BSCS-Spring-2021

CL118-Programming Fundamentals

Lab Manual 13

LAB 13 Manual

Programming Fundamentals

Topic: Arrays-II (char and multi-dimensional arrays)

C++ provides following two types of string representations

• The C-style character string.

• The string class type introduced with Standard C++.

The C-Style Character String:

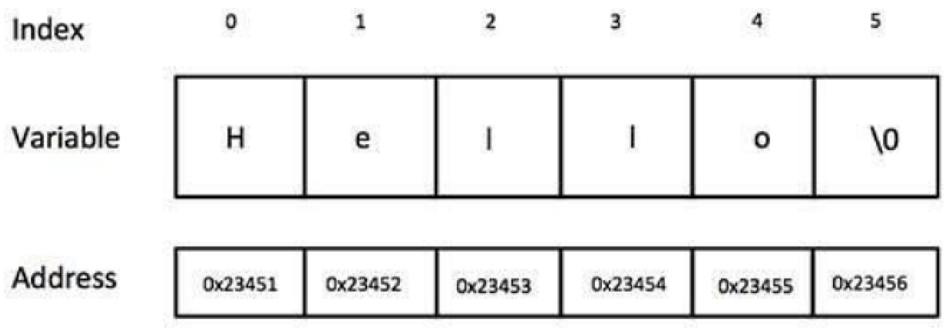
The C-style character string originated within the C language and continues to be supported within C++. This string is actually a one-dimensional array of characters which is terminated by a null character '\0'. Thus a null-terminated string contains the characters that comprise the string followed by a null.

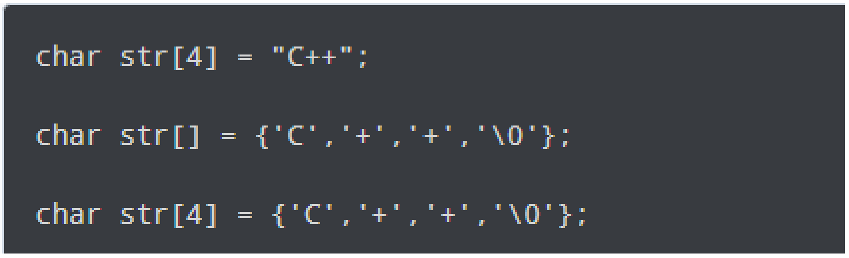
Defining a C-string:

The following declaration and initialization create a string consisting of the word "Hello". To hold the null character at the end of the array, the size of the character array containing the string is one more than the number of characters in the word "Hello."

char greeting[6] = {'H', 'e', 'l', 'l', 'o', '\0'};

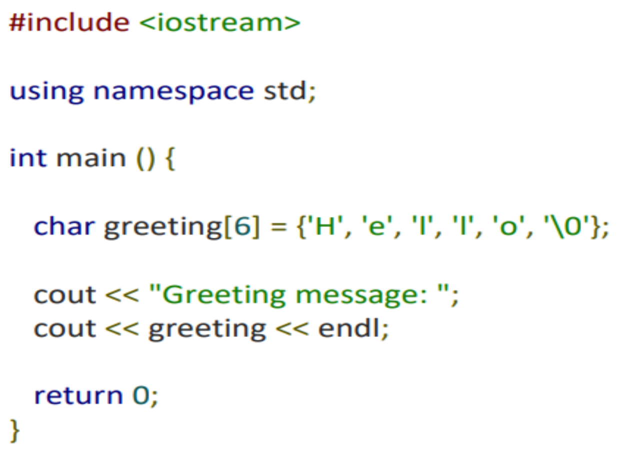
Following is the memory presentation of above defined string in C/C++.





```
char str[4] = "C++";

char str[] = {'C','+','+','\0'};

char str[4] = {'C','+','+','\0'};
```

Example:

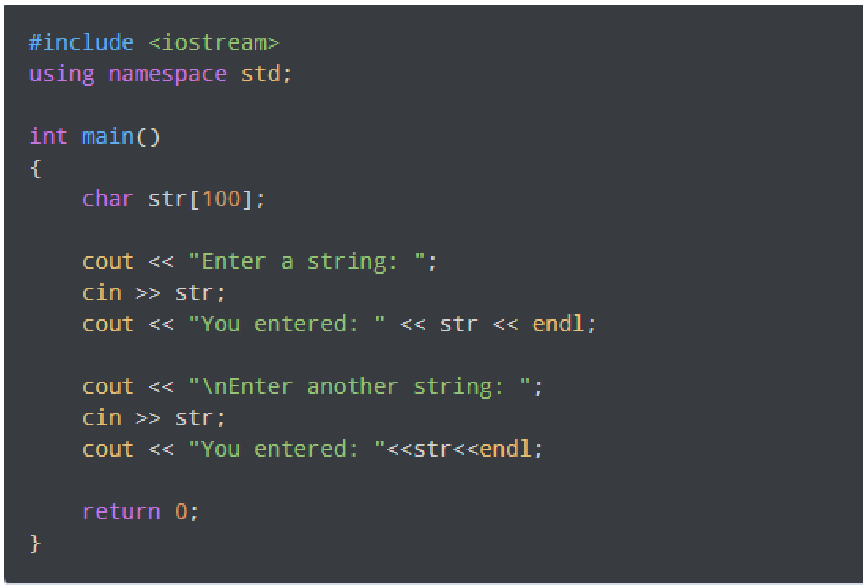


```
#include <iostream>

using namespace std;

int main () {

   char greeting[6] = {'H', 'e', 'l', 'l', 'o', '\0'};

   cout << "Greeting message: ";
   cout << greeting << endl;

   return 0;
}
```

Output:

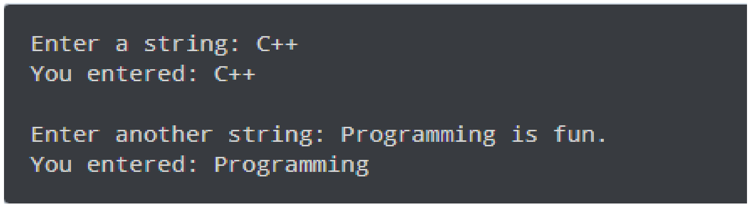
Greeting message: Hello

Example:



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

int main()
{
    char str[100];

    cout << "Enter a string: ";
    cin >> str;
    cout << "You entered: " << str << endl;

    cout << "\nEnter another string: ";
    cin >> str;
    cout << "You entered: "<<str<<endl;

    return 0;
}
```

Output:



```
Enter a string: C++
You entered: C++

Enter another string: Programming is fun.
You entered: Programming
```

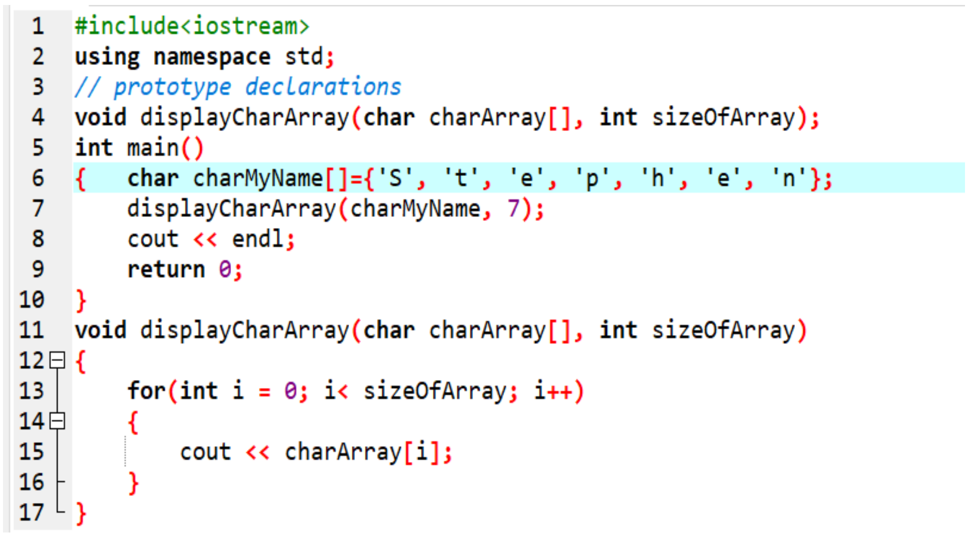
Example:

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

int main()
{
    char str[100];
    cout << "Enter a string: ";
    cin.get(str, 100);

    cout << "You entered: " << str << endl;
    return 0;
}
```

