https://www.arduino.cc/reference/en/language/variables/data-types/array/

[Reference](https://www.arduino.cc/reference/en) > [Language](https://www.arduino.cc/reference/en#language) > [Variables](https://www.arduino.cc/reference/en#variables) > [Data types](https://www.arduino.cc/reference/en#data-types) > Array

# array

[Data Types]

### Description

An array is a collection of variables that are accessed with an index number. Arrays in the C++ programming language Arduino sketches are written in can be complicated, but using simple arrays is relatively straightforward.

### Creating (Declaring) an Array

All of the methods below are valid ways to create (declare) an array.

int myInts[6];

int myPins[] = {2, 4, 8, 3, 6};

int mySensVals[6] = {2, 4, -8, 3, 2};

char message[6] = "hello";

You can declare an array without initializing it as in myInts.  
In myPins we declare an array without explicitly choosing a size. The compiler counts the elements and creates an array of the appropriate size.  
Finally you can both initialize and size your array, as in mySensVals. Note that when declaring an array of type char, one more element than your initialization is required, to hold the required null character.

### Accessing an Array

Arrays are zero indexed, that is, referring to the array initialization above, the first element of the array is at index 0, hence

mySensVals[0] == 2, mySensVals[1] == 4, and so forth.

It also means that in an array with ten elements, index nine is the last element. Hence:

int myArray[10]={9, 3, 2, 4, 3, 2, 7, 8, 9, 11};

// myArray[9] contains 11

// myArray[10] is invalid and contains random information (other memory address)

For this reason you should be careful in accessing arrays. Accessing past the end of an array (using an index number greater than your declared array size - 1) is reading from memory that is in use for other purposes. Reading from these locations is probably not going to do much except yield invalid data. Writing to random memory locations is definitely a bad idea and can often lead to unhappy results such as crashes or program malfunction. This can also be a difficult bug to track down.

Unlike BASIC or JAVA, the C++ compiler does no checking to see if array access is within legal bounds of the array size that you have declared.

### To assign a value to an array:

mySensVals[0] = 10;

### To retrieve a value from an array:

x = mySensVals[4];

### Arrays and FOR Loops

Arrays are often manipulated inside for loops, where the loop counter is used as the index for each array element. For example, to print the elements of an array over the serial port, you could do something like this:

for (byte i = 0; i < 5; i = i + 1) {

Serial.println(myPins[i]);

}