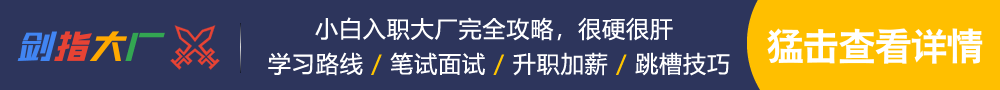
C++ string类（C++字符串）完全攻略

[](http://c.biancheng.net/view/9563.html?source=arctop)

[C语言中文网推出辅导班啦，包括「C语言辅导班、C++辅导班、算法/数据结构辅导班」，全部都是一对一教学：一对一辅导 + 一对一答疑 + 布置作业 + 项目实践 + 永久学习。QQ在线，随时响应！](http://fudao.biancheng.net/)

string 类是 [STL](http://c.biancheng.net/stl/) 中 basic\_string 模板实例化得到的模板类。其定义如下：

typedef basic\_string <char> string;

basic\_string 此处可以不必深究。  
  
string 类的成员函数有很多，同一个名字的函数也常会有五六个重载的版本。篇幅所限，不能将这些原型一一列出并加以解释。这里仅对常用成员函数按功能进行分类，并直接给出应用的例子，通过例子，读者可以基本掌握这些成员函数的用法。  
  
要想更深入地了解 string 类，还要阅读 [C++](http://c.biancheng.net/cplus/) 的参考手册或编译器自带的联机资料。对于前面介绍过的字符串处理的内容，这里不再重复说明。

**1. 构造函数**

string 类有多个构造函数，用法示例如下：

1. string s1(); // si = ""
2. string s2("Hello"); // s2 = "Hello"
3. string s3(4, 'K'); // s3 = "KKKK"
4. string s4("12345", 1, 3); //s4 = "234"，即 "12345" 的从下标 1 开始，长度为 3 的子串

为称呼方便，本教程后文将从字符串下标 n 开始、长度为 m 的字符串称为“子串(n, m)”。  
  
string 类没有接收一个整型参数或一个字符型参数的构造函数。下面的两种写法是错误的：

1. string s1('K');
2. string s2(123);

**2. 对 string 对象赋值**

可以用 char\* 类型的变量、常量，以及 char 类型的变量、常量对 string 对象进行赋值。例如：

1. string s1;
2. s1 = "Hello"; // s1 = "Hello"
3. s2 = 'K'; // s2 = "K”

string 类还有 assign 成员函数，可以用来对 string 对象赋值。assign 成员函数返回对象自身的引用。例如：

1. string s1("12345"), s2;
2. s3.assign(s1); // s3 = s1
3. s2.assign(s1, 1, 2); // s2 = "23"，即 s1 的子串(1, 2)
4. s2.assign(4, 'K'); // s2 = "KKKK"
5. s2.assign("abcde", 2, 3); // s2 = "cde"，即 "abcde" 的子串(2, 3)

**3. 求字符串的长度**

length 成员函数返回字符串的长度。size 成员函数可以实现同样的功能。

**4. string对象中字符串的连接**

除了可以使用+和+=运算符对 string 对象执行字符串的连接操作外，string 类还有 append 成员函数，可以用来向字符串后面添加内容。append 成员函数返回对象自身的引用。例如：

1. string s1("123"), s2("abc");
2. s1.append(s2); // s1 = "123abc"
3. s1.append(s2, 1, 2); // s1 = "123abcbc"
4. s1.append(3, 'K'); // s1 = "123abcbcKKK"
5. s1.append("ABCDE", 2, 3); // s1 = "123abcbcKKKCDE"，添加 "ABCDE" 的子串(2, 3)

**5. string对象的比较**

除了可以用 <、<=、==、!=、>=、> 运算符比较 string 对象外，string 类还有 compare 成员函数，可用于比较字符串。compare 成员函数有以下返回值：

