

Ne vous connectez pas avec votre mot de passe habituel !

☞ Utiliser l'identifiant `interro1` et le mot de passe `blabla`

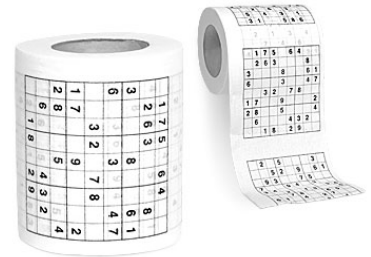
Comme promis, le sujet traite de Sudoku.

Initialement, la grille de Sudoku contient des cases vides et des cases déjà remplies avec un chiffre de 0 à 9.

Nous nous intéressons au processus de résolution d'une grille. Nous travaillons sur des grilles contenant dans chaque case un ensemble de chiffres possibles (au lieu de contenir juste un chiffre). Cet ensemble est représenté par un tableau de 9 booléens. Par exemple :

1	2	3	4	5	6	7	8	9
True	True	False	False	True	False	False	True	False

valeurs possibles : {1, 2, 5, 8}



- Si la valeur de la case est connue, un seul chiffre est possible : toutes les cases du tableau de booléens sont à False, sauf une.
- Si la valeur de la case est inconnue, plusieurs chiffres sont possibles : plusieurs cases du tableau de booléen sont à True.
- Au départ, les cases vides contiennent les 9 chiffres possibles : le tableau de booléen ne contient que des True.

Une grille contient donc 9×9 cases dont chacune est représentée par un tableau de 9 booléens

Consignes

- La partie 2 peut être commencée sans avoir terminé la partie 1.
- D'une manière générale, vous pouvez laisser de côté certaines questions et passer à la suite lorsque vous êtes bloqués. Dans ce cas, mettez un commentaire pour indiquer que c'est volontaire de votre part.

Consignes administratives

- Le barème donné est approximatif, il pourra être ajusté.
- PENSEZ À METTRE VOTRE NOM AU DÉBUT DE CHAQUE PROGRAMME

Principaux critères d'évaluation

- Le programme compile sans erreur et remplit correctement la mission.
- Le code est indenté (touche TAB) et contient des commentaires judicieux et dosés.
- L'algorithme est le plus simple possible.
- Les noms des types et des variables sont bien choisis.
- Les procédures de test sont présentes et pertinentes.

Dans cette première partie, nous allons simplement éliminer chaque valeur connue de la ligne, colonne, et sous-matrice à laquelle cette valeur appartient (et exception faite de la case contenant cette valeur).

- Ouvrez le fichier mission1.adb et mettez votre nom au début. Examinez l'acteur sudogrille.ads
- Juste avant le **begin** du programme principal, déclarer une variable Sudoku initialisée avec une grille de niveau Piece_Of_Cake. Dans le corps du programme principal, afficher cette grille.
- Définissez un type article `Un_Résultat` comprenant deux attributs entiers : un attribut Combien et un attribut Valeur. Ce type sert à compter le nombre de valeurs possibles d'une case.
- Écrivez une fonction `Compte_Possibles` qui prend en argument un tableau de type `Les_Chiffres_Possibles` et qui renvoie `Un_Résultat`. Cette fonction compte le nombre de cases à True (ce nombre est mis dans l'attribut Combien) et trouve un numéro de case dont la valeur est True (ce numéro de case est mis dans l'attribut Valeur). Par exemple, pour le tableau donné au recto de la feuille, la fonction doit renvoyer un résultat avec Combien = 4 et Valeur = 8 (ou 1, ou 2, ou 5). **Remarquez que lorsque Combien = 1, la case est complètement déterminée, et elle contient le chiffre égal à Valeur.** Le cas Combien = 0 n'est pas possible, et la fonction doit échouer (avec Failif) si cela se produit.
- Pour tester cette fonction, affichez le résultat pour la case (2,1) et le résultat pour la case (2,2) de la grille Piece_Of_Cake. Ajouter un commentaire dans votre programme pour expliquer ce que l'on doit observer.
- Écrire une fonction `Total_Possibles` qui prend en argument une grille et renvoie le nombre de valeurs possibles sur toute la grille (c'est-à-dire, le nombre total de True contenus dans toutes les cases de la grille). Tester, ajouter un commentaire pour expliquer ce que l'on doit obtenir.

