DEPARTMENT OF COMPUTER & SOFTWARE ENGINEERING

COLLEGE OF E&ME, NUST, RAWALPINDI

Fundamental of Programming

Semester Final Project
"SUDOKU GAME"

SUBMITTED TO:

Asst. Prof Jahan Zeb / LE Syed Abubakar

SUBMITTED BY:

Student Name:	**Registration number:**
Talha Razzaq	418013
Wahab Sohail	418159
Muhammad Sumair	415339

DE- 44 **Dept.** CE **Syn.** 'A'

SUBMISSION DATE:

31 May 2023

Project Description:

The Sudoku Game project is a digital implementation of the popular Sudoku puzzle game. Sudoku is a logic-based number placement puzzle that challenges players to fill a 9x9 grid with digits so that each column, each row, and each of the nine 3x3 sub grids contain all of the digits from 1 to 9.

The project aims to provide a user-friendly interface where players can solve Sudoku puzzles of varying difficulty levels. The game will generate puzzles with different numbers of pre-filled cells to cater to players of all skill levels. It will also include a hint system to assist players when they get stuck, providing suggestions for the next logical move.

The expected outcome of the project is a fully functional Sudoku game with a visually appealing interface and intuitive controls.

In conclusion, the Sudoku Game project aims to deliver an engaging and enjoyable digital version of the Sudoku puzzle game, offering a challenging and entertaining experience for players of all ages.

Console Window Display

Start Game Option:

Upon launching the Sudoku Game, users will be presented with a console interface that displays several options. The "Start Game" option allows players to begin playing Sudoku. Once selected, the game will initialize and present the Sudoku grid to the player.

Difficulty Level Selection:

The Sudoku Game offers players the ability to choose their preferred difficulty level. After selecting the "Start Game" option, the console will present choices such as "Easy," "Medium," and "Hard" difficulty levels. Players can select the desired level by inputting the corresponding option. The chosen difficulty level will determine the number of pre-filled cells in the Sudoku grid, offering a suitable challenge for the player.

Help Option:

The Sudoku Game includes a "Help" option that provides instructions and guidance to players. Upon selecting the "Help" option from the console, the game will display detailed instructions on how to play Sudoku. This feature aims to assist players in understanding the rules and strategies of the game, enabling them to make informed moves.

Exit Option:

The Sudoku Game allows players to gracefully exit the game at any time. By choosing the "Exit" option from the console, the game will terminate, concluding the gameplay session. This feature ensures a convenient and user-friendly experience for players.

Purpose of included Libraries:

#include <SFML/Graphics.hpp>: Includes the SFML library for graphics and window management.

<iostream>: Provides input/output stream functionality.

<cstdlib> and <ctime>: Used for functions related to random number generation.

<iomanip>: Used for input/output manipulations.

<vector>: Provides the vector container class.

#define statements: Define color codes for output formatting.

Conditional includes and definitions: Provide platform-specific definitions for Windows and Unix-like systems.

Working of Functions:

clearScreen():

- This function clears the console screen.
- It uses system-specific commands (cls for Windows and clear for Linux/Unix) to clear the screen.

showHelp():

- This function displays the help information for the Sudoku game.
- It prints various sections explaining the objective, setup, rules, controls, strategies, and tips of the game.
- After displaying the help information, it waits for the user to press any key before returning to the main menu.

chooseDifficultyLevel():

- This function allows the user to choose the difficulty level for the game.
- It prompts the user to enter a number representing the difficulty level (1 for easy, 2 for medium, 3 for hard).
- It validates the user's input and repeats the prompt until a valid input is provided.
- After receiving a valid input, it waits for the user to press any key before returning to the main menu.
- Finally, it returns the chosen difficulty level.
-

validity_check():

- This function checks the validity of placing a number in a Sudoku board.
- It takes the Sudoku board, the row index, column index, and the number to be checked as parameters.
- It verifies if the number is valid in the given row, column, and 3x3 box of the board according to Sudoku rules.
- Returns true if the number is valid, and false otherwise.

solveSudoku():

- This function recursively solves the Sudoku board using backtracking.
- It takes the Sudoku board, the current row, and the current column as parameters.
- It uses a recursive approach to fill the board cell by cell.
- Returns true if a solution is found, and false otherwise.

isUniqueSolution():

