STL容器

前注：

STL（标准模板库）是一个C++的软件库，也是C++标准程序库的一部分。

这些容器，应该都是STL里面的一个类。

vector封装数组、list封装链表、map和set封装二叉树

一、list

在不懂的时候，list可以理解为双向链表（很像，但事实上不是）。

（1）声明一个list对象：

①包含头文件list：#include<list>

②声明他：std::list<int> one; //声明一个list对象

③需要注意，list位于std名称空间之中，因此如果使用using namespace std，可以省略std::

（2）使用 **迭代器**：

①迭代器，在不懂的时候，可以理解为指针，指向对象。但事实上，迭代器不是指针（例如，指针可以加一个int值，但迭代器是不可以的）。

②声明一个list类迭代器：std::list<int>::iterator pr = one.begin(); //一个迭代器，用于指向one的第一个对象

③迭代器可以使用++和-- 这样的形式；

pr++;

pr--;

++pr;

--pr;

效果是：++（指向list对象内的下一项），--（指向上一个项）；

**Note**：假如当前迭代器已经指向最后或最初，则不能超出界限，否则会出错（例如最后时不能+，第一个时不能-）

④利用迭代器插入一个成员（有疑问）。

int a = 1;

one.insert(pr, a);

//注意，根据观察，每添加一个对象，迭代器便会自动指向下一个位置。

⑤查询当前容器里有多少个项目：

cout << one.size();

⑥删除一个成员

更多的使用，参考链接：

http://www.cppblog.com/Lee7/archive/2008/04/14/47036.html

**成员函数概观**[[编辑](https://zh.wikipedia.org/w/index.php?title=List_(STL)&action=edit&section=2" \o "编辑小节：成员函数概观)]

* 迭代 (Iterator)
  + list.begin() 回传指向第一个元素的 Iterator。
  + list.end() 回传指向最末元素的下一个位置的 Iterator。
  + list.rbegin() 回传指向最末个元素的反向 Iterator。
  + list.rend() 回传指向第一个元素的前一个位置的反向 Iterator。
* Capacity/Size:
  + list.empty() 若list内部为空，则回传true值。
  + list.size() 回传list内实际的元素个数。
  + list.resize() 重新分派list的长度。
* 存取元素的方法
  + list.front() 存取第一个元素。
  + list.back() 存取最末个元素。
* Modify methods
  + list.push\_front() 增加一个新的元素在 list 的前端。
  + list.pop\_front() 删除 list 的第一个元素。
  + list.push\_back() 增加一个新的元素在 list 的尾端。
  + list.pop\_back() 删除 list 的最末个元素。
  + list.insert() - 插入一个或多个元素至 list内的任意位置。
  + list.erase() - 删除 list中一个或多个元素。
  + list.clear() - 清空所有元素。
* 重新配置／重设长度
  + list.reserve() - 如有必要，可改变 list的容量大小（配置更多的内存）。
  + list.resize() - 改变 list目前持有的元素个数。

二、map

如代码：

#include<map>

map<char\*, int> MAP; //声明一个map

MAP["first"] = 1; //first是关键字key，1是值。值和key是成对出现的。可以把first理解为下标，只不过和平常的int下标不同

MAP["second"] = 2;

MAP["qqq"] = 3;

map<char\*, int>::iterator pr = MAP.begin(); //迭代器指向开始（first）

pr++; //迭代器指向下一个

cout << pr->first << " : "; //输出key值，注意不要括号

cout << pr->second << endl; //输出值（在这里就是int值）

MAP.erase(pr++); //删除当前迭代器指向的元素，需要使用pr++，如果在这行之后使用的话，迭代器会无法使用（除非重新赋值）

cout << pr->first << " : "; //此时迭代器指向的原本的第三个（当前的第二个）

cout << pr->second << endl;

pr--; //此时迭代器指向原本的第一个（当前第一个），原本第二个被删除了

MAP.insert(pr,{ "ww",4 }); //插入一个（插入位置是最后，第三个），当目前迭代器还指向的是第一个

pr++;

pr++; //

cout << pr->first << " : "; //此时迭代器指向的原本的第三个（当前的第二个）

cout << pr->second << endl;

