**Project Main Report**

(Matrix Algebra)

**Problem statement:**

Write an Object Oriented program C++ program for **Matrix Algebra**. The program should have the following features:

* **Create** a Matrix of size NxN (where N is specified )
* **Initialize** an NxN Matrix with random numbers in the range 1-9
* **Print** an NxN Matrix in Tabular form using <<operator
* **Sum** the diagnol elements of an NxN Matrix
* **Transpose** an NxN Matrix using ! operator overloading
* **Add** two matrices using + operator overloading
* **Multiply** two matrices using \* operator overloading

Write an interactive menu-driven program to test the above-mentioned features.

**Objectives:**

Main objectives of this project are undermentioned:

1. To develop a program for Matrix Arithmetic that can perform above mentioned features
2. To understand the concept of Unary and Binary operator overloading and their syntax
3. To understand the concept of stream insertion operators <<
4. To practically apply the concept of classes, overloading and C++ for an interactive program

**Conclusion:**

After completing this project, we are practically aware of the concepts of classes and operator overloading altogether. We have developed an interactive program that can perform Matrix Arithmetic. I have cleared many queries about the concept of operator overloading and its syntax. Thus it concludes that operator overloading is a very important concept in practical/professional programming environment.

**Source Code**

#include<iostream>

#include<conio.h>

#include<string>

#include<stdlib.h>

using namespace std;

class Matrix{

private:

int sum, sum1;// Declaring Data members and arrays

int n;

int mat[90][90];

int summ[90][90];

int mat2[90][90];

int prod[90][90];

int trans[90][90];

public:

Matrix()//Constructor

{

n = 0;

sum = 0;

sum1 = 0;

}

void getn()

{

cout << "Enter the number of rows and coloumn you want in the Matrix: " << endl;

cin >> n;

}

int get()

{

return n;

}

int msum(int a,int b)

{

return summ[a][b];

}

int mprod(int y,int z)

{

return prod[y][z];

}

int mtrans(int o,int s)

{

return trans[o][s];

}

void createMatrix();

void displayMatrix();

void displayMatrixsum();

int sumleftdiagnolMatrix();

int sumrightdiagnolMatrix();

Matrix operator + (const Matrix &c)//It will overload + operator

{

Matrix new\_mat;

for (int i = 0; i<n; i++){

for (int j = 0; j<n; j++){

new\_mat.summ[i][j] = mat[i][j] + c.mat[i][j];

}

}

return new\_mat;

}

Matrix operator\*(const Matrix &d)//It will overload \* operator

{

Matrix new\_mult;

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

{

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

{

int sun=0;

for(int k=0;k<n;k++)

{

sun=sun+mat[i][k]\*d.mat[k][j];

}

new\_mult.prod[i][j]=sun;

}

}

return new\_mult;

}

Matrix operator!()//It will overload ! Operator

{

Matrix new\_trans;

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

{

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

{

new\_trans.trans[i][j]=mat[j][i];

}

}

return new\_trans;

}

friend ostream& operator<<(ostream& output,Matrix& D)//It will overload << Operator

{

for (int i = 0; i<D.get(); i++)

{

for (int j = 0; j<D.get(); j++)

{

output << "\t" << D.mat[i][j] << "\t";

}

cout << "\n\n";

}

return output;

}

};

void Matrix::createMatrix()//Create Matrix

{

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

{

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

{

mat[i][j] = (rand() % 9) + 1;

}

}

}

int Matrix::sumleftdiagnolMatrix()//Sum left Diagonol of Matrix

{

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

{

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

{

if (i == j)

{

sum = sum + mat[i][j];

}

}

}

return sum;

}

int Matrix::sumrightdiagnolMatrix()//Sum right Diagonol of Matrix

{

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

{

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

{

if (i + j == n - 1)

{

sum1 = sum1 + mat[i][j];

}

}

}

return sum1;

}

int main()

{

system("color A2");//Change output screen colour to parrott green

Matrix M, M1, M2;//It will initialize 3 objects.

int choice;

