<https://blog.csdn.net/fanyun_01/article/details/56881515>

**一、List定义：**

List是stl实现的双向链表，与向量(vectors)相比, 它允许快速的插入和删除，但是随机访问却比较慢。使用时需要添加头文件

#include <list>

**二、List定义和初始化：**

    list<int>lst1;          //创建空list

    list<int> lst2(5);       //创建含有5个元素的list

    list<int>lst3(3,2);  //创建含有3个元素的list

    list<int>lst4(lst2);    //使用lst2初始化lst4

    list<int>lst5(lst2.begin(),lst2.end());  //同lst4

**三、List常用操作函数：**

Lst1.assign() 给list赋值   
Lst1.back() 返回最后一个元素   
Lst1.begin() 返回指向第一个元素的迭代器   
Lst1.clear() 删除所有元素   
Lst1.empty() 如果list是空的则返回true   
Lst1.end() 返回末尾的迭代器   
Lst1.erase() 删除一个元素   
Lst1.front() 返回第一个元素   
Lst1.get\_allocator() 返回list的配置器   
Lst1.insert() 插入一个元素到list中   
Lst1.max\_size() 返回list能容纳的最大元素数量   
Lst1.merge() 合并两个list   
Lst1.pop\_back() 删除最后一个元素   
Lst1.pop\_front() 删除第一个元素   
Lst1.push\_back() 在list的末尾添加一个元素   
Lst1.push\_front() 在list的头部添加一个元素   
Lst1.rbegin() 返回指向第一个元素的逆向迭代器   
Lst1.remove() 从list删除元素   
Lst1.remove\_if() 按指定条件删除元素   
Lst1.rend() 指向list末尾的逆向迭代器   
Lst1.resize() 改变list的大小   
Lst1.reverse() 把list的元素倒转   
Lst1.size() 返回list中的元素个数   
Lst1.sort() 给list排序   
Lst1.splice() 合并两个list   
Lst1.swap() 交换两个list   
Lst1.unique() 删除list中重复的元素

**四、List使用示例：**

示例1：遍历List

    //迭代器法

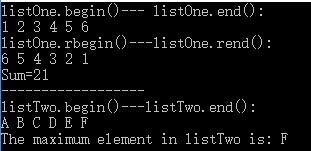
**[cpp]** [view plain](https://blog.csdn.net/fanyun_01/article/details/56881515) [copy](https://blog.csdn.net/fanyun_01/article/details/56881515)

1. **for**(list<**int**>::const\_iteratoriter = lst1.begin();iter != lst1.end();iter++)
2. {
3. cout<<\*iter;
4. }
5. cout<<endl;

示例2：

**[cpp]** [view plain](https://blog.csdn.net/fanyun_01/article/details/56881515) [copy](https://blog.csdn.net/fanyun_01/article/details/56881515)

1. #include <iostream>
2. #include <list>
3. #include <numeric>
4. #include <algorithm>
5. #include <windows.h>
6. **using** **namespace** std;
8. **typedef** list<**int**> LISTINT;
9. **typedef** list<**int**> LISTCHAR;
11. **void** main()
12. {
13. //用LISTINT创建一个list对象
14. LISTINT listOne;
15. //声明i为迭代器
16. LISTINT::iterator i;
18. listOne.push\_front(3);
19. listOne.push\_front(2);
20. listOne.push\_front(1);
22. listOne.push\_back(4);
23. listOne.push\_back(5);
24. listOne.push\_back(6);
26. cout << "listOne.begin()--- listOne.end():" << endl;
27. **for** (i = listOne.begin(); i != listOne.end(); ++i)
28. cout << \*i << " ";
29. cout << endl;
31. LISTINT::reverse\_iterator ir;
32. cout << "listOne.rbegin()---listOne.rend():" << endl;
33. **for** (ir = listOne.rbegin(); ir != listOne.rend(); ir++) {
34. cout << \*ir << " ";
35. }
36. cout << endl;
38. **int** result = accumulate(listOne.begin(), listOne.end(), 0);
39. cout << "Sum=" << result << endl;
40. cout << "------------------" << endl;
42. //用LISTCHAR创建一个list对象
43. LISTCHAR listTwo;
44. //声明i为迭代器
45. LISTCHAR::iterator j;
47. listTwo.push\_front('C');
48. listTwo.push\_front('B');
49. listTwo.push\_front('A');
51. listTwo.push\_back('D');
52. listTwo.push\_back('E');
53. listTwo.push\_back('F');
55. cout << "listTwo.begin()---listTwo.end():" << endl;
56. **for** (j = listTwo.begin(); j != listTwo.end(); ++j)
57. cout << **char**(\*j) << " ";
58. cout << endl;
60. j = max\_element(listTwo.begin(), listTwo.end());
61. cout << "The maximum element in listTwo is: " << **char**(\*j) << endl;
62. Sleep(10000);
63. }



