

Team Members : Submitted to:

S.Umais Mustafa k24\_0693 Ms.Khadija tul Kubra

Abdul Wasay k24\_0744 (Leader)

Ashar Adnan k24\_0617

Contents

[1. Introduction 4](#_Toc183516155)

[2. System Design 7](#_Toc183516156)

[3. Code Explanation 8](#_Toc183516157)

[4. Testing and Results 14](#_Toc183516158)

[15](#_Toc183516159)

[5.Pseudocode 15](#_Toc183516160)

[START GAME 15](#_Toc183516161)

[DISPLAY initial level and maze 15](#_Toc183516162)

[GET position of the Start(‘S’) and END(‘E’) 15](#_Toc183516163)

[SET player position at the START 15](#_Toc183516164)

[WHILE player position is not at the END : 15](#_Toc183516165)

[DISPLAY maze 15](#_Toc183516166)

[GET player input (W,A,S,D) 15](#_Toc183516167)

[If input is is invalid: 16](#_Toc183516168)

[PRINT “INVALID MOVE” CONTINUE to next input 16](#_Toc183516169)

[IF move is valid and not a wall:\ 16](#_Toc183516170)

[UPDATE player position in maze 16](#_Toc183516171)

[CHECK IF player has reached the END 16](#_Toc183516172)

[IF player reaches END: 16](#_Toc183516173)

[PRINT “Level Complete” 16](#_Toc183516174)

[MOVE to next level 16](#_Toc183516175)

[END GAME 16](#_Toc183516176)

[END GAme 16](#_Toc183516177)

[6. Conclusion 16](#_Toc183516178)

CS1002-Programming Fundamentals

# 1. Introduction

This section provides an overview of the Maze Game project, its purpose, and the technology used. It introduces the main gameplay mechanics and sets the context for the code implementation.

**1.1 Objective of the Project**

The objective of this project is to create a simple, text-based Maze Game where a player navigates through a maze from a start ('S') to an end ('E') point. The player can move up, down, left, or right using keyboard inputs. The goal is to reach the endpoint while avoiding obstacles (walls).

**1.2 Technology Used**

The game is written in **C** programming language using basic constructs like arrays, loops, conditionals, and functions. The program uses a simple **text-based interface** to display the maze and interact with the player.

***FLOWCHART***

DISPLAY LEVELS AND MAZE

GET PLAYER POSITION (S)

PLAYER MOVEMENT INPUT

VALIDATE MOVE(W,A,S,D)

A

A

A

YES

NO

IF VALID MOVE

NO

YES

REPEAT OR EXIT THE GAME

UPDATE MAZE WITH

NEW POSITION

INVALID MOVE ! TRY AGAIN

IS PLAYER AT THE END(E)?

CS1002-Programming Fundamentals

# 2. System Design

This section explains the core design and flow of the program, including how the maze is represented and how player interactions are managed.

#### ****2.1 Maze Representation****

The maze is represented as a 2D array of characters, where:

* ' ' represents a **path** the player can walk on.
* '#' represents a **wall** that the player cannot pass through.
* 'S' represents the **start** point of the maze.
* 'E' represents the **end** point of the maze.
* 'P' represents the **player's position**.

#### ****2.2 Game Flow****

The game begins by displaying the maze and positioning the player at the starting point. The player can move using the following controls:

* **W**: Move up.
* **S**: Move down.
* **A**: Move left.
* **D**: Move right.
* **1**: Exit the game.

The game continues until the player reaches the end point ('E') or chooses to quit.

CS1002-Programming Fundamentals

# 3. Code Explanation

In this section, we break down the key functions in the code and explain their purpose in detail.

**3.1 printMatrix()**

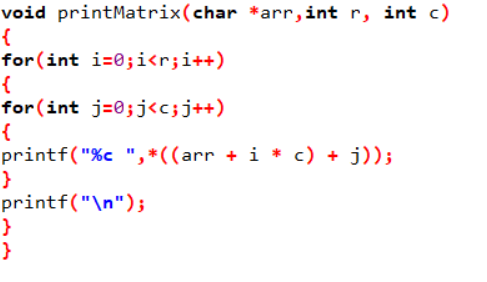
This function is responsible for displaying the maze in the console. It iterates through the maze array and prints each character in a grid-like format.

**Parameters:**

* arr: Pointer to the maze array.
* r: Number of rows in the maze.
* c: Number of columns in the maze.

**Description:**

* It loops through the array and prints each element in a formatted grid layout.



**3.2 getPositionOfCharacter()**

This function finds the row and column of a specific character in the maze (such as 'S', 'E', or 'P').

