Introduction to C++ (Season 1)

Unit 4: Objects and Classes
第4单元: 物以类聚 – 对象和类

Section 6 : The C++ string Class
第6节： C++字符串类

The C++ string Class

- C++ 使用 string 类处理字符串
- string类中的函数
  1. 构造
  2. 追加
  3. 赋值
  4. 位置与清除
  5. 长度与容量
  6. 比较
  7. 子串
  8. 搜索
  9. 运算符

```
+string()                                                    构造一个空字符串
+string(value: string)                                       由指定的字符串文字常量构造一个字符串
+string(value: char[])                                       由指定的字符串数组构造一个字符串
+string(ch: char, n: int)                                    构造一个字符串，初值为n个指定字符

+append(s: string): string                                   将字符串s追加在当前string对象后
+append(s: string, index: int, n: int): string              将s中从index起的n个字符追加在当前字符串后

+append(s[]: char, n: int): string                           将s的前n个字符追加在当前字符串后
+append(n: int, ch: char): string                            将n个字符ch追加在当前字符串后
+assign(s[]: char): string                                   将一个字符数组或一个字符串s赋予当前字符串
+assign(s: string, index: int, n: int): string              将s中从index起的n个字符赋予当前字符串

+assign(s: string, n: int): string                           将s的前n个字符赋予当前字符串
+assign(n: int, ch: char): string                            将当前字符串赋值为n个字符ch
+at(index: int): char                                        返回当前字符串中index处的字符
+length(): int                                               返回当前字符串的长度
+size(): int                                                 与length()相同
+capacity(): int                                             返回为当前字符串分配的存储空间大小
+clear(): void                                               清除当前字符串中所有字符
+erase(index: int, n: int): string                          删除当前字符串从index开始的n个字符
+empty(): bool                                               若当前字符串为空，则返回true
+compare(s: string): int                                     这两个比较函数与7.9.4节中介绍的strcmp函数类
+compare(index: int, n: int, s: string): int                似，返回值也相同
+copy(s[]: char, n: int, index: int): void                  将当前字符串从index开始的n个字符复制到s
+data(): char*                                               将当前字符串内容以一个字符数组返回
+substr(index: int, n: int): string                         返回当前字符串从index开始的n个字符的子串

+substr(index: int): string                                  返回当前字符串从index开始的子串
+swap(s: string): void                                       交换当前字符串和s的内容
+find(ch: char): int                                         返回当前字符串中字符ch出现的第一个位置
+find(ch: char, index: int): int                            返回当前字符串中从index开始ch出现的第一个位置

+find(s: string): int                                        返回当前字符串中子串s出现的第一个位置
+find(s: string, index: int): int                           返回当前字符串中从index开始s出现的第一个位置

+replace(index: int, n: int, s: string): string             将本字符串从index开始的n个字符替换为s的内容

+insert(index: int, s: string): string                       将字符串s插入到本字符串index处
+insert(index: int, n: int, ch: char): string               将n个ch插入到本字符串index处
```

注意事项

❖操作string对象中的字符串内容时，有时会用到“index”。

很多string的函数接受两个数字参数： index, n

index: 从index号位置开始

n: 之后的n个字符

Constructing a String (创建 string 对象)

- Create an empty string using string's no-arg constructor(用无参构造函数创建一个空字串):

```
string newString;
```

- Create a string object from a string value or from an array of characters (由一个字符串常量或字符串数组创建string对象) :

```
string message("Aloha World!");
```

```
char charArray[] = {'H', 'e', 'l', 'l', 'o', '\0'};
string message1(charArray);
```

Appending a String (追加字符串)

- You can use several overloaded functions to add new contents to a string. (一系列的重载函数可以将新内容附加到一个字符串中)

```
string s1("Welcome");
s1.append(" to C++"); // appends " to C++" to s1
cout << s1 << endl; // s1 now becomes Welcome to C++

string s2("Welcome");
s2.append(" to C and C++", 3, 2); // appends " C" to s2
cout << s2 << endl; // s2 now becomes Welcome C

string s3("Welcome");
s3.append(" to C and C++", 5); // appends " to C" to s3
cout << s3 << endl; // s3 now becomes Welcome to C

string s4("Welcome");
s4.append(4, 'G'); // appends "GGGG" to s4
cout << s4 << endl; // s4 now becomes WelcomeGGGG
```

Assigning a String (为字符串赋值)

