Data Structures and Arrays

You have seen so far that data structure uses some algorithms and need storage for storing values. For storing these values, programmers must need to have the fundamental data type's names such as char, int, float & double. As you know, these particular data types are beneficial for declaring variables, constants or a return type for a function; they are in control by the fact that, these types can store only a specific form of value at a time. For many applications, there may arise some circumstances where programmers need to have a single name to store multiple values. For processing such a large amount of data, programmers need powerful data types that would facilitate efficient storage, accessing and dealing with such data items. Using C++, you can implement the concept of arrays.

What are arrays?

The array is a fixed-size sequenced collection of variables belonging to the same data types. The array has adjacent memory locations to store values. Since the array provides a convenient structure for representing data, it falls under the category of the data structures in C. The syntax for declaring array are:

```
data_type array_name [array_size];
```

Following are the essential terminologies used for understanding the concepts of Arrays:

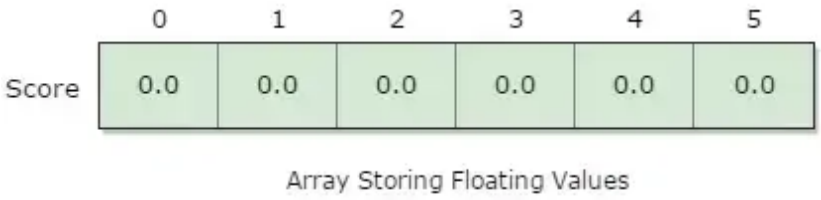
Element: Every item stored in an array is termed as an element

Index: each memory location of an element in an array is denoted by a numerical index which is used for identifying the element

Why Do You Need Arrays for Building a Specific Data Structure?

When a program works with many variables which hold comparable forms of data, then organizational and managerial difficulty quickly arise. If you are not using arrays, then the number of variables used will increase. Using the array, the number of variables reduces, i.e., you can use a single name for multiple values, you need to deal with its index values (starting from 0 to n).

So if the total run of each player is getting stored in separate variables, using arrays you can bring them all into one array having single name like: plrscore[11];



Array Storing Floating Values

Collecting Input Data in Arrays

Arrays are particularly helpful for making a collection of input data which arrive in random order. An excellent example will be vote counting: You can write a program which tallies the votes of a four-candidate in an election. (For your ease, you will say use the candidates' names as Cand 0, Cand 1, Cand 2, and Cand 3.) Votes arrive once at a time, where a vote for Candidate i is denoted by the number, i. So according to this example, two votes for Cand 3 followed by one vote for Cand 0 would appear:

```
3

3

0
and so on....
```

i.e., the structure will be:

```
int votes[4];
```

Basic Operations

There is some specific operation that can be performed or those that are supported by the array. These are:

- **Traversing:** It prints all the array elements one after another.
- **Inserting:** It adds an element at given index.
- **Deleting:** It is used to delete an element at given index.
- **Searching:** It searches for an element(s) using given index or by value.
- **Updating:** It is used to update an element at given index.
