库函数查询手册

龙马高新教育搜集整理制作

索引

A

B

C

D

E

F

G

I

K

L

M

N

O

P

R

S

T

U

W

A

any(bitset)函数

函数定义

```
bool any() const;
```

函数说明

如果一个 bitset 变量有位被置 1，则返回 TRUE，否则返回 FALSE。

函数示例

```
bitset<8> bt(6);
bool bRet = bt.any();
if(bRet)
cout << “True” << endl;
else
cout << “False” << endl;
```

append(string)函数

函数定义

```
basic_string &append( const basic_string &str );
basic_string &append( const char *str );
basic_string &append( const basic_string &str, size_type index, size_type len );
basic_string &append( const char *str, size_type num );
basic_string &append( size_type num, char ch );
basic_string &append(input_iterator start, input_iterator end);
```

函数说明

函数功能说明如下：

在字符串的末尾添加 str;

在字符串的末尾添加 str 的子串,子串以 index 索引开始，长度为 len.

在字符串的末尾添加 str 中的 num 个字符.

在字符串的末尾添加 num 个字符 ch.

在字符串的末尾添加以迭代器 start 和 end 表示的字符序列.

函数示例

```
string str = “hello, world!”;
str.append(10, ‘*’);
cout << str << endl;
```

输出：hello, world!**********

assign(deque)函数

函数定义

```
void assign(input_iterator start, input_iterator end);
void assign(Size num, const TYPE &val);
```

函数说明

assign()函数用 start 和 end 指示的范围为双向队列赋值，或者设置成 num 个 val。

函数示例

```
deque<char> dq1(3);
deque<char> dq2(3, ‘a’);
dq1.assign(dq2.begin(), dq2.begin() + 2);
deque<char>::iterator pIndex = dq1.begin();
for(pIndex; pIndex != dq1.end(); pIndex++)
{
cout << *pIndex;
}
```

输出：bb

assign(list)函数

函数定义

```
void assign(input_iterator start, input_iterator end);
void assign(size_type num, const TYPE &val);
```

函数说明

assign()函数以迭代器 start 和 end 指示的范围为 list 赋值或者为 list 赋值 num 个以 val 为值的元素。

函数示例

```
list<int> list1, list2;
list<int>::iterator pIndex;
list1.push_front(1);
list1.push_back(2);
list2.assign(list1.begin(), list1.end());
for(pIndex = list2.begin(); pIndex != list2.end(); pIndex++)
cout << *pIndex;
```

输出:12

assign(string)函数

函数定义

```
basic_string &assign( const basic_string &str );
basic_string &assign( const char *str );
basic_string &assign( const char *str, size_type num );
basic_string &assign( const basic_string &str, size_type index, size_type len );
basic_string &assign( size_type num, char ch );
```

函数说明

函数通过以下几种方式给字符串赋值：
用 str 为字符串赋值。
用 str 的开始 num 个字符为字符串赋值。
用 str 的子串为字符串赋值,子串以 index 索引开始，长度为 len。
用 num 个字符 ch 为字符串赋值。

函数示例

```
string str1 = “War and Peace”;
string str2;
str2.assign(str1, 4, 3);
cout << str2 << endl;
```

输出: and

assign(vector)函数

函数定义

```
void assign(input_iterator start, input_iterator end);
void assign(size_type num, const TYPE &val);
```

函数说明

将区间[start, end)的元素赋到当前 vector。

将 num 个值为 val 的元素到 vector 中，这个函数将会清除掉为 vector 赋值以前的内容

函数示例

```
vector<int> vec1, vec2;
vector<int>::iterator i;
vec1.push_back(1);
vec1.push_back(2);
vec2.assign(vec1.begin(), vec1.end());
for(i = vec2.begin(); i != vec2.end(); i++)
cout << *i;
```

输出:12

at(deque)函数

函数定义

```
reference at(size_type pos);
const_reference at(size_type pos) const;
```

函数说明

at()函数返回一个引用，指向双向队列中位置 pos 上的元素。

函数示例

```
deque<int> dq;
dq.push_back(10);
dq.push_back(20);
int &first = dq.at(0);
const int &second = dq.at(1);
cout <<"First:"<< first << endl;
```

```
cout << “Second:”<< second << endl;
```

输出:First:10
Second:20;

at(string)函数

函数定义

```
reference at(size_type pos);
const_reference at(size_type pos) const;
```

函数说明

数返回一个引用，指向在 index 位置的字符. 如果 index 不在字符串范围内, at() 将报告"out of range"错误，并抛出 out_of_rang 异常。

函数示例

```
string str = “hello”;
char ch = str.at(1);
cout << ch << endl;
```

输出:e

at(vector)函数

函数定义

```
reference at(size_type pos);
const_reference at(size_type pos) const;
```

函数说明

at() 函数 返回当前 Vector 指定位置 pos 元素的引用. at() 函数 比 [ ] 运算符更加安全, 因为它不会让你去访问到 Vector 内越界的元素。

函数示例

```
vector<int> vec(3, 2);
for(int i = 0; i < 3; i+)
cout << vec.at(i) ;
```

输出:222

B

back(deque)函数

函数定义

```
reference back();
const_reference back() const;
```

函数说明

back()返回一个引用，指向双向队列中最后一个元素。

函数示例

```
deque<int> dq;
dq.push_front(1);
dq.push_back(2);
cout << dq.back() ;
```

输出: 2

back(list)函数

函数定义

```
reference back();
const_reference back() const;
```

函数说明

back()函数返回一个引用，指向 list 的最后一个元素。

函数示例

```
list<int> lst;
lst.push_back(2);
lst.push_front(1);
cout << lst.back();
```

输出:2

back(queue)函数

函数定义

```
reference back();
const_reference back() const;
```

函数说明

back()返回一个引用，指向队列的最后一个元素。

函数示例

```
queue<int> que;
que.push(1);
que.push(2);
cout << que.back();
```

输出:2

back(vector)函数

函数定义

```
reference back();
const_reference back() const;
```

函数说明

back()返回当前 vector 的最后一个元素的引用。

函数示例

```
vector<int> vec;
vec.push_back(2);
vec.push_front(1);
cout << vec.back();
```

输出:2

bad(io)函数

函数定义

```
bool bad() const;
```

函数说明

如果当前流发生致命错误，bad()返回 TRUE，否则返回 FALSE。

函数示例

```
bool bRet = cout.bad();
if(bRet)
cout <<”No Stream Error”;
else
cout <<” Stream Error”;
```

输出: No Stream Error

begin(deque)函数

函数定义

```
iterator begin();
const_iteratoe begin() const;
```

函数说明

返回一个迭代器，指向双向队列的第一个元素，

函数示例

```
deque<int> dq;
dq.push_back(10);
dq.push_back(20);
deque<int>::iterator i;
i = dq.begin();
cout <<dq.at(0)<< endl;
cout << *i << endl;
```

输出: 10
10

begin(list)函数

函数定义

```
iterator begin();
const_iteratoe begin() const;
```

函数说明

返回一个迭代器，指向 list 的第一个元素，

函数示例

```
list<int> lst;
lst.push_back(10);
lst.push_back(20);
deque<int>::iterator i;
i = lst.begin();
cout <<lst.at(0)<< endl;
cout << *i << endl;
```

输出: 10
10

begin(map)函数

函数定义

```
iterator begin();
const_iteratoe begin() const;
```

函数说明

返回一个迭代器，指向 map 的第一个元素，

函数示例

```
map<int, int> mp;
map<int, int>::iterator mp_i;
typedef pair<int, int> Int_pair;
mp.insert(Int_pair(1, 2));
mp.insert(Int_pair(2, 3));
mp.insert(Int_pair(3, 4));
```

```
mp_i = mp.begin();
cout << mp_i ->first;
```

输出:1

begin(multimap)函数

函数定义

```
iterator begin();
const_iteratoe begin() const;
```

函数说明

返回一个迭代器，指向 multimap 的第一个元素，

函数示例

```
multimap<int, int> mp;
multimap<int, int>::iterator mp_i;
typedef pair<int, int> Int_pair;
mp.insert(Int_pair(1, 2));
mp.insert(Int_pair(2, 3));
mp.insert(Int_pair(3, 4));
mp_i = mp.begin();
cout << mp_i ->first;
```

输出:1

begin(multiset)函数

函数定义

```
iterator begin();
const_iteratoe begin() const;
```

函数说明

返回一个迭代器，指向 multiset 的第一个元素，

函数示例

```
multiset<int> mulset;
multiset<int>::iterator mulset_i;
```

```
mulset.insert(2);
mulset.insert(3);
mulset.insert(4);
mulset_i = mulset.begin();
cout << *mp_i ;
```

输出:2

begin(set)函数

函数定义

```
iterator begin();
const_iteratoe begin() const;
```

函数说明

返回一个迭代器，指向 set 的第一个元素，

函数示例

```
set<int> st;
set<int>::iterator st_i;
st.insert(1);
st.insert(2);
st_i = st.begin();
cout << *st_i;
```

输出:1

begin(string)函数

函数定义

```
iterator begin();
const_iteratoe begin() const;
```

函数说明

返回一个迭代器，指向字符串的第一个元素，

函数示例

```
string str = “hello”;
```

```
basic_string<char>::iterator str_i;
str_i =   str.begin();
cout << *str_i;
```

输出:h

begin(vector)函数

函数定义

```
iterator begin();
const_iteratoe begin() const;
```

函数说明

返回一个迭代器，指向 vector 的第一个元素，

函数示例

```
vector<int> vec;
vector<int>::iterator vec_i;
vec.push_back(1);
vec.push_back(2);
vec_i = vec.begin();
cout << *vec_i;
```

输出: 1

C

clear(deque)函数

函数定义

```
void clear();
```

函数说明

清除双向队列中的所有元素。

函数示例

```
deque<int> dq;
dq.push_front(1);
dq.push_back(2);
cout <<”Size:”<<dq.size()<<endl;
dq.clear();
cout <<”After clear Size:”<<dq.size()<<endl;
```

输出:Size:2
After Clear Size:0

clear(list)函数

函数定义

```
void clear();
```

函数说明

清除 list 中的所有元素。

函数示例

```
list<int> lst;
lst.push_front(1);
lst.push_back(2);
```

```
cout <<”Size:”<<lst.size()<<endl;
lst.clear();
cout <<”After clear Size:”<<lst.size()<<endl;
```

输出:Size:2
After Clear Size:0

clear(set)函数

函数定义

```
void clear();
```

函数说明

清除 set 中的所有元素。

函数示例

```
set<int> st
st.insert(1);
st.insert(2);
cout <<”Size:”<<st.size()<<endl;
st.clear();
cout <<”After clear Size:”<<st.size()<<endl;
```

输出:Size:2
After Clear Size:0

clear(map)函数

函数定义

```
void clear();
```

函数说明

清除 map 的所有元素。

函数示例

```
map<int, int> mp;
map<int, int>::iterator mp_i;
typedef pair<int, int> Int_pair;
```

```
mp.insert(Int_pair(1, 1));
mp.insert(Int_pair(2, 4));
mp_i = mp.size();
cout <<”Size:”<<mp_i<<endl;
mp.clear();
mp_i = mp.clear();
cout <<”After clear Size:”<<mp.size()<<endl;
```

