

2048 Game In C

Dlithe project report | CP011

Shawn Saldanha | Samarth Shetty | Mohammed Salah | Keerthan Suvarna

4MT21AI050 4MT21AI051 4MT21AI030 4MT21AI026

REPORT

# Abstract:

This project presents the implementation of the classic 2048 game in the C programming language. The game is a single-player sliding block puzzle where the player's objective is to merge numbered tiles on a grid to form a tile with the number 2048. The project includes features such as player name input, scoring, and the ability to continue playing to achieve larger scores. This report details the development of the game, including its architecture, design, and functionality.

# Introduction:

## 2.1 Background:

The 2048 game is a well-known and popular puzzle game that challenges players to combine tiles strategically to reach the elusive 2048 tile. This project aims to recreate this game in C, offering an engaging and entertaining experience for players while demonstrating C programming skills

## 2.2 Objectives:

The primary objectives of this project include:

- Developing a playable version of the 2048 game in C.

- Providing a user-friendly interface for player interaction.

- Implementing game logic for tile movement, merging, and scoring.

- Allowing players to input their names and track their scores.

- Supporting multiple rounds of play with a play-again option.

- Demonstrating proficiency in C programming, including handling user input and randomization.

# 3. Technologies Used:

The technologies used in this project include:

- C programming language

- Standard C libraries: `stdio.h`, `stdlib.h`, `conio.h`, `time.h`

# 4. System Architecture:

## 4.1 Front-End:

The front-end of the game is text-based and relies on the console for input and output. It presents the game grid, player information, and score to the user.

## 4.2 Back-End:

The back-end is responsible for game logic, including initializing the grid, generating random tiles, handling player input, and checking for game over conditions.

## 4.3 Database:

No external database is used in this project as the game state is stored in a 2D array within the program.

# 5. Project Modules:

The project consists of the following major modules:

- Initialization and setup

- User input and interaction

- Game logic and tile movement

- Score tracking and display

- Game over detection and handling

# 6. Design and Implementation:

## 6.1 Front-End Design:

The front-end design is minimalistic, using a console-based interface to display the grid and game information. It relies on simple text characters for visual representation.

## 6.2 Back-End Design:

The back-end is responsible for the core game logic, including initializing the grid, generating random tiles, handling user input for tile movement, and scoring. It uses a 2D array to represent the game grid.

## 6.3 Database Design:

No external database is used in this project. Game state information is stored in memory within the program.

# 7. Features and Functionality:

The game features:

- Player name input

- Scoring and score tracking

- Random tile generation

- Tile movement in four directions (up, left, down, right)

- Tile merging and scoring logic

- Game over detection

- Option to play multiple rounds

# 8. Testing:

The project underwent testing to ensure its functionality and reliability. Testing methods included unit testing for individual functions and user acceptance testing to confirm that players can interact with the game effectively.

# 9. Challenges Faced:

During development, several challenges were encountered, including managing user input, implementing the game logic for tile merging, and handling random tile generation. These challenges were addressed through careful debugging and testing.

# 10. Future Enhancements:

Possible future enhancements for the project include:

- Adding a graphical user interface (GUI) for a more visually appealing game.

- Implementing a leaderboard to track high scores.

- Allowing players to customize the grid size.

# 11. Conclusion:

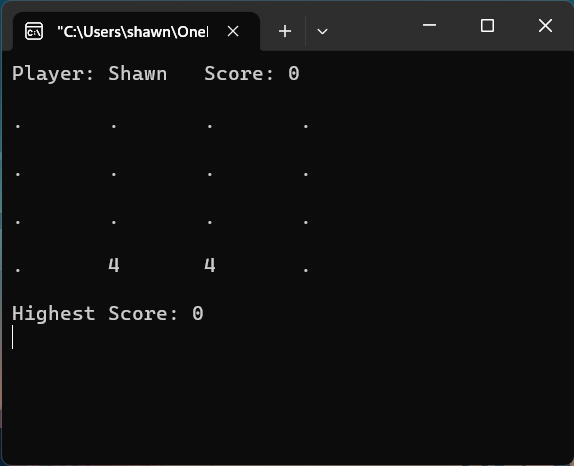
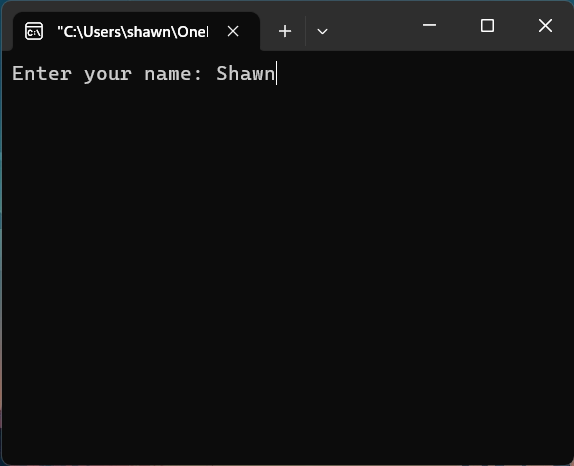
In conclusion, this project successfully implemented the 2048 game in the C programming language. It demonstrated proficiency in C programming, game development, and user interaction. The game provides an engaging experience for players and serves as a testament to the developer's programming skills.

