C++ string 的常用函数用法总结

C++提供了一种更方便和强大的字符处理工具--string 类.

String 类的头文件为:<string>

假设定义 string s;

输入方式:　　//以下输入输出方式的头文件为:<iostream>

1. 1:cin>>s

   C++的标准输入,只能输入一个单词或者字母 ， 遇到空格或者回车键结束.

2. 2:getline(cin,s,‘#’)

   说明:

   输入一整行.
   第一个参数表示采用输入流的方式输入;
   第二个参数表示要存储到哪个地方;
   第三个参数是表示输入的结束符号(当遇到给符号时候输入才结束),该参数如果省略不写,则默认是以回车键结束,如果这个参数不是回车换行键是其他字符时候,那么可以输入多行(即遇到回车键不会输入结束,直到遇到规定字符输入才结束;

输出方法:cout<<s

一. string 的构造函数的形式

1. string str：生成空字符串
2. string s(str)：生成字符串为 str 的复制品
3. string s(str, str_begin, str_len)：将字符串 str 中从下标 strbegin 开始、长度为 strlen 的部分作为字符串初值
4. string s(cstr, char_len)：以 C_string 类型 cstr 的前 char_len 个字符串作为字符串 s 的初值
5. string s(num, char)：生成 num 个 c 字符的字符串
6. string s(str, str_index)：将字符串 str 中从下标 str_index 开始到字符串结束的位置作为字符串初值

```
string str1;                    //生成空字符串
```

```
string str2("123456789");    //生成"1234456789"的复制品

string str3("12345", 0, 3);  //结果为"123"

string str4("012345", 5);    //结果为"01234"

string str5(5, '1');         //结果为"11111"

string str6(str2, 2);        //结果为"3456789"
```

二. string 的大小和容量

1. size()和 length()：返回 string 对象的字符个数，他们执行效果相同。
2. max_size()：返回 string 对象最多包含的字符数，超出会抛出 length_error 异常
3. capacity()：重新分配内存之前，string 对象能包含的最大字符数

```
string str("1234567");

cout << "size=" << s.size() << endl;         // size=7

cout << "length=" << s.length() << endl;     // length=7

cout << "max_size=" << s.max_size() << endl; // max_size=4294967294

cout << "capacity=" << s.capacity() << endl; // capacity=15
```

三. string 的字符串比较

1. 比较操作符: >, >=, <, <=, ==, !=

   这些操作符根据“当前字符特性”将字符按字典顺序进行逐一比较，字典排序靠前的字符小，比较的顺序是从前向后比较，遇到不相等的字符就按这个位置上的两个字符的比较结果确定两个字符串的大小(前面减后面)

2. 成员函数 compare()

支持多参数处理，支持用索引值和长度定位子串进行比较，前面减去后面的 ASCII 码，>0 返回 1，<0 返回-1，相同返回 0

str1.compare(str2)：compare str1[0, _Mysize) with _Right

str1.compare(_Off, _N0, str2)：compare str1[_Off, _Off + _N0) with str2

str1.compare(_Off, _N0, str2, _Roff, _Count)：compare str1[_Off, _Off + _N0) with str2[_Roff, _Roff + _Count)

str1.compare(*_Ptr)：compare str1[0, _Mysize) with [_Ptr, <null>)

str1.compare(_Off, _N0, *_Ptr, _Count)：compare str1[_Off, _Off + _N0) with [_Ptr, _Ptr + _Count)

```
string A("aBcdf");

string B("AbcdF");

string C("123456");

string D("123dfg");



cout << "A.compare(B): " << A.compare(B) << endl;
//"aBcdf"和"AbcdF"比较，a>A，返回 1

cout << "A.compare(2, 2, B):" << A.compare(2, 2, B)<< endl;
//"cd"和“AbcdF"比较，c>A，返回 1

cout << "A.compare(2, 2, B, 2, 2):" << A.compare(2, 2, B, 2, 2) << endl;
 //"cd"和“cd"比较，返回 0

cout << "C.compare(0, 4, D, 0, 4):" << C.compare(0, 4, D, 0, 4)<< endl;
 //"1234"和"123d"比较，返回-1
```

四. string 的插入：push_back() 和 insert()

```
    // 尾插一个字符
```

```
    s1.push_back('a');

    s1.push_back('b');

    s1.push_back('c');

    cout<<"s1:"<<s1<<endl; // s1:abc



    // insert(pos,char):在制定的位置 pos 前插入字符 char

    s1.insert(s1.begin(),'1');

    cout<<"s1:"<<s1<<endl; // s1:1abc
```

五、string 拼接字符串：append() 、 +

```
//方法一：append()

string s1("abc");

s1.append("def");

cout<<"s1:"<<s1<<endl; // s1:abcdef

// 方法二：+ 操作符

string s2 = "abc";/*s2 += "def";*/

string s3 = "def";

s2 += s3.c_str();

cout<<"s2:"<<s2<<endl; // s2:abcdef
```

六、 string 的遍历：借助迭代器 或者 下标法

1. 正向迭代器 str.begin()、str.end()
2. 反向迭代器 str.rbegin()、str.rend()

```
string s1("abcdef");

// 正向迭代器

string::iterator iter = s1.begin();for( ; iter < s1.end() ; iter++){

    cout<<*iter;}

cout<<endl;   //abcdef

// 反向迭代器

string::reverse_iterator riter = s1.rbegin();for( ; riter < s1.rend() ;
riter++){

    cout<<*riter;}

cout<<endl;  //fedcba
```

七、 string 的删除：erase()