输出:Size:2
After Clear Size:0

clear(vector)函数

函数定义

```
void clear();
```

函数说明

清除 vector 中的所有元素。

函数示例

```
vector<int> vec;
vec.push_front(1);
vec.push_back(2);
cout <<”Size:”<<vec.size()<<endl;
vec.clear();
cout <<”After clear Size:”<<vec.size()<<endl;
```

输出:Size:2
After Clear Size:0

compare(string)函数

函数定义

```
int compare( const basic_string &str );
int compare( const char *str );
int compare( size_type index, size_type length, const basic_string &str );
int compare( size_type index, size_type length, const basic_string &str,
size_type index2, size_type length2);
```

```
int compare( size_type index, size_type length, const char *str, size_type
length2 );
```

函数说明

如果源字符串小于目标字符串，返回值 小于零；
如果源字符串等于目标字符串，返回值 等于零；
如果源字符串大于目标字符串，返回值 大于零；
各个函数功能
比较自己和 str.
比较自己的子串和 str,子串以 index 索引开始，长度为 length。
比较自己的子串和 str 的子串，其中 index2 和 length2 引用 str，index 和 length 引用自己。
比较自己的子串和 str 的子串，其中 str 的子串以索引 0 开始，长度为 length2，自己的子串以 index 开始，长度为 length。

函数示例

```
string str1 = “HELLO”;
string str2 = “HELLO”;
int nCmp;
nCmp = str1.compare(str2);
if(nCmp <   0)
cout <<”str1 < str2”<<endl;
else if(nCmp == 0)
cout <<”str1 == str2”<<endl;
else
cout <<”str1 > str2”<<endl;
```

输出:str1 == str2

copy(string)函数

函数定义

```
size_type copy( char *str, size_type num, size_type index );
```

函数说明

拷贝自己的 num 个字符到 str 中（从索引 index 开始）。返回值是拷贝的字符数。

函数示例

```
string strSrc = “Test”;
char strDst[10] = {0};
```

```
strSrc.copy(strDst, 2, 0);
cout << strDst << endl;
```

输出:Te

count(bitset)函数

函数定义

```
size_type count();
```

函数说明

函数返回 bitset 中被设置成 1 的位的个数。

函数示例

```
bitset<8> bset(6);
int nCount = bset.count();
cout << “bset:”<<bset<<endl;
cout <<”count:”<<nCount<<endl;
```

输出:bset:00000110
2

count(map)函数

函数定义

```
size_type count(const KEY_TYPE &key);
```

函数说明

如果 key 在 map 中，则返回 1， 否则返回 0；

函数示例

```
map<int, int> mp;
map<int, int>:;size_type mp_i;
typedef pair<int, int> Int_pair;
mp.insert(Int_pair(1, 2));
mp.insert(Int_pair(4, 2));
mp_i = mp.count(2);
cout << mp_i << endl;
```

```
        mp_i = mp.count(1);
        cout << mp_i << endl;
        mp_i = mp.count(3);
        cout << mp_i<<endl;
```

输出:1
1
0

count(set)函数

函数定义

```
        size_type count(const KEY_TYPE &key);
```

函数说明

如果 key 在 set 中，则返回 1， 否则返回 0；

函数示例

```
        set<int> st;
        set<int>::size_type st_i;
        st.insert(1);
        st.insert(1);
        st_i = st.count(1);
        cout << st_i<<endl;
        st_i = st.count(2);
        cout << st_i << endl;
```

输出:1
0

D

data(string)函数

函数定义

```
const char *data();
```

函数说明

返回指向自己的第一个字符的指针。

函数示例

```
string str = "hello";
const char *tmp = str.data();
cout << str << endl;
cout << tmp << endl;
```

输出: hello
hello

E

empty(deque)函数

函数定义

```
bool empty();
```

函数说明

如果双向队列是空的，返回 TRUE，否则返回 FALSE。

函数示例

```
deque<int> dq;
if(dq.empty())
cout << “empty”<<endl;
else
cout << “ not empty” << endl;
```

输出: empty

empty(dlist)函数

函数定义

```
bool empty();
```

函数说明

如果 list 是空的，返回 TRUE，否则返回 FALSE。

函数示例

```
list<int> lst;
if(lst.empty())
cout << “empty”<<endl;
else
cout << “ not empty” << endl;
```

输出: empty

empty(map)函数

函数定义

```
bool empty();
```

函数说明

如果 map 是空的，返回 TRUE，否则返回 FALSE。

函数示例

```
map<int, int> mp;
if(mp.empty())
cout << “empty”<<endl;
else
cout << “ not empty” << endl;
```

输出: empty

empty(queue)函数

函数定义

```
bool empty();
```

函数说明

如果队列是空的，返回 TRUE，否则返回 FALSE。

函数示例

```
queue<int> que;
que.push(1);
if(que.empty())
cout << “empty”<<endl;
else
cout << “ not empty” << endl;
```

输出: not empty

empty(set)函数

函数定义

```
bool empty();
```

函数说明

如果集合是空的，返回 TRUE，否则返回 FALSE。

函数示例

```
set<int> st;
st.insert(1);
if(st.empty())
cout << “empty”<<endl;
else
cout << “ not empty” << endl;
```

输出: not empty

empty()函数

函数定义

```
bool empty(stack);
```

函数说明

如果当前栈是空的，返回 TRUE，否则返回 FALSE。

函数示例

```
stack<int> s;
s.push(1);
if(s.empty())
cout << “empty”<<endl;
else
cout << “ not empty” << endl;
```

输出: not empty

empty(string)函数

函数定义

```
bool empty();
```

函数说明

如果当前字符串是空的，返回 TRUE，否则返回 FALSE。

函数示例

```
string str = “hello”;
if(dstr.empty())
cout << “empty”<<endl;
else
cout << “ not empty” << endl;
```

输出:not empty

empty(vector)函数

函数定义

```
bool empty();
```

函数说明

如果双向队列是空的，返回 TRUE，否则返回 FALSE。

函数示例

```
vector<int> vec;
if(vec.empty())
cout << “empty”<<endl;
else
cout << “ not empty” << endl;
```

输出: empty

end(deque)函数

函数定义

```
iterator end();
const_iterator end() const;
```

函数说明

返回一个指向双向队列尾部元素的迭代器。

函数示例

```
deque<int> dq;
deque<int>::iterator dq_i;
dq.push_front(1);
dq.push_back(2);
dq_i = dq.end();
dq_i--;
cout << *dq_i <<endl;
```

输出:2

end(list)函数

函数定义

```
iterator end();
const_iterator end() const;
```

函数说明

返回一个指向 list 尾部元素的迭代器。

函数示例

```
list<int> lstq;
list<int>::iterator lst_i;
lst.push_front(1);
lst.push_back(2);
lst_i = lst.end();
lst_i--;
cout << *lst_i <<endl;
```

输出:2

end(vector)函数

函数定义

```
iterator end();
const_iterator end() const;
```

函数说明

返回一个指向 vector 尾部元素的迭代器。

函数示例

```
vector<int> vec;
vector<int>::iterator vec_i;
vec.push_front(1);
vec.push_back(2);
vec_i = vec.end();
vec_i--;
cout << *vec_i <<endl;
```

输出:2

end(map)函数

函数定义

```
iterator end();
const_iterator end() const;
```

函数说明

返回一个指向 map 尾部元素的迭代器。

函数示例

```
map<int, int> mp;
map<int, int>::iterator mp_i;
typedef pair<int, int> Int_pair;
mp.insert(Int_pair(1, 2));
mp.insert(Int_pair(2, 4));
mp_i = mp.end();
mp_i--;
```

```
cout <<mp_i->first<<"\t"<<mp_i->second<<endl;
```

输出:2 4

end(set)函数

函数定义

```
iterator end();
const_iterator end() const;
```

函数说明

返回一个指向 set 尾部元素的迭代器。

函数示例

```
set<int> st;
set<int>::iterator st_i;
set.insert(1);
set.insert(2);
st_i = st.end();
st_i--;
cout <<*st_i <<endl;
```

输出:2

end(string)函数

函数定义

```
iterator end();
const_iterator end() const;
```

函数说明

返回一个指向该字符串最后一个字符的迭代器。

函数示例

```
string str(“hello”);
basic_string<char>::iterator str_i;
str_i = str.end();
str_i--;
```

```
cout << *str_i<<endl;
```

输出:o

eof(io)函数

函数定义

```
bool eof();
```

函数说明

如果达到输入文件的结尾，返回 TRUE，否则返回 FALSE。

函数示例

```
fstream fs;
int tmp = 1;
fs.open(“e:\test.txt”);//共测试用的一个空文件
cout <<fs.eof()<<endl;
fs >> n;
cout << fs.eof() << endl;
```

输出:0
1

erase(deque)函数

函数定义

```
iterator erase(iterator pos);
iterator erase(iterator start, iterator end);
```

函数说明

删除 pos 位置上的元素，或者是 start 和 end 之间的所有元素，返回一个指向被删除元素后一个元素的迭代器。

函数示例

```
deque<int> dq;
deque<int>::iterator dq_i;
dq.pusth_back(1);
dq.push_back(2);
```

```
dq.push_back(3);
dq_i = dq.begin();
cout << “begin:”<<*dq_i<<endl;
dq.erase(dq_i);
dq_i = dq.begin();
cout <<”Now begin:”<< *dq_i<<endl;
```

输出: begin:1
Now begin:2

erase(list)函数

函数定义

```
iterator erase(iterator pos);
iterator erase(iterator start, iterator end);
```

函数说明

删除 pos 位置上的元素，或者是 start 和 end 之间的所有元素，返回一个指向被删除元素后一个元素的迭代器。

函数示例

```
list<int> lst;
lst.push_back(2);
lst.push_back(4);
lst.push_back(5);
cout <<”First element:”<<lst.at(0)<<endl;
lst.erase(lst.begin());
cout<< “Now First element:”<<lst.at(0)<<endl;
```

输出:First element:2
Now First element:4

erase(vector)函数

函数定义

```
iterator erase(iterator pos);
iterator erase(iterator start, iterator end);
```

函数说明

删除 pos 位置上的元素，或者是 start 和 end 之间的所有元素，返回一个指向被删除元素后一个元素的迭代器。

函数示例

```
vector<int> vec;
vec.push_back(5);
vec.push_back(6);
vec.push_back(7);
vec.push_back(8);
vec.erase(vec.at(0), vec.at(2));// 删除 5， 6， 7
cout << vec.at(0)<<endl;
```

输出:8

erase(map)函数

函数定义

```
iterator erase(iterator pos);
iterator erase(iterator start, iterator end);
size_type erase(const KEY_TYPE &key);
```

函数说明

删除 pos 位置上的元素，或者是 start 和 end 之间的所有元素，或者删除所有被 key 指定的键值

函数示例

```
map<int, int> mp;
map<int, int>::iterator mp_i;
ypedef pair<int, int> Int_pair;
int I = 0;
for( i; i < 3; i++)
{
mp.insert(Int_pair(i, i + 1));
}
mp.erase(1);
for(mp_i = mp.begin; mp_i != mp.end(); mp_i++)
{
```

```
cout<<mp_i->first<<mp_i->second<<endl;
}
```

输出:01
23

erase(set)函数

函数定义

```
iterator erase(iterator pos);
iterator erase(iterator start, iterator end);
size_type erase(const KEY_TYPE &key);
```

函数说明

删除 pos 位置上的元素，或者是 start 和 end 之间的所有元素，或者删除所有被 key 指定的键值

函数示例

```
set<int> st;
set<int>::iterator st_i;
st.insert(1);
st.insert(2);
st.insert(3);
st.erase(st.begin());
for(st_i = st.begin(); st_i != st.end(); st_i++)
{
cout << *st_i<<endl;
}
```

输出：2
3

erase(string)函数

函数定义

```
iterator erase(iterator pos);
iterator erase(iterator start, iterator end);
basic_string &erase(size_type index = 0, size_type num = npos);
```

函数说明

删除 pos 指向的字符，返回指向下一个字符的迭代器。

删除从 start 到 end 的所有字符，返回一个迭代器，指向被删除的最后一个字符的下一个字符的位置。

删除从 index 索引开始的 num 个字符。

函数示例

```
string str(“helloworld”);
str.erase(str.begin());
cout << str<< endl;
str.erase(2);
cout << str<<endl;
```

输出:elloword

el

F

fill(io)函数

函数定义

```
char fill();
char fill(char ch);
```

函数说明

返回当前填充字符，还可设置当前字符为 ch.

