Joseph Baruch

CS 121

Professor: Bruce Bolden

Program 4: Maze Assignment

Due: 4/7/2023

Program Design

Objective

The objective of this program/programming assignment is to test and explore our understanding of queue’s, dynamically allocated 2D arrays, and reading from files.

Program Description

The finalized program will read in a maze from a file and store it as a dynamically allocated 2D array. The program will then find its way through the maze by using a queue to add all surrounding available locations (North, East, South, West) to the queue. The program will then dequeue the next location and repeat, adding all available locations onto the queue. The program will solve all possible routes at the same time and will stop the program as soon as a next available location is the goal.

Requirements

* The maze must be read into a dynamically allocated 2D character array from an external ‘.txt’ file.
* The program must utilize a queue, which stores position, to solve the maze.
* The program cannot move diagonally through the maze.
* The program should not need to be recompiled to solve a different maze (different maze will be in a different file).

Program Contents

* Classes
  + Queue: The Queue class is used to enqueue and dequeue items to aid in solving the maze. The methods and functions of this class are only called in the MazeClass (not called in main).
  + MazeClass: The MazeClass is where all the work is done. The class instance is declared in the main file. The main functions are moveEnqueue, dequeue, and print (described below). This class houses all the algorithms that solve the maze.
* Class Methods: Queue
  + Public
    - Queue(): Queue constructor which initializes the head of the queue equal to NULL.
    - ~Queue(): Queue deconstructor which deletes the queue if there are remaining nodes. This is called in the MazeClass deconstructor.
    - Void Enqueue(int c, int r): This does nothing else then a standard enqueue function. It allocates a new node, initializes the row and col, and places it at the end of the queue.
    - Void Dequeue(int& c, int& r): Changes the value of x and y (pass-by-reference) and removes the node at the front of the queue.
  + Private
    - Struct qNode: Struct aids the creation of a node pointer. Also, houses other variable used in the program.
      * Int row: Saves the y coordinate of whatever location was enqueued.
      * Int col: Saves the x coordinate of whatever location was enqueued.
      * qNode\* next: Will be used to point to the next node.
    - typedef qNode \* QNodePtr: Creating node pointer variable.
    - QNodePtr head: Creating node pointer head.
* Class Methods: MazeClass
  + Public
    - MazeClass(): Constructor of the maze class. This function performs most of the program operation work.
      * Opens file.
      * Reads in height and width values from the file.
      * Dynamically allocates 2D array.
      * Initializes 2D array.
      * Closes file.
      * Finds the starting positions and enqueues it.
      * Initializes the printing aids.
    - ~MazeClass(): Deconstructor of the maze class. Calls the queue deconstructor, and deallocates 2D array.
    - Void moveEnqueue(): Contains the algorithms for finding if North, East, South or West of the current position dequeued are valid places to move. If they are valid moves enqueue them.
    - Void dequeue(): Dequeues the next location and sets the current character.
    - Void Print(): Prints the 2D Array (maze) according to the printing aid variable equation (prints every n times).
    - Char currChar: Set once the next position is dequeued. This is the character of the current position in the 2D array (maze).
    - Int printCounter: Counts every time the print function in called.
    - Int PrintEvery: The value of how often the 2D Array is printed.
  + Private
    - Int choice: User initialized variable that determines the maze that the user wants to solve.
    - Int width: The width of the maze (read in from the maze text file).
    - Int height: The height of the maze (read in from the maze text file).
    - Int x: Current x position on the 2D array. This is changed every time the queue is dequeued.
    - Int y: Current y position on the 2D array. This is changed every time the queue is dequeued.
    - Queue myQ: Declaration of an instance of the queue class.
    - Char \*\*mazeArr: Dynamically allocated 2D Array.
* Functions (not inside classes)
  + Void solveMaze(): This function is only in the maze.cpp file and it is called the first time and every time the user chooses to continue. This function calls other functions which solves the maze.
* Data structures
  + Queue: FIFO structured linked list used to help solve the maze.
  + Dynamically allocated 2D character Array: Stores the maze after it is read in from the text file.
* Other
  + Int cont: The variable that changes depending on the users choice (between 1 an 0).

