

Matrix Multiplication

Contents

1	Introduction.....	2
2	Problem Analysis.....	3
	2.1 Problem Statement.....	3
	2.2 Solution Design.....	3
3	Implementation and Results.....	4
	3.1 Code Structure.....	4
	3.2 Results.....	5
4	Conclusion	7
	Reference List.....	8

1 Introduction

This document presents a matrix multiplication program in C++ that takes as input from the keyboard: the size and values of each matrix being multiplied, and prints out the result. The program also stores the matrices values on a file.

This documentation consists in two parts:

- Problem Analysis covers the examination of the problem and the designing of the solution.
- Implementation and Results: gives an overview of the code and results obtained.

2 Problem Analysis

This chapter describes the first stages of development of the Matrix Multiplication program:
analysis of the problem and solution design.

2.1 Problem Statement

Write a C++ program for matrix multiplication with following specifications:

- 1) Use constructor dynamic memory allocation for matrix.
- 2) Use `getdata()` function to input values for matrix.
- 3) Use `show()` to display the matrix.
- 4) Use `operator*()` for matrix multiplication.
- 5) To perform the write operation to a file in addition to screen.

2.2 Solution Design

The goal is to implement a matrix product operation. Hence, it was designed a simple program that accepts only integers as matrices values. In order to evaluate the operation between matrix A and B correctly, the program should consider the number of columns in A equal to the number of rows in B in :



Each value across the m row in A are multiplied one by one with each element across a column p and sum up together. In the end, the product is a matrix $AB[n][p]$.

3 Implementation and Results

A brief description of the code implementation for the matrix multiplication is presented in this chapter.

3.1 Code Structure

The overview of the calculator code, without any implementation details is:

1) Matrix.h:

Header file deals with the declarations needed to use the Matrix class.

```
class Matrix {
private:
    int    rowSize, columnSize;
public:
    Matrix() { ptrArray = 0; rowSize = 0; columnSize = 0; } Matrix(int rows,
int columns);
    Matrix(const Matrix&);
    friend Matrix operator*(const Matrix &m1, const Matrix& m2);
    friend ostream& operator<<(ostream& os, Matrix m);
    void getData(); //set values in the matrix from keyboard int getRowSize()
    const { return rowSize; };
    int getColumnSize() const { return columnSize; };
    ~Matrix();
};
```

2) Matrix.cpp:

Source file contains the definitions of the Matrix class.

```
#include "Matrix.h"
Matrix::Matrix(int rows, int columns) : rowSize(rows), columnSize(columns){
    ptrArray = new int[rows*columns];
}
Matrix::Matrix(const Matrix& m) : rowSize(m.rowSize), columnSize(m.columnSize){...}

Matrix operator*(const Matrix &a, const Matrix& b) {...}

ostream& operator<<(ostream& os, Matrix m) {...}

void Matrix::getData() {...}

Matrix::~Matrix() {delete[] ptrArray;}
```

3) main.cpp:

This is the testing file that implements the uses of the functions defined previously.

```
#include "Matrix.h"

void writeFile(ofstream& file, Matrix& mtx) {}

int main() {
    string nameFile;
```

```

int m,n,p = 0; //a[m][n]      b[n][p]      ab[m][p]

cout << "          Matrix Multiplication\n" << endl;
cout << "-----\n" << endl;
cout << "Enter file name to store the result (e.g. one.txt):" << endl; //
cin >> nameFile;
ofstream myfile(nameFile);

//-----Prompt Matrix 1-----

cout << "\nEnter the number of rows and columns for the Matrix 1:" << endl;
cin >> m >> n;
Matrix m1(m,n);
cout << "Enter the values of Matrix[" << m << "][" << n << "]:" << endl;
m1.getData();
cout << "\n Matrix 1: \n" << m1 << endl;
writeFile(myfile, m1);

//-----Prompt Matrix 2-----
cout << "\nEnter the number of columns
of Matrix 2: " << endl;
cin >> p;
Matrix m2(n,p);
cout << "\nEnter the values of Matrix[" << n << "][" << p << "]:" << endl;
m2.getData();
cout << "\n Matrix 2: \n" << m2 << endl;
writeFile(myfile, m2);

//-----Result -----
cout << "\nThe result is matrix[" << m <<
"][" << p << "]: " << endl;
cout << m1*m2 << endl;
writeFile(myfile,m1*m2);

//-----
myfile.close();
cin.clear();
cin.ignore();
cin.get(); return 0;
}