函数示例

```
cout << setw(5)<<'a'<<endl;
cout.fill('x');
cout <<sew(5) <<'a'<<endl;
cout <<cout.fill()<<endl;
```

输出:　　a
xxxxa
x

find(map)函数

函数定义

```
iterator find(const KET_TYPE &key);
```

函数说明

返回一个指向键值 key 的迭代器，如果没找到键值 key ,返回指向尾部的迭代器。

函数示例

```
map<int, int> mp;
map<int, int>::iterator mp_i;
typedef pair<int, int> Int_pair;
mp.insert(Int_pair(1, 2));
```

```
mp.insert(Int_pair(2, 3));
mp.insert(Int_pair(4, 5));
mp_i = mp.find(3);
if(mp_i == mp.end())
cout<<”not find”<<endl;
else
cout << mp_i->second<<endl;
```

输出:not find

find(set)函数

函数定义

```
iterator find(const KET_TYPE &key);
```

函数说明

返回一个指向键值 key 的迭代器，如果没找到键值 key ,返回指向尾部的迭代器。

函数示例

```
set<int> st;
set<int>::iterator st_i;
st.insert(1);
st.insert(2);
st.insert(4);
st_i = st.find(2);
if(st_i == st.end())
cout <<”Not find”<<endl;
else
cout <<*st_i<<endl;
```

输出：2

find(string)函数

函数定义

```
size_type find( const basic_string &str, size_type index );
size_type find( const char *str, size_type index );
size_type find( const char *str, size_type index, size_type length );
```

```
size_type find( char ch, size_type index );
```

函数说明

返回 str 在字符串中第一次出现的位置（从 index 开始查找）。如果没找到则返回 string::npos.

返回 str 在字符串中第一次出现的位置（从 index 开始查找，长度为 length）。如果没找到就返回 string::npos.

返回字符 ch 在字符串中第一次出现的位置（从 index 开始查找）。如果没找到就返回 string::npos.

函数示例

```
string str("abcdefg");
int local = str.find('d', 0);
if(local != string::npos)
cout <<"d at:"<< local <<endl;
else
cout <<"not find"<<endl;
```

输出:d at:3

find_first_not_of(string)函数

函数定义

```
size_type find_first_not_of( const basic_string &str, size_type index = 0 );
size_type find_first_not_of( const char *str, size_type index = 0 );
size_type find_first_not_of( const char *str, size_type index, size_type num );
size_type find_first_not_of( char ch, size_type index = 0 );
```

函数说明

在字符串中查找第一个与 str 中的字符都不匹配的字符，返回它的位置。搜索从 index 开始。如果没找到，就返回 string::npos.

在字符串中查找第一个与 str 中的字符都不匹配的字符，返回它的位置。搜索从 index 开始，最多查找 num 个字符，如果没找到就返回 string::npos.

在字符串中查找第一个与 ch 不匹配的字符，返回它的位置。搜索从 index 开始。如果没找到就返回 string::npos.

函数示例

```
string str("abcdefg");
string strtmp("abc");
```

```
int local = str.find_first_not_of(strtmp, 0);
if(local != string::npos)
cout <<"local:"<< local <<endl;
else
cout <<"not find"<<endl;
```

输出:3

find_first_of(string)函数

函数定义

```
size_type find_first_of( const basic_string &str, size_type index = 0 );
size_type find_first_of( const char *str, size_type index = 0 );
size_type find_first_of( const char *str, size_type index, size_type num );
size_type find_first_of( char ch, size_type index = 0 );
```

函数说明

返回从 index 开始在字符串中查找第一个与 str 中某个字符匹配的字符的位置，如何没找到，就返回 string::npos.

返回从 index 开始在字符串中查找第一个与 str 中某个字符匹配的字符的位置，最多查找 num 个字符，如何没找到，就返回 string::npos.

返回从 index 开始在字符串中查找第一个与字符 ch 匹配的位置，如何没找到，就返回 string::npos.

函数示例

```
string str("abcdefg");
string strtmp("cde");
int local = str.find_first_of(strtmp, 0);
if(local != string::npos)
cout <<"local:"<< local <<endl;
else
cout <<"not find"<<endl;
```

输出:2

find_last_not_of(string)函数

函数定义

```
size_type find_last_not_of( const basic_string &str, size_type index = npos );
size_type find_last_not_of( const char *str, size_type index = npos);
size_type find_last_not_of( const char *str, size_type index, size_type num );
size_type find_last_not_of( char ch, size_type index = npos);
```

函数说明

返回从 index 开始在字符串中查找最后一个与 str 字符都不匹配的字符的位置，如果没找到，就返回 string::npos.
返回从 index 开始在字符串中查找最后一个与 str 字符都不匹配的字符的位置，最多搜索 num 个字符，如果没找到，就返回 string::npos.
返回从 index 开始在字符串中查找最后一个与字符 ch 不匹配的字符的位置，如果没有找到，就返回 string::npos.

函数示例

```
string str("abcdefg");
int local = str.find_last_not_of('b', 0);
if(local != string::npos)
cout <<"local:"<< local <<endl;
else
cout <<"not find"<<endl;
```

输出:6

fint_last_of(string)函数

函数定义

```
size_type find_last_of( const basic_string &str, size_type index = npos );
size_type find_last_of( const char *str, size_type index = npos );
size_type find_last_of( const char *str, size_type index, size_type num);
size_type find_last_of( char ch, size_type index = npos );
```

函数说明

返回从 index 开始在字符串中查找最后一个与 str 中的某个字符匹配的字符的位置，如果没找到，就返回 string::npos.
返回从 index 开始在字符串中查找最后一个与 str 中的某个字符匹配的字符的位置，最多查找 num 个字符，如果没找到，就返回 string::npos.
返回从 index 开始在字符串中查找最后一个与字符 ch 匹配的字符的位置，如果没找到，就返回 string::npos.

函数示例

```
string str(“abcdefg”);
int local = str.find_last_of(‘b’, 0);
if(local != string::npos)
cout <<”local:”<< local <<endl;
else
cout <<”not find”<<endl;
```

输出:1

flags(io)函数

函数定义

```
fmtflags flags();
fmtflags flags( fmtflags f );
```

函数说明

或者返回当前流的格式标志，或者为当前流设置标志为 f。

函数示例

```
cout<<cout.flags()<<endl;
cout.flags(ios::dec || ios::boolalpha);
cout << cout.flags()<<endl;
```

输出：513
16896

flip(bitset)函数

函数定义

```
bitset<N> &flip();
bitset<N> &flip(size_type pos);
```

函数说明

返回一个与原 bitset 所有位都相反的 bitset, 或则返回一个只有 pos 位被置反得 bitset.

函数示例

```
bitset<8> bt(6);
```

```
cout <<bt<<endl;
bbitset<8> bttmp = bt.flip(2);
cout << bt<<endl;
cout <<bttmp<<endl;
```

输出：00000110
00000010
00000010

flush(io)函数

函数定义

```
basic_ostream &flush();
```

函数说明

把当前流的缓冲写到输出设备，并返回一个基本输出对象。

函数示例

```
cout << “test”;
cout.flush();
```

输出:test

front(deque)函数

函数定义

```
reference front(0;
const_referenct &front() const;
```

函数说明

返回指向一个双向队列头部的引用.

函数示例

```
deque<char> dq;
dq.push_back(‘a’);
dq.push_back(‘b’);
dq.push_back(‘c’);
cout << dq.front()<<endl;
```

输出：a

front(list)函数

函数定义

```
reference front(0;
const_referenct &front() const;
```

函数说明

返回指向一个指向链表第一个元素的引用.

函数示例

```
list<char> lst;
lst.push_back('a');
lst.push_back('b');
lst.push_back('c');
cout << dq.front()<<endl;
```

输出：a

front(queue)函数

函数定义

```
reference front(0;
const_referenct &front() const;
```

函数说明

返回队列的第一个元素的引用.

函数示例

```
queue<char> que;
que.push('a');
que.push('b');
que.push('c');
char &ch = que.front();
cout << ch<<endl;
```

输出：a

front(vector)函数

函数定义

```
reference front(0;
const_referenct &front() const;
```

函数说明

返回 vector 的第一个元素的引用.

函数示例

```
vector<char> vec;
vec.push_back('a');
vec.push_back('b');
vec.push_back('c');
char &ch = vec.front();
cout << ch<<endl;
```

输出：a

fstream(io)函数

函数定义

```
fstream( const char *filename, openmode mode );
ifstream( const char *filename, openmode mode );
ofstream( const char *filename, openmode mode );
```

函数说明

ifstream，和 ofstream 对象用于文件输入/输出，可通过模式参数 mode，决定如何打开一个文件，filename 与被打开的流对象关联。

函数示例

```
ofstream fout("e:\\tmp.txt", ios::app);
int test = 10;
fout << test;// 把一个 int 值通过流写进文件
fout.close;
ifstream fin("e:\\tmp.txt");
int test;
```

```
fin >> test;//从文件读取一个 int 值。
```

G

gcount(io)函数

函数定义

```
streamsize gcount();
```

函数说明

用于输入流，返回上一次操作被读入的字符的数目。

函数示例

```
cout<<"Type the letter 'a':";
ws(cin);
char c[10];
cin.get(&c[0], 9);
cout << c << endl;
cout << cin.gcount()<<endl;
```

输出：Type the letter'a':
a
1

get(io)函数

函数定义

```
int get();
istream &get( char &ch );
istream &get( char *buffer, streamsize num );
istream &get( char *buffer, streamsize num, char delim );
istream &get( streambuf &buffer );
istream &get( streambuf &buffer, char delim );
```

函数说明

读入一个字符并返回它的值。

读入一个字符并把它存储在 ch.
读取字符到 buffer 直到 num - 1 个字符被读入，或者碰到 EOF 或换行标志.
读取字符到 buffer 直到已读如 num -1 个字符，或者碰到 EOF 或 delim(delim 直到下一次不会被读去)。
读取字符到 buffer 中，直到碰到换行或 EOF。
读取字符到 buffer 中，直到碰到换行，EOF 或 delim.