Program Operation Description

* Loop while the user chooses to continue.
  + Go to solveMaze function.
    - Declare MazeClass instance (constructor)
      * Read in the maze file into a 2d dynamically allocated array.
      * Find the starting position and enqueue it to the queue.
    - Loop while the current character is not a ‘G’.
      * Dequeue the next location that is in the queue. Set the current char equal to that character (this will be used to check for ‘G’).
      * Using the current position, check to see if each direction (North, East, South and West, in that order) are valid places to move (not out of bounds or not already visited). If they are valid, add them to the queue.
        + This also check to see if any surrounding locations are ‘G’. If any surround location is ‘G’, set the currChar to ‘G’. This will break the loop.
        + Note: I realized in office hours I could have make one move validity function and change the arguments for the different directions but chose not to do this to save time.
      * Print the maze according to the value of the printEvery variable or if currChar is ‘G’.
  + Ask the user if they wish to continue and loop while their input is not valid.

Programming Log

Total Time: 14 Hours

3/31/2023: 15 Minutes

* Read through program assignment and began to understand how the program works.
* Listened to class lecture discussing how the maze will be solved.

4/1/2023: 45 Minutes

* Began to write pseudo-code and developed a simplistic outline for the program.
* At this time, I was still very confused with how the maze will be solved.
* I thought that the queue would be storing multiple locations on the same node.

4/2/2023: 2 Hours

* I was in the library with a fellow student doing homework and they explained how the whole processes worked and it finally made sense after some thinking and drawing diagrams.
* Even though I was slightly confused on the processes I knew how to implement a queue and a separate class with all my other functions in it.
* Wrote the outline to all my code and came up with all my methods and functions for my separate class.
* Looked at Professor Boldens notes on queues and decided it had a bit more that I needed and was slightly more confusing than what I needed so I made a general outline for my own queue class.
* Decided I would have a stand-alone queue class (not one that worked with a linked list class).

4/3/2023: 2 Hours

* Continued and completed implementing my queue class.
* Implemented all the algorithms of my mazeClass and finalized how main would perform.
  + I decided the method that I would choose to solve the maze would dequeue from the queue first and then enqueue. This is different than how Bolden suggested in the assignment description.
  + The reason I chose to dequeue first is because my constructor adds the starting position to the queue before the constructor.
* I ended the day with almost finishing all the little stuff including comments. I didn’t do any test or compiling.

4/4/2023: 3.5 Hours

* I ran into a big problem which was giving me a bunch of weird errors and warnings. I didn’t really know what was going on, so I went to the CSAC, and the tutor looked out my code. We found that the file guards in one of my ‘.h’ files were copied exactly from the other. Once I fixed this the error went away.
* After resolving the above issue my program compiled but I was getting a whole bunch of segmentation faults. After adding commented checkpoints, I figured them out.
  + The first segmentation fault was because I was checking to see if an array location that was out of bounds was valid. I fixed this by adding another if statement for each direction.
  + The next segmentation fault was happening because I thought that going up in the array was going north but it was going south. I adjusted my if statements accordingly.
* After the above problems were solved, I was basically finished and got a good output what I was still getting a weird output at the end of my whole output. The output error basically explained that I was deallocating my 2D array more than one time.
* I went into Professor Bolden’s office hours for the above error, and he didn’t know why exactly I was getting this problem. He said that I comment in my assignment that I didn’t need to worry about finding the source of the issues. The problem is probably associated with a bug in Visual Studio Code or could be fixed by making the size and array global variables.
* Another thing I learned in office hours is that I could compress my code by making a “checkvalidmove()” function and then change the arguments for the direction. Right now, I have four manually written out. I may fix this later but might not have enough time.