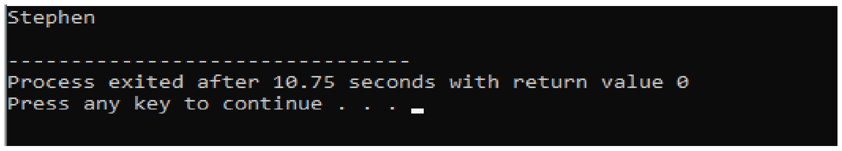
Output:

```
Enter a string: Programming is fun.
You entered: Programming is fun.
```

Passing array to a function:



```cpp
#include<iostream>
using namespace std;
// prototype declarations
void displayCharArray(char charArray[], int sizeOfArray);
int main()
{   char charMyName[]={'S', 't', 'e', 'p', 'h', 'e', 'n'};
    displayCharArray(charMyName, 7);
    cout << endl;
    return 0;
}
void displayCharArray(char charArray[], int sizeOfArray)
{
    for(int i = 0; i< sizeOfArray; i++)
    {
        cout << charArray[i];
    }
}
```



```
Stephen

--------------------------------
Process exited after 10.75 seconds with return value 0
Press any key to continue . . .
```

The String Class in C++:

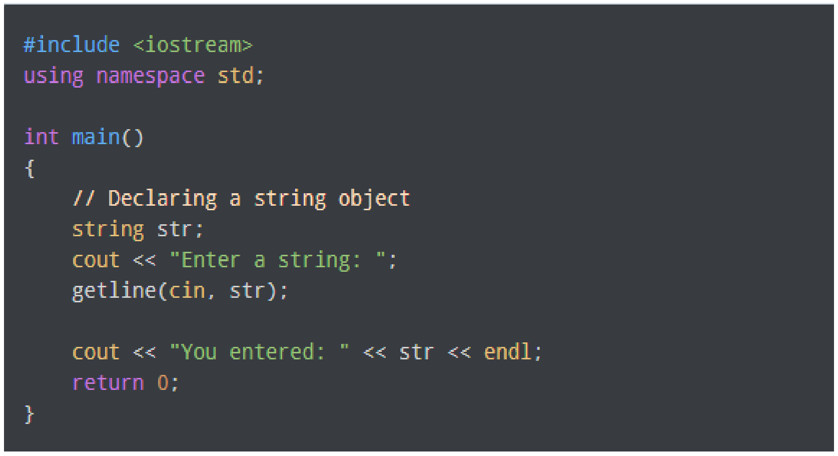
In C++, you can also create a string object for holding strings. Unlike using char arrays, string objects has no fixed length, and can be extended as per your requirement.

string str1 = "Hello";

int len = str1.size();   //length of str1

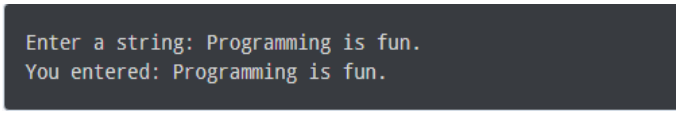
cout << "str1.size( ) : " << len << endl;   //output will be 5

Example:



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

int main()
{
    // Declaring a string object
    string str;
    cout << "Enter a string: ";
    getline(cin, str);

    cout << "You entered: " << str << endl;
    return 0;
}
```

Output:



```
Enter a string: Programming is fun.
You entered: Programming is fun.
```

Multi-dimensional Arrays:

C++ allows multidimensional arrays. Here is the general form of a multidimensional array declaration:

type name[size1][size2]...[sizeN];

For example, the following declaration creates a three dimensional 5 . 10 . 4 integer array

int threeD[5][10][4];

• Can define arrays with any number of dimensions:

short rectSolid[2][3][5];

double timeGrid[3][4][3][4];

• When used as parameter, specify all but 1st dimension in prototype, heading:

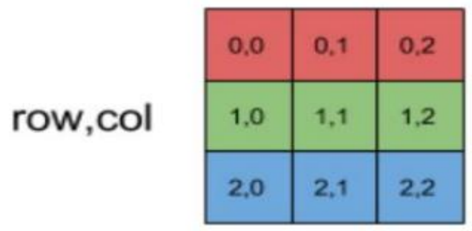
void getRectSolid(short [][3][5]);

2D Array:

The simplest form of the multidimensional array is the two-dimensional array.

You can define one array for multiple sets of data

Like a table in a spreadsheet



Use two size declarators in definition:

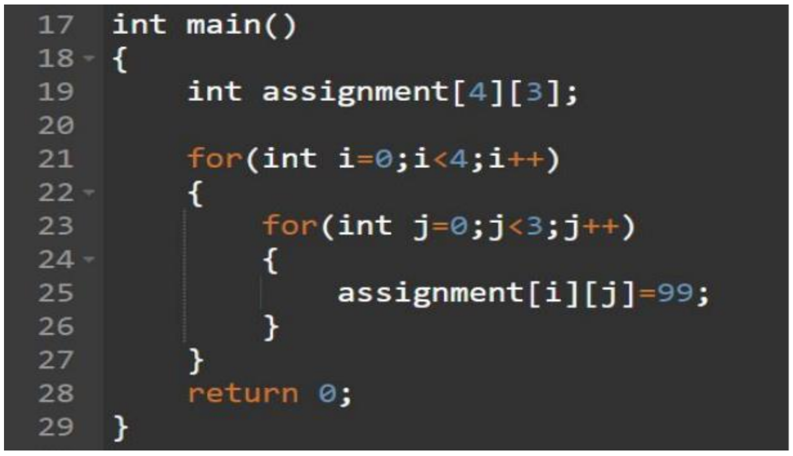
const int ROWS = 3, COLS = 3;

int exams[ROWS][COLS];

First declarator is number of rows; second is number of columns

Use two subscripts to access element: **exams[2][2] = 86;**

Looping over a 2D Array:



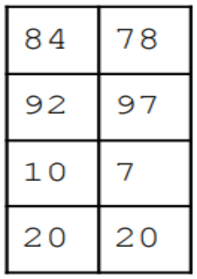
```
int main()
{
    int assignment[4][3];

    for(int i=0;i<4;i++)
    {
        for(int j=0;j<3;j++)
        {
            assignment[i][j]=99;
        }
    }
    return 0;
}
```

Initializing Two-Dimensional Arrays:

Two-dimensional arrays are initialized row-by-row:

const int ROWS = 4, COLS = 2;

int exams[ROWS][COLS]={ {84, 78}, {92, 97}, {10, 7}, {20, 20} };



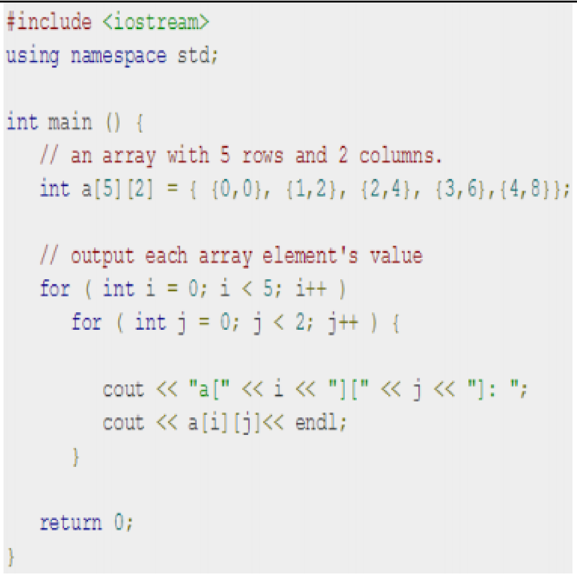
84	78
92	97
10	7
20	20

Accessing Two-Dimensional Array Elements:

An element in 2-dimensional array is accessed by using the subscripts, i.e., row index and column index of the array. For example:

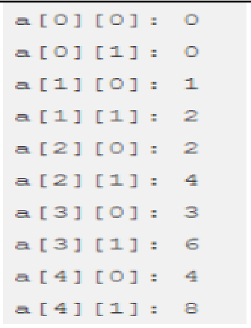
int val = a[2][3];

Example



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

int main () {
   // an array with 5 rows and 2 columns.
   int a[5][2] = { {0,0}, {1,2}, {2,4}, {3,6},{4,8}};

   // output each array element's value
   for ( int i = 0; i < 5; i++ )
      for ( int j = 0; j < 2; j++ ) {

         cout << "a[" << i << "][" << j << "]: ";
         cout << a[i][j]<< endl;
      }

   return 0;
}
```

OUTPUT



```
a[0][0]: 0
a[0][1]: 0
a[1][0]: 1
a[1][1]: 2
a[2][0]: 2
a[2][1]: 4
a[3][0]: 3
a[3][1]: 6
a[4][0]: 4
a[4][1]: 8
```

