EXPERIMENT NO 8

Arrays in C++

Objectives:

In this lab students will learn:

- To use the array to represent a set of related data items.
- Learn how to declare arrays, initialize arrays, and refer to the individual elements of arrays.

Equipment required:

- Dev-C++/Eclipse/Visual Studio installed in PC/Windows

DISCUSSION

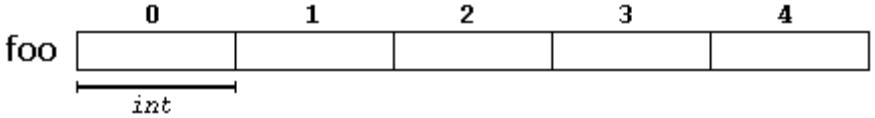
1. Pre-Lab

Array

A collection of individual values, all of the same data type, stored in adjacent memory locations. That means, for example, five values of type int can be declared as an array without having to declare 5 different variables (each with its identifier). Instead, using an array, the five int values are stored in contiguous memory locations, and all five can be accessed using the same identifier, with the proper index.

One Dimensional Array

An array with a single variable index. Using the array name together with an integral valued index in square brackets refers to the individual values. The first array element always has the subscript 0. The second array element has subscript 1, etc.
For example, an array containing 5 integer values of type int called foo could be represented as:



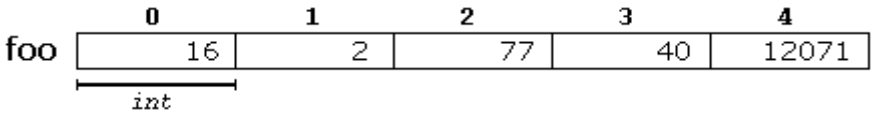
where each blank panel represents an element of the array. In this case, these are values of type int. These elements are numbered from 0 to 4, with being 0 the first and 4 the last; In C++, the first element in an array is always numbered with a zero (not a one), no matter its length.

Initializing Arrays

By default, arrays are left uninitialized. This means that none of its elements is set to any value; their contents are undetermined at the point the array is declared, but the elements in an array can be explicitly initialized to specific values when it is declared, by enclosing those initial values in braces {}. For example:

int foo [5] = {16, 2, 77, 40, 12071};

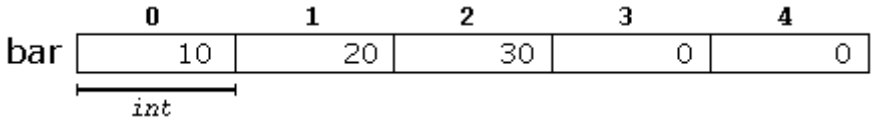
This statement declares an array that can be represented like this:



The number of values between braces {} shall not be greater than the number of elements in the array. For example, in the example above, foo was declared having 5 elements (as specified by the number enclosed in square brackets, [ ], and the braces { } contained exactly 5 values, one for each element. If declared with less, the remaining elements are set to their default values (which for fundamental types, means they are filled with zeroes). For example:

int bar [5] = {10, 20, 30};

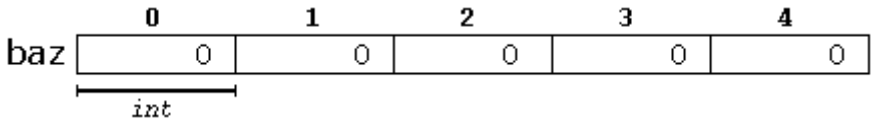
Will create an array like this:



The initializer can even have no values, just the braces:

int baz [5] = { };

This creates an array of five int values, each initialized with a value of zero:

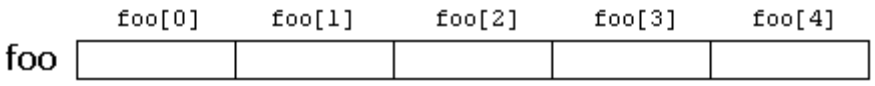


Accessing the values of an array

The values of any of the elements in an array can be accessed just like the value of a regular variable of the same type. The syntax is:

Name [index]

Following the previous examples in which foo had 5 elements and each of those elements was of type int, the name which can be used to refer to each element is the following:



For example, the following statement stores the value 75 in the third element of foo:

foo [2] = 75;

and, for example, the following copies the value of the third element of foo to a variable called x:

```
x = foo [2];
```

Therefore, the expression foo [2] is itself a variable of type int.

Implicit Array Sizing

It is possible to define an array without specifying its size, as long as you provide an initialization list. C++ automatically makes the array large enough to hold all the initialization values. For example, the following definition creates an array with five elements:

```
double ratings [ ] = {1.0, 1.5, 2.0, 2.5, 3.0};
```

Printing the Contents of an Array

Suppose we have the following array definition:

```
int numbers [5] = {10, 20, 30, 40, 50};
```

You now know that an array's name is seen as the array's beginning memory address. This explains why the following statement cannot be used to display the contents of array:

```
cout<< numbers <<endl; //Wrong!
```

When this statement executes, cout will display the array's memory address, not the array's contents. You must use a loop to display the contents of all the elements of array.

```
for (int count = 0; count < 5; count++)
cout<< numbers[count] <<endl;
```

2. Post-Lab (Lab Tasks)

1. Take ten inputs from the user and store them in an array, then split the array into two different arrays. Display both the original and split arrays

   Example:
   Original array:

58	24	13	15	63	9	8	81	1	78

   After spliting :

58	24	13	15	63

9	8	81	1	78

2. Write a program that takes data of five students from the user (names, roll numbers, and GPA). After storing the data, prompt the user to enter a roll number, search in the stored data and display the complete grade report for that roll number.

3. Take five inputs (temperature in Fahrenheit) from the user then convert the temperatures to Celsius and store them in another array. Display both the arrays in the form of a table having two columns, one for the temperature in Fahrenheit and the other in Celsius.

4. Write a program that takes five values from user, stores them in an array and displays the sum and average of these values.

5. Write a program that uses five arrays of size 5,
   - **Numbers** array that stores the values of its indexes
   - **Square** array that stores the squares of its indexes
   - **Cube** array that stores the cubes of its indexes
   - **Factorial** array that stores the factorials of its indexes
   - **Sum** array that stores the sum of corresponding indexes of the above 4 arrays

   The program should display all arrays and a single grand sum of all values of the 5 arrays.

6. Take ten numbers from the user and print the following:
   - number of odd numbers
   - number of even numbers
   - number of positive numbers
   - number of negative numbers
   - number of zeros
   - number of prime numbers
   - number of composite numbers
   - number of palindromes

END