4/5/2023: 2.5 Hours

* This day was dedicated to getting the ‘work’ done for this assignment. I worked on going through the assignment description and fulfilling all the requirement.
  + I added an option for the user to choose different files (mazes) asked them at the end if they would like to pick a different file.
    - The option for them to choose what maze they wanted worked well but after the user chose to continue I the output was just the weird error from the day before.
    - I think this is the reason why I am not able to pick a different maze. I am going to go into office hours on the next day to figure out a solution.

4/6/2023: 3 Hours

* Sent Professor Bolden an email regarding the issue above.
* Worked on completing the write up including the programming log, program operation description and the program contents description.
* Professor Bolden’s email said that I don’t have to loop the program, so I took this out of the code.
* Went into office hours and found out the solution to the double free issue. It was because the constructor was being called twice. This was fixed by not calling the constructor.
* The other issue with not allowing to loop was because of object abuse. This was because I was freeing the 2D Array memory and trying to use it again. This was solved by putting everything into a function in main. This is called solveMaze. Everything works great.
* The last thing I addressed was ending and printing out the solved maze as soon as it was solved. I noticed in the bigger maze it would process more locations on the stack even though it was solved. Printing out the final maze also allowed me to pinpoint the exact time the maze was solved.
* After this I cleaned up the code (added some final comments and removed all wrap around lines), pasted everything into this document and finished the right up. Done.

4/7/2023: N/A

* Turn in assignment by 9:30 am (at the beginning of Friday’s lecture).

Outputs

Note: The spaces for the printed mazes are unbalance inside Microsoft word but are balanced (and looks nicer in the terminal where these are copied from). Request screen shots if needed.

joseph.baruch@Josephs-MacBook-Air Program4 Good copy11 % g++ maze.cpp mazeclass.cpp Queue.cpp

joseph.baruch@Josephs-MacBook-Air Program4 Good copy11 % ./a.out

Which Maze would you like to choose?

1 = (5 x 5)

2 = (10 x 10)

3 = (20 x 20)

4 = (40 x 40)

Maze: 1

Move: 5

S>>>.

V##.#

V#...

V#.#.

...#G

Move: 10

S>>>>

V##V#

V#.V.

V#.#.

V>.#G

Move: 15

S>>>>

V##V#

V#<V>

V#V#V

V>>#G

Move: 16

S>>>>

V##V#

V#<V>

V#V#V

V>>#G

Continue? (Yes = 1, No = 0): 1

Which Maze would you like to choose?

1 = (5 x 5)

2 = (10 x 10)

3 = (20 x 20)

4 = (40 x 40)

Maze: 2

Move: 15

S##^##^##.

V>>>>>>>..

#VVV####.#

.#V#...#.#

.<V#.#....

.#V###.##.

.#V#...###

.#.#.##...

........#.

##..##.##G

Move: 30

S##^##^##^

V>>>>>>>>>

#VVV####V#

^#V#...#V#

<<V#.#..V.

V#V###.##.

V#V#...###

V#V#.##...

.<V>....#.

##V.##.##G

Move: 45

S##^##^##^

V>>>>>>>>>

#VVV####V#

^#V#..^#V#

<<V#.#<<V>

V#V###V##V

V#V#^>.###

V#V#^##...

<<V>>>>.#.

##VV##.##G

Move: 59

S##^##^##^

V>>>>>>>>>

#VVV####V#

^#V#<<^#V#

<<V#V#<<V>

V#V###V##V

V#V#^>V###

V#V#^##^>>

<<V>>>>>#V

##VV##V##G

Continue? (Yes = 1, No = 0): 1

Which Maze would you like to choose?

1 = (5 x 5)

2 = (10 x 10)

3 = (20 x 20)

4 = (40 x 40)

Maze: 3

Move: 50

....###..#####<^#<<S

.##...#....#<<<<<V#V

....#...#.#<V#V###<V

##.....##<<V##V####V

#..##.#...#<<<V^#<<V

..###......#V##<<<V#

..###.##..#.##<V#<V#

#.......#.##.<<<<<V>

#...###....#.#V###VV

..#...#.#....##...#V

.##..#..#..#..###.#V

.#...#.#.#........<V

..##.#..#####.##.##V

.#.......####..###..