函数示例

```
char c[10];
c[0] = cin.get();
cin.get(c[1]);
cin.get(&c[2], 3);
cin.get(&c[4], 4, '7');
cout << c << endl;
```

输入：12
输出：12

get_allocator(deque)函数

函数定义

```
Allocator get_allocator() const;
```

函数说明

返回双向队列的配置器。

函数示例

```
deque<int> dq1;
deque<int, allocator<int>> dq2 = deque<int, alloator<int>>(alloator<int>());
deque<int> dq3(dq1.get_allocator());
deque<int>::allocator_type xlst = dq1.get_allocator();
```

get_allocator(list)函数

函数定义

```
Allocator get_allocator() const;
```

函数说明

返回链表的配置器。

函数示例

```
list<int> lst1;
list<int, allocator<int>> lst2 = list<int, alloator<int>>(alloator<int>());
list<int> lst3(lst1.get_allocator());
list<int>::allocator_type xlst = lst1.get_allocator()
```

get_allocator(map)函数

函数定义

```
Allocator get_allocator() const;
```

函数说明

返回 map 的配置器。

函数示例

```
map<int, int>::allocator_type type1;
map<int, doubel>::allocator_type type2;
map<int, int, allocator<int>> mp1;
map<int, double, allocator<double>> mp2;
tpye1 = mp1.get_allocator();
type2 = mp2.get_allocator();
```

get_allocator(set)函数

函数定义

```
Allocator get_allocator() const;
```

函数说明

返回集合的配置器。

函数示例

```
set<int, allocator<int> > st1 = set<int, alloator<int> >(alloator<int>());
set<int> st2(st1.get_allocator());
set<int>::allocator_type xlst = st1.get_allocator()
```

get_allocator(string)函数

函数定义

```
Allocator get_allocator() const;
```

函数说明

返回本字符串的配置器。

函数示例

```
basic_string<char> str;
basic_string<char> str2 = str.get_allocator();
basic_string<char>::allocator_type = str.get_allocator();
```

get_allocator(vector)函数

函数定义

```
Allocator get_allocator() const;
```

函数说明

返回当前 vector 的内存分配器。

函数示例

```
vector<int, allocator<int> > vec1 = vector<int, allocator<int> >(allocator<int> ());
vector<int vec2(vec1.get_allocator());
vector<int>:;allocator_type xvec = vec2.get_allocator();
```

getline(io)函数

函数定义

```
istream &getline( char *buffer, streamsize num )；
istream &getline( char *buffer, streamsize num, char delim );
```

函数说明

用于输入流，读取字符到 buffer 中，直到下列情况发生：
num - 1 个字符已经读入。
碰到一个换行标志。

碰到一个 EOF
任意地读入，直到读到字符 delim, delim 字符不会被放入 buffer 中。

函数示例

```
char ch[10];
cin.getline(&c[0], 5, '2');
cout << c <<endl;
```

输入：1245
输出：1

good(io)函数

函数定义

```
bool good();
```

函数说明

如果当前流没发生错误，返回 TRUE，否则返回 FALSE。

函数示例

```
cout << "test"<<endl;
bool b = cout.good();
cout << b << endl;
```

输出：test
1

I

ignore(io)函数

函数定义

```
istream &ignore( streamsize num=1, int delim=EOF );
```

函数说明

忽略读入的 num 个字符，直到遇到 delim 字符。

函数示例

```
char ch[10];
cout <<"Type 'abcdefg':";
cin.ignore(5, 'c');
cin >> ch;
cout << ch;
```

输入：abcdefg
输出：Type 'abcdefg'
defg

insert(deque)函数

函数定义

```
iterator insert(iterator pos, size_type num, const TYPE &val);
void insert(iterator pos, intpu_iterator start, input_iterator end);
```

函数说明

在 pos 前插入 num 个 val 值。
在 pos 前插入从 start 到 end 的值。

函数示例

```
deque<int> dq;
deque<int>::iterator dq_i;
```

```
dq.push_back(10);
dq.push_back(20);
dq.push_back(30)
for(dq_i = dq.begin(); dq_i != dq.end(); dq_i++)
{
cout << *dq_i;
}
cout << endl;
dq_i = dq.begin();
dq_i++;
dq.insert(dq_i, 100);
for(dq_i = dq.begin(); dq_i != dq.end(); dq_i++)
{
cout << *dq_i;
}
cout << endl;
```

输出:10 20 30
10 100 20 30

insert(list)函数

函数定义

```
iterator insert(iterator pos, const TYPE &val);
void insert(iterator pos, size_type num, const TYPE &val);
iterator insert(iterator pos, intpu_iterator start, input_iterator end);
```

函数说明

在 pos 处插入 val。
在 pos 前插入 num 个 val 值。
在 pos 前插入从 start 到 end 的值，返回一个指向插入元素的迭代器。

函数示例

```
list<int> lst;
list<int>::iterator lst_i;
lst_i = lst.begin();
list.push_back(10);
list.push_back(20);
```

```
for(lst_i; lst_i != lst.end(); lst_i++)
{
cout << *lst_i;
}
cout <<endl;
lst_i++;
lst.insert(lst_i, 30);
for (lst_i; lst_i != lst.end(); lst_i++)
{
cout <<*lst_i;
}
```

输出：10 20
10 30 20

insert(vector)函数

函数定义

```
iterator insert(iterator pos, const TYPE &val);
void insert(iterator pos, size_type num, const TYPE &val);
void insert(iterator pos, intpu_iterator start, input_iterator end);
```

函数说明

在 pos 处插入 val。
在 pos 前插入 num 个 val 值。
在 pos 前插入从 start 到 end 的值。

函数示例

```
vector<int> vec;
vector<int>::iterator vec_i;
vector.push_back(100);
vector.push_back(200);
vector.push_back(300);
vec_i = vec.begin();
for(vec_i; vec_i != vec.end(); vec_i++)
{
cout << *vec_i;
}
```

```
cout <<endl;
vec_i++;
vec.insert(vec_i, 2, 400);
for(vec_i = vec.begin(); vec_i !=vec.end(); vec_i++)
{
cout <<*vec_i;
}
cout <<endl;
```

输出：100 200 300
100 400 400 200 300

insert(map)函数

函数定义

```
iterator inset(iterator pos, const pair<KEY_TYPE, VALUE_TYPE> &val);
void insert(input_iterator start, input_iterator end);
pair<iterator, bool> insert(const pair<KEY_TYPE, VALUE_TYPE> &val);
```

函数说明

插入 val 到 pos 前面，然后返回一个指向这个元素的迭代器。
插入 start 到 end 的元素到 map 中。
只有在 val 不存在时才插入 val, 返回一个指向被插入的元素的迭代器和一个描述是否被插入的 bool 值。

函数示例

```
map<int, int> mp;
map<int, int>::iterator mp_i;
typedef pair<int, int> Int_pair;
mp.insert(Int_pair(1, 10));
mp.insert(Int_pair(2, 20));
mp.insert(Int_pair(3, 30));
for(mp_i = mp.begin(); mp_i != mp.end(); mp_i++)
{
cout <<mp_i->first<< "\t"<<mp_i->second<<endl;
}
cout <<endl;
pair<map<int, int>::iterator, bool> pr;
```

```
pr = mp.insert(Int_pair(1, 20));
if(pr.second == true)
cout << "Insert success"<<endl;
else
cout <<"key number 1 already exist"<<endl;
```

输出：1 10
2 20
3 30
key number 1 already exist

insert(set)函数

函数定义

```
iterator insert(iterator pos, const TYPE &val);
void insert(input_iterator start, input_iterator end);
pair<set<int>::iterator, bool> insert(const TYPE &val);
```

函数说明

在 pos 出插入 val.
将从 start 到 end 的元素插入到集合中。
只有在 val 不存在时才插入 val, 返回一个指向被插入的元素的迭代器和一个描述是否被插入的 bool 值。

函数示例

```
set<int> st;
set<int>::iterator st_i;
st.insert(10);
st.insert(20);
for(st_i = st.begin(); st_i != mp.end(); st_i++)
{
cout <<*st_i;
}
cout <<endl;
pair<set<int>::iterator, bool> pr;
pr = st.insert(10);
if(pr.second == true)
cout << "Insert success"<<endl;
```

```
        else
        cout <<"10 already exist"<<endl;
```

输出：10 20
10 already exist

insert(string)函数

函数说明

```
        iterator insert(iterator pos, const char &ch);
        basic_string &insert( size_type index, const basic_string &str );
        basic_string &insert( size_type index, const char *str );
        basic_string &insert( size_type index1, const basic_string &str, size_type
        index2, size_type num );
        basic_string &insert( size_type index, const char *str, size_type num );
        basic_string &insert( size_type index, size_type num, char ch );
        void insert(iterator pos, size_type num, const char &ch);
        void insert(iterator pos, iterator start, iterator end);
```

函数说明

在 pos 前面插入一个字符 ch。
在位置 index 处插入字符串 str。
在位置 index 处插入字符串 str 的字串(位置从 index2 开始，长 num 个字符)。
在位置 index 处插入字符串 str 的 num 个字符。
在位置 index 才插入 nim 个字符 ch 的拷贝。
在 pos 前面插入 num 个字符 ch 的拷贝
在 pos 前面插入一段从 start 开始到 end 结束的字符串。

函数示例

```
        string str(“helloworld”);
        string strtmp(“test”);
        str.insert(5, strtmp);
        cout << str<<endl
        str.insert(5, 5, ‘x’);
        cout <<str<<endl;
```

输出：hellotestworld
helloxxxxxtestworld

K

key_comp(map)函数

函数定义

```
key_compare key_comp();
```

函数说明

返回一个比较 key 的函数。

函数示例

```
map<int, int , less<int> > mp;
map<int, int, less<int> >::key_compare kmp = mp.key_comp();
bool result = kmp(2, 3);
if(result == true)
{
cout <<”kmp(2, 3) returns value of true!”<<endl;
}
else
{
cout <<”kmp(2, 3) returns value of false!”<<endl;
}
```

输出：kmp(2, 3) returns value of true

key_comp(set)函数

函数定义

```
key_compare key_comp();
```

函数说明

返回一个用于元素间值比较的一个函数对象。

函数示例

```
set <int, less<int> > st;
set<int, less<int> >::key_compare kst = st.key_comp();
bool res = kst(2, 3);
if(res == false)
cout << "false"<<endl;
else
cout << "true"<<endl
```

