

# Learning Objectives: String Functions

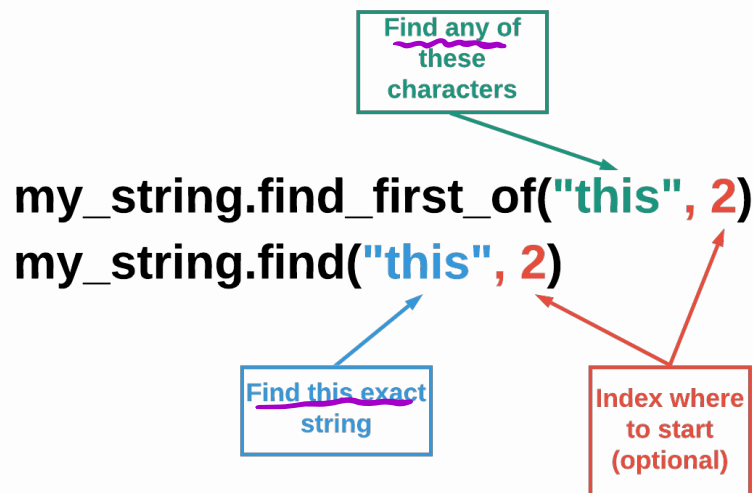
---

- **Identify the functions and applications of the following string functions:**
  - `find_first_of()` & `find_last_of()`
  - `push_back()` & `insert()`
  - `pop_back()` & `erase()`
  - `replace()`
  - `append()`
  - `toupper()` & `tolower()`

# Find First Of & Last Of

## The `find_first_of()` Function

The `find_first_of()` function works similarly to how the `find()` function does. However, the `find_first_of()` function will search for **any** matching characters specified. For example, given the string "this is his string", `my_string.find_first_of("his")` will return 1 because the character h within his appears first at index number 1. Like the `find()` function, you can optionally specify an index number to direct the system where to start searching.



[.guides/img/FindFirstOf](#)

### ▼ The `find_last_of()` function

You can use the `find_last_of()` function to search for a set of characters in a string that occur last. Here is an example, given the string "this is his string", `my_string.find_last_of("his")` will return 15 because the i in his occurs last at index 15. If you don't want the system to search the whole string, you can specify an index as a second parameter to direct the system where to start searching.

```
string string1 = "The brown dog jumps over the lazy fox.";
string string2 = "brown";

cout << string1.find_first_of(string2) << endl; → 4
```

challenge

## What happens if you:

- Set string2 to "wornb"? → 4
- Change the cout statement to `string1.find_first_of(string2, 14)`? → 20
- Set string2 to "axe" and change the cout statement to `string1.find_first_of(string2, 34)`? → 36
- Set string2 to "i" and change the cout statement to `string1.find_first_of(string2)`? →

**Remember** that 18446744073709551615 is equivalent to -1 which also means not found.

# Push Back & Insert

## The push\_back() Function

In a previous module, you were introduced to vectors. The `push_back()` function works the same way in strings as it does in vectors. It adds a specific character to the end of the string.

```
string my_string = "Today is Satur";  
my_string.push_back('d');  
  
cout << my_string << endl;
```

*single quotes*  
*→ Today is Saturd*

challenge

### What happens if you:

- add the code `my_string.push_back("ay");` below `my_string.push_back('d');`? *→ Today is Saturday*
- replace `my_string.push_back("ay");` with `my_string.push_back('a');`?
- add the code `my_string.push_back('y');` below `my_string.push_back('a');`?

## The insert() Function

Unfortunately, the `push_back()` function cannot add multiple characters (string) to an existing string. However, the `insert()` function can. Unlike many functions where specifying the starting index number is optional, doing so is necessary for `insert()` to work.

Note that the index specification comes **before** the string you want the system to add. For example, `my_string.insert(0, "abc")` will add the string `abc` to the 0th index which is also the beginning of the string. To add to the end of the string, you can use `my_string.length()`. Note that you do not need to subtract 1 from `my_string.length()` because the system will add characters starting at the index **after** the last character of the string.

```
string my_string = "Today is Satur";  
my_string.insert(my_string.length(), "day");  
  
cout << my_string << endl; → Today is Saturday
```

challenge

### What happens if you:

- replace `my_string.insert(my_string.length(), "day");` with `my_string.insert(0, "day");`? → dayToday is Satur
- change the code back to `my_string.insert(my_string.length(), "day");` and add `my_string.insert(9, "a good ");` below it? → Today is a goodSatur

# Pop Back & Erase

## The pop\_back() Function

The `pop_back()` function removes a single character from the end of a string. You do not include anything within parentheses `()`.

```
string my_string = "Today is my birthday!";  
my_string.pop_back();  
  
cout << my_string << endl; → Today is my birthday
```

## The erase() Function

The `erase()` function can remove multiple characters from a string or the entire string itself. To remove the whole string, leave the parentheses `()` empty. Alternatively, you can specify one index number to remove all characters starting at that index to the end of the string. Specify two index numbers to start at an index and erase a certain number of characters at that index forward.

```
string my_string = "Today is my birthday!";  
my_string.erase(my_string.length()-1);  
  
cout << my_string << endl; → Today is my birthday
```

challenge

### What happens if you:

- Replace `my_string.erase(my_string.length()-1);` with `my_string.erase(12);`? → Today is my
- Replace `my_string.erase(12);` in your current code with `my_string.erase(12, 5);`? → Today is my day!
- Replace `my_string.erase(12, 5);` in your current code with `my_string.erase();` → Remove entire string

▼ Why do I see Command was successfully executed.?