- Écrire une procédure `Propage` qui prend en argument **in out** une grille, et en arguments **in** un numéro de ligne, de colonne et une valeur entière. Cette procédure est appelée lorsque la case (Ligne, Colonne) ne contient qu'une seule valeur possible. Elle doit alors éliminer cette valeur des chiffres possibles de toute la ligne, de toute la colonne et de toute la sous-matrice (sauf dans la case concernée, évidemment).
 - ▷ Écrivez cette procédure par étapes, en éliminant d'abord sur toute la ligne (sauf la case Ligne, Colonne). Tester avec (Sudoku, 1, 1, 9) en affichant la grille après propagation.
 - ▷ Les coordonnées de la sous-matrice concernée sont DepLigne .. DepLigne + 2 (pour les lignes) et DepCol .. DepCol + 2 (pour les colonnes), avec :

$$\text{DepLigne} = 3 * ((\text{Ligne} - 1) / 3) + 1 \quad (\text{attention aux parenthèses})$$

$$\text{DepCol} = 3 * ((\text{Colonne} - 1) / 3) + 1$$
- Écrire une procédure `Resous` qui prend en argument **in out** une grille, et effectue, en boucle, les opérations suivantes :
 - ▷ Afficher la grille, attendre une touche.
 - ▷ Propager les valeurs déterminées (procédure Propage) de toutes les cases déterminées. Dans un premier temps, faire une boucle infinie. Puis, une fois que cela fonctionne, ajouter une condition pour arrêter dès que la grille est résolue. Enfin, affiner la condition pour arrêter dès que la grille est résolue ou dès que le nombre total de chiffres possibles de la grille cesse de décroître (ce qui signifie que notre technique de résolution ne suffit pas à trouver la solution).
- Tester avec différents niveaux : cette technique permet de résoudre complètement la grille Piece_Of_Case seulement.

- Ouvrez le fichier mission2.adb et ajoutez-y le type `Un_Résultat` défini à la partie 1.
- Le programme Mission2 déjà écrit remplit le même rôle que la partie 1. Vous allez le compléter pour le rendre plus malin : si un chiffre x n'est possible que dans une seule case C d'une ligne, d'une colonne, ou d'une sous-grille, c'est que le chiffre x doit forcément se trouver en C et on peut donc éliminer les autres possibilités de la case C .
Par exemple, si sur la première ligne, le chiffre 1 n'est possible que dans la troisième case (cette case contenant d'autres chiffres possibles, disons 2 et 6), c'est que la troisième case **doit** contenir 1, et on peut donc éliminer le 2 et le 6 de cette case.
Voici comment procéder :
- Pour commencer, compléter le corps du programme principal en appelant la procédure `Resous` sur la variable `Soduk` que vous initialisez avec la grille moyenne. Vérifiez que ça compile et en exécutant, vérifiez que la grille est partiellement résolue, comme dans la partie 1.
- Écrire une fonction `Possibles_Ligne` qui prend en argument une grille, un numéro de ligne, et une valeur entière, et renvoie `Un_Résultat`. Cette fonction parcourt la ligne de la grille et compte combien de fois la valeur est possible sur la ligne. Le résultat renvoyé contient dans l'attribut `Combien` le nombre de fois que la valeur est possible sur la ligne, et dans l'attribut `Valeur` un numéro de colonne où la valeur est possible.
- Écrire une procédure `Examiner_Ligne` qui prend en argument **in out** une grille et un numéro de ligne, et, pour chaque chiffre, examine si le chiffre n'est possible que dans une seule case de la ligne. Si c'est le cas, il faut éliminer les autres valeurs de la case concernée (une fonction de l'acteur `SudoGrille` peut servir). Ne cherchez pas à tester cette procédure tout de suite, sauf si vous avez une idée de test simple.
- Compléter la boucle principale pour effectuer ce traitement sur toutes les lignes.
- Faites de même pour toutes les colonnes. Maintenant le programme est capable de résoudre la grille moyenne.
- Enfin, faites de même pour les sous-matrices. Un peu d'initiative est nécessaire. Ceci permet de résoudre la grille difficile (mais ça ne suffit pas en général).
- La grille diabolique n'est pas résolue par cette méthode. À vous de trouver d'autres techniques.

