

FUNDAMENTALS OF PROGRAMMING (LAB)

LAB MANUAL 9 (LAB +HOME TASKS)

SUBMITTED BY: - ABEER ZAHRA JAFARI (476474) ME-15(C)

TASK # 01

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

main.cpp		Run	Output	Clear
<pre>1 #include <iostream> 2 using namespace std; 3 int main() { 4 int matrix[3][3]; //deifining size of matrix 5 cout << "Enter the elements of the 3x3 matrix:" <<endl; 6 for (int i = 0; i < 3; ++i) { //no. of rows 7 for (int j = 0; j < 3; ++j) { //no. of columns 8 cout << "Enter element at position [" << i << "][" << j << "]: "; 9 cin >> matrix[i][j]; } } //taking input from users 10 cout << "matrix:" <<endl; 11 for (int i = 0; i < 3; ++i) { //displaying the user defined matrix 12 for (int j = 0; j < 3; ++j) { 13 cout << matrix[i][j] << " "; 14 cout <<endl; } 15</pre>			<pre>/tmp/7ryvvZtoD3.o Enter the elements of the 3x3 matrix: Enter element at position [0][0]: 1 Enter element at position [0][1]: 2 Enter element at position [0][2]: 3 Enter element at position [1][0]: 4 Enter element at position [1][1]: 5 Enter element at position [1][2]: 6 Enter element at position [2][0]: 7 Enter element at position [2][1]: 8 Enter element at position [2][2]: 9 matrix: 1 2 3 4 5 6 7 8 9 Sum of left diagonal: 15 Sum of right diagonal: 15</pre>	

TASK # 01 (contd)

```
15
16 int sum1 = 0;
17 for (int i = 0; i < 3; ++i) {
18     sum1 = sum1 + matrix[i][i];
19     cout << "Sum of left diagonal: " << sum1 << endl;
20
21     int sum2 = 0;
22     for (int i = 0; i < 3; ++i) {
23         sum2 = sum2 + matrix[i][2-i];
24     }
25     cout << "Sum of right diagonal: " << sum2 << endl;
26     return 0;
}
```

Output

```
/tmp/7ryvvZtoD3.o
Enter the elements of the 3x3 matrix:
Enter element at position [0][0]: 1
Enter element at position [0][1]: 2
Enter element at position [0][2]: 3
Enter element at position [1][0]: 4
Enter element at position [1][1]: 5
Enter element at position [1][2]: 6
Enter element at position [2][0]: 7
Enter element at position [2][1]: 8
Enter element at position [2][2]: 9
matrix:
1 2 3
4 5 6
7 8 9
Sum of left diagonal: 15
Sum of right diagonal: 15
```

TASK # 02

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

```
main.cpp
1  #include <iostream>
2  using namespace std;
3  void // creating a function
4  sum(int matrix1[3][3], int matrix2[3][3], int matrix[3][3]) {
5      for (int i = 0; i < 3; ++i) {
6          for (int j = 0; j < 3; ++j) {
7              matrix[i][j] = matrix1[i][j] + matrix2[i][j];}}
8  int main() {
9      int matrix1[3][3], matrix2[3][3], matrix[3][3];
10         //declaring matrices
11  cout << "Enter elements for the first 3x3 matrix:" <<endl;
12  for (int i = 0; i < 3; ++i) { //creating first matrix
13      for (int j = 0; j < 3; ++j) {
14          cout << "Enter element at position [" << i << "][" << j << "]"
15              : " ";
16          cin >> matrix1[i][j]; }}
17  cout << "Enter elements for the second 3x3 matrix:" <<endl;
18  for (int i = 0; i < 3; ++i) { //creating second matrix
19      for (int j = 0; j < 3; ++j) {
20          cout << "Enter element at position [" << i << "][" << j << "]"
21              : " ";
22          cin >> matrix2[i][j]; }}
23  sum(matrix1, matrix2, matrix); //calling the function
24  cout << "Sum of the matrices:" <<endl;
25  for (int i = 0; i < 3; ++i) { // displaying output
26      for (int j = 0; j < 3; ++j) {
27          cout << matrix[i][j] << " "; }
28      cout << endl;}
29  return 0;}
```


TASK # 02 (OUTPUT)

Output

^ /tmp/INqC2Grivt.o

Enter elements for the first 3x3 matrix:

Enter element at position [0][0]: 1

Enter element at position [0][1]: 2

Enter element at position [0][2]: 3

Enter element at position [1][0]: 4

Enter element at position [1][1]: 5

Enter element at position [1][2]: 6

Enter element at position [2][0]: 7

Enter element at position [2][1]: 8

Enter element at position [2][2]: 9

Enter elements for the second 3x3 matrix:

