Move Zeroes

在c++中，vector是一个十分有用的容器，下面对这个容器做一下总结。

1.基本操作

(1)头文件#include<vector>.

(2)创建vector对象，vector<int> vec;

(3)尾部插入数字：vec.push\_back(a);

(4)使用下标访问元素，cout<<vec[0]<<endl;记住下标是从0开始的。

(5)使用迭代器访问元素.

vector<int>::iterator it;for(it=vec.begin();it!=vec.end();it++)

cout<<\*it<<endl;

(6)插入元素：    vec.insert(vec.begin()+i,a);在第i+1个元素前面插入a;

(7)删除元素：

vec.erase(vec.begin()+2);删除第3个元素

vec.erase(vec.begin()+i,vec.end()+j);删除区间[i,j-1];区间从0开始

(8)向量大小:vec.size();

(9)清空:vec.clear();

2.vector的元素不仅仅可以使int,double,string,还可以是结构体，但是要注意：结构体要定义为全局的，否则会出错。下面是一段简短的程序代码：

[IMG_256](http://www.cnblogs.com/wang7/archive/2012/04/27/javascript:void(0);)

#include<stdio.h>

#include<algorithm>

#include<vector>

#include<iostream>using namespace std;

typedef struct rect

{

int id;

int length;

int width;  
  
　　//对于向量元素是结构体的，可在结构体内部定义比较函数，下面按照id,length,width升序排序。  
　　bool operator< (const rect &a)  const  
    {  
        if(id!=a.id)  
            return id<a.id;  
        else  
        {  
            if(length!=a.length)  
                return length<a.length;  
            else  
                return width<a.width;  
        }  
    }

}Rect;

int main()

{

vector<Rect> vec;

Rect rect;

rect.id=1;

rect.length=2;

rect.width=3;

vec.push\_back(rect);

vector<Rect>::iterator it=vec.begin();

cout<<(\*it).id<<' '<<(\*it).length<<' '<<(\*it).width<<endl;

return 0;

}

[IMG_257](http://www.cnblogs.com/wang7/archive/2012/04/27/javascript:void(0);)

 3.算法

(1) 使用reverse将元素翻转：需要头文件#include<algorithm>

reverse(vec.begin(),vec.end());将元素翻转(在vector中，如果一个函数中需要两个迭代器，

一般后一个都不包含.)

(2)使用sort排序：需要头文件#include<algorithm>，

**sort(vec.begin(),vec.end());**(默认是按升序排列,即从小到大).

可以通过重写排序比较函数按照降序比较，如下：

定义排序比较函数：

bool Comp(const int &a,const int &b)  
{  
    return a>b;  
}  
调用时:sort(vec.begin(),vec.end(),Comp)，这样就降序排序。

vector中的erase方法用法注意事项：

以前就发现了vector中的erase方法有些诡异(^\_^)，稍不注意，就会出错。今天又一次遇到了，就索性总结一下，尤其是在循环体中用erase时，由于vector.begin() 和vector.end()是变化的，因此就引入了错误的可能性。

erase的函数原型有两种形式:

iterator erase(iterator position);

iterator erase(iterator first, iterator last);

vector<int> veci;  
veci.push\_back(1);  
veci.push\_back(2);  
veci.push\_back(3);  
veci.push\_back(4);  
veci.push\_back(5);  
veci.push\_back(3);  
veci.push\_back(2);  
veci.push\_back(3);

for(vector<int>::iterator iter=veci.begin(); iter!=veci.end(); iter++)  
{  
      if( \*iter == 3)  
             veci.erase(iter);  
}

乍一看这段代码，很正常。其实这里面隐藏着一个很严重的错误：当veci.erase(iter)之后，iter就变成了一个野指针，对一个野指针进行 iter++ 是肯定会出错的。

查看MSDN，对于erase的返回值是这样描述的：An iterator that designates the first element remaining beyond any elements removed, or a pointer to the end of the vector if no such element exists，于是改代码：

for(vector<int>::iterator iter=veci.begin(); iter!=veci.end(); iter++)  
{  
      if( \*iter == 3)  
             iter = veci.erase(iter);  
}

这段代码也是错误的：1）无法删除两个连续的"3"； 2）当3位于vector最后位置的时候，也会出错（在veci.end()上执行 ++ 操作）

正确的代码应该为：

for(vector<int>::iterator iter=veci.begin(); iter!=veci.end(); )  
{  
     if( \*iter == 3)  
          iter = veci.erase(iter);  
      else  
            iter ++ ;  
}

为了避免对野指针进行操作，另一种解决方法如下：

vector<int>::iterator itor2;

for(vector<int>::iterator iter=veci.begin(); iter!=veci.end(); )  
{  
     if( \*iter == 3)

     {

           itor2=iter;  
          veci.erase(itor2);  
     }

      else  
            iter ++ ;  
}

要解决无法删除两个连续的3的另一种方法如下：

vector<int>  veci;

veci.erase(remove(veci.begin(),veci.end(),6),veci.end());

这里用到了remove（）函数，

注：remove是个stl的通用算法std::remove(first,last,val)移除[first, last)范围内等于val的元素在vector里面用就类似于iter = std::remove(vec.begin(), vec.end(), val)但这个函数只是把val移到vec的末尾，并不真正删除真正删除还是要调用一次erase函数

综上，该题代码并不难，如下：

void moveZeroes(vector<int>& nums) {

int count=0;

int i;

vector<int>::iterator it;

for(it=nums.begin();it!=nums.end();)

{

if(\*it==0)

{

count++;

it=nums.erase(it);

}

else

it++;

}

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

{

nums.push\_back(0);

}

}

另一种解法,对其位置进行操作：

class Solution {

public:

void moveZeroes(vector<int>& nums) {

if(nums.size()<=0) return;

int p=0;

int i=0;

for(i;i<nums.size();i++)

{

if(nums[i]==0)continue;

nums[p++]=nums[i];

}

for(i=p;i<nums.size();i++)

{

nums[i]=0;

}

}

};