**Name : Faizan Ali**

**Cms: 462013**

**Sec A**

**Lab Manual 9 LAB+ Home task**

**Task 1 :**

#include <iostream>

using namespace std;

int main() {

const int x = 3;

int mat[x][x];

cout << "please enter elements of 3x3 matrix \n";

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

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

cout << "Enter element for the position [" << i << "][" << j << "]: ";

cin >> mat[i][j];

}

}

cout << "The entered matrix is:\n";

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

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

cout << mat[i][j] << "\t";

}

cout << "\n";

}

int lsum = 0;

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

lsum += mat[i][i];

}

cout << "\nLeft diagonal sum: " << lsum << "\n";

int rsum = 0;

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

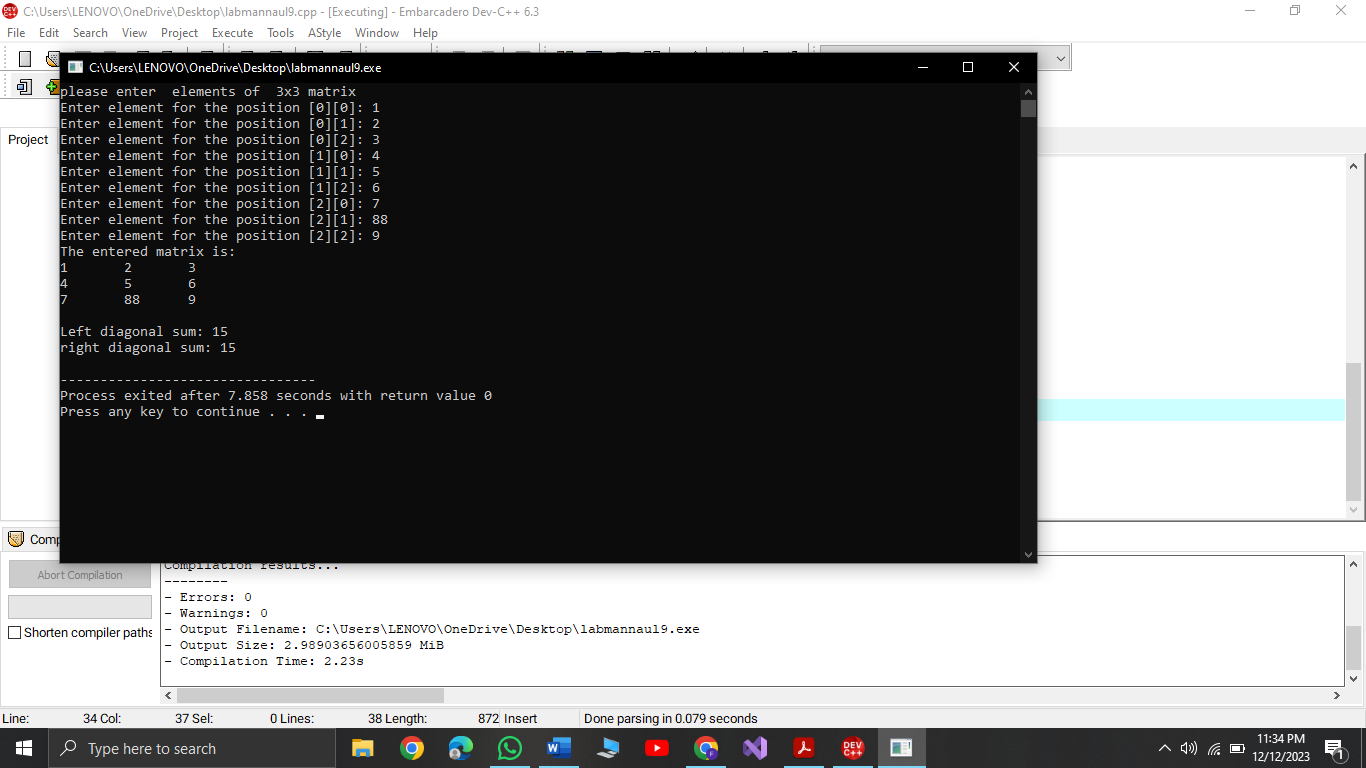
rsum += mat[i][x - 1 - i];

