

Lab 6

Source

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

main.cpp

```
#include "checkMatrix.cpp"
#include "matrixAdd.cpp"
#include "matrixMult.cpp"
#include "matrixSize.cpp"
#include "reader.cpp"
#include "matrixTrans.cpp"

#include <vector>
#include <iostream>

using namespace std;

void PrintMatrix (vector<vector<int> > m);

int main()
{
    vector<vector<int> > matrixAdd1;
    vector<vector<int> > matrixAdd2;
    vector<vector<int> > matrixMult1;
    vector<vector<int> > matrixMult2;
    vector<vector<int> > matrixTrans1;

    if(ReadFile("add1.csv", matrixAdd1) && ReadFile("add2.csv", matrixAdd2))
    {
        if(checkMatrix(matrixAdd1) && checkMatrix(matrixAdd2))
        {
            if(matrixSize(matrixAdd1, matrixAdd2, 1))
            {
                cout << "-----Adding matrixes-----\n";
                cout << "Matrix 1:\n";
                PrintMatrix(matrixAdd1);
                cout << "Added with Matrix 2:\n";
                PrintMatrix(matrixAdd2);
                cout << "Equals matrix 3:\n";
                PrintMatrix(matrixAdd(matrixAdd1, matrixAdd2));
            }else{
```

```

        cout << "Matrix is not the right size (add)." << endl;
    }
} else {
    cout << "File is not a matrix." << endl;
}
} else {
    cout << "Failed to read addition matrix." << endl;
}

if(ReadFile("mult1.csv", matrixMult1) && ReadFile("mult2.csv", matrixMult2))
{
    if(checkMatrix(matrixMult1) && checkMatrix(matrixMult2))
    {
        if(matrixSize(matrixMult1, matrixMult2, 2))
        {
            cout << "-----Multiplying matrixes-----\n";
            cout << "Matrix 1:\n";
            PrintMatrix(matrixMult1);
            cout << "Multiplied with Matrix 2:\n";
            PrintMatrix(matrixMult2);
            cout << "Equals matrix 3:\n";
            PrintMatrix(matrixMult(matrixMult1, matrixMult2));
        } else {
            cout << "Matrix is not the right size (mult)." << endl;
        }
    } else {
        cout << "File is not a matrix." << endl;
    }
} else {
    cout << "Failed to read multiplication matrix." << endl;
}

if(ReadFile("mult1.csv", matrixTrans1))
{
    if(checkMatrix(matrixTrans1))
    {
        if(matrixSize(matrixTrans1, matrixTrans1, 3))
        {
            cout << "-----Transposing matrix-----\n";
            cout << "Matrix 1:\n";
            PrintMatrix(matrixTrans1);
            transpose(matrixTrans1);
            cout << "The Transpose of matrix 1 is:\n";
            PrintMatrix(matrixTrans1);
        } else {
            cout << "Matrix is not the right size." << endl;
        }
    } else {

```

```

        cout << "File is not a matrix." << endl;
    }
} else {
    cout << "Failed to read transpose matrix." << endl;
}

return 0;
}

void PrintMatrix(vector<vector<int> > m)
{
    for(int i = 0; i < m.size(); i++)
    {
        for(int j = 0; j < m[0].size(); j++)
        {
            cout << m[i][j] << " ";
        }

        cout << endl;
    }
}

```

////////////////////////////////////
checkMatrix.cpp

```

#include <vector>
#include <iostream>

```

```

using namespace std;

```

```

//takes in a 2D vector of ints

```

```

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

```

```

    //goes along each row of the input vector, ends at second-to-last row to avoid calling an out-of-
    bounds index does not need to run for the last row, it will be compared with the second-to-last row
    for (int i=0; i < matrix.size()-1; i++) {

```

```

        //checks if the ith and i+1th rows have the same # of elements, TRUE(1) if not equal, FALSE(0)
        otherwise
        if(matrix[i].size() != matrix[i+1].size()) {

```

```

            //returns FALSE(0) to function call when two rows are not equal, indicates that the 2D vector is
            not a valid matrix
            return false;
        }
    }
}

```

```

//returns TRUE(1) to function call if the input vector passes all tests, indicates that the 2D vector is a

```

```

valid matrix
    return true;
};

