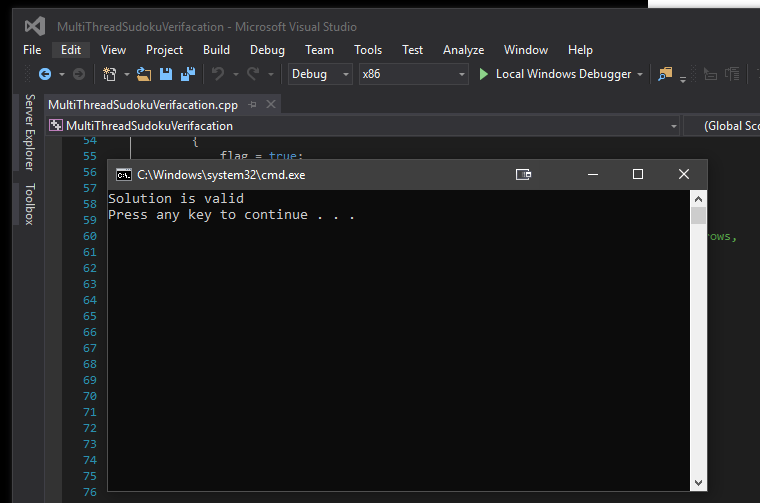
**Output:**



**Source code:**

// SudokuVerifyMultiThread.cpp : Defines the entry point for the console application.

//

#include "stdafx.h"

#include <iostream>

#include <thread>

using namespace std;

//Initialize functions

bool checkDigit(int a[][9], int rowStart, int rowEnd, int columnStart, int columnEnd);

void checkRows(bool a[], int b[][9]);

void checkColumns(bool a[], int b[][9]);

void checkBlock(bool a[], int b[][9], int i);

int main()

{

//Initialize threads for functions, 1 for rows, 1 for columns and 9 for subblocks of sudoku array

thread t[11];

//Array to share values of functions to be used by all threads

bool valid[27];

int sudoku[9][9] = {

{ 6, 2, 4, 5, 3, 9, 1, 8, 7 },

{ 5, 1, 9, 7, 2, 8, 6, 3, 4 },

{ 8, 3, 7, 6, 1, 4, 2, 9, 5 },

{ 1, 4, 3, 8, 6, 5, 7, 2, 9 },

{ 9, 5, 8, 2, 4, 7, 3, 6, 1 },

{ 7, 6, 2, 3, 9, 1, 4, 5, 8 },

{ 3, 7, 1, 9, 5, 6, 8, 4, 2 },

{ 4, 9, 6, 1, 8, 2, 5, 7, 3 },

{ 2, 8, 5, 4, 7, 3, 9, 1, 6 }

};

//Thread for row verifier function

t[0] = thread(checkRows, valid, sudoku);

//Thread for column verifier function

t[1] = thread(checkColumns, valid, sudoku);

//Loop to create 9 threads for each subblock to run subblock verifier function

//Begin counter at 2 to initialize each thread subsequent to previous threads

for (int i = 2; i < 11; i++)

{

t[i] = thread(checkBlock, valid, sudoku, i - 2);

}

//Create flag to monitor value of verification results from the funtions and flag

//any false value returned from the functions

bool flag = false;

for (int i = 0; i < 27; i++)

{

if (valid[i] == false)

{

flag = true;

}

}

//If flag is true this means a function found a repeated digit in any of the rows,

//columns or subblocks

//Output validity of solution based on flag result

if (flag == true)

cout << "Solution is invalid\n";

else

cout << "Solution is valid\n";

//Join all threads before terminating program

for (int i = 0; i < 11; i++)

{

t[i].join();

}

return 0;

}

void checkRows(bool a[], int b[][9])

{//Function to check rows using the check digit function

for (int i = 0; i < 9; i++)

{

a[i] = checkDigit(b, i, i + 1, 0, 9);

}

}

void checkColumns(bool a[], int b[][9])

{//Function to check columns using the check digit function

for (int i = 0; i < 9; i++)

{

a[9 + i] = checkDigit(b, 0, 9, i, i + 1);

}

}

void checkBlock(bool a[], int b[][9], int i)

{//Function to check subblocks using the check digit function

for (int row = 0; row < 9; row = row + 3)

for (int column = 0; column < 9; column = column + 3)

a[18 + i] = checkDigit(b, row, row + 3, column, column + 3);

}

bool checkDigit(int a[][9], int rowStart, int rowEnd, int columnStart, int columnEnd)

{//Function to check rows, columns or subblocks for non repeated digits from 1 to 9

int count[9];

for (int i = 0; i < 9; i++) count[i] = 0;

for (int row = rowStart; row < rowEnd; row++)

for (int column = columnStart; column < columnEnd; column++)

{

count[a[row][column] - 1]++;

}

for (int i = 0; i < 9; i++) if (count[i] != 1) return false;

return true;

}