

Computer Systems & Programming

Instructor:
Dr. Talha
Shahid



Lab+Home Tasks of Lab Manual 9

SYED FAKHAR ABBAS

ME-15-C

466960

Lab Task# 1:

Make 2D Array in C++ and print left diagonal and right diagonal sum of a 3x3 matrix.

Code:

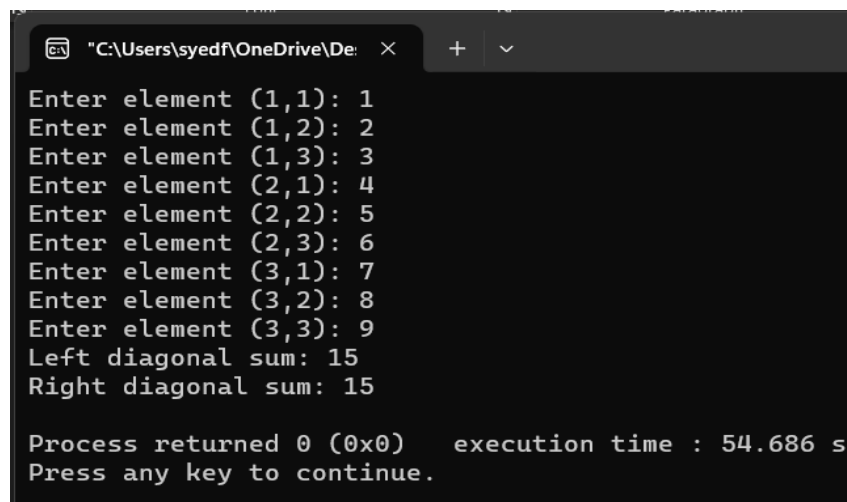
```
#include <iostream>
using namespace std;
const int N = 3;
int main() {
    int matrix[N][N];
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            cout << "Enter element (" << i + 1 << ", " << j + 1 << "): ";
            cin >> matrix[i][j];
        }
    }

    int leftDiagonalSum = 0;
    for (int i = 0; i < N; i++) {
        leftDiagonalSum += matrix[i][i];
    }

    for (int i = 0; i < N; i++) {
        rightDiagonalSum += matrix[i][N - 1 - i];
    }

    cout << "Left diagonal sum: " << leftDiagonalSum << endl;
    cout << "Right diagonal sum: " << rightDiagonalSum << endl;
    return 0;
}
```

Result:



```
"C:\Users\syedf\OneDrive\De  X  +  v
Enter element (1,1): 1
Enter element (1,2): 2
Enter element (1,3): 3
Enter element (2,1): 4
Enter element (2,2): 5
Enter element (2,3): 6
Enter element (3,1): 7
Enter element (3,2): 8
Enter element (3,3): 9
Left diagonal sum: 15
Right diagonal sum: 15

Process returned 0 (0x0)    execution time : 54.686 s
Press any key to continue.
```

Lab Task# 2:

Write a function to add two 2D arrays of size 3x3.

Code:

```
#include <iostream>

using namespace std;

void addMatrices(int matrix1[3][3], int matrix2[3][3], int result[3][3]) {
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            result[i][j] = matrix1[i][j] + matrix2[i][j];
        }
    }
}

int main() {
    int matrix1[3][3] = { {1, 2, 3},
                           {4, 5, 6},
                           {7, 8, 9} };

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

    int result[3][3];

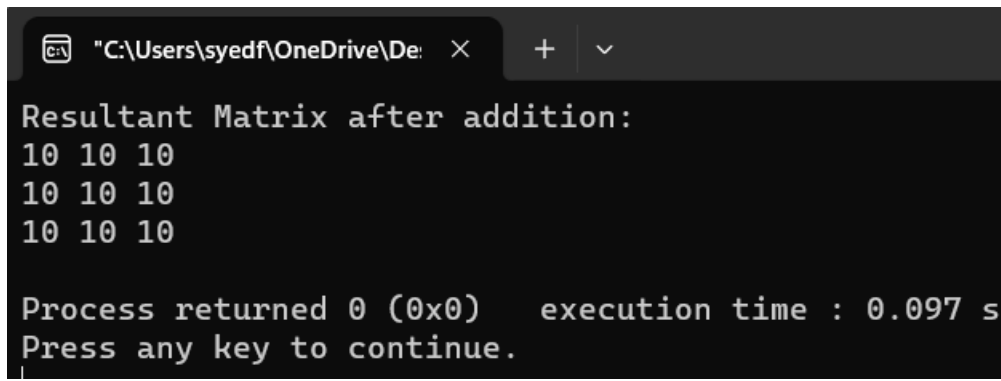
    addMatrices(matrix1, matrix2, result);

    cout << "Resultant Matrix after addition:" << endl;

    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            cout << result[i][j] << " ";
        }
    }
```

```
    cout << endl;
}
return 0;
}
```

Result:



```
"C:\Users\syedf\OneDrive\De:  ×  +  v
Resultant Matrix after addition:
10 10 10
10 10 10
10 10 10

Process returned 0 (0x0)   execution time : 0.097 s
Press any key to continue.
```

Lab Task# 3:

Using 2D arrays in C++, take transpose of a 3x3 matrix. Make a transpose function.

Code:

```
#include<iostream>
using namespace std;

void transpose(int a[3][3]) {
    int trans[3][3];
    for(int i=0; i<3; i++) {
        for(int j=0; j<3; j++) {
            trans[i][j] = a[j][i];
        }
    }
}

cout << "Transpose of the matrix: " << endl;
for(int i=0; i<3; i++) {
```

```

        for(int j=0; j<3; j++) {
            cout << trans[i][j] << " ";
        }
        cout << endl;
    }
}

int main() {
    int a[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
    transpose(a);
    return 0;
}

```

Result:

```

C:\Users\syedf\OneDrive\De
Transpose of the matrix:
1 4 7
2 5 8
3 6 9

Process returned 0 (0x0)   execution time : 0.077 s
Press any key to continue.

```

Lab Task# 4:

Using 2D arrays in C++, implement 3x3 matrix multiplication. Make a function.

Code:

```

#include <iostream>
using namespace std;

void multiplyMatrices(int matrix1[3][3], int matrix2[3][3], int result[3][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] += matrix1[i][k] * matrix2[k][j];
            }
        }
    }
}

```

```

    }
}

int main() {
    int matrix1[3][3] = { { 1, 2, 3},
                          {4, 5, 6},
                          {7, 8, 9} };

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

    int result[3][3];

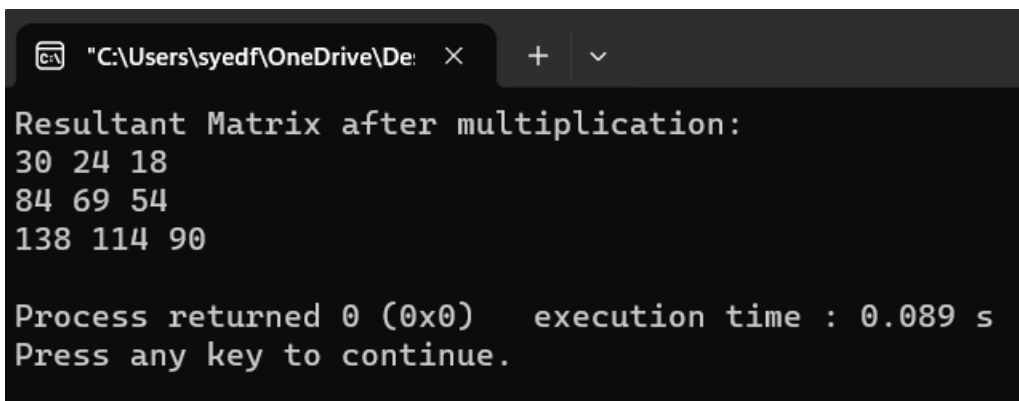
    multiplyMatrices(matrix1, matrix2, result);

    cout << "Resultant Matrix after multiplication:" << endl;
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            cout << result[i][j] << " ";
        }
        cout << endl;
    }

    return 0;
}

```

Result:



```

"C:\Users\syedf\OneDrive\De"
Resultant Matrix after multiplication:
30 24 18
84 69 54
138 114 90

Process returned 0 (0x0)   execution time : 0.089 s
Press any key to continue.

