C++语言程序设计

贺利坚 主讲

顺序容器

（向量、双端队列、列表）

顺序容器的基本功能

- ► 构造

  S s(n,t); S s(n); S s(q1,q2);

- ►赋值

  s.assign(n,t); s.assign(n); s.assign(q1,q2)

- ►插入

  s.insert(p1,t); s.insert(p1,n,t); s.insert(p1, q1,q2);

  s.push_front(t); s.push_front();//对list和deque

  s.push_back(t); s.push_back();

- ►删除

  s.erase(p1); s.erase();s.clear() ,

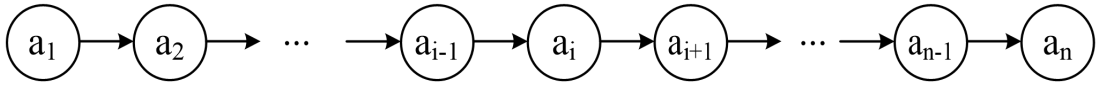
  s.pop_front(); s.pop_back(); //对list和deque

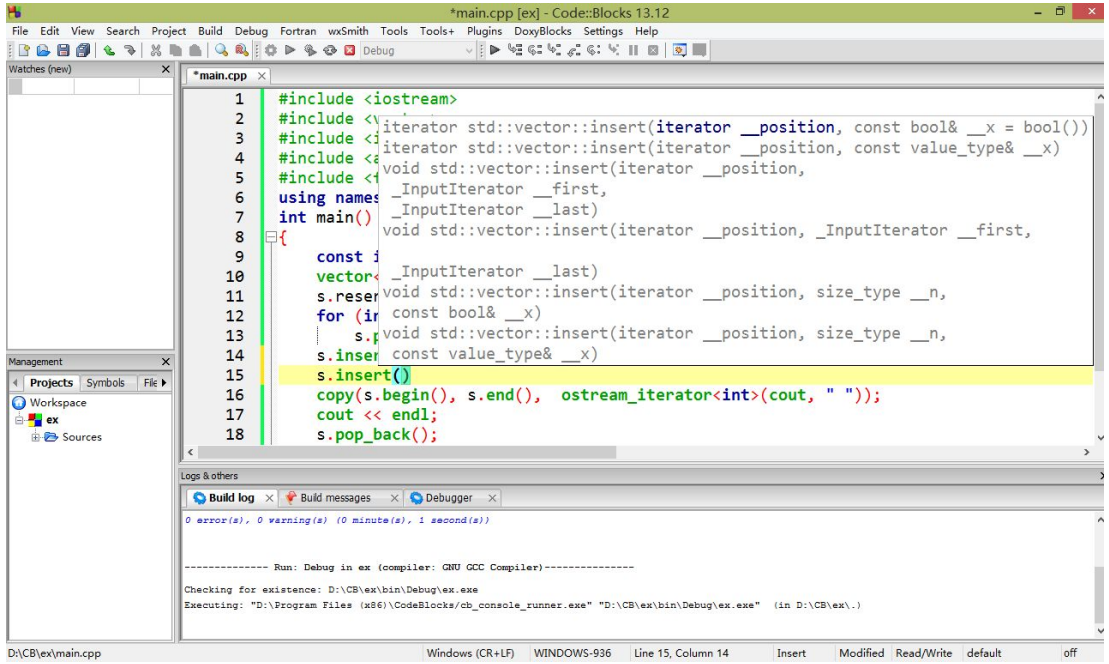
- ►首尾元素直接访问

  s.front(); s.back();

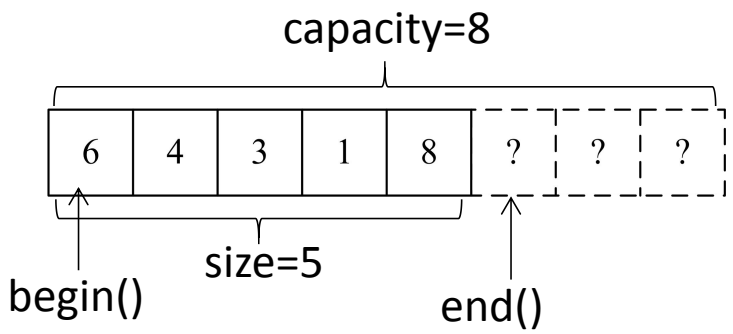
- ►改变大小

  s.resize();