Enter element at position [0][0]: 1

Enter element at position [0][1]: 2

Enter element at position [0][2]: 3

Enter element at position [1][0]: 4

Enter element at position [1][1]: 5

Enter element at position [0][1]: 2

Enter element at position [0][2]: 3

Enter element at position [1][0]: 4

Enter element at position [1][1]: 5

Enter element at position [1][2]: 6

Enter element at position [2][0]: 7

Enter element at position [2][1]: 8

Enter element at position [2][2]: 9

Sum of the matrices:

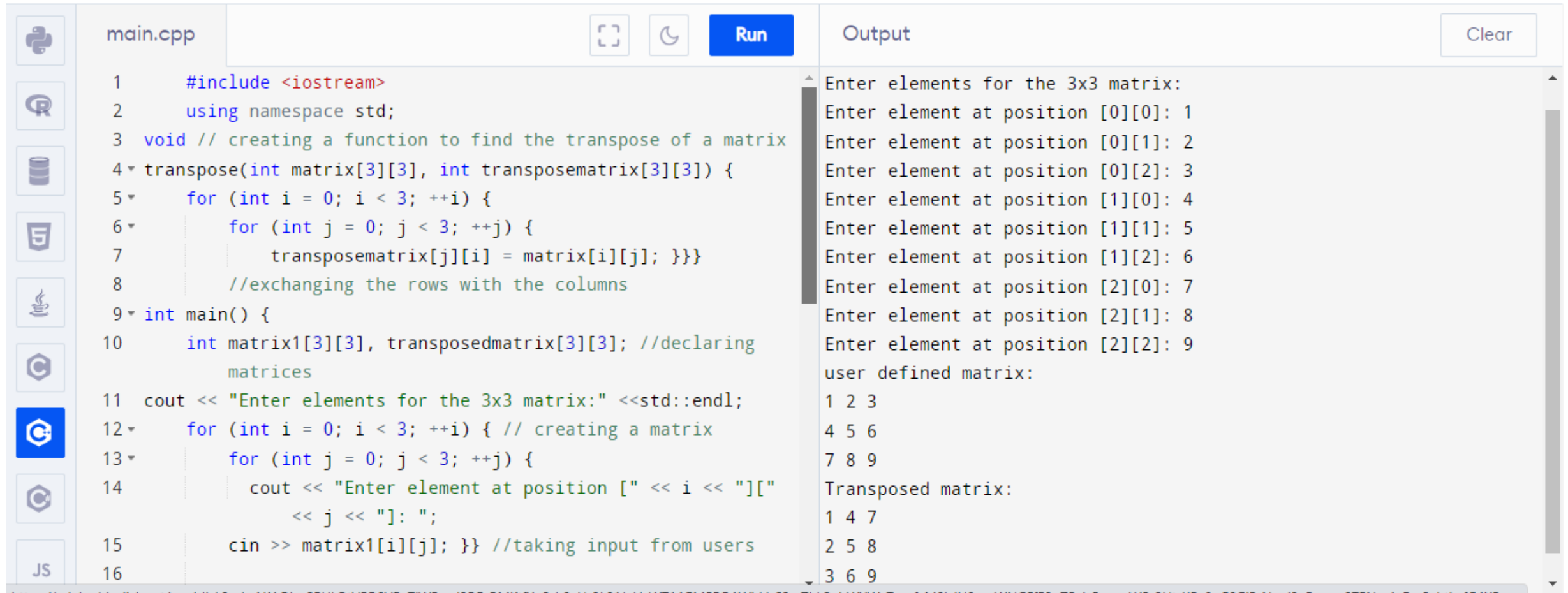
2 4 6

8 10 12

14 16 18

TASK # 03

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



The screenshot shows a C++ IDE with a file named `main.cpp`. The code defines a `transpose` function and a `main` function. The `transpose` function takes a 3x3 matrix and a transpose matrix as input and swaps the elements. The `main` function prompts the user to enter elements for a 3x3 matrix, reads the input, and then displays the transposed matrix.

```
main.cpp
1  #include <iostream>
2  using namespace std;
3  void // creating a function to find the transpose of a matrix
4  transpose(int matrix[3][3], int transposematrix[3][3]) {
5      for (int i = 0; i < 3; ++i) {
6          for (int j = 0; j < 3; ++j) {
7              transposematrix[j][i] = matrix[i][j]; }}}
8      //exchanging the rows with the columns
9  int main() {
10     int matrix1[3][3], transposedmatrix[3][3]; //declaring
        matrices
11     cout << "Enter elements for the 3x3 matrix:" <<std::endl;
12     for (int i = 0; i < 3; ++i) { // creating a matrix
13         for (int j = 0; j < 3; ++j) {
14             cout << "Enter element at position [" << i << "]["
                << j << "]: ";
15             cin >> matrix1[i][j]; }} //taking input from users
16
```

