Project title: Treasure Hunt

Creator: Hamza Bektaş

Submission Date: 03/04/2025

Course Name: Algorithms And Programming

Student Number: 240101022

1. Introduction

The "Treasure Hunt" is a single-player, text-based game where the player navigates a 5x5 grid to find hidden treasures while avoiding traps. The game involves file operations, command-line arguments, and a hint system to guide the player.

The game involves file operations, command-line arguments, and a hint system to guide the player.

2. Game Overview

The game board is a 5x5 grid where 3 treasures ('T') and 3 traps ('X') are randomly placed.

The player starts at position (0,0) and must collect all treasures without stepping on traps.

Players can move using commands: 'u' (up), 'd' (down), 'l' (left), 'r' (right).

Each move provides a hint system showing the number of nearby treasures and traps.

The game ends when the player collects all treasures (win) or steps on a trap (lose).

3. Key Features

Matrix Representation: Uses a 2D array to track the board state.

Random Placement: Treasures and traps are randomly placed at game start.

Clue System: Displays the count of treasures and traps in adjacent cells.

File Operations: Allows saving/loading game states and maintaining a leaderboard.

Command-Line Arguments: Supports starting a new game, loading a saved game, and viewing the leaderboard.

4. Implementation Details

4.1 Game Setup

The game initializes a 5x5 grid where all cells are marked as unexplored ('?').
Treasures ('T') and traps ('X') are placed randomly, ensuring they don't overlap with each other or the starting position.

4.2 Player Movement

The player enters movement commands ('u', 'd', 'l', 'r').
The system ensures the player stays within bounds.
If a player moves to a treasure cell ('T'), the treasure count increases.
If a player moves to a trap cell ('X'), the game ends immediately.
The board updates the player’s position and marks visited cells ('=').

4.3 Hint System

After each move, the game calculates the number of treasures ('T') and traps ('X') in the adjacent (up to 8 surrounding) cells.
This information helps the player make strategic moves.

4.4 File Operations

Saving the Game: The player can save their progress by entering 's', providing a file name.
Loading a Game: The player can load a saved game using:

```
./treasureHunt.exe p player_name load saved_game
```

The system verifies if the save file belongs to the player before loading it.
Leaderboard Management: When a player wins, their number of moves is recorded in leaderboard.txt.

4.5 Command-Line Arguments

Start a new game:

```
./treasureHunt.exe p player_name
```

Load a saved game:

```
./treasureHunt.exe p player_name load game_file
```

View leaderboard before starting:

```
./treasureHunt.exe p player_name leaders
```

Load a game and view leaderboard:

```
./treasureHunt.exe p player_name load game_file leaders
```

5. Example Game Output

```
= = ? ? ?
? = ? ? ?
? T = * *
? ? T ? ?
P = = ? ?
Treasures collected: 2
Nearby treasures: 1, Nearby traps: 1
Enter move (u/d/l/r):
```

6. Deliverables

The project submission includes:

C source file: treasureHunt.c.

Saved game files: At least three different game states.

Leaderboard file: leaderboard.txt with at least five recorded scores.

Documentation (this file): Explaining the game logic, implementation, and screenshots.

7. Conclusion

This project demonstrates C programming skills, including file handling, arrays, randomization, and user interaction. The game provides an engaging way to practice structured programming while implementing a simple yet strategic game.

Good luck, and happy treasure hunting! 🏆