- You can use several overloaded functions to assign new contents to a string(一系列的重载函数可以将一个字符串赋以新内容)

```
string s1("Welcome");
s1.assign("Dallas"); // assigns "Dallas" to s1
cout << s1 << endl; // s1 now becomes Dallas

string s2("Welcome");
s2.assign("Dallas, Texas", 1, 3); // assigns "all" to s2
cout << s2 << endl; // s2 now becomes all

string s3("Welcome");
s3.assign("Dallas, Texas", 6); // assigns "Dallas" to s3
cout << s3 << endl; // s3 now becomes Dallas

string s4("Welcome");
s4.assign(4, 'G'); // assigns "GGGG" to s4
cout << s4 << endl; // s4 now becomes GGGG
```

Functions at, clear, erase, and empty

- at(index): 返回当前字符串中index位置的字符
- clear(): 清空字符串
- erase(index, n): 删除字符串从index开始的n个字符
- empty(): 检测字符串是否为空

```
string s1("Welcome");

cout << s1.at(3) << endl; // s1.at(3) returns c

cout << s1.erase(2, 3) << endl; // s1 is now Weme

s1.clear(); // s1 is now empty

cout << s1.empty() << endl; // s1.empty returns 1 (means true)
```

Comparing Strings (比较字符串)

❖compare() 函数用于比较两个字符串。它与C语言中的 strcmp() 函数很像。

```
string s1("Welcome");
string s2("Welcomg");

cout << s1.compare(s2) << endl; // returns -2
cout << s2.compare(s1) << endl; // returns 2

cout << s1.compare("Welcome") << endl; // returns 0
```

Obtaining Substrings (获取子串)

❖at() 函数用于获取一个单独的字符； 而substr() 函数则可以获取一个子串

```
string s1("Welcome");
cout << s1.substr(0, 1) << endl; // returns W;    从0号位置开始的1个字符
cout << s1.substr(3) << endl; // returns come;    从3号位置直到末尾的子串
cout << s1.substr(3, 3) << endl; // returns com;  从3号位置开始的3个字符
```

Searching in a String (搜索字符串)

❖find() 函数可以在一个字符串中搜索一个子串或者一个字符

```
string s1("Welcome to HTML");
cout << s1.find("co") << endl; // returns 3;   返回子串出现的第一个位置
cout << s1.find("co", 6) << endl; // returns -1  从6号位置开始查找子串出现的第一个位置
cout << s1.find('o') << endl; // returns 4      返回字符出现的第一个位置
cout << s1.find('o', 6) << endl; // returns 9    从6号位置开始查找字符出现的第一个位置
```

Inserting and Replacing Strings (插入和替换字符串)

- insert() : 将某个字符/字符串插入到当前字符串的某个位置
- replace() 将本字串从某个位置开始的一些字符替换为其它内容

```
string s1("Welcome to HTML");
s1.insert(11, "C++ and ");
cout << s1 << endl; // s1 becomes Welcome to C++ and HTML

string s2("AA");
s2.insert(1, 4, 'B'); //在1号位置处连续插入4个相同字符
cout << s2 << endl; // s2 becomes to ABBBBA

string s3("Welcome to HTML");
s3.replace(11, 4, "C++"); //从11号位置开始向后的4个字符替换掉。注意'\0'
cout << s3 << endl; // returns Welcome to C++
```

String Operators (字符串运算符)

```
string s1 = "ABC"; // The = operator
string s2 = s1;    // The = operator
for (int i = s2.size() - 1; i >= 0; i--)
    cout << s2[i]; // The [] operator

string s3 = s1 + "DEFG"; // The + operator
cout << s3 << endl; // s3 becomes ABCDEFG

s1 += "ABC";
cout << s1 << endl; // s1 becomes ABCABC

s1 = "ABC";
s2 = "ABE";
cout << (s1 == s2) << endl; // Displays 0
cout << (s1 != s2) << endl; // Displays 1
cout << (s1 >  s2) << endl; // Displays 0
cout << (s1 >= s2) << endl; // Displays 0
cout << (s1 <  s2) << endl; // Displays 1
cout << (s1 <= s2) << endl; // Displays 1
```

Operator	Description
[ ]	用数组下标运算符访问字符串中的字符
=	将一个字符串的内容复制到另一个字符串
+	连接两个字符串得到一个新串
+=	将一个字符串追加到另一个字符串末尾
<<	将一个字符串插入一个流
>>	从一个流提取一个字符串，分界符为空格或者空结束符
==, !=, <, <=, >, >=	用于字符串比较