G: cout<<"\t\t\*\*\*\*\tWelcome To Matrix Arithmatic\t\t\*\*\*\*\n\t\t\*\*\*\*\t\t\t\t\t\t\*\*\*\*\n\t\t\*\*\*\*\tPress 1 to Create your Matrix\t\t\*\*\*\*\n\t\t\*\*\*\*\tPress 2 to sum Left Diagnol and Right Diagonal\n\t\t\*\*\*\*\n\t\t\*\*\*\*\t\t\t\t\t\t\*\*\*\*\n\t\t\*\*\*\*\tPress 3 to Add Two Matrices\t\t\*\*\*\*\n\t\t\*\*\*\*\tPress 4 to Get Transpose of Matrix\t\*\*\*\*\n\t\t\*\*\*\*\tPress 5 to Multiply Two Matrices\t\*\*\*\*\n\n\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n\t\t\t\*\*\*\*\*Press 8 to Exit\*\*\*\*\*\n";//menu

cin>>choice;

do{

switch(choice)

{

case 1://To create Matrix

{

M.getn();

M.createMatrix();

cout<<"Matrix\n";

cout<<M;

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

goto G;

}

case 2://To Sum Diagnol elements both left and right

{

M.getn();

M.createMatrix();

cout<<M;

cout << "Sum of left Diagnol of Matrix is: " << M.sumleftdiagnolMatrix();

cout<<endl;

cout << "Sum of right Diagnol of Matrix is: " << M.sumrightdiagnolMatrix();

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

goto G;

}

case 3://To add two matrices

{

cout<<"For First Matrix: \n";

M.getn();

M.createMatrix();

cout<<"For Second Matrix: \n";

M1.getn();

M1.createMatrix();

cout<<"Matrix 1:\n"; cout<<M; cout<<endl;

cout<<"Matrix 2:\n"; cout<<M1;

M2 = M + M1;

cout<<" Sum: \n";

for (int i = 0; i < M1.get(); i++)

{

for (int j = 0; j < M1.get(); j++)

{

cout <<"\t"<< M2.msum(i,j)<<"\t";

}

cout<<"\n\n";

}

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

goto G;

}

case 5://To multiply two matrices

{

cout<<"For First Matrix: \n";

M.getn();

M.createMatrix();

cout<<"For Second Matrix: \n";

M1.getn();

M1.createMatrix();

cout<<"Matrix 1:\n"; cout<<M; cout<<endl;

cout<<"Matrix 2:\n"; cout<<M1;

M2=M\*M1;

cout<<" Product: \n";

for (int i = 0; i < M1.get(); i++)

{

for (int j = 0; j < M1.get(); j++)

{

cout <<"\t"<< M2.mprod(i,j)<<"\t";

}

cout<<"\n\n";

}

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

goto G;

}

case 4://To get Transpose of a Matrix

{

M.getn();

M.createMatrix();

cout<<"Matrix: \n";

cout<<M;

cout<<"Transpose: \n";

M2=!M;

for (int i = 0; i < M.get(); i++)

{

for (int j = 0; j < M.get(); j++)

{

cout <<"\t"<< M2.mtrans(i,j)<<"\t";

}

cout<<"\n\n";

}

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

goto G;

}

}

}while(choice!=8);//For Exit Condition

system("pause");

return 0;

