Documentation and description of the program

The program is a product in C that solves a maze using a recursive backtracking algorithm. The program has the following features:

- It allows the user to define a maze as a 2D array of characters with different symbols for start, end, wall, and free cells.

- It validates the input maze by checking the number of rows and columns, the number of start and end cells, and their coordinates.

- It finds the solution to the maze by moving a solver in different directions and marking the path taken.

- It displays the maze and the solution on the screen with symbols and animation.

- It uses a stack data structure to store the path of the solver and pop back when a dead end is reached.

The program uses a static array to store the maze and a dynamic array to implement the stack. The program also uses constants and macros to define the symbols and dimensions of the maze. The program uses the windows.h header file to access the Sleep function, which pauses the program for a few seconds to show the animation. The program is divided into several functions that perform different tasks such as refreshing the screen, checking if a move is valid, finding the solution, and printing statistics.

The main function of the program is main(), which calls other functions and controls the flow of the program. It declares and initializes several global variables and constants that are used throughout the program. These include:

- startSymbol: A char variable that stores the symbol for the start cell in the maze ('S').

- endSymbol: A char variable that stores the symbol for the end cell in the maze ('E').

- freeSymbol: A char variable that stores the symbol for a free cell in the maze (' ').

- wallSymbol: A char variable that stores the symbol for a wall cell in the maze ('#').

- pathSymbol: A char variable that stores the symbol for a cell that is part of the solution path in the maze ('\*').

- visitedSymbol: A char variable that stores the symbol for a cell that has been visited by the solver in the maze ('+').

- notVisitedSymbol: A char variable that stores the symbol for a cell that has not been visited by the solver in the maze ('-').

- movingObjectSymbol: A char variable that stores the symbol for a cell that contains the solver in the maze ('@').

The main function also declares and initializes several other variables that are used to store information about the maze and its solution. These include:

- startFound: An int variable that counts how many start cells are found in the maze (should be 1).

- endFound: An int variable that counts how many end cells are found in the maze (should be 1).

- startRow: An int variable that stores the row index of the start cell in the maze.

- startColumn: An int variable that stores the column index of the start cell in the maze.

- endRow: An int variable that stores the row index of the end cell in the maze.

- endColumn: An int variable that stores the column index of the end cell in the maze.

- currentRow: An int variable that stores the current row index of the solver in the maze.

- currentColumn: An int variable that stores the current column index of the solver in the maze.

- steps: An int variable that counts how many steps the solver has taken in the maze.

- elements: An int variable that counts how many elements are in the maze (should be equal to MAZE\_ROWS \* MAZE\_COLUMNS).

- rows: An int variable that counts how many rows are in the maze (should be equal to MAZE\_ROWS).

The main function also declares and initializes several boolean variables that are used to indicate whether the solver can move in certain directions. These include:

- canMoveRight: A boolean variable that is true if the solver can move right in the maze, and false otherwise.

- canMoveLeft: A boolean variable that is true if ....... and so on