Data Structure

Lab: #01

Liner Data Structure

Array

Asst. Instructor

Ali A. Al-Ani

Introduction

C++ is a statically typed, compiled, general-purpose, case-sensitive, free-form programming language that supports procedural, object-oriented, and generic programming. C++ is regarded as a **middle-level** language, as it comprises a combination of both high-level and low-level language features. C++ was developed by Bjarne Stroustrup starting in 1979 at Bell Labs in Murray Hill, New Jersey, as an enhancement to the C language and originally named C with Classes but later it was renamed C++ in 1983. C++ is a superset of C, and that virtually any legal C program is a legal C++ program.

C++ Program Structure

Let us look at a simple code that would print the words Hello World.

```
#include <iostream.h>
// main() is where program execution begins.
int main() {
   cout << "Hello World"; // prints Hello World
   return 0;
}
```

Let us look at the various parts of the above program:

- The C++ language defines several headers, which contain information that is either necessary or useful to your program. For this program, the header <iostream.h> is needed.
- The next line '// main() is where program execution begins.' is a single-line comment available in C++. Single-line comments begin with // and stop at the end of the line.
- The line int main() is the main function where program execution begins.
- The next line cout << "This is my first C++ program."; causes the message "This is my first C++ program" to be displayed on the screen.
- The next line return 0; terminates main( )function and causes it to return the value 0 to the calling process.

Arrays

C++ provides a data structure, the array, which stores a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type. Instead of declaring individual variables, such as number0, number1, ..., and number99, you declare one array variable such as numbers and use numbers [0], numbers [1], and ..., numbers [99] to represent individual variables. A specific element in an array is accessed by an index. All arrays consist of contiguous memory locations. The lowest address corresponds to the first element and the highest address to the last element.

Declaring Arrays

To declare an array in C++, the programmer specifies the type of the elements and the number of elements required by an array as follows:

```
type arrayName [ arraySize ];
```

This is called a single-dimension array. The arraySize must be an integer constant greater than zero and type can be any valid C++ data type. For example, to declare a 10-element array called balance of type double, use this statement:

```
double balance[10];
```

Accessing Array Elements

An element is accessed by indexing the array name. This is done by placing the index of the element within square brackets after the name of the array. For example:

```
double salary = balance[9];
```

The above statement will take 10th element from the array and assign the value to salary variable. Following is an example, which will use all the above-mentioned three concepts viz. declaration, assignment and accessing arrays:

```
#include <iostream.h>
int main () {
   int n[ 10 ]; // n is an array of 10 integers
    // initialize elements of array n to 0
   for ( int i = 0; i < 10; i++ ) {
```

```
      n[ i ] = i + 100; // set element at location i to i + 100
   }
   cout << "Element" << "       " << "Value" << endl;
    // output each array element's value
   for ( int j = 0; j < 10; j++ ) {
      cout << j << "       " << n[ j ] << endl;
   }
    return 0;
}
```

When the above code is compiled and executed, it produces the following result −

```
Element      Value
    0        100
    1        101
    2        102
    3        103
    4        104
    5        105
    6        106
    7        107
    8        108
    9        109
```

Two-Dimensional Arrays

The simplest form of the multidimensional array is the two-dimensional array. A two-dimensional array is, in essence, a list of one-dimensional arrays. To declare a two-dimensional integer array of size x,y, you would write something as follows :

```
type arrayName [ x ][ y ];
```

Where type can be any valid C++ data type and arrayName will be a valid C++ identifier. A two-dimensional array can be think as a table, which will have x number of rows and y number of columns. A 2-dimensional array **a**, which contains three rows and four columns can be shown as below:

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

Thus, every element in array **a** is identified by an element name of the form **a[i][j],** where **a** is the name of the array, and **i** and **j** are the subscripts that uniquely identify each element in **a**.

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];
```

The above statement will take 4th element from the 3rd row of the array. You can verify it in the above diagram.

```
#include <iostream.h>
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;
}
```

When the above code is compiled and executed, it produces the following result:

```
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
```

Exercises:

1. Write C++ program, to find the minimum value in 1D array of 8 numbers.
2. Write C++ program, to find (search) X value in 1D array, and return the index of it's location.
3. Write C++ program, to split the odd numbers and even numbers of one 1D array into two 1D arrays:
4. Write C++ program, to read 15 numbers in 2D, 5 numbers per row, the print them.
5. Write C++ program, to read 4*4 2D-array, then find the summation of the array elements, finally print these elements.
6. Write C++ program, to read 3*4 2D-array, then find the summation of each row.
7. Write C++ program, to read 3*4 2D-array, then replace each value equal 5 with 0.
8. Write C++ program, to replace each element in the main diameter (diagonal) with zero.
9. Write C++ program, to convert 2D-array into 1D-array.
10. Write a C++ program, using function, to find if the array's elements are in order or not.
11. Write a C++ program, using function, to compute the number of zeros in the array.
12. Write a C++ program, using function, to find the value of array C from add array A and array B. C[i] = A [i] + B [i];
13. Write a C++ program, using function, to multiply the array elements by 2. A[i] = A [i] * 2;
14. Write a C++ program, using function, to reads temperatures over the 30 days and calculate the average of them.
15. Write a C++ program, using function, to merge two arrays in one array.
16. Write C++ program, to read 3*4 2D-array, then find the summation of each col.
17. Write C++ program, to replace each element in the second diameter (diagonal) with zero.

18. Write C++ program, to replace the elements of the main diameter with the elements of the second diameter.
19. Write C++ program, to find the summation of odd numbers in 2D-array.
20. Write C++ program, to find (search) X value in 2D-array, and return the index of it's location.
21. Write C++ program, to convert 1D-array that size [16] to 2D-array that size of [4] [4].
22. Write C++ program, to read A[n, n] of character, then find array B and array C, such that B contain only capital letters and C contain only small letters.
23. Write C++ program, to read A[n, n] of numbers, then put 10 instead each even positive number.
24. Write C++ program, to read A[n, n] of numbers, then put 10 instead each even positive number in the first diagonal.
25. Write C++ program, to read A[n, n] of numbers, then find the minimum number in array.
26. Write C++ program, to exchange row1 and row3 in 4*3 array.
27. Write C++ program, to exchange row0 with col3 in 4*4 array.
28. Write C++ program, to find the greatest number in the second diagonal, in 3*3 array.
29. Write C++ program, to read X[n], and rotate the elements to the left by one position.