|  |
| --- |
| Linear Algebra Operations  Using C++ |
|  |
| February 28  Authored by: Abhik Kumar Dey |

# Document Control

## Document Revision

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Change Status |
| 1.0 | 28 February 2020 | Abhik Kumar Dey | Initial Draft |
|  |  |  |  |
|  |  |  |  |

## Distribution

|  |  |  |
| --- | --- | --- |
| Name | Title | Review/Approval |
|  |  |  |
|  |  |  |
|  |  |  |

Contents

[Introduction 4](#_Toc33825606)

[System Requirements 4](#_Toc33825607)

[Hardware Requirements 4](#_Toc33825608)

[Software Requirements 4](#_Toc33825609)

[Technical Specifications 5](#_Toc33825610)

[Class LinearAlgebraOperations 5](#_Toc33825611)

[Private Members 5](#_Toc33825612)

[Public Members 8](#_Toc33825613)

[Compile and Execution 9](#_Toc33825614)

# Introduction

The goal is to implement a high-performance, portable linear algebra library. As a start this library will perform two matrix operations:

1. Transpose of a Matrix
2. Matrix Multiplication

# System Requirements

## Hardware Requirements

* Operating System – Windows (Vista and higher), Linux or Mac
* Minimum RAM Size – 4 MB
* Minimum Hard Drive Space – 20 MB

## Software Requirements

* Language – C++
* IDE –
* Windows: Any IDE (Visual Studio, Borland C++, Turbo C++, etc.)
* MAC and Linux: Text Editor and Terminal

# Technical Specifications

The code implements the below Linear Algebra Operations:

1. Matrix Transpose
2. Matrix Multiplication

## Class LinearAlgebraOperations

The class **LinearAlgebraOperations** is defined to perform the different Linear Algebra Related Operations. As of now it performs Matrix Transpose and Matrix Multiplications. Below described are the member functions of the class.

### Private Members

1. **createMatrix** – Takes Rows and Columns as parameter and creates a Matrix by taking input from user.

* **Syntax** – float\*\* createMatrix(int row, int column)
* **Parameter** – row -> Integer

column -> Integer

* **Return –** Reference to the 2D Matrix created. Pointer to pointers of type float
* **Function Calling –**

float \*\* matrix;

LinearAlgebraOperations mo;

matrix = mo.createMatrix(row, cols);

* **Validations –** Matrix values cannot be characters. It must be integer or decimal value. This is handled by **validateMatrixInput** function.

1. **validateMatrixInput -** Takes Row and Column as parameter and validates the input given by the user to create the matrix

* **Syntax** – float validateMatrixInput(int row, int column);
* **Parameter** – row -> Integer

column -> Integer

* **Return –** Correct user input.
* **Function Calling –**

mat[r][c] = validateMatrixInput(r+1, c+1);

* **Validations –** No character is allowed as input to the matrix

1. **printMatrix –** Displays the matrix to the user. Takes matrix, rows and columns as parameters.

* **Syntax** – void printMatrix(float \*\* matrix, int row, int column);
* **Parameter** – matrix -> Reference to the 2D Matrix. Pointer to pointers of

type float

row -> Integer

column -> Integer

* **Return –** Displays the matrix in the output
* **Function Calling –**

LinearAlgebraOperations mo;

matrix = mo.createMatrix(rows, columns);

cout << "Matrix:\n";

cout << "--------------------" << "\n";

mo.printMatrix(matrix, rows, columns);

* **Validations –** No validations required.

1. **transposeMatrix –** Transposes a Matrix**.** Takes matrix, rows and columns as parameters.

* **Syntax** – float\*\* **transposeMatrix** (float\*\* matrix, int row, int column)
* **Parameter** – matrix -> Reference to the 2D Matrix. Pointer to pointers of

type float

row -> Integer

column -> Integer

* **Return –** Reference to the transposed matrix, pointer to pointers of type float
* **Function Calling –**

float\*\* transposeMatrix;

float\*\* matrix;

LinearAlgebraOperations mo;

matrix = mo.createMatrix(rows, cols);

transposeMatrix = mo.transposeMatrix(matrix, rows, cols);

* **Validations –** No validations required.

1. **matrixMultiplication –** Multiplies two Matrices**.** Takes matrix1, matrix2, rows and columns as parameters.

* **Syntax** – float\*\* matrixMultiplication(float \*\* matrix1, float \*\* matrix2, int row1, int column1, int row2, int column2)
* **Parameter** – matrix1 -> Reference to the 2D Matrix. Pointer to pointers of

type float

matrix2 -> Reference to the 2D Matrix. Pointer to pointers of

type float

row1 -> Integer, Row of Matrix1

column1 -> Integer, Column of Matrix1

row2 -> Integer, Row of Matrix2

column2 -> Integer, Column of Matrix2

* **Return –** Reference to the Multiplied result matrix, pointer to pointers of type float
* **Function Calling –**

float\*\* matrix1;

float\*\* matrix2;

float\*\* resultMatrix;

LinearAlgebraOperations mo;

matrix1 = mo.createMatrix(row1, col1);

matrix2 = mo.createMatrix(row2, col2);

resultMatrix = mo.matrixMultiplication(matrix1, matrix2, row1, col1, row2, col2);

* **Validations –** Column of Matrix 1 should be equal to row of Matrix2.

### Public Members

To access the private members of class LinearAlgebraOperations, public member functions have been defined.

1. performLinearAlgOperation: This function is a public friend function, defined to access the private linear algebra operations. The function takes choice from user and calls the respective private functions of the class LinearAlgebraOperations.

* **Syntax** – friend void performLinearAlgOperation(LinearAlgebraOperations mo, int choice );
* **Parameter** – mo -> Object of class LinearAlgebraOperations

choice -> Integer

* **Return –** None
* **Function Calling –**

Int choice;

LinearAlgebraOperations mo;

performLinearAlgOperation (mo, choice);

* **Validations –** Choice should be of type integer only. Validated by **validateInput** function

1. validateInput: Validates the inputs given by user like choice, matrix rows and columns. These inputs will be only integer type and cannot be character or floating type and this function is used to validate the same.

* **Syntax** – int validateInput(string inputType);
* **Parameter** – inputType -> String. Values: “choice”, “rows”, “columns”
* **Return –** Input given by the user.
* **Function Calling –**

int option;

LinearAlgebraOperations mo;

option = matr.validateInput("choice");

* **Validations –** Choice should be of type integer only. Characters or decimal values not allowed

# Compile and Execution

**Linux**:

1. Download the zip and extract the folder DEY\_ABHIK
2. Open terminal and navigate to the folder DEY\_ABHIK
3. Run the below command in the terminal to compile the file

**g++ -o main dey\_abhik.cpp**

A file with the name “main” will be generated in the folder

1. Run the below command in the terminal to execute the file

./main

**Mac**:

1. Download the zip and extract the folder DEY\_ABHIK
2. Open terminal and navigate to the folder DEY\_ABHIK
3. Run the below command in the terminal to compile the file

**g++ -o main dey\_abhik.cpp -std=c++11**

A file with the name “main” will be generated in the folder

1. Run the below command in the terminal to execute the file

./main

**Windows**:

1. Download the zip and extract the folder DEY\_ABHIK
2. Open the file **dey\_abhik.cpp** in a C++ IDE like Microsoft Visual Studio
3. Compile and Run the program according to IDE’s feature.