```

Lab Task# 5:

Print the multiplication table of 15 using recursion.

Code:

```
#include <iostream>
using namespace std;

void printTable(int num, int multiplier = 1) {
    if (multiplier <= 10) {
        cout << num << " x " << multiplier << " = " << num * multiplier << endl;
        printTable(num, multiplier + 1);
    }
}

int main() {
    int number = 15;

    cout << "Multiplication table of " << number << ":" << endl;
    printTable(number);

    return 0;
}
```

Result:

```
"C:\Users\syedf\OneDrive\De:  ×  +  ∨

Multiplication table of 15:
15 x 1 = 15
15 x 2 = 30
15 x 3 = 45
15 x 4 = 60
15 x 5 = 75
15 x 6 = 90
15 x 7 = 105
15 x 8 = 120
15 x 9 = 135
15 x 10 = 150

Process returned 0 (0x0)   execution time : 0.083 s
Press any key to continue.
```

Home Task# 1:

Write a C++ program to take inverse of a 3x3 matrix using its determinant and adjoint.

Code:

```
#include <iostream>

using namespace std;

double det(int A[3][3]) {
    return A[0][0] * (A[1][1] * A[2][2] - A[2][1] * A[1][2])
        - A[0][1] * (A[1][0] * A[2][2] - A[2][0] * A[1][2])
        + A[0][2] * (A[1][0] * A[2][1] - A[2][0] * A[1][1]);
}

void adj(int A[3][3], int adjMat[3][3]) {
    adjMat[0][0] = A[1][1] * A[2][2] - A[2][1] * A[1][2];
    adjMat[0][1] = A[0][2] * A[2][1] - A[0][1] * A[2][2];
    ... // remaining assignments for adjMat
}

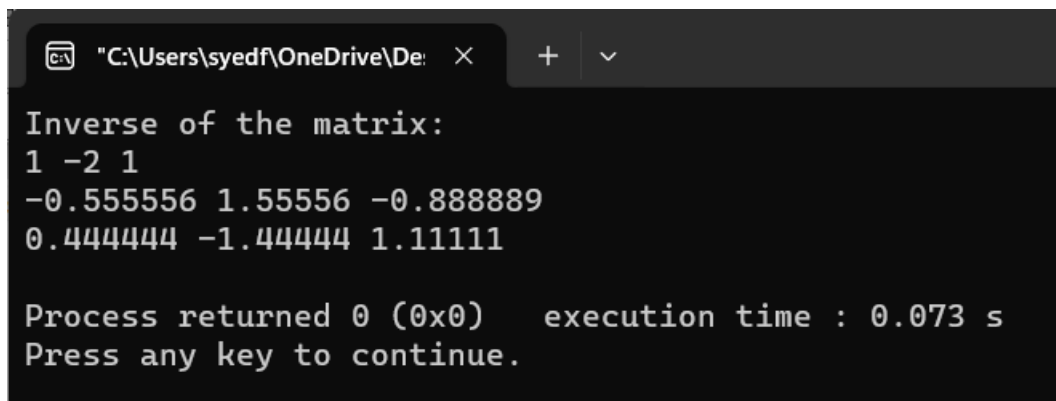
bool inv(int A[3][3], double invMat[3][3]) {
    double d = det(A);
    if (d == 0) {
        cout << "Matrix is singular, no inverse exists." << endl;
        return false;
    }
    int adj[3][3];
    adj(A, adj);
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            invMat[i][j] = adj[i][j] / d;
        }
    }
    return true;
}

int main() {
    int mat[3][3] = {{4, 7, 2}, {2, 6, 3}, {1, 5, 4}};
```



```
double invMat[3][3];
if (inv(mat, invMat)) {
    cout << "Inverse of the matrix:" << endl;
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            cout << invMat[i][j] << " ";
        }
        cout << endl;
    }
}
return 0;
}
```

Output:



```
"C:\Users\syedf\OneDrive\De:  ×  +  v

Inverse of the matrix:
1 -2 1
-0.555556 1.55556 -0.888889
0.444444 -1.44444 1.11111

Process returned 0 (0x0)   execution time : 0.073 s
Press any key to continue.
```