

# 程序设计实习

## C++ 面向对象程序设计

张勤健

zqj@pku.edu.cn

北京大学信息科学技术学院

2024 年 3 月 27 日

# string 类

string 类是模板类: `typedef basic_string<char> string;`

# string 类

string 类是模板类: `typedef basic_string<char> string;`  
使用 string 类要包含头文件 `<string>`

# string 类

string 类是模板类: `typedef basic_string<char> string;`

使用 string 类要包含头文件 `<string>`

string 对象的常用初始化:

- `string s1("Hello");`
- `string month = "March";`
- `string s2(8, 'x');`

完整的构造函数可以参考:

[zh.cppreference.com/w/cpp/string/basic\\_string/basic\\_string](http://zh.cppreference.com/w/cpp/string/basic_string/basic_string)

错误的初始化方法:

- `string error1 = 'c';` // 错
- `string error2('u');` // 错
- `string error4(8);` // 错

错误的初始化方法:

- `string error1 = 'c';` // 错
- `string error2('u');` // 错
- `string error4(8);` // 错

可以将字符赋值给 string 对象

```
string s; s = 'n';
```

operator= 函数可以参考:

[zh.cppreference.com/w/cpp/string/basic\\_string/operator%3D](http://zh.cppreference.com/w/cpp/string/basic_string/operator%3D)

# string 类程序样例

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4  int main(int argc, char* argv[]){
5      string s1("Hello");
6      cout << s1 << endl;
7      string s2(8, 'x');
8      cout << s2 << endl;
9      string month = "March";
10     cout << month << endl;
11     string s;
12     s='n';
13     cout << s << endl;
14     return 0;
15 }
```

# string 类程序样例

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4  int main(int argc, char* argv[]){
5      string s1("Hello");
6      cout << s1 << endl;
7      string s2(8, 'x');
8      cout << s2 << endl;
9      string month = "March";
10     cout << month << endl;
11     string s;
12     s='n';
13     cout << s << endl;
14     return 0;
15 }
```

输出：

```
Hello
xxxxxxx
March
n
```



# string 类的输入

- string 支持流插入运算符 `cin >> s;`
- string 支持流提取运算符 `cout << s;`
- string 支持 `getline` 函数 `getline(cin ,s);`

# string 类的容量

- string 对象的长度用成员函数 `length()` 读取 `s.length()`
- string 对象的长度也可以用成员函数 `size()` 读取 `s.size()`
- 检查 string 是否无字符使用 `empty()` 函数 `s.empty()`;
- `capacity` 函数是返回当前对象分配的存储空间能保存的字符数量

- 逐个访问 string 对象中的字符：

- operator[] 函数：s[i]
- at 函数：s.at(j)

成员函数 at 会做范围检查，如果超出范围，会抛出 out\_of\_range 异常，而下标运算符 [] 不做范围检查。

- 转换成 C 语言式 const char \* 字符串 s.c\_str()
- 返回指向作为字符存储工作的底层数组的指针 s.data()

# string 类的赋值和连接

- 用 = 赋值
- 用 assign 成员函数复制  
`s3.assign(s1);`  
`s3.assign(s1, 1, 3);` //从 s1 中下标为 1 的字符开始复制 3 个  
参[zh.cppreference.com/w/cpp/string/basic\\_string/assign](http://zh.cppreference.com/w/cpp/string/basic_string/assign)
- 用 +/+ = 运算符连接字符串 `s3 = s3 + s1;`
- 用成员函数 append 连接字符串  
`s1.append(s2);`  
`s2.append(s1, 3, 5);` //下标为 3 开始, 5 个字符 如果 s1 字符串内没有足够字符, 则复制到字符串最后一个字符  
参[zh.cppreference.com/w/cpp/string/basic\\_string/append](http://zh.cppreference.com/w/cpp/string/basic_string/append)

# string 类的比较

- 用关系运算符比较 string 的大小 `==` , `>`, `>=`, `<`, `<=`, `!=`
- 用成员函数 `compare` 比较 string 的大小  
返回值：
  - `>` 正
  - `=` 0
  - `<` 负

# string 类的查找

- `find()`: `string s1("hello world"); s1.find("lo");`  
在 `s1` 中从前向后查找“lo”第一次出现的地方，如果找到，返回“lo”开始的位置，即 `l` 所在的位置下标。如果找不到，返回 `string::npos` (string 中定义的静态常量)  
`s1.find("ll", 2)` // 从下标 2 开始查找“ll”
- `rfind()`: `string s1("hello world"); s1.rfind("lo");`  
在 `s1` 中从后向前查找“lo”第一次出现的地方，如果找到，返回“lo”开始的位置，即 `l` 所在的位置下标。如果找不到，返回 `string::npos`。  
`s1.rfind("ll", 4)` // 从下标 4 反向查找“ll”

[zh.cppreference.com/w/cpp/string/basic\\_string/find](http://zh.cppreference.com/w/cpp/string/basic_string/find)

[zh.cppreference.com/w/cpp/string/basic\\_string/rfind](http://zh.cppreference.com/w/cpp/string/basic_string/rfind)

