



Présentation les fonctions du projet Sudoku

- La liste des fonctions:

```
void initialiser();  
void imprimerSudoku();  
void entrerSudoku();  
void genererFichierImprimerSudoku();  
void genererFichierCompletSudoku();  
void genererFichierModeleSudoku();  
void swapPetitLigne(int ligne_1, int ligne_2);  
void swapGrandeLigne(int lgrande_1, int lgrande_2);  
void swapPetitColonne(int col_1, int col_2);  
void swapGrandColonne(int colgrande_1, int colgrande_2);  
void rotationVertical();  
void rotationHorizontale();  
void swapColonneLigne();  
void inverserTab();  
void remplacement(int a, int b);  
void genererSudoku();
```

void initialiser(int sudokuTab[9][9])

- Pour assigner le tableau `sudoku[9][9] = exempleTab[9][9]`
- Il s'agit simplement de créer un tableau et de l'affecter au tableau existante.

```
void initialiser(int sudokuTab[9][9]){  
    int i, j;  
  
    for(i = 0; i < TAILLE; i++) {  
        for(j = 0; j < TAILLE; j++) {  
            sudoku[i][j] = sudokuTab[i][j];  
        }  
    }  
}
```

```
int exempleTab[TAILLE][TAILLE] = {  
    {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 }  
};
```


void imprimerSudoku()

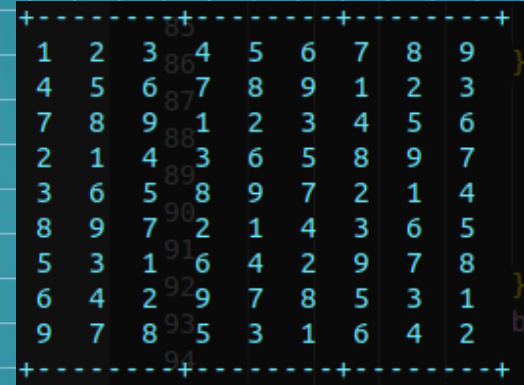
- Pour imprimer le sudoku sur Terminal

```
void imprimerSudoku(){
    int i, j;

    printf("+-----+-----+-----+\n");

    for(i = 0; i < TAILLE; i++) {
        for(j = 0; j < TAILLE; j++) {
            printf(" %d ", sudoku[i][j]);
        }
        printf("\n");
    }

    printf("+-----+-----+-----+\n");
}
```



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

void entrerSudoku()

- Pour entrer une table de sudoku incomplète et imprimer le résultat du solveur

```
void entrerSudoku(){
    int i, j;

    for(i=0; i<TAILLE; i++){
        for(j=0; j<TAILLE; j++){
            scanf("%d", &sudoku[i][j]);
        }
    }

    imprimerSudoku();
    solveSudoku(0, 0);
}
```

```
Entrez le sudoku souhaité et entrez 0 pour les entrées inconnues:
1 0 3 4 0 0 7 0 9
0 5 6 0 8 9 0 2 3
0 8 9 1 0 3 4 0 6
2 1 4 0 6 5 0 9 7
3 0 0 8 0 7 0 1 4
8 0 7 0 1 4 0 6 5
0 3 1 0 4 0 9 7 8
6 4 0 9 7 0 5 3 1
0 7 8 0 0 1 0 4 2

+-----+
1 0 3 4 0 7 0 9
0 5 6 0 8 9 0 2 3
0 8 9 1 0 3 4 0 6
2 1 4 0 6 5 0 9 7
3 0 0 8 0 7 0 1 4
8 0 7 0 1 4 0 6 5
0 3 1 0 4 0 9 7 8
6 4 0 9 7 0 5 3 1
0 7 8 0 0 1 0 4 2

+-----+
LE SUDOKU RÉSOLU:
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 3 6 5 4
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

+-----+

141
142     printf("-----+-----\n");
143 }
144
145 void entrerSudoku(){
146     int i, j;
147
148     for(i=0; i<TAILLE; i++){
149         for(j=0; j<TAILLE; j++){
150             scanf("%d", &sudoku[i][j]);
151         }
152     }
153     imprimerSudoku();
154     solveSudoku(0, 0);
155 }
156
157 void swapPetitLigne(int ligne,
158 int j, templ);
159
160 for(j = 0; j < TAILLE; j++)
161     templ = sudoku[ligne_1
162
163     sudoku[ligne_1][j] = s
164     sudoku[ligne_2][j] = t
165 }
166
167 void swapGrandeLigne(int lgran
168 if(lgrande_1 > 2 || lgrand
169 red();
170 printf("ERREUR DE GRAN
171 reset();
172
173 else /
```


void swapPetitLigne(int ligne_1, int ligne_2)

- Pour trier, échanger la position de deux nombres **ligne_1** et **ligne_2** dans chaque ligne de 1 à 9.

```
void swapPetitLigne(int ligne_1, int ligne_2){  
    int j, tempL;  
  
    for(j = 0; j < TAILLE; j++) {  
        tempL = sudoku[ligne_1][j];  
  
        sudoku[ligne_1][j] = sudoku[ligne_2][j];  
        sudoku[ligne_2][j] = tempL;  
    }  
}
```

void swapGrandeLigne(int lgrande_1, int lgrande_2)

- Identique à la fonction **swapPetitLigne()** mais avec 3 petites lignes à la fois

```
void swapGrandeLigne(int lgrande_1, int lgrande_2){  
    if(lgrande_1 > 2 || lgrande_2 > 2){  
        red();  
        printf("ERREUR DE GRANDE LIGNE !!!");  
        reset();  
    }else {  
        swapPetitLigne(lgrande_1 * 3, lgrande_2 * 3);  
        swapPetitLigne(lgrande_1 * 3 + 1, lgrande_2 * 3 + 1);  
        swapPetitLigne(lgrande_1 * 3 + 2, lgrande_2 * 3 + 2);  
    }  
}
```


void swapPetitColonne(int col_1, int col_2)

