Read me

About the project

Sudoku is a logic-based, combinatorial number-placement puzzle. In classic Sudoku, the objective is to fill a 9×9 grid with digits so that each column, each row, and each of the nine 3×3 sub grids that compose the grid (also called "boxes", "blocks", or "regions") contain all of the digits from 1 to 9. The puzzle setter provides a partially completed grid, which for a well-posed puzzle has a single solution.

The project i.e. Sudoku solver hence is a simple menu driven program capable of generating correct solution of a 9 by 9 classic Sudoku. In case the entered Sudoku is invalid or if there is no possible solution to the problem then appropriate message is generated by the program itself

Technology used

The whole code is written in C++ and the concept of backtracking is used to solve the problem/Sudoku.

Backtracking can be defined as a general algorithmic technique that considers searching every possible combination in order to solve a computational problem. And thus instead of generating every possible way to solve the Sudoku, we use backtracking.

Apart from this the whole program is coded using simple C++ concepts like 2d array, pointers, functions etc.

How to run the program

In order to run the program

First make sure you have the code and an IDE or gcc compiler for running the code.

Build and run the program and a menu option will appear prompting to enter any one option from

1) Use the default Sudoku

2) Enter your own 9 by 9 Sudoku

3) Exit

Option 1

The program is having a default unsolved Sudoku and if the user want, they can check the working of code by entering '1', this option is just for checking the working of code and giving the idea that how it will work.

Option 2

If you want to enter your problem/unsolved Sudoku you can press '2' and can then proceed to entering your unsolved Sudoku row by row.

Option 3

It will end the program.

Once chosen any option, the program will attempt to solve the problem and if no solution is there a message will be displayed stating ‘No solution’.

What all function(s) are used

1) Main()

int main()

The main function is the driver code. And the menu driven code is inside this function only, from here the user will choose some option and the flow of the program call the functions accordingly.

Return type: int

2) Solve_it

bool Solve_it(int mat[N][N])

This function is the core function of this program as it holds the code for solving the problem. The function first looks for any empty or say unassigned place in the Sudoku with the help of Find_empty function. further it try to fill that empty or unassigned place by hit and trial method, thus it will start with number '1' and will check if it is valid according to rules of Sudoku with help of isSafe() function , which then ensures that the number is valid by calling usedinbox(), usedinrow(), usedincol().

If everything goes right then the number will be stored in that empty location. Else next number i.e. '2' will be used and it will go on till no '9' till a valid number is found.

Further a recursive call is made and the process goes on and on.

return type: bool

3) Find_empty

bool Find_empty(int mat[N][N], int& row, int& col)

In order to solve the souk or puzzle, program to need to search for empty places so

This function looks for empty position in the Sudoku represented by ‘0’, and returns a bool value accordingly

For achieving this it use for loop.

return type bool

4) isSafe

bool isSafe(int mat[N][N], int row,int col, int num)

During the hit and trial approach this function will help the program to ensure the validity of that number at that particular empty location. usedinbox(), usedinrow() and usedincol() will be used to ensure that the chosen number is not in the 3 by 3 box, in that particular row and in that particular column respectively.

return type: bool

5) UsedInBox

bool UsedInBox(int mat[N][N], int boxStartRow,int boxStartCol, int num)

This function looks if the number is not occurring in the box i.e. in that 3*3 grid where it will be placed as according to rules of Sudoku one can only use a number out of 1 to 9 only once in this 3*3 grid.

return type: bool

6) UsedInCol

bool UsedInCol(int mat[N][N], int col, int num)

This function will ensure that the number(int num) is unique in the column where it is supposed to be filled. It will return a bool value according to the outcome.

return type: bool

7) UsedInRow

bool UsedInRow(int mat[N][N], int row, int num)

Similar to previous one, This function will ensure that the number(int num) is unique in the column where it is supposed to be filled. It will return a bool value according to the outcome.

return type: bool

8) print_it

void print_it(int mat[N][N])

This function will simply print the solved Sudoku puzzle on the output screen.

return type void