....####.#....#...#.

##.#.#....##...###..

.#.....#.#.#.......#

.#.##..#.....#.###.#

....#..##.###...##.#

#...#....G#####....#

Move: 100

....###<^#####<^#<<S

.##...#<<^>#<<<<<V#V

....#.<V#^#<V#V###<V

##.....##<<V##V####V

#..##.#<<V#<<<V^#<<V

..###<<<<V>#V##<<<V#

..###.##<V#.##<V#<V#

#.......#V##<<<<<<V>

#...###.<V^#V#V###VV

..#...#.#<<<V##<<^#V

.##..#..#.V#V>###^#V

.#...#.#.#<<V<<<<<<V

..##.#..#####V##V##V

.#.......####..###<V

....####.#....#...#V

##.#.#....##...###<V

.#.....#.#.#.....<V#

.#.##..#.....#.###V#

....#..##.###...##.#

#...#....G#####....#

Move: 150

..<^###<^#####<^#<<S

.##<<^#<<^>#<<<<<V#V

..<^#<<V#^#<V#V###<V

##<<<<V##<<V##V####V

#.V##^#<<V#<<<V^#<<V

..###<<<<V>#V##<<<V#

..###V##<V#.##<V#<V#

#.<<<V<^#V##<<<<<<V>

#..V###<<V^#V#V###VV

..#...#V#<<<V##<<^#V

.##..#<V#<V#V>###^#V

.#...#V#.#<<V<<<<<<V

..##.#V.#####V##V##V

.#.......####V>###<V

....####.#<<<V#...#V

##.#.#....##<V^###<V

.#.....#.#.#<<<<<<V#

.#.##..#.....#V###V#

....#..##.###.V.##V#

#...#....G#####<<<V#

Move: 200

<<<^###<^#####<^#<<S

^##<<^#<<^>#<<<<<V#V

<<<^#<<V#^#<V#V###<V

##<<<<V##<<V##V####V

#<V##^#<<V#<<<V^#<<V

<V###<<<<V>#V##<<<V#

<^###V##<V#.##<V#<V#

#<<<<V<^#V##<<<<<<V>

#<<V###<<V^#V#V###VV

<V#V>>#V#<<<V##<<^#V

V##VV#<V#<V#V>###^#V

V#<VV#V#.#<<V<<<<<<V

..##^#V>#####V##V##V

.#<<<<VV>####V>###<V

...V####V#<<<V#...#V

##.#.#..V.##<V^###<V

.#.....#.#^#<<<<<<V#

.#.##..#<<<<V#V###V#

....#..##V###<V^##V#

#...#....G#####<<<V#

Move: 201

<<<^###<^#####<^#<<S

^##<<^#<<^>#<<<<<V#V

<<<^#<<V#^#<V#V###<V

##<<<<V##<<V##V####V

#<V##^#<<V#<<<V^#<<V

<V###<<<<V>#V##<<<V#

<^###V##<V#.##<V#<V#

#<<<<V<^#V##<<<<<<V>

#<<V###<<V^#V#V###VV

<V#V>>#V#<<<V##<<^#V

V##VV#<V#<V#V>###^#V

V#<VV#V#.#<<V<<<<<<V

..##^#V>#####V##V##V

.#<<<<VV>####V>###<V

...V####V#<<<V#...#V

##.#.#..V.##<V^###<V

.#.....#.#^#<<<<<<V#

.#.##..#<<<<V#V###V#

....#..##V###<V^##V#

#...#....G#####<<<V#

Continue? (Yes = 1, No = 0): 1

Which Maze would you like to choose?

1 = (5 x 5)

2 = (10 x 10)

3 = (20 x 20)

4 = (40 x 40)

Maze: 4

Move: 250

###<<<<<^##.....##S>>>#<<<^#<<<S>>>>#^>>

<<<V####<<^###.##^##V#<^##<<V######V>>#V

V#<<<^#<<<<<^##<<<<<V>#^<<<V##<<^>#V###V

V.#<<<<<V##<<<<V##<V#V>>#<V#..##^######V

.#<<<<V##^>#<V#V>>####V###V###<^>>####<V

..#<V#V##^#.#V>##V>>>^##<<V>>##^#V####V#

.#<<V#V>>>>#.##.#####<<<V###V>>>>#<<<<V#

<<<V#.##V#V>#.##..#.###<V>##VV##V>>V####

.###..#<V>V#...#.#.#^#<V#V####<<VVVV>###

..###.#V##V.#...#.##^##V#####<<<V##V###.

#......###.#..##..##<<<V##^#####V>###.#.

.###.##..#.#.##.#..#<<<V>>>>##<<VV>#^#..

.###.##.......#..#.###<V######V###V>>>#.

..##.#.#...##..####.###V>>>###V####V#V>#

##...###.......###..###V##V###V##..#.##.

..##...###.##...#..##<<V>>V########..##.

.###.##..#..##..#.#<<<<V#VV>##.#..#.....

...,.#..#.##.##....##V##.#####.###.##..#

####....##..###.#.<<<V###..##......####.

..#....#..#.#...##.##V##.##....###...#..

....##.....#..##.#..#V#..##.###...###.#.

.###...##.#..##..##..V.....#...#..#..##.

..#.#...#.##.##.#..#..#..#...#..#..#.#.#

.###.##.......#...#..#..####..#.##.####.

..###....#..##.#....##..####...#..###..#

.##..###..#####.#.#...#..##........#..#.

.#.##....#..##...#..##.#.....#...#..###.

..#.###...#...#......###.##.#.#...#.#...

.###.##.#..#.##.#..##..###....#..#...###

.##.##.#.#..#..#...##.#..#...####.##....

.####.#.##...##.##...#.#...#.##..###.##.

#.#..#..#.##...##..##.#...###.###.#.##..

...#..####...#.......#.#...###.#....#...

.##....##.##..#.###.###.#.#...####.#...#

...##...##...#...#.....#..##..##...#..##

##......#.##...###..#....#...######.#...

.....##........##..#.##....#...#.....##.

.##..##.#..#..########.#.###########....

.#.#.#....##...##.....##....####.....##.

G...###.###........######........#######

Move: 500

###<<<<<^##.....##S>>>#<<<^#<<<S>>>>#^>>

<<<V####<<^###.##^##V#<^##<<V######V>>#V

V#<<<^#<<<<<^##<<<<<V>#^<<<V##<<^>#V###V

V>#<<<<<V##<<<<V##<V#V>>#<V#..##^######V

V#<<<<V##^>#<V#V>>####V###V###<^>>####<V

V>#<V#V##^#.#V>##V>>>^##<<V>>##^#V####V#

^#<<V#V>>>>#.##.#####<<<V###V>>>>#<<<<V#

<<<V#^##V#V>#^##..#.###<V>##VV##V>>V####

V###<^#<V>V#<^>#.#.#^#<V#V####<<VVVV>###

V>###^#V##V>#^>>#.##^##V#####<<<V##V###.

#V>>>>>###V#^>##..##<<<V##^#####V>###.#.

.###V##<^#V#^##.#..#<<<V>>>>##<<VV>#^#..

.###V##<<<V>>>#..#.###<V######V###V>>>#.

..##V#.#<<V##V>####^###V>>>###V####V#V>#

##<<V###<<V>>V^###^>###V##V###V##..#.##.

..##V>>###V##<<^#^>##<<V>>V########..##.

.###V##^.#V>##<^#^#<<<<V#VV>##.#..#.....

