOL nombrePossible(int nbr,uneCase grille[9][9],int lig,int col) » du type « BOOL », une autre énumération qu’on a crée comportant les constantes « VRAI » et « FAUX », et qui retourne vrai si le nombre est seul sur ligne, seul sur colonne et sur le bloc 3x3, faux sinon. Après remplissage, la phase du vidage se déroule en remplaçant les valeurs de quelques cases par 0 et en les assignons le type de case VIDE et termine finalement par l’affichage de la grille :

|  |  |
| --- | --- |
| Mini-Projet Programmation C | **2017-2018** |

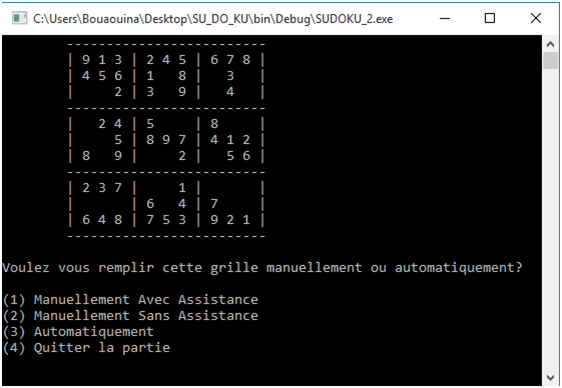


FIGURE 4 : Grille générée avec les méthodes du remplissage possible

1. **Saisie de la grille manuellement :**

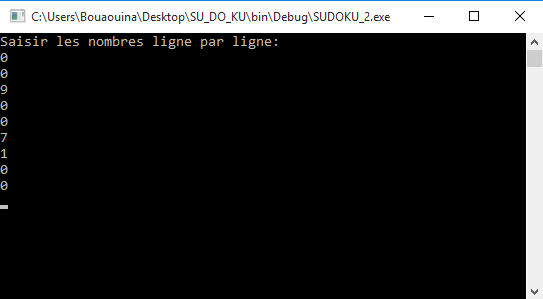
En choisissons le deuxième choix, le programme fait appel à la fonction « void nouvellePartieSaisie(uneCase grille[9][9]) » qui prend comme paramètre une grille 9x9 de type uneCase . Cette fonction demande à l’utilisateur de tapez les valeurs dans la grille ligne par ligne :

FIGURE 5 : Saisie de la grille

|  |  |
| --- | --- |
| Mini-Projet Programmation C | **2017-2018** |

en assignant le type de case VIDE pour les 0 et le type de case ORIGNAL pour les nombres entre 1 et 9. Après avoir saisie une grille correcte, le programme l’affiche :

FIGURE 6 : Grille saisie avec les méthodes du remplissage possible

* **Reprise d’une partie :**

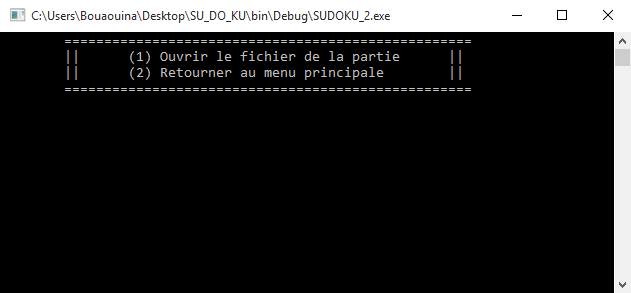
Si l’utilisateur opte pour le deuxième choix lors du démarrage du jeu, une nouvelle console apparait :

FIGURE 7 : Menu de « Reprendre une partie »

|  |  |
| --- | --- |
| Mini-Projet Programmation C | **2017-2018** |

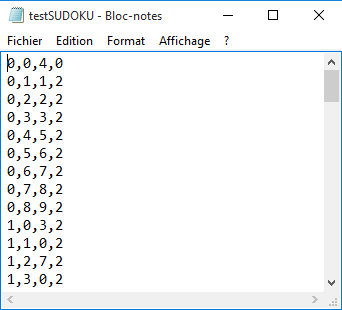
Dans notre programme, on a choisi les fichiers textes (.txt) pour la sauvegarde et la reprise des grilles, avec ces fichiers textes ayant la forme suivantes :

FIGURE 8 : Un fichier texte comportant une grille

Chaque ligne dans les fichiers textes représente une case, respectivement la ligne ou se trouve cette case, la colonne, sa valeur et le type de cette case (0 pour ORIGNAL, 1 pour NOUVELLE et 2 pour VIDE).

