TP : Solveur de Sudoku

Objectifs : Ce TP a pour but de résoudre des SUDOKU.

1 Obtenir des problèmes de Sudoku

1.1 Depuis un terminal

Vous pouvez obtenir des problèmes de SUDOKU en utilisant la commande :

```
wget --post-data "methode=num&numero=10&format=txt" http://www.lifl.fr/~wegrzyno/Sudoku/obtenirGrill
```

Cette commande affiche à l'écran le problème numéro 10 sous la forme

```
 0 5 6 0 3 9 0 8 1
 9 0 0 0 0 8 0 0 0
 0 8 0 0 0 2 3 0 9
 0 0 2 0 8 0 0 0 0
 0 0 7 6 0 0 2 0 4
 4 0 5 2 0 1 8 9 0
 3 4 8 0 0 0 1 6 0
 5 0 0 0 1 0 0 0 0
 0 0 0 3 5 4 0 2 8
---
1 (niveau grille)
---
10 (numéro grille)
```

Vous pouvez rediriger l'affichage dans un fichier texte avec la commande :

```
wget --post-data "methode=num&numero=10&format=txt" http://www.lifl.fr/~wegrzyno/Sudoku/obtenirGrill
```

Dans la commande précédente, vous pouvez remplacer `numero=10` par `numero=`n avec $1 \leq n \leq 2895$.

1.2 Depuis un navigateur

En tapant l'adresse `http ://www.lifl.fr/w̃egrzyno/Sudoku/` dans un navigateur Internet, vous obtenez un formulaire dans lequel vous devez préciser

1. le type de choix : par niveau de difficulté ou par numéro ;
2. le format de réception du problème : format texte ou format HTML.

Pour pouvoir soumettre le problème de SUDOKU à votre programme choisissez le format texte, et enregistrez-le dans votre dossier personnel.

2 Lire et afficher un Sudoku

On utilise les déclarations suivantes (cf cours) :

```
const
   RN = 3  ;
   N  = RN*RN  ;
type
   SYMBOLE = 0..N;
   INDICE  = 1..N;
   STATUT  = 0..succ(N);
   POSSIBILITE = array[SYMBOLE] of BOOLEAN;
```

```
CELLULE = record
    val : SYMBOLE;              // valeur de la case
    possibles : POSSIBILITE; // valeurs possibles pour la case
end {CELLULE};
SUDOKU =  array[INDICE,INDICE] of CELLULE;
```

et on suppose donnée la procédure de mise à jour

```
// mise à jour des  champs possibles après attribution
// d'une valeur à la case de coordonnées ival, jval
procedure mettreAJour(var sudo : SUDOKU;
                      const ival,jval : INDICE);
var
    i,j : INDICE;
    ci,cj,ii,jj : 0..RN;
begin
    // mise à jour des champs possibles de la ligne ival
    for j := low(INDICE) to high(INDICE) do
        if (j<>jval) and (sudo[ival,j].val = 0) then
            sudo[ival,j].possibles[sudo[ival,jval].val] := FALSE;
    // mise à jour des champs possibles de la ligne jval
    for i := low(INDICE) to high(INDICE) do
        if (i<>ival) and (sudo[i,jval].val = 0) then
            sudo[i,jval].possibles[sudo[ival,jval].val] := FALSE;
    // mise à jour des champs possibles du carré local à (ival,jval)
    ci := (ival-1) div RN ;
    cj := (jval-1) div RN ;
    for ii := 1 to RN do begin
        i := ci*RN + ii;
        for jj := 1 to RN do begin
            j := cj*RN + jj;
            if ((i<>ival) or (j<>jval)) and (sudo[i,j].val=0) then
                sudo[i,j].possibles[sudo[ival,jval].val] := FALSE;
        end {for};
    end {for};
end {mettreAJour};
```

Q 1 . Réalisez une procédure de lecture d'un SUDOKU et une procédure d'affichage.
La procédure de lecture doit pouvoir lire des données présentées sous cette forme :

```
0 5 6 0 3 9 0 8 1
9 0 0 0 0 8 0 0 0
0 8 0 0 0 2 3 0 9
0 0 2 0 8 0 0 0 0
0 0 7 6 0 0 2 0 4
4 0 5 2 0 1 8 9 0
3 4 8 0 0 0 1 6 0
5 0 0 0 1 0 0 0 0
0 0 0 3 5 4 0 2 8
---
1 (niveau grille)
---
10 (numéro grille)
```

La procédure d'affichage doit produire un affichage de même forme (sans les quatre dernières lignes).
Q 2 . Vérifiez le bon fonctionnement de vos deux procédures (lecture et affichage) avec l'exemple ci-dessus que vous saisissez "à la main".
Q 3 . Vérifiez ensuite avec un fichier contenant une grille obtenu par l'un ou l'autre des procédés décrits dans la section 1. Pour cela, utilisez la commande

```
./toto < grille10.txt
```

où `toto` doit être remplacé par le nom de votre programme.

3 Détermination du statut d'un Sudoku

Une grille de SUDOKU peut
- avoir une ou plusieurs solutions ;
- n'avoir aucune solution.

Le *statut* d'une grille sera codé par un entier compris entre 0 et $N+1$:
- si le statut est égal à $N+1$ cela signifie que la grille est entièrement remplie ;
- si le statut est inférieur ou égal à N, il représente le degré de liberté minimal d'une case de la grille. En particulier, s'il est nul, cela signifie que la grille n'a pas de solution.

```
type
  STATUT  =  0..succ(N);
```

Q 4 . Réalisez la procédure spécifiée ci-dessous.

```
// st indique le statut du sudoku lu :
//  st=0 : sudoku sans solution
//  st=succ(N) : sudoku résolu
//  0<st<=N : degré de liberté minimale d'une case
//  imin, jmin : coordonnées de la case à degré de liberté minimale
procedure determinerStatut(const sudo : SUDOKU;
                           out st     : STATUT;
                           out imin,jmin : INDICE);
```

4 Résolution d'un Sudoku

Q 5 . Réalisez la procédure spécifiée ci-dessous.

```
// sudo est une grille non remplie
// imin et jmin sont les coordonnées d'une case
// vide
procedure resoudre(const sudo : SUDOKU;
                   const imin,jmin : INDICE);
```

Q 6 . Écrivez un programme qui lit un SUDOKU et affiche toutes ses solutions.

Testez votre programme avec les grilles `grille10.txt`, `grille21.txt`, et `grille765-1.txt` (les deux premières sont les exemples du cours, la dernière a plusieurs solutions).

5 Résolution d'un Sudoku 16×16

Q 7 . Que devez-vous changer dans votre programme pour pouvoir résoudre un problème de SUDOKU 16×16 ?

Testez avec cette grille.