}

cout << "right diagonal sum: " << rsum << "\n";

return 0;

}

****

**Task 2:**

#include <iostream>

using namespace std;

const int x = 3;

void addMatrices(int mat1[][x], int mat2[][x], int result[][x]) {

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

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

result[i][j] = mat1[i][j] + mat2[i][j];

}

}

}

int main() {

int matrix1[x][x];

int matrix2[x][x];

int resultMatrix[x][x];

cout << "Enter the elements for the first 3x3 matrix:\n";

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

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

cout << "Enter element at position [" << i << "][" << j << "]: ";

cin >> matrix1[i][j];

}

}

cout << "Enter the elements of the second 3x3 matrix:\n";

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

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

cout << "Enter element at position [" << i << "][" << j << "]: ";

cin >> matrix2[i][j];

}

}

addMatrices(matrix1, matrix2, resultMatrix);

cout << "The sum of the two matrices is:\n";

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

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

cout << resultMatrix[i][j] << "\t";

}

cout << "\n";

}

return 0;

}

A screenshot of a computer

Description automatically generated

**Task 3**:

#include <iostream>

using namespace std;

void transposeMatrix(int inputMatrix[][3], int transposedMatrix[][3]) {

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

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

transposedMatrix[j][i] = inputMatrix[i][j];

}

}

}

int main() {

int orMat[3][3];

cout << "Enter elements for the 3x3 matrix:\n";

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

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

cout << "Enter element at position [" << i << "][" << j << "]: ";

cin >> orMat[i][j];

}

}

int transposedMatrix[3][3];

transposeMatrix(orMat, transposedMatrix);

cout << "\nThe original matrix is:\n";

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

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

cout << orMat[i][j] << "\t";

}

cout << "\n";

}

cout << "\nThe transposed matrix is:\n";

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

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

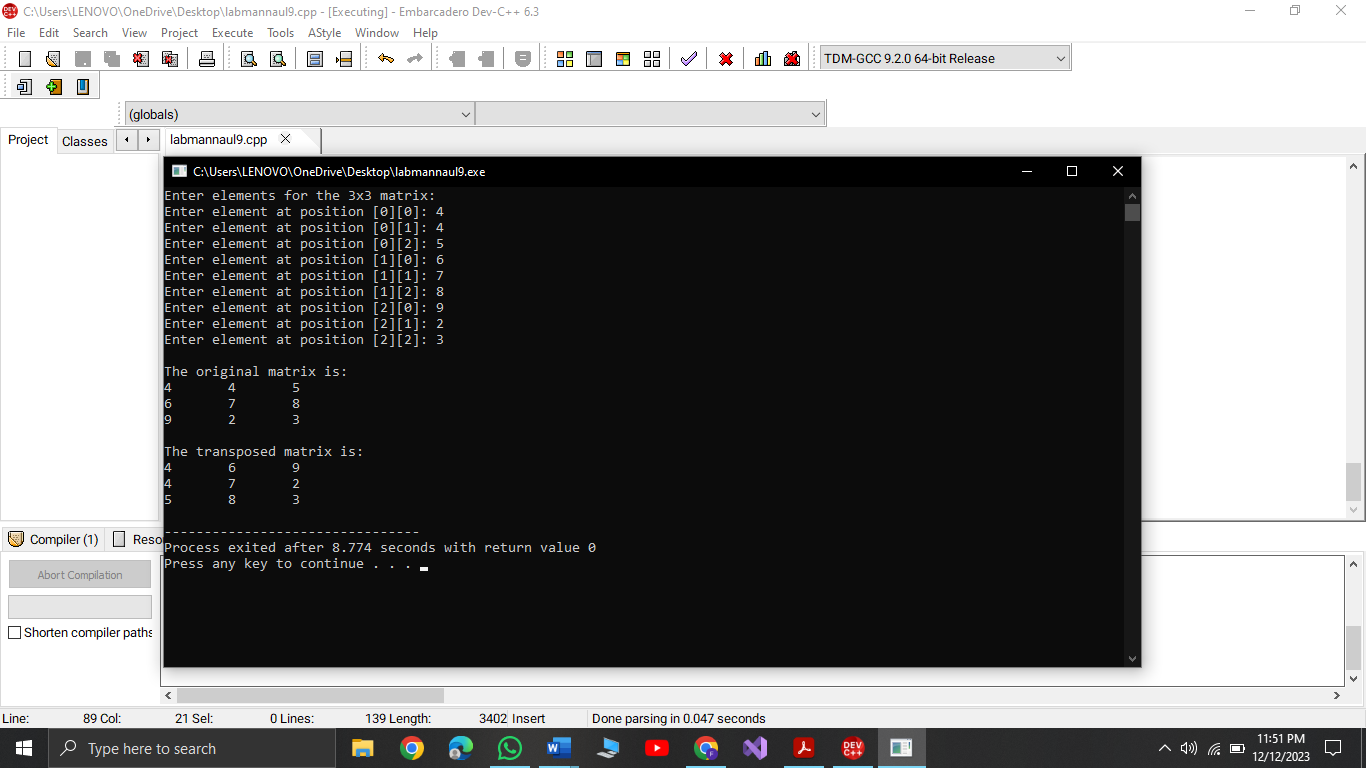
cout << transposedMatrix[i][j] << "\t";

