

## Arrays (One-dimensional array or 1D array)

An array is a collection of a **fixed number of components all of the same data type**. A one-dimensional array is an array in which the components are arranged in a list form. Individual elements are referred to using common name and unique index of the elements. The array itself is given name and its elements are referred to by their subscripts. In C++, an array is denoted as follows:

The general form for declaring a one-dimensional array is:

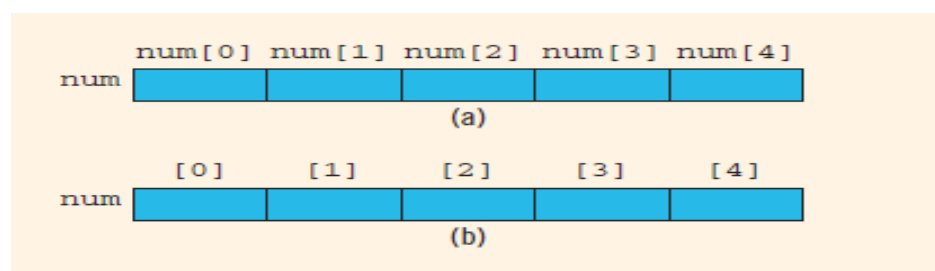
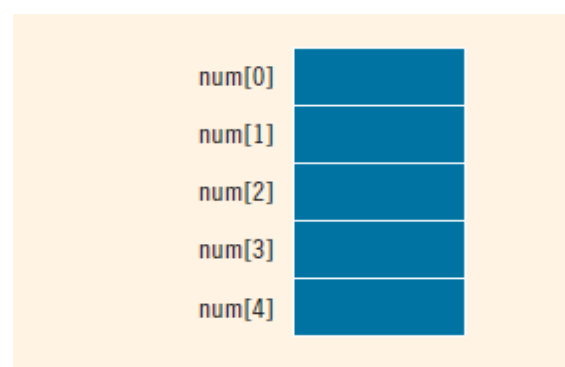
```
dataType arrayName[intExp];
```

in which intExp is any **constant** expression that evaluates to a **positive integer**. Also, intExp specifies the number of components in the array.

For example, the statement:

```
int num[5];
```

declares an array num of five components. Each component is of type **int**. The components are num[0], num[1], num[2], num[3], and num[4]. Figure below illustrates the array num.



## Accessing Array Components

The general form (syntax) used for accessing an array component is:

```
arrayName[indexExp]
```

in which `indexExp`, called the index, is any expression whose value is a **nonnegative integer and should be ranged from 0 to `intExp - 1`** (i.e. from 0 to the number of components in the array minus one).

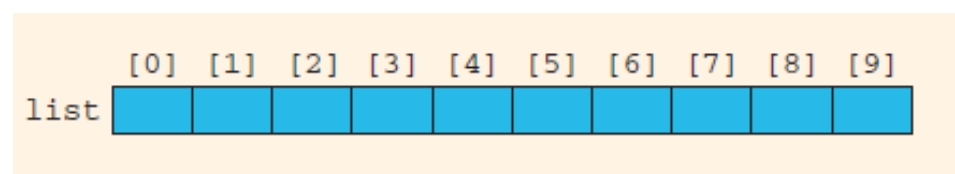
The index value specifies the position of the component in the array.

In C++, `[ ]` is an operator called the **array subscripting operator**. Moreover, in C++, the array index starts at 0.

Consider the following statement:

```
int list[10];
```

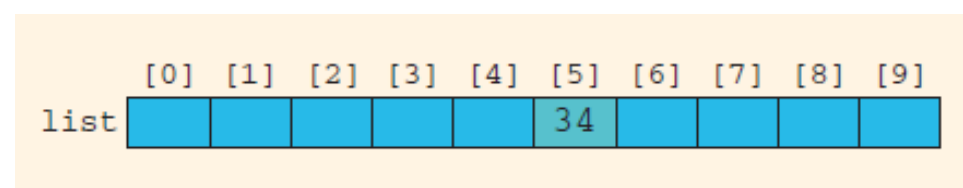
This statement declares an array `list` of 10 components. The components are `list[0]`, `list[1]`, . . . , `list[9]`. In other words, we have declared 10 variables (as shown in the following figure).



The assignment statement:

```
list[5] = 34;
```

stores (i.e. **assigns**) 34 in `list[5]`, which is the sixth component of the array `list`:



**Suppose**  $i$  is an `int` variable. Then, the assignment statement:

```
list[3] = 63;
```

is equivalent to the assignment statements:

```
i = 3;
```

```
list[i] = 63;
```

**If**  $i$  is 4, then the assignment statement:

```
list[2 * i - 3] = 58;
```

stores 58 in `list[5]` because  $2 * i - 3$  evaluates to 5. The index expression is evaluated first, giving the position of the component in the array.

