MUHAMMAD MUDASSAR ANWAR

ME-15 SEC-A

CMS#456925

LAB MANUAL 9 LAB & HOME TASK

CS-114

11 NOV 2023

LAB TASKS

TASK 1:

#include <iostream>

using namespace std;

int main() {

const int size = 3;

int matrix[size][size];

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

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

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

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

cin >> matrix[i][j];

}

}

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

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

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

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

}

cout << "\n";

}

int leftDiagonalSum = 0;

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

leftDiagonalSum += matrix[i][i];

}

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

int rightDiagonalSum = 0;

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

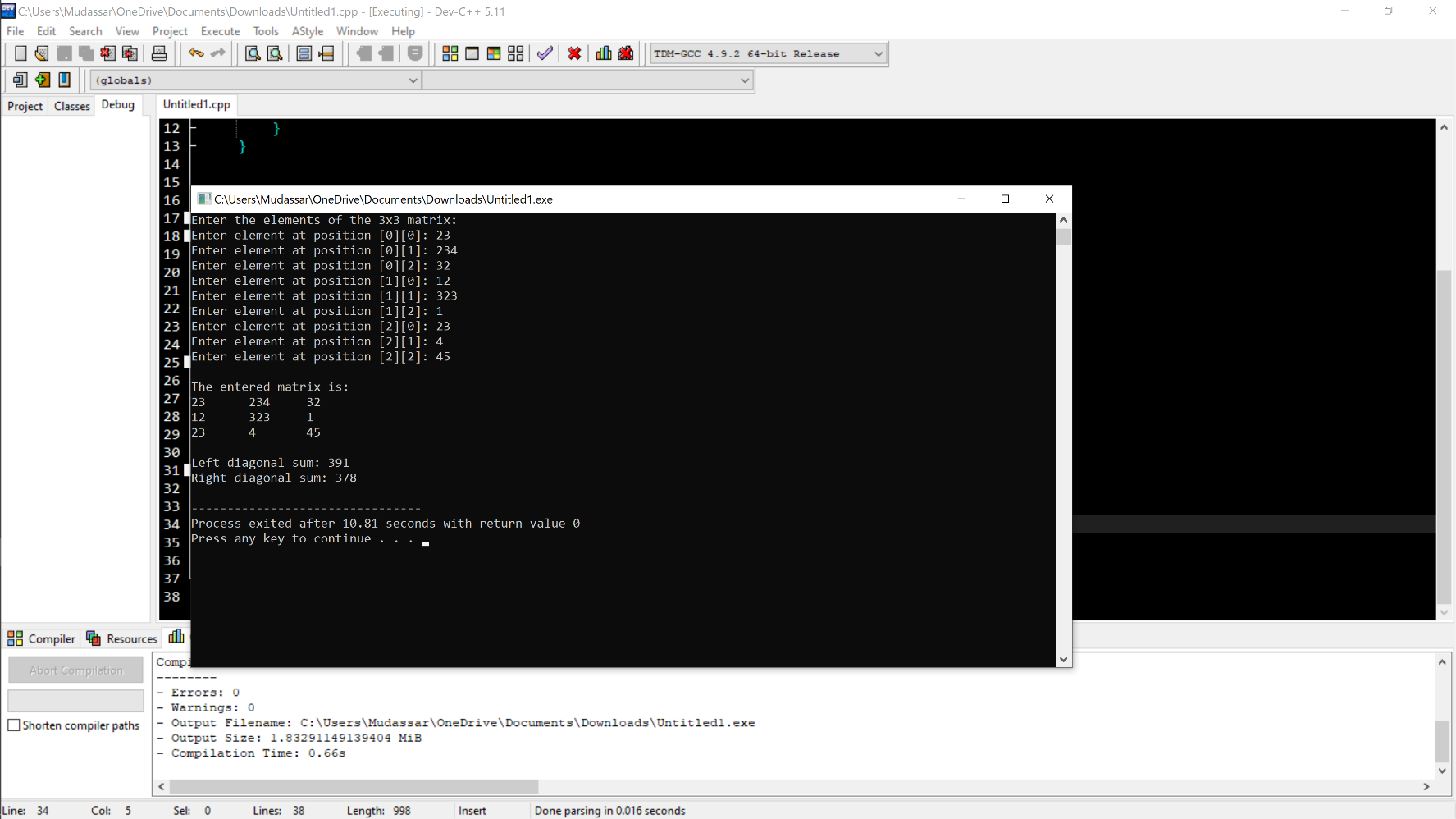
rightDiagonalSum += matrix[i][size - 1 - i];

}

cout << "Right diagonal sum: " << rightDiagonalSum << "\n";

return 0;

}



TASK 2:

#include <iostream>

const int size = 3;

// Function to add two 3x3 matrices

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

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

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

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

}

}

}

int main() {

int matrix1[size][size];

int matrix2[size][size];

int resultMatrix[size][size];

// Input elements for the first matrix

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

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

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

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

std::cin >> matrix1[i][j];