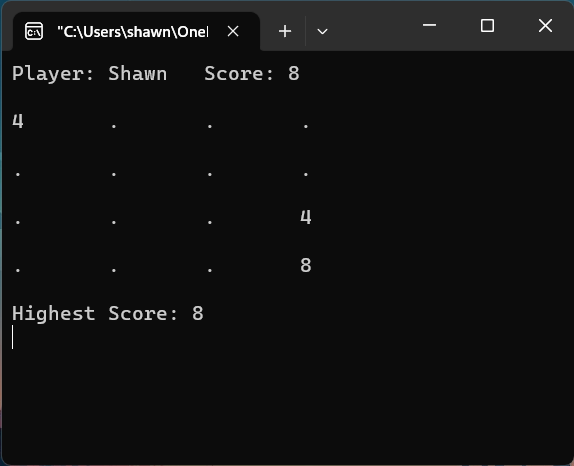
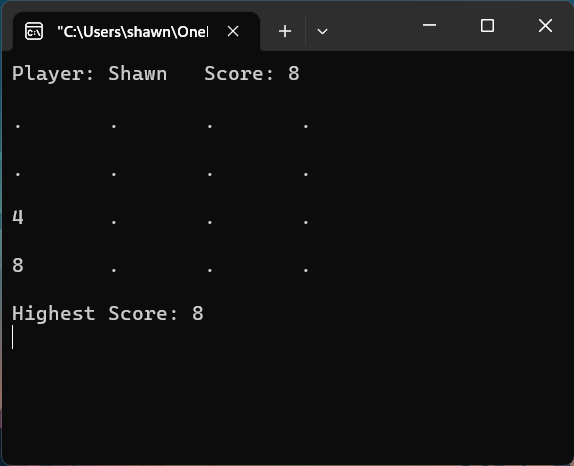
# 12. References:

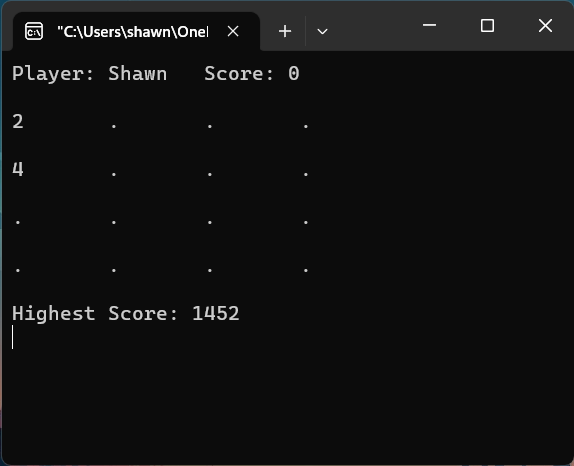
# <https://github.com/gabrielecirulli/2048>

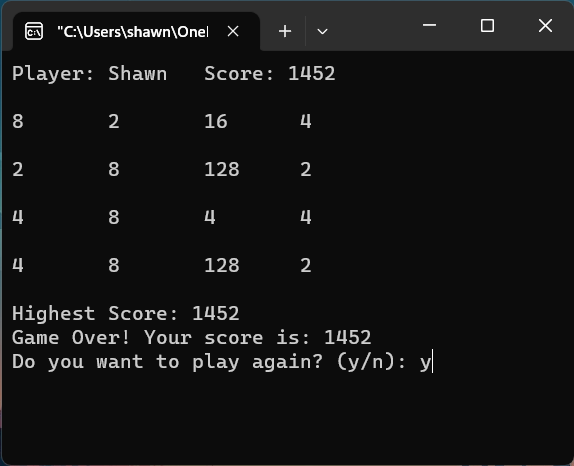
# 13. Appendices:

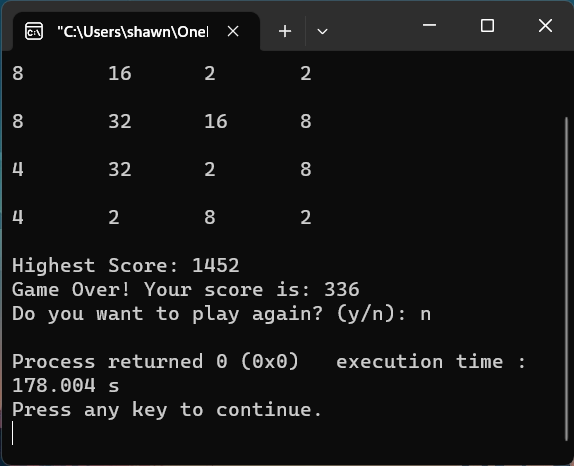
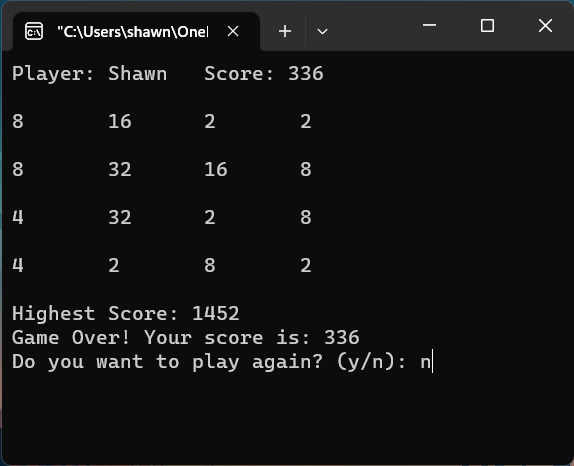
## 13.1 Screenshots:

1) 2)

3)  4)



5) 6)

7)  8)

## 13.2 Code Snippets:

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

#include <time.h>

#define SIZE 4

int grid**[**SIZE**][**SIZE**];**

char playerName**[**50**];**

int score **=** 0**;**

void initializeGrid**()** **{**

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

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

grid**[**i**][**j**]** **=** 0**;**

**}**

**}**

**}**

void printGrid**()** **{**

system**(**"cls"**);**

printf**(**"Player: %s\tScore: %d\n\n"**,** playerName**,** score**);**

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

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

**if** **(**grid**[**i**][**j**]** **==** 0**)** **{**

printf**(**".\t"**);**

**}** **else** **{**

printf**(**"%d\t"**,** grid**[**i**][**j**]);**

**}**

**}**

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

**}**

**}**

void placeRandomTile**()** **{**

int emptyCells**[**SIZE **\*** SIZE**][**2**];**

int numEmptyCells **=** 0**;**

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

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

**if** **(**grid**[**i**][**j**]** **==** 0**)** **{**

emptyCells**[**numEmptyCells**][**0**]** **=** i**;**

emptyCells**[**numEmptyCells**][**1**]** **=** j**;**

numEmptyCells**++;**

**}**

**}**

**}**

**if** **(**numEmptyCells **>** 0**)** **{**

int index **=** rand**()** **%** numEmptyCells**;**

int value **=** **(**rand**()** **%** 2 **+** 1**)** **\*** 2**;**

int row **=** emptyCells**[**index**][**0**];**

int col **=** emptyCells**[**index**][**1**];**

grid**[**row**][**col**]** **=** value**;**

**}**

**}**

void moveTiles**(**char direction**)** **{**

int moved **=** 0**;**

**switch** **(**direction**)** **{**

**case** 'w'**:**

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

**for** **(**int i **=** 1**;** i **<** SIZE**;** i**++)** **{**

**if** **(**grid**[**i**][**j**]** **!=** 0**)** **{**

int row **=** i**;**

**while** **(**row **>** 0 **&&** grid**[**row **-** 1**][**j**]** **==** 0**)** **{**

grid**[**row **-** 1**][**j**]** **=** grid**[**row**][**j**];**

grid**[**row**][**j**]** **=** 0**;**

row**--;**

moved **=** 1**;**

**}**

**if** **(**row **>** 0 **&&** grid**[**row **-** 1**][**j**]** **==** grid**[**row**][**j**])** **{**

grid**[**row **-** 1**][**j**]** **\*=** 2**;**

score **+=** grid**[**row **-** 1**][**j**];**

grid**[**row**][**j**]** **=** 0**;**

moved **=** 1**;**

**}**

**}**

**}**

**}**

**break;**

**case** 'a'**:**

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

**for** **(**int j **=** 1**;** j **<** SIZE**;** j**++)** **{**

**if** **(**grid**[**i**][**j**]** **!=** 0**)** **{**

int col **=** j**;**

**while** **(**col **>** 0 **&&** grid**[**i**][**col **-** 1**]** **==** 0**)** **{**

grid**[**i**][**col **-** 1**]** **=** grid**[**i**][**col**];**

grid**[**i**][**col**]** **=** 0**;**

col**--;**

moved **=** 1**;**

**}**

**if** **(**col **>** 0 **&&** grid**[**i**][**col **-** 1**]** **==** grid**[**i**][**col**])** **{**

grid**[**i**][**col **-** 1**]** **\*=** 2**;**

score **+=** grid**[**i**][**col **-** 1**];**

grid**[**i**][**col**]** **=** 0**;**

moved **=** 1**;**

**}**

**}**

**}**

**}**

**break;**

**case** 's'**:**

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

**for** **(**int i **=** SIZE **-** 2**;** i **>=** 0**;** i**--)** **{**

**if** **(**grid**[**i**][**j**]** **!=** 0**)** **{**

int row **=** i**;**

**while** **(**row **<** SIZE **-** 1 **&&** grid**[**row **+** 1**][**j**]** **==** 0**)** **{**

grid**[**row **+** 1**][**j**]** **=** grid**[**row**][**j**];**

grid**[**row**][**j**]** **=** 0**;**

row**++;**

moved **=** 1**;**

**}**

**if** **(**row **<** SIZE **-** 1 **&&** grid**[**row **+** 1**][**j**]** **==** grid**[**row**][**j**])** **{**

grid**[**row **+** 1**][**j**]** **\*=** 2**;**

score **+=** grid**[**row **+** 1**][**j**];**

grid**[**row**][**j**]** **=** 0**;**

moved **=** 1**;**

**}**

**}**

**}**

**}**

**break;**

**case** 'd'**:**

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

**for** **(**int j **=** SIZE **-** 2**;** j **>=** 0**;** j**--)** **{**

**if** **(**grid**[**i**][**j**]** **!=** 0**)** **{**

int col **=** j**;**

**while** **(**col **<** SIZE **-** 1 **&&** grid**[**i**][**col **+** 1**]** **==** 0**)** **{**

grid**[**i**][**col **+** 1**]** **=** grid**[**i**][**col**];**

grid**[**i**][**col**]** **=** 0**;**

col**++;**

moved **=** 1**;**

**}**

**if** **(**col **<** SIZE **-** 1 **&&** grid**[**i**][**col **+** 1**]** **==** grid**[**i**][**col**])** **{**

grid**[**i**][**col **+** 1**]** **\*=** 2**;**

score **+=** grid**[**i**][**col **+** 1**];**

grid**[**i**][**col**]** **=** 0**;**

moved **=** 1**;**

**}**

**}**

**}**

**}**

**break;**

**default:**

**break;**

**}**

**if** **(**moved**)** **{**

placeRandomTile**();**

**}**

**}**

int isGameOver**()** **{**

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

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

**if** **(**grid**[**i**][**j**]** **==** 0**)** **{**

**return** 0**;**

**}**

**}**

**}**

**return** 1**;**

**}**

int main**()** **{**

srand**(**time**(**0**));**

int highestScore **=** 0**;**

char playAgain**;**

**do** **{**

printf**(**"Enter your name: "**);**

scanf**(**"%s"**,** playerName**);**

getchar**();**

score **=** 0**;**

initializeGrid**();**

placeRandomTile**();**

placeRandomTile**();**

printGrid**();**

printf**(**"Highest Score: %d\n"**,** highestScore**);**

char move**;**

**while** **(**1**)** **{**

move **=** getch**();**

**if** **(**move **==** 'q'**)** **{**

printf**(**"You quit the game. Your score is: %d\n"**,** score**);**

**break;**

**}**

moveTiles**(**move**);**

**if** **(**score **>** highestScore**)** **{**

highestScore **=** score**;**

**}**

printGrid**();**

printf**(**"Highest Score: %d\n"**,** highestScore**);**

**if** **(**isGameOver**())** **{**

printf**(**"Game Over! Your score is: %d\n"**,** score**);**

**break;**

**}**

**}**

printf**(**"Do you want to play again? (y/n): "**);**

scanf**(**" %c"**,** **&**playAgain**);**

getchar**();**

**}** **while** **(**playAgain **==** 'y' **||** playAgain **==** 'Y'**);**

**return** 0**;**

**}**