When Codio does not detect anything being printed to the console, you will see the message `Command was successfully executed.` displayed. The command `my_string.erase()` causes the entire string to become empty and since an empty string has no output, you will see `Command was successfully executed.` instead.

# Replace

---

## The replace() Function

The `replace()` function is really a combination of the `erase()` and `insert()` functions. Let's take a look at an example.

```
my_string.replace(1, 2, "3")
```

There are three parameters of interest within `replace()` above. The 1 represents the index at which we want to start erasing. The 2 tells the system how many characters to erase starting at index 1. And the "3" is the string that the system will insert into the string at index 1.

```
string string1 = "Hello world!";
string string2 = "Codio.";
string string3 = string1.replace(6, 5, string2);
// erase all characters from index 6 plus 5 chars to the right
// in string1
// then insert string2 at index 6 within string1

cout << string3 << endl; → Hello Codio.
```

challenge

### What happens if you:

- Change `string3` to `string1.replace(6, 6, string2)`? → Hello Codio.
- Change `string3` to `string1.replace(2, 3, "y")`? → Hey world!



# Append

## The append() function

An alternative way to **concatenate** or combine strings is to use the `append()` function. The `append()` function works in the same way as adding literal strings together using the + operator.

```
string a = "High";  
string b = " Five";  
  
cout << a.append(b) << endl; → High Five
```

challenge

### What happens if you:

- Change the cout statement to `a.append(b + "!")`? → High Five!
- Change the cout statement to `a.append("Five" + "!")`? → error
- Change the cout statement to `"High" + " Five" + "!"`? → error
- Change the cout statement to `a + " Five" + "!"`? → High Five!
- Change the cout statement back to `a.append(b)` and replace string `b = " Five";` with `int b = 5;` → error

important

## IMPORTANT

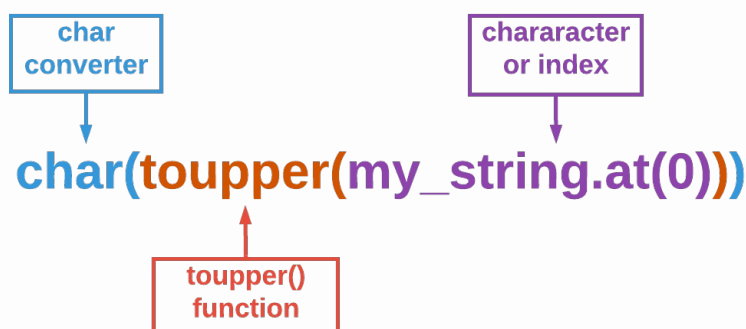
**NOTE** that the `append()` function is exclusively for strings. Thus, you cannot include other data types like ints when using `append()` unless they are converted to strings first. Additionally, when using the + operator to combine two strings together, make sure that at least one of the strings is a string variable. Otherwise, the system will think you are trying to manipulate a string literal, which is not allowed.

# Uppercase & Lowercase

## The toupper() Function

The `toupper()` function returns the uppercase form of a specified character. Insert the character you wish to convert as a parameter.

**Note** that by default, `toupper()` returns the **ASCII** representation of the uppercase letter. Thus, you'll need `char` to convert the ASCII code into alphabetical.



[.guides/img/StringToUpper](#)

```
string my_string = "the big brown dog";  
  
cout << char(toupper(my_string.at(0))) << endl; → T
```

challenge

### What happens if you:

- Change the cout statement to `char(toupper(my_string.at(my_string.length()-1)))?` → `g`
- Change the cout statement to `char(toupper('t'))?` → `T`
- Change the cout statement to `char(toupper(my_string))?` → error  
only convert one character

## The tolower() Function

The `tolower()` function returns the lowercase form of a specified character. It has all of the technicalities that the `toupper()` function has.

```
string my_string = "THE BIG BROWN DOG";  
  
cout << char(tolower(my_string.at(1))) << endl; →h
```

challenge

## What happens if you:

- Change the cout statement to `char(tolower(my_string.at(my_string.length()-1)))?` →g
- Change the cout statement to `char(tolower('B'))?` →b
- Change the cout statement to `char(tolower('%'))?` →%