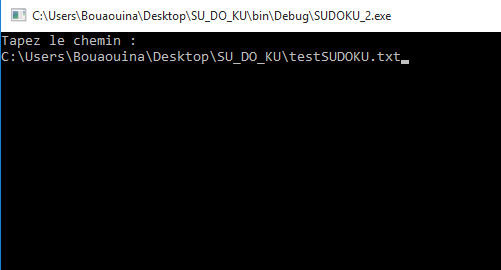
En choisissons d’ouvrir le fichier de la partie déjà sauvegardée, une autre console apparait pour indiquer à l’utilisateur de taper le chemin du fichier :

FIGURE 9 : Reprise d’une partie en indiquant le chemin du fichier texte

|  |  |
| --- | --- |
| Mini-Projet Programmation C | **2017-2018** |

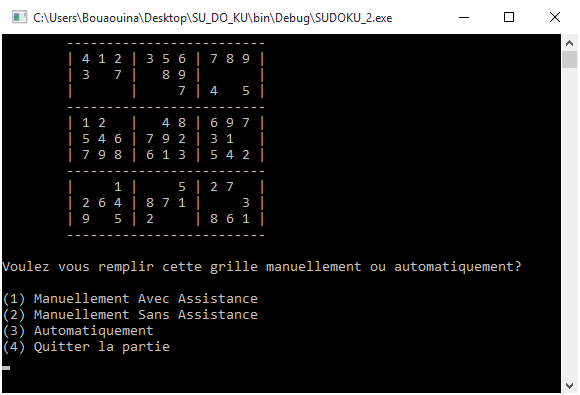
Après avoir saisie le chemin d’un fichier existant, le programme affiche la grille :

FIGURE 10 : Grille obtenue à partir du fichier texte avec les méthodes du remplissage possible

* **Résolution de la grille :**

Peu importe le choix d’obtention d’une grille, une console comportant cette grille et les méthodes de la remplir apparait pour l’utilisateur :

|  |  |
| --- | --- |
| Mini-Projet Programmation C | **2017-2018** |

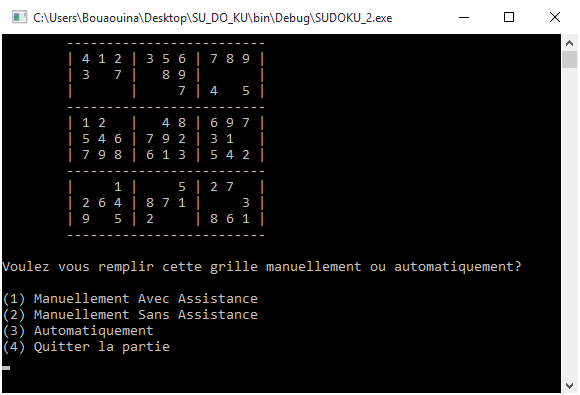


FIGURE 11 : La console de la phase de remplissage

Le programme permet à l’utilisateur à ce moment là de la remplir manuellement avec assistance, sans assistance, automatiquement ou bien de quitter.

Commençons par le choix le plus simple, c’est de quitter la partie. En prenons ce choix, on lui offre la possibilité de vouloir sauvegarder cette partie dans un fichier texte, en tapant « oui » en puis le chemin ou existera cette sauvegarde :

|  |  |
| --- | --- |
| Mini-Projet Programmation C | **2017-2018** |

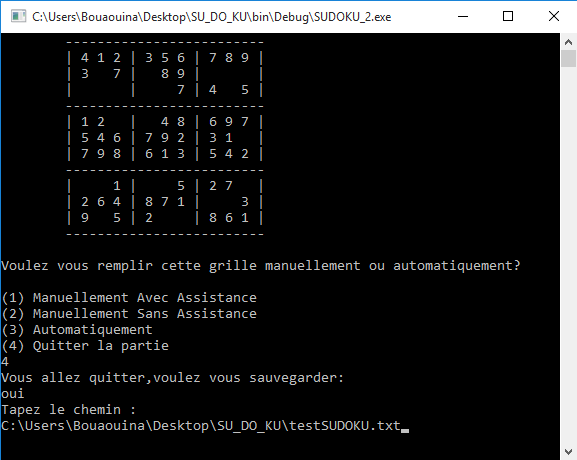


FIGURE 12 : La sauvegarde et la quittance de la partie

Apercevons maintenant les trois autres choix :

