

Date de l'énoncé : Mercredi 29 novembre 2017

Date de remise : Vendredi 15 décembre 2017

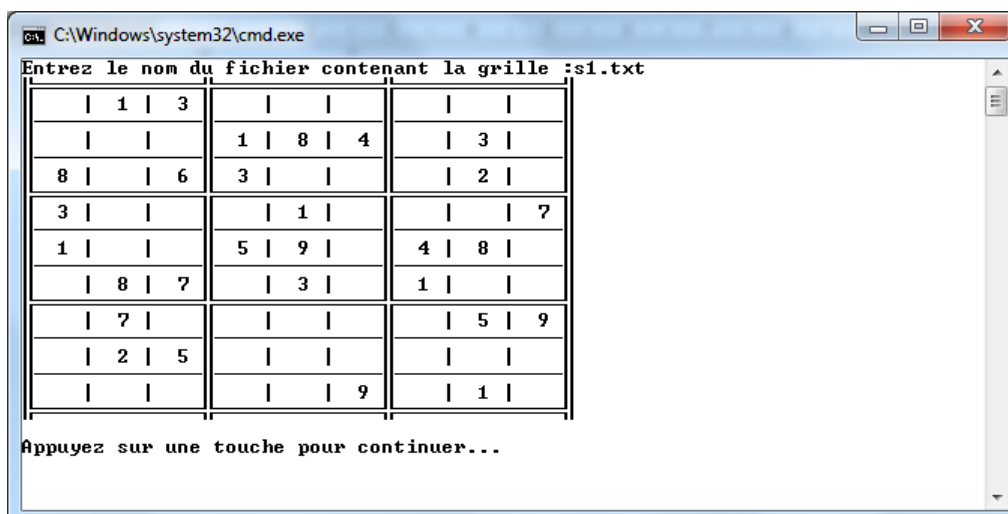
***** Sudoku *****
Travail pratique # 3

Le **Sudoku** est un jeu de déduction apparu il y a environ 8 ou 9 ans. Il s'agit d'une grille de 9 carreaux horizontaux par 9 carreaux verticaux soit 81 carreaux au total dans lesquels on doit seulement inscrire des chiffres de 1 à 9 selon les règles suivantes :

1. Sur chaque ligne, les chiffres de 1 à 9 ne doivent apparaître qu'une seule fois dans les 9 cases horizontales.
2. Sur chaque colonne, les chiffres de 1 à 9 ne doivent apparaître qu'une seule fois aussi dans les 9 cases verticales.
3. Dans chaque sous-ensemble de 9 carreaux (3 X 3) à partir de la position 0,0 les chiffres de 1 à 9 ne doivent apparaître qu'une seule fois également.

Au départ, la grille de jeu qui est proposée est partiellement complétée. Le jeu consiste à la compléter. Votre travail pratique consistera à écrire un programme pour trouver les chiffres manquants de la grille et solutionner le jeu dans le temps le plus court possible.

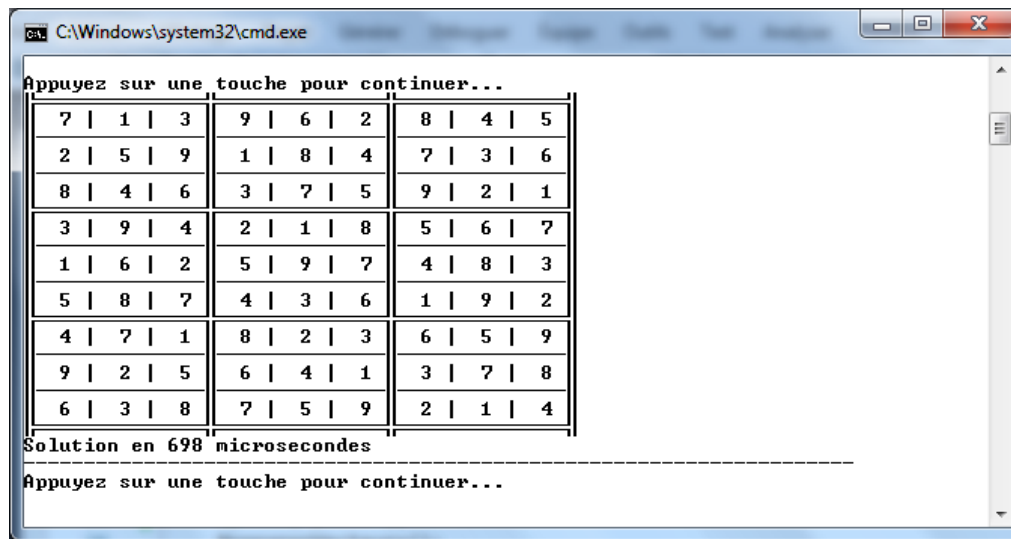
Par exemple, on pourrait avoir un fichier s1.txt qui, une fois lu, produirait la grille de départ suivante :



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

Appuyez sur une touche pour continuer...

Une fois votre programme exécuté, il devra trouver la solution :



Notez que pour chaque carreau de 3 X 3, les chiffres de 1 à 9 ne se répètent qu'une seule fois dans la solution :

<table> <tr><td>7</td><td>1</td><td>3</td></tr> <tr><td>2</td><td>5</td><td>9</td></tr> <tr><td>8</td><td>4</td><td>6</td></tr> </table>	7	1	3	2	5	9	8	4	6	<table> <tr><td>9</td><td>6</td><td>2</td></tr> <tr><td>1</td><td>8</td><td>4</td></tr> <tr><td>3</td><td>7</td><td>5</td></tr> </table>	9	6	2	1	8	4	3	7	5	<table> <tr><td>8</td><td>4</td><td>5</td></tr> <tr><td>7</td><td>3</td><td>6</td></tr> <tr><td>9</td><td>2</td><td>1</td></tr> </table>	8	4	5	7	3	6	9	2	1
7	1	3																											
2	5	9																											
8	4	6																											
9	6	2																											
1	8	4																											
3	7	5																											
8	4	5																											
7	3	6																											
9	2	1																											
<table> <tr><td>3</td><td>9</td><td>4</td></tr> <tr><td>1</td><td>6</td><td>2</td></tr> <tr><td>5</td><td>8</td><td>7</td></tr> </table>	3	9	4	1	6	2	5	8	7	<table> <tr><td>2</td><td>1</td><td>8</td></tr> <tr><td>5</td><td>9</td><td>7</td></tr> <tr><td>4</td><td>3</td><td>6</td></tr> </table>	2	1	8	5	9	7	4	3	6	<table> <tr><td>5</td><td>6</td><td>7</td></tr> <tr><td>4</td><td>8</td><td>3</td></tr> <tr><td>1</td><td>9</td><td>2</td></tr> </table>	5	6	7	4	8	3	1	9	2
3	9	4																											
1	6	2																											
5	8	7																											
2	1	8																											
5	9	7																											
4	3	6																											
5	6	7																											
4	8	3																											
1	9	2																											
<table> <tr><td>4</td><td>7</td><td>1</td></tr> <tr><td>9</td><td>2</td><td>5</td></tr> <tr><td>6</td><td>3</td><td>8</td></tr> </table>	4	7	1	9	2	5	6	3	8	<table> <tr><td>8</td><td>2</td><td>3</td></tr> <tr><td>6</td><td>4</td><td>1</td></tr> <tr><td>7</td><td>5</td><td>9</td></tr> </table>	8	2	3	6	4	1	7	5	9	<table> <tr><td>6</td><td>5</td><td>9</td></tr> <tr><td>3</td><td>7</td><td>8</td></tr> <tr><td>2</td><td>1</td><td>4</td></tr> </table>	6	5	9	3	7	8	2	1	4
4	7	1																											
9	2	5																											
6	3	8																											
8	2	3																											
6	4	1																											
7	5	9																											
6	5	9																											
3	7	8																											
2	1	4																											

Éléments techniques à respecter

Vous lirez un fichier de string et vous écrirez votre solution à la console. Je vous fournirai une méthode d'affichage pour vous simplifier la vie, ainsi que le main requis pour votre programme.

La grille de jeu sera représentée par un objet **Matrice** de 9 lignes par 9 colonnes. Chaque case de la matrice représentera une case de la grille de jeu et devra donc contenir un nombre de 1 à 9 lorsque la solution sera trouvée.

On retrouvera dans un fichier les données de départ de la grille de jeu. Ce fichier contiendra une suite de caractères représentant les données de la grille; si le caractère est un chiffre de 1 à 9, il s'agit d'un indice qui se trouve dans la case de la grille de départ et cette valeur est immuable,

c'est-à-dire que vous ne devez pas la changer en cours de recherche d'une solution. Si le caractère est une étoile, ceci signifie que la case est à combler dans la grille de départ et que votre programme doit en trouver la valeur. Il n'y a pas de validation nécessaire du fichier qui sera considéré comme correct en toutes circonstances.

