Saé S1.01 : implémentation d'un besoin client

Le jeu de Sudoku

Consignes pour la partie 2 - Programmation

NB : en annexe vous trouverez des consignes précises pour l'élaboration du cahier de tests et pour la programmation en langage C du jeu de Sudoku.

Les livrables que vous devez fournir sont les suivants.

- Un **cahier de tests** où sont rassemblés tous les cas de test pour chaque test prévu à l'annexe 1.
- Le **code source** commenté qui devra respecter les consignes fournies en annexe 2.
- Une **documentation** Doxygen de votre code source, au format texte (fichier *rtf*).
- Une **trace d'exécution** de chaque cas de test prévu dans le cahier de tests. Une trace prendra la forme d'une copie-écran qui montre que le cas de test a bien été réalisé.

Il est conseillé de commencer la programmation du jeu de Sudoku dès le début de cette période car c'est sans doute ce qui vous demandera la plus de temps.

ANNEXE 1: liste des tests à prévoir

Test 1 : chargement du fichier

Test 2 : affichage de la grille

Test 3 : saisie coordonnées d'une case

Test 4 : saisie d'une valeur à insérer

Test 5 : fin de partie

ANNEXE 2 : consignes pour le programme Sudoku

Le programme doit permettre à un joueur humain de remplir une grille de Sudoku. Les valeurs d'une grille de Sudoku seront stockées dans un tableau 2D d'entiers.

Structure de données

Pour le stockage des valeurs d'une grille de sudoku, vous utiliserez dans ce programme le type :

```
type tGrille = tableau[1..TAILLE, 1..TAILLE] de entier ;
```

Par convention, une case non remplie prendra la valeur 0. Les autres auront des valeurs comprises entre 1 et n^2.

Algorithme principal

```
programme sudoku c'est

   type tGrille = tableau[1..TAILLE, 1..TAILLE] de entier[1];

   var   grille1 : tGrille;
         numLigne, numColonne, valeur : entier;

début
   chargerGrille(grille1);
   tant que la grille n'est pas pleine[2] faire
        afficherGrille(grille1);
        ecrireEcran("Indices de la case ? ");
        saisir(numLigne);
        saisir(numColonne);
        si (grille1[numLigne][numColonne]!=0)alors
            ecrireEcran("IMPOSSIBLE, la case n'est pas libre.");
        sinon
            ecrireEcran("Valeur à insérer ? ");
            saisir(valeur);
            si (possible(grille1, numLigne, numColonne, valeur)) alors
                grille1[numLigne][numColonne] = valeur;
            finsi
        finsi
    finfaire
    ecrireEcran("Grille pleine, fin de partie");
fin
```

1 À vous de définir les constantes N et TAILLE du programme.
2 À vous de trouver comment vérifier qu'une grille est pleine

Liste des fonctions/procédures à programmer

procédure *chargerGrille*

Cette procédure vous est fournie ci-dessous. Elle permet de charger en mémoire une grille de sudoku existante à partir d'un fichier dont le nom est lu au clavier.

paramètre
S : la grille de jeu

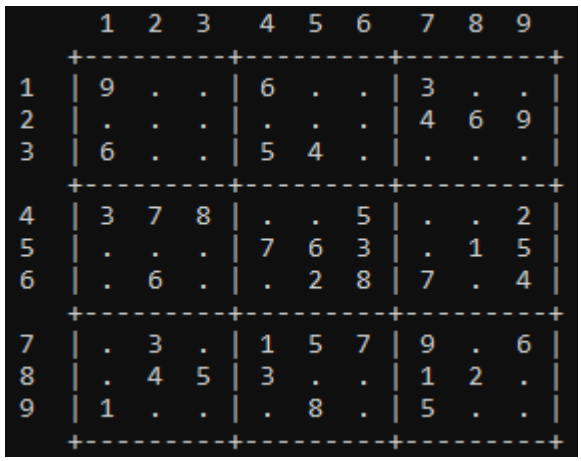
```
void chargerGrille(tGrille g){
    char nomFichier[30];
    FILE * f;

    printf("Nom du fichier ? ");
    scanf("%s", nomFichier);
    f = fopen(nomFichier, "rb");
    if (f==NULL){
        printf("\n ERREUR sur le fichier %s\n", nomFichier);
    } else {
        fread(g, sizeof(int), TAILLE*TAILLE, f);
        fclose(f);
    }

}
```

procédure *afficherGrille*

Cette procédure réalise l'affichage à l'écran de la grille et de son contenu, conformément à cet exemple :



Remarquez les caractères +, - et | qui permettent de "dessiner" le cadre de la grille. Remarquez aussi que pour faciliter le jeu, la procédure affichera les numéros de ligne et de colonne.

Sur une ligne, chaque valeur sera encadrée par un espace de chaque côté.

Une case vide sera représentée par un point.

paramètre
E : la grille de jeu à afficher

procédure *saisir*

Cette procédure est chargée de lire au clavier une valeur. La saisie se répète tant que la valeur n'est pas valide. En effet, la valeur lue doit être un **entier,** et cet entier **doit être compris entre 1 et n^2**. Les messages d'erreur devront être explicites.
Cette procédure sert aussi bien à saisir une numéro de ligne ou un numéro de colonne qu'à saisir une valeur à insérer dans la grille.

paramètre
S : la valeur lue, entier compris entre 1 et n^2

Remarque
En langage C, pour tester qu'une valeur est un entier, on peut utiliser la fonction `sscanf`. Le principe est de lire au clavier une chaine de caractères et ensuite de tester si cette valeur peut être convertie en entier : c'est le rôle de `sscanf` qui retourne 0 si la conversion n'est pas possible.

Dans l'exemple suivant, `ch` est une variable de type chaine de caractères et `x` est la variable de type entier qui contiendra la valeur lue si la conversion de chaine en entier a réussi.

```
scanf("%s", ch);
if (sscanf(ch, "%d", &x) !=0){
     // la conversion a réussi, x contient la
     // valeur entière lue au clavier
} else {
     //la conversion en entier a échoué
}
```

fonction *possible*

Cette fonction vérifie si une valeur peut être ajoutée dans une case particulière de la grille, c'est-à-dire si cela respecte les règles du sudoku :
- la valeur n'est pas déjà présente sur la même ligne que la case
- la valeur n'est pas déjà présente sur la même colonne que la case
- la valeur n'est pas déjà présente dans le même bloc que la case

En cas d'erreur, il faudra afficher un message explicite.

paramètres
E : la grille de jeu
E : entier, correspond au numéro de ligne de la case visée
E : entier, correspond au numéro de colonne de la case visée
E : la valeur à insérer dans la grille

résultat
booléen : VRAI si la valeur peut être ajoutée à la grille ; FAUX sinon.