22.6 — std::string appending

ALEX AUGUST 24, 2022

Appending

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

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

- Both functions append the characters of str to the string.
- Both functions 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:

```
    std::string sString{"one"};

    sString += std::string{" two"};

    std::string sThree{" three"};
    sString.append(sThree);

    std::cout << sString << '\n';
```

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:

```
    std::string sString{"one "};

    const std::string sTemp{"twothreefour"};
    sString.append(sTemp, 3, 5); // append substring of sTemp starting at index 3 of length 5
    std::cout << sString << '\n';
```

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 functions 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:

```
    std::string sString{"one"};

    sString += " two";
    sString.append(" three");
    std::cout << sString << '\n';
```

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:

```
    std::string sString{"one "};

    sString.append("threefour", 5);
    std::cout << sString << '\n';
```

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:

```
    std::string sString{"one"};

    sString += ' ';
    sString.push_back('2');
    std::cout << sString << '\n';
```

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:

```
    std::string sString{"aaa"};

    sString.append(4, 'b');
    std::cout << sString << '\n';
```

Output:

```
aaabbbb
```

There’s one final flavor of append() that works with iterators:

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.

Next lesson
22.7 std::string inserting

Back to table of contents

Previous lesson
22.5 std::string assignment and swapping