【C++】Day36

Class	C++
Date	@January 11, 2022
Material	
Series Number	
Summary	string search operation

【Ch9】Sequential Container

9.5.3 string Search Operations

The table below describes the search members and their arguments of the string class.

Table 9.14. string Search Operations

Search operations return the index of the desired character or npos if not found	
`s.find(args)`	Find the first occurrence of *args* in s.
`s.rfind(args)`	Find the last occurrence of *args* in s.
`s.find_first_of(args)`	Find the first occurrence of any character from *args* in s.
`s.find_last_of(args)`	Find the last occurrence of any character from *args* in s.
`s.find_first_not_of(args)`	Find the first character in s that is not in *args*.
`s.find_last_not_of(args)`	Find the last character in s that is not in *args*.
***args* must be one of**	
`c, pos`	Look for the character c starting at position pos in s. pos defaults to 0.
`s2, pos`	Look for the string s2 starting at position pos in s. pos defaults to 0.
`cp, pos`	Look for the C-style null-terminated string pointed to by the pointer cp. Start looking at position pos in s. pos defaults to 0.
`cp, pos, n`	Look for the first n characters in the array pointed to by the pointer cp. Start looking at position pos in s. No default for pos or n.

If there is no match, the function returns a static member named `string::npos`. The library defines `npos` as a `const string::size_type` initialized with the value -1. Because `npos` is an unsigned type, this initializer means `npos` is equal to the largest possible size any string could have.

Warning: The string search functions return `string::size_type`, which is an unsigned type. As a result, it is as bad idea to use an int, or other signed type, to hold the return from these functions.

The `find` function does the simplest search. It looks for its argument and returns the index of the first match that is found, or `npos` if there is no match:

```
string name("AnnaBelle");
auto pos1 = name.find("Anna"); //pos1 == 0
```

A slightly more complicated problem requires finding a match to any character in the search string. For example, the following locates the first digit within name:

```
string number("0123456789"), name("r2d2");
auto pos = name.find_first_of(numbers);
```

Instead of looking for a match, we might call `find_first_not_of` to find the first position that is not in the search argument.

For example, to find the first nonnumeric character of a string, we can write:

```
string dept("03714p3");
auto pos = dept.find_first_not_of(numbers);
```

Specifying Where to Start the Search

We can pass an optional starting position to the find operations. This optional argument indicates the position from which to start the search. By default, that position is set to zero. One common programming pattern uses this optional argument to loop through a string finding all occurrences:

```
string::size_type pos = 0;
while( (pos = name.find_first_of(numbers, pos)) != string::npos) {
  cout << "found number at index: " << pos << endl;
  ++pos;
}
```

Searching Backward

The find operations we've used so far execute left to right. The library provides analogous operations that search from right to left. The `rfind` member searches for the last-that is, right-most-occurrence of the indicated substring.

```
string river("Mississippi");
auto first_po = river.find("is"); //returns 1
auto last_po = river.rfind("is"); //returns 4
```

Similarly, the `find_last` functions behave like the `find_first` functions, except that they return the last match rather than the first:

- `find_last_of` searches for the last character that matches any element of the search string
- `find_last_not_of` searches for the last character that does not match any element of the search string

9.5.4 The compare Functions

The string library provides a set of `compare` functions that are similar to the C library `strcmp` function.

Like `strcmp`, `s.compare` returns zero or a positive or negative value depending on whether s is equal to, greater than, or less than the string formed from the given arguments.

Table 9.15. Possible Arguments to s.compare

`s2`	Compare `s` to `s2`.
`pos1, n1, s2`	Compares `n1` characters starting at `pos1` from `s` to `s2`.
`pos1, n1, s2, pos2, n2`	Compares `n1` characters starting at `pos1` from `s` to the `n2` characters starting at `pos2` in `s2`.
`cp`	Compares `s` to the null-terminated array pointed to by `cp`.
`pos1, n1, cp`	Compares `n1` characters starting at `pos1` from `s` to `cp`.
`pos1, n1, cp, n2`	Compares `n1` characters starting at `pos1` from `s` to `n2` characters starting from the pointer `cp`.

9.5.5 Numeric Conversions

The new standard introduced several functions that convert between numeric data and library string:

```
int i = 42;
string s = to_string(i); //converts the int i to this character
double d = stod(s); //converts the string s to floating-point
```

Table 9.16. Conversions between strings and Numbers

`to_string(val);`	Overloaded functions returning the `string` representation of `val`. `val` can be any arithmetic type (§ 2.1.1, p. 32). There are versions of `to_string` for each floating-point type and integral type that is `int` or larger. Small integral types are promoted (§ 4.11.1, p. 160) as usual.

`stoi(s, p, b)`
`stol(s, p, b)`
`stoul(s, p, b)`
`stoll(s, p, b)`
`stoull(s, p, b)`
	Return the initial substring of `s` that has numeric content as an `int`, `long`, `unsigned long`, `long long`, `unsigned long long`, respectively. `b` indicates the numeric base to use for the conversion; `b` defaults to 10. `p` is a pointer to a `size_t` in which to put the index of the first nonnumeric character in `s`; `p` defaults to 0, in which case the function does not store the index.

`stof(s, p)`
`stod(s, p)`
`stold(s, p)`
	Return the initial numeric substring in `s` as a `float`, `double`, or `long double`, respectively. `p` has the same behavior as described for the integer conversions.

Exercise

Exercise 9.50: Write a program to process a `vector<string>`s whose elements represent integral values. Produce the sum of all the elements in that `vector`. Change the program so that it sums of `string`s that represent floating-point values.

See 9_50.cpp for code