输出：true

L

length(string)函数

函数定义

```
size_type length();
```

函数说明

返回一个字符串的长度，和 size()的返回值相等。

函数示例

```
string str(“hello”);
cout << str.length()<<endl;
```

输出：5

lower_bound(map)函数

函数定义

```
iterator lower_bound();
```

函数说明

返回一个指向 map 中键值>=key 的第一个元素的迭代器。

函数示例

```
map<int, int> mp;
map<int, int>::iterator mp_i;
typedef pair<int, int> Int_pair;
mp.insert(Int_pair(1, 10));
mp.insert(Int_pair(2, 20));
mp.insert(Int_pair(3, 30));
mp_i = mp.lower_bound(2);
cout << mp_i->first<< "\t"<<mp_i->second;
```

输出：2 20

lower_bound(set)函数

函数定义

```
iterator lower_bound();
```

函数说明

返回指向第一个大于等于 key 值的元素的迭代器。

函数示例

```
set<int> st;
set<int>::iterator st_i;
st.insert(1);
st.insert(2);
st.insert(3);
st_i = st.lower_bound(2);
cout << *st_i<<endl;
```

输出：2

M

max_size(deque)函数

函数定义

```
size_type max_size();
```

函数说明

返回双向队列的最大长度。

函数示例

```
deque<int> dq;
deque<int>::size_type i;
i = dq.max_size();
cout <<”Max size:”<<i <<endl;
```

输出:Max size:1073741823

max_size(list)函数

函数定义

```
size_type max_size();
```

函数说明

返回链表能容纳的元素的最大数目。

函数示例

```
list<int> lst;
int i = lst.max_size();
cout <<”Max size:”<<i <<endl;
```

输出:Max size:1073741823

max_size(map)函数

函数定义

```
size_type max_size();
```

函数说明

返回 map 能容纳的元素的最大数目。

函数示例

```
map<int, int> mp;
map<int, int>::size_type i = mp.max_size();
cout <<”Max size:”<<i <<endl;
```

输出:Max size:536870911

max_size(set)函数

函数定义

```
size_type max_size();
```

函数说明

返回集合能容纳的元素的最大数目。

函数示例

```
set<int> st;
set<int>::size_type i = st.max_size();
cout <<”Max size:”<<i <<endl;
```

输出:Max size:1073741823

max_size()函数

函数定义

```
size_type max_size();
```

函数说明

返回字符串能容纳的字符的最大数目。

函数示例

```
string str;
int i = str.max_size();
cout <<"Max size:"<<i <<endl;
```

输出:Max size:1073741823

max_size(vector)函数

函数定义

```
size_type max_size();
```

函数说明

返回 vector 能容纳的元素的最大数目。

函数示例

```
vector<int> vec;
int i = vec.max_size();
cout <<"Max size:"<<i <<endl;
```

输出:Max size:1073741823

merge(list)函数

函数定义

```
void merge(list &lst);
void merge(list &lst, Comp compfunc);
```

函数说明

merge()函数把自己和 lst 链表连接在一起，产生一个整齐排列的组合链表。如果指定 compfunc，则将此函数作为比较的依据。

函数示例

```
list<int> lst1, lst2, lst3;
list<int>::iterator lst1_i, lst2_i lst3_i;
lst1.push_back(3);
lst1.push_back(6);
lst2.push_back(2);
```

```
        lst2.push_back(4);
        lst3.push_back(5);
        lst3.push_back(1);
        cout <<"lst1 = ";
        for(lst1_i = lst1.begin(); lst1_i != lst1.end(); lst1_i++)
        cout <<" "<< * lst1_i;
        cout <<endl;
        cout <<"lst2 = ";
        for(lst2_i = lst2.begin(); lst2_i != lst2.end(); lst2_i++)
        cout <<" "<< * lst2_i;
        cout <<endl;
        lst2.merge(lst1);
        lst2.sort(greater<int>());
        cout << "after merge and sort lst2 = ";
        for(lst2_i = lst2.begin(); lst2_i != lst2.end(); lst2_i++)
        cout << " "<< *lst2_i;
        cout <<endl;
        cout << "lst3 = ";
        for (lst3_i = lst3.begin(); lst3_i != lst3.end(); lst3_i++);
        cout <<" "<<*lst3_i;
        cout <<endl;
        lst2.merge(lst3, greater<int>());
        cout << "after merge lst2 = ";
        for(lst2_i = lst2.begin(); lst2_i != lst2.end(); lst2_i++)
        cout << "   "<<*lst2_i;
        cout <<endl;
```

输出：lst1 = 3 6
lst2 = 2 4
after merge and sort lst2 = 6 4 3 2
lst3 = 5 1;
after merge lst2 = 6 5 4 3 2 1

N

none(bitset)函数

函数定义

```
bool none();
```

函数说明

如果没有位被置 1 返回 TRUE，否则返回 FALSE。

函数示例

```
bitset<8> bt(2);
bool ret = bt.none();
if(ret)
cout <<”true”
else
cout <<”false”
```

输出：false

O

open(io)函数

函数定义

```
void open(const char *filename);
void open(const char *filename, openmode mode);
```

函数说明

用于打开文件流，打开以 filename 名字的文件，并与当前流关联，可选择的打开模式有：
ios::添加输出
ios::ate 当以打开时寻找到 EOF
ios::binary 以二进制模式打开
ios::in 为读取打开文件
ios::out 为写入打开文件
ios::trunc 覆盖存在的文件

函数示例

```
ifsream file;
char ch;
file.open(“e:\\test.txt”, ios:in);// 该文件内容是 test
file >> ch;
cout << ch << endl;
```

输出：t

P

peek(io)函数

函数定义

```
int peek();
```

函数说明

返回将要被读取的下一个字符，如果到文件末尾，返回 EOF，它并不读取该字符。

函数示例

```
char ch1[10], ch2;
cout <<”type ‘abcd’”;
ch2 = cin.peek();
cin.getline(ch1, 10);
cout << ch2<< ch1<<endl;
```

输出：a abcd

pop(queue)函数

函数定义

```
void pop();
```

函数说明

删除队列的一个元素，从第一个开始删。

函数示例

```
queue<int> que;
que.push(10);
que.push(20);
que.push(30);
cout <<”length:”<<que.size()<<endl;
cout <<”front:”<<que.front()<<endl;
```

```
que.pop();
cout <<”after pop”<<endl;
cout <<”length:”<<que.size()<<endl;
cout <<”front:”<<que.front()<<endl;
```

输出:length:3
front:10
after pop
length:2
front:20

pop(stack)函数

函数定义

```
void pop();
```

函数说明

移除栈顶的一个元素。

函数示例

```
stack<int> s;
for(int i = 0; i < 5; i++)
s.push(i);
cout <<”length:”<<s.size()<<endl;
cout <<”top:”<<s.top()<<endl;
s.pop();
cout <<”after pop”;
cout <<”length:”<<s.size()<<endl;
cout <<”top:”<<s.top()<<endl;
```

输出：length:5
top:4
after pop
length:4
top:4

pop_back(deque)函数

函数定义

```
void pop_back();
```

函数说明

删除双向队列尾部的一个元素。

函数示例

```
deque<int> dq;
dq.push_back(1);
dq.push_back(2);
cout << “first:”<<dq.front()<<endl;
cout <<”last:”<<dq.back()<<endl;
dq.pop_back();
cout<<”after pop_back”<<endl;
cout << “first:”<<dq.front()<<endl;
cout <<”last:”<<dq.back()<<endl;
```

输出：first:1
slast:2
after pop_back
first:1
back:1

pop_back(list)函数

函数定义

```
void pop_back();
```

函数说明

删除链表的最后一个元素。

函数示例

```
list<int> lst;
lst.push_back(1);
lst.push_back(2);
```

```
cout<<"size:"<<lst.size()<<endl;
cout<<"last:"<<lst.back()<<endl;
lst.pop_back();
cout <<"after pop_back"<<endl;
cout<<"size:"<<lst.size()<<endl;
cout<<"last:"<<lst.back()<<endl;
```

输出：size:2
last:2
size:1
last:1

pop_back(vector)函数

函数定义

```
void pop_back();
```

函数说明

删除当前 vector 的最后一个元素。

函数示例

```
vector<int> vec;
vec.push_back(10);
vec.push_back(20);
vec.push_back(30);
cout<<"size:"<<vec.size()<<endl;
cout<<"last:"<<vec.back()<<endl;
vec.pop_back();
cout <<"after pop_back"<<endl;
cout<<"size:"<<vec.size()<<endl;
cout<<"last:"<<vec.back()<<endl;
```

输出：size:3
last:30
after pop_back
size:2
last:20

pop_front(deque)函数

函数定义

```
void pop_front();
```

函数说明

删除双向队列的头部元素。

函数示例

```
deque<int> dq;
dq.push_back(1);
dq.push_back(2);
cout << “first:”<<dq.front()<<endl;
cout <<”last:”<<dq.back()<<endl;
dq.pop_front();
cout<<”after pop_front”<<endl;
cout << “first:”<<dq.front()<<endl;
cout <<”last:”<<dq.back()<<endl;
```

输出：first:1
last:2
after pop_front
first:2
last:2

pop_front(list)函数

函数定义

```
void pop_front();
```

函数说明

删除链表的第一个元素。

函数示例

```
list<int> lst;
lst.push_back(1);
lst.push_back(2);
```

```
cout << “first:”<<lst.front()<<endl;
cout <<”last:”<<lst.back()<<endl;
lst.pop_front();
cout<<”after pop_front”<<endl;
cout << “first:”<<lst.front()<<endl;
cout <<”last:”<<lst.back()<<endl;
```

输出：first:1
last:2
after pop_front
first:2
last:2

precision(io)函数

函数定义

```
streamsize precision();
streamsize precision(streamsize    size);
```

函数说明

设置或者返回当前要被显示的浮点变量的位数。

函数示例

```
float pi = 3.1415926;
cout.precision(3);
cout << pi <<endl;
```

输出：3.14

push(queue)函数

函数定义

```
void push(const TYPE &val);
```

函数说明

向队列中加入一个元素。

函数示例

```
queue<int> que;
que.push(10);
que.push(20);
que.push(30);
cout << “front:”<<que.front()<<endl;
```

输出：front:10

push(stack)函数

函数定义

```
void push(const TYPE &val);
```

函数说明

将值 val 压入栈，使其成为栈顶元素。

函数示例

```
stack<int> s;
s.push(10);
s.push(20);
s.push(30);
cout << “front:”<<s.top()<<endl;
```

输出：front:30

push_back(deque)函数

函数定义

```
void push_back(const TYPE &val);
```

函数说明

向双向队列尾部加入一个元素。

函数示例

```
deque<int> dq;
dq.push_back(10);
cout <<”last:”<<dq.back()<<endl;
dq.push_back(20);
```

```
cout <<”last:”<<dq.back()<<endl;
```

输出：last:10
last:20

push_back(list)函数

函数定义

```
void push_back(const TYPE &val);
```

函数说明

将 val 连接到链表的最后一个位置。

函数示例

```
list<int> lst;
lst.push_back(10);
cout <<”last:”<<lst.back()<<endl;
lst.push_back(20);
cout <<”last:”<<lst.back()<<endl;
```

输出：last:10
last:20

push_back(vector)函数

函数定义

```
void push_back(const TYPE &val);
```

函数说明

在 vector 末尾添加一个值为 val 的元素。

函数示例

```
vector<int> vec;
vec.push_back(10);
cout <<”last:”<<vec.back()<<endl;
vec.push_back(20);
cout <<”last:”<<vec.back()<<endl;
```