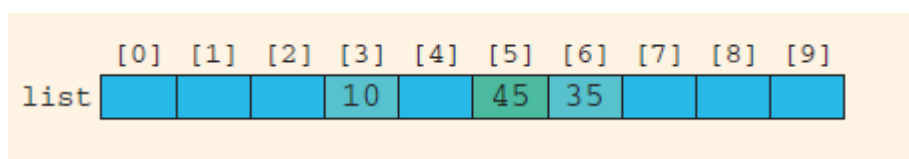
**Next**, consider the following statements:

```
list[3] = 10;
```

```
list[6] = 35;
```

```
list[5] = list[3] + list[6];
```

The first statement stores 10 in `list[3]`, the second statement stores 35 in `list[6]`, and the third statement adds the contents of `list[3]` and `list[6]` and stores the result in `list[5]` (see Figure below).



### Processing One-Dimensional Arrays

Some of the basic operations performed on a one-dimensional array are **initializing, inputting data, outputting data stored in an array, and finding the largest and/or smallest element**. Moreover, if the data is numeric, some other basic operations are finding the sum and average of the elements of the array. Each of these operations requires the ability to step through the elements of the

array. This is easily accomplished using a loop. For example, suppose that we have the following statements:

```
int list[100]; //list is an array of size 100
int i;
```

The following **for** loop steps through each element of the array `list`, starting at the first element of `list`:

```
for (i = 0; i < 100; i++) //Line 1
//process list[i] //Line 2
```

If processing the list requires inputting data into `list`, the statement in Line 2 takes the form of an input statement, such as the `cin` statement. For example, the following statements read 100 numbers from the keyboard and store the numbers in `list`:

```
main()
{
    int list[100];
    int i;
    for (i = 0; i < 100; i++)
        cin >> list[i];
}
```

### Initialize One Dimensional Array in C++

Here is an example, declaring and initializing values to the array name `arr` of type `int`, containing 10 elements

```
int arr[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
```

## C++ One Dimensional Array Example

Here are some example program, demonstrating one dimensional array in C++

```
/* C++ One Dimensional Array */
```

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

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

```
void main()
```

```
{
```

```
    int arr[5] = {1, 2, 3, 4, 5};
```

```
    int i;
```

```
    for(i=0; i<5; i++)
```

```
    {
```

```
        cout<< arr[i] << endl;
```

```
    }
```

```
    getch();
```

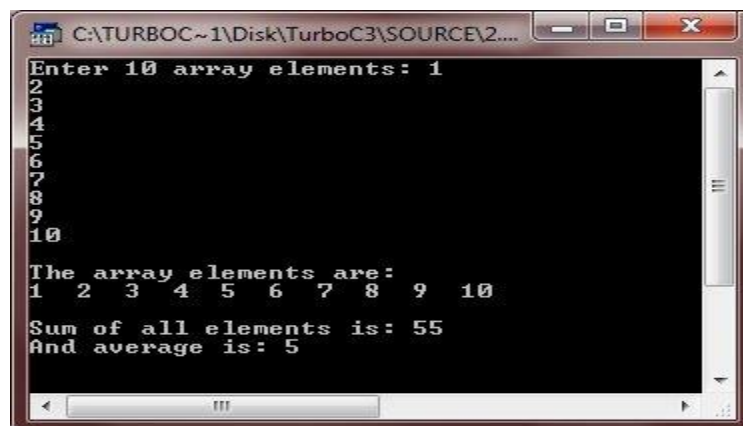
```
}
```

Here is another C++ example, also demonstrating one dimension array in C++

```
/* C++ One Dimensional Array */
```

```
#include<iostream.h>
#include<conio.h>
void main()
{
    int arr[10];
    int i;
    int sum=0, avg=0;
    cout <<"Enter 10 array elements: ";
    for(i=0; i < 10; i++)
    {
        cin >> arr[i];
        sum = sum + arr[i];
    }
    cout<<"\n\nThe array elements are: \n";
    for(i=0; i < 10; i++)
    {
        cout << arr[i] <<" ";
    }
    cout<<"\n\nSum of all elements is: " << sum << endl;
    avg = sum/10;
    cout<<"And average is: " << avg;
}
```

Here is the sample run of the above C++ program:



```
C:\TURBOC~1\Disk\TurboC3\SOURCE\2...
Enter 10 array elements: 1
2
3
4
5
6
7
8
9
10
The array elements are:
1 2 3 4 5 6 7 8 9 10
Sum of all elements is: 55
And average is: 5
```