**Sudoku Game Program Explanation**

**Source Code :**

Here is the source code to implement and solve sudoku program in C.

#include <stdio.h>

#define N 9

*// Function declarations*

int isSafe(int grid[N][N], int row, int col, int num);

int findUnassignedLocation(int grid[N][N], int \*row, int \*col);

int solveSudoku(int grid[N][N]);

void printGrid(int grid[N][N]);

int main() {

int grid[N][N];

*// Get Sudoku grid from the user*

printf("Enter the Sudoku grid (row by row, use 0 for empty cells):**\n**");

**for** (int i = 0; i < N; i++) {

**for** (int j = 0; j < N; j++) {

scanf("%d", &grid[i][j]);

}

}

**if** (solveSudoku(grid)) {

printf("**\n**Solution:**\n**");

printGrid(grid);

} **else** {

printf("**\n**No solution exists.**\n**");

}

**return** 0;

}

*// Check whether it's safe to place the number 'num' at position (row, col)*

int isSafe(int grid[N][N], int row, int col, int num) {

*// Check if 'num' is not present in the current row, column, and the 3x3 subgrid*

**for** (int x = 0; x < N; x++) {

**if** (grid[row][x] == num || grid[x][col] == num || grid[row - row % 3 + x / 3][col - col % 3 + x % 3] == num) {

**return** 0;

}

}

**return** 1;

}

*// Find an unassigned location in the Sudoku grid*

int findUnassignedLocation(int grid[N][N], int \*row, int \*col) {

**for** (\*row = 0; \*row < N; (\*row)++) {

**for** (\*col = 0; \*col < N; (\*col)++) {

**if** (grid[\*row][\*col] == 0) {

**return** 1; *// Found an unassigned location*

}

}

}

**return** 0; *// No unassigned location found*

}

*// Solve the Sudoku puzzle using backtracking*

int solveSudoku(int grid[N][N]) {

int row, col;

*// Check if there is any unassigned location*

**if** (!findUnassignedLocation(grid, &row, &col)) {

**return** 1; *// No unassigned location, puzzle is solved*

}

*// Try placing a number from 1 to 9 in the current unassigned location*

**for** (int num = 1; num <= 9; num++) {

**if** (isSafe(grid, row, col, num)) {

*// Assign the number if it's safe*

grid[row][col] = num;

*// Recur to try and solve the rest of the puzzle*

**if** (solveSudoku(grid)) {

**return** 1; *// If a solution is found, return true*

}

*// If placing 'num' at (row, col) doesn't lead to a solution, backtrack*

grid[row][col] = 0;

}

}

**return** 0; *// Backtrack if no number can be placed at the current location*

}

*// Print the Sudoku grid*

void printGrid(int grid[N][N]) {

**for** (int i = 0; i < N; i++) {

**for** (int j = 0; j < N; j++) {

printf("%2d ", grid[i][j]);

}

printf("**\n**");

}

}

This program uses a backtracking algorithm to fill in the Sudoku grid. If a solution exists, it will print the solved grid; otherwise, it will indicate that no solution exists.

**Program Explanation :**

Let's go through the C code step by step to understand how the Sudoku solver works :

1. **Include Header Files :**

#include <stdio.h>

This line includes the standard input-output library, which is necessary for input/output operations.

1. **Define Constants :**

#define N 9

The constant **N** is defined to represent the size of the Sudoku grid. Since a standard Sudoku grid is 9x9, **N** is set to 9.

1. **Function Declarations :**

int isSafe(int grid[N][N], int row, int col, int num);

int findUnassignedLocation(int grid[N][N], int \*row, int \*col);

int solveSudoku(int grid[N][N]);

void printGrid(int grid[N][N]);

These lines declare the functions used in the program. The functions are responsible for checking if a number can be safely placed in a given position, finding an unassigned location in the Sudoku grid, solving the Sudoku puzzle using a backtracking algorithm, and printing the Sudoku grid.

1. **Main Function :**

int main() {

int grid[N][N];

The **main** function initializes a 9x9 array called **grid** to store the Sudoku puzzle.

printf("Enter the Sudoku grid (row by row, use 0 for empty cells):**\n**");

**for** (int i = 0; i < N; i++) {

**for** (int j = 0; j < N; j++) {

scanf("%d", &grid[i][j]);

}

}

The user is prompted to enter the Sudoku grid row by row. Each element of the grid is scanned using **scanf**. The user can use '0' to represent empty cells.

**if** (solveSudoku(grid)) {

printf("**\n**Solution:**\n**");

printGrid(grid);

} **else** {

printf("**\n**No solution exists.**\n**");

}

The **solveSudoku** function is called, and based on its return value, the program either prints the solved Sudoku grid or indicates that no solution exists.

1. **Sudoku Solving Functions :**

int isSafe(int grid[N][N], int row, int col, int num);

This function checks whether it is safe to place the number **num** at the specified position **(row, col)** in the Sudoku grid. It checks if **num** is not present in the current row, column, and 3x3 subgrid.

int findUnassignedLocation(int grid[N][N], int \*row, int \*col);

This function finds an unassigned location in the Sudoku grid and updates the values of **row** and **col** accordingly. It returns 1 if an unassigned location is found and 0 otherwise.

int solveSudoku(int grid[N][N]);

This is the main backtracking function that attempts to solve the Sudoku puzzle. It uses recursion to fill in the grid with numbers from 1 to 9 and backtracks if a conflict is encountered. It returns 1 if a solution is found and 0 otherwise.

void printGrid(int grid[N][N]);

This function is responsible for printing the Sudoku grid.

1. **Utility Functions :**

void printGrid(int grid[N][N]) {

**for** (int i = 0; i < N; i++) {

**for** (int j = 0; j < N; j++) {

printf("%2d ", grid[i][j]);

}

printf("**\n**");

}

}

The **printGrid** function prints the Sudoku grid in a readable format.

1. **Compile and Run :**

Compile the code using a C compiler, and run the executable. Enter the Sudoku grid as prompted, and the program will attempt to solve it, displaying the solution or indicating that no solution exists.

This program demonstrates a basic implementation of a Sudoku solver using a backtracking algorithm in the C programming language.

Top of Form