[STL：队列(queue)、优先级队列(priority\_queue )及双向队列(deque)总结](http://blog.csdn.net/dabusideqiang/article/details/38898961)

标签： [STL](http://www.csdn.net/tag/STL)[queue](http://www.csdn.net/tag/queue)[deque](http://www.csdn.net/tag/deque)[priority\_queue](http://www.csdn.net/tag/priority_queue)

2014-08-28 15:38 371人阅读 [评论](http://blog.csdn.net/dabusideqiang/article/details/38898961#comments)(0) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/dabusideqiang/article/details/38898961#report)

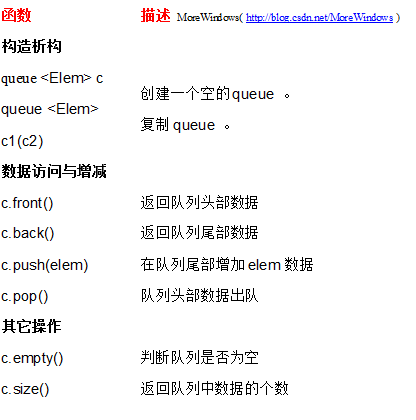
http://static.blog.csdn.net/images/category_icon.jpg 分类：

数据结构及算法（7） http://static.blog.csdn.net/images/arrow_triangle%20_down.jpg

参考博客：<http://blog.csdn.net/column/details/stl-morewindows.html>

一、queue

         queue单向队列与[栈](http://blog.csdn.net/morewindows/article/details/6950881)有点类似，一个是在同一端存取数据，另一个是在一端存入数据，另一端取出数据。单向队列中的数据是先进先出（First In First Out,FIFO）。在STL中，单向队列也是以别的容器作为底部结构，再将接口改变，使之符合单向队列的特性就可以了。因此实现也是非常方便的。下面就给出单向队列的函数列表和使用示例。单向队列一共6个常用函数（front()、back()、push()、pop()、empty()、size()），与[栈](http://blog.csdn.net/morewindows/article/details/6950881)的常用函数较为相似。



     下面给出单向队列的使用示例：

#include <queue>

#include <list>

#include <cstdio>

using namespace std;

int main()

{

//优先级队列默认是使用vector作容器。

priority\_queue<int> a;

int i;

//压入数据

for (i = 0; i < 10; i++)

{

a.push(i \* 2 - 5);

//b.push(i); //编译错误

}

//优先级队列的大小

printf("%d\n", a.size());

//取优先级队列数据并将数据移出队列

while (!a.empty())

{

printf("%d ", a.top());

a.pop();

}

putchar('\n');

return 0;

}

二、priority\_queue

          priority\_queue 优先级队列是一个拥有权值概念的[单向队列queue](http://blog.csdn.net/morewindows/article/details/6950917)，在这个队列中，所有元素是按优先级排列的（也可以认为queue是个按进入队列的先后做为优先级的优先级队列——先进入队列的元素优先权要高于后进入队列的元素）。在计算机操作系统中，优先级队列的使用是相当频繁的，进线程调度都会用到。在STL的具体实现中，priority\_queue也是以别的容器作为底部结构，再根据堆的处理规则来调整元素之间的位置。下面给出priority\_queue的函数列表和使用示例。

|  |  |
| --- | --- |
| priority\_queue函数列表 | |
| 函数 | 描述      by MoreWindows( <http://blog.csdn.net/MoreWindows> ) |
| 构造析构 |  |
| priority\_queue <Elem> c | 创建一个空的queue 。 注：priority\_queue构造函数有7个版本，请查阅MSDN |
| 数据访问与增减 |  |
| c.top() | 返回队列头部数据 |
| c.push(elem) | 在队列尾部增加elem数据 |
| c.pop() | 队列头部数据出队 |
| 其它操作 |  |
| c.empty() | 判断队列是否为空 |
| c.size() | 返回队列中数据的个数 |
|  |  |

可以看出priority\_queue的函数列表与栈[stack](http://blog.csdn.net/morewindows/article/details/6950881)的函数列表是相同的。

