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

2025 年 4 月 16 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

string类

错误的初始化方法：

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

string类

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

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

```cpp
#include <iostream>
#include <string>
using namespace std;
int main(int argc, char* argv[]){
  string s1("Hello");
  cout << s1 << endl;
  string s2(8,'x');
  cout << s2 << endl;
  string month = "March";
  cout << month << endl;
  string s;
  s='n';
  cout << s  << endl;
  return 0;
}
```

输出:

```
Hello
xxxxxxxx
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

string类的元素访问

- 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 个字符给 s3
  ```
  参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

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
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`

string类的查找

```
1  string s1("hello worlld");
2  cout << s1.find("ll") << endl; //2
3  cout << s1.find("abc") << endl; //4294967295 (string::npos 的值)
4  cout << s1.rfind("ll") << endl; //9
5  cout << s1.rfind("abc") << endl; //4294967295
6  cout << s1.find_first_of("abcde") << endl; //1
7  cout << s1.find_first_of("abc") << endl; //4294967295
8  cout << s1.find_last_of("abcde") << endl; //1
9  cout << s1.find_last_of("abc") << endl;//4294967295
10 cout << s1.find_first_not_of("abcde") << endl;//0
11 cout << s1.find_first_not_of("hello world") << endl;//4294967295
12 cout << s1.find_last_not_of("abcde") << endl;//10
13 cout << s1.find_last_not_of("hello world") << endl;//4294967295
```

string类的删除

```cpp
#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

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

string类的拷贝

```
1  string s1("hello world");
2  int len = s1.length();
3  char * p2 = new char[len+1];
4  s1.copy(p2,5,0);
5  p2[5]=0;
6  cout << p2 << endl;
7  // 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
8  //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

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

输出

```
o wor
```

string类的交换

swap()函数

```cpp
#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
```

字符串流处理

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osteram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。

```
1  #include <string>
2  #include <iostream>
3  #include <sstream>
```

字符串输入流 istringstream

```
1  string input("Input test 123 4.7 A");
2  istringstream inputString(input);
3  string string1, string2;
4  int i;
5  double d;
6  char c;
7  inputString >> string1 >> string2 >> i >> d >> c;
8  cout << string1 << endl << string2 << endl;
9  cout << i << endl << d << endl << c <<endl;
10 long L;
11 if(inputString >> L) cout << "long\n";
12 else cout << "empty\n";
```

输出

```
Input
test
123
4.7
A
empty
```

字符串输出流 ostringstream

```
1  ostringstream outputString;
2  int a = 10;
3  outputString << "This " << a << "ok" << endl;
4  cout << outputString.str();
```

输出

```
This 10ok
```