

# 字符串(string 类)

## 声明

```
1  #include<string>
2  #include<iostream>
3  using namespace std;
4
5  string str1;//生成空字符串
6  cin << str1;
7  cout << str1;
8
9  string str2("hello");//生成并初始化
10 cout << str2;
11
12 string str3(str2);//hello
13 cout << str3;
14
15 string str4(10, 'H');//HHHHHHHHHH
16 cout << str4;
17
18 char a[] = "hello";//hello
19 string str5(a);
20 cout << str5;
```

## 连接 (追加) : append()函数

```
1  string str("Hello");
2  char sz[] = "word";
3  string str6("word");
4
5  str.append(10, 'w');//追加单个字符
6
7  str.append(sz);//追加c风格字符串
8
9  str.append(str6);//追加字符串
```

## 比较: compare()函数

返回值: 0、1、-1

前 > 后 返回1、前 < 后 返回-1、前 = 后 返回0;

```

1 string date1("21");
2 string date2("22");
3 string date3("22");
4 char date4[] = "23";
5
6 cout << date1.compare(date2) << endl; //-1
7 cout << date2.compare(date1) << endl; //1
8 cout << date2.compare(date3) << endl; //0
9 cout << date3.compare(date4) << endl; //-1

```

## 查找：find()函数

```

1 string findStr("ABCDEF");
2 cout << findStr.find('D', 2) << findStr.find('D', 5);
3 //第一个参数是要查找的字符，第二个参数是起始查找的位置。返回值为该字符在字符串中的位置int
  (0位起)，若无则返回-1.
4
5 char findChar = "BCD";
6 cout << findStr.find(findChar, 0);
7 //返回字符串在string中的位置，本题为1
8
9 string findStr1("ABC");
10 cout << findStr.find(findStr1, 0);

```

## 替换：replace()函数

```

1 string replaceStr("ABCDEFGH");
2 cout << replaceStr.replace(2, 5, "2345"); //先删除C-E，在插入2345。AB2345FG。
3 cout << replaceStr.replace(2, 5, 10, 'M'); //先删除，再插入10个M

```

## 插入：insert()函数

```

1 string incertStr("123456");
2 string incertStr1("ABCDEF");
3 cout << incertStr.insert(3, incertStr1);

```

## 删除 & 提取子串：erase() & substr()函数

```

1 string str_test("ABCDEFGH");
2 str_test.erase(3, 3); //从D起 删除3个字符
3
4 str_test.substr(3, 3); //返回string，提取DEF。

```

获取字符串长度 `str.length()`

#### 5.4.2. C方式的字符处理函数

- ① `isdigit(char ch)` //判断是否数字(0-9)
- ② `isalpha(char ch)` //判断是否字母(A~Z, a-z)
- ③ `isalnum(char ch)` //判断是否字母或数字(A~Z, a-z, 0-9)
- ④ `isxdigit(char ch)` //判断是否16进制数字(0-9, A-F, a-f)
- ⑤ `isspace(char ch)` //判断是否空格(含回车/换行/换页/tab/竖向tab)
- ⑥ `islower(char ch)` //判断是否小写字母(a-z)
- ⑦ `isupper(char ch)` //判断是否大写字母(A-Z)
- ⑧ `tolower(char ch)` //大写转小写, 其余原值返回
- ⑨ `toupper(char ch)` //小写转大写, 其余原值返回



赋值运算符: `=` 将后面的字符串赋值给前面的字符串  $O(n)$   
比较运算符: `== != < <= > >=` 比较的是两个字符串的字典序大小  $O(n)$   
连接运算符: `++` 将一个运算符加到另一个运算符后面  $O(n)$   
`s[index]` 返回字符串s中下标为index的字符, `string`中下标也是从0开始  $O(1)$   
`s.substr(p,n)` 返回从s的下标p开始的n个字符组成的字符串, 如果n省略就取到底  $O(n)$   
`s.length()` 返回字符串的长度  $O(1)$   
`s.empty()` 判断s是否为空, 空返回1, 不空返回0,  $O(1)$   
`s.erase(p0,len)` 删除s中从p0开始的len个字符, 如果len省略就删到底  $O(n)$   
`s.erase(s.begin()+i)` 删除下标为i个字符  $O(n)$   
`s1.insert(p0,s2,pos,len)` 后两个参数截取s2, 可以省略  $O(n)$   
`s.insert(p0,n,c)` 在p0处插入n个字符c  $O(n)$   
`s1.replace(p0,len0,s2,pos,len)` 删除p0开始的len0个字符, 然后在p0处插入串s2中从pos开始的len个字符, 后两个参数可以省略  $O(n)$   
`s1.find(s2,pos)` 从前往后, 查找成功时返回第一次出现的下标, 失败返回`string::npos`的值  $(-1)$   $O(n*m)$   
`s1.rfind(s2,pos)` 从pos开始从后向前查找字符串s2中字符串在当前串后边第一次出现的下标  $O(n*m)$

## 字符串流 (常用于类型转换)

### 注意

1. 再进行**多次转换**的时候, 必须调用`stringstream`的成员函数`clear()`.
2. `clear()`重置流的标志状态; `str()`清空流的内存缓冲, 重复使用内存消耗不再增加!
3. 在多次数据类型转换的场景下, 必须使用 `clear()` 方法清空 `stringstream`, 不使用 `clear()` 方法或使用 `str("")` 方法, 都不能得到数据类型转换的正确结果。

```
1 #include<sstream>
2 stringstream stream;
3 stream.clear();
4 stream.str("");
```

### 用法1 数据类型的转换

```
1 // int -> string
2 stringstream sstream;
3 string strResult;
4 int nValue = 1000;
5
6 // 将int类型的值放入输入流中
7 sstream << nValue;
8 // 从sstream中抽取前面插入的int类型的值, 赋给string类型
9 sstream >> strResult;
```

## 用法2 [多字符串的拼接](#)

```
1 stringstream sstream;
2 // 将多个字符串放入 sstream 中
3 sstream << "first" << " " << "string,";
4 sstream << " second string";
5 // 可以使用 str() 方法, 将 stringstream 类型转换为 string 类型;
6 cout << "strResult is: " << sstream.str() << endl;
```

## 用法3 [可以用于分割被空格、制表符等符号分割的字符串](#)

```
1 #include<iostream>
2 #include<sstream>           //istringstream 必须包含这个头文件
3 #include<string>
4 using namespace std;
5 int main(){
6     string str="i am a boy";
7     istringstream is(str);
8     string s;
9     while(is>>s) {
10         cout<<s<<endl;
11     }
12 }
```

## 用法4 [实现任意类型的转换](#)

```
1 template<typename out_type, typename in_value>
2 out_type convert(const in_value & t){
3     stringstream stream;
4     stream<<t; //向流中传值
5     out_type result; //这里存储转换结果
6     stream>>result; //向result中写入值
7     return result;
8 }
9
10 int main(){
11     string s = "1 23 # 4";
12     stringstream ss;
13     ss<<s;
14     while(ss>>s){
15         cout<<s<<endl;
16         int val = convert<int>(s);
17         cout<<val<<endl;
18     }
19     return 0;
20 }
```