}

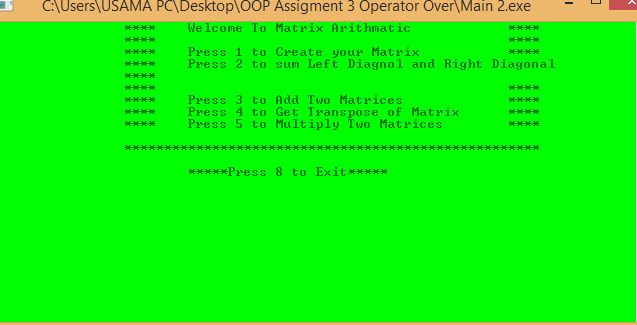
**UML Diagram**

|  |
| --- |
|  |

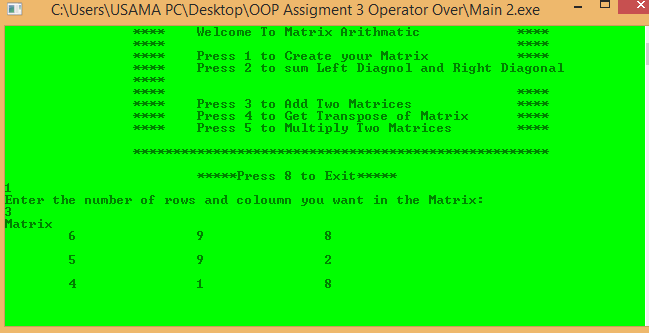
|  |
| --- |
| **Matrix Arithmetic** |
| -int Sum  -int Sum1  -int mat[90][90]  -int sum[90][90]  -int mat2[90][90]  -int prod[90][90]  -int trans[90][90] |
| **+ ~**Matrix  + getn():void  + get():void  + msum():int  + mprod():int  + mtrans():int  + createMatrix():void  + displayMatrixsum():void  + sumleftdiagnolMatrix():int  +sumrightdiagnolMatrix() int  + operator+():Matrix  + operator\*():Matrix  + operator!():Matrix  + friend operator<<():ostream |

**Sample Outputs**

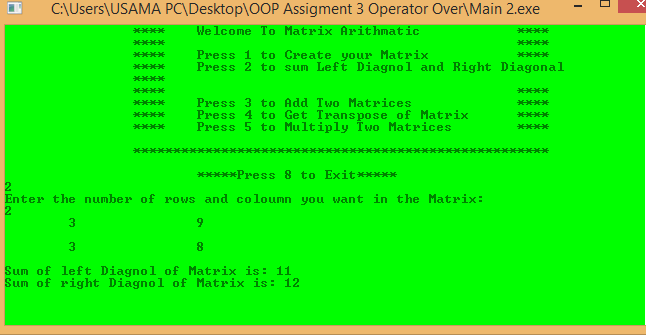
First Interface/Menu/Display:



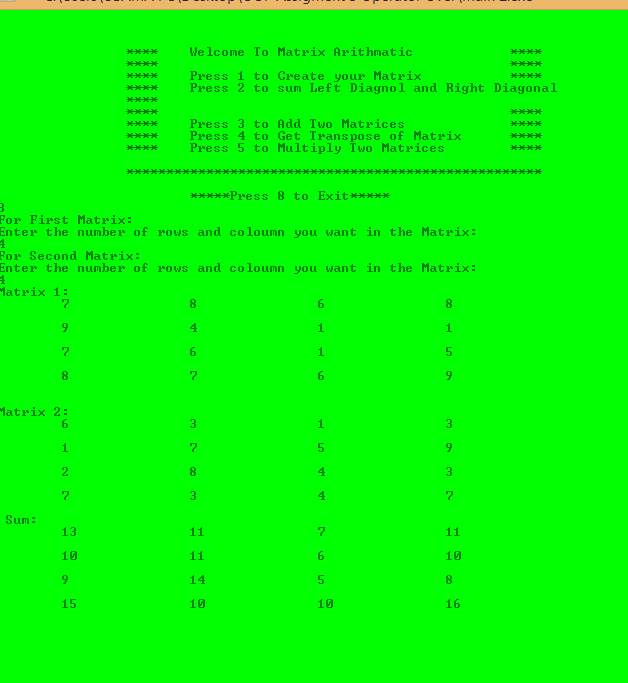
After you Press 1 to create a Matrix:



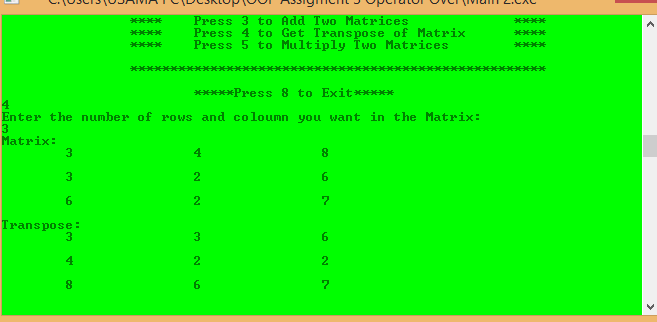
After you Press 2 to get Sum of Left and Right Diagonals Separately:



After you Press 3 to Add Two Matrices:



After you Press 4 to get Transpose of a Matrix:



After you Press 5 to Multiply Two Matrices:

