School Of Mechanical & Manufacturing Engineering, NUST

Department of Mechanical Engineering

CS-114 - Fundamentals of Programing

Course Instructor: Dr Jawad Khan

Lab Instructor: Sir Affan

Student Name: Juveriah Waqqas

CMS ID: 460510

ME-15/A

LAB MANUAL 9

LAB TASKS

Q-No # 1

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

Code

```
#include<iostream>
using namespace std;
int main()
{
    int matrix[3][3];
    cout<<"Input the elements of the matrix: "<<endl;
    for(int i=0; i<3; i++)
    {
        for(int j=0; j<3; j++)
        {
            cin>>matrix[i][j];
        }
    }

    int sum_RD=0; int sum_LD=0;
    cout<<"The matrix you have input is : "<<endl;
    for(int i=0; i<3; i++)
    {
        for(int j=0; j<3; j++)
        {
            cout<<matrix[i][j]<<"  ";

            if(i==j)
            {
                sum_RD += matrix[i][j];
            }

            if( j == 3-i-1 )
            {
                sum_LD += matrix[i][j];
            }
        }
    cout<<endl;
    }
    cout<<"The sum of the right diagonal elements is  : "<<sum_RD<<endl;
    cout<<"The sum of the left diagonal elements is : "<<sum_LD;

return 0;
}
```

Execution (Example)

```
Input the elements of the matrix:
1
2
3
4
5
6
7
8
9
The matrix you have input is :
1  2  3
4  5  6
7  8  9
The sum of the right diagonal elements is  : 15
The sum of the left diagonal elements is : 15
--------------------------------
Process exited after 2.986 seconds with return value 0
Press any key to continue . . .
```

```
Input the elements of the matrix:
2
4
-6
5

7
8
-2
3
4
The matrix you have input is :
2  4  -6
5  7  8
-2  3  4
The sum of the right diagonal elements is  : 13
The sum of the left diagonal elements is : -1
--------------------------------
Process exited after 6.717 seconds with return value 0
Press any key to continue . . .
```

Q-No # 2

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

Code

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

    void inputarray ( int arr[3][3])
    {
        cout<<"Input the 2-d array ( starting from each element of the 1st row and so on): "<<endl;
        for(int i=0; i<3; i++)
        {
            for(int j=0; j<3; j++)
            {
                cin>>arr[i][j];
            }
        }
    }

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

    void sum_array (int a1[3][3], int a2[3][3] , int sumoas[3][3])
    {
        for(int i=0; i<3; i++)
        {
            for(int j=0; j<3; j++)
            {
                sumoas[i][j] = a1[i][j] + a2[i][j];
            }
        }
    }

int main()
{
    int firstarray[3][3] , secondarray[3][3] ,  sumofthearrays[3][3];

        cout<<"FRIST ARRAY"<<endl;
        inputarray (firstarray);
        cout<<"SECOND ARRAY"<<endl;
        inputarray (secondarray);

        cout<<"FRIST ARRAY"<<endl;
        outputarray (firstarray);
        cout<<"SECOND ARRAY"<<endl;
        outputarray (secondarray);

        sum_array
        (firstarray, secondarray, sumofthearrays);

        cout<<"The sum of the arrays is : "<<endl;
        outputarray(sumofthearrays);
return 0;
}
```

Execution (Example)

```
FRIST ARRAY
Input the 2-d array ( starting from each element of the 1st row and so on):
1
2
4
5
7
-3
4
5
-10
SECOND ARRAY
Input the 2-d array ( starting from each element of the 1st row and so on):
1
1
1
4
5
7
-9
7
4
FRIST ARRAY
1  2  4
5  7  -3
4  5  -10
SECOND ARRAY
1  1  1
4  5  7
-9  7  4
The sum of the arrays is :
2  3  5
9  12  4
-5  12  -6
```

Q-No # 3

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

Code

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

    void inputarray( int array[3][3])
    {
        for(int i=0; i<3; i++)
        {
            for(int j=0; j<3; j++)
            {
                cin>>array[i][j];
            }
        }
    }
```

```
void outputarray ( int array [3][3])
{
    for(int i=0; i<3; i++)
    {
        for(int j=0; j<3; j++)
        {
            cout<<array[i][j]<<"  ";
        }
    cout<<endl;
    }
}

void transpose( int array [3][3], int transposearray[3][3])
{
    for (int i=0; i<3; i++)
    {
        for(int j=0; j<3; j++)
        {
            transposearray[i][j] = array[j][i];
        }
    }
}

int main()
{
    int myarray[3][3], transarray[3][3];

    cout<<"Input the array (row by row) : "<<endl;
    inputarray (myarray);

    cout<<"Your inputted array is : "<<endl;
    outputarray (myarray);

    transpose( myarray , transarray);

    cout<<"The transpose of the array is "<<endl;
    outputarray (transarray);

return 0;
}
```

Execution (Example)

```
Input the array (row by row) :
2
4
5
1
7
-4
3
2
7
Your inputted array is :
2  4  5
1  7  -4
3  2  7
The transpose of the array is
2  1  3
4  7  2
5  -4  7

--------------------------------
Process exited after 7.222 seconds with return value 0
Press any key to continue . . .
```

```
Input the array (row by row) :
0
55
0
3
-6
-77
1
4
0
Your inputted array is :
0  55  0
3  -6  -77
1  4  0
The transpose of the array is
0  3  1
55  -6  4
0  -77  0

--------------------------------
Process exited after 7.807 seconds with return value 0
Press any key to continue . . .
```

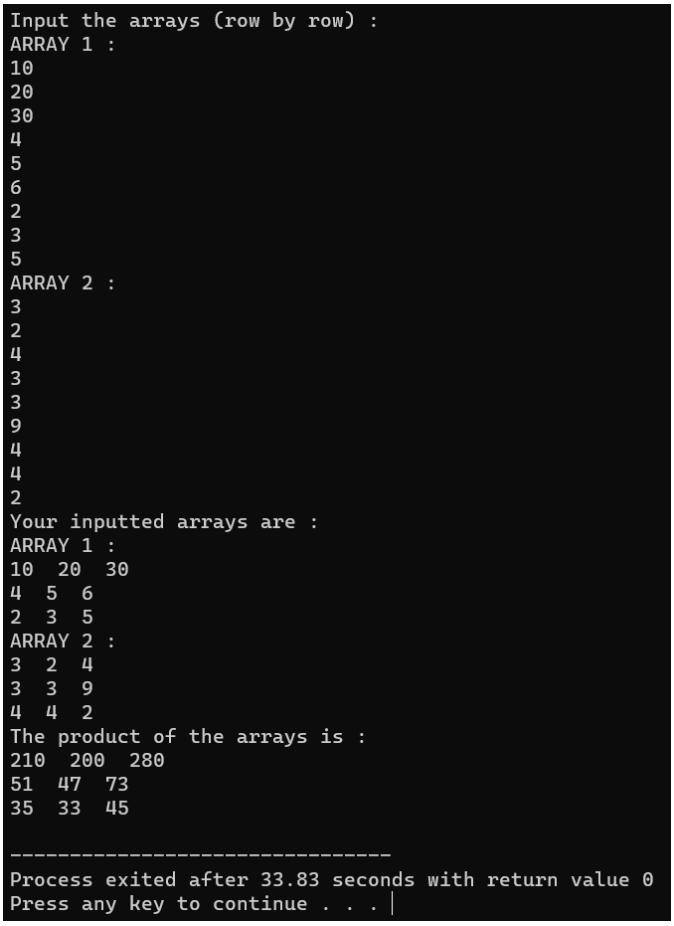
Q-No # 4

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

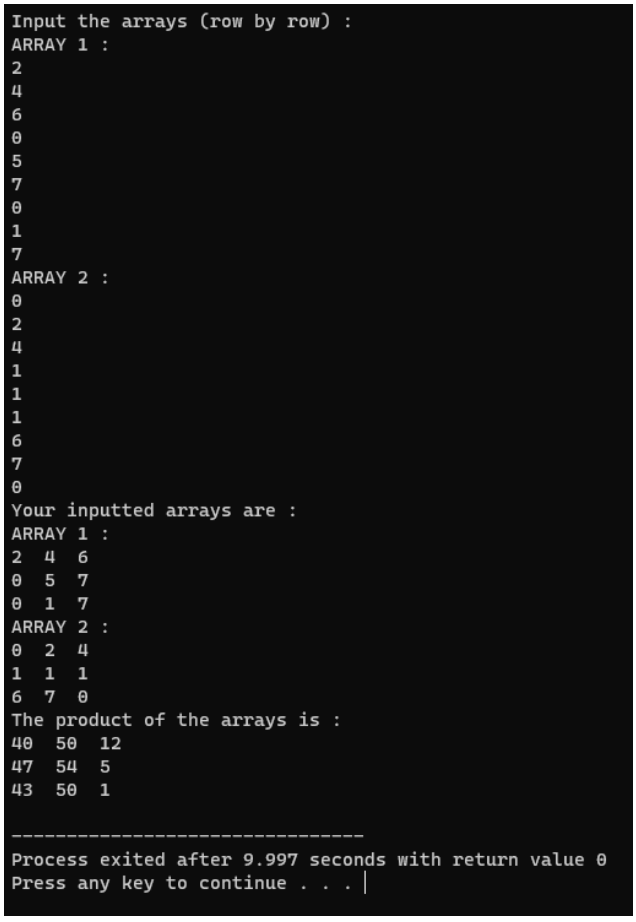
Code

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

    void inputarray( int array[3][3])
    {
        for(int i=0; i<3; i++)
        {
            for(int j=0; j<3; j++)
            {
                cin>>array[i][j];
            }
        }
    }

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


    void multiply2arrays ( int a1[3][3], int a2[3][3] , int product[3][3])
    {

        for(int i=0; i<3; i++)
        {
            for(int j=0; j<3; j++)
            {
                product[i][j]=0;
                for (int k=0; k<3; k++)
                {
                    product[i][j] += a1[i][k] * a2[k][j];
                }
            }
        }
    }

int main()
{
    int array1[3][3], array2[3][3], product[3][3];

    cout<<"Input the arrays (row by row) : "<<endl;
    cout<<"ARRAY 1 : "<<endl;
    inputarray (array1);
    cout<<"ARRAY 2 : "<<endl;
    inputarray (array2);


    cout<<"Your inputted arrays are : "<<endl;
    cout<<"ARRAY 1 : "<<endl;
    outputarray (array1);
    cout<<"ARRAY 2 : "<<endl;
    outputarray (array2);

    multiply2arrays ( array1, array2, product );
    cout<<"The product of the arrays is : "<<endl;
    outputarray (product);

return 0;
}
```

Execution (Example)



```
Input the arrays (row by row) :
ARRAY 1 :
10
20
30
4
5
6
2
3
5
ARRAY 2 :
3
2
4
3
3
9
4
4
2
Your inputted arrays are :
ARRAY 1 :
10  20  30
4  5  6
2  3  5
ARRAY 2 :
3  2  4
3  3  9
4  4  2
The product of the arrays is :
210  200  280
51  47  73
35  33  45

--------------------------------
Process exited after 33.83 seconds with return value 0
Press any key to continue . . .
```



```
Input the arrays (row by row) :
ARRAY 1 :
2
4
6
0
5
7
0
1
7
ARRAY 2 :
0
2
4
1
1
1
6
7
0
Your inputted arrays are :
ARRAY 1 :
2  4  6
0  5  7
0  1  7
ARRAY 2 :
0  2  4
1  1  1
6  7  0
The product of the arrays is :
40  50  12
47  54  5
43  50  1

--------------------------------
Process exited after 9.997 seconds with return value 0
Press any key to continue . . .
```

Q-No # 5

5. Print the multiplication table of 15 using recursion.

Code

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

    void multiplicationtable(int num, int limit, int i=1)
    {
        if ( i > limit){  return ;  }
        cout<<num<<" x "<<i<<" = "<<num*i<<endl;
        multiplicationtable (num, limit, i+1);
    }

int main()
{
    int lim;
    cout<<"Input the number till which you want the table : ";
    cin>>lim;

    cout<<"The multiplication table of 15 is : "<<endl;
    multiplicationtable(15, lim);

return 0;
}
```