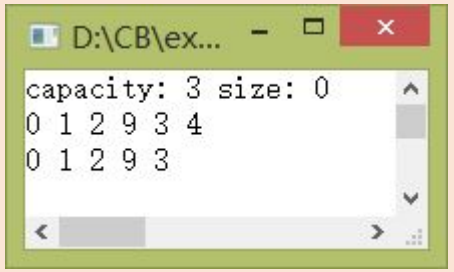


向量(Vector)

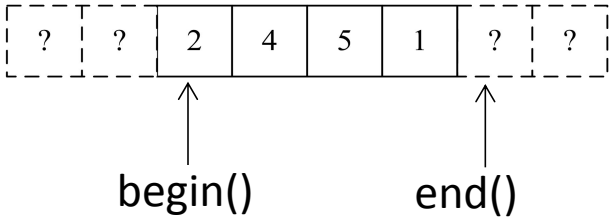


- 特点
  - 一个可以扩展的动态数组
  - 随机访问
  - 在尾部插入或删除元素快, 在中间或头部插入或删除元素慢
- 向量的容量
  - 容量(capacity)：实际分配空间的大小
  - s.capacity() ：返回当前容量
  - s.reserve(n)：若容量小于n，则对s进行扩展，使其容量至少为n

```
#include <iostream>
#include <vector>
#include <iterator>
#include <algorithm>
#include <functional>
using namespace std;
int main()
{
    const int N = 5;
    vector<int> s;
    s.reserve(3);
    cout<<"capacity: "<<s.capacity()<<" size: "<<s.size()<<endl;
    for (int i = 0; i < N; i++)
        s.push_back(i);
    s.insert(s.begin()+3, 9);
    copy(s.begin(), s.end(),  ostream_iterator<int>(cout, " "));
    cout << endl;
    s.pop_back();
    copy(s.begin(), s.end(),  ostream_iterator<int>(cout, " "));
    cout << endl;
    return 0;
}
```



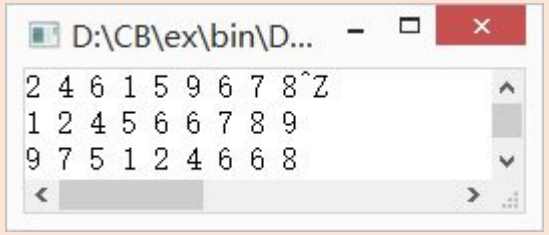
双端队列(deque)



- 特点
  - 在两端插入或删除元素快
  - 在中间插入或删除元素慢
  - 随机访问较快，但比向量容器慢
- 存储结构
  - 数组
  - 分段数组
- 再例：奇偶排序
  - 先按照从大到小顺序输出奇数，再按照从小到大顺序输出偶数。

```
#include <vector>
#include <deque>
#include <algorithm>
#include <iterator>
#include <iostream>
using namespace std;
```

```
int main()
{
    istream_iterator<int> i1(cin), i2;
    vector<int> s1(i1, i2);
    sort(s1.begin(), s1.end());
    copy(s1.begin(), s1.end(), ostream_iterator<int>(cout, " "));
    cout << endl;
    deque<int> s2;

    for (vector<int>::iterator iter = s1.begin(); iter != s1.end(); ++iter)
    {
        if (*iter % 2 == 0)
            s2.push_back(*iter);
        else
            s2.push_front(*iter);
    }

    copy(s2.begin(), s2.end(), ostream_iterator<int>(cout, " "));
    cout << endl;
    return 0;
}
```



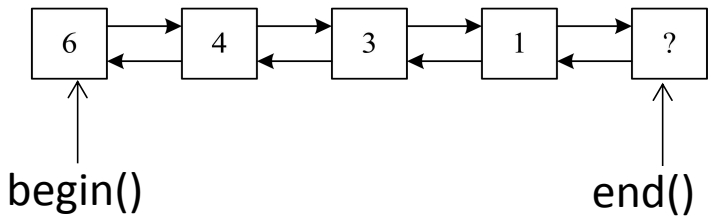
双端队列再例：

```
#include <iostream>
#include <list>
#include <deque>
#include <iterator>
#include <algorithm>
using namespace std;
template <class T>
void printContainer(const char* msg, const T& s)
{
    cout << msg << ": ";
    copy(s.begin(), s.end(), ostream_iterator<int>(cout, " "));
    cout << endl;
}
```

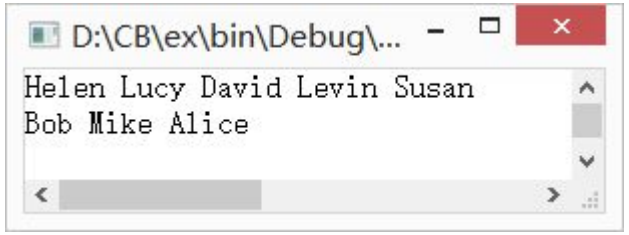
```
for (int i = 0; i < 10; i++)
{
    int x;
    cin >> x;
    s.push_front(x);
}
```

```
int main()
{
    deque<int> s;
    //输入数据
    printContainer("deque at first", s);

    list<int> l(s.rbegin(), s.rend());
    printContainer("list at first", l);

    list<int>::iterator iter = l.begin();
    while (iter != l.end())
    {
        int v = *iter;
        iter = l.erase(iter);
        l.insert(++iter, v);
    }
    printContainer("list at last", l);

    s.assign(l.begin(), l.end());
    printContainer("deque at last", s);
    return 0;
}
```

列表(list)



- 特点
  - 在任意位置插入和删除元素都很快
  - 不支持随机访问
- 接合(splice)操作
  - s1.splice(p, s2, q1, q2)：将s2中[q1, q2)移动到s1中p所指向元素之前
- 例：列表容器的splice操作

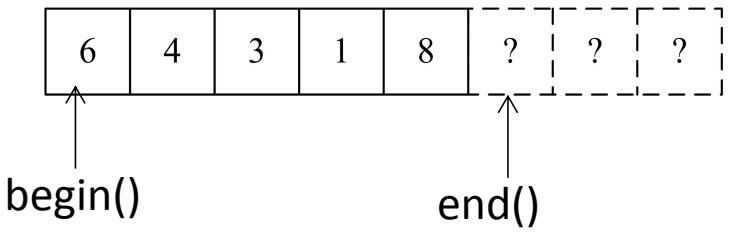


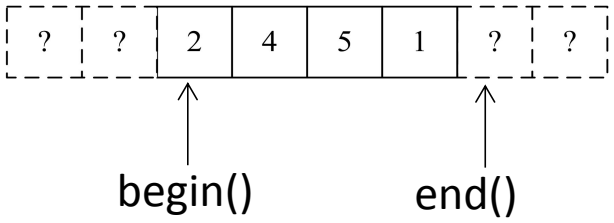

```
int main()
{
    string names1[] = { "Alice", "Helen", "Lucy", "Susan" };
    string names2[] = { "Bob", "David", "Levin", "Mike" };
    list<string> s1(names1, names1 + 4);
    list<string> s2(names2, names2 + 4);

    s2.splice(s2.end(), s1, s1.begin());
    list<string>::iterator iter1 = s1.begin();
    advance(iter1, 2);
    list<string>::iterator iter2 = s2.begin();
    ++iter2;
    list<string>::iterator iter3 = iter2;
    advance(iter3, 2);
    s1.splice(iter1, s2, iter2, iter3);

    copy(s1.begin(), s1.end(), ostream_iterator<string>(cout, " "));
    cout << endl;
    copy(s2.begin(), s2.end(), ostream_iterator<string>(cout, " "));
    cout << endl;
    return 0;
}
```

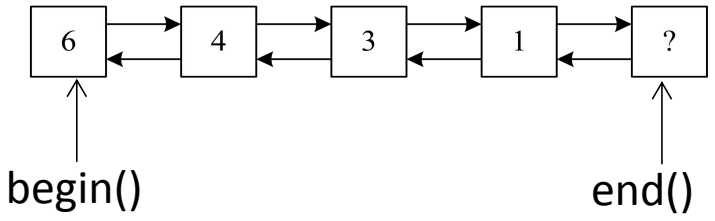
```
#include <list>
#include <iterator>
#include <string>
#include <iostream>
using namespace std;
```

三种顺序容器的比较与选用

- 如果需要执行大量的随机访问操作，而且当扩展容器时只需要向容器尾部加入新的元素，就应当选择**向量容器vector**；
- 如果需要少量的随机访问操作，需要在容器两端插入或删除元素，则应当选择**双端队列容器deque**；、
- 如果不需要对容器进行随机访问，但是需要在中间位置插入或者删除元素，就应当选择**列表容器list**。







顺序容器的插入迭代器

- 插入迭代器
  - 用于向容器头部、尾部或中间指定位置插入元素的迭代器
  - 包括前插迭代器（front_inserter）、后插迭代器（back_insrter）和任意位置插入迭代器（inserter）
- 例：
  - `list<int> s;`
  - `back_inserter iter(s);`
  - `*(iter++) = 5; //通过iter把5插入s末尾`