



程序设计实习

郭炜 微博 <http://weibo.com/guoweiofpku>

<http://blog.sina.com.cn/u/3266490431>

刘家瑛 微博 <http://weibo.com/pkuliujiaying>



标准模板库STL

概述

容器

vector

deque

list

set/multiset

map/multimap

stack

queue

priority_queue

容器上的迭代器类别

随机访问

随机访问

双向

双向

双向

不支持迭代器

不支持迭代器

不支持迭代器

容器

vector

deque

list

set/multiset

map/multimap

stack

queue

priority_queue

容器上的迭代器类别

随机访问

随机访问

双向

双向

双向

不支持迭代器

不支持迭代器

不支持迭代器

有的算法，例如sort，binary_search需要通过随机访问迭代器来访问容器中的元素，那么list以及关联容器就不支持该算法！



vector的迭代器是随机迭代器，

遍历 vector 可以有以下几种做法 (deque亦然)：

```
vector<int> v(100);
```

```
int i;
```

```
for(i = 0; i < v.size() ; i ++)
```

```
    cout << v[i]; //根据下标随机访问
```

```
vector<int>::const_iterator ii;
```

```
for( ii = v.begin(); ii != v.end (); ii ++ )
```

```
    cout << * ii;
```

```
for( ii = v.begin(); ii < v.end (); ii ++ )
```

```
    cout << * ii;
```

//间隔一个输出：

```
ii = v.begin();
```

```
while( ii < v.end()) {
```

```
    cout << * ii;
```

```
    ii = ii + 2;
```

```
}
```

list 的迭代器是双向迭代器,

正确的遍历list的方法:

```
list<int> v;
```

```
list<int>::const_iterator ii;
```

```
for( ii = v.begin(); ii != v.end (); ++ii )
```

```
    cout << * ii;
```

错误的做法:

```
for( ii = v.begin(); ii < v.end (); ii ++ )
```

```
    cout << * ii;
```

//双向迭代器不支持 <, list没有 [] 成员函数

```
for(int i = 0; i < v.size() ; i ++)
```

```
    cout << v[i];
```



ERROR

算法简介

- 算法就是一个个函数模板，大多数在`<algorithm>` 中定义
- STL中提供能在各种容器中通用的算法，比如查找，排序等
- 算法通过迭代器来操纵容器中的元素。许多算法可以对容器中的一个局部区间进行操作，因此需要两个参数，一个是起始元素的迭代器，一个是终止元素的后面一个元素的迭代器。比如，排序和查找
- 有的算法返回一个迭代器。比如 `find()` 算法，在容器中查找一个元素，并返回一个指向该元素的迭代器
- 算法可以处理容器，也可以处理普通数组

算法示例：find()

```
template<class InIt, class T>  
InIt find(InIt first, InIt last, const T& val);
```

- `first` 和 `last` 这两个参数都是容器的迭代器，它们给出了容器中的查找区间起点和终点 `[first, last)`。区间的起点是位于查找范围之中的，而终点不是。**`find`在 `[first, last)` 查找等于 `val` 的元素**
- 用 **`==`** 运算符判断相等
- 函数返回值是一个迭代器。如果找到，则该迭代器**指向被找到的元素**。如果找不到，则该迭代器**等于 `last`**


```
#include <vector>
#include <algorithm>
#include <iostream>
using namespace std;
int main() { //find算法示例
    int array[10] = { 10,20,30,40};
    vector<int> v;
    v.push_back(1);    v.push_back(2);
    v.push_back(3);    v.push_back(4);
    vector<int>::iterator p;
    p = find(v.begin(),v.end(),3);
    if( p != v.end())
        cout << * p << endl; //输出3
```

输出:
3
not found
3
20

```
p = find(v.begin(),v.end(),9);
```

```
if( p == v.end())
```

```
    cout << "not found " << endl;
```

```
p = find(v.begin()+1,v.end()-2,1); //整个容器：[1,2,3,4], 查找区间：[2,3)
```

```
if( p != v.end())
```

```
    cout << * p << endl;
```

```
int * pp = find( array,array+4,20); //数组名是迭代器
```

```
cout << * pp << endl;
```

```
}
```

输出：

3

not found

3

20

STL中“大”“小”的概念

- 关联容器内部的元素是从小到大排序的
- 有些算法要求其操作的区间是从小到大排序的，称为“有序区间算法”
例：binary_search
- 有些算法会对区间进行从小到大排序，称为“排序算法”
例：sort
- 还有一些其他算法会用到“大”，“小”的概念
使用STL时，在缺省的情况下，以下三个说法等价：
 - 1) x比y小
 - 2) 表达式“ $x < y$ ”为真
 - 3) y比x大

STL中“相等”的概念

- 有时，“x和y相等”等价于“**x==y为真**”

例：在未排序的区间上进行的算法，如顺序查找find

.....

- 有时“x和y相等”等价于“**x小于y和y小于x同时为假**”

例：

有序区间算法，如binary_search

关联容器自身的成员函数find

.....

STL中“相等”概念演示

```
#include <iostream>
#include <algorithm>
using namespace std;

class A
{
    int v;
public:
    A(int n):v(n) { }
    bool operator < ( const A & a2) const {
        cout << v << "<" << a2.v << "?" << endl;
        return false;
    }
    bool operator ==(const A & a2) const {
        cout << v << "==" << a2.v << "?" << endl;
        return v == a2.v;
    }
};
```

STL中“相等” 概念演示

```
int main()
{
    A a [] = { A(1),A(2),A(3),A(4),A(5) };
    cout << binary_search(a,a+4,A(9)); //折半查找
    return 0;
}
```

STL中“相等” 概念演示

```
int main()
{
    A a [] = { A(1),A(2),A(3),A(4),A(5) };
    cout << binary_search(a,a+4,A(9));
    return 0;
}
```

输出结果:

3<9?

2<9?

1<9?

9<1?

1

In-Video Quiz

1. 假设p和p1是set上的迭代器，以下5个表达式有几个是非法的？

(1) ++p

(2) --p

(3) p+1

(4) p<p1

(5) p[2]

A) 1 B) 2 C) 3 D) 4

2. binary_search在查找过程中，比较元素和被查找的值是否相等时，用哪个运算符进行比较？

A) = B) == C) < D) < 和 ==，一个都不能少