Arduino - Strings

Strings are used to store text. They can be used to display text on an LCD or in the Arduino IDE Serial Monitor window. Strings are also useful for storing the user input. For example, the characters that a user types on a keypad connected to the Arduino.

There are two types of strings in Arduino programming −

- Arrays of characters, which are the same as the strings used in C programming.
- The Arduino String, which lets us use a string object in a sketch.

In this chapter, we will learn Strings, objects and the use of strings in Arduino sketches. By the end of the chapter, you will learn which type of string to use in a sketch.

String Character Arrays

The first type of string that we will learn is the string that is a series of characters of the type **char**. In the previous chapter, we learned what an array is; a consecutive series of the same type of variable stored in memory. A string is an array of char variables.

A string is a special array that has one extra element at the end of the string, which always has the value of 0 (zero). This is known as a "null terminated string".

String Character Array Example

This example will show how to make a string and print it to the serial monitor window.

Example

```
void setup() {
   char my_str[6]; // an array big enough for a 5 character string
   Serial.begin(9600);
   my_str[0] = 'H'; // the string consists of 5 characters
   my_str[1] = 'e';
   my_str[2] = 'l';
   my_str[3] = 'l';
   my_str[4] = 'o';
   my_str[5] = 0; // 6th array element is a null terminator
   Serial.println(my_str);
}

void loop() {

}
```

The following example shows what a string is made up of; a character array with printable characters and 0 as the last element of the array to show that this is where the string ends. The string can be printed out

to the Arduino IDE Serial Monitor window by using **Serial.println()** and passing the name of the string.

This same example can be written in a more convenient way as shown below −

Example

```
void setup() {
   char my_str[] = "Hello";
   Serial.begin(9600);
   Serial.println(my_str);
}

void loop() {

}
```

In this sketch, the compiler calculates the size of the string array and also automatically null terminates the string with a zero. An array that is six elements long and consists of five characters followed by a zero is created exactly the same way as in the previous sketch.

Manipulating String Arrays

We can alter a string array within a sketch as shown in the following sketch.

Example

```
void setup() {
   char like[] = "I like coffee and cake"; // create a string
   Serial.begin(9600);
   // (1) print the string
   Serial.println(like);
   // (2) delete part of the string
   like[13] = 0;
   Serial.println(like);
   // (3) substitute a word into the string
   like[13] = ' '; // replace the null terminator with a space
   like[18] = 't'; // insert the new word
   like[19] = 'e';
   like[20] = 'a';
   like[21] = 0; // terminate the string
   Serial.println(like);
}

void loop() {

}
```

Result

```
I like coffee and cake
I like coffee
I like coffee and tea
```

The sketch works in the following way.

Creating and Printing the String

In the sketch given above, a new string is created and then printed for display in the Serial Monitor window.

Shortening the String

The string is shortened by replacing the 14th character in the string with a null terminating zero (2). This is element number 13 in the string array counting from 0.

When the string is printed, all the characters are printed up to the new null terminating zero. The other characters do not disappear; they still exist in the memory and the string array is still the same size. The only difference is that any function that works with strings will only see the string up to the first null terminator.

Changing a Word in the String

Finally, the sketch replaces the word "cake" with "tea" (3). It first has to replace the null terminator at like[13] with a space so that the string is restored to the originally created format.

New characters overwrite "cak" of the word "cake" with the word "tea". This is done by overwriting individual characters. The 'e' of "cake" is replaced with a new null terminating character. The result is that the string is actually terminated with two null characters, the original one at the end of the string and the new one that replaces the 'e' in "cake". This makes no difference when the new string is printed because the function that prints the string stops printing the string characters when it encounters the first null terminator.

Functions to Manipulate String Arrays

The previous sketch manipulated the string in a manual way by accessing individual characters in the string. To make it easier to manipulate string arrays, you can write your own functions to do so, or use some of the string functions from the **C** language library.

> Given below is the list Functions to Manipulate String Arrays

The next sketch uses some C string functions.

Example

```
void setup() {
   char str[] = "This is my string"; // create a string
   char out_str[40]; // output from string functions placed here
```

```
   int num; // general purpose integer
   Serial.begin(9600);

   // (1) print the string
   Serial.println(str);

   // (2) get the length of the string (excludes null terminator)
   num = strlen(str);
   Serial.print("String length is: ");
   Serial.println(num);

   // (3) get the length of the array (includes null terminator)
   num = sizeof(str); // sizeof() is not a C string function
   Serial.print("Size of the array: ");
   Serial.println(num);

   // (4) copy a string
   strcpy(out_str, str);
   Serial.println(out_str);

   // (5) add a string to the end of a string (append)
   strcat(out_str, " sketch.");
   Serial.println(out_str);
   num = strlen(out_str);
   Serial.print("String length is: ");
   Serial.println(num);
   num = sizeof(out_str);
   Serial.print("Size of the array out_str[]: ");
   Serial.println(num);
}

void loop() {

}
```

Result

```
This is my string
String length is: 17
Size of the array: 18
This is my string
This is my string sketch.
String length is: 25
Size of the array out_str[]: 40
```

The sketch works in the following way.

Print the String

The newly created string is printed to the Serial Monitor window as done in previous sketches.

Get the Length of the String

The strlen() function is used to get the length of the string. The length of the string is for the printable characters only and does not include the null terminator.

The string contains 17 characters, so we see 17 printed in the Serial Monitor window.

Get the Length of the Array

The operator sizeof() is used to get the length of the array that contains the string. The length includes the null terminator, so the length is one more than the length of the string.

sizeof() looks like a function, but technically is an operator. It is not a part of the C string library, but was used in the sketch to show the difference between the size of the array and the size of the string (or string length).

Copy a String

The strcpy() function is used to copy the str[] string to the out_num[] array. The strcpy() function copies the second string passed to it into the first string. A copy of the string now exists in the out_num[] array, but only takes up 18 elements of the array, so we still have 22 free char elements in the array. These free elements are found after the string in memory.

The string was copied to the array so that we would have some extra space in the array to use in the next part of the sketch, which is adding a string to the end of a string.

Append a String to a String (Concatenate)

The sketch joins one string to another, which is known as concatenation. This is done using the strcat() function. The strcat() function puts the second string passed to it onto the end of the first string passed to it.

After concatenation, the length of the string is printed to show the new string length. The length of the array is then printed to show that we have a 25-character long string in a 40 element long array.

Remember that the 25-character long string actually takes up 26 characters of the array because of the null terminating zero.

Array Bounds

When working with strings and arrays, it is very important to work within the bounds of strings or arrays. In the example sketch, an array was created, which was 40 characters long, in order to allocate the memory that could be used to manipulate strings.

If the array was made too small and we tried to copy a string that is bigger than the array to it, the string would be copied over the end of the array. The memory beyond the end of the array could contain other important data used in the sketch, which would then be overwritten by our string. If the memory beyond the end of the string is overrun, it could crash the sketch or cause unexpected behavior.