# 22.6 — std::string appending

▲ ALEX ■ AUGUST 26, 2021

# **Appending**

Appending strings to the end of an existing string is easy using either operator+=, append(), or push\_back() function.

string& string::operator+= (const string& str)

string& string::append (const string& str)

- Both functions append the characters of str to the string.
- Both function return \*this so they can be "chained".
- Both functions throw a length\_error exception if the result exceeds the m

# Sample code:

```
string sString("one");

sstring += string("
two");

string sThree(" three");
sstring.append(sThree);

cout << sString <</pre>
```

# **Output:**

one two three

There's also a flavor of append() that can append a substring:

string& string::append (const string& str, size\_type index, size\_type num)

- This function appends num characters from str, starting at index, to the string.
- Returns \*this so it can be "chained".
- Throws an out\_of\_range if index is out of bounds
- Throws a length\_error exception if the result exceeds the maximum number of characters.

# Sample code:

```
string sString("one ");

const string sTemp("twothreefour");
sString.append(sTemp, 3, 5); // append substring of sTemp starting at index 3 of length
cout << sString << endl;</pre>
```

### **Output:**

one three

Operator+= and append() also have versions that work on C-style strings:

There is an additional flavor of append() that works on C-style strings:

```
string& string::append (const char* str, size_type len)

• Appends the first len characters of str to the string.

• Returns *this so they can be "chained".

• Throw a length_error exception if the result exceeds the maximum number of characters.

• Ignores special characters (including ")

Sample code:

1 | string sString("one ");
2 | 3 | sString.append("threefour", 5);
4 | cout << sString << endl;

Output:

one three

This function is dangerous and its use is not recommended.
```

There is also a set of functions that append characters. Note that the name of the non-operator function to append a character is push\_back(), not append()!

string& string::operator+= (char c)

void string::push\_back (char c)

- Both functions append the character c to the string.
- Operator += returns \*this so it can be "chained".
- Both functions throw a length\_error exception if the result exceeds the maximum number of characters.

# Sample code:

```
string sString("one");

sString += ' ';
sString.push_back('2');
cout << sString <<
endl;</pre>
```

#### **Output:**

one 2

Now you might be wondering why the name of the function is push\_back() and not append(). This follows a naming convent

It turns out there is an append() function for characters, that looks like this:

string& string::append (size\_type num, char c)

- Adds num occurrences of the character c to the string
- Returns \*this so it can be "chained".
- Throws a length\_error exception if the result exceeds the maximum number of characters.

# Sample code:

```
string sString("aaa");

sString.append(4, 'b');
cout << sString <<
endl;</pre>
```

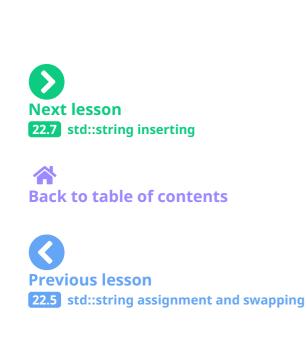
### **Output:**

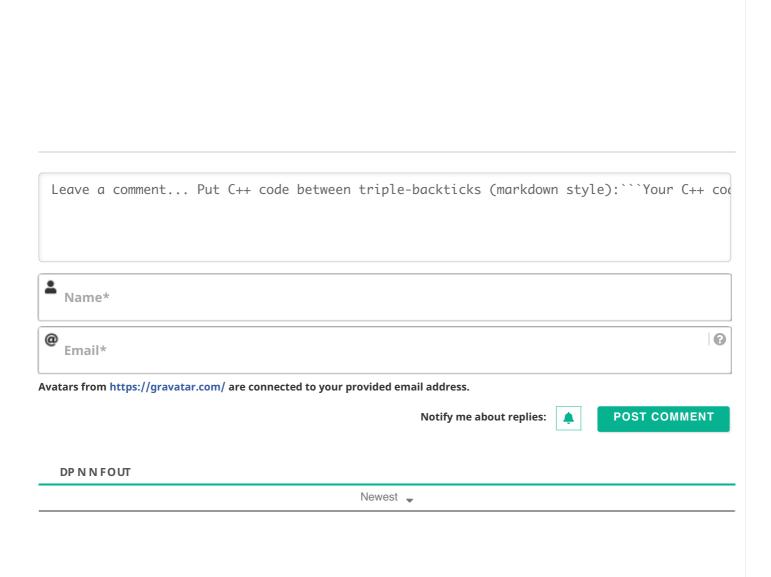
aaabbbb

There's one final flavor of append() that you won't understand unless you know what iterators are. If you're not familiar with iterators, you can ignore this function.

string& string::append (InputIterator start, InputIterator end)

- Appends all characters from the range [start, end) (including start up to but not including end)
- Returns \*this so it can be "chained".
- Throws a length\_error exception if the result exceeds the maximum number of characters.





©2021 Learn C++



