STL的Deque介绍

2013年09月24日 12:23:25 阅读数：2096

Deque是双端队列，在画动态图表的时候使用起来比较方便。因为当数据填满图表后，需要从队列的后方插入一个元素，然后再从队列的前方删除一个元素。使用Deque可以很方便的做到。使用push(pop)-back(front)就ok了

下面摘录了一个Deque的使用教程

```cpp
/*deque: 是一个double-ended queue,
    1)支持随即存取，也就是[]操作符，
    2)支持两端操作，push(pop)-back(front)，在两端操作上与list效率差不多

    因此在实际使用时，如何选择这三个容器中哪一个，应根据你的需要而定，一般应遵循下面的原则：
    1、如果你需要高效的随即存取，而不在乎插入和删除的效率，使用vector
    2、如果你需要大量的插入和删除，而不关心随即存取，则应使用list
    3、如果你需要随即存取，而且关心两端数据的插入和删除，则应使用deque。
*/

#include <iostream>
#include <deque>
using namespace std;

void printDeque(deque<int> d)
{
//使用下标
//for (unsigned int i = 0; i < d.size(); i++)
//{
// cout<<"d["<<i<<"] = "<<d[i]<<", ";
//}

//使用迭代器
//deque<int>::iterator iter = d.begin();
//for (;iter != d.end(); iter ++)
//{
// cout<<"d["<<iter-d.begin()<<"] = "<<(*iter)<<", ";
//}

//使用迭代器指针
deque<int>::iterator *pIter = new deque<int>::iterator;
if ( NULL == pIter )
{
   return ;
}
for (*pIter = d.begin(); *pIter != d.end(); (*pIter)++)
{
   cout<<"d["<<*pIter - d.begin() <<"]="<<**pIter<<", ";
}
if (NULL != pIter)
{
   delete pIter;
   pIter = NULL;
}

cout<<endl;
}

void main()
{
//创建deque
deque<int> d1; //创建一个没有任何元素的deque对象
deque<int> d2(10);//创建一个具有10个元素的deque对象，每个元素值为默认
deque<double> d3(10, 5.5); //伊妹一个具有10个元素的deque对象,每个元素的初始值为5.5
deque<double> d4(d3); //通过拷贝一个deque对象的元素值，创建一个新的deque对象
int iArray[] = {11, 13, 19, 23, 27};
deque<int> d5(iArray, iArray+5);//将迭代器区间[first, last)所指的元素拷贝到一个新创建的deque对象中

//初始化赋值：同vector一样,使用尾部插入函数push_back()
for (int i = 1; i < 6 ; i++)
   d1.push_back(i*10);
//遍历元素: 1-下标方式 2-迭代器方式 反向遍历（略）
cout<<"printDeque(d1) : "<<endl;
printDeque(d1);

//元素插入：尾部插入用push_back()，头部插入用push_front()，其它位置插入用insert(&pos, elem)
cout<<"d1.push_front(100): "<<endl;
d1.push_front(100);
printDeque(d1);
cout<<"d1.insert(d1.begin()+3, 200): "<<endl; //支持随机存取(即[]操作符)，所以begin()可以+3
d1.insert(d1.begin()+2,200);
printDeque(d1);

//元素删除 尾部删除用pop_back();头部删除用pop_front();
//任意迭代位置或迭代区间上的元素删除用erase(&pos)/erase(&first, &last)；删除所有元素用clear();
cout<<"d1.pop_front(): "<<endl;
```

```
77.  d1.pop_front();
78.  printDeque(d1);
79.
80.  cout<<"d1.erase(d1.begin()+1): "<<endl;
81.  d1.erase(d1.begin()+1); //删除第2个元素d1[1]
82.  printDeque(d1);
83.
84.  cout<<"d1.erase(d1.begin(), d1.begin() + 2) = "<<endl;
85.  d1.erase(d1.begin(), d1.begin() + 2);
86.  printDeque(d1);
87.
88.  cout<<"d1.clear() :"<<endl;
89.  d1.clear();
90.  printDeque(d1);
91.
92.
93.  //其它常用
94.  cout<<"其它常用用法: "<<endl;
95.  int flag = 0;
96.  while(flag < 2)
97.  {
98.     if (0 == flag )
99.     {
100.     for (int i = 1; i < 6 ; i++) //恢复
101.       d1.push_back(i*10);
102.    }
103.    else
104.    {
105.     d1.clear();
106.     cout<<"after d1.clear() , d1.front(), d1.back() is abnormal! other info.:"<<endl;
107.    }
108.    cout<<"d1.empty() = "<<d1.empty()<<endl;
109.    cout<<"d1.size() = "<<d1.size()<<endl;
110.    cout<<"d1.max_size() = "<<hex<<d1.max_size()<<endl;
111.    if (!d1.empty())
112.    {
113.     cout<<"d1.front() = "<<d1.front()<<endl;
114.     cout<<"d1.back() = "<<d1.back()<<endl;
115.    }
116.
117.    flag++;
118.
119. }
120.
121. //交换
122. cout<<"d1.swap(d5)= "<<endl;
123. d1.swap(d5);
124. cout<<"d1 = ";
125. printDeque(d1);
126. cout<<"d5 = ";
127. printDeque(d5);
128. //printDeque(d)
129.
130.
131. }
```


文章标签： stl deque

个人分类： 纯编程