Chapter 9: Arrays and Strings

In this chapter you will learn about:

- Passing Two-Dimensional Arrays as Parameters to Functions

Passing Two-Dimensional Arrays as Parameters to Functions

Two-dimensional arrays can be passed as parameters to a function, and they are passed by reference. The base address (that is, the address of the first component of the actual parameter) is passed to the formal parameter. If `matrix` is the name of a two-dimensional array, then `matrix[0][0]` is the first component of `matrix`.

When storing a two-dimensional array in the computer's memory, C++ uses the **row order form**. That is, the first row is stored first, followed by the second row, followed by the third row, and so on.

In the case of a one-dimensional array, when declaring it as a formal parameter, we usually omit the size of the array. Because C++ stores two-dimensional arrays in row order form, to compute the address of a component correctly, the compiler must know where one row ends and the next row begins. Thus, when declaring a two-dimensional array as a formal parameter, you can omit the size of the first dimension, but not the second; that is, you must specify the number of columns.

Suppose we have the following declaration:

```
const int NUMBER_OF_ROWS = 6;
const int NUMBER_OF_COLUMNS = 5;
```

Consider the following definition of the function `printMatrix`:

```
void printMatrix(int matrix[][NUMBER_OF_COLUMNS],
                 int noOfRows)
{
    int row, col;

    for (row = 0; row < noOfRows; row++)
    {
        for (col = 0; col < NUMBER_OF_COLUMNS; col++)
            cout << setw(5) << matrix[row][col] << " ";

        cout << endl;
    }
}
```

This function takes as a parameter a two-dimensional array of an unspecified number of rows and `five` columns, and outputs the content of the two-dimensional array. During the function call, the number of columns of the actual parameter must match the number of columns of the formal parameter.

Similarly, the following function outputs the sum of the elements of each row of a two-dimensional array whose elements are of type int.

```
void sumRows(int matrix[][NUMBER_OF_COLUMNS], int noOfRows)
{
    int row, col;
    int sum;

         //Sum of each individual row
    for (row = 0; row < noOfRows; row++)
        {
            sum = 0;

            for (col = 0; col < NUMBER_OF_COLUMNS; col++)
                sum = sum + matrix[row][col];

            cout << "Sum of row " << (row + 1) << " = " << sum
                 << endl;
        }
    }
```

The following function determines the largest element in each row:

```
void largestInRows(int matrix[][NUMBER_OF_COLUMNS],
                   int noOfRows)
{
    int row, col;
    int largest;

         //Largest element in each row
    for (row = 0; row < noOfRows; row++)
    {
        largest = matrix[row][0]; //Assume that the first element
                                  //of the row is the largest.
        for (col = 1; col < NUMBER_OF_COLUMNS; col++)
            if (largest < matrix[row][col])
                largest = matrix[row][col];

        cout << "The largest element of row " << (row + 1)
             << " = " << largest << endl;
    }
}
```

References

1. Malik, D. S. (2010). *C++ programming: Program design including data structures.* Course Technology.