...,V#^>#.##.##<<<^##V##.#####.###.##..#

####V>>>##..###V#<<<<V###..##......####.

..#<VVV#..#.#<<V##V##V##^##....###...#..

..<V##V..^.#<V##^#V^#V#^>##.###...###.#.

.###...##^#<V##^>##<<V>>>>>#^>>#..#..##.

..#.#..^#^##V##^#^>#<V#VV#V>>#V>#..#.#.#

.###.##<<<<<V>#<<^#<V#<V####V>#V##.####.

..###..<V#<V##.#<<<V##<V####VV>#^>###..#

.##..###V.#####^#V#V>>#V>##<VVV>>>>#..#.

.#.##....#..##<<^#<V##.#V>>>V#VVV#V>###.

..#.###...#...#<<<<V>###V##V#^#VV>#.#...

.###.##.#..#.##V#<V##..###<V>>#VV#...###

.##.##.#.#..#..#<<V##.#.^#<VV####.##....

.####.#.##...##.##V>>#.#<<V#V##..###.##.

#.#..#..#.##..^##<V##.#.<V###.###.#.##..

...#..####...#<<<<V>>#.#.V.###.#....#...

.##....##.##..#V###V###.#.#...####.#...#

...##...##...#.V.#<V>>.#..##..##...#..##

##......#.##...###<V#....#...######.#...

.....##........##..#.##....#...#.....##.

.##..##.#..#..########.#.###########....

.#.#.#....##...##.....##....####.....##.

G...###.###........######........#######

Move: 717

###<<<<<^##.....##S>>>#<<<^#<<<S>>>>#^>>

<<<V####<<^###.##^##V#<^##<<V######V>>#V

V#<<<^#<<<<<^##<<<<<V>#^<<<V##<<^>#V###V

V>#<<<<<V##<<<<V##<V#V>>#<V#..##^######V

V#<<<<V##^>#<V#V>>####V###V###<^>>####<V

V>#<V#V##^#.#V>##V>>>^##<<V>>##^#V####V#

^#<<V#V>>>>#.##.#####<<<V###V>>>>#<<<<V#

<<<V#^##V#V>#^##..#.###<V>##VV##V>>V####

V###<^#<V>V#<^>#.#.#^#<V#V####<<VVVV>###

V>###^#V##V>#^>>#.##^##V#####<<<V##V###.

#V>>>>>###V#^>##..##<<<V##^#####V>###.#.

.###V##<^#V#^##.#..#<<<V>>>>##<<VV>#^#..

.###V##<<<V>>>#..#.###<V######V###V>>>#.

..##V#.#<<V##V>####^###V>>>###V####V#V>#

##<<V###<<V>>V^###^>###V##V###V##..#.##.

..##V>>###V##<<^#^>##<<V>>V########..##.

.###V##^>#V>##<^#^#<<<<V#VV>##.#..#.....

...,V#^>#.##.##<<<^##V##.#####.###.##..#

####V>>>##..###V#<<<<V###..##......####.

<^#<VVV#<^#.#<<V##V##V##^##....###...#..

<<<V##V><^>#<V##^#V^#V#^>##.###...###.#.

V###<<V##^#<V##^>##<<V>>>>>#^>>#..#..##.

V>#.#<<^#^##V##^#^>#<V#VV#V>>#V>#..#.#.#

V###.##<<<<<V>#<<^#<V#<V####V>#V##.####.

V>###<<<V#<V##.#<<<V##<V####VV>#^>###..#

V##..###V>#####^#V#V>>#V>##<VVV>>>>#..#.

V#.##<<<V#..##<<^#<V##.#V>>>V#VVV#V>###.

V>#.###<V>#...#<<<<V>###V##V#^#VV>#V#...

