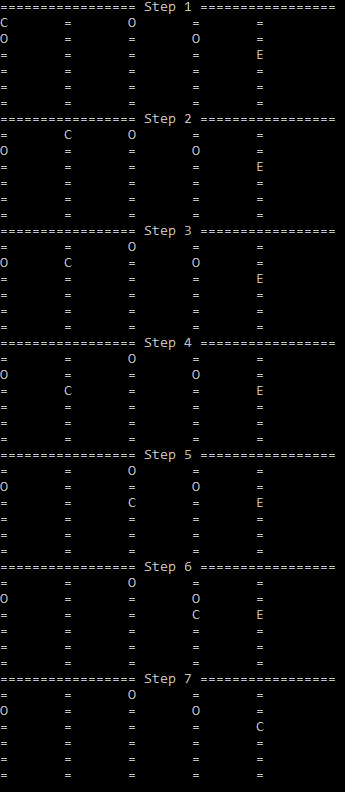
a. Your name – Joseph Rossitto

b. Id # - I don’t know how to find my Id #

c. Assignment # - Assignment 2

d. Due date – 9/30/20

e. Submitted date - 9/30/20



//a.File Name - program.cpp

//b.Author - Joseph Rossitto

//c.Date - 9/30/20

//d.Compiler Used - Visual Studio

//e.Brief Description of the file - Contains the main program, and how to deal with data passed in from mazeData.txt

#include <iostream>

#include <string>

#include <fstream>

#include "matrix.h"

using namespace std;

void readFile();

int main()

{

//matrix m1(2, 3), m2(2, 3), m3(2, 3);

//m1(0, 0) = 5;

//m2(0, 0) = 7;

//m3 = m2;

//cout << m3(0, 0) << endl;

//m3 += m1;

//cout << "Now m3(0,0) = " << m3(0, 0);

readFile();

return 0;

}

void readFile()

{

string mazeData;

bool timeToCreateMaze = true;

bool timeToPlaceEntrance = false;

bool timeToPlaceExit = false;

bool timeToPlaceObsticals = false;

bool timeToEndProgram = false;

bool fullString = false;

bool lookingForX = true;

bool lookingForY = false;

int x = 0;

int y = 0;

//int x = 6;

//int y = 5;

int currentX = 0;

int currentY = 0;

int endX = 0;

int endY = 0;

ifstream inFile;

matrix mazeBuilder;

//Criteria 1: Use file io

inFile.open("mazeData.txt");

if (!inFile) {

cout << "Unable to open file";

exit(1); // terminate with error

}

while (inFile >> mazeData)

{

//cout << "Maze Data: " << mazeData << endl;

for (char& c : mazeData)

{

if (c == '-')

{

timeToPlaceObsticals = false;

timeToEndProgram = true;

}

if (c != '('

&& c != ')'

&& c != ',')

{

if (lookingForX == true)

{

lookingForX = false;

lookingForY = true;

fullString = false;

x = ((int)c - '0' -1);

}

else

{

lookingForX = true;

lookingForY = false;

fullString = true;

y = ((int)c - '0' -1);

}

}

}

if (timeToCreateMaze == true && fullString == true)

{

timeToCreateMaze = false;

timeToPlaceEntrance = true;

//cout << mazeBuilder;

matrix initializeMaze(x+1, y+1);

mazeBuilder = initializeMaze;

//mazeBuilder(initializeMaze);

//cout << "building maze: " << x << y << endl;

}

else if (timeToPlaceEntrance == true && fullString == true)

{

timeToPlaceEntrance = false;

timeToPlaceExit = true;

mazeBuilder(x, y) = 1;

currentX = x;

currentY = y;

//cout << "Placing Entrance: " << x << y << endl;

}

else if (timeToPlaceExit == true && fullString == true)

{

timeToPlaceExit = false;

timeToPlaceObsticals = true;

endX = x;

endY = y;

mazeBuilder(x, y) = 2;

//cout << "Placing Exit: " << x << y << endl;

}

else if(timeToPlaceObsticals == true && fullString == true)

{

mazeBuilder(x, y) = 3;

//cout << "placing obstical " << x << y << endl;

}

else if (timeToEndProgram == true)

{

//cout << "All Data is read" << endl;

bool finished = false;

int count = 0;

bool moveUpOpen = false;

bool shouldMoveUp = false;

bool moveLeftOpen = false;

bool shouldMoveLeft = false;

while (finished == false)

{

bool moveRightOpen = mazeBuilder(currentX + 1, currentY) != 3;

bool shouldMoveRight = currentX < endX;

bool moveDownOpen = mazeBuilder(currentX, currentY + 1) != 3;

bool shouldMoveDown = currentY < endY;

if (currentY != 0)

{

bool moveUpOpen = mazeBuilder(currentX, currentY - 1) != 3;

bool shouldMoveUp = currentY > endY;

}

else

{

bool moveUpOpen = false;

}

if (currentX != 0)

{

bool moveLeftOpen = mazeBuilder(currentX - 1, currentY) != 3;

bool shouldMoveLeft = currentX > endX;

}

else

{

bool moveLeftOpen = false;

}

//cout << " current x: " << currentX << " current y: " << currentY << " end x: " << endX << " end y: " << endY << endl;

count = count + 1;

cout << "================= Step " << count << " =================" << endl;

cout << mazeBuilder;

if (moveRightOpen && shouldMoveRight)