- This function checks if a Sudoku board has a unique solution.
- It creates a copy of the original board and attempts to solve it.
- If the copied board has a unique solution, it returns true; otherwise, it returns false.
-

generate_puzzle():

- This function generates a Sudoku puzzle by solving a complete board and removing a specified number of cells based on the chosen difficulty level.
- It takes the Sudoku board and the chosen difficulty level as parameters.
- It solves the board using solveSudoku() function and then removes cells randomly while ensuring the puzzle still has a unique solution.
- Finally, it checks if the final puzzle has a unique solution and returns true if it does, and false otherwise.

playGame():

- This function handles the main gameplay loop of the Sudoku game.
- It takes the SFML Render Window object and the chosen difficulty level as parameters.
- It initializes the Sudoku board and sets up various SFML graphical elements for rendering the game.
- It handles user input events such as mouse clicks and keyboard input for placing numbers in the Sudoku grid.
- It updates the graphical display of the Sudoku board based on the user's input and the game state.
- It checks if the Sudoku board is complete and displays a completion message if all cells are filled correctly.
- This function is responsible for rendering the graphical elements on the SFML window.

Further Explanation of SFML codes

void playGame(sf::RenderWindow& window, int difficulty) : This line defines a function named playGame that takes a reference to an sf::RenderWindow object and an integer difficulty as parameters and returns nothing (void).

int sudoku_board[GRID_SIZE][GRID_SIZE] = { 0 };: This line declares a two-dimensional array named sudoku_board with dimensions GRID_SIZE by GRID_SIZE and initializes all its elements to zero.

generate_puzzle(sudoku_board, difficulty);: This line calls a function named generate_puzzle and passes the sudoku_board array and the difficulty variable as arguments.

sf::Font font;: This line declares an sf::Font object named font.

if (!font.loadFromFile("digitalism.ttf")) {: This line attempts to load a font from a file named "digitalism.ttf". If the font loading fails, the code within the if block is executed.

std::cerr << "Error loading font file." << std::endl;: This line outputs an error message to the standard error stream (std::cerr).

return;: This line exits the playGame function early if there was an error loading the font file.

sf::Text text("", font, 32);: This line declares an sf::Text object named text with an empty string, the font as its font, and a character size of 32 pixels.

text.setFillColor(sf::Color::Black);: This line sets the fill color of the text object to black.

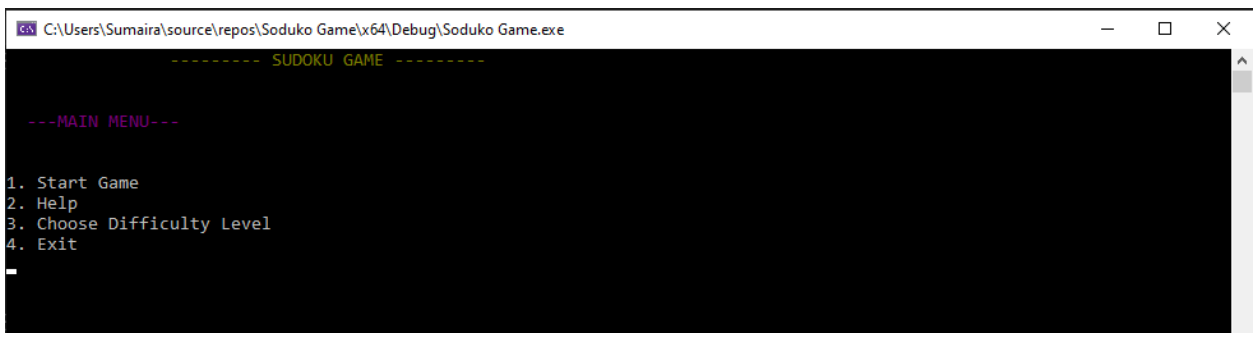
sf::RectangleShape selectedCell(sf::Vector2f(CELL_SIZE, CELL_SIZE));: This line declares an sf::RectangleShape object named selectedCell with dimensions CELL_SIZE by CELL_SIZE.

selectedCell.setFillColor(sf::Color(255, 255, 0, 128));: This line sets the fill color of the selectedCell object to a yellow color with some transparency.