}

}

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

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

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

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

std::cin >> matrix2[i][j];

}

}

addMatrices(matrix1, matrix2, resultMatrix);

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

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

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

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

}

std::cout << "\n";

}

return 0;

}

A screenshot of a computer

Description automatically generated

TASK 3:

#include <iostream>

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

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

}

}

int transposedMatrix[3][3];

transposeMatrix(originalMatrix, transposedMatrix);

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

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

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

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

}

std::cout << "\n";

}

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

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

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

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

}

std::cout << "\n";

}

return 0;

}

A screenshot of a computer

Description automatically generated

TASK 4:

#include <iostream>

// Function to multiply two 3x3 matrices

void multiplyMatrices(int mat1[][3], int mat2[][3], int result[][3]) {

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

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

result[i][j] = 0;

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

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

}

}

}

}

int main() {

// Declare two 3x3 matrices

int matrix1[3][3], matrix2[3][3], resultMatrix[3][3];

// Input elements for the first matrix

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

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

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

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

std::cin >> matrix1[i][j];

}

}

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

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

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

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

std::cin >> matrix2[i][j];

}

}

multiplyMatrices(matrix1, matrix2, resultMatrix);

std::cout << "\nThe result matrix after multiplication is:\n";

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

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

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

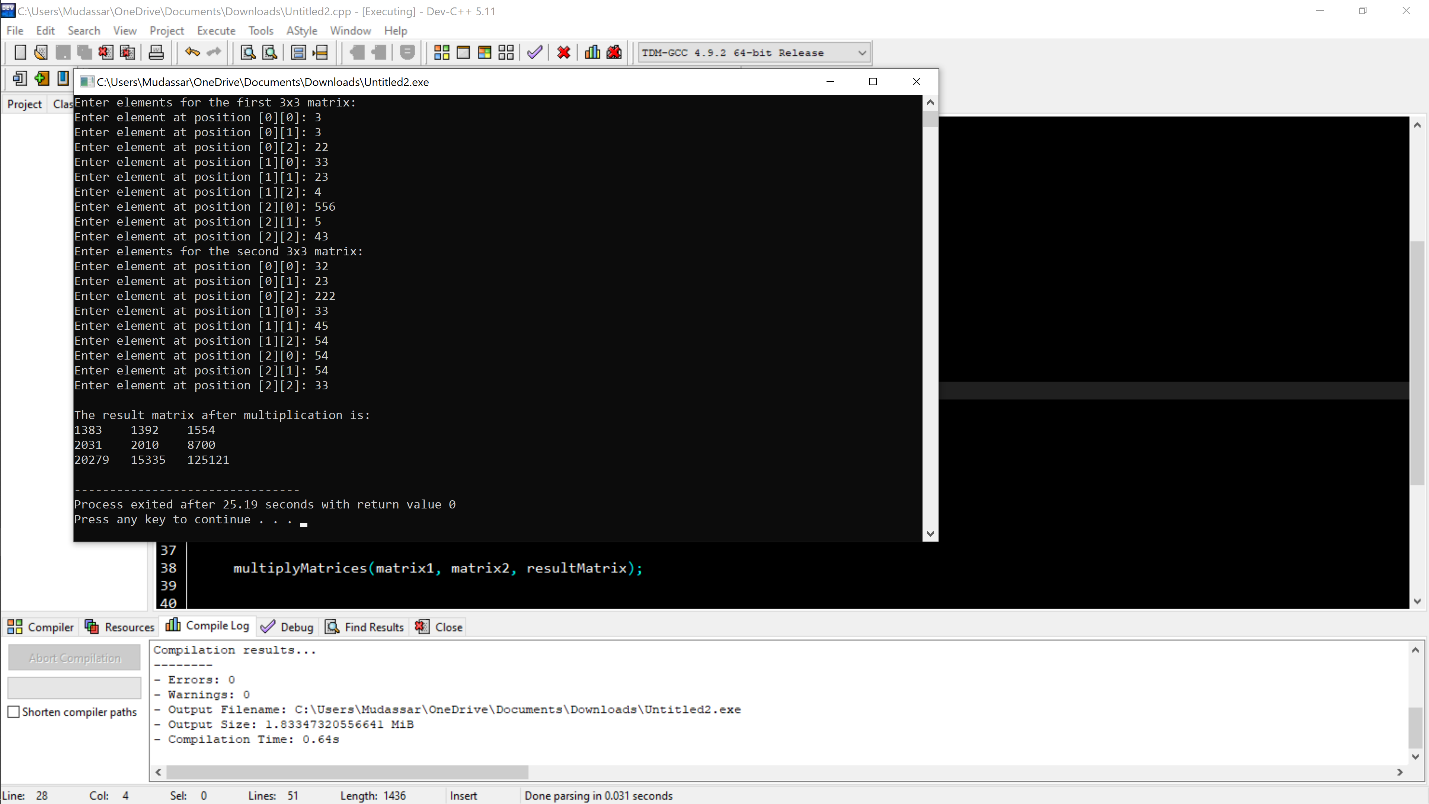
}

std::cout << "\n";

