INFO0952

Feedback Devoir 1

Dernière mise à jour le 23 octobre 2018

Solution : check_unique

Une première solution :

```
int check_unique(int tab[], int SIZE) {

  for (int i = 1; i < SIZE; i++)
    for (int j = 0; j < i; j++)
      if (tab[i] == tab[j])
         return 0;

  return 1;

}
```

(Invariant : `tab[0..i-1]` ne contient pas de doublon.)

Solution : check_unique

Une deuxième solution (équivalente à la première) :

```
int check_unique(int tab[], int SIZE) {

  for (int i = 0; i < SIZE-1; i++)
    for (int j = i+1; j < SIZE; j++)
      if (tab[i] == tab[j])
         return 0;

  return 1;

}
```

(Invariant : les éléments de `tab[0..i-1]` n'apparaissent qu'une seule fois dans le tableau)

Solution : check_unique

Une troisième solution (inefficace !) :

```
int check_unique(int tab[], int SIZE) {

  for (int i = 0; i < SIZE; i++)
    for (int j = 0; j < SIZE; j++)
      if (i != j && tab[i] == tab[j])
         return 0;

  return 1;

}
```

Fait deux fois plus de tests que nécessaire

Invariant ? ?

Solution : check_unique

Une quatrième solution :

```
int check_unique(int tab[], int SIZE) {

  for (int number = 1; number <= SIZE; number++) {
    int j = 0;
    while (j < SIZE && tab[j] != number)
       j++;
    if (j == SIZE)
       return 0;
  }

  return 1;

}
```

Invariant : boucle externe : tab contient les nombres de 0 à i-1.

Suppose que tab ne contient que des chiffres de 1 à SIZE.

Solution : check_unique

Une cinquième solution (la plus efficace !) :

```
int check_unique(int tab[], int SIZE) {

  int mask[SIZE];

  for (int i = 0; i < SIZE; i++)
    mask[i] = 0;

  for (int i = 0; i < SIZE; i++) {
    int cur = tab[i];
    if (mask[cur-1])
      return 0;
    mask[cur-1] = 1;
  }

  return 1;
}
```

(Invariant : `mask[k]=1` si `k+1` apparaît dans `tab[0..i-1]`, 0 sinon)

Solution : check_unique

Une solution incorrecte :

- Contre-exemple : `tab = [2 2 2 4 5 6 7 8 9]`

```
int check_unique(int tab[], int SIZE) {

  int sum = 0;

  for (int i = 1; i < SIZE; i++)
      sum += tab[i];

  return (sum == SIZE*(SIZE+1)/2);

}
```

Le produit ne marche pas non plus.

- Contre-exemple : `[1 6 3 4 5 6 7 8 3]`.

Peut-être que les deux combinés fonctionnent mais alors il faut le prouver.

Solution : is_sudoku

```c
int is_sudoku(int **grid, int SIZE) {
  int arr[SIZE]; // array used for check_unique

  // check rows
  for(int i = 0; i < SIZE; i++)
    if (!check_unique(grid[i], SIZE))
       return 0;

  // check columns
  for (int i = 0; i < SIZE; i++) {
    for (int j = 0; j < SIZE; j++)
      arr[j] = grid[j][i];
    if (!check_unique(arr, SIZE))
      return 0;
  }

  // check squares
  const int SQRT_SIZE = sqrt(SIZE);
  for (int i = 0; i < SIZE; i++) {
    int current_corner_r = (i % SQRT_SIZE) * SQRT_SIZE;
    int current_corner_c = (i / SQRT_SIZE) * SQRT_SIZE;
    for (int j = 0; j < SIZE; j++) {
      arr[j] = grid[current_corner_r + j % SQRT_SIZE][current_corner_c + j / SQRT_SIZE];
    }
    if (!check_unique(arr, SIZE))
       return 0;
  }

  return 1;
}
```

(Possible aussi de l'implémenter avec 4 boucles imbriquées)

Problèmes récurrents

- Nombreux problèmes d'indentation :

```
int is_sudoku(int **grid, int SIZE) {

int i = 0;
int j = 0;
...
```

- Lignes trop longues (en particulier les commentaires) (essayer de réduire à 80 caractères maximum)
- Trop d'espaces
- Ne pas laisser les `printf` de débogage, même en commentaires
- Utilisation de `while` quand un `for` pourrait être utilisé
- Déclarer les variables aussi près que possible de leur utilisation :
  - Ne pas définir tous les indices de boucle au début de la fonction
  - Préférer '`for (int i = 0;...)`' à '`for (i = 0;...)`'
- Vérifier les Warnings de compilation ! (11 devoirs en déclenchent).

Problèmes récurrents

- Variables inutiles :

```
check = check_unique(tab, SIZE);
if (check != 1)
   return 0;
```

$\Rightarrow$

```
if (!check_unique(tab, SIZE))
   return 0;
```

- Construction inefficace :

```
check1=check_unique(tablig,SIZE);
check2=check_unique(tabcol,SIZE);
if(check1==0)
        return 0;
if(check2==0)
        return 0;
```

$\Rightarrow$

```
if (!check_unique(tablig, SIZE))
   return 0;
if (!check_unique(tabcol, SIZE))
   return 0;
```

- Eviter de modifier les indices de boucles

```
for (int i = 0; i < SIZE; i++){
    for (int j = 0; j < SIZE; j++){
            if (j == i)
                j++;
            if (tab[i] == tab[j])
               return 0;
    }
}
```

$\Rightarrow$

```
for (int i = 0; i < SIZE; i++){
    for (int j = 0; j < SIZE; j++){
            if (j == i)
                break;
            if (tab[i] == tab[j])
               return 0;
    }
}
```