输出：last:10

last:20

push_front(deque)函数

函数定义

```
void push_front(const TYPE &val);
```

函数说明

向双向队列头部加入一个元素。

函数示例

```
deque<int> dq;
dq.push_front(10);
cout <<”first:”<<dq.front()<<endl;
dq.push_front(20);
cout <<”first:”<<dq.front()<<endl;
```

输出：first:10
first:20

push_front(list)函数

函数定义

```
void push_front(const TYPE &val);
```

函数说明

将值 val 连接到链表头部。

函数示例

```
list<int> lst;
lst.push_front(10);
cout <<”first:”<<lst.front()<<endl;
lst.push_front(20);
cout <<”first:”<<lst.front()<<endl;
```

输出：first:10
first:20

put(io)函数

函数定义

```
ostream &put(char ch);
```

函数说明

用于输出流，把字符 ch 写入流中。

函数示例

```
cout.put('c');
cout<<endl;
```

输出：c

putback(io)函数

函数定义

```
istream &putback( char ch );
```

函数说明

用于输入流，返回以前读入的字符 ch 到输入流中。

函数示例

```
char c1[10], c2, c3;
c2 = cin.get();
c3 = cin.get();
cin.putback(c2);
cin.getline(c1, 9);
cout << c1<<endl;
```

输入：ab
输出：a

R

rbegin(deque)函数

函数定义

```
reverse_iterator rbegin();
```

函数说明

返回一个指向双向队列尾部的逆向迭代器。

函数示例

```
deque<int> dq;
deque<int>::reverse_iterator dq_i;
dq.push_back(1);
dq.push_back(2);
dq.push_back(3);
dq_i = dq.rbegin();
cout <<”last:”<<*dq_i<<endl;
```

输出：3

rbegin(list)函数

函数定义

```
reverse_iterator rbegin();
```

函数说明

返回一个指向链表尾部的逆向迭代器。

函数示例

```
list<int> lst;
list<int>::reverse_iterator lst_i;
lst.push_back(1);
lst.push_back(2);
```

```
lst.push_back(3);
lst_i lst.rbegin();
cout<<”last:”<<*lst_i<<endl;
```

输出：last:3

rbegin(map)函数

函数定义

```
reverse_iterator rbegin();
```

函数说明

返回一个指向 map 尾部的逆向迭代器。

函数示例

```
map<int, int> mp;
map<int, int>::reverse_iterator mp_i;
typedef pair<int, int> Int_pair;
map.insert(Int_pair(1, 10));
map.insert(Int_pair(2, 20));
map.insert(Int_pair(3, 30));
mp_i = mp.rbegin();
cout <<”last key:”<<mp_i->first<<endl;
```

输出：last key:3

rbegin(set)函数

函数定义

```
reverse_iterator rbegin();
```

函数说明

返回当前集合中指向最后一个元素的逆向迭代器。

函数示例

```
set<int> st;
set<int>::reverse_iterator st_i;
st.insert(10);
```

```
st.insert(20);
st.insert(30);
st_i = st.rbegin();
cout <<”last:”<<*st_i<<endl;
```

输出：last:30

rbegin(string)函数

函数定义

```
reverse_iterator rbegin();
```

函数说明

返回一个逆向迭代器，指向最后一个字符。

函数示例

```
string str(“hello”);
basic_string<char>::reverse_iterator str_i;
str_i = str.rbegin();
str_i--;
cout <<”last letter:”<<*str_i<<endl;
```

输出：last letter: o

rbeing(vector)函数

函数定义

```
reverse_iterator rbegin();
```

函数说明

返回指定当前 vector 末尾的逆向迭代器。

函数示例

```
vector<int> vec3;
vector<int>::reverse_iterator vec3_i;
vec3.push_back(1);
vec3.push_back(2);
vec3_i = vec3.rbegin();
```

```
        cout<<"vec3:"<<*vec3_i<<endl;
```

输出;vec3:2

read(io)函数

函数定义

```
        istream &read(char *buffer, streamsize num);
```

函数说明

用于输入流，读取 num 个字符到 buffer 中，如果遇到 EOF，则终止。

函数示例

```
        char ch[10];
        int count = 5;
        cin.read(ch, 5);
        ch[count] = 0;
        cout << ch <<endl;
```

输入：abcdefg
输出：abcde

remove(list)函数

函数定义

```
        void remove(const TYPE &val);
```

函数说明

删除链表中所有值为 val 的元素。

函数示例

```
        list<int> lst;
        lst.push_front(1);
        lst.push_back(2);
        lst.push_back(3);
        cout<<”front:”<<lst.front()<<endl;
        lst.remove(1);
        cout<<”after remove(1)<<endl;
```

```
        cout<<”front:”<<lst.front()<<endl;
```

输出：front:1
after remove(1)
front:2

rend(deque)函数

函数定义

```
        reverse_iterator rend();
```

函数说明

返回一个指向双向队列头部的逆向迭代器。

函数示例

```
        deque<int> deq;
        deque<int>::iterator deq_i;
        deque<int>::reverse_iterator deq_ri;
        deq.push_back(1);
        deq.push_back(2);
        deq.push_back(3);
        for(deq_i = deq.begin(); deq_i != deq.end(); deq_i++)
        cout <<" "<<*deq_i;
        cout <<endl;
        deq_ri = deq.rbegin();
        for(deq_ri; deq_ri != deq.rend(); deq_ri++)
        cout <<" "<<*deq_ri;
        cout <<endl;
```

输出：1 2 3
3 2 1

rend(list)函数

函数定义

```
        reverse_iterator rend();
```

函数说明

返回一个指向链表头部的逆向迭代器。

函数示例

```
list<int> lst;
list<int>::iterator lst_i;
list<int>::reverse_iterator lst_ri;
lst.push_back(1);
lst.push_back(2);
lst.push_back(3);
for(lst_i = lst.begin(); lst_i != lst.end(); lst_i++)
cout <<" "<<*lst_i;
cout <<endl;
for(lst_ri = lst.rbegin(); lst_ri != lst.rend(); lst_ri++)
cout <<" "<<*lst_ri;
cout <<endl ;
```

输出：1 2 3
3 2 1

rend(map)函数

函数定义

```
reverse_iterator rend();
```

函数说明

返回一个指向 map 头部的逆向迭代器。

函数示例

```
map<int, int> mp;
map<int, int>::iterator mp_i;
map<int, int>::reverse_iterator mp_ri;
typedef pair<int, int> Int_pair;
mp.insert(Int_pair(1, 10));
mp.insert(Int_pair(2, 20));
mp.insert(Int_pair(3, 30));
for(mp_i = mp.begin(); mp_i != mp.end(); mp_i++)
cout <<"("<<mp_i->first<<","<<mp_i->second<<")"<<endl;
for(mp_ri=mp.rbegin(); mp_ri != mp.rend(); mp_ri++)
```

```
cout <<"("<<mp_ri->first<<","<<mp_ri->second<<")"<<endl;
```

输出：（1, 10）
(2,20)
(3,30)
(3,30)
(2,20)
(1,10)

rend(set)函数

函数定义

```
reverse_iterator rend();
```

函数说明

返回指向当前集合中最后一个元素的逆向迭代器。

函数示例

```
set<int> st;
set<int>::iterato st_i;
set<int>::reverse_iterator st_ri;
st.insert(100);
st.insert(200);
st.insert(300);
for(st_i = st.begin(); st_i != st.end(); st_i++)
cout << " "<<*st_i;
cout <<endl;
for(st_ri = st.rbegin(); st_ri != st.rend(); st_ri++)
cout <<" "<<*st_ri;
cout <<endl;
```

输出：100 200 300
300 200 100

rend(string)函数

函数定义

```
reverse_iterator rend();
```

函数说明

返回一个逆向迭代器，指向字符串开头。

函数示例

```
string str("HELLO");
basic_string<char>::iterator str_i;
basic_string<char>::reverse_iterator str_ri;
cout <<str<<endl;
str_ri = str.rbegin();
for(str_ri; str_ri != str.rend(); str_ri++)
cout << *str_ri;
cout <<endl;
```

输出：HELLO
OLLEH

rend(vector)函数

函数定义

```
reverse_iterator rend();
```

函数说明

返回指向当前 vector 起始位置的逆向迭代器。

函数示例

```
vertor<int> vec;
vector<int>::reverse_iterator vec_ri;
vec.push_back(1);
vec.push_back(2);
for(vec_ri = vec.rbegin(); vec_ri != vec.rend(); vec_ri++)
cout << *vec_ri<<endl;
```

输出：2
1

replace(string)函数

函数定义

```
basic_string &replace( size_type index, size_type num, const basic_string
&str );
basic_string &replace( size_type index1, size_type num1, const basic_string
&str, size_type index2，size_type num2);
basic_string &replace( size_type index, size_type num, const char *str );
basic_string &replace( size_type index, size_type num1, const char *str,
size_type num2 );
basic_string &replace( size_type index, size_type num1, size_type num2, char
ch );
basic_string &replace( iterator start, iterator end, const basic_string &str );
basic_string &replace( iterator start, iterator end, const char *str );
basic_string &replace( iterator start, iterator end, const char *str, size_type
num );
basic_string &replace( iterator start, iterator end, size_type num, char ch );
```

函数说明

用 str 中的 num 个字符替换本字符串中从 index 开始的字符。

用 str 中的 num2 个从 index2 开始的字符替换本字符串中的从 index1 开始，最多 num1 个字符。

用 str 中的 num 个从 index 开始的字符替换本字符串中的字符

用 str 中的 num2 个从 index2 开始的字符替换本字符串中的从 index1 开始，num1 个字符。

用 num2 个 ch 字符替换本字符串中从 index 开始的字符。

用 str 中的字符替换本字符串中的迭代器 start 和 end 指示的内容。

用 str 中的 num 个指示的字符替换本字符串的迭代器 start 和 end 指示的内容。

用 num 个 ch 字符替换本字符串中的迭代器 start 和 end 指示范围的内容。

函数示例

```
string str(“helloworld”);
basic_string<char>::iterator str_i1, str_i2;
string rep1(“bigd ”);
string rep2(“tsing”);
str.replace(5, rep1.length(), rep1);
cout << str<<endl;
str_i1 = str.begin();
str_i2 = str.begin() + 5;
str.replace(str_i1, str_i2, rep2);
cout << str<<endl;
```

输出:hellobigd //bigd 后面有个空格
tsingbigd //bigd 后面有个空格

reserve(string)函数

函数定义

```
void reserve(size_type num=0);
```

函数说明

设置本字符串的 capacity，在字符串内保留 num 个字符空间。

函数示例

```
string str(“hello, wold”);
cout <<”Original capacity:”<<str.capacity()<<endl;
str.reserve(30);
cout <<”after reserve capacity:”<<str.capacity()<<endl;
```