1. iterator erase(iterator p)：删除字符串中 p 所指的字符
2. iterator erase(iterator first, iterator last)：删除字符串中迭代器区间 [first, last) 上所有字符
3. string& erase(size_t pos, size_t len)：删除字符串中从索引位置 pos 开始的 len 个字符
4. void clear()：删除字符串中所有字符

```
string s1 = "123456789";



s1.erase(s1.begin() + 1);                // 结果：13456789

s1.erase(s1.begin() + 1, s1.end() - 2);   // 结果：189
```

```
s1.erase(1,6);                          // 结果：189
```

八、 string 的字符替换

1. string& replace(size_t pos, size_t n, const char *s)：将当前字符串从 pos 索引开始的 n 个字符，替换成字符串 s
2. string& replace(size_t pos, size_t n, size_t n1, char c)：将当前字符串从 pos 索引开始的 n 个字符，替换成 n1 个字符 c
3. string& replace(iterator i1, iterator i2, const char* s)：将当前字符串[i1,i2)区间中的字符串替换为字符串 s

```
string s1("hello,world!");



s1.replace(6, 5, "girl");                       // 结果：hello,girl.

s1.replace(s1.size() - 1, 1, 1, '.');           // 结果：hello,world.

s1.replace(s1.begin(), s1.begin() + 5, "boy");  // 结果：boy,girl.
```

九、string 大小写转换：tolower() 和 toupper() 或者 STL 中的 transform 算法

1. tolower(char) 和 toupper(char) ：将字符进行大小写转换
2. transform(_InIt _First, _InIt _Last, _OutIt _Dest, _Fn1 _Func)：transform [_First, _Last) with _Func

```
string s = "ABCDEFG";

for( int i = 0; i < s.size(); i++ ){

    s[i] = tolower(s[i]);}
```

```
cout << s << endl;  //abcdefg

transform(s.begin(), s.end(), s.begin(), ::toupper);

cout << s << endl;  //"ABCDEFG"
```

十、 string 的查找：find

1. find (const char* s, size_t pos)：在当前字符串的 pos 索引位置开始，查找子串 s，返回找到的位置索引
2. find (char c, size_t pos)：在当前字符串的 pos 索引位置开始，查找字符 c，返回找到的位置索引
3. rfind (const char* s, size_t pos)：在当前字符串的 pos 索引位置开始，反向查找子串 s，返回找到的位置索引
4. rfind (char c, size_t pos)：在当前字符串的 pos 索引位置开始，反向查找字符 c，返回找到的位置索引
5. find_first_of (const char* s, size_t pos)：在当前字符串的 pos 索引位置开始，查找子串 s 的字符，返回找到的位置索引
6. find_first_not_of (const char* s, size_t pos)：在当前字符串的 pos 索引位置开始，查找第一个不位于子串 s 的字符，返回找到的位置索引
7. find_last_of(const char* s, size_t pos)：在当前字符串的 pos 索引位置开始，向前查找第一个位于子串 s 的字符，返回找到的位置索引
8. find_last_not_of (const char* s, size_t pos)：在当前字符串的 pos 索引位置开始，向前查找第一个不位于子串 s 的字符，返回找到的位置索引

```
string s("dog bird chicken bird cat");

//字符串查找-----找到后返回首字母在字符串中的下标// 1. 查找一个字符串

cout << s.find("chicken") << endl;        // 结果是：9

// 2. 从下标为 6 开始找字符'i'，返回找到的第一个 i 的下标

cout << s.find('i', 6) << endl;           // 结果是：11
```

```
// 3. 从字符串的末尾开始查找字符串，返回的还是首字母在字符串中的下标

cout << s.rfind("chicken") << endl;       // 结果是：9

// 4. 从字符串的末尾开始查找字符

cout << s.rfind('i') << endl;             // 结果是：18

// 5. 在该字符串中查找第一个属于字符串 s 的字符

cout << s.find_first_of("13br98") << endl;  // 结果是：4("b")

// 6. 在该字符串中查找第一个不属于字符串 s 的字符，先匹配 dog，然后 bird 匹配不到，所以打印 4

cout << s.find_first_not_of("hello dog 2006") << endl; // 结果是：4

// 7. 在该字符串从后往前查找第一个属于字符串 s 的字符

cout << s.find_last_of("13r98") << endl;              // 结果是：19

// 8. 在该字符串从后往前查找第一个不属于字符串 s 的字符，先匹配 tac，然后空格匹配不到，所以打印 21

cout << s.find_last_not_of("teac") << endl;           // 结果是：21
```

十一、 string 的排序

sort(iterator iter1, iterator iter2)：对[iter1, iter2)进行排序

```
string s = "cdbaef";sort(s.begin(), s.begin() + 3);

cout << "s: " << s << endl;    //s: bcdaef
```

十二、 string 的分割/截取字符串：substr()

```
string s1("0123456789");
```

```
string s2 = s1.substr(2, 5);

cout << s2 << endl;    // 结果：23456，参数 5 表示截取的字符串的长度



string str("I,am,a,student; hello world!");

string split(",; !");

int iCurrentIndex = 0;int iSplitIndex;while (iCurrentIndex < str.size())
 {

    iSplitIndex = str.find_first_of(split, iCurrentIndex);

    if (iSplitIndex == -1)

        iSplitIndex = str.size();



    if (iSplitIndex != iCurrentIndex)

        cout << str.substr(iCurrentIndex, iSplitIndex - iCurrentIndex) <<
 endl;



     iCurrentIndex = iSplitIndex + 1;}/*****************************
*

结果:

    I

    am

    a

    student

    hello

    world
```

```
*/
```