2D array as Function arguments:

•Use array name as argument in function call:

getExams(exams, 2);

•Use empty [] for row, size declarator for column in prototype, header:

const int COLS = 2;

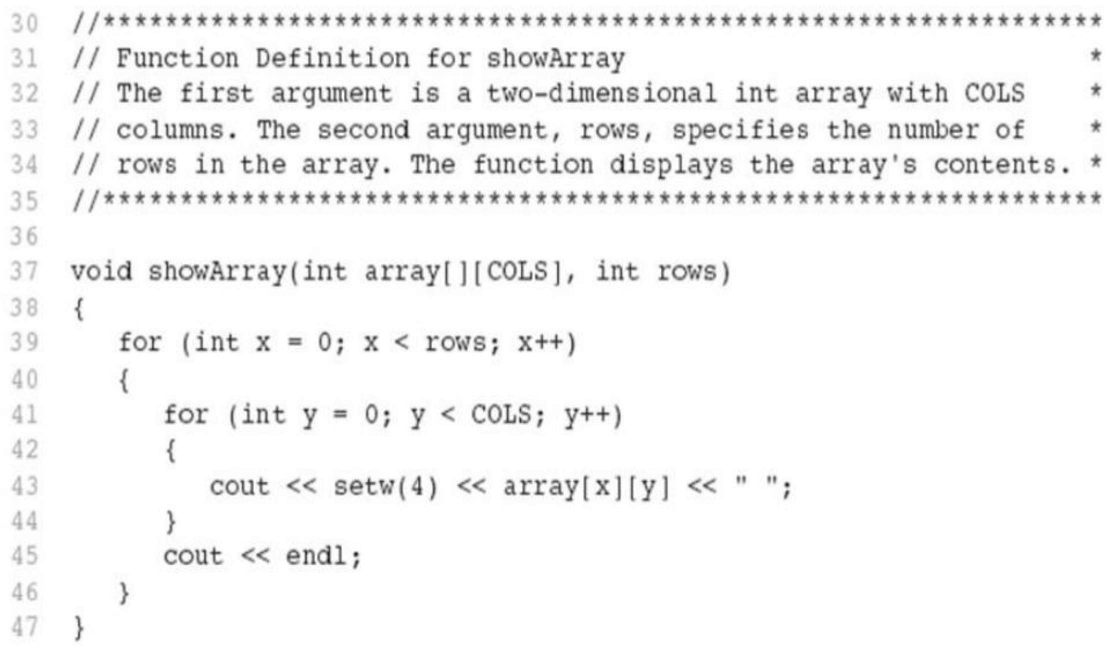
// Prototype

void getExams(int [][COLS], int);

// Header

void getExams(int exams[][COLS], int rows)

Example:



```
//*****************************************************************
// Function Definition for showArray                              *
// The first argument is a two-dimensional int array with COLS    *
// columns. The second argument, rows, specifies the number of    *
// rows in the array. The function displays the array's contents. *
//*****************************************************************

void showArray(int array[][COLS], int rows)
{
   for (int x = 0; x < rows; x++)
   {
      for (int y = 0; y < COLS; y++)
      {
         cout << setw(4) << array[x][y] << " ";
      }
      cout << endl;
   }
}
```

```cpp
   int table1[TBL1_ROWS][COLS] = {{1, 2, 3, 4},
                                  {5, 6, 7, 8},
                                  {9, 10, 11, 12}};
   int table2[TBL2_ROWS][COLS] = {{10, 20, 30, 40},
                                  {50, 60, 70, 80},
                                  {90, 100, 110, 120},
                                  {130, 140, 150, 160}};

   cout << "The contents of table1 are:\n";
   showArray(table1, TBL1_ROWS);
   cout << "The contents of table2 are:\n";
   showArray(table2, TBL2_ROWS);
```

Program 7-18

```cpp
// This program demonstrates a two-dimensional array.
#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
   const int NUM_DIVS = 3;            // Number of divisions
   const int NUM_QTRS = 4;            // Number of quarters
   double sales[NUM_DIVS][NUM_QTRS]; // Array with 3 rows and 4 columns.
   double totalSales = 0;             // To hold the total sales.
   int div, qtr;                      // Loop counters.

   cout << "This program will calculate the total sales of\n";
   cout << "all the company's divisions.\n";
   cout << "Enter the following sales information:\n\n";

