PROJECT2: Sudoku program using C++

A simple Sudoku C++ program that takes a matrix of  9×9 grid with digits so that each column, each row, and each of the nine 3×3 subgrids 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.

**TECHNOLOGY USED:**

Programming Language : C++

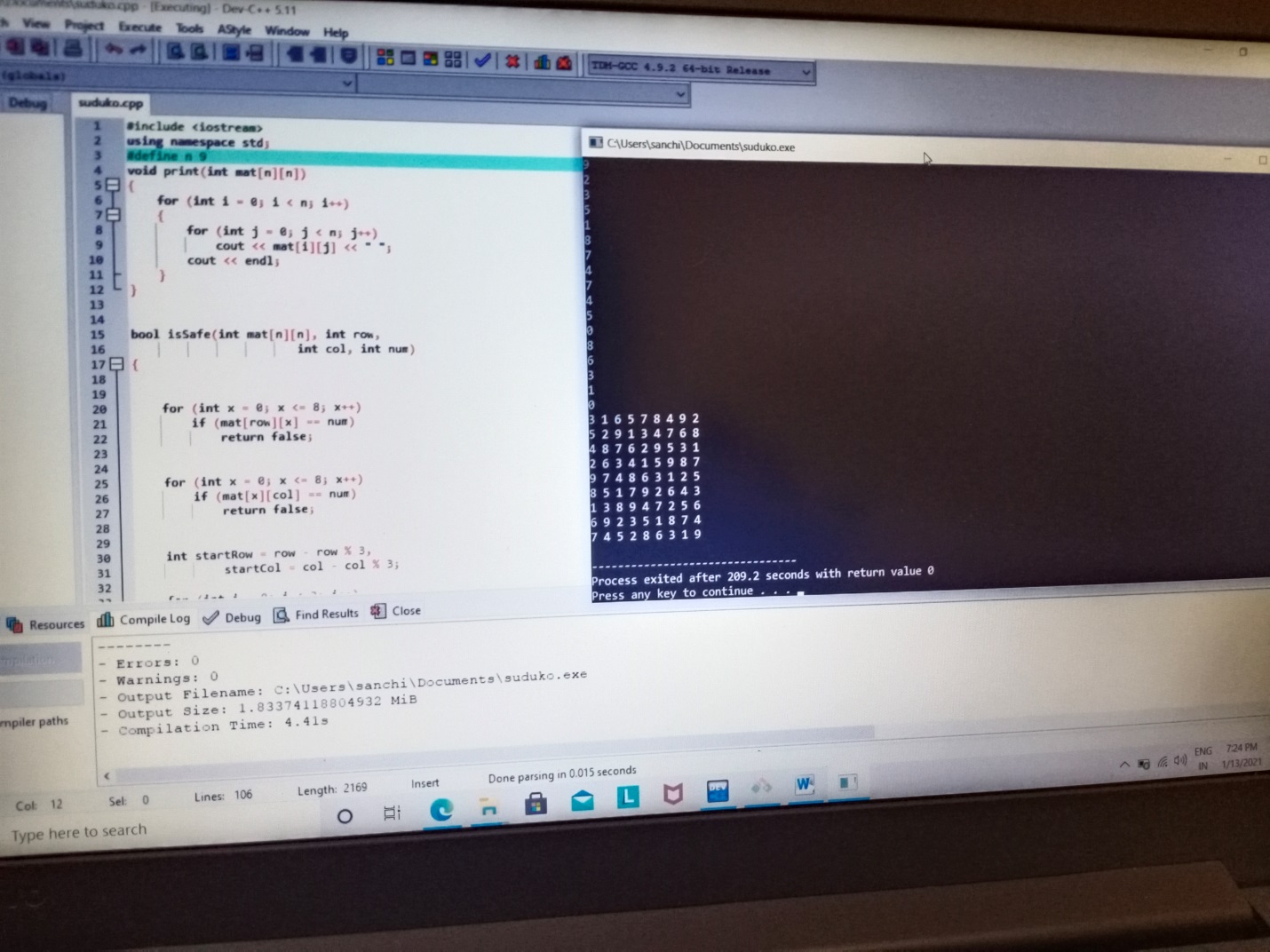
Soft-wares: DEV-C++

Platforms:

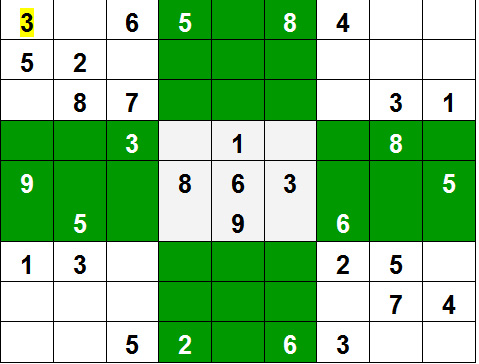
[Download DEV-C++ 5.11.492 for Windows - Filehippo.com](https://filehippo.com/download_dev-c/)

**SEPTS TO RUN THE PROGRAM:**

1. Open file Sudoku.cpp input the sudoku puzzle with "." (don't include quotations) for blank spaces to be solved and a newline every 9 characters (don't add spaces), once entered save this file
2. Run the program by clicking Sudoku.exe



3. Read the output on the Command promt or read the log file: Sudoku.cpp for the results



**This is a C++ program to run a SUDOKU ,in this we use three functions:**

**1.for printing**

**2. for checking**

**3. for sloving**

**HOW PROGRAM WORKS::**

1. Create a function that checks if the given matrix is valid sudoku or not. Keep Hashmap for the row, column and boxes. If any number has a frequency greater than 1 in the hashMap return false else return true;
2. Create a recursive function that takes a grid and the current row and column index.
3. Check some base cases. If the index is at the end of the matrix, i.e. i=N-1 and j=N then check if the grid is safe or not, if safe print the grid and return true else return false. The other base case is when the value of column is N, i.e j = N, then move to next row, i.e. i++ and j = 0.
4. if the current index is not assigned then fill the element from 1 to 9 and recur for all 9 cases with the index of next element, i.e. i, j+1. if the recursive call returns true then break the loop and return true.
5. if the current index is assigned then call the recursive function with index of next element, i.e. i, j+1

**SAMPLE INPUT::**

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

**SAMPLE OUTPUT::**

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

Source:-official website of geeksforgeeks