输出：Original capacity:11
after reserve capacity:30

reserve(vector)函数

函数定义

```
void reserve(size_type num=0);
```

函数说明

为当前 vector 预留至少能容纳 num 个元素的空间。

函数示例

```
vector<int> vec;
vec.push_back(1);
cout<<”Original capacity:”<<vec.capacity()<<endl;
vec.reseve(20);
cout<<”after reserve capacity:”<<vec.capacity()<<endl;
```

输出：Original capacity:1
after reserve capacity:20

reset(bitset)函数

函数定义

```
bitset &reset();
bitset &reset(size_type pos);
```

函数说明

重置 bitset,即所有位置 0，如果知道 pos，则只有 pos 位置 0，其他位不变。

函数示例

```
bitset<8> bt(6);
cout <<bt<<endl;
bt.reset();
cout << bt <<endl;
bt.reset(1);
cout << bt<< endl;
```

输出;00000110
00000000
00000010

resize(deque)函数

函数定义

```
void resize(size_type num, TYPE val);
```

函数说明

改变双向队列的大小为 num，比原来的双向队列多出的元素用 val 填充。

函数示例

```
deque<int> dq;
deque<int>::iterator dq_i;
dq.push_back(1);
dq.push_back(2);
for(dq_i = dq.begin(); dq_i != dq.end(); dq_i++)
cout<<" "<<*dq_i;
cout <<endl;
```

```
dq.resize(5, 5);
for(dq_i = dq.begin(); dq_i != dq.end(); dq_i++)
cout<<" "<<*dq_i;
cout <<endl;
```

输出：1 2
1 2 5 5 5

resize(list)函数

函数定义

```
void resize(size_type num, TYPE val);
```

函数说明

改变链表的大小为 num，比原来链表多出的元素用 val 填充。

函数示例

```
list<int> lst;
list<int>::iterator lst_i;
lst.push_back(1);
lst.push_back(2);
for(lst_i = lst.begin(); lst_i != lst.end(); lst_i++)
cout<<" "<<*lst_i;
cout <<endl;
lst.resize(5, 5);
for(lst_i = lst.begin(); lst_i != lst.end(); lst_i++)
cout<<" "<<*lst_i;
cout <<endl;
```

输出：1 2
1 2 5 5 5

resize(vector)函数

函数定义

```
void resize(size_type num, TYPE val);
```

函数说明

改变当前 vector 的大小为 num，比原来 vector 多出的元素用 val 填充。

函数示例

```
vector<int> vec;
vector<int>::iterator vec_i;
vec.push_back(1);
vec.push_back(2);
for(vec_i = vec.begin(); vec_i != vec.end(); vec_i++)
cout<<" "<<*vec_i;
cout <<endl;
vec.resize(5, 5);
for(vec_i = vec.begin(); vec_i != vec.end(); vec_i++)
cout<<" "<<*vec_i;
cout <<endl;
```

输出：1 2
1 2 5 5 5

resize(string)函数

函数定义

```
void resize(size_type num);
void resize(size_type num, char ch);
```

函数说明

改变字符串的大小为 num，新空间的值不确定，或则 用 ch 字符填充。

函数示例

```
string str("hello");
cout << str<<endl;
str.resize(3);
cout <<str<<endl;
str.resize(5, 'x');
cout <<str<<endl;
```

输出：hello
hel
helxx

reverse(list)函数

函数定义

```
void reverse();
```

函数说明

把链表所有元素倒转。

函数示例

```
list<int> lst;
list<int>::iterator lst_i;
lst.push_back(1);
lst.push_back(2);
lst.push_back(3);
cout <<”original:”<<endl;
for(lst_i = lst.begin(); lst_i != lst.end(); lst_i++)
cout <<” “<<*lst_i;
cout <<endl;
lst.reverse();
cout<<”reverse:”<<endl;
for(lst_i = lst.begin(); lst_i != lst.end(); lst_i++)
cout <<” “<<*lst_i;
cout <<endl;
```

输出：1 2 3
3 2 1

refind(string)函数

函数定义

```
size_type rfind( const basic_string &str, size_type index );
size_type rfind( const char *str, size_type index );
size_type rfind( const char *str, size_type index, size_type num );
size_type rfind( char ch, size_type index );
```

函数说明

返回从字符串开始到 index 查找最后一个与 str 中的某个字符匹配的字符的位置。如果

没找到就返回 string::npos.

返回从字符串开始到 index 查找，最多查找 num 个字符的最后一个与 str 中的某个字符匹配的字符的位置。如果没找到就返回 string::npos.

返回从字符串开始到 index 查找最后一个与 ch 匹配的字符的位置。如果没找到就返回 string::npos.

函数示例

```
string str(“this is a test”);
int loc;
loc = str.rfind(“is”, 8);
cout << loc<<endl;
loc = str.rfinf(“is”, 3);
cout <<loc<<endl;
```

输出：5

2

S

seekg(io)函数

函数定义

```
istream &seekg( off_type offset, ios::seekdir origin );
istream &seekg( pos_type pos );
```

函数说明

函数 seekg()用于输入流，并且它将重新设置"get"指针到当前流的从 origin 便宜 offset 个字节的位置上，或者是值 get 指针到 pos 位置。

函数示例

```
ifstream file;
char ch;
file.open("e:\\test.txt");//test.txt 内容：0123456789
file.seekg(2);
file >> ch;
cout << ch<<endl;
file.seekg( 1, ios_base::beg );
file >> ch;
cout << ch <<endl;
file.seekg( -2, ios_base::end );
file >> ch;
cout << ch << endl;
```

输出：2
1
8

seekp(io)函数

函数定义

```
ostream &seekg( off_type offset, ios::seekdir origin );
```

```
ostream &seekg( pos_type pos );
```

函数说明

用于输出流，功能和 seekg 类似，参照 seekg.

函数示例

```
ofstream file("e:\\test.txt");//test.txt 内容:0123456789
streamoff i = file.tellp();
cout << i << endl;
file <<"testing";
i = file.tellp();
cout << i << endl;
file.seekp(2);
file<<" ";//在文件位置 2 出读入一个空格
file.seekp(2, ios::end);
file << 'z';
```

输出：0
7

set(bitset)函数

函数定义

```
bitset &set();
bitset &set(size_type pos, bool val = 1);
```

函数说明

设置 bitset 所有的位为 1，如果指定 pos 和 val，那么就只设定 pos 上的值为 val。

函数示例

```
bitset<8> bt(6);
cout << bt<< endl;
cout << bt.set()<<endl;
cout << bt.set(0, 0)<<endl;
```

输出：00000110
11111111
11111110

setf(io)函数

函数定义

```
fmtflags setf( fmtflags flags );
fmtflags setf( fmtflags flags, fmtflags needed );
```

函数说明

设置当前流的格式化标志为 flags，返回前一个设置的标志。

函数示例

```
int tmp = 10;
cout << tmp << endl;
cout.unsetf(ios_base::dec);
cout.setf(ios_base::hex);
cout <<tmp<<endl;
cout.setf(ios_base::dec);
cout << tmp<<endl;
```

输出：10
a
10

size(bitset)函数

函数定义

```
size_type size();
```

函数说明

返回 bitset 能容纳的位数。

函数示例

```
bitset<8> bt(1);
cout << bt <<endl;
cout <<"Size:"<<bt.size()<<endl
```

输出：00000001
8

size(deque)函数

函数定义

```
size_type size();
```

函数说明

返回双向队列中的元素的个数。

函数示例

```
deque<int> dq;
dq.push_back(1);
dq.push_back(2);
cout <<"size:"<<dq.size()<<endl;
dq.resize(5);
cout << "resize:"<<dq.size()<<endl;
```

输出：size:2
resize:5

size(list)函数

函数定义

```
size_type size();
```

函数说明

返回链表中元素的数量。

函数示例

```
list<int> lst;
lst.push_back(1);
lst.push_back(2);
cout <<"size:"<<lst.size()<<endl;
```

输出：size:2

size(map)函数

函数定义

```
size_type size();
```

函数说明

返回 map 中保存的元素的个数。

函数示例

```
map<int, int> mp;
map<int, int>::iterator mp_i;
map<int, int>::reverse_iterator mp_ri;
typedef pair<int, int> Int_pair;
mp.insert(Int_pair(1, 10));
mp.insert(Int_pair(2, 20));
mp.insert(Int_pair(3, 30));
cout<<"size:"<<mp.size()<<endl;
```

输出：size:3

size(queue)函数

函数定义

```
size_type size();
```

函数说明

返回队列中元素的个数。

函数示例

```
queue<int> que;
que.push(1);
que.push(2);
que.push(4);
cout <<"size:"<<que.size()<<endl;
```

输出：size:3

size(set)函数

函数定义

```
size_type size();
```

函数说明

返回当前集合中的元素数目。

函数示例

```
set<int> st;
int i = 0;
for(i ; i < 10; i++)
st.insert(i);
cout <<”size:”<<st.size()<<endl;
```

输出：size:10

size(stack)函数

函数定义

```
size_type size();
```

函数说明

返回当前堆栈中的元素数目。

函数示例

```
stack<int> stk;
int k = 0;
for( k; k <20; k++)
stk.push(k);
cout <<”size:”<<stk.size()<<endl;
```

输出：size:20

size(string)函数

函数定义

```
size_type size();
```

函数说明

返回当前字符串中拥有的字符数。

函数示例

```
string str(“hello”);
cout <<”size:”<<str.size()<<endl;
```

输出：size:5

size(vector)函数

函数定义

```
size_type size();
```

函数说明

返回当前 vector 能容纳的元素的数数。

函数示例

```
vector<int> vec;
int m = 0;
for(m; m <6; m++)
vec.push_back(m);
cout <<”size:”<<vec.size()<<endl;
```

输出：size:6

sort(list)函数

函数定义

```
void sort();
void sort(Comp compfunc);
```

函数说明

该函数为链表排序，默认为升序，如果指定 compfunc，则用该函数判定两个元素的大小。

函数示例

```
list<int> lst;
list<int>::iterator lst_i;
lst.push_back(20);
lst.push_back(10);
lst.push_back(30);
for(lst_i = lst.begin(); lst_i != lst.end(); lst_i++)
cout <<" "<<*lst_i;
cout << endl;
lst.sort();
for(lst_i = lst.begin(); lst_i != lst.end(); lst_i++)
cout <<" "<<*lst_i;
cout << endl;
lst.sort(greater<int>());
for(lst_i = lst.begin(); lst_i != lst.end(); lst_i++)
cout <<" "<<*lst_i;
cout << endl;
```

输出：20 10 30
10 20 30
30 20 10

splice(list)函数

函数定义

```
void splice(iterator pos, list &lst);
void splice(iterator pos, list &lst, iterator val);
void splice(iterator pos, list &lst, iterator start, iterator end);
```

函数说明

把 lst 连接到链表的 pos 位置。

函数示例

```
list<int> lst1, lst2, lst3;
list<int>::iterator lst_i;
lst1.push_back(11);
lst1.push_back(12);
lst2.push_back(21);
lst2.push_back(22);
```

```
lst_i = lst1.begin();
lst1.splice(lst_i, lst2);
cout <<"test splice"<<endl;
for(lst_i = lst1.begin(); lst_i != lst1.end(); lst_i++)
cout << " "<<*lst_i;
cout << endl;
```