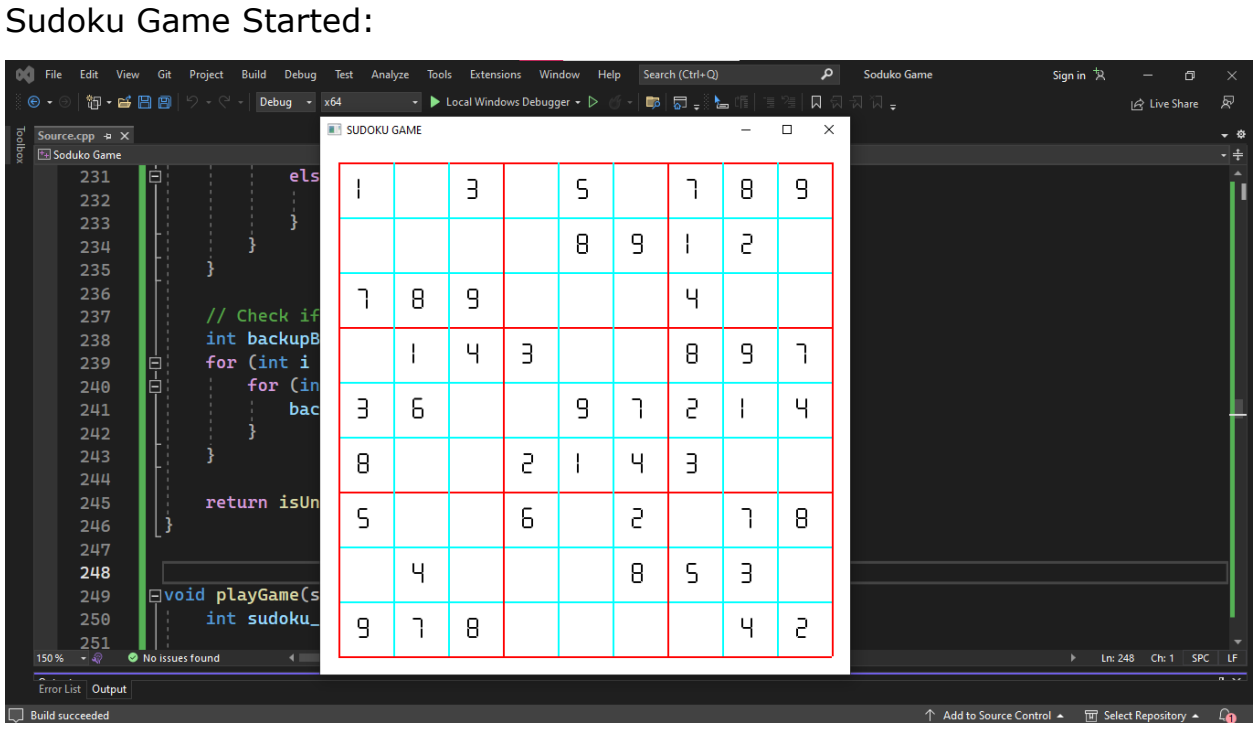
while (window.isOpen()) : This line starts a while loop that continues as long as the window is open.