}

cout << "\n";

}

return 0;

}

Task 4:

#include <iostream>

using namespace std;

const int SIZE = 3;

void multiplyMatrices(const int mat1[SIZE][SIZE], const int mat2[SIZE][SIZE], int result[SIZE][SIZE]) {

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

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

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

result[i][j] += mat1[i][k] \* mat2[k][j];

}

void displayMatrix(const int mat[SIZE][SIZE]) {

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

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

cout << mat[i][j] << ' ';

cout << '\n';

}

}

int main() {

const int matrix1[SIZE][SIZE] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};

const int matrix2[SIZE][SIZE] = {{9, 8, 7}, {6, 5, 4}, {3, 2, 1}};

int resultMatrix[SIZE][SIZE] = {0};

multiplyMatrices(matrix1, matrix2, resultMatrix);

cout << "Matrix 1:\n";

displayMatrix(matrix1);

cout << "\nMatrix 2:\n";

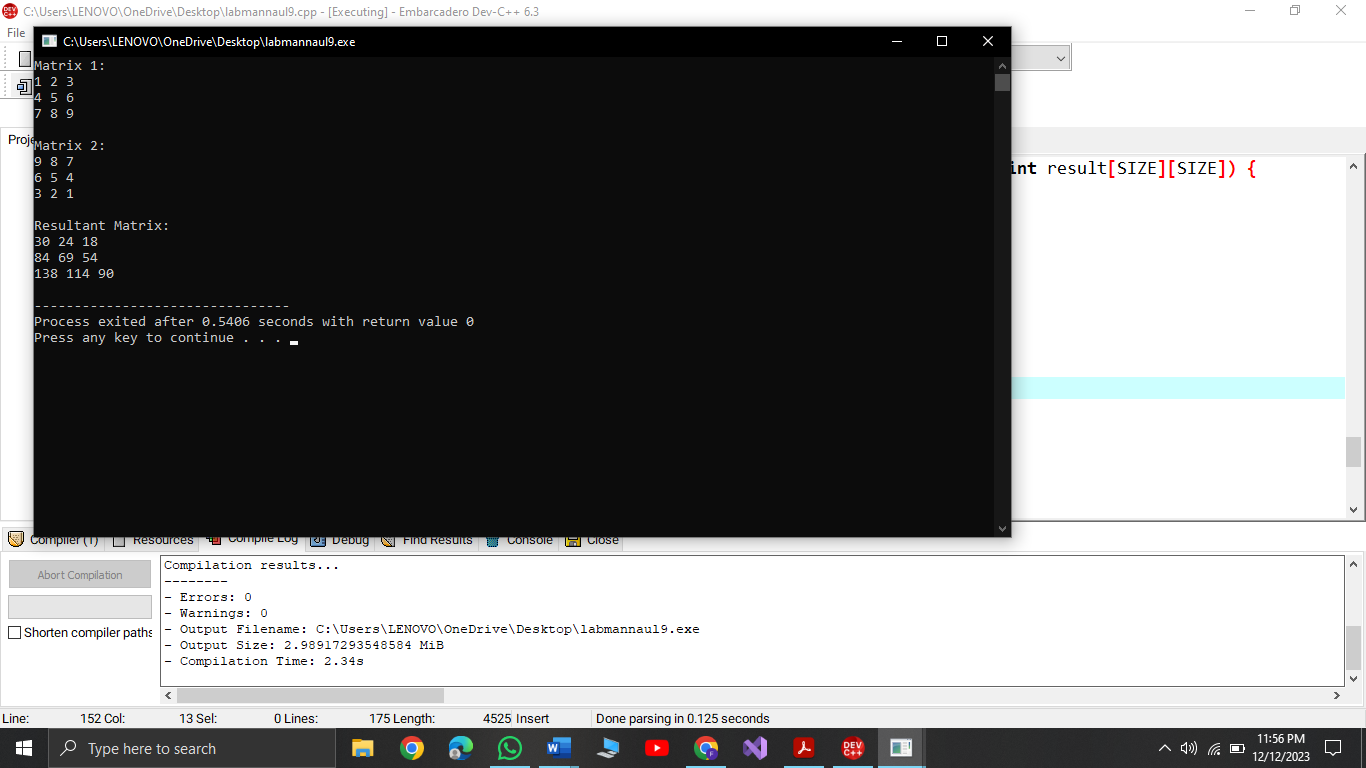
displayMatrix(matrix2);

cout << "\nResultant Matrix:\n";

displayMatrix(resultMatrix);

return 0;

}

****

Task 5:

#include <iostream>

using namespace std;

void printMultiplicationTable(int number, int multiplier = 1) {

if (multiplier <= 10) {

cout << number << " x " << multiplier << " = " << number \* multiplier << '\n';

printMultiplicationTable(number, multiplier + 1);

}

}

int main() {

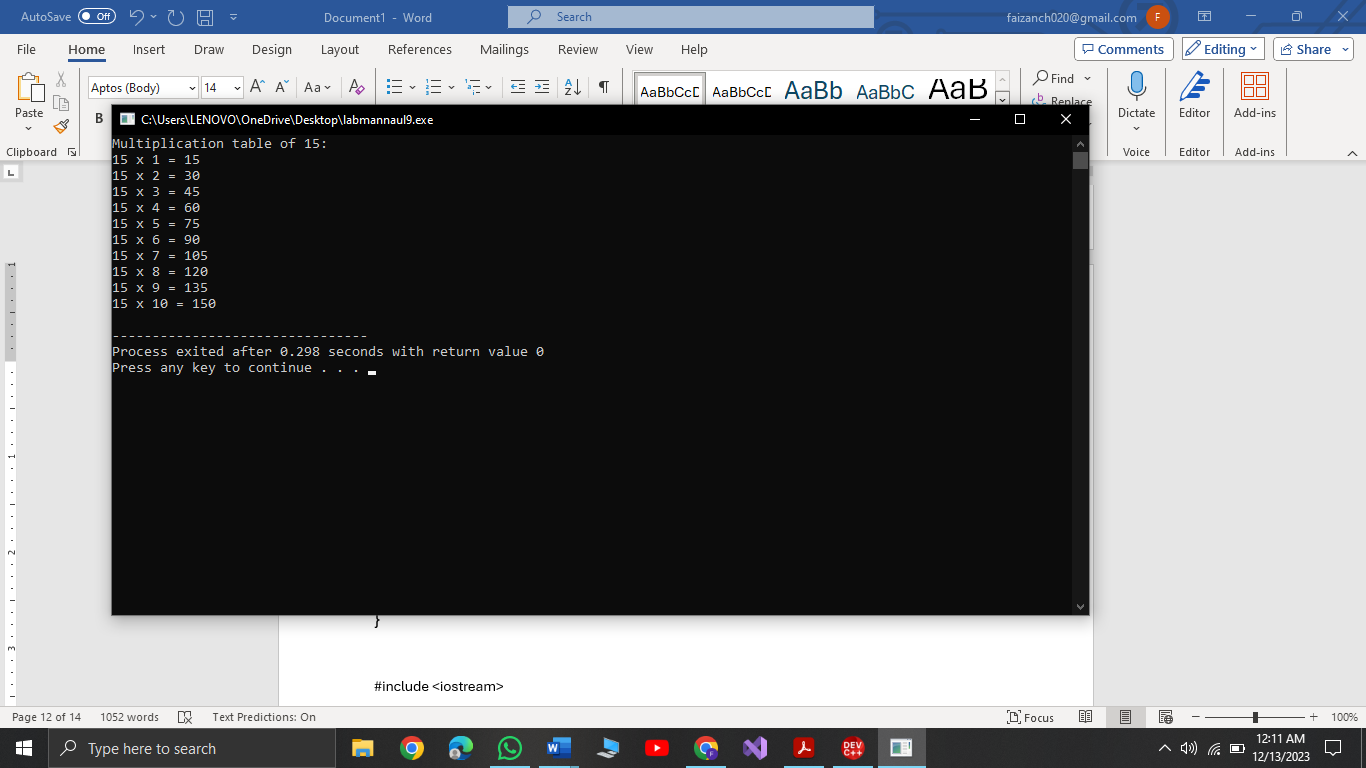
int tableNumber = 15;

