【C++】Day eight(3)

Class	C++
Date	@November 26, 2021
Material	
Series Number	
Summary	for each loop

【Ch3】Strings

Use the C++ Version of C Library Headers

In addition to facilities defined specifically for C++, the C++ library incorporates the C library. Headers in C have names of the form name .h. The C__ versions of these headers are named c name—they remove the .h suffix and precede the name with the letter c.

For example, `<cstdio>` has the same contents as `<stdio.h>`, but in a form that is appropriate for C++ programs. In particular, the names defined in the `cname` headers are defined inside the `std` namespace, whereas those defined in the .h versions are not.

3.2.3 Dealing with the Characters in a string

`isalnum(c)`	`true` if `c` is a letter or a digit.
`isalpha(c)`	`true` if `c` is a letter.
`iscntrl(c)`	`true` if `c` is a control character.
`isdigit(c)`	`true` if `c` is a digit.
`isgraph(c)`	`true` if `c` is not a space but is printable.
`islower(c)`	`true` if `c` is a lowercase letter.
`isprint(c)`	`true` if `c` is a printable character (i.e., a space or a character that has a visible representation).
`ispunct(c)`	`true` if `c` is a punctuation character (i.e., a character that is not a control character, a digit, a letter, or a printable whitespace).
`isspace(c)`	`true` if `c` is whitespace (i.e., a space, tab, vertical tab, return, newline, or formfeed).
`isupper(c)`	`true` if `c` is an uppercase letter.
`isxdigit(c)`	`true` if `c` is a hexadecimal digit.
`tolower(c)`	If `c` is an uppercase letter, returns its lowercase equivalent; otherwise returns `c` unchanged.
`toupper(c)`	If `c` is a lowercase letter, returns its uppercase equivalent; otherwise returns `c` unchanged.

Use Range-Based for

It we want to something to every character in a string, by far the best approach is to use a statement introduced by the new standard: the range for statement. This statement iterates through the elements in a given sequence and performs some operation on each value in that sequence. The syntactic form is :

```
for (declaration : expression)
  statement
```

where expression is an object of a type that represents a sequence, and declaration defines the variable that we'll use to access the underlying elements in the sequence. On each iteration, the variable in declaration is initialized from the value of the next element in expression.

See the following code for an example:

```
string str("Some string");
for(auto c : str)
```

```
  cout << c << endl; //print the current character followed by a new line
```

Example: Count punctuation in a string:

```
string str;
cin >> str;
int punct_count = 0;
for(auto c : str)
  if(ispunct(c))
    punct_count++;
cout << "There are " << punct_count << " punctuations in the string.";
```