* 小于 0 表示当前的字符串小；
* 等于 0 表示两个字符串相等；
* 大于 0 表示另一个字符串小。

例如：

1. string s1("hello"), s2("hello, world");
2. int n = s1.compare(s2);
3. n = s1.compare(1, 2, s2, 0, 3); //比较s1的子串 (1,2) 和s2的子串 (0,3)
4. n = s1.compare(0, 2, s2); // 比较s1的子串 (0,2) 和 s2
5. n = s1.compare("Hello");
6. n = s1.compare(1, 2, "Hello"); //比较 s1 的子串(1,2)和"Hello”
7. n = s1.compare(1, 2, "Hello", 1, 2); //比较 s1 的子串(1,2)和 "Hello" 的子串(1,2)

**6. 求 string 对象的子串**

substr 成员函数可以用于求子串 (n, m)，原型如下：

string substr(int n = 0, int m = string::npos) const;

调用时，如果省略 m 或 m 超过了字符串的长度，则求出来的子串就是从下标 n 开始一直到字符串结束的部分。例如：

1. string s1 = "this is ok";
2. string s2 = s1.substr(2, 4); // s2 = "is i"
3. s2 = s1.substr(2); // s2 = "is is ok"

**7. 交换两个string对象的内容**

swap 成员函数可以交换两个 string 对象的内容。例如：

1. string s1("West”), s2("East");
2. s1.swap(s2); // s1 = "East"，s2 = "West"

**8. 查找子串和字符**

string 类有一些查找子串和字符的成员函数，它们的返回值都是子串或字符在 string 对象字符串中的位置（即下标）。如果查不到，则返回 string::npos。string: :npos 是在 string 类中定义的一个静态常量。这些函数如下：

* find：从前往后查找子串或字符出现的位置。
* rfind：从后往前查找子串或字符出现的位置。
* find\_first\_of：从前往后查找何处出现另一个字符串中包含的字符。例如：
* s1.find\_first\_of("abc");  //查找s1中第一次出现"abc"中任一字符的位置
* find\_last\_of：从后往前查找何处出现另一个字符串中包含的字符。
* find\_first\_not\_of：从前往后查找何处出现另一个字符串中没有包含的字符。
* find\_last\_not\_of：从后往前查找何处出现另一个字符串中没有包含的字符。

下面是 string 类的查找成员函数的示例程序。

1. #include <iostream>
2. #include <string>
3. **u**[**sin**](http://c.biancheng.net/ref/sin.html)**g** **namespace** std;
4. int main()
5. {
6. string s1("Source Code");
7. int n;
8. **if** ((n = s1.find('u')) != string::npos) //查找 u 出现的位置
9. cout << "1) " << n << "," << s1.substr(n) << endl;
10. //输出 l)2,urce Code
11. **if** ((n = s1.find("Source", 3)) == string::npos)
12. //从下标3开始查找"Source"，找不到
13. cout << "2) " << "Not Found" << endl; //输出 2) Not Found
14. **if** ((n = s1.find("Co")) != string::npos)
15. //查找子串"Co"。能找到，返回"Co"的位置
16. cout << "3) " << n << ", " << s1.substr(n) << endl;
17. //输出 3) 7, Code
18. **if** ((n = s1.find\_first\_of("ceo")) != string::npos)
19. //查找第一次出现或 'c'、'e'或'o'的位置
20. cout << "4) " << n << ", " << s1.substr(n) << endl;
21. //输出 4) l, ource Code
22. **if** ((n = s1.find\_last\_of('e')) != string::npos)
23. //查找最后一个 'e' 的位置
24. cout << "5) " << n << ", " << s1.substr(n) << endl; //输出 5) 10, e
25. **if** ((n = s1.find\_first\_not\_of("eou", 1)) != string::npos)
26. //从下标1开始查找第一次出现非 'e'、'o' 或 'u' 字符的位置
27. cout << "6) " << n << ", " << s1.substr(n) << endl;
28. //输出 6) 3, rce Code
29. **return** 0;
30. }

