22.5 — std::string assignment and swapping

ALEX SEPTEMBER 16, 2022

String assignment

The easiest way to assign a value to a string is to use the overloaded operator= function. There is also an assign() member function that duplicates some of this functionality.

string& string::operator= (const string& str)
string& string::assign (const string& str)
string& string::operator= (const char* str)
string& string::assign (const char* str)
string& string::operator= (char c)

- These functions assign values of various types to the string.
- These functions return *this so they can be "chained".
- Note that there is no assign() function that takes a single char.

Sample code:

```
	std::string sString;

	// Assign a string value
	sString = std::string("One");
	std::cout << sString << '\n';

	const std::string sTwo("Two");
	sString.assign(sTwo);
	std::cout << sString << '\n';

	// Assign a C-style string
	sString = "Three";
	std::cout << sString << '\n';

	sString.assign("Four");
	std::cout << sString << '\n';

	// Assign a char
	sString = '5';
	std::cout << sString << '\n';

	// Chain assignment
	std::string sOther;
	sString = sOther = "Six";
	std::cout << sString << ' ' << sOther << '\n';
```

Output:

```
One
Two
Three
Four
5
Six Six
```

The assign() member function also comes in a few other flavors:

string& string::assign (const string& str, size_type index, size_type len)

- Assigns a substring of str, starting from index, and of length len
- Throws an out_of_range exception if the index is out of bounds
- Returns *this so it can be “chained”.

Sample code:

```
    const std::string sSource("abcdefg");
    std::string sDest;

    sDest.assign(sSource, 2, 4); // assign a substring of source from index 2 of length 4
    std::cout << sDest << '\n';
```

Output:

```
cdef
```

string& string::assign (const char* chars, size_type len)

- Assigns len characters from the C-style array chars
- Throws an length_error exception if the result exceeds the maximum number of characters
- Returns *this so it can be “chained”.

Sample code:

```
    std::string sDest;

    sDest.assign("abcdefg", 4);
    std::cout << sDest << '\n';
```

Output:

```
abcd
```

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

string& string::assign (size_type len, char c)

- Assigns len occurrences of the character c
- Throws a length_error exception if the result exceeds the maximum number of characters
- Returns *this so it can be “chained”.

Sample code:

```
    std::string sDest;

    sDest.assign(4, 'g');
    std::cout << sDest << '\n';
```

Output:

```
gggg
```

Swapping

If you have two strings and want to swap their values, there are two functions both named swap() that you can use.

void string::swap (string& str)
void swap (string& str1, string& str2)

- Both functions swap the value of the two strings. The member function swaps *this and str, the global function swaps str1 and str2.
- These functions are efficient and should be used instead of assignments to perform a string swap.

Sample code:

```
    std::string sStr1("red");
    std::string sStr2("blue");

    std::cout << sStr1 << ' ' << sStr2 << '\n';
    swap(sStr1, sStr2);
    std::cout << sStr1 << ' ' << sStr2 << '\n';
    sStr1.swap(sStr2);
    std::cout << sStr1 << ' ' << sStr2 << '\n';
```

Output:

```
red blue
blue red
red blue
```


Next lesson
22.6 std::string appending

Back to table of contents

Previous lesson
22.4 std::string character access and conversion to C-style arrays