}

return 0;

}



TASK 5:

#include <iostream>

void printTable(int number, int multiplier) {

if (multiplier > 10) {

return; // base case to stop recursion after 10 multipliers

}

std::cout << number << " \* " << multiplier << " = " << number \* multiplier << std::endl;

printTable(number, multiplier + 1);

}

int main() {

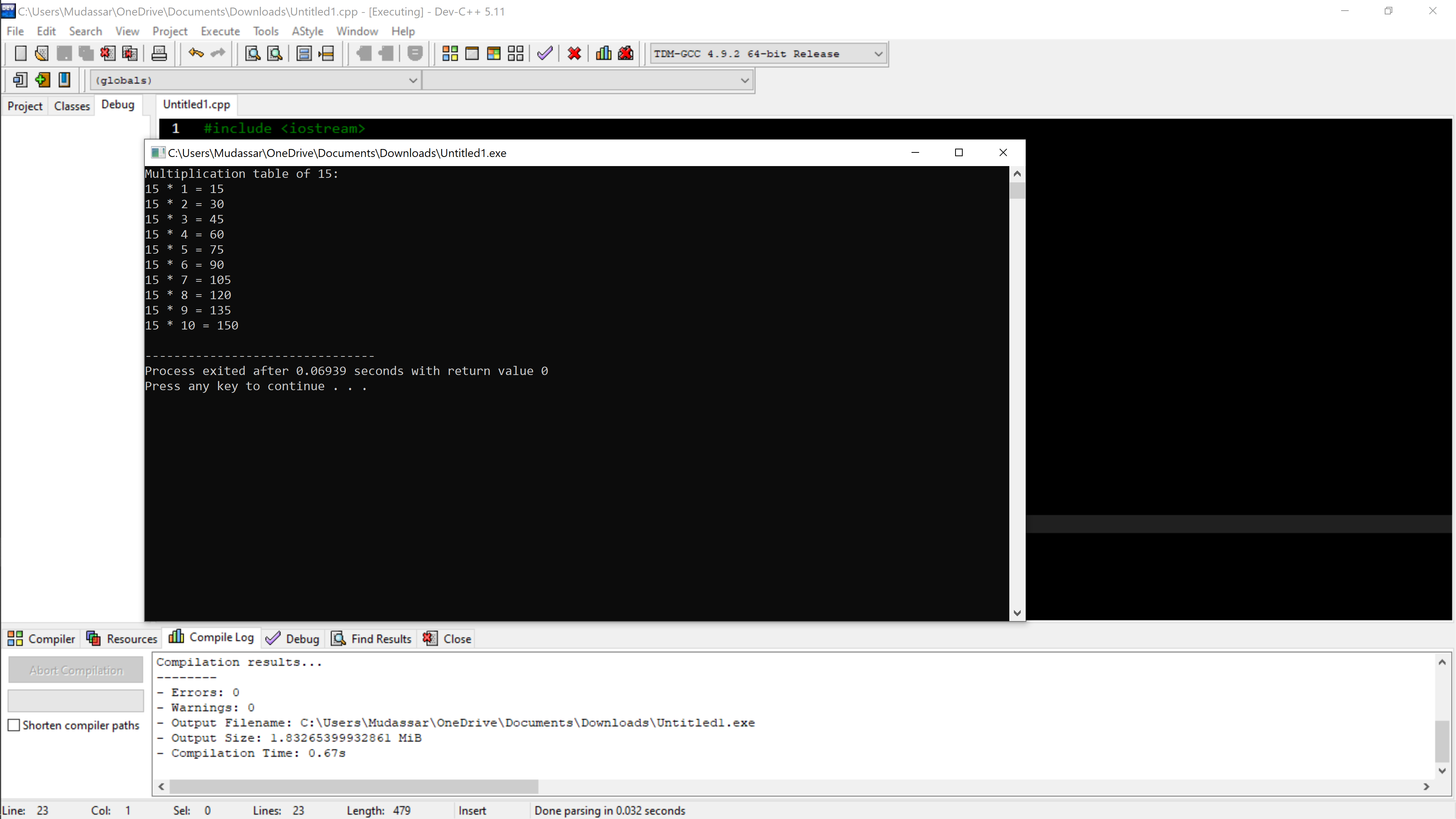
int number = 15;

std::cout << "Multiplication table of " << number << ":\n";

printTable(number, 1);

return 0;

}



{ HOME TASK }

#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 originalMatrix[3][3], inverseMatrix[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 >> originalMatrix[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 << originalMatrix[i][j] << "\t";

std::cout << "\n";

}

int det = determinant3x3(originalMatrix);

if (det != 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] = {

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

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

inverseMatrix[i][j] = adjointMatrix[j][i] / det;

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

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

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

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

std::cout << "\n";

}

} else {

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

}

return 0;

}

A screenshot of a computer

Description automatically generated