**9. 替换子串**

replace 成员函数可以对 string 对象中的子串进行替换，返回值为对象自身的引用。例如：

1. string s1("Real Steel");
2. s1.replace(1, 3, "123456", 2, 4); //用 "123456" 的子串(2,4) 替换 s1 的子串(1,3)
3. cout << s1 << endl; //输出 R3456 Steel
4. string s2("Harry Potter");
5. s2.replace(2, 3, 5, '0'); //用 5 个 '0' 替换子串(2,3)
6. cout << s2 << endl; //输出 HaOOOOO Potter
7. int n = s2.find("OOOOO"); //查找子串 "00000" 的位置，n=2
8. s2.replace(n, 5, "XXX"); //将子串(n,5)替换为"XXX"
9. cout << s2 < < endl; //输出 HaXXX Potter

**10. 删除子串**

erase 成员函数可以删除 string 对象中的子串，返回值为对象自身的引用。例如：

1. string s1("Real Steel");
2. s1.erase(1, 3); //删除子串(1, 3)，此后 s1 = "R Steel"
3. s1.erase(5); //删除下标5及其后面的所有字符，此后 s1 = "R Ste"

**11. 插入字符串**

insert 成员函数可以在 string 对象中插入另一个字符串，返回值为对象自身的引用。例如：

1. string s1("Limitless"), s2("00");
2. s1.insert(2, "123"); //在下标 2 处插入字符串"123"，s1 = "Li123mitless"
3. s1.insert(3, s2); //在下标 2 处插入 s2 , s1 = "Li10023mitless"
4. s1.insert(3, 5, 'X'); //在下标 3 处插入 5 个 'X'，s1 = "Li1XXXXX0023mitless"

**12. 将 string 对象作为流处理**

使用流对象 istringstream 和 ostringstream，可以将 string 对象当作一个流进行输入输出。使用这两个类需要包含头文件 sstream。  
  
示例程序如下：

1. #include <iostream>
2. #include <sstream>
3. #include <string>
4. **using** **namespace** std;
5. int main()
6. {
7. string src("Avatar 123 5.2 Ti[tan](http://c.biancheng.net/ref/tan.html)ic K");
8. istringstream istrStream(src); //建立src到istrStream的联系
9. string s1, s2;
10. int n; double d; char c;
11. istrStream >> s1 >> n >> d >> s2 >> c; //把src的内容当做输入流进行读取
12. ostringstream ostrStream;
13. ostrStream << s1 << endl << s2 << endl << n << endl << d << endl << c <<endl;
14. cout << ostrStream.str();
15. **return** 0;
16. }

程序的输出结果是：  
Avatar  
Titanic  
123  
5.2  
K  
  
第 11 行，从输入流 istrStream 进行读取，过程和从 cin 读取一样，只不过输入的来源由键盘变成了 string 对象 src。因此，"Avatar" 被读取到 s1，123 被读取到 n，5.2 被读取到 d，"Titanic" 被读取到s2，'K' 被读取到 c。  
  
第 12 行，将变量的值输出到流 ostrStream。输出结果不会出现在屏幕上，而是被保存在 ostrStream 对象管理的某处。用 ostringstream 类的 str 成员函数能将输出到 ostringstream 对象中的内容提取出来。

**13. 用 STL 算法操作 string 对象**

string 对象也可以看作一个顺序容器，它支持随机访问迭代器，也有 begin 和 end 等成员函数。STL 中的许多算法也适用于 string 对象。下面是用 STL 算法操作 string 对象的程序示例。

[纯文本复制](http://c.biancheng.net/view/400.html)

1. #include <iostream>
2. #include <algorithm>
3. #include <string>
4. **using** **namespace** std;
5. int main()
6. {
7. string s("afgcbed");
8. string::iterator p = find(s.begin(), s.end(), 'c');
9. **if** (p!= s.end())
10. cout << p - s.begin() << endl; //输出 3
11. sort(s.begin(), s.end());
12. cout << s << endl; //输出 abcdefg
13. next\_permutation(s.begin(), s.end());
14. cout << s << endl; //输出 abcdegf
15. **return** 0;
16. }