**[cpp]** [view plain](https://blog.csdn.net/fanyun_01/article/details/56881515) [copy](https://blog.csdn.net/fanyun_01/article/details/56881515)

1. #include <iostream>
2. #include <list>
3. #include <windows.h>
5. **using** **namespace** std;
6. **typedef** list<**int**> INTLIST;
8. //从前向后显示list队列的全部元素
9. **void** put\_list(INTLIST list, **char** \*name)
10. {
11. INTLIST::iterator plist;
13. cout << "The contents of " << name << " : ";
14. **for** (plist = list.begin(); plist != list.end(); plist++)
15. cout << \*plist << " ";
16. cout << endl;
17. }
19. //测试list容器的功能
20. **void** main(**void**)
21. {
22. //list1对象初始为空
23. INTLIST list1;
24. INTLIST list2(5, 1);
25. INTLIST list3(list2.begin(), --list2.end());
27. //声明一个名为i的双向迭代器
28. INTLIST::iterator i;
30. put\_list(list1, "list1");
31. put\_list(list2, "list2");
32. put\_list(list3, "list3");
34. list1.push\_back(7);
35. list1.push\_back(8);
36. cout << "list1.push\_back(7) and list1.push\_back(8):" << endl;
37. put\_list(list1, "list1");
39. list1.push\_front(6);
40. list1.push\_front(5);
41. cout << "list1.push\_front(6) and list1.push\_front(5):" << endl;
42. put\_list(list1, "list1");
44. list1.insert(++list1.begin(), 3, 9);
45. cout << "list1.insert(list1.begin()+1,3,9):" << endl;
46. put\_list(list1, "list1");
48. //测试引用类函数
49. cout << "list1.front()=" << list1.front() << endl;
50. cout << "list1.back()=" << list1.back() << endl;
52. list1.pop\_front();
53. list1.pop\_back();
54. cout << "list1.pop\_front() and list1.pop\_back():" << endl;
55. put\_list(list1, "list1");
57. list1.erase(++list1.begin());
58. cout << "list1.erase(++list1.begin()):" << endl;
59. put\_list(list1, "list1");
61. list2.assign(8, 1);
62. cout << "list2.assign(8,1):" << endl;
63. put\_list(list2, "list2");
65. cout << "list1.max\_size(): " << list1.max\_size() << endl;
66. cout << "list1.size(): " << list1.size() << endl;
67. cout << "list1.empty(): " << list1.empty() << endl;
69. put\_list(list1, "list1");
70. put\_list(list3, "list3");
71. cout << "list1>list3: " << (list1 > list3) << endl;
72. cout << "list1<list3: " << (list1 < list3) << endl;
74. list1.sort();
75. put\_list(list1, "list1");
77. list1.splice(++list1.begin(), list3);
78. put\_list(list1, "list1");
79. put\_list(list3, "list3");
80. Sleep(10000);
81. }

