

17.6 — std::string appending

BY ALEX ON JULY 18TH, 2010 | LAST MODIFIED BY ALEX ON JANUARY 23RD, 2020

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 maximum number of characters.

Sample code:

```
1  string sString("one");
2
3  sString += string(" two");
4
5  string sThree(" three");
6  sString.append(sThree);
7
8  cout << sString << endl;
```

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:

```
1  string sString("one ");
2
3  const string sTemp("twothreefour");
4  sString.append(sTemp, 3, 5); // append substring of sTemp starting at index 3 of length 5
5  cout << sString << endl;
```

Output:

one three

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

string& string::operator+= (const char* str)
string& string::append (const char* 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 maximum number of characters.
- str should not be NULL.

Sample code:

```
1 string sString("one");  
2  
3 sString += " two";  
4 sString.append(" three");  
5 cout << sString << endl;
```

Output:

one two three

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:

```
1 string sString("one");  
2  
3 sString += ' ';  
4 sString.push_back('2');  
5 cout << sString << endl;
```

Output:

one 2

Now you might be wondering why the name of the function is `push_back()` and not `append()`. This follows a naming convention used for stacks, where `push_back()` is the function that adds a single item to the end of the stack. If you envision a string as a stack of characters, using `push_back()` to add a single character to the end makes sense. However, the lack of an `append()` function is inconsistent in my view!

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:

```
1 string sString("aaa");  
2  
3 sString.append(4, 'b');  
4 cout << sString << endl;
```

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.



17.7 -- std::string inserting



Index