V###.##V#V>#.##V#<V##..###<V>>#VV#<V>###

V##.##.#.#<^#..#<<V##.#<^#<VV####.##V>>>

V####.#.##<<^##.##V>>#.#<<V#V##..###V##V

#.#<^#..#.##<<^##<V##.#<<V###.###.#.##<V

^..#<^####<<V#<<<<V>>#.#<V>###.#....#<<V

^##<<<^##.##V>#V###V###.#V#<^>####.#<<V#

<<^##<<^##<<^#<V>#<V>>>#<V##^>##...#<V##

##<<<<<^#^##<<V###<V#VV>V#^>>######.#V>>

<<<<V##<<<<<<<V##<V#.##VV>>#V>>#.....##V

V##<V##V#<V#<V########.#V###########<<<V

V#.#V#<<<V##<V>##^>>>>##V>>>####^>>>>##V

G...###V###<<VV>>>>######VVV>>>>>#######

Continue? (Yes = 1, No = 0): 0

joseph.baruch@Josephs-MacBook-Air Program4 Good copy11 %

Program

Note: The comments were minimized to avoid wrap around. Also, the indentation is very small inside word but is more appropriate inside nano or VSCode.

/\* maze.cpp

Joseph Baruch

Professor: Bruce Bolden

CS 121

Due: 4/7/2023

\*/

#include <iostream>

#include "mazeclass.h"

using namespace std;

void solveMaze(); // prototype

int main(){

int cont = 1;

while(cont == 1 ){ // user chooses to loop

solveMaze();

do{ // check for valid input

cout << "Continue? (Yes = 1, No = 0): ";

cin >> cont;

cout << endl;

}while( cont != 1 && cont != 0 );

}

}

void solveMaze(){

// declare class instance (contructor runs)

MazeClass maze;

// loop until current position is the goal

while( maze.currChar != 'G' ){

// update currChar and gets current position

maze.dequeue();

// Checks for valid move and enqueue

maze.moveEnqueue();

// prints output

maze.Print();

}

}

/\* mazeclass.cpp

Joseph Baruch

CS 121

\*/

#include <iostream>

#include <fstream>

#include "mazeclass.h"

using namespace std;

MazeClass::MazeClass(){

ifstream myFile; // open file

do{ // loops if user input is invalid

cout << "Which Maze would you like to choose? " << endl;

cout << " 1 = (5 x 5) " << endl;

cout << " 2 = (10 x 10) " << endl;

cout << " 3 = (20 x 20) " << endl;

cout << " 4 = (40 x 40) " << endl;

cout << "Maze: ";

cin >> choice;

}while( choice != 1 && choice != 2 && choice != 3 && choice != 4 );

// file is opened according to user

// printEvery is change depending on size of maze

switch( choice ){ // user chooses maze

case 0: // base case

break;

case 1:

myFile.open("maze1.txt");

printEvery = 5;

break;

case 2:

myFile.open("maze2.txt");

printEvery = 15;

break;

case 3:

myFile.open("maze3.txt");

printEvery = 50;

break;

case 4:

myFile.open("maze4.txt");

printEvery = 250;

break;

}

myFile >> width; // read in height and width

myFile >> height;

mazeArr = new char \*[height]; // allocate 2d array

for( int i = 0; i < height; i++){

mazeArr[i] = new char[width];

}

// initialize 2d array by reading file

for(int i = 0; i < height; i++){

for(int j = 0; j < width; j++){

myFile >> mazeArr[i][j];

}

}

myFile.close(); // close file

// find 's', init x and y, enqueue

for(int i = 0; i < height; i++){

for(int j = 0; j < width; j++){

if( mazeArr[i][j] == 'S'){

x = j;

y = i;

myQ.Enqueue(x,y);

}

}

}

// initialize printing aids

printCounter = 0;

}

MazeClass::~MazeClass(){

myQ.~Queue(); // deletes queue

// deallocate 2D array

for( int i = 0; i < height; i ++ ){

delete [] mazeArr[i];

}

delete [] mazeArr;

}

void MazeClass::moveEnqueue(){

// if north exists

if( 0 <= (y - 1) ){ // north

// if north is unvisited

if(mazeArr[y - 1][x] == '.'){

// enqueue

myQ.Enqueue(x, (y - 1));

