Initiation à la Programmation en C (L1 CPEI)*

Projet 2019

Résolution du sudoku

§1. Informations

Groupes. Ce projet doit être fait par groupe de 2. Les groupes de 1 ou 3 ne sont pas acceptés, sauf dans certaines circonstances exceptionnelles.

Rendu. Comme pour les TP notés, le rendu de votre projet se fera sur Moodle (rubrique "*Projet*"). La date limite est le **14 Avril 2019 à 23h59**. Aucun retard ne sera accepté.

Soutenance. Un après-midi (la date exacte sera annoncé ultérieurement) sera dédié aux soutenances des projets. Vous serez invité à présenter et expliquer votre implémentation, et à répondre à quelques questions. Votre note finale dépendra en partie de cette soutenance. Sauf cas exceptionnel, votre projet ne pourra être soutenu qu'une seule fois, même si vous passez en session 2.

Note finale. Le projet est fait en groupe, mais la note est individuelle, et dépend du rendu, de votre contribution personnelle, et de la soutenance. **Comme pour les TP notés, les rendu ne compilant pas recevront la note 0/20**. Pour rappel, ce projet compte pour 1/3 de la note finale du cours, que vous passiez en session 1 ou 2.

Tricherie. La communication et l'aide entre groupes est autorisée, mais les rendus trop similaires seront lourdement sanctionnés. Si vous ne comprenez pas votre code, cela se verra tout de suite lors de la soutenance. Nous rappelons que le plagiat (du travail d'autres groupes, ou de ressources sur internet) est passible de renvoi.

*Cours donné par prof. Roberto Amadio. Moniteur 2019 : Cédric Ho Thanh. TPs/TDs basés sur ceux des précédents moniteurs : Florien Bourse (2017), Antoine Dallon (2018). Autres contributeurs : Juliusz Chroboczek, Gabriel Radanne.

§2. Le backtracking

Comment résoudre une grille de sudoku ? Un humain utilise un certain nombre de méthodes plus ou moins compliquées et heuristiques, mais un ordinateur peut se permettre une approche plus directe.

Le *backtracking* est un algorithme général pour résoudre des problèmes en testant toutes les possibilités jusqu'à trouver la bonne. Par exemple, l'algorithme d'énumération des permutations du cours repose sur le backtracking. Pour résoudre une grille de sudoku, on procède comme suit.

1. On choisit une case vide, et on y met un chiffre, par exemple 1 (même s'il y a une forte chance que ça ne soit pas le bon), **en vérifiant tout de même qu'il n'est pas déjà présent dans la même ligne, colonne, ou bloc**.
2. On choisit une autre case vide, et on recommence...
3. Au bout d'un moment, on trouvera très probablement un case vide pour laquelle aucun chiffre n'est possible. Cela signifie que nous avons écrit un chiffre incorrect quelque part, mais où ? Une manière de contourner ce problème est de revenir à la dernière case que l'on a rempli, et d'essayer un autre chiffre. Si l'on a épuisé toutes les possibilités, alors on revient à la case que l'on avait remplit avant. En bref, *à chaque fois que l'on est bloqué, on revient sur le dernier choix que l'on a fait. Si on est toujours bloqué, on revient sur le choix précédent, etc.*
4. On va donc épuiser toutes les possibilités de remplissage de la grille jusqu'à trouver la bonne.

§3. Énoncé

Vous trouverez ci-dessous quelques étapes à traiter. L'énoncé est volontairement incomplet et parfois même vague. Vous êtes donc libre de procéder comme vous le souhaitez, tant que vous produisez un code robuste (qui résiste aux erreurs), lisible, et de qualité. **Notez que le barème n'est donné qu'à titre indicatif.**

§3.1. Étapes obligatoires

Étape 1. (1 point) **Pour ce projet, vous utiliserez un tableau à une seule dimension pour stocker une grille de sudoku**. Réfléchissez bien à comment cela peut être fait. Pensez en particulier aux cases vides. Implémentez la fonction `print_grid` qui affiche à l'écran une grille de sudoku de manière lisible.

Étape 2. (2 points) Implémentez la fonction `read_grid` qui lit une grille de sudoku depuis un ficher et la stocke en mémoire. Vous devez être capable de correctement traiter le fichier suivant, contenant une grille 9×9 :

```
1 2 3 4 5 6 7 8 9
4 5 6 7 8 9 1 2 3
7 8 9 1 2 3 4 5 6
2 1 4 3 6 5 8 9 7
3 6 5 8 9 7 2 1 4
8 9 7 2 1 4 3 6 5
5 3 1 6 4 2 9 7 8
6 4 2 9 7 8 5 3 1
9 7 8 5 3 1 6 4 2
```

Bien entendu, cette grille est déjà pleine. Comment cette fonction doit-elle se comporter si le fichier ne contient pas une grille valide ? si le ficher contient des espaces et des retours à la ligne aux mauvais endroits ?

Étape 3. (1 point) Implémentez la fonction `write_grid` qui écrit une grille de sudoku dans un fichier de manière lisible.

Étape 4. (1 point) Implémentez la fonction `check_grid` qui vérifie si une grille est valide.

Étape 5. (5 points) Implémentez la fonction `solve` qui implémente l'algorithme du backtracking pour remplir des grilles de sudoku. Vous pouvez procéder de manière itérative ou récursive. Comment cette fonction doit-elle se comporter si la grille est totalement vide ? déjà pleine ? s'il n'existe pas de solution ?

§3.2. Bonus

Les extensions libres sont encouragées, et seront notées suivant leur intérêt et leur difficulté. Voici quelques exemples. Assurez-vous cependant d'avoir terminé les étapes obligatoires avant de commencer les bonus.

Bonus 1. (1 point) Rendez votre programme interactif. Au minimum, votre programme devrait demander à l'utilisateur quel fichier ouvrir, afficher la solution, et lui proposer de l'enregistrer.

Bonus 2. (1 point) Votre code comporte des fonctions intermédiaires judicieuses, et aussi peu de duplication de code que possible.

Bonus 3. (1 point) Votre code est parfaitement indenté. Vous pouvez utiliser un outil dédié, par exemple `cpplint`.

Bonus 4. (1 point) Votre code est parfaitement documenté. Vous pouvez utiliser un outil dédié, par exemple `doxygen`.

Bonus 5. (5 points) Modifiez votre programme afin qu'il prenne en charge des grilles de taille $n^2 \times n^2$, où $n \geq 1$ (les grilles de sudoku classiques sont celles où $n = 3$). La valeur de n sera demandée à l'utilisateur avant qu'il (ou elle) choisisse le fichier à ouvrir.

Bonus 6. (5 points) Comme vous l'avez peut-être remarqué (?), la complexité du backtracking est exponentielle en la taille de la grille. Implémentez un meilleur algorithme qui utilise des méthodes "humaines" quand c'est possible, et recourt au backtracking dans les cas plus difficiles.