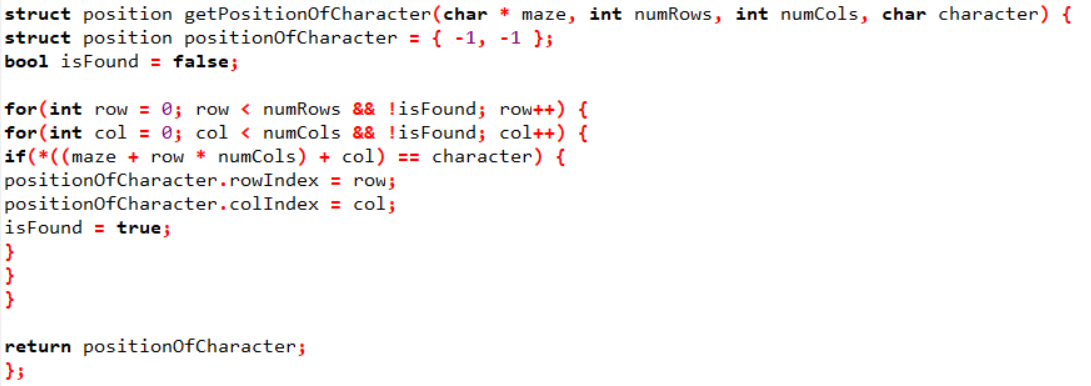
**Parameters:**

* maze: Pointer to the maze array.
* numRows: Number of rows in the maze.
* numCols: Number of columns in the maze.
* character: The character to find.

CS1002-Programming Fundamentals

**Description:**

* The function iterates over the maze and returns the position of the first occurrence of the given character.



**3.3 getCharacterAtPosition()**

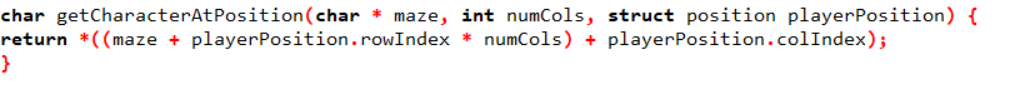
This function returns the character located at a specified position in the maze.

**Parameters:**

* maze: Pointer to the maze array.
* numCols: Number of columns in the maze.
* playerPosition: A position struct containing the player's row and column.

**Description:**

* It calculates the memory address of the desired position and returns the value at that address.



CS1002-Programming Fundamentals

**3.4 setCharacterAtPosition()**

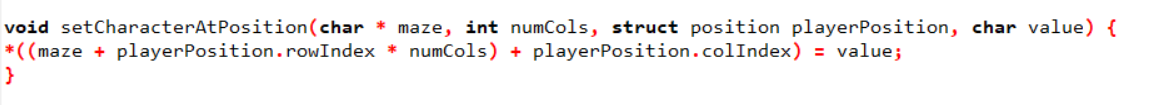
This function updates the maze array with a new character at a specified position (e.g., placing the player at a new location).

**Parameters:**

* maze: Pointer to the maze array.
* numCols: Number of columns in the maze.
* playerPosition: A position struct containing the position to update.
* value: The new character to set (e.g., 'P' for the player).

**Description:**

* The function modifies the maze by placing the character at the given position.



**3.5 game()**

This function controls the gameplay loop. It allows the player to make moves, checks whether the move is valid, and updates the maze accordingly.