输出：21 22 11 12

substr(string)函数

函数定义

```
basic_string substr( size_type index, size_type num = npos );
```

函数说明

返回本字符串的一个字串，从 index 开始长 num，如果没有指定将默认为 string::npos，这样就返回从 index 开始到结尾的字串。

函数示例

```
string str(“hello, world”);
string strtest = str.substr(3);
cout << str <<endl;
cout << strtest<<endl;
```

输出：hello, world
lo, world

swap(deque)函数

函数定义

```
void swap(deque &val);
void swap(deque &src, deque &dst);
```

函数说明

交换 val 和现在双向队列的元素。
交换双向队列 src 和 dst 的元素。

函数示例

```
deque<int> dq1, dq2, dq3;
deque<int>::iterator dq_i;
dq1.push_back(1);
dq1.push_back(2);
dq1.push_back(3);
dq2.push_back(10);
dq2.push_back(20);
dq3.push_back(100);
for(dq_i = dq1.begin(); dq_i != dq1.end(); dq_i++)
cout <<" "<<*dq_i;
cout << endl;
dq1.swap(dq2);
for(dq_i = dq1.begin(); dq_i != dq1.end(); dq_i++)
cout <<" "<<*dq_i;
cout << endl;
swap(dq1, dq3);
for(dq_i = dq1.begin(); dq_i != dq1.end(); dq_i++)
cout <<" "<<*dq_i;
cout << endl;
```

输出：1 2 3
10 20
100

swap(list)函数

函数定义

```
void swap(list &val);
void swap(list &src, list &dst);
```

函数说明

交换 val 和现在链表的元素。
交换链表 src 和 dst 的元素。

函数示例

```
list<int> lst1, lst2, lst3;
list<int>::iterator lst_i;
lst1.push_back(1);
```

```
        lst1.push_back(2);
        lst1.push_back(3);
        lst2.push_back(10);
        lst2.push_back(20);
        lst3.push_back(100);
        cout <<" lst1 original:"<<endl;
        for(lst_i = lst1.begin(); lst_i != lst1.end(); lst_i++)
        cout <<" "<<*lst_i;
        cout <<endl;
        lst1.swap(lst2);
        cout <<"lst1 swap with lst2:"<<endl;
        for(lst_i = lst1.begin(); lst_i != lst1.end(); lst_i++)
        cout <<" "<<*lst_i;
        cout <<endl;
        swap(lst1, lst3);
        cout <<"lst1 swap with lst3:"<<endl;
        for(lst_i = lst1.begin(); lst_i != lst1.end(); lst_i++)
        cout <<" "<<*lst_i;
        cout <<endl;
```

输出;lst1 original:
1 2 3
lst1 swap with lst2:
10 20
lst1 swap with lst3:
100

swap(set)函数

函数定义

```
        void swap(set &val);
        void swap(set &src, set &dst);
```

函数说明

交换 val 和现在集合的元素。
交换集合 src 和 dst 的元素。

函数示例

```
set<int> st1, st2, st3;
set<int>::iterator st_i;
st1.insert(1);
st1.insert(2);
st1.insert(3);
st2.insert(10);
st2.insert(20);
st3.insert(100);
cout <<" st1 original:"<<endl;
for(st_i = st1.begin(); st_i != st1.end(); st_i++)
cout <<" "<<*st_i;
cout <<endl;
st1.swap(st2);
cout <<"st1 swap with st2:"<<endl;
for(st_i = st1.begin(); st_i != st1.end(); st_i++)
cout <<" "<<*st_i;
cout <<endl;
swap(st1, st3);
cout <<"st1 swap with st3:"<<endl;
for(st_i = st1.begin(); st_i != st1.end(); st_i++)
cout <<" "<<*st_i;
cout <<endl;
```

输出;st1 original:
1 2 3
st1 swap with st2:
10 20
st1 swap with st3:
100

swap(map)函数

函数定义

```
void swap(map &val);
```

函数说明

交换 val 和现在 map 中的所有元素。

函数示例

```
map<int, int> mp1, mp2;
map<int, int>::iterator mp_i;
typedef pair<int, int> Int_pair;
mp1.insert(Int_pair(1, 1));
mp1.insert(Int_pair(2, 2));
mp2.insert(Int_pair(10, 10));
cout <<"original mp1:"<<endl;
for(mp_i = mp1.begin(); mp_i != mp1.end(); mp_i++)
cout <<mp_i->first<<" "<<mp_i->second<<endl;
mp1.swap(mp2);
cout <<"after swap mp1:"<<endl;
for(mp_i = mp1.begin(); mp_i != mp1.end(); mp_i++)
cout <<mp_i->first<<" "<<mp_i->second<<endl;
```

输出;original mp1:
1 1
2 2
after swap mp1:
10 10

swap(string)函数

函数定义

```
void swap(basic_string &str);
```

函数说明

将本字符串和字符串 str 交换。

函数示例

```
string first("first");
string second("second");
cout <<"First:"<<first<<endl;
cout <<"Second:"<<second<<endl;
first.swap(second);
cout <<"after swap:"<<endl;
cout <<"First:"<<first<<endl;
cout <<"Second:"<<second<<endl;
```

输出：First:first
Second:second
after swap:
First:second
Second:first

swap(vector)函数

函数定义

```
void swap(vector &from);
void swap(vector &from, vector &to);
```

函数说明

把 from 的内容和该 vector 的内容交换。
交换 from 和 to 的内容。

函数示例

```
vector<int> vec1, vec2;
vector<int>::iterator vec_i;
vec1.push_back(1);
vec1.push_back(11);
vec1.push_back(111);
vec2.push_back(2);
vec2.push_back(22);
vec2.push_back(222);
cout<<"original vec1:"<<endl;
for(vec_i = vec1.begin(); vec_i != vec1.end(); vec_i++)
cout << " "<<*vec_i;
cout <<endl;
vec1.swap(vec2);
cout <<"after swap vec1:"<<endl;
for(vec_i = vec1.begin(); vec_i != vec1.end(); vec_i++)
cout << " "<<*vec_i;
cout <<endl;
```

输出：original vec1:
1 11 111
after swap vec1:

2 22 222

T

tellg(io)函数

函数定义

```
pos_type tellg();
```

函数说明

用于输入流，返回流中‘get’指针当前的位置。

函数示例

```
ifstream file;
char c;
streamoff i;
file.open("basic_istream_tellg.txt");//文件内容:0123456789
i = file.tellg();
file >> c;
cout << c << " " << i << endl;
i = file.tellg();
file >> c;
cout << c << " " << i << endl;
```

输出：0 0
1 1

tellp(io)函数

函数定义

```
pos_type tellp();
```

函数说明

用于输出流，返回当前流中‘put’指针的位置。

函数示例

```
string str(“test”);
ofstream fout(“e:\\output.txt”);
int k;
for(k = 0; k < str.length(); k++)
{
cout<<”File point:”<<fout.tellp();
fout.put(str[k]);
cout <<” “<<str[k]<<endl;
}
fout.close();
```

输出：File point:0 t
File point:1 e
File point:2 s
File point:3 t

test(bitset)函数

函数定义

```
bool test(size_type pos);
```

函数说明

返回当前 bitset 的 pos 上的值，如果超出其位数，抛出 out of rang 异常。

函数示例

```
bitset<8> bt(4);
cout <<”位 2 = “<<bt.test(2)<<endl;
tyr{
cout <<”位 10 = “<<bt.test(10)<<endl;
} catch(exception &e){
cout<<”out of rang exception”<<endl;
}
```

输出;位 2 = 1
out of rang exception

to_string(bitset)函数

函数定义

```
basic_string<CharType, Traits, Alloc> to_string() const;
```

函数说明

返回 bitset 的字符串形式。

函数示例

```
string str;
bitset<8> bt(12);
str = bt.to_string<char, char_traits<char>, allocator<char> >();
cout << bt<<endl;
cout <<str<<endl;
```

输出：00001100
00001100

to_ulong(bitset)函数

函数定义

```
unsigned long to_ulong() const;
```

函数说明

返回 bitset 的无符号长整数形式。

函数示例

```
bitset<8> bt(7);
unsigned long tmp = bt.to_ulong();
cout << bt<<endl;
cout << tmp<<endl;
```

输出：00000111
7

top(stack)函数

函数定义

```
TYPE &top();
```

函数说明

返回堆栈的栈顶元素。

函数示例

```
stack<int> st;
st.push(1);
st.push(2);
cout << st.top()<<endl;
```

输出:2

U

unique(list)函数

函数定义

```
void unique();
void unique(BinaryPredicate bp);
```

函数说明

删除链表中所有相邻重复的元素，如果指定 bp，则通过 bp 判定是否删除。

函数示例

```
list<int> lst;
list<int>::iterator lst_i;
lst.push_back(1);
lst.push_back(2);
lst.push_back(2);
lst.push_back(1);
for(lst_i = lst.begin(); lst_i != lst.end(); lst_i++)
cout <<" "<<*lst_i;
cout <<endl;
lst.unique();
cout<<"after unique"<<endl;
for(lst_i = lst.begin(); lst_i != lst.end(); lst_i++)
cout <<" "<<*lst_i;
cout <<endl;
```

输出：1 2 2 1
after unique
1 2 1

upper_bound(map)函数

函数定义

```
iterator upper_bound(const KEY_TYPE &key);
```

函数说明

返回一个指向 map 中键值大于等于 key 的第一个元素的迭代器。

函数示例

```
map<int, int> mp;
map<int, int>::iterator mp_i;
typedef pair<int, int> Int_pair;
mp.insert(Int_pair(1, 10));
mp.insert(Int_pair(2, 20));
mp.insert(Int_pair(3, 30));
cout <<"map:"<<endl;
for(mp_i = mp.begin(); mp_i != mp.end(); mp_i++)
cout <<mp_i->first<<" "<<mp_i->second<<endl;
mp_i = mp.upper_bound(2);
cout<<"first > 2 key"<<endl;
cout << mp_i ->first<<" "<<mp_i->second<<endl
```

输出:map:
1 10
2 20
3 30
first > 2 key
3 30

upper_bound(set)函数

函数定义

```
iterator upper_bound(const KEY_TYPE &key);
```

函数说明

返回当前集合中第一个大于 key 值的元素的迭代器。

函数示例

```
set<int> st;
st.insert(1);
st.insert(2);
st.insert(3);
```

```
        cout <<” first bigger than 2 is:”;
        cout <<*st.upper_bound(2)<<endl;
```

输出：first bigger than 2 is:3

W

width(io)函数

函数定义

```
streamsize width();
streamsize width(streamsize wide);
```

函数说明

返回当前宽度，或者设置当前的宽度。宽度值一次显示的字符的个数。

函数示例

```
cout.width( 20 );
cout << cout.width( ) << endl;
cout << cout.width( ) << endl;
```

输出：　　20//20 前有 19 个空格。
0

write(io)函数

函数定义

```
ostream &write(const char *buffer, streamsize num);
```

函数说明

此函数用于输出流，从 buffer 中写 num 个字符到输出流中。

函数示例

```
char ch[16] = "test text";
ofstream out( "test.txt" );
streamsize size = 6;
file.write( ch, size);
```