```

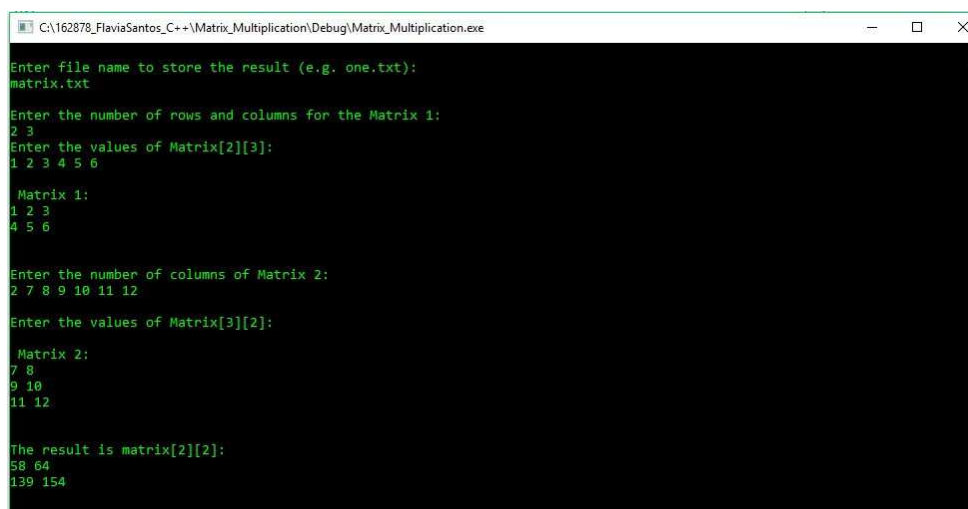
3.2 Results

The user interaction with the program is by using the computer keyboard and its screen, it is a very simple solution implemented with cin and cout from C++'s standard library.

When the program starts running the user is asked to enter the name of the file to store the data. In addition, the user is asked to enter: number of rows, number of columns and the values for each matrix.

After that, the program created performs the calculation and prints out the result on the console window.

One example of the calculation is displayed in Figure 3-2:



```
C:\162878_FlaviaSantos_C++\Matrix_Multiplication\Debug\Matrix_Multiplication.exe

Enter file name to store the result (e.g. one.txt):
matrix.txt

Enter the number of rows and columns for the Matrix 1:
2 3
Enter the values of Matrix[2][3]:
1 2 3 4 5 6

Matrix 1:
1 2 3
4 5 6

Enter the number of columns of Matrix 2:
2 7 8 9 10 11 12
Enter the values of Matrix[3][2]:

Matrix 2:
7 8
9 10
11 12

The result is matrix[2][2]:
58 64
139 154
```

Figure 3-2: Results

The file generated after the operation is described on the Figure 3-2.1:



```
matrix.txt - Notepad
File Edit Format View Help

-----
1 2 3
4 5 6
-----

7 8
9 10
11 12
-----

58 64
139 154
|
```

Figure 3-2.1: Text File

4 Conclusion

The Matrix Multiplication program works perfectly solving the operation with the two matrices typed by the user and prints out the desired result on the screen, in addition to store the values typed on a file.

Reference List

- [1] B. Stroustrup, Programming Principles and Practice Using C++. Addison-Wesley Professional, 2014.