1. **Remplissage manuel avec assistance :**

En choisissons le premier choix du remplissage, le programme fait appel à la fonction « void demanderCoupHumain(char chaine[]) » qui permet à l’utilisateur soit de taper le coup, qui est de type leCoup, un type qu’on a crée comme une structure comportant les trois champs « numLigne »,  « numColonne » et « valeurCoup » de type entier, sous la forme « ligne/colonne/valeur » , soit de taper « quit » pour quitter la partie (avec la possibilité de sauvegarder déjà vu, ceci est encore possible avec le remplissage sans assistance). Ensuite, en faisant appel à la fonction « void verifChaineHumain(char chaine[],leCoup \*coup,uneCase grille[9][9]) » , le programme teste si la chaine insérée est « quit » d’où la quittance de la partie ou un coup qu’on va inséré à l’aide de la fonction « void insertionValeurPositionAssist(uneCase grille[9][9],leCoup \*coup) »

|  |  |
| --- | --- |
| Mini-Projet Programmation C | **2017-2018** |

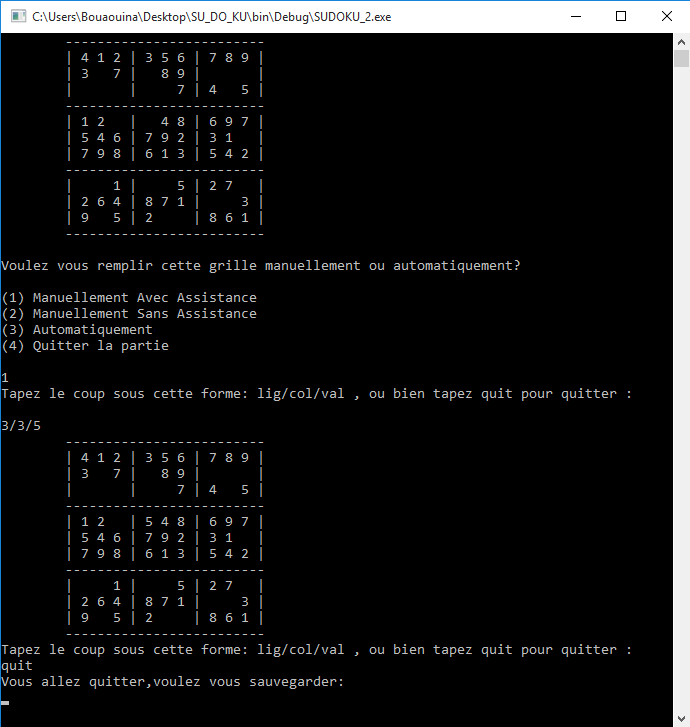
pour terminer finalement avec l’affichage de la grille avec cette modification :

FIGURE 13 : Saisie d’un coup et l’insertion de a valeur dans la grille

La phase de remplissage se déroule tant que l’utilisateur ne choisit pas de quitter ou tant que la grille n’est pas totalement remplisse.

Dans le cas ou l’utilisateur tape un nombre incorrect dans une case vide, le programme lui indique que ce nombre n’est pas possible :

