TUTORIAL 2: STRING OPERATIONS IN C++

-Jaydeep Shah

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;
```

Note : In C++ string is object not a datatype, it supports multiple operation on it.

Append

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;
```

NOTE: C++ uses the + operator for both addition and concatenation. Numbers are added. Strings are concatenated.

Below code will generate error.

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

ERROR !!

SIZE OF STRING:

String Length

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

Example

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

Access Strings

You can access the characters in a string by referring to its index number inside square brackets [],like ARRAY.

This example prints the first character in myString:

Example

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

Outputs : H

Note: String indexes start with 0: [0] is the first character. [1] is the second character, etc.

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

Note:

However, cin considers a space (whitespace, tabs, etc) as a terminating character, which means that it can only display a single word (even if you type many words):

Example

```
string fullName;

cout << "Type your full name: ";

cin >> fullName;

cout << "Your name is: " << fullName;
```

OUTPUT:

```
Type your full name: John Doe

Your name is: John
```

That's why, when working with strings, we often use the getline() function to read a line of text. It takes cin as the first parameter, and the string variable as second:

Example

```
string fullName;
cout << "Type your full name: ";
getline (cin, fullName);
cout << "Your name is: " << fullName;
```

OUTPUT

```
Type your full name: John Doe
 Your name is: John Doe
```