Images of Executed Program

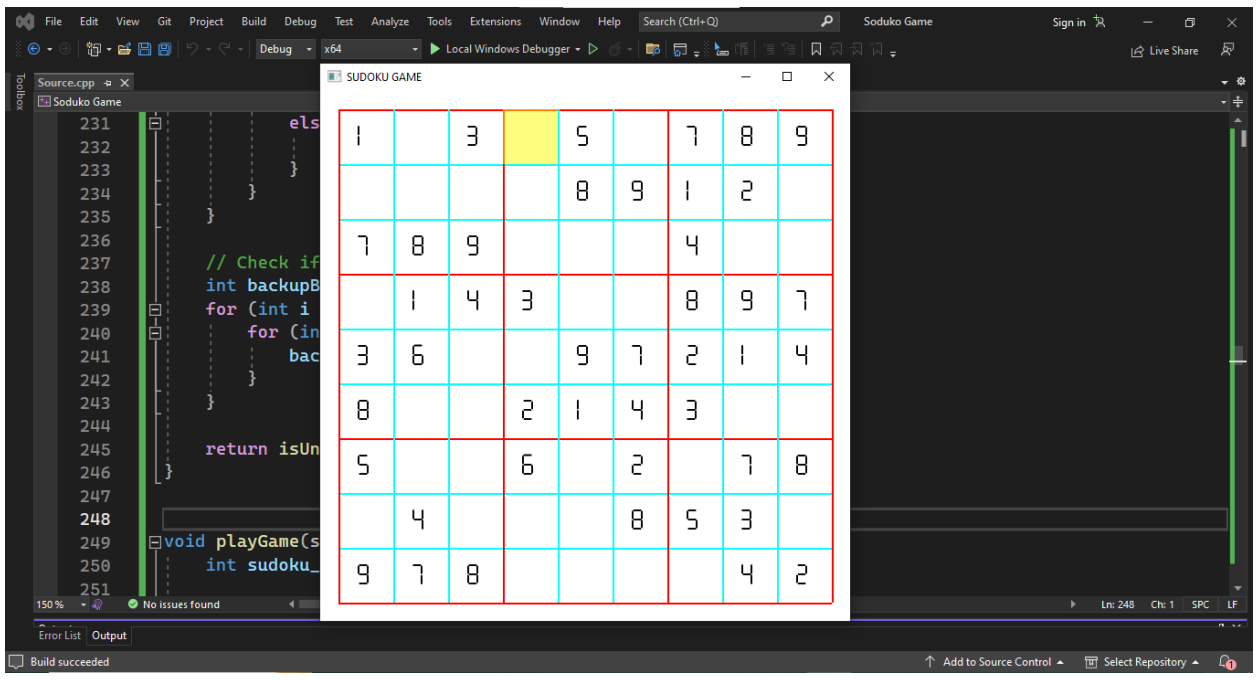
Main Menu:



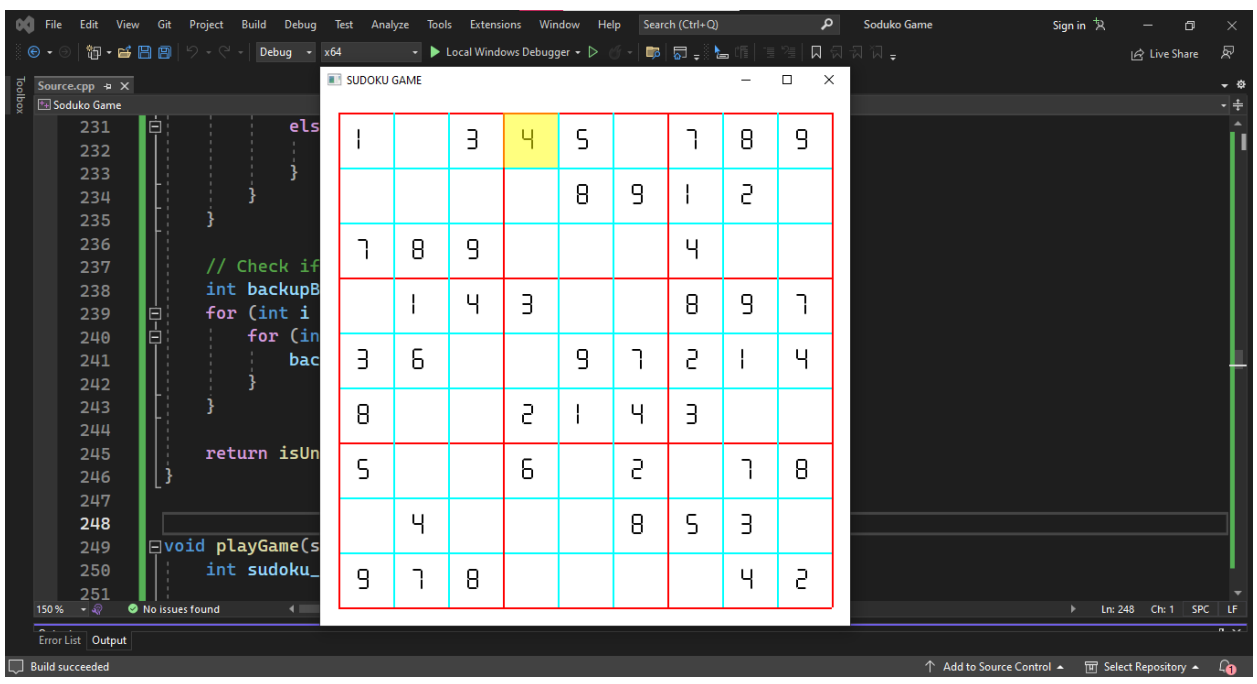
Sudoku Game Started:



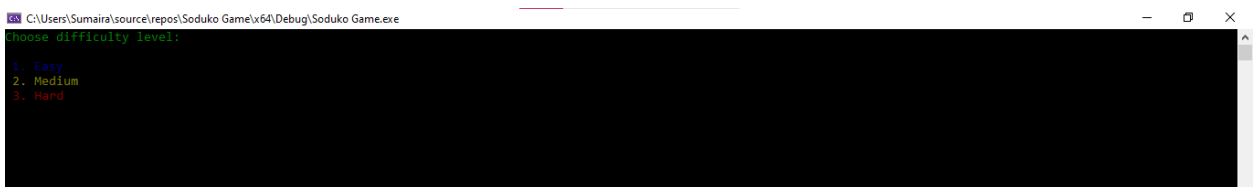
Selected box color turned yellow:



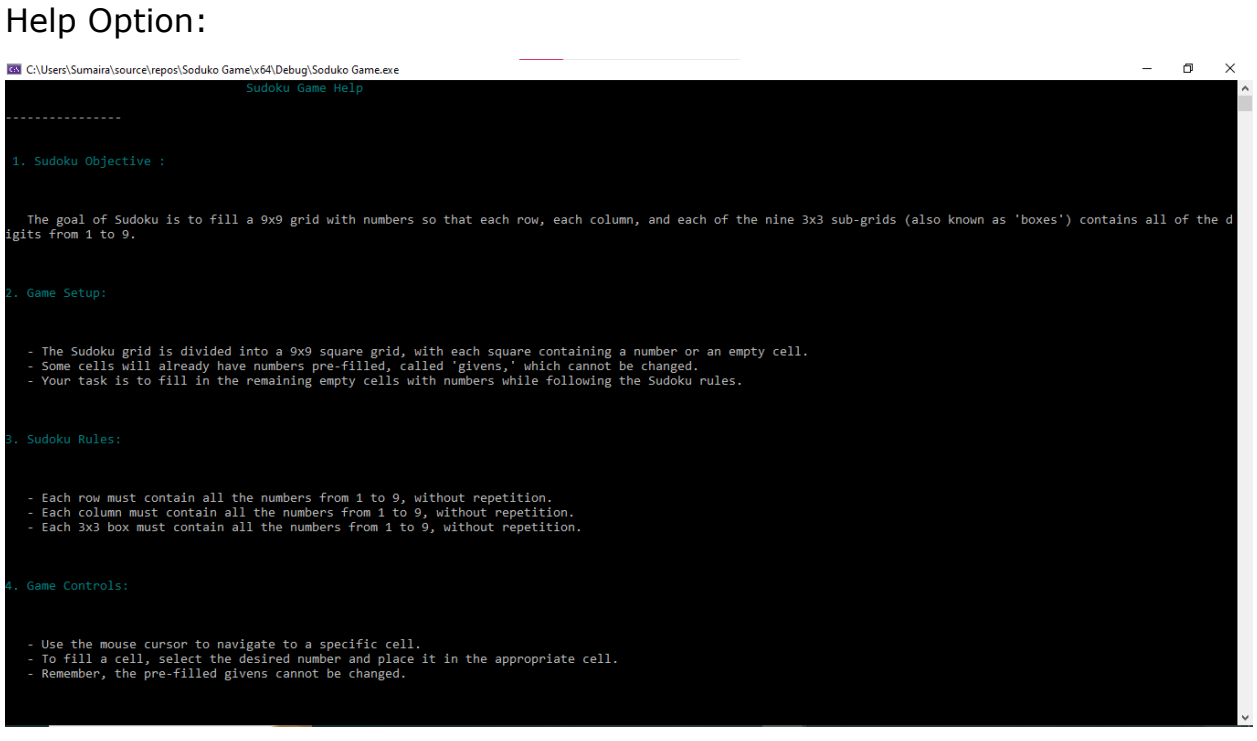
Enter number to the selected box:



Difficulty level Option:



Help Option:



Conclusion

In conclusion, the Sudoku game project successfully implements the rules and functionalities of the game, showcasing the effective use of C++ programming concepts. It features a user-friendly interface, random grid generation and validation system. The code follows best practices and algorithms. The project incorporates various solving techniques, error handling, and provides an engaging experience for Sudoku enthusiasts. Overall, it demonstrates strong programming skills and serves as a valuable resource for learning C++ and game development.