Output

```
Enter elements for the 3x3 matrix:
Enter element at position [0][0]: 1
Enter element at position [0][1]: 2
Enter element at position [0][2]: 3
Enter element at position [1][0]: 4
Enter element at position [1][1]: 5
Enter element at position [1][2]: 6
Enter element at position [2][0]: 7
Enter element at position [2][1]: 8
Enter element at position [2][2]: 9
user defined matrix:
1 2 3
4 5 6
7 8 9
Transposed matrix:
1 4 7
2 5 8
3 6 9
```

TASK # 03 (contd)

```
16
17 transpose(matrix1, transposedmatrix);
18 //calling the predefined function
19 cout << "user defined matrix:" <<endl; //displaying the
    original matrix
20 for (int i = 0; i < 3; ++i) {
21     for (int j = 0; j < 3; ++j) {
22         cout << matrix1[i][j] << " "; }
23     cout <<endl; }
24
25 cout << "Transposed matrix:" <<endl; //displaying output
26 for (int i = 0; i < 3; ++i) {
27     for (int j = 0; j < 3; ++j) {
28         cout << transposedmatrix[i][j] << " ";}
29     cout << endl;}
30 return 0;}
```

Output

```
Enter elements for the 3x3 matrix:
Enter element at position [0][0]: 1
Enter element at position [0][1]: 2
Enter element at position [0][2]: 3
Enter element at position [1][0]: 4
Enter element at position [1][1]: 5
Enter element at position [1][2]: 6
Enter element at position [2][0]: 7
Enter element at position [2][1]: 8
Enter element at position [2][2]: 9
user defined matrix:
1 2 3
4 5 6
7 8 9
Transposed matrix:
1 4 7
2 5 8
3 6 9
```

TASK # 04

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

```
main.cpp
1  #include <iostream>
2  using namespace std;
3  void //creating a function
4  multiplication(int matrix1[3][3], int matrix2[3][3],
5  int matrix[3][3]) {
6  for (int i = 0; i < 3; ++i) {
7  for (int j = 0; j < 3; ++j) {
8  matrix[i][j] = matrix1[i][j] * matrix2[i][j];}}
9
10 int main() {
11     int matrix1[3][3], matrix2[3][3], matrix[3][3];
12     // declaring matrices
13
14     cout << "Enter elements for the first 3x3 matrix:" << endl;
15     for (int i = 0; i < 3; ++i) { //taking input for first
        matrix
16     for (int j = 0; j < 3; ++j) {
17     cout << "Enter element at position [" << i << "][" << j << "]"
```

```
main.cpp
18
19 cout << "Enter elements for the second 3x3 matrix:" << endl;
20 for (int i = 0; i < 3; ++i) { //taking input for second
    matrix
21     for (int j = 0; j < 3; ++j) {
22         cout << "Enter element at position [" << i << "]["
            << j << "]: ";
23         cin >> matrix2[i][j]; }
24
25 multiplication(matrix1, matrix2, matrix);
26 //calling the predefined function
27 cout << "product of the matrices:" << std::endl;
28 for (int i = 0; i < 3; ++i) { //displaying output
29     for (int j = 0; j < 3; ++j) {
30         cout << matrix[i][j] << " "; }
31     cout << endl;}
32 return 0;}
33
```


TASK # 04 (OUTPUT)

Output

Clear

^ /tmp/INqC2Grivt.o

Enter elements for the first 3x3 matrix:

Enter element at position [0][0]: 1

Enter element at position [0][1]: 2

Enter element at position [0][2]: 3

Enter element at position [1][0]: 4

Enter element at position [1][1]: 5

Enter element at position [1][2]: 6

Enter element at position [2][0]: 7

Enter element at position [2][1]: 8

Enter element at position [2][2]: 9

Enter elements for the second 3x3 matrix:

Enter element at position [0][0]: 1

Enter element at position [0][1]: 2

Enter element at position [0][2]: 3

Enter element at position [1][0]: 4

Enter element at position [1][1]: 5

Enter element at position [1][2]: 6

Enter element at position [2][0]: 7

Enter element at position [2][1]: 8

Enter element at position [2][2]: 9

product of the matrices:

1 4 9

16 25 36

49 64 81

TASK # 05

- Print the multiplication table of 15 using recursion.

```
1  #include <iostream>
2  using namespace std;
3  void Table( int n, int m, int lim) {
4      if (m > lim) { //setting the limit
5          return;} // when the multipler becomes greater than
                    // the limit, the function is stopped to stop an
                    // infinite loop
6  cout << n << " x " << m << " = " << (n * m) << endl; //
    creating function
7  Table(n, m + 1, lim);} //setting parameters
8
9  int main() {
10     int num = 15; // declaring variables
11     int limit = 12;
12     cout << "Multiplication table of " << num << ":" << endl;
13     Table(num, 1, limit); //calling function
14     return 0;}
15
```

/tmp/kgSM0rm02B.o

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
15 x 11 = 165
15 x 12 = 180
```

TASK # 01 (HOME)

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

```
main.cpp  [Icons] [Run]

1  #include <iostream>
2  using namespace std;
3
4  int det(int m[3][3]) {
5      return m[0][0] * (m[1][1] * m[2][2] - m[2][1] * m[1][2])
6      - m[0][1] * (m[1][0] * m[2][2] - m[2][0] * m[1][2])
7      + m[0][2] * (m[1][0] * m[2][1] - m[2][0] * m[1][1]);
8
9  void transpose(int m[3][3], int x[3][3]) {
10     for (int i = 0; i < 3; ++i) {
11         for (int j = 0; j < 3; ++j) {
12             x[i][j] = m[j][i];
13         }
14     }
15
16     int cofactor(int a, int b, int c, int d) {
17         return a * d - b * c;
18     }
19
20     void adjoint(int m[3][3], int x[3][3]) {
21         x[0][0] = cofactor(m[1][1], m[1][2], m[2][1], m[2][2]);
22         x[0][1] = -cofactor(m[1][0], m[1][2], m[2][0], m[2][2]);
23         x[0][2] = cofactor(m[1][0], m[1][1], m[2][0], m[2][1]);
24         x[1][0] = -cofactor(m[0][1], m[0][2], m[2][1], m[2][2]);
25         x[1][1] = cofactor(m[0][0], m[0][2], m[2][0], m[2][2]);
26         x[1][2] = -cofactor(m[0][0], m[0][1], m[2][0], m[2][1]);
27         x[2][0] = cofactor(m[0][1], m[0][2], m[1][1], m[1][2]);
28         x[2][1] = -cofactor(m[0][0], m[0][2], m[1][0], m[1][2]);
29         x[2][2] = cofactor(m[0][0], m[0][1], m[1][0], m[1][1]);
30     }
31
32     void inverse(int m[3][3], int x[3][3]) {
33         int d = det(m);
34         if (d == 0) {
35             cout << "Matrix is singular. Inverse does not exist."
36                 << endl;
37             return;
38         }
39
40         adjoint(m, x);
41         for (int i = 0; i < 3; ++i) {
42             for (int j = 0; j < 3; ++j) {
43                 x[i][j] = x[i][j] / d;
44             }
45         }
46     }
47 }
```

TASK # 01 (contd)

main.cpp



Run

```
36 for (int i = 0; i < 3; ++i) {
37     for (int j = 0; j < 3; ++j) {
38         x[i][j] = adj[i][j] / d; }}}
39
40 void result(int m[3][3]) {
41     for (int i = 0; i < 3; ++i) {
42         for (int j = 0; j < 3; ++j) {
43             cout << m[i][j] << " ";
44         }
45         cout << endl;
46 }
47
48 int main() {
49     int m[3][3];
50     cout << "Enter the elements of the 3x3 matrix:" << endl;
```

```
48     cout << "Enter the elements of the 3x3 matrix:" << endl;
49     for (int i = 0; i < 3; ++i) {
50         for (int j = 0; j < 3; ++j) {
51             cin >> m[i][j];
52         }
53     }
54     int im[3][3];
55     inverse(m, im);
56     cout << "Original matrix:" << endl;
57     result(m);
58     cout << endl;
59     cout << "Inverse matrix:" << endl;
60     result(im);
61     return 0;
62 }
```

TASK # 01 (OUTPUT)

```
Output Clear  
^ /tmp/wKCprZTI0b.o  
Enter the elements of the 3x3 matrix:  
7  
-3  
-3  
-1  
1  
0  
-1  
0  
1  
Original matrix:  
7 -3 -3  
-1 1 0  
-1 0 1  
  
Inverse matrix:  
1 3 3  
1 4 3  
1 3 4
```