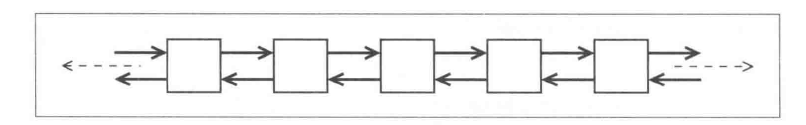
list使用双向串列管理元素。



list自身提供两个pointer，指向第1个和最后一个元素，每个元素都有两个pointer指向前一个下一个元素。

list不支持随机访问。在任何位置（不只是两端）插入和删除元素速度都很快。而且插入和删除元素不会造成指向其他元素的pointer、reference和iterator失效。

例：程序stl\_test46

void PrintLists(const *list*<int>& list1, const *list*<int>& list2)

{

*cout* << "list1: ";

*copy*(list1.*cbegin*(), list1.*cend*(), *ostream\_iterator*<int>(*cout*, " "));

*cout* << *endl* << "list2: ";

*copy*(list2.*cbegin*(), list2.*cend*(), *ostream\_iterator*<int>(*cout*, " "));

*cout* << *endl* << *endl*;

}

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

{

*list*<int> list1, list2;

for (int i = 0; i < 6; ++i)

{

list1.*push\_back*(i);

list2.*push\_front*(i);

}

PrintLists(list1, list2);

// 将list1的所有元素挪移到list2的末端

// 此时的list1将为空链表

list2.*splice*(list2.*end*(), list1);

PrintLists(list1, list2);

// 将list2的第1个元素挪到末端

list2.*splice*(list2.*end*(), list2, list2.*begin*());

PrintLists(list1, list2);

list2.*sort*();

list1 = list2;

// 移除若干相邻并且值相同的元素，只保留一个

list2.*unique*();

PrintLists(list1, list2);

// 将两个已排序的链表合并

// 并且合并后的list仍为排序

// 此时list2为空链表

list1.*merge*(list2);

PrintLists(list1, list2);

return 0;

}

输出为：

list1: 0 1 2 3 4 5

list2: 5 4 3 2 1 0

list1:

list2: 5 4 3 2 1 0 0 1 2 3 4 5

list1:

list2: 4 3 2 1 0 0 1 2 3 4 5 5

list1: 0 0 1 1 2 2 3 3 4 4 5 5

list2: 0 1 2 3 4 5

list1: 0 0 0 1 1 1 2 2 2 3 3 3 4 4 4 5 5 5

list2: