9. Arrays and Strings

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.

9.1 Declaration of 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];
```

9.2 Initialization of array

You can initialize C++ array elements either one by one or using a single statement as follows:

```
double balance[5] = {1000.0, 2.0, 3.4, 17.0, 50.0};
```

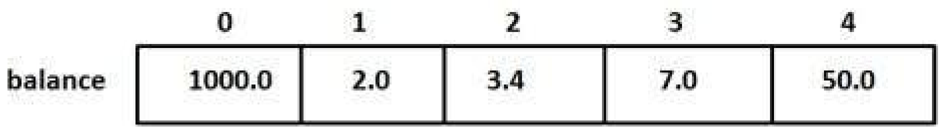
The number of values between braces { } can not be larger than the number of elements that we declare for the array between square brackets [ ]. Following is an example to assign a single element of the array: If you omit the size of the array, an array just big enough to hold the initialization is created. Therefore, if you write:

```
double balance[] = {1000.0, 2.0, 3.4, 17.0, 50.0};
```

You will create exactly the same array as you did in the previous example.

```
balance[4] = 50.0;
```

The above statement assigns element number 5th in the array a value of 50.0. Array with 4th index will be 5th, i.e., last element because all arrays have 0 as the index of their first element which is also called base index. Following is the pictorial representation of the same array we discussed above:



	0	1	2	3	4
balance	1000.0	2.0	3.4	7.0	50.0

9.3 Accessing elements of array

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.

9.4 I/O of arrays

Following is an example, which will use all the above-mentioned three concepts viz. declaration, assignment and accessing arrays:

```
#include <iostream.h>
#include<conio.h>

int main ()
{
   int n[ 10 ];

   for ( int i = 0; i < 10; i++ )
   {
      n[ i ] = i + 100;
   }
   cout << "Element" << “    “ << "Value" << endl;

   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
```

9.4.1 To reverse the elements of an array

The array elements are in a particular order and they can be printed by using loops as follows in reverse order as:

```
#include <iostream.h>
#include<conio.h>

void main ()
{
   int n[10] = {1,2,3,4,5,6,7,8,9,10};

   for (int i=9;i>=0;i--)
   {
      cout << n[i];
   }
}
```

The above code will print the array elements in reverse order.

The code to reverse the array elements is as follows:

```
#include <iostream.h>
#include<conio.h>

void main ()
{
   int n[10] = {1,2,3,4,5,6,7,8,9,10};
   int temp[10];

   int j=0;
   for (int i=9;i>=0;i--)
   {
      temp[i] = a[j];
      j++;
   }

   for(i=0;i<10;i++)
   {
      a[i] = temp[i];
   }
}
```

9.5 Passing arrays as arguments to a function

C++ does not allow to pass an entire array as an argument to a function. However, You can pass a pointer to an array by specifying the array's name without an index.

If you want to pass a single-dimension array as an argument in a function, you would have to declare function formal parameter in one of following three ways and all three