# *National University of Science and Technology*

## School of Mechanical and Manufacturing Engineering

Lab Manual #09

# CS-114 Fundamentals of Programming

**Course Instructor:** Khawaja Fahad Iqbal

**Lab Instructor:** Muhammad Affan

## Introduction:

**Name:** Muhammad Furqan Ul Arsh

**CMS ID:** 476347

**Section:** ME-15B

**Date:** 13-12-2023

# Lab Tasks

## Task 1:

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

## Code:

```cpp
//Lab Task 1
#include <bits/stdc++.h>
using namespace std;
int main(){
    int sum1=0,sum2=0;
    int A[3][3]={{1,2,3},
                 {4,5,6},
                 {7,8,9}};
    for(int i=0;i<3;i++){
        for(int j=0;j<3;j++){
            if(i==j){
                sum1+=A[i][j];
            }
            if((i+j)==2){
                sum2+=A[i][j];
            }
        }
    }
    cout<<"The Principle Diagnol Sum is "<<sum1<<endl;
    cout<<"The Right Diagnol Sum is "<<sum2<<endl;
return 0;}
```

## Output:

```
C:\Users\zennshi\Documents\Lab Manual 9.exe
The Principle Diagnol Sum is 15
The Right Diagnol Sum is 15

--------------------------------
Process exited after 0.4579 seconds with return value 0
Press any key to continue . . .
```

## Task 2:

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

## Code:

```cpp
//Task 2
#include <bits/stdc++.h>
using namespace std;

void ArraySum(int A1[3][3], int A2[3][3], int A3[3][3]) {
    for(int i=0;i<3;i++) {
        for(int j=0;j<3;j++){
            A3[i][j]=A1[i][j]+A2[i][j];
        }
    }
}
int main() {
    int A[3][3]={{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
    int B[3][3]={{10, 11, 12}, {13, 14, 15}, {16, 17, 18}};
    int Sum[3][3];
    cout<<"The Sum of Two Arrays is "<<endl;
    ArraySum(A,B,Sum);
    for(int i=0;i<3;i++) {
        for(int j=0;j<3;j++) {
            cout<<Sum[i][j]<<"  ";
        }
        cout<<endl;
    }
    return 0;}
```

## Output:

```
C:\Users\zennshi\Documents\Lab Task 9.2RAW2.exe
The Sum of Two Arrays is
11  13  15
17  19  21
23  25  27

--------------------------------
Process exited after 0.2975 seconds with return value 0
Press any key to continue . . .
```

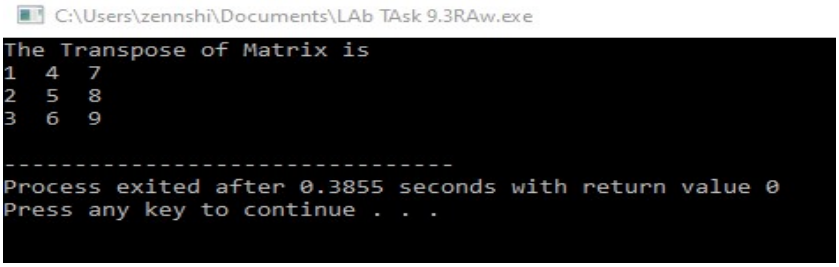
## Task 3:

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

## Code:

```cpp
//Task 3
#include <bits/stdc++.h>
using namespace std;
void Transpose(int A[3][3]){
    int temp;
    for(int i=0;i<3;i++){
        for(int j=i+1;j<3;j++){
        temp=A[i][j];
            A[i][j]=A[j][i];
            A[j][i]=temp;
        } }
}
int main(){
    int Alpha[3][3]={{1,2,3},
                     {4,5,6},
                     {7,8,9}};
    Transpose(Alpha);
    cout<<"The Transpose of Matrix is "<<endl;
            for(int i=0;i<3;i++){
        for(int j=0;j<3;j++){
        cout<<Alpha[i][j]<<"  ";
    }
    cout<<endl;
}
return 0;}
```

## Output:




# Task 4:

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

# Code:

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

// Function to multiply two 3x3 matrices
void multiplyMatrices(int firstMatrix[3][3], int secondMatrix[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] += firstMatrix[i][k] * secondMatrix[k][j];
            }
        }
    }
}

// Function to display a 3x3 matrix
void displayMatrix(int matrix[3][3]) {
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            cout << matrix[i][j] << " ";
        }
        cout << endl;
    }
}

int main() {
    int firstMatrix[3][3], secondMatrix[3][3], result[3][3];

    cout << "Enter the elements of the first 3x3 matrix:" << endl;
```

```cpp
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            cin >> firstMatrix[i][j];
        }
    }

    cout << "Enter the elements of the second 3x3 matrix:" << endl;
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            cin >> secondMatrix[i][j];
        }
    }

    multiplyMatrices(firstMatrix, secondMatrix, result);

    cout << "Result of matrix multiplication:" << endl;
    displayMatrix(result);

    return 0;
}
```

## Output:

```
Enter the elements of the first 3x3 matrix:
4
5
6
9
2
6
2
7
5
Enter the elements of the second 3x3 matrix:
1
2
3
4
5
9
5
4
6
Result of matrix multiplication:
54 57 93
47 52 81
55 59 99

--------------------------------
Process exited after 22.05 seconds with return value 0
Press any key to continue . . .
```

# Task 5:

Print the multiplication table of 15 using recursion.

# Code:

```cpp
#include <iostream>

using namespace std;

void printtable(int multiplier, int limit){

if(multiplier>limit){

return ;

}

cout<<" 15 x "<<multiplier<<" = "<<(15*multiplier)<<endl;

printtable (multiplier+1,limit);

}

int main(){

int limit=10;

cout<<"The multiplication table of 15 is: \n";

printtable (1,limit );

}
```

# Output:

```
The multiplication table of 15 is:
 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 exited after 0.1856 seconds with return value 0
Press any key to continue . . .
```

# Home Tasks

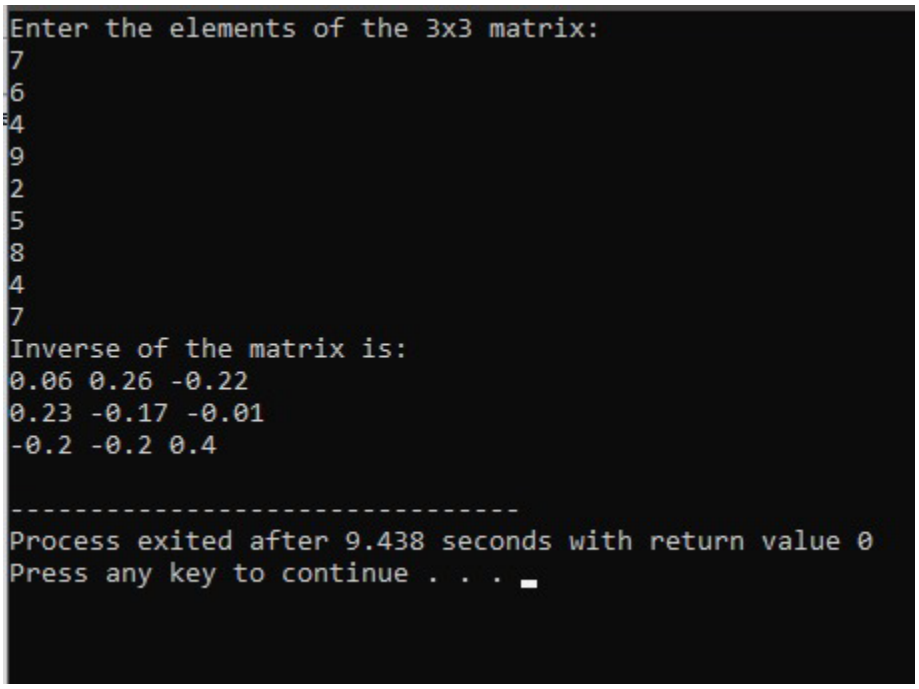
## Task 1:

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

## Code:

```cpp
#include <iostream>
using namespace std;
double determinant2x2(double a, double b, double c, double d) {
using namespace std;
    return a * d - b * c;
}
double determinant3x3(double 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]);
}
void adjoint3x3(double matrix[3][3], double adj[3][3]) {
    adj[0][0] = determinant2x2(matrix[1][1], matrix[1][2], matrix[2][1], matrix[2][2]);
    adj[0][1] = -determinant2x2(matrix[1][0], matrix[1][2], matrix[2][0], matrix[2][2]);
    adj[0][2] = determinant2x2(matrix[1][0], matrix[1][1], matrix[2][0], matrix[2][1]);

    adj[1][0] = -determinant2x2(matrix[0][1], matrix[0][2], matrix[2][1], matrix[2][2]);
    adj[1][1] = determinant2x2(matrix[0][0], matrix[0][2], matrix[2][0], matrix[2][2]);
    adj[1][2] = -determinant2x2(matrix[0][0], matrix[0][1], matrix[2][0], matrix[2][1]);

    adj[2][0] = determinant2x2(matrix[0][1], matrix[0][2], matrix[1][1], matrix[1][2]);
    adj[2][1] = -determinant2x2(matrix[0][0], matrix[0][2], matrix[1][0], matrix[1][2]);
    adj[2][2] = determinant2x2(matrix[0][0], matrix[0][1], matrix[1][0], matrix[1][1]);
}
void inverse3x3(double matrix[3][3], double inverse[3][3]) {
    double det = determinant3x3(matrix);
    if (det == 0) {
        cout << "Inverse does not exist as the determinant is zero." << endl;
        return;
    }
    double adj[3][3];
    adjoint3x3(matrix, adj);
    for (int i = 0; i < 3; ++i) { for (int j = 0; j < 3; ++j) { inverse[i][j] = adj[i][j] / det; } }
}
void displayMatrix(double matrix[3][3]) {
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            cout << matrix[i][j] << " ";
        }
        cout << endl;
    }
}

int main() {
    double matrix[3][3];
    cout << "Enter the elements of the 3x3 matrix:" << endl;
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            cin >> matrix[i][j];
        }
    }
    double inverse[3][3];
    inverse3x3(matrix, inverse);

    cout << "Inverse of the matrix is:" << endl;
    displayMatrix(inverse);

    return 0;
}
```

## Output:


Enter the elements of the 3x3 matrix:
7
6
4
9
2
5
8
4
7
Inverse of the matrix is:
0.06 0.26 -0.22
0.23 -0.17 -0.01
-0.2 -0.2 0.4
--------------------------------
Process exited after 9.438 seconds with return value 0
Press any key to continue . . .