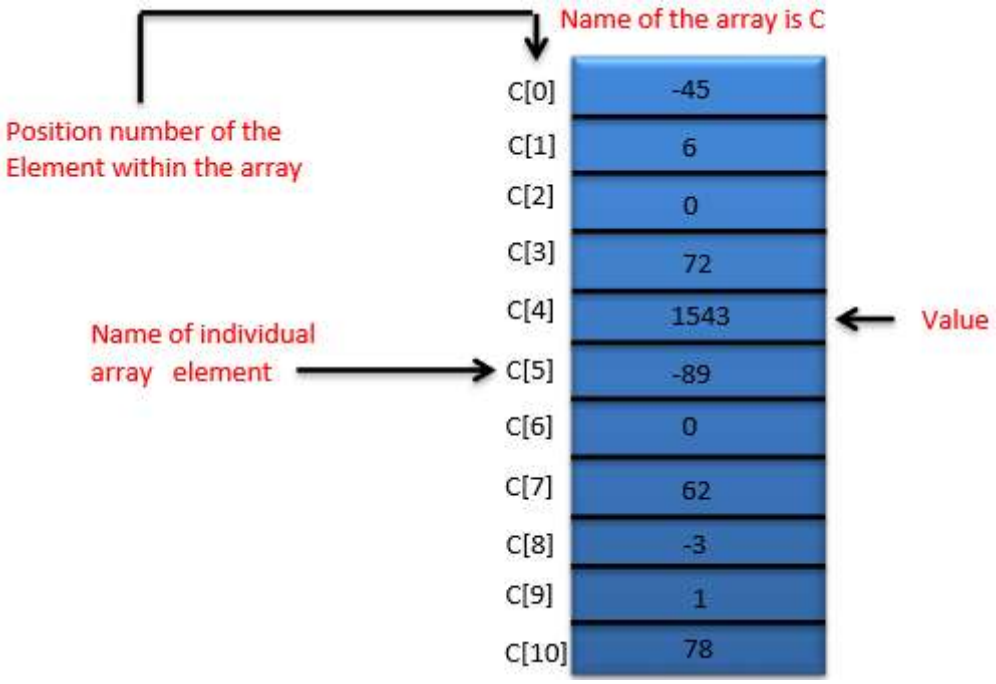
# Arduino - Arrays

An array is a consecutive group of memory locations that are of the same type. To refer to a particular location or element in the array, we specify the name of the array and the position number of the particular element in the array.

The illustration given below shows an integer array called C that contains 11 elements. You refer to any one of these elements by giving the array name followed by the particular element's position number in square brackets ([]). The position number is more formally called a subscript or index (this number specifies the number of elements from the beginning of the array). The first element has subscript 0 (zero) and is sometimes called the zeros element.

Thus, the elements of array C are C[0] (pronounced "C sub zero"), C[1], C[2] and so on. The highest subscript in array C is 10, which is 1 less than the number of elements in the array (11). Array names follow the same conventions as other variable names.




A subscript must be an integer or integer expression (using any integral type). If a program uses an expression as a subscript, then the program evaluates the expression to determine the subscript. For example, if we assume that variable a is equal to 5 and that variable b is equal to 6, then the statement adds 2 to array element C[11].

A subscripted array name is an lvalue, it can be used on the left side of an assignment, just as non-array variable names can.

Let us examine array C in the given figure, more closely. The name of the entire array is C. Its 11 elements are referred to as C[0] to C[10]. The value of C[0] is -45, the value of C[1] is 6, the value of C[2] is 0, the value of C[7] is 62, and the value of C[10] is 78.

To print the sum of the values contained in the first three elements of array C, we would write −

```
Serial.print (C[ 0 ] + C[ 1 ] + C[ 2 ] );
```

To divide the value of C[6] by 2 and assign the result to the variable x, we would write −

```
x = C[ 6 ] / 2;
```

## Declaring Arrays

Arrays occupy space in memory. To specify the type of the elements and the number of elements required by an array, use a declaration of the form −

```
type arrayName [ arraySize ] ;
```

The compiler reserves the appropriate amount of memory. (Recall that a declaration, which reserves memory is more properly known as a definition). The arraySize must be an integer constant greater than zero. For example, to tell the compiler to reserve 11 elements for integer array C, use the declaration −

```
int C[ 12 ]; // C is an array of 12 integers
```

Arrays can be declared to contain values of any non-reference data type. For example, an array of type string can be used to store character strings.