|  |  |
| --- | --- |
| Mini-Projet Programmation C | **2017-2018** |

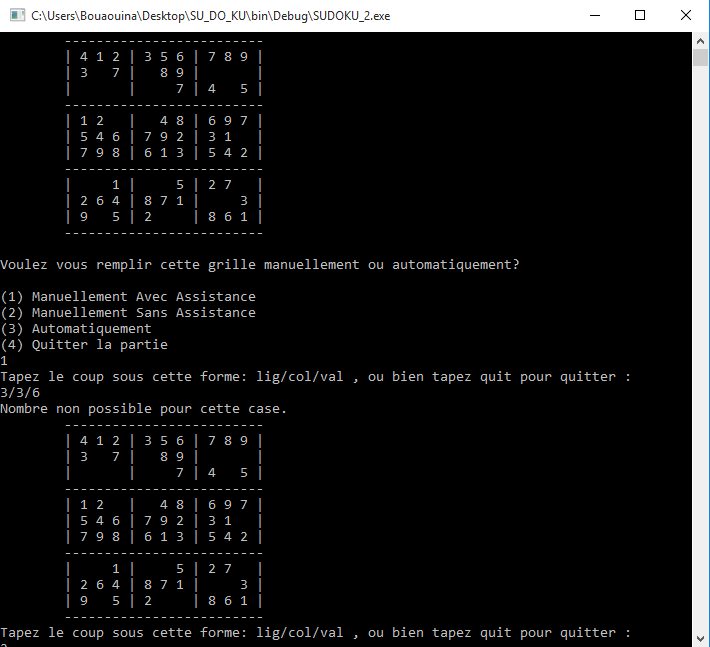


FIGURE 14 : Saisie d’un coup non possible

1. **Remplissage manuel sans assistance :**

En choisissons le deuxième choix du remplissage, une grille auxiliaire est crée à l’aide de la fonction « void copierGrille(uneCase grilleOriginale[9][9],uneCase grilleCopie[9][9]) » qui est ensuite résolue automatiquement par la fonction « int resolutionAutomatique(uneCase grille[9][9]) » qui parcourt cette grille ligne par ligne en remplissant les cases vides par des nombres possibles (c'est-à-dire qu’ils respectent les règles du jeu). Puis, comme avec le premier type de remplissage, le programme fait appel aux deux fonctions « void demanderCoupHumain(char chaine[]) » et « void verifChaineHumain(char chaine[],leCoup \*coup,uneCase grille[9][9]) », sauf que dans ce cas, la fonction d’insertion est « void insertionValeurPositionSeul(uneCase

|  |  |
| --- | --- |
| Mini-Projet Programmation C | **2017-2018** |

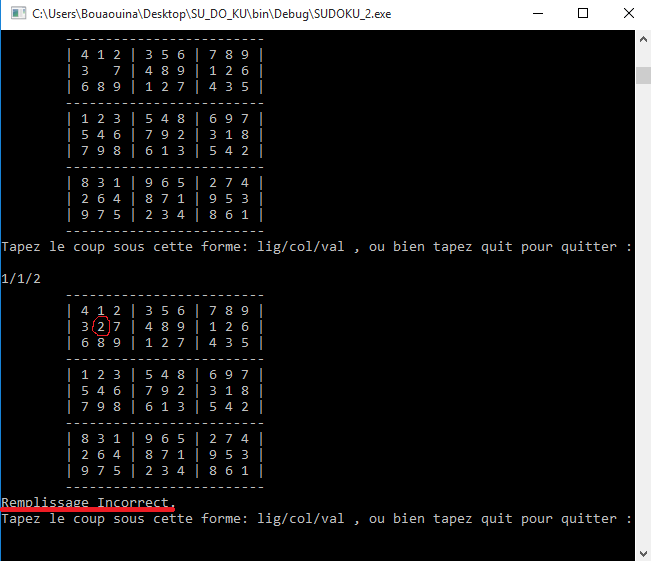
grille[9][9],leCoup \*coup) » qui n’indique pas à l’utilisateur si le nombre inséré est correct ou non. Lorsque la grille est totalement remplisse, le programme vérifie si le remplissage est correcte ou non, en comparant cette grille avec la grille auxiliaire contenant le remplissage correct, à l’aide de la fonction « int remplissageIncorrecte(uneCase grille[9][9],uneCase grilleAux[9][9]) » :

FIGURE 15 : Remplissage incorrecte d’une case

La phase de remplissage se déroule tant que l’utilisateur ne choisit pas de quitter ou tant que la grille n’est pas totalement et correctement remplisse.