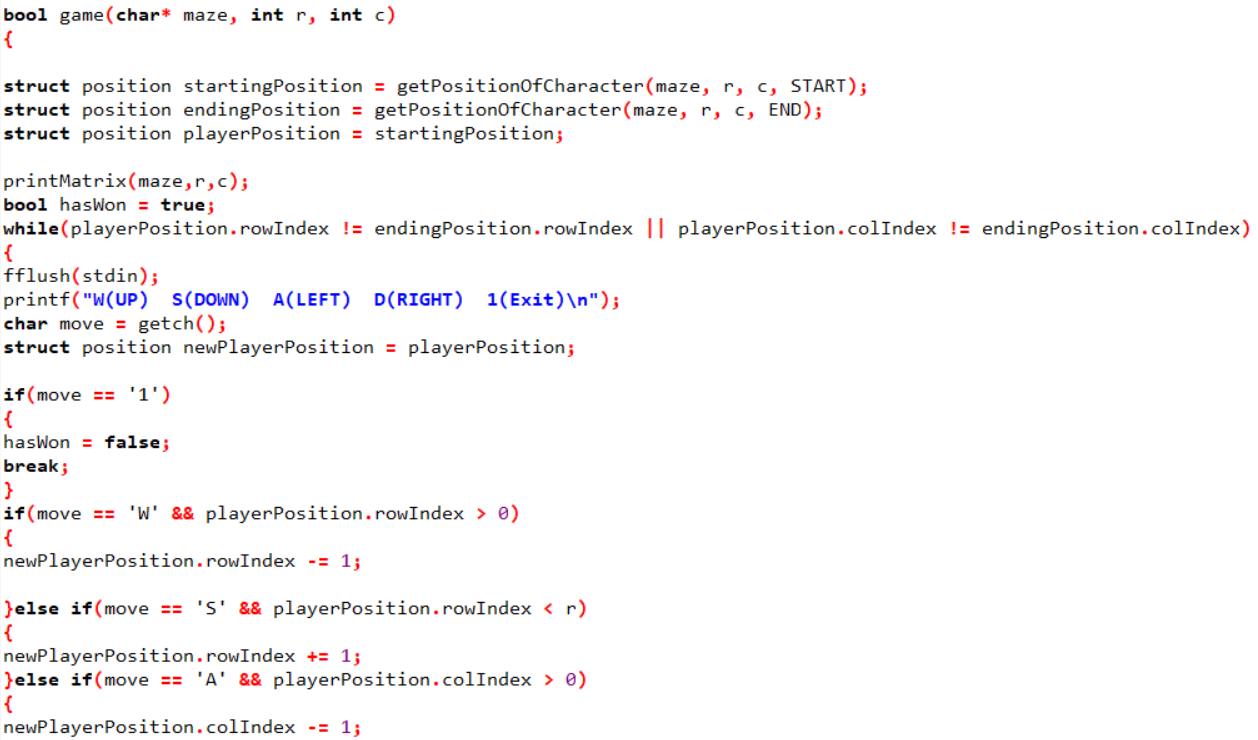
**Parameters:**

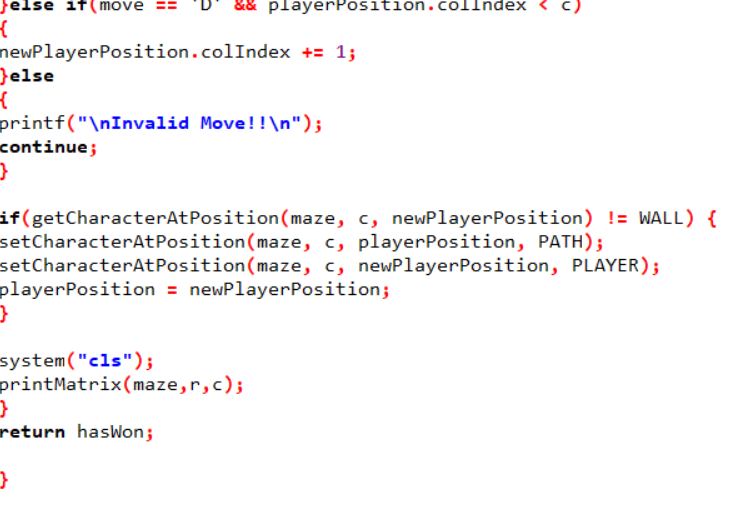
* maze: Pointer to the maze array.
* r: Number of rows in the maze.
* c: Number of columns in the maze.

**Description:**

CS1002-Programming Fundamentals

* The function checks the player's current position, listens for input, validates moves, and updates the maze until the player reaches the end ('E') or exits.





CS1002-Programming Fundamentals

**3.6 runTheGame()**

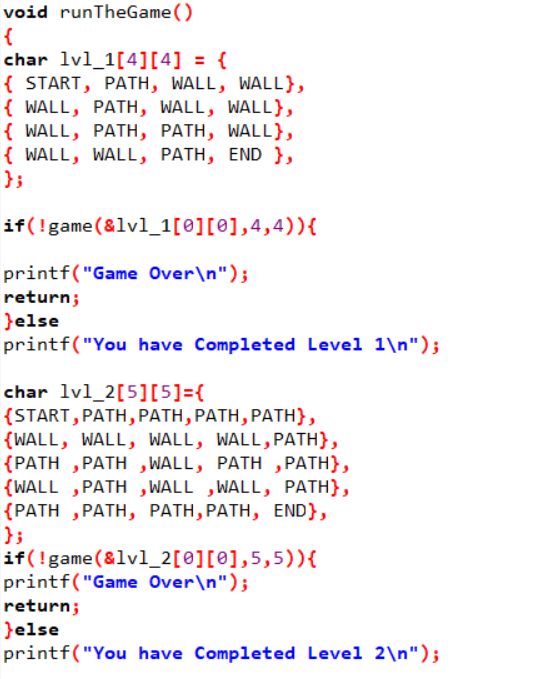
This function runs the game through multiple levels, from Level 1 to Level 7, calling the game() function for each level.