// make north visited (invalid)

mazeArr[y - 1][x] = '^';

}

// check for 'G'

if(mazeArr[y - 1][x] == 'G' ){

// will break loop

currChar = 'G';

}

}

if( (width - 1) >= (x + 1) ){ // east

if(mazeArr[y][x + 1] == '.'){

myQ.Enqueue((x + 1), y);

mazeArr[y][x + 1] = '>';

}

if(mazeArr[y][x + 1] == 'G' ){

currChar = 'G';

}

}

if( (height - 1) >= (y + 1) ){ // south

if(mazeArr[y + 1][x] == '.'){

myQ.Enqueue(x, (y + 1));

mazeArr[y + 1][x] = 'V';

}

if(mazeArr[y + 1][x] == 'G' ){

currChar = 'G';

}

}

if( 0 <= (x - 1) ){ // west

if(mazeArr[y][x - 1] == '.'){

myQ.Enqueue((x - 1), y);

mazeArr[y][x - 1] = '<';

}

if(mazeArr[y][x - 1] == 'G' ){

currChar = 'G';

}

}

}

void MazeClass::dequeue(){

// dequeue next current position

myQ.Dequeue(x, y);

// set currChar

currChar = mazeArr[y][x];

}

void MazeClass::Print(){

printCounter++;

// prints array every pritnEvery times

if(printCounter % printEvery == 0 || currChar == 'G'){

cout << "Move: " << printCounter << endl;

for(int i = 0; i < height; i++){

for(int j = 0; j < width; j++){

cout << mazeArr[i][j];

}

cout << endl;

}

cout << endl;

}

}

/\* mazeclass.h

Joseph Baruch

CS 121

\*/

#ifndef MAZECLASS\_H // file guards

#define MAZECLASS\_H

#include <iostream>

#include <fstream>

#include "Queue.h"

using namespace std;

class MazeClass{

public:

MazeClass(); // constructor

~MazeClass(); // deconstructor

void moveEnqueue();

void dequeue();

void Print();

char currChar; // current character

int printCounter; // Counts the number of steps

int printEvery; // print every amount time

private:

int choice;

int width; // width of array

int height; // height of array

int x; // == col

int y; // == row

Queue myQ; // declares class instance

char \*\*mazeArr; // 2D array variable

};

#endif // file guards

/\* Queue.cpp

Joseph Baruch

CS 121

\*/

#include <iostream>

#include "Queue.h"

using namespace std;

Queue::Queue(){

head = NULL;

}

Queue::~Queue(){

while( head != NULL){ // go until h == NULL

QNodePtr del = head; // aux pointer = head

head = head->next; // move head

del->next = NULL;

delete del; // delete node

}

head = NULL; // queue is empty

}

void Queue::Enqueue(int c, int r){

QNodePtr n = new qNode; // allocate

n->row = r; // initialize

n->col = c;

if( head == NULL){ // if queue is empty

head = n;

}else{

QNodePtr p = head; // aux pointer

// move to the end of the queue

while( p->next != NULL){

p = p->next;

}

p->next = n;

}

}

void Queue::Dequeue( int& c, int& r){

// aux pointer equal to head

QNodePtr p = head;

// updates current position

r = head->row;

c = head->col;

if( p == NULL){

// nothing

}else{

head = p->next;

p->next = NULL;

delete p;

}

}

/\* Queue.h

Joseph Baruch

CS 121

\*/

#ifndef QUEUE\_H // file guards

#define QUEUE\_H

#include <iostream>

using namespace std;

class Queue{

public:

Queue(); // constructor

~Queue(); // deconstructor

void Enqueue(int c, int r);

void Dequeue(int& c, int& r);

private:

struct qNode{

int row; // y coordinate

int col; // x coordinate

qNode\* next;

};

typedef qNode \* QNodePtr;

QNodePtr head;

};

#endif