cout << "Multiplication table of " << tableNumber << ":\n";

printMultiplicationTable(tableNumber);

return 0;

}



#include <iostream>

int determinant2x2(int a, int b, int c, int d) {

return a \* d - b \* c;

}

int determinant3x3(int matrix[3][3]) {

return matrix[0][0] \* determinant2x2(matrix[1][1], matrix[1][2], matrix[2][1], matrix[2][2]) -

matrix[0][1] \* determinant2x2(matrix[1][0], matrix[1][2], matrix[2][0], matrix[2][2]) +

matrix[0][2] \* determinant2x2(matrix[1][0], matrix[1][1], matrix[2][0], matrix[2][1]);

}

int main() {

int inputMatrix[3][3], outputMatrix[3][3];

std::cout << "Enter elements for the 3x3 matrix:\n";

for (int i = 0; i < 3; ++i)

for (int j = 0; j < 3; ++j)

std::cin >> inputMatrix[i][j];

std::cout << "\nThe original matrix is:\n";

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

for (int j = 0; j < 3; ++j)

std::cout << inputMatrix[i][j] << "\t";

std::cout << "\n";

}

int determinant = determinant3x3(inputMatrix);

if (determinant != 0) {

int adjointMatrix[3][3];

for (int i = 0; i < 3; ++i)

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

int sign = ((i + j) % 2 == 0) ? 1 : -1;

int minorMatrix[2][2] = {

{inputMatrix[(i + 1) % 3][(j + 1) % 3], inputMatrix[(i + 1) % 3][(j + 2) % 3]},

{inputMatrix[(i + 2) % 3][(j + 1) % 3], inputMatrix[(i + 2) % 3][(j + 2) % 3]}

};

adjointMatrix[i][j] = sign \* determinant2x2(minorMatrix[0][0], minorMatrix[0][1],

minorMatrix[1][0], minorMatrix[1][1]);

}

for (int i = 0; i < 3; ++i)

for (int j = 0; j < 3; ++j)

outputMatrix[i][j] = adjointMatrix[j][i] / determinant;

std::cout << "\nThe inverse matrix is:\n";

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

for (int j = 0; j < 3; ++j)

std::cout << outputMatrix[i][j] << "\t";

std::cout << "\n";

}

} else {

std::cout << "The matrix is not invertible (determinant is zero).\n";

}

return 0;

}

A computer screen with a black screen

Description automatically generated