Programmer Manual

Programmer Manual
Maze Runner

1. Problem Description

This program uses a depth first maze solving algorithm to, if possible, find an exit in a maze provided to the program by the user. A random maze generation algorithm is also available if the user does not wish to provide their own maze. If no path from the starting position to the exit is possible, the program lets the user know.

2. Data Types and Classes

The data types used in this program fall into two categories: predefined data types and programmer-defined data types. The following subsections address the data types used.

2.1 int (predefined type)

Variables:

input	user input for the main menu
x	horizontal position of a tile in the maze
y	vertical position of a tile in the maze

2.2 bool (predefined type)

Variables:

initialized	flag determining whether a maze has data in it or not
visited	flag determining whether a tile has been put in the path yet or not

2.3 char (predefined type)

Variables:

resolve	character controlling whether the user wants to enter a new starting position in the same maze
wallChar	character used for the walls of the maze
groundChar	character used for the empty ground tiles in the maze
exitChar	character used for the exit in the maze
startChar	character used for the starting position in the maze
board[][]	two dimensional array containing the maze data

2.4 string (predefined type)

Variables:

inFileName	name of the file containing the maze data

2.5 ifstream (predefined type)

Variables:

inFile	input file

2.6 Maze (programmer-defined type)

This class has:

Data members:	Tile* current
	char board[][]
	bool init
Member functions:	Maze
	solveMaze
	generateMaze
	mazeFromFile

See the programmer manual for the Maze class for more details

2.7 Tile (programmer-defined type)

This struct has:

Data members:	int x
	int y
	bool visited
	Tile* above
	Tile* below
	Tile* left
	Tile* right

See the programmer manual for the Tile struct for more details

3. High Level Program Solution

```
Main Program
Print the title
Ask the user if they want to provide a maze file or generate a random maze
If maze file is provided:
    Ask the user for the name of the file
    Call getMazeFromFile with the provided name and open the file
    Populate the maze with the data from the file
If a random maze is generated:
    Call generateMaze to randomly generate a maze
Print the empty maze
Ask the user for a starting position
Validate the starting position
Call solveMaze to actually solve the maze
Print the solved maze
Reprompt the user for either a starting position or a menu option

function printTitle()
    Print the title screen art
```

4. Limitations and Suggestions

The current program works for a 10 X 10 maze. The program could be expanded upon in order to get an arbitrary maze size from the user. The current solved maze also just shows which tiles were used in the path to the exit. It could instead actually show the direction the program stepped over each tile to show the full path from the starting position to the exit.