John Metcalfe, Josh Jenson, Jake Kennedy

EECS 448

**Lab 6**

**Source**

main.cpp

//////////////////////////////////

**checkMatrix.cpp**

#include <vector>

bool checkMatrix(vector<vector<int> > mat) {

for (int i=0; i < mat.size() - 1; i++) {

if(mat[i].size() != mat[i+1].size()) {return false;}

};

return true;

};

////////////////////////////////

**matrixSize.cpp**

#include <vector>

bool matrixSize(vector<vector<int> > mat1, vector<vector<int> > mat2, int op){

if (op == 1){//Addition

//Matricies must be same size

if(mat1[1].size == mat2[1].size() && mat1.size() == mat2.size()){

//Checks dimensions

return true;

}else{

return false;

}

}else if(op == 2){//Multiplication

//1st matrix is 2\*3, second is 3\*4

if(mat1.size() == 2 && mat1[1].size() == 3 && mat2.size() == 3 && mat2[1].size() == 4){

return true;

}else

return false;

}

}else if(op == 3){//transpose

//Only uses 1 matrix, is 2\*3

if(mat1.size() == 2 && mat1[1].size() == 3){

return true;

}else{

return false;

}

}else{//op is bad

return false;

}

}

////////////////////////////////

**reader.cpp**

#include <string>

#include <vector>

#include <fstream>

#include <iostream>

#include <algorithm>

#include <sstream>

using namespace std;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// filepath: path to the .cvs file to read

// matrix: a reference to the matrix that the function will fill with the contents of the tile

// return: whether the read was successful

bool ReadFile(string filepath, vector<vector<int> > & matrix)

{

string buffer;

ifstream file;

//Opens the file

file.open(filepath.c\_str());

//Check to see if file opened successfully

if(file.is\_open())//file open

{

int rows = 0;

int columns = 0;

int tempColumns;

//Find the number of rows and columns in the file

while( getline(file, buffer))

{

tempColumns = 1;

//Count ',' to find columns

for(int i = 0; i < buffer.size(); i++)

{

if(buffer[i] == ',')

{

tempColumns++;

}

}

if(columns == 0)

{

columns = tempColumns;

}else{

if(tempColumns != columns)

{

cout << "The file is not correctly formatted. Cannot be read.\n";

return false;

}

}

rows++;

}

//Resize the vectors to the right size

matrix.resize(rows);

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

{

matrix[i].resize(columns);

}

//Reset the ifstream

file.clear();

file.seekg(0, ios::beg);

//Read in an assign the values

int vectorRowIndex = 0;

int vectorColIndex;

int beginningOfNumIndex;

stringstream ss;

while( getline(file, buffer))

{

vectorColIndex = 0;

beginningOfNumIndex = 0;

for(int i = 0; i <= buffer.size(); i++)

{

//If its the end of a number

if((buffer[i] == ',') || (i == (buffer.size())))

{

//Put that chunk in a string stream

for(int j = beginningOfNumIndex; j < i; j++)

{

ss << buffer[j];

}

//Put the stringstream into the vector to convert to int

ss >> matrix[vectorRowIndex][vectorColIndex];

//Check for conversion failure

if(ss.fail())

{

cout << "Failed to create matrix (bad conversion)." << endl;

return false;

}

//Reset and clear the string stream

ss.str("");

ss.clear();

beginningOfNumIndex = i+1;

vectorColIndex++;

}

}

vectorRowIndex++;

}

}else //file failed to open

{

cout << "File failed to open.\n";

return false;

}

//Close file

file.close();

return true;

}

////////////////////////////////

**matrixAdd.cpp**

#include <vector>

#include <iostream>

using namespace std;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// m1: the first matrix to be added

// m2: the second matrix to be added

// return: a matrix that is the result of the addition

vector<vector<int>> matrixAdd(vector<vector<int> > m1, vector<vector<int> > m2)

{

vector<vector<int> > result = {{0}};

//make sure they are the same size

if((m1.size() != m2.size()) || (m1[0].size() != m2[0].size()))

{

cout << "Matrixes are not the same size. Matrixes not added" << endl;

return result;

}

//Resize the result matrix

result.resize(m1.size());

for(int i = 0; i < result.size(); i++)

{

result[i].resize(m1[0].size());

}

//Assign the values

for(int i = 0; i < result.size(); i++)

{

for(int j = 0; j < result[0].size(); j++)

{

result[i][j] = m1[i][j] + m2[i][j];

}

}

//Return the result

return result;

}

////////////////////////////////

**matrixMult.cpp**

//Matrix Multiplication

//Author: Jake Kennedy

//Made on 3/26/15

#include <vector>

#include <iostream>

using namespace std;

vector<vector<int> > matrixMult(vector<vector<int> > m1, vector<vector<int> > m2){

//Should be a 2\*3 matrix times a 3\*4 matrix to produce a 2\*4 matrix

//Create a 2\*4 vector

vector<int> rsltCol(4);

vector<vector<int> > rslt(2,rsltCol);

//Multiplication and Storage

for(int i=0;i<rslt.size();i++){

for(int j=0;j<rslt[i].size();j++){

int product = 0;

for(int k=0;k<3;k++){

//Multiply the values and add to the product

product += m1[i][k]\*m2[k][j];

}

//assign the product to the correct matrix cell

rslt[i][j] = product;

}

}

//Here is where you would return/print the resulting matrix.

return rslt;

}

////////////////////////////////

**matrixTranspose.cpp**

**GitHub**

Link: <https://github.com/jmkennedy/448_Lab6>

**Output**