# Arrays II

**T1** Chapters: **24-26**

So far, we have worked with string literals without really understanding what they actually are. To some extent we seem to be using them the same way we have been using integers or float or character literals. However, the type of the variable that stores a string literal is differs from that of the other types.

## String

A string literal has the type const char\*, meaning when a variable is declared with the intent of holding a string literal, this should be its type.

const char\* str = "foo bar";

**NOTE**

It is a good idea to get into the habit of reading variable declarations from the right to left in C/C++.

For instance, the above declaration is to be read as *pointer to constant character*. The const, here is a qualifier, which forbids modifications to the contents of str.

We'll get into what a pointer is later, but for now and for the purpose of variables that hold string literals, like in the case above, just remember that it points to the first character of the string.

String literals in C/C++ always terminate or end with the escape sequence or whitespace character \0 or NUL. This becomes important in the context of the relationship between arrays and strings.

## Arrays and Strings

A string is made up of characters, and thus can be viewed as a special subset of an array. More specifically, **a string is an array of characters that end with a NUL character**. The last part is important because as we will see soon, many functions provided in the standard library that work with strings depend on the existence of a NUL character in the string.

**NOTE**

When declaring a character array to represent a string, there must be at least one extra space for the NUL character.

char hello[6] = "Hello";
  
char hello[6] = {'H', 'e', 'l', 'l', 'o'}; // alternative declaration
  
  
char world[6] = "World";
  
hello = world; // ERROR: Incorrect. Array is not assignable

## cstring

The cstring library provides several functions that enable a user to work with strings conveniently. Some of the most commonly used functions are:

1. strlen - Number of characters in a string, not including the NUL character.
2. strcmp - Compare two strings in lexicographic order.
3. strcat - Join two strings.
4. strcpy - Copy string to another string.

See the reference website for more details.

**NOTE**

In the real world, these functions are meant to be used when working with long strings and not short ones, this is because they compute the length of the string every time they are used, which can be inefficient when calling this function quite frequently. However, for purposes of learning we are going to ignore this fact.

Except for the existence of a NUL character signifying the end, character arrays are otherwise the same as arrays of any other type. It is important to recognize the distinction between a string literal and a character array. The former will contain a NUL character at the end while the latter may or may not depending on how and where it is being used. However, it is custom to include a NUL character at the end just in case it might used in a similar context to a string literal.

char hello[6] = "Hello";
  
const char\* phello = hello;
  
const char\* phello = &hello[0];

In the code snippet above, the last two declarations are equivalent. When an array variable is declared, the variable does not hold the elements of the array, instead it holds what is called a **pointer** or the **address** to the start of the array. This is made explicit in the last declaration where the & operator is used to take the address of the first element in hello. Thus, phello, whose type is const char\* or *pointer to constant character* holds the address or pointer to the start of hello and is called a **pointer variable**.

# Exercises

1. Write a program that copies a string into another, overwriting the original. (E)
2. Write a program that inserts a string into the middle of another string. (M)
3. Write a program that trims the prefix and suffix of a string. (M)
4. Write a program that implements the functionality of the strcat and strcpy functions using just plain loops. (M)
5. Write a program that computes the CRC8 hash of a user provided string. (H)

# Homework

1. Read **T1** Chapters **24-26**
2. Implement the above programs using pointers, instead of the subscript operator. (M)