## Examples Using Arrays

This section gives many examples that demonstrate how to declare, initialize and manipulate arrays.

### Example 1: Declaring an Array and using a Loop to Initialize the Array's Elements

The program declares a 10-element integer array **n**. Lines a–b use a **For** statement to initialize the array elements to zeros. Like other automatic variables, automatic arrays are not implicitly initialized to zero. The first output statement (line c) displays the column headings for the columns printed in the subsequent for statement (lines d–e), which prints the array in tabular format.

**Example**

```
int n[ 10 ] ; // n is an array of 10 integers

void setup () {

}

void loop () {
   for ( int i = 0; i < 10; ++i ) // initialize elements of array n to 0 {
      n[ i ] = 0; // set element at location i to 0
      Serial.print (i) ;
```

```
      Serial.print ('\r') ;
   }
   for ( int j = 0; j < 10; ++j ) // output each array element's value {
      Serial.print (n[j]) ;
      Serial.print ('\r') ;
   }
}
```

**Result** − It will produce the following result −

| Element | Value |
|---|---|
| 0 | 0 |
| 1 | 0 |
| 2 | 0 |
| 3 | 0 |
| 4 | 0 |
| 5 | 0 |
| 6 | 0 |
| 7 | 0 |
| 8 | 0 |
| 9 | 0 |

### Example 2: Initializing an Array in a Declaration with an Initializer List

The elements of an array can also be initialized in the array declaration by following the array name with an equal-to sign and a brace-delimited comma-separated list of initializers. The program uses an initializer list to initialize an integer array with 10 values (line a) and prints the array in tabular format (lines b–c).

**Example**

```
// n is an array of 10 integers
int n[ 10 ] = { 32, 27, 64, 18, 95, 14, 90, 70, 60, 37 } ;

void setup () {

}

void loop () {
   for ( int i = 0; i < 10; ++i ) {
      Serial.print (i) ;
      Serial.print (‘\r’) ;
   }
   for ( int j = 0; j < 10; ++j ) // output each array element's value {
      Serial.print (n[j]) ;
      Serial.print (‘\r’) ;
   }
}
```

**Result** − It will produce the following result −

| Element | Value |
|---|---|
| 0 | 32 |
| 1 | 27 |
| 2 | 64 |
| 3 | 18 |
| 4 | 95 |
| 5 | 14 |
| 6 | 90 |
| 7 | 70 |
| 8 | 60 |
| 9 | 37 |

### Example 3: Summing the Elements of an Array

Often, the elements of an array represent a series of values to be used in a calculation. For example, if the elements of an array represent exam grades, a professor may wish to total the elements of the array and use that sum to calculate the class average for the exam. The program sums the values contained in the 10-element integer array **a.**

**Example**

```
const int arraySize = 10; // constant variable indicating size of array
int a[ arraySize ] = { 87, 68, 94, 100, 83, 78, 85, 91, 76, 87 };
int total = 0;

void setup () {
```

```
}
void loop () {
   // sum contents of array a
   for ( int i = 0; i < arraySize; ++i )
      total += a[ i ];
   Serial.print ("Total of array elements : ") ;
   Serial.print(total) ;
}
```

**Result** − It will produce the following result −

```
Total of array elements: 849
```

Arrays are important to Arduino and should need a lot more attention. The following important concepts related to array should be clear to a Arduino −

| S.NO. | Concept & Description |
|---|---|
| 1 | Passing Arrays to Functions<br>To pass an array argument to a function, specify the name of the array without any brackets. |
| 2 | Multi-Dimensional Arrays<br>Arrays with two dimensions (i.e., subscripts) often represent tables of values consisting of information arranged in rows and columns. |