**Parameters:**

* None (handles all levels).

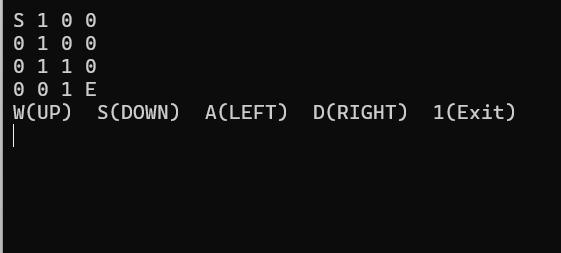
**Description:**

* The function initializes the mazes for each level and calls game() for each one. If the player completes a level, they move to the next; if they fail, the game ends.

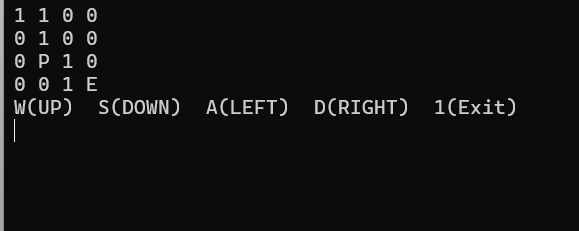


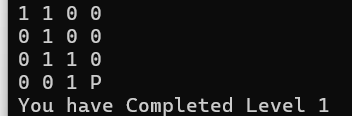
CS1002-Programming Fundamentals

**Level 1 Of Maze**



**Movement of The Player**

**  
End Of the level 1**



# 4. Testing and Results

**4.1 Test Plan**

Before finalizing the game, a thorough testing process was carried out. Below are the tests that were performed:

* **Input Validation:** Tested player movements (W, A, S, D) to ensure that invalid moves (e.g., moving through walls) are properly handled.
* **Level Completion:** Ensured that the game correctly detects when the player has reached the end ('E') and moves on to the next level.
* **Edge Cases:** Checked boundary conditions such as the player trying to move outside the maze or into walls.

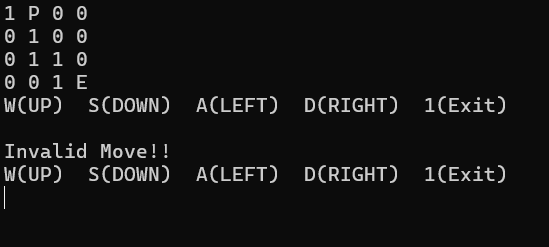
**4.2 Test Results**

* **Movement Tests:** All movement inputs ('W', 'A', 'S', 'D') were successfully processed. Invalid inputs (such as trying to move through walls or off the grid) were rejected.
* **Level Progression:** The player could successfully complete each level by reaching the end ('E'). The game correctly transitioned between levels after completion.
* **Edge Case Handling:** The game prevented the player from moving beyond the maze boundaries and correctly handled invalid moves.

Testing Summary: All major functionalities of the game were tested, and the system worked as expected without any crashes or errors.

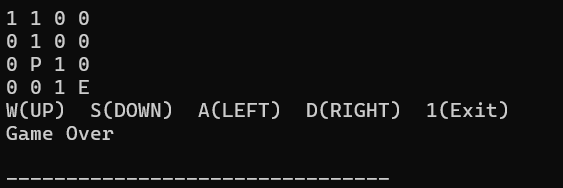
**The code below showing Different Outputs on Different Cases of conditions.**

**1.Invalid move**

****



2.Exiting the game



3.Completing the Level

### 

# 5.Pseudocode

### START GAME

### DISPLAY initial level and maze

### GET position of the Start(‘S’) and END(‘E’)

### SET player position at the START

### WHILE player position is not at the END :

### DISPLAY maze

### GET player input (W,A,S,D)

### If input is is invalid:

### PRINT “INVALID MOVE” CONTINUE to next input

### IF move is valid and not a wall:\

### UPDATE player position in maze

### CHECK IF player has reached the END

### IF player reaches END:

### PRINT “Level Complete”

### MOVE to next level

### END GAME

### ****END GAme****

# 6. Conclusion

The Maze Game project successfully implements a text-based navigation system in a 2D maze. The program demonstrates the use of arrays, functions, and user input to create an interactive and engaging game. The system is fully functional across multiple levels, and it includes proper handling of invalid moves, level progression, and player interaction.

CS1002-Programming Fundamentals