Strings

By: **Annamalai A**

A string is a combination of multiple characters, and in `C`, there is no specific data type for strings. However, you can create strings using the given statement.

In `C`, a string is nothing but an array of characters.

```
char message[] = "Hello!"; // Method 1
char message[10] = {'H', 'e', 'l', 'l', 'o', '!', '\0'};
```

Internally, the string is stored as an array of characters: `{'H', 'e', 'l', 'l', 'o', '!', '\0'}`. The placeholder used for printing strings is `%s`.

However, you may not be able to do this:

```
char message[10];
message = "Hello!";
```

The `string` library

This language, although does not have a data type for strings, it provides a set of functions compatible with strings. It is the `string` library. The functons in it are:

Function	Purpose
`strlen(src)`	Returns the length of string excluding `\0`
`strcopy(dest, src)`	Copies a string from source to destination
`strncopy(dest, src)`	Copies the first `n` bytes of a string from source to destination
`strcat(str1, str2)`	Concatinate (Join) two strings
`strcmp(s1, s2)`	Compare two strings (returns 0 if equal)
`strchr(str, ch)`	Finds the first occurance of the passed character
`strstr(str, sub)`	Finds the first occurance of a substring

Function	Purpose
`strrev(str)`	Reverses a string

Now, to explain the problem above (why we may not declare it that way), string is basically an *array* in `C`. But there is another way to do it, which is to use `strcpy()`.

```
char message[10];
strcpy(message, "Hello!");
```

`\0` Character

This is a special character called **null character** in this language used to mark the end of strings. It is not visible on the output or while processing, but it tells the compiler where to stop while reading. The compiler stops when it reads this character.

This character has an ASCII value 0.

When we don't provide this character while defining strings, the compiler reads till the end of file, and eventually find a zero byte in memory and stops to print a garbage value. Hence, it is very important while using strings in `C`.

String Input

Although `scanf` can be used to input strings, it terminates at a new line, space or tab, and strings are meant for writing long text with such characters. Hence, it does not read multiple words.

That is why, there two other functions which work really well with strings — `gets` and `fgets`.

`gets` Function

This function gets a string input from the user including spaces. However it stops at a newline. The syntax is as follows: `gets(<str>)`.

`fgets` Function

This function gets a string input from the user including spaces and newline upto `size - 1` bytes. It is a safer version `gets`.

The syntax is as follows: `fgets(<str>, <size>, stdin)`.