- Identique à la fonction **swapPetitLigne()** mais pour échanger la position de deux nombres **col_1** et **col_2** dans chaque colonne de 1 à 9.

```
void swapPetitColonne(int col_1, int col_2){  
    int i, tempC;  
  
    for(i = 0; i < TAILLE; i++) {  
        tempC = sudoku[i][col_1];  
  
        sudoku[i][col_1] = sudoku[i][col_2];  
        sudoku[i][col_2] = tempC;  
    }  
}
```



void swapGrandColonne(int colgrande_1, int colgrande_2)

```
void swapGrandColonne(int colgrande_1, int colgrande_2){  
    if(colgrande_1 > 2 || colgrande_2 > 2){  
        red();  
        printf("ERREUR DE GRANDE COLONNE !!!");  
        reset();  
    }else {  
        swapPetitColonne(colgrande_1 * 3, colgrande_2 * 3);  
        swapPetitColonne(colgrande_1 * 3 + 1, colgrande_2 * 3 + 1);  
        swapPetitColonne(colgrande_1 * 3 + 2, colgrande_2 * 3 + 2);  
    }  
}
```


Les fonctions rotation:

- Pour faire pivoter la planche de sudoku verticalement et horizontalement.
- Ce qui est tourné, c'est que les position de tous les nombres dans le tableau sont modifiées dans le sens de la rotation.

```
void rotationVertical(){
    int i, j, tempV;

    for(i = 0; i < TAILLE; i++) {
        for(j = 0; j < TAILLE/2; j++) {
            tempV = sudoku[i][j];

            sudoku[i][j] = sudoku[i][8 - j];
            sudoku[i][8 - j] = tempV;
        }
    }
}
```

```
void rotationHorizontale(){
    int i, j, tempH;

    for(i = 0; i < TAILLE/2; i++) {
        for(j = 0; j < TAILLE; j++) {
            tempH = sudoku[i][j];

            sudoku[i][j] = sudoku[8 - i][j];
            sudoku[8 - i][j] = tempH;
        }
    }
}
```

void swapColonneLigne()

- Pour changer l'ordre des coordonnées du tableau sudoku:
`sudoku[ligne][colonne]` ↔ `sudoku[colonne][ligne]`

```
void swapColonneLigne(){
    int i, j, tempSwapColonneLigne;

    for(i = 0; i < TAILLE; i++) {
        for(j = i; j < TAILLE; j++) {
            tempSwapColonneLigne = sudoku[i][j];

            sudoku[i][j] = sudoku[j][i];
            sudoku[j][i] = tempSwapColonneLigne;
        }
    }
}
```


void inverserTab() et void inverserTab2()

- Pour inverser les positions des nombres dans le tableau

```
void inverserTab(){
    int i, j;
    int sudokuTemp[TAILLE][TAILLE];

    for(i = 0; i < TAILLE; i++) {
        for(j = 0; j < TAILLE; j++) {
            sudokuTemp[i][j] = sudoku[i][j];
        }
    }

    for(i = 0; i < TAILLE; i++) {
        for(j = 0; j < TAILLE; j++) {
            sudoku[i][j] = sudokuTemp[j][i];
        }
    }
}

void inverserTab2(){

    rotationVertical();
    inverserTab();
}
```

void genererSudoku()

```
void genererSudoku(){
    initialiser(exempleTab);
    srand(time(NULL));

    int melanger, i, bloc;

    for(i = 0; i < 10; i++) {

        melanger = rand() % 10;
        bloc = rand() % 3;

        switch(melanger) {
            case 0:
                switch(bloc) {
                    case 0:
                        swapPetitLigne(rand() % 3, rand() % 3);
                        break;

                    case 1:
                        swapPetitLigne((rand() % 3) + 3, (rand() % 3) + 3);
                        break;

                    case 2:
                        swapPetitLigne((rand() % 3) + 6, (rand() % 3) + 6);
                        break;
                }
                break;
            case 1:
                swapGrandeLigne((rand() % 3), (rand() % 3));
                break;
            case 2:
                switch(bloc) {
                    case 0:
                        swapPetitColonne((rand() % 3), (rand() % 3));
                        break;

                    case 1:
                        swapPetitColonne((rand() % 3) + 3, (rand() % 3) + 3);
                        break;

                    case 2:
                        swapPetitColonne((rand() % 3) + 6, (rand() % 3) + 6);
                        break;
                }
                break;
        }
    }
}
```

```
case 2:
    switch(bloc) {
        case 0:
            swapPetitColonne((rand() % 3), (rand() % 3));
            break;

        case 1:
            swapPetitColonne((rand() % 3) + 3, (rand() % 3) + 3);
            break;

        case 2:
            swapPetitColonne((rand() % 3) + 6, (rand() % 3) + 6);
            break;
    }
    break;
case 3:
    swapGrandColonne((rand() % 3), (rand() % 3));
    break;

case 4:
    rotationVerticale();
    break;

case 5:
    rotationHorizontale();
    break;

case 6:
    swapColonneLigne();
    break;

case 7:
    inverserTab();
    break;

case 8:
    inverserTab2();
    break;

case 9:
    remplacement((rand() % 9) + 1, (rand() % 9) + 1);
    break;
    }
}
```

- Pour mélanger aléatoire tout les positions des nombres dans la table sudoku en appelant toutes les fonctions ci-dessus et en ajoutant les **rand()**
- Cette nouvelle table fusionnée est valide parce que les fonctions de swap, inverse, rotate sont placées séparément dans chaque **case**.