Execution (Example)

```
Input the number till which you want the table : 15
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
15 x 11 = 165
15 x 12 = 180
15 x 13 = 195
15 x 14 = 210
15 x 15 = 225

--------------------------------
Process exited after 0.7037 seconds with return value 0
Press any key to continue . . .
```

HOME TASKS

Q-No # 1

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

Code

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

    void inputmatrix( double matrix[3][3])
    {
        for(int i=0; i<3; i++)
        {
            for(int j=0; j<3; j++)
            {
                cin>>matrix[i][j];
            }
        }
    }

    void outputmatrix( double matrix[3][3])
    {
        for(int i=0; i<3; i++)
        {
            for(int j=0; j<3; j++)
            {
                cout<<matrix[i][j]<<"   ";
            }
        cout<<endl;
        }
    }
```

```
    double det2by2 (double a , double b, double c, double d)
    {
     return a*d - b*c;
    }

    double det3by3 (double matrix[3][3])
    {
        return matrix[0][0] * det2by2( matrix[1][1], matrix[1][2], matrix [2][1], matrix [2][2]) -
               matrix[0][1] * det2by2( matrix[1][0], matrix[1][2], matrix [2][0], matrix [2][2]) +
               matrix[0][2] * det2by2( matrix[1][0], matrix[1][1], matrix [2][0], matrix [2][1]);
    }

    void adjoint( double matrix[3][3], double adjmatrix[3][3])
    {
        adjmatrix[0][0] = + det2by2( matrix[1][1], matrix[1][2], matrix [2][1], matrix [2][2] );
        adjmatrix[0][1] = - det2by2( matrix[0][1], matrix[0][2], matrix [2][1], matrix [2][2] );
        adjmatrix[0][2] = + det2by2( matrix[0][1], matrix[0][2], matrix [1][1], matrix [1][2] );

        adjmatrix[1][0] = - det2by2( matrix[1][0], matrix[1][2], matrix [2][0], matrix [2][2] );
        adjmatrix[1][1] = + det2by2( matrix[0][0], matrix[0][2], matrix [2][0], matrix [2][2] );
        adjmatrix[1][2] = - det2by2( matrix[0][0], matrix[0][2], matrix [1][0], matrix [1][2] );

        adjmatrix[2][0] = + det2by2( matrix[1][0], matrix[1][1], matrix [2][0], matrix [2][1] );
        adjmatrix[2][1] = - det2by2( matrix[0][0], matrix[0][1], matrix [2][0], matrix [2][1] );
        adjmatrix[2][2] = + det2by2( matrix[0][0], matrix[0][1], matrix [1][0], matrix [1][1] );
    }

    double inverse ( double matrix[3][3], double inverseofmatrix[3][3])
    {
     int det = det3by3(matrix);

        if(det == 0)
        {
            cout<<"The matrix is SINGULAR ,it has NO inverse "<<endl;
        }

        else
        {
            double adj[3][3];
            adjoint (matrix , adj);

            for(int i=0; i<3; i++)
            {
                for(int j=0; j<3; j++)
                {
                    inverseofmatrix[i][j] = adj [i][j] / det;
                }
            }
        }
    }

int main()
{
    double matrix [3][3];

    cout<<"Input the matrix (row by row) : "<<endl;
    inputmatrix (matrix);

    cout<<"The matrix is : "<<endl;
    outputmatrix (matrix);

    double inv [3][3];
    inverse ( matrix, inv );
    cout<<"The inverse of this matrix is : "<<endl;
    outputmatrix ( inv );

return 0;
}
```

Execution (Example)

```
Input the matrix (row by row) :
2
0
-1
5
1
0
0
1
3
The matrix is :
2   0   -1
5   1   0
0   1   3
The inverse of this matrix is :
3   -1   1
-15   6   -5
5   -2   2

--------------------------------
Process exited after 8.658 seconds with return value 0
Press any key to continue . . .
```

```
Input the matrix (row by row) :
1
0
5
-3
7
22
-5
0
3
The matrix is :
1   0   5
-3   7   22
-5   0   3
The inverse of this matrix is :
0.107143   -0   -0.178571
-0.515306   0.142857   -0.188776
0.178571   -0   0.0357143

--------------------------------
Process exited after 10.6 seconds with return value 0
Press any key to continue . . .
```