# string 类的查找

- `find_first_of():s1.find_first_of("abcd");`  
在 `s1` 中从前向后查找“abcd”中任何一个字符第一次出现的地方，如果找到，返回找到字母的位置，如果找不到，返回 `string::npos`。
- `find_last_of():s1.find_last_of("abcd");`  
在 `s1` 中查找“abcd”中任何一个字符最后一次出现的地方，如果找到，返回找到字母的位置，如果找不到，返回 `string::npos`。
- `find_first_not_of():s1.find_first_not_of("abcd");`  
在 `s1` 中从前向后查找不在“abcd”中的字母第一次出现的地方，如果找到，返回找到字母的位置，如果找不到，返回 `string::npos`。
- `find_last_not_of():s1.find_last_not_of("abcd");`  
在 `s1` 中从后向前查找不在“abcd”中的字母第一次出现的地方，如果找到，返回找到字母的位置，如果找不到，返回 `string::npos`。

[zh.cppreference.com/w/cpp/string/basic\\_string](http://zh.cppreference.com/w/cpp/string/basic_string)

# string 类的查找

```
string s1("hello worlld");
cout << s1.find("ll") << endl; //2
cout << s1.find("abc") << endl; //4294967295 (string::npos 的值)
cout << s1.rfind("ll") << endl; //9
cout << s1.rfind("abc") << endl; //4294967295
cout << s1.find_first_of("abcde") << endl; //1
cout << s1.find_first_of("abc") << endl; //4294967295
cout << s1.find_last_of("abcde") << endl; //1
cout << s1.find_last_of("abc") << endl; //4294967295
cout << s1.find_first_not_of("abcde") << endl; //0
cout << s1.find_first_not_of("hello world") << endl; //4294967295
cout << s1.find_last_not_of("abcde") << endl; //10
cout << s1.find_last_not_of("hello world") << endl; //4294967295
```



# string 类的删除

```
#include <iostream>
#include <algorithm>
#include <string>

int main() {
    std::string s = "This is an example";
    std::cout << s << '\n';

    s.erase(0, 5); // 擦除 "This "
    std::cout << s << '\n';

    s.erase(std::find(s.begin(), s.end(), ' ')); // 擦除第一个' '
    std::cout << s << '\n';

    s.erase(s.find(' ')); // 从 ' ' 到字符串尾裁剪
    std::cout << s << '\n';
    return 0;
}
```

## 输出

This is an example  
is an example  
isan example  
isan

# string 类的替换

replace()

- `s1.replace(2,3, "haha");`  
将 s1 中下标 2 开始的 3 个字符换成“haha”
- `s1.replace(2,3, "haha", 1, 2);`  
将 s1 中下标 2 开始的 3 个字符换成“haha”中下标 1 开始的 2 个字符

[zh.cppreference.com/w/cpp/string/basic\\_string/replace](http://zh.cppreference.com/w/cpp/string/basic_string/replace)

# string 类的插入

insert()

- `s1.insert(5,s2);`  
将 `s2` 插入 `s1` 下标 5 的位置
- `s1.insert(2,s2,5,3);`  
将 `s2` 中下标 5 开始的 3 个字符插入 `s1` 下标 2 的位置

[zh.cppreference.com/w/cpp/string/basic\\_string/insert](http://zh.cppreference.com/w/cpp/string/basic_string/insert)

# string 类的拷贝

```
string s1("hello world");  
int len = s1.length();  
char * p2 = new char[len+1];  
s1.copy(p2,5,0);  
p2[5]=0;  
cout << p2 << endl;  
// s1.copy(p2,5,0) 从 s1 的下标 0 的字符开始制作一个最长 5 个字符长度的字符串副本  
//并将其赋值给 p2。返回值表明实际复制字符串的长度。
```

输出

hello

# string 类的子串

## substr() 函数

```
string s1("hello world"), s2;  
s2 = s1.substr(4,5); // 下标 4 开始 5 个字符  
cout << s2 << endl;
```

## 输出

o wor

# string 类的交换

## swap() 函数

```
#include <string>
#include <iostream>
int main() {
    std::string a = "AAA";
    std::string b = "BBB";
    std::cout << "before swap" << '\n';
    std::cout << "a: " << a << '\n';
    std::cout << "b: " << b << '\n';
    a.swap(b);
    std::cout << "after swap" << '\n';
    std::cout << "a: " << a << '\n';
    std::cout << "b: " << b << '\n';
}
```

## 输出

```
before swap
a: AAA
b: BBB
after swap
a: BBB
b: AAA
```

除了标准流和文件流输入输出外，还可以从 `string` 进行输入输出；类似 `istream` 和 `ostream` 进行标准流输入输出，我们用 `istringstream` 和 `ostringstream` 进行字符串上的输入输出，也称为内存输入输出。

```
#include <string>
#include <iostream>
#include <sstream>
```

# 字符串输入流 istream

```
string input("Input test 123 4.7 A");
istringstream inputString(input);
string string1, string2;
int i;
double d;
char c;
inputString >> string1 >> string2 >> i >> d >> c;
cout << string1 << endl << string2 << endl;
cout << i << endl << d << endl << c << endl;
long L;
if(inputString >> L) cout << "long\n";
else cout << "empty\n";
```

## 输出

Input  
test  
123  
4.7  
A  
empty



# 字符串输出流 ostream

```
ostream outputString;  
int a = 10;  
outputString << "This " << a << "ok" << endl;  
cout << outputString.str();
```

输出

This 10ok