CS1050 – Lab 9

Fall 2024

Concepts to Practice

- Pointers
- Pass-by-reference

Description

Captain Codebeard and his crew of merry pirates have found an ancient map leading to a hidden treasure. However, the map is encoded with locations that need to be deciphered using pointers in C. Your task is to help Captain Codebeard find the treasure by writing a C program that uses pointers to navigate the map.

Objectives

1. Understand how to use pointers and pass them to functions.
2. Utilize symbolic constants effectively.
3. Implement functions that manipulate and navigate through arrays using pointers.

Instructions

1. Setup the Map:
    - Create a 5x5 grid representing the map.
    - Use symbolic constants to define the size of the grid and the coordinates of the treasure.
2. Define Functions:
    - Write a function to initialize the map with 0 values.
    - Write a function to display the map.
    - Write a function to move to a location using pointers.
    - Write a function to check if the current location is the treasure.
    - Write a function to leave traces on the map after each move. **(ONLY HONORS)**
3. Treasure Hunt Logic:
    - Allow the user to move in different directions (up, down, left, right) on the map.
    - Use pointers to update the current position of the player.
    - Check after each move if the player has found the treasure.

Assignment Requirements:

1. Map Initialization:
    - Create a function `void initializeMap(int map[SIZE][SIZE])` that fills the map with 0.
    - Use symbolic constants for the size of the map.
2. Map Display:
    - Create a function `void displayMap(int map[SIZE][SIZE])` that prints the map to the console.
3. Move Function:

- Create a function **`void move(int *x, int *y, char direction)`** that updates the coordinates based on the direction ('U' for up, 'D' for down, 'L' for left, 'R' for right).
- **Error Handling:** If the user tries to move out of the grid, do not change the coordinates (do nothing).

4. Treasure Check:
    - Create a function **`int checkTreasure(int x, int y)`** that returns 1 if the current location is the treasure, and 0 otherwise.
5. Main Game Logic:
    - Implement the main function that uses a loop to allow the user to input directions to move.
    - Use pointers to keep track of the current position on the map.
    - Display the map and the current position after each move.
    - Check if the treasure is found after each move.
6. Leaving Traces **(HONORS ONLY)**:
    - Create a function called **`int leaveTrace(int map[SIZE][SIZE], int x, int y)`** that returns 1, if your current position (coordinate) has never been visited before. If the position has been visited before, return 0.
    - The function must set the coordinate on the map with 1 to leave a mark to check if this coordinate has been visited before. Please remember that the map values are all initially set to 0.

Sample Output

```
ekin@ekin$ ./a.out
Welcome to Captain Codebeard's Treasure Hunt!
Here is your map:
[P] [O] [O] [O] [O]
[O] [O] [O] [O] [O]
[O] [O] [O] [O] [O]
[O] [O] [O] [O] [O]
[O] [O] [O] [O] [O]

Your current position is (0, 0).
Enter direction (U/D/L/R): D

Here is your map:
[O] [O] [O] [O] [O]
[P] [O] [O] [O] [O]
[O] [O] [O] [O] [O]
[O] [O] [O] [O] [O]
[O] [O] [O] [O] [O]

Your current position is (1, 0).
Enter direction (U/D/L/R): R

Here is your map:
[O] [O] [O] [O] [O]
[O] [P] [O] [O] [O]
[O] [O] [O] [O] [O]
[O] [O] [O] [O] [O]
[O] [O] [O] [O] [O]

Your current position is (1, 1).
Enter direction (U/D/L/R): D


Here is your map:
[O] [O] [O] [O] [O]
[O] [O] [O] [O] [O]
[O] [P] [O] [O] [O]
[O] [O] [O] [O] [O]
[O] [O] [O] [O] [O]

Your current position is (2, 1).
Enter direction (U/D/L/R): R
```

```
Here is your map:
[O] [O] [O] [O] [O]
[O] [O] [O] [O] [O]
[O] [O] [P] [O] [O]
[O] [O] [O] [O] [O]
[O] [O] [O] [O] [O]

Your current position is (2, 2).
Enter direction (U/D/L/R): R

Congratulations! You found the treasure!
```

