**Arrays Y19 Acm 4th Feb 2020**

Intro:

An array is collection of items stored at contiguous memory locations. The idea is to store multiple items of same type together. This makes it easier to calculate the position of each element by simply adding an offset to a base value, i.e., the memory location of the first element of the array (generally denoted by the name of the array).

For example, an array of 10 [32-bit](https://en.wikipedia.org/wiki/32-bit) (4 bytes) integer variables, with indices 0 through 9, may be stored as 10 [words](https://en.wikipedia.org/wiki/Word_(data_type)) at memory addresses 2000, 2004, 2008, …, 2036, so that the element with index *i* has the address 2000 + (*i* × 4).[[4]](https://en.wikipedia.org/wiki/Array_data_structure#cite_note-4)

For simplicity, we can think of an array a fleet of stairs where on each step is placed a value (let’s say one of your friends). Here, you can identify the location of any of your friends by simply knowing the count of the step they are on.

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.

**Indexing:**

Array elements are numbered starting with zero, which may seem confusing at first but is an important detail for many programming languages. The first element is at position [0], the second is at [1], and so on. The position of each element is determined by its offset from the start of the array. The first element is at position [0] because it has no offset; the second element is at position [1] because it is offset one place from the beginning. The last position in the array is calculated by subtracting 1 from the array length. In this example, the last element is at position [4] because there are five elements in the array.

Define an Array

Arrays are declared similarly to other data types, but they are distinguished with brackets, [ and ]. When an array is declared, the type of data it stores must be specified. (Each array can store only one type of data.) After the array is declared, it must be created with the keyword new, just like working with objects. This additional step allocates space in the computer’s memory to store the array’s data. After the array is created, the values can be assigned. There are different ways to declare, create, and assign arrays. In the following examples that explain these differences, an array with five elements is created and filled with the values 19, 40, 75, 76, and 90. Note the different way each technique for creating and assigning elements of the array relates to setup().

## **Declaring Arrays**

To declare an array in C, a 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-dimensional* 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];

Here *balance* is a variable array which is sufficient to hold up to 10 double numbers.

## **Initializing Arrays**

You can initialize an array in C either one by one or using a single statement as follows −

double balance[5] = {1000.0, 2.0, 3.4, 7.0, 50.0};

The number of values between braces { } cannot be larger than the number of elements that we declare for the array between square brackets [ ].

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, 7.0, 50.0};

You will create exactly the same array as you did in the previous example. Following is an example to assign a single element of the array −

balance[4] = 50.0;

The above statement assigns the 5th element in the array with a value of 50.0. All arrays have 0 as the index of their first element which is also called the base index and the last index of an array will be total size of the array minus 1. Shown below is the pictorial representation of the array we discussed above −



## **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 the 10th element from the array and assign the value to salary variable. The following example Shows how to use all the three above mentioned concepts viz. declaration, assignment, and accessing arrays −