LAB 10: Arrays

Objective(s):
To Understand about:
Apply Arrays in C++
Use of Two and Three Dimensional Array
CLOs: CLO1, CLO4
2D Array:
An array is useful for storing and working with a set of data. Sometimes, though, it's necessary to work with multiple sets of data. For example, in a grade-averaging program a teacher might record all of one student's test scores in an array of doubles. If the teacher has 30 students, that means she'll need 30 arrays of doubles to record the scores for the entire class. Instead of defining 30 individual arrays, however, it would be better to define a two-dimensional array.
The arrays that you have studied so far are one-dimensional arrays. They are called *one dimensional* because they can only hold one set of data. Two-dimensional arrays, which are sometimes called *2D arrays*, can hold multiple sets of data. It's best to think of a two dimensional array as having rows and columns of elements, as shown in Figure. This figure shows an array of test scores, having three rows and four columns.

	Column 0	Column 1	Column 2	Column 3
Row 0	scores[0] [0]	scores[0] [1]	scores[0] [2]	scores[0] [3]
Row 1	scores[1] [0]	scores[1] [1]	scores[1] [2]	scores[1] [3]
Row 2	scores[2] [0]	scores[2] [1]	scores[2] [2]	scores[2] [3]

To define a two-dimensional array, two size declarators are required. The first one is for the number of rows and the second one is for the number of columns. Here is an example definition of a two-dimensional array with three rows and four columns:
The first size declarator specifies the number of rows, and the second size declarator specifies the number of columns. Notice that each number is enclosed in its own set of brackets. When processing the data in a two-dimensional array, each element has two subscripts: one for its row and another for its column. In the scores array defined above, the elements in row 0 are referenced as
scores[0][0]
scores[0][1]
scores[0][2]
scores[0][3]
The elements in row 1 are
scores[1][0]
scores[1][1]
scores[1][2]
scores[1][3]
And the elements in row 2 are
scores[2][0]

scores[2][1]
scores[2][2]
scores[2][3]
The subscripted references are used in a program just like the references to elements in a single dimensional array, except now you use two subscripts. The first subscript represents the row position, and the second subscript represents the column position. For example, the following statement assigns the value 92.25 to the element at row 2, column 1 of the scores array:
scores[2][1] = 92.25;
And the following statement displays the element at row 0, column 2:
cout << scores[0][2];

Example:

```
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";

   // 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]
```

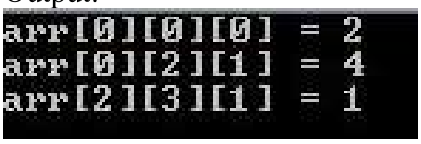
3D Array:

Three dimensional (3D) array contains three for loops in programming. So, to initialize and print three dimensional array, you have to use three for loops. Third for loop (the innermost loop) forms 1D array, Second for loop forms 2D array and the third for loop (the outermost loop) forms 3D array, as shown here in the following program.

Example:

```
#include<iostream.h>
#include<conio.h>
void main()
{
        clrscr();
        int arr[3][4][2] = {
                                {
                                    {2, 4},
                                    {7, 8},
                                    {3, 4},
                                    {5, 6}
                                },
                                {
                                    {7, 6},
                                    {3, 4},
                                    {5, 3},
                                    {2, 3}
                                },
                                {
                                    {8, 9},
                                    {7, 2},
                                    {3, 4},
                                    {5, 1}
                                }
                            };
        cout<<"arr[0][0][0] = "<<arr[0][0][0]<<"\n";
        cout<<"arr[0][2][1] = "<<arr[0][2][1]<<"\n";
        cout<<"arr[2][3][1] = "<<arr[2][3][1]<<"\n";
        getch();
}
```

Output:



Lab Tasks:

Array: ROWS=4, COLS=5

1. Summing all values of 2D Array
2. Summing the rows of 2D array
3. Summing the columns of 2D array
4. Get highest in 2D array
5. Get lowest in 2D array
6. get highest in specified row
7. get lowest in specified row
8. Add Two Matrices(2*2) using 2D Arrays
9. SubtractTwo Matrices(2*2) using 2D Arrays