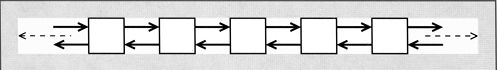
List 使用双向链表来管理元素。



**一. List 与 vector或deque的区别**

1.List 不支持随机存取。如果你要存取第5个元素，就必须从头开始数。

2.任何位置插入和删除都很快。

3.插入和删除**不会**造成其它元素pointers， references 和 iterators 失效。

4.List 的迭代器是双向迭代器，所以凡是用到随机存取的迭代器的算法（特别是排序算法）都不能用，只能用list 提供的特殊版本。

**二. Remove() 函数**

List 提供了 deque 所有的功能，还增加了 remove() 和 remove\_if() 算法

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/lwbeyond/article/details/7307135)

1. //移除所有其值为 val 的元素
2. c.remove (val)
3. //移除所有op为true的元素
4. c.remove\_if (op)

//移除所有其值为 val 的元素

c.remove (val)

//移除所有op为true的元素

c.remove\_if (op)

**c.remove() 与 remove() 函数的区别：**

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/lwbeyond/article/details/7307135)

1. #include <list>
2. #include <algorithm>
3. using namespace std;
4. **int** main(**int** argc, **char**\* argv[])
5. {
6. list<**int**> coll;
7. //6 5 4 3 2 1 1 2 3 4 5 6
8. for (**int** i=1; i<=6; ++i)
9. {
10. coll.push\_front(i);
11. coll.push\_back(i);
12. }
13. //打印
14. cout << "pre: ";
15. copy (coll.begin(), coll.end(), //source
16. ostream\_iterator<**int**> (cout," ")); //destination
17. cout << endl;
18. //全局 remove 函数
19. //remove (coll.begin(), coll.end(), 3); //6 5 4 2 1 1 2 4 5 6 5 6
20. //list 版本的 remove 函数
21. coll.remove (3); //6 5 4 2 1 1 2 4 5
22. //打印
23. cout << "post: ";
24. copy (coll.begin(), coll.end(), //source
25. ostream\_iterator<**int**> (cout," ")); //destination
26. cout << endl;
27. return 0;
28. }

#include <list>

#include <algorithm>

using namespace std;

int main(int argc, char\* argv[])

{

list<int> coll;

//6 5 4 3 2 1 1 2 3 4 5 6

for (int i=1; i<=6; ++i)

{

coll.push\_front(i);

coll.push\_back(i);

}

//打印

cout << "pre: ";

copy (coll.begin(), coll.end(), //source

ostream\_iterator<int> (cout," ")); //destination

cout << endl;

//全局 remove 函数

//remove (coll.begin(), coll.end(), 3); //6 5 4 2 1 1 2 4 5 6 5 6

//list 版本的 remove 函数

coll.remove (3); //6 5 4 2 1 1 2 4 5

//打印

cout << "post: ";

copy (coll.begin(), coll.end(), //source

ostream\_iterator<int> (cout," ")); //destination

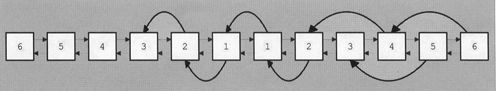
cout << endl;

return 0;

}

**注意：**

全局的 remove() 函数并没有改变容量元素的数量，并且在操作上是后面的元素覆盖了前面的元素。



而 list.remove() 不会出现这种情况。

**三. 变动性函数**

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/lwbeyond/article/details/7307135)

1. //移除重复元素，只留下一个
2. c.unique()
3. c.unique(op)
4. //将 c2 所有元素转移到 c1 内，迭代器 pos 之前
5. c1.splice(pos, c2)
6. //将 c2 内 c2pos 所指元素转移到 c1 内，迭代器 pos 之前
7. c1.splice(pos, c2, c2pos)
8. //将 c2 内 [c2beg,c2end) 区间内的所有元素转移到 c1 内的迭代器 pos 之前
9. c1.splice(pos, c2, c2beg, c2end)
10. //按从小到大排序
11. c.sort()
12. //按op排序
13. c.sort(op)
14. //c1 和 c2 容器都是已序的，将 c2 的全部元素转移到 c1，并保证合并后的 list 仍为已序。
15. c1.merge(c2)
16. //c1 和 c2 容器都是已序好，将 c2 的全部元素转移到 c1，并保证合并后的 list 在op()原则下仍为已序
17. c1.merge(c2, op)
18. 将所有元素反序
19. c.reverse()

//移除重复元素，只留下一个

c.unique()

c.unique(op)

//将 c2 所有元素转移到 c1 内，迭代器 pos 之前

c1.splice(pos, c2)

//将 c2 内 c2pos 所指元素转移到 c1 内，迭代器 pos 之前

c1.splice(pos, c2, c2pos)

//将 c2 内 [c2beg,c2end) 区间内的所有元素转移到 c1 内的迭代器 pos 之前

c1.splice(pos, c2, c2beg, c2end)

//按从小到大排序

c.sort()

//按op排序

c.sort(op)

//c1 和 c2 容器都是已序的，将 c2 的全部元素转移到 c1，并保证合并后的 list 仍为已序。

c1.merge(c2)

//c1 和 c2 容器都是已序好，将 c2 的全部元素转移到 c1，并保证合并后的 list 在op()原则下仍为已序

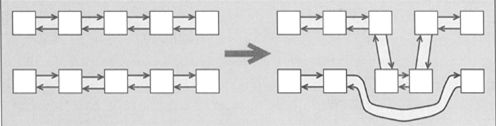
c1.merge(c2, op)

将所有元素反序

c.reverse()

**注意：**

1. 这些变动性算法，一个容器中元素增加，则另一个容器的元素必须减少。如spice 拼接操作，被拼接的c2容器内元素将不再存在。



2. merge() 操作，要求两个容器是已序的。

**四. 实例代码**

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/lwbeyond/article/details/7307135)

1. #include <iostream>
2. #include <list>
3. #include <algorithm>
4. using namespace std;
5. void printLists (const list<**int**>& list1, const list<**int**>& list2)
6. {
7. cout << "list1: ";
8. copy (list1.begin(), list1.end(), ostream\_iterator<**int**>(cout," "));
9. cout << endl << "list2: ";
10. copy (list2.begin(), list2.end(), ostream\_iterator<**int**>(cout," "));
11. cout << endl << endl;
12. }
13. **int** main()
14. {
15. //create two empty lists
16. list<**int**> list1, list2;
17. for (**int** i=0; i<6; ++i)
18. {
19. list1.push\_back(i); //0 1 2 3 4 5
20. list2.push\_front(i); //5 4 3 2 1 0
21. }
22. printLists(list1, list2);
23. //-find() 操作返回元素3的迭代器的位置
24. //把list1中的所有元素插入到list2中元素为3之前
25. list2.splice( find(list2.begin(),list2.end(), 3), list1 );
26. printLists(list1, list2);
27. //相当于倒置
28. list2.splice(list2.end(), list2, list2.begin());
29. printLists(list1, list2);
30. //排序，并去掉重复
31. list2.sort();
32. list1 = list2;
33. list2.unique();
34. printLists(list1, list2);
35. //合并
36. list1.merge(list2);
37. printLists(list1, list2);
38. return 0;
39. }