```

(program continues)

Program 7-18 *(continued)*

```
    // Nested loops to fill the array with quarterly
    // sales figures for each division.
    for (div = 0; div < NUM_DIVS; div++)
    {
       for (qtr = 0; qtr < NUM_QTRS; qtr++)
       {
          cout << "Division " << (div + 1);
          cout << ", Quarter " << (qtr + 1) << ": $";
          cin >> sales[div][qtr];
       }
       cout << endl; // Print blank line.
    }

    // Nested loops used to add all the elements.
    for (div = 0; div < NUM_DIVS; div++)
    {
       for (qtr = 0; qtr < NUM_QTRS; qtr++)
          totalSales += sales[div][qtr];
    }

    cout << fixed << showpoint << setprecision(2);
    cout << "The total sales for the company are: $";
    cout << totalSales << endl;
    return 0;
 }
```

Program Output with Example Input Shown in Bold

```
This program will calculate the total sales of
all the company's divisions.
Enter the following sales data:

Division 1, Quarter 1: $31569.45 [Enter]
Division 1, Quarter 2: $29654.23 [Enter]
Division 1, Quarter 3: $32982.54 [Enter]
Division 1, Quarter 4: $39651.21 [Enter]

Division 2, Quarter 1: $56321.02 [Enter]
Division 2, Quarter 2: $54128.63 [Enter]
Division 2, Quarter 3: $41235.85 [Enter]
Division 2, Quarter 4: $54652.33 [Enter]

Division 3, Quarter 1: $29654.35 [Enter]
Division 3, Quarter 2: $28963.32 [Enter]
Division 3, Quarter 3: $25353.55 [Enter]
Division 3, Quarter 4: $32615.88 [Enter]

The total sales for the company are: $456782.34
```

Lab Tasks

Note: Work in functions for the following lab tasks

Q1) Write a C++ program to reverse a string taken as a user input.

Sample Output:

```
Original string: Program

Reverse string: margorP
```

Q2) Write a C++ program to count all the vowels in a string taken as user input.

Sample Output:

```
Original string: hello          number of vowels: 2

Original string: AeiOu          number of vowels: 5
```

Q3) Write a C++ program to find the frequency of characters in a string.

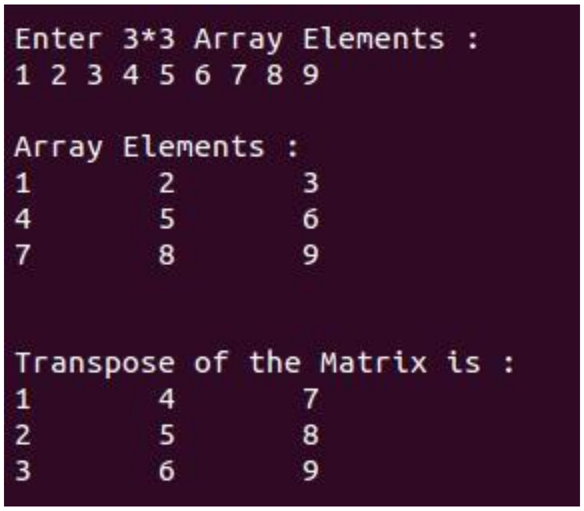
Sample Output:

```
Original string: Programming is awesome

Number of a = 2

Number of P = 1

Number of m = 3   and so on…
```

Q4) Write a C++ Program to Multiply Two Matrix Using Multi-Dimensional Arrays. This program takes two matrices of order r1*c1 and r2*c2 respectively. Then, the program multiplies these two matrices (if possible) and displays it on the screen. **Note:** To multiply two matrices, the number of columns of first matrix should be equal to the number of rows of second matrix.

Q5) Write a C++ program that takes a matrix of r*c from the user and computes the transpose of the matrix.

Sample Output:



```
Enter 3*3 Array Elements :
1 2 3 4 5 6 7 8 9

Array Elements :
1       2       3
4       5       6
7       8       9


Transpose of the Matrix is :
1       4       7
2       5       8
3       6       9
```

Submission Instructions:

1. Save all .cpp files with your roll no and task number e.g. i20XXXX_Task01.cpp

2. Now create a new folder with name ROLLNO_LAB13 e.g. i20XXXX_LAB13

3. Move all of your .cpp files to this newly created directory and compress it into .zip file.

4. Now you have to submit this zipped file on Google Classroom.