|  |  |
| --- | --- |
| Mini-Projet Programmation C | **2017-2018** |

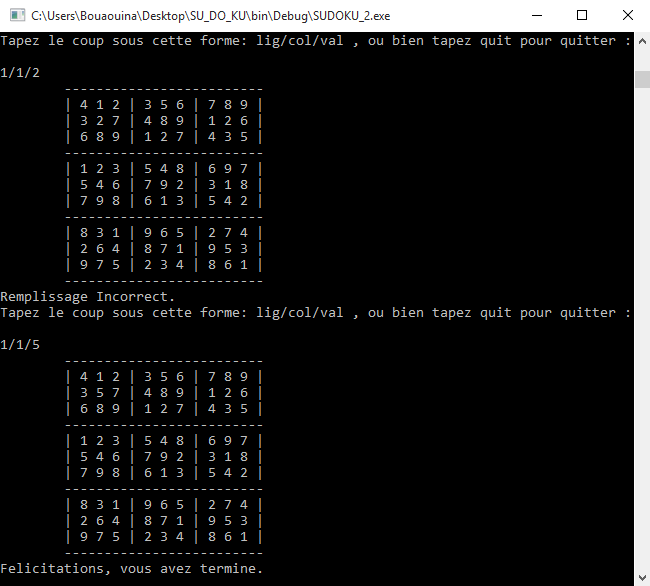


FIGURE 16 : Remplissage totale et correcte

1. **Remplissage automatique :**

En choisissons le dernier choix du remplissage, le programme fait appel à la fonction « void remplissageTotalMachine(uneCase grille[9][9]) » qui affiche la grille remplisse, si c’est possible, en utilisant la fonction « int resolutionAutomatique(uneCase grille[9][9]) » mentionné dans le remplissage sans assistance, avec les cases qui ont été remplisses :

|  |  |
| --- | --- |
| Mini-Projet Programmation C | **2017-2018** |

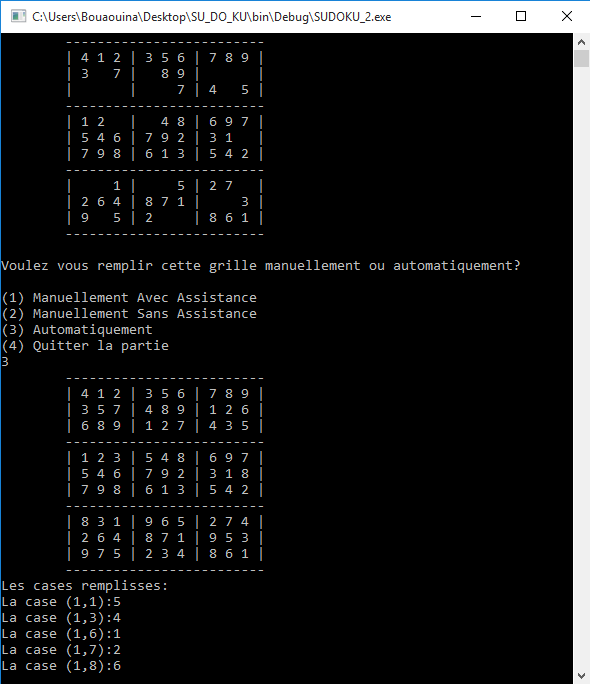


FIGURE 17 : Remplissage automatique et les coordonnées des cases remplissent

I.3 Conclusion :

Cette partie nous a expliqué avec plus de détails ce qui a été présenté dans le diagramme de flux.

|  |  |
| --- | --- |
| Mini-Projet Programmation C | **2017-2018** |

**PARTIE III : Réalisation**

I.1 Introduction :

Dans cette partie, nous parlons de la manière de l’implémentation du jeu ainsi que présentons de quelques cas.

I.2 Implémentation :

Le jeu a été développé en langage C avec l’IDE (environnement du développement intégré) « CodeBlocks ». On a choisi des tableaux statiques de dimensions 9x9 pour les grilles et les fichiers textes (.txt) pour la sauvegarde et la reprise des grilles (comme déjà indique à la page 13).

I.3 Tests de quelques cas :

* Dans les menus de « Créer une nouvelle partie » et « Reprendre une partie », on peut choisir de retourner au menu principal.
* Dans le cas ou l’utilisateur choisit de quitter une partie, s’il tape autre chose que « oui » lors de la demande de vouloir sauvegarder ou non, le programme considère ça comme non.
* Si l’utilisateur tape un nombre pour un choix inexistant dans l’un des menus, le programme lui affichera « Choix invalide, veuillez choisir correctement » et lui permettra d’essayer une autre fois
* Lorsque l’utilisateur tape le chemin de sauvegarde, si le fichier n’existe pas, alors il sera crée à ce moment là.
* Lors de remplissage, si l’utilisateur tape quelque chose autre qu’un coup de la forme « ligne/colonne/valeur » ou « quit », le programme lui affichera « Erreur du coup » et lui permettra d’essayer une autre fois.

I.4 Conclusion :

Avec le langage C, on a pu arriver à ce point avec notre projet.

|  |  |
| --- | --- |
| Mini-Projet Programmation C | **2017-2018** |

**Conclusion Générale**

Nous avons essayé d’expliquer le plus possible le fonctionnement du programme, ça n’indique pas qu’il est parfait vu la nécessité d’améliorer le générateur des grilles ainsi que la possibilité d’implémentation dynamique du jeu, mais c’est pour cela qu’on espère de perfectionner le jeu.