Sample Output for Honors

```
ekin@ekin$ ./a.out
Welcome to Captain Codebeard's Treasure Hunt!
New area unlocked!
[P] [0] [0] [0] [0]
[0] [0] [0] [0] [0]
[0] [0] [0] [0] [0]
[0] [0] [0] [0] [0]
[0] [0] [0] [0] [0]
Your current position is (0, 0).
Enter direction (U/D/L/R): D

New area unlocked!
[1] [0] [0] [0] [0]
[P] [0] [0] [0] [0]
[0] [0] [0] [0] [0]
[0] [0] [0] [0] [0]
[0] [0] [0] [0] [0]
Your current position is (1, 0).
Enter direction (U/D/L/R): D

New area unlocked!
[1] [0] [0] [0] [0]
[1] [0] [0] [0] [0]
[P] [0] [0] [0] [0]
[0] [0] [0] [0] [0]
[0] [0] [0] [0] [0]
Your current position is (2, 0).
Enter direction (U/D/L/R): D

New area unlocked!
[1] [0] [0] [0] [0]
[1] [0] [0] [0] [0]
[1] [0] [0] [0] [0]
[P] [0] [0] [0] [0]
[0] [0] [0] [0] [0]
Your current position is (3, 0).
Enter direction (U/D/L/R): U

You have already been here before!
[1] [0] [0] [0] [0]
[1] [0] [0] [0] [0]
[P] [0] [0] [0] [0]
[1] [0] [0] [0] [0]
[0] [0] [0] [0] [0]
Your current position is (2, 0).
Enter direction (U/D/L/R): R
```

```
New area unlocked!
[1] [0] [0] [0] [0]
[1] [0] [0] [0] [0]
[1] [P] [0] [0] [0]
[1] [0] [0] [0] [0]
[0] [0] [0] [0] [0]
Your current position is (2, 1).
Enter direction (U/D/L/R): R

New area unlocked!
[1] [0] [0] [0] [0]
[1] [0] [0] [0] [0]
[1] [1] [P] [0] [0]
[1] [0] [0] [0] [0]
[0] [0] [0] [0] [0]
Your current position is (2, 2).
Enter direction (U/D/L/R): R

Congratulations! You found the treasure!
```

Guidelines for Grading Lab 8
40 Points Possible

General

If your program does not compile or produce any input/output (I/O) because most of the source code is commented out then your lab will receive a grade of ZERO POINTS. Further, if your program does not actually follow the specifications, but merely prints out lines that make it appear to follow the specifications, you will receive a grade of ZERO POINTS. For partial credit your C program must not only compile but also produce some valid I/O that meets the lab specifications. You program is expected to have a comment header at the top that includes your name, pawprint, the course you are taking, and the lab that you are solved (e.g., "Lab 9"). Your code should be nicely indented. **You will lose up to 10 points if you do not meet these basic requirements**. If you do not write functions as suggested and try to accomplish everything in main(), you will lose at least 50% of the points you would have otherwise received. If you use a global variable, you will get zero points.

Non-Honors

3 points: Correctly defines the function initializeMap. The function signature is correct, and the map is passed as an argument.
5 points: Fills the map with 0 values. The map is correctly populated with 0 values using nested loops.
2 points: Uses symbolic constants for size. The size of the map is defined using symbolic constants.
2 points: Correctly defines the function displayMap. The function signature is correct, and the map and player position are passed as arguments.
3 points: Displays the map correctly. The map is printed with proper formatting, and the player's current position is marked with [P].
2 points: Correctly defines the function move. The function signature is correct, and the coordinates and direction are passed as arguments.
5 points: Updates coordinates based on direction. The function correctly updates the player's coordinates based on the input direction (U/D/L/R) and ensures the player doesn't move out of the map's boundaries.
3 points: Handles invalid direction inputs. The function provides feedback or handles cases when an invalid direction is entered.
2 points: Correctly defines the function checkTreasure. The function signature is correct, and the current coordinates are passed as arguments.
3 points: Accurately checks for treasure location. The function correctly checks if the player's current coordinates match the treasure location and returns the appropriate value.
2 points: Initializes the map. The main function correctly calls initializeMap to populate the map.
3 points: Implements a loop to get user input. The main function correctly uses a loop to repeatedly prompt the user for input directions.
3 points: Updates player position using move function. The player's position is updated correctly using the move function based on user input
2 points: Displays map and checks for treasure. After each move, the map is displayed, and the treasure check is performed correctly.

Honors

3 points: Correctly defines the function initializeMap. The function signature is correct, and the map is passed as an argument.
2 points: Fills the map with 0 values. The map is correctly populated with 0 values using nested loops.
1 points: Uses symbolic constants for size. The size of the map is defined using symbolic constants.
2 points: Correctly defines the function displayMap. The function signature is correct, and the map and player position are passed as arguments.
3 points: Displays the map correctly. The map is printed with proper formatting, and the player's current position is marked with [P].
2 points: Correctly defines the function move. The function signature is correct, and the coordinates and direction are passed as arguments.
5 points: Updates coordinates based on direction. The function correctly updates the player's coordinates based on the input direction (U/D/L/R) and ensures the player doesn't move out of the map's boundaries.
3 points: Handles invalid direction inputs. The function provides feedback or handles cases when an invalid direction is entered.
2 points: Correctly defines the function checkTreasure. The function signature is correct, and the current coordinates are passed as arguments.
3 points: Accurately checks for treasure location. The function correctly checks if the player's current coordinates match the treasure location and returns the appropriate value.
2 points: Initializes the map. The main function correctly calls initializeMap to populate the map.
2 points: Implements a loop to get user input. The main function correctly uses a loop to repeatedly prompt the user for input directions.
3 points: Updates player position using move function. The player's position is updated correctly using the move function based on user input
2 points: Displays map and checks for treasure. After each move, the map is displayed, and the treasure check is performed correctly.
2 points: Correctly defines the function leaveTrace. The function signature is correct, and the map and coordinates are passed as arguments.
3 points: Updates the map correctly for leaving traces and uses the function return type to print the message if it is a new area or an already visited area.