三、vector

如代码：

#include<vector>

vector<int>abc(10); //类型为int，数量为10

abc[1] = 1; //第二个元素赋值为1

cout << abc[1] << endl; //输出第二个元素

vector<int>::iterator pr = abc.begin(); //迭代器指向第一个元素

pr++;

cout << \*pr << endl; //输出该元素的值

cout << abc.size() << endl; //数组元素数量

abc.front(); //数组第一个元素的引用

abc.back(); //数组最后一个元素的引用

其他用法：

<https://zh.wikipedia.org/wiki/Vector_(STL)>

### **成员函数概观**[[编辑](https://zh.wikipedia.org/w/index.php?title=Vector_(STL)&action=edit&section=2" \o "编辑小节：成员函数概观)]

vector 类是以[容器（Container）](http://www.sgi.com/tech/stl/Container.html) 模式为基准设计的，也就是说，基本上它有 begin()，end()，size()，max\_size()，empty() 以及 swap() 这几个方法。

* 访问元素的方法
  + vec[i] - 访问索引值为 i 的元素引用。 (索引值从零起算，故第一个元素是vec[0]。)
  + vec.at(i) - 访问索引值为 i 的元素的引用，以 at() 访问会做数组边界检查，如果访问越界将会抛出一个例外，这是与operator[]的唯一差异。
  + vec.front() - 回传 vector 第一个元素的引用。
  + vec.back() - 回传 vector 最尾元素的引用。
* 新增或移除元素的方法
  + vec.push\_back() - 新增元素至 vector 的尾端，必要时会进行内存配置。
  + vec.pop\_back() - 删除 vector 最尾端的元素。
  + vec.insert() - 插入一个或多个元素至 vector 内的任意位置。
  + vec.erase() - 删除 vector 中一个或多个元素。
  + vec.clear() - 清空所有元素。
* 获取长度/容量
  + vec.size() - 获取 vector 目前持有的元素个数。
  + vec.empty() - 如果 vector 内部为空，则传回 true 值。
  + vec.capacity() - 获取 vector 目前可容纳的最大元素个数。这个方法与内存的配置有关，它通常只会增加，不会因为元素被删减而随之减少。
* 重新配置／重置长度
  + vec.reserve() - 如有必要，可改变 vector 的容量大小（配置更多的内存）。在众多的 STL 实做，容量只能增加，不可以减少。
  + vec.resize() - 改变 vector 目前持有的元素个数。
* 迭代 (Iterator)
  + vec.begin() - 回传一个Iterator，它指向 vector 第一个元素。
  + vec.end() - 回传一个Iterator，它指向 vector 最尾端元素的下一个位置（请注意：它不是最末元素）。
  + vec.rbegin() - 回传一个反向Iterator，它指向 vector 最尾端元素的。
  + vec.rend() - 回传一个Iterator，它指向 vector 的第一个元素。

四、set

（1）set以结点的方式来存储。

（2）插入和删除之后，iterator可能失效。

（3）set中使用二分查找法，效率是f(n)=log n;

（4）常用代码：

#include<set>

set<int>abc; //声明一个set对象

abc.insert(1); //插入一个

abc.insert(100); //再插入一个

abc.insert(50); //插入第三个

set<int>::iterator pr = abc.begin(); //迭代器指向第一个

pr++; //指向下一个，注意，不是插入的第二个（100），而是值从小到大的第二个50

cout << \*pr << endl; //输出50

cout << abc.size() << endl; //abc总共的元素个数

cout << abc.count(50) << endl; //输出值50在整个abc里面出现的次数

cout << abc.max\_size() << endl; //abc的最大能容纳的个数（很多很多），大概是INT\_MAX的1/10