Strings

- Strings are used for storing text/characters.
- For example, "Hello World" is a string.
- A string variable contains a collection of characters surrounded by double quotes:

Example: Create a variable of type string and assign it a value:

```
string greeting = "Hello";
```

To use strings, you must include an additional header file in the source code, the **<string>** library:

```
// Include the string library
#include <string>

// Create a string variable
string greeting = "Hello";
```

String Concatenation

The + operator can be used between strings to add them together to make a new string. This is called **concatenation**:

Example:

```
string firstName = "John ";
string lastName = "Doe";
string fullName = firstName + lastName;
cout << fullName;
```

In the example above, we added a space after firstName to create a space between John and Doe on output. However, you could also add a space with quotes (" " or ' '):

Example:

```
string firstName = "John";
string lastName = "Doe";
string fullName = firstName + " " + lastName;
cout << fullName;
```

Append () function

A string in C++ is actually an object, which contain functions that can perform certain operations on strings. For example, you can also concatenate strings with the append () function:

Example:

```
string firstName = "John ";
string lastName = "Doe";
string fullName = firstName.append(lastName);
cout << fullName;
```

Numbers and Strings

C++ uses the + operator for both **addition** and **concatenation**. Numbers are added. Strings are concatenated. If you add two numbers, the result will be a number:

Example:

```
int x = 10;
int y = 20;
int z = x + y;      // z will be 30 (an integer)
```

If you add two strings, the result will be a string concatenation:

Example:

```
string x = "10";
string y = "20";
string z = x + y;   // z will be 1020 (a string)
```

If you try to add a number to a string, an error occurs:

Example:

```
string x = "10";
int y = 20;
string z = x + y;
```

String Length

To get the length of a string, use the **length()** function:

Example:

```
string txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
cout << "The length of the txt string is: " << txt.length();
```

NOT: You might see some C++ programs that use the size() function to get the length of a string. This is just an alias of length(). It is completely up to you if you want to use length() or size():

Example:

```
string txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
cout << "The length of the txt string is: " << txt.size();
```

Access Strings

You can access the characters in a string by referring to its index number inside square brackets []. This example prints the **first character** in **myString**:

Example:

```
string myString = "Hello";
cout << myString[0];
// Outputs H
```

This example prints the **second character** in **myString**:

Example:

```
string myString = "Hello";
cout << myString[1];
// Outputs e
```

To print the **last character** of a string, you can use the following code:

Example:

```
string myString = "Hello";
cout << myString[myString.length() - 1];
// Outputs o
```

Change String Characters

To change the value of a specific character in a string, refer to the index number, and use single quotes:

Example:

```
string myString = "Hello";
myString[0] = 'J';
cout << myString;
// Outputs Jello instead of Hello
```

The at() function

The <string> library also has an at() function that can be used to access characters in a string:

Example:

```
string myString = "Hello";
cout << myString; // Outputs Hello

cout << myString.at(0);  // First character
cout << myString.at(1);  // Second character
cout << myString.at(myString.length() - 1);  // Last character

myString.at(0) = 'J';
cout << myString;  // Outputs Jello
```

Strings - Special Characters

Because strings must be written within quotes, C++ will misunderstand this string, and generate an error:

Example:

```
string txt = "We are the so-called "Vikings" from the north.";
```

The solution to avoid this problem, is to use the **backslash escape character**. The backslash (\) escape character turns special characters into string characters:

Escape character	Result	Description
\'	'	Single quote
\"	"	Double quote
\\	\	Backslash

The sequence \" inserts a double quote in a string:

Example:

```
string txt = "We are the so-called \"Vikings\" from the north.";
```

The sequence \' inserts a single quote in a string:

Example:

```
string txt = "It\'s alright.";
```

The sequence \\ inserts a single backslash in a string:

Example:

```
string txt = "The character \\ is called backslash.";
```

A list of popular string functions can be found in the table below.

Function	Description
at()	Returns an indexed character from a string
length()	Returns the length of a string
size()	Alias of length(). Returns the length of a string
max_size()	Returns the maximum length of a string
empty()	Checks wheter a string is empty or not
append()	Appends a string (or a part of a string) to another string
substr()	Returns a part of a string from a start index (position) and length
find()	Returns the index (position) of the first occurrence of a string or character
rfind()	Returns the index (position) of the last occurrence of a string or character
replace()	Replaces a part of a string with another string
insert()	Inserts a string at a specified index (position)
erase()	Removes characters from a string
compare()	Compares two strings