```

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

matrixSize.cpp

```

#include <vector>

```

```

bool matrixSize(vector<vector<int> > mat1, vector<vector<int> > mat2, int op){
    if (op == 1){//Addition
        //Matrices 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 .csv file to read
// matrix: a reference to the matrix that the function will fill with the contents of the file
// 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

```

#include <vector>
#include <iostream>

```

```

using namespace std;

```



```

//takes in a 2D vector of ints by reference
void transpose(vector<vector<int> > &matrix) {

    int Aheight = matrix.size();
    int Awidth = matrix[0].size();

    //creates 2D vector of transposed dimensions
    vector< vector<int> > transposed (Awidth , vector<int> (Aheight));

    //fills in new matrix with the transposed values
    for(int i=0 ;i<Aheight ;i++) {
        for(int j=0 ;j<Awidth ;j++) {
            transposed[j][i] = matrix[i][j];
        }
    }

    //resizes original matrix height to transposed size (original width)
    matrix.resize(Awidth);

    //resizes # of elements in ith row of original matrix to transposed size (original height)
    for(int i=0; i<Awidth; i++){
        matrix[i].resize(Aheight);
    }

    //copies values from transposed matrix to the original
    for(int i=0 ;i<Awidth ;i++) {
        for(int j=0 ;j<Aheight ;j++) {
            matrix[i][j] = transposed[i][j];
        }
    }
}

```

GitHub

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

https://github.com/jmkennedy/448_Lab6

You don't have any verified emails. We recommend [verifying](#) at least one email.

Email verification helps our support team verify ownership if you lose account access and allows you to receive all the notifications you ask for.

jmkennedy / 448_Lab6

Unwatch 3Star 0Fork 0

24 commits1 branch0 releases2 contributors

branch: master448_Lab6 / +

Updated files to use my transpose and checkmatrix files. Added .csv f... Joshua Jenson authored 4 minutes ago latest commit 4d68785f7d

Makefile	Added matrixTrans.cpp to the makefile	6 hours ago
README.md	Initial commit	20 days ago
Report.docx	Added the report	a day ago
Test	Updated files to use my transpose and checkmatrix files. Added .csv f...	4 minutes ago
add1.csv	Updated files to use my transpose and checkmatrix files. Added .csv f...	4 minutes ago
add2.csv	Updated files to use my transpose and checkmatrix files. Added .csv f...	4 minutes ago
checkMatrix.cpp	Updated files to use my transpose and checkmatrix files. Added .csv f...	4 minutes ago
main.cpp	Updated files to use my transpose and checkmatrix files. Added .csv f...	4 minutes ago
main.o	Updated files to use my transpose and checkmatrix files. Added .csv f...	4 minutes ago
matrixAdd.cpp	Added documentation	6 days ago
matrixMult.cpp	Fixed minor errors	6 days ago
matrixSize.cpp	Changed conflicting function name and fixed size() calls	a day ago
matrixTrans.cpp	Updated files to use my transpose and checkmatrix files. Added .csv f...	4 minutes ago
mult1.csv	Updated files to use my transpose and checkmatrix files. Added .csv f...	4 minutes ago
mult2.csv	Updated files to use my transpose and checkmatrix files. Added .csv f...	4 minutes ago
reader.cpp	Fixed seg fault	7 days ago
trans1.csv	Updated files to use my transpose and checkmatrix files. Added .csv f...	4 minutes ago

README.md

448_Lab6

Code

Issues 0

Pull requests 0

Wiki

Pulse

Graphs

HTTPS clone URL

https://github.com/jmkennedy/448_Lab6

You can clone with [HTTPS](#), [SSH](#), or [Subversion](#).

Download ZIP

Program Output

```
[jjenson@1005-17 448_Lab6]$ ./Test
-----Adding matrixes-----
Matrix 1:
1 2 3
4 5 6
7 8 9
Added with Matrix 2:
90 80 70
60 50 40
30 20 10
Equals matrix 3:
91 82 73
64 55 46
37 28 19
-----Multiplying matrixes-----
Matrix 1:
1 2 3
2 4 6
Multiplied with Matrix 2:
1 9 5 3
0 4 2 7
2 8 6 10
Equals matrix 3:
7 41 27 47
14 82 54 94
-----Transposing matrix-----
Matrix 1:
1 2 3
2 4 6
The Transpose of matrix 1 is:
1 2
2 4
3 6
[jjenson@1005-17 448_Lab6]$ █
```