Voici deux exemples de contenu de fichiers corrects :

2*****9** ***7*9*5* 5*6324*** 3**1*65*7 1***9***** *5***8*** **46*7*** *8*4**3** **2**5**1	239***** *****5* 4***79*13 **2**8**9 *974***** *****1*2* *85*6***** 32**4*6** *****5*8
--	--

Contexte de réalisation

À la construction de votre objet Sudoku, vous devrez lire le fichier dont on reçoit la référence en paramètre. Une référence sur un flux de sortie sera aussi reçue en paramètre et conservée par l'objet.

Le fichier contient des caractères de 1 à 9 ainsi que des étoiles pour marquer les cases que le programme doit compléter. Dans la matrice représentant le jeu, le plus simple est sans doute d'utiliser une matrice d'entiers; utilisez une valeur sentinelle (0 ou -1 par exemple) dans les cases 'vides' à compléter pour pouvoir les identifier. Cette valeur vous permettra de détecter les cases à compléter par votre algorithme.

Je m'attends naturellement à ce que vous utilisiez la classe *template* **Matrice** fondée sur un *vector* de *vector* que nous avons étudié et utilisé en début de session.

Votre classe **Sudoku** devra aussi mettre à la disposition du programme client une méthode publique **Solutionner()** qui fera appel à une méthode récursive pour trouver la solution.

Votre recherche de solution doit utiliser les principes de la récursivité et du *backtracking*. C'est donc dire que ce travail n'est pas un travail qui exige beaucoup de code. Il exige au contraire de faire une réflexion algorithmique afin de trouver une solution qui puisse s'exécuter dans un délai raisonnable.

Pour vous donner une idée, la solution récursive que j'ai mise au point comporte, pour l'ensemble de toutes les méthodes obtenues après découpage, moins de 150 lignes de code. Une version optimisée m'a pris 300 lignes de code. C'est dire que ce n'est pas un travail très long. C'est un travail qui vous demande toutefois de penser de manière différente.

Implantez dans votre projet la classe **Chrono** que vous pourrez instancier pour chronométrer le temps requis à votre programme pour trouver la solution. Après avoir affiché la solution, vous indiquerez combien de temps votre programme a mis à la trouver.

Votre programme doit faire en sorte que le temps de recherche de la solution soit le plus court possible. Un programme qui ne trouverait la solution qu'après un très long délai de traitement sera considéré comme fonctionnant très mal et sera corrigé en conséquence. On s'entend qu'un très long délai de traitement se calcule en secondes et non pas en minutes. Attendre plus de 30 secondes pour une solution est excessif.

Ma solution, pour n'importe quelle grille de départ, a trouvé la réponse en moins d'une seconde. C'est donc un objectif très atteignable.

Remise du travail

La date de remise est fixée au vendredi 15 décembre 2017. Un retard est possible jusqu'au lundi 18 décembre à 17h00. Passé cette échéance, je me réserve le droit de ne plus recevoir votre travail et y attribuer la note 0, la raison étant que je dois remettre mes notes finales au collègue et je ne veux pas être en attente de vos travaux.

Vous devez me fournir une copie imprimée de votre projet. Cette copie imprimée contient le *main* en premier suivi des fichiers .H et .CPP des différentes parties de votre code. Les fichiers doivent être dans cet ordre et **brochés** (la brocheuse n'est pas fournie avec l'imprimante, à vous de prévoir). L'impression doit se faire en **mode paysage**, comme prévu dans la norme du département. Vous glisserez cette copie imprimée sous ma porte de bureau avant la fin de la journée de la remise.

Vous devez également m'envoyer la version électronique de votre projet afin que je le teste. Procédez par courriel dans une archive zippée sous le nom **NomPrénom.zip**. Si vous êtes en équipe de deux personnes, le nom doit être **NomPrénometNomPrénom.zip**. Expurgez de votre projet les éléments non nécessaires pour réduire la taille du fichier.

Le barème de correction sera :

Fonctionnement (tests)	60%
Autres	40% -2 pts par erreur « ordinaire » -4 par erreur grave -6 par erreur abominable -10 pour une implantation incorrecte

Bon travail.

Pierre

P.S. Je vous invite à réfléchir à la notion de ‘quadrant’ à l’intérieur de votre grille de Sudoku. Ainsi, on peut dire que la grille comporte 9 quadrants et que, dans chaque quadrant, les nombres de 1 à 9 doivent apparaître une seule fois...

1	2	3
4	5	6
7	8	9