{

//cout << "Moving Right" << endl;

mazeBuilder(currentX, currentY) = 0;

mazeBuilder(currentX + 1, currentY) = 1;

currentX = currentX + 1;

}

else if (moveDownOpen && shouldMoveDown)

{

//cout << "Moving Down" << endl;

mazeBuilder(currentX, currentY) = 0;

mazeBuilder(currentX, currentY + 1) = 1;

currentY = currentY + 1;

}

else if (moveLeftOpen && shouldMoveLeft)

{

//cout << "Moving Down" << endl;

mazeBuilder(currentX, currentY) = 0;

mazeBuilder(currentX - 1, currentY) = 1;

currentX = currentX - 1;

}

else if (moveUpOpen && shouldMoveUp)

{

//cout << "Moving Down" << endl;

mazeBuilder(currentX, currentY) = 0;

mazeBuilder(currentX, currentY - 1) = 1;

currentY = currentY - 1;

}

else

{

cout << "Help I am stuck!";

}

if (currentX == endX && currentY == endY)

{

finished = true;

count = count + 1;

cout << "================= Step " << count << " =================" << endl;

cout << mazeBuilder;

//cout << "Yay we are done";

}

if (count > 20)

{

finished = true;

cout << "break here and check program" << endl;

}

}

}

else

{

//cout << "Error error will robinson";

}

}

}

//a.File Name - matrix.cpp

//b.Author - Joseph Rossitto

//c.Date - 9/30/20

//d.Compiler Used - Visual Studio

//e.Brief Description of the file - Contains matrix function implementation details

#include "matrix.h"

#include <stdlib.h>

#include <iostream>

#include <cassert>

using namespace std;

matrix::matrix()

{

//Matrix = matrix(1, 1);

}

matrix::matrix(int row=1, int column=1) :rows(row), columns(column)

{

p = new int\* [rows];

for (int i = 0; i < rows; i++)

{

p[i] = new int[columns];

}

for (int i = 0; i < rows; i++)

{

for (int j = 0; j < columns; j++)

{

p[i][j] = 0;

}

}

}

matrix::matrix(const matrix& m) :rows(m.rows), columns(m.columns)

{

p = new int\* [rows];

for (int i = 0; i < rows; i++)

{

p[i] = new int[columns];

}

for (int i = 0; i < rows; i++)

{

for (int j = 0; j < columns; j++)

{

p[i][j] = m.p[i][j];

}

}

}

matrix::~matrix()

{

for (int i = 0; i < rows; i++)

{

delete[] p[i];

}

delete[]p;

}

matrix matrix::operator=(const matrix& m)

{

//Deallocate old memory

for (int i = 0; i < rows; i++)

{

delete[] p[i];

}

delete[]p;

//update data members

rows = m.rows;

columns = m.columns;

//initalize new matrix

//matrix newMatrix(rows, columns);

p = new int\* [rows];

for (int i = 0; i < rows; i++)

{

p[i] = new int[columns];

}

for (int i = 0; i < rows; i++)

{

for (int j = 0; j < columns; j++)

{

p[i][j] = 0;

}

}

//return new matrix

return (\*this);

//assert(m.columns == columns && m.rows == rows);

//int i, j;

//for (i = 0; i < rows; i++)

// for (j = 0; j < columns; j++)

// p[i][j] = m.p[i][j];

//return(\*this);

}

float\*\* matrix::setSize(int x, int y)

{

return nullptr;

}

//matrix matrix::operator+=(matrix& m)

//{

// return matrix();

//}

ostream& operator<<(ostream& os, matrix& matrix)

{

for (int i = 0; i < matrix.rows; i++)

{

for (int j = 0; j < matrix.columns; j++)

{

if (matrix(i, j) == 0)

{

os << "=\t";

}

if (matrix(i, j) == 1)

{

os << "C\t";

}

if (matrix(i, j) == 2)

{

os << "E\t";

}

if (matrix(i, j) == 3)

{

os << "O\t";

}

}

os << endl;

}

return os;

}

//a.File Name - matrix.h

//b.Author - Joseph Rossitto

//c.Date - 9/30/20

//d.Compiler Used - Visual Studio

//e.Brief Description of the file - the function definitions for matrix.cpp

#pragma once

#include <iostream>

#include <string>

using namespace std;

class matrix

{

private:

int rows;

int columns;

int\*\* p;

public:

matrix();

matrix(int, int);

matrix(const matrix& m);//avoid shallow copy by doing deep copy

~matrix();//

int& operator()(int i, int j) { return p[i][j]; }

matrix operator=(const matrix& m);//avoid shallow copy by doing deep copy

friend ostream& operator<<(ostream& os, matrix& matrix);

float \*\* setSize(int x, int y);

//matrix operator+=(matrix& m);

};

What I learned:

I’m new to C++, my main languages are python and C# so I learned quite a lot while programming this exercise. I haven’t had a lot of experience programming with pointers before so overloading the matrix operator= was pretty hard for me. I learned in order to correctly reallocate the memory I had to first deallocate the old matrix. I also had to construct a default constructor that defaulted to a small matrix rather then a (0,0) matrix because that will give you memory issues in the future. Working with the pointers was challenging and took me a little bit to wrap my head around but in the end I think I ended up doing ok with the solution.