下面先给出优级先级队列的使用示例。

//by MoreWindows( http://blog.csdn.net/MoreWindows )

#include <queue>

#include <cstring>

#include <cstdio>

using namespace std;

//结构体

struct Node

{

char szName[20];

int priority;

Node(int nri, char \*pszName)

{

strcpy(szName, pszName);

priority = nri;

}

};

//结构体的比较方法 改写operator()

struct NodeCmp

{

bool operator()(const Node &na, const Node &nb)

{

if (na.priority != nb.priority)

return na.priority <= nb.priority;

else

return strcmp(na.szName, nb.szName) > 0;

}

};

void PrintfNode(Node na)

{

printf("%s %d\n", na.szName, na.priority);

}

int main()

{

//优先级队列默认是使用vector作容器，底层数据结构为堆。

priority\_queue<Node, vector<Node>, NodeCmp> a;

//有5个人进入队列

a.push(Node(5, "小谭"));

a.push(Node(3, "小刘"));

a.push(Node(1, "小涛"));

a.push(Node(5, "小王"));

//队头的2个人出队

PrintfNode(a.top());

a.pop();

PrintfNode(a.top());

a.pop();

printf("--------------------\n");

//再进入3个人

a.push(Node(2, "小白"));

a.push(Node(2, "小强"));

a.push(Node(3, "小新"));

//所有人都依次出队

while (!a.empty())

{

PrintfNode(a.top());

a.pop();

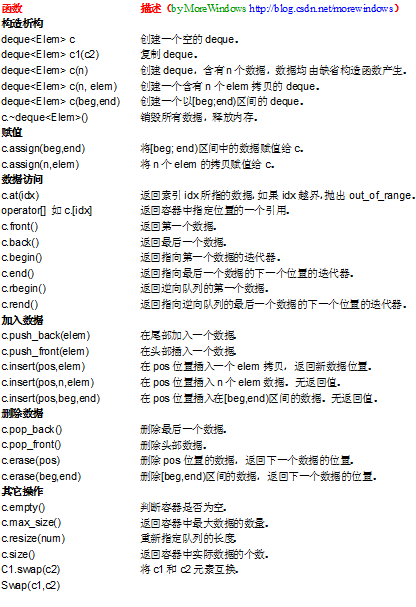
}

return 0;

}

三、deque

           deque双向队列是一种双向开口的连续线性空间，可以高效的在头尾两端插入和删除元素，deque在接口上和vector非常相似，下面列出deque的常用成员函数：



   下面给出deque的使用示例：

//双向队列 deque

//by MoreWindows http://blog.csdn.net/morewindows

#include <deque>

#include <cstdio>

#include <algorithm>

using namespace std;

int main()

{

deque<int> ideq(20); //Create a deque ideq with 20 elements of default value 0

deque<int>::iterator pos;

int i;

//使用assign()赋值 assign在计算机中就是赋值的意思

for (i = 0; i < 20; ++i)

ideq[i] = i;

//输出deque

printf("输出deque中数据:\n");

for (i = 0; i < 20; ++i)

printf("%d ", ideq[i]);

putchar('\n');

//在头尾加入新数据

printf("\n在头尾加入新数据...\n");

ideq.push\_back(100);

ideq.push\_front(i);

//输出deque

printf("\n输出deque中数据:\n");

for (pos = ideq.begin(); pos != ideq.end(); pos++)

printf("%d ", \*pos);

putchar('\n');

//查找

const int FINDNUMBER = 19;

printf("\n查找%d\n", FINDNUMBER);

pos = find(ideq.begin(), ideq.end(), FINDNUMBER);

if (pos != ideq.end())

printf("find %d success\n", \*pos);

else

printf("find failed\n");

//在头尾删除数据

printf("\n在头尾删除数据...\n");

ideq.pop\_back();

ideq.pop\_front();

//输出deque

printf("\n输出deque中数据:\n");

for (pos = ideq.begin(); pos != ideq.end(); pos++)

printf("%d ", \*pos);

putchar('\n');

return 0;

}