原 简单的程序诠释C++ STL算法系列之十五：swap

相信大家看到swap这个词都一定不会感到陌生，甚至会有这样想法：这不就是简单的元素交换嘛。的确，swap交换函数是仅次于Hello word这样老得不能老的词，然而，泛型算法东风，这个小小的玩意儿却在C++ STL中散发着无穷的魅力。本文不仅详细地阐述STL泛型算法swap，并借助泛型算法这股东风，展现STL容器中swap成员函数的神奇魅力。注意哦，泛型算法swap和容器中的swap成员函数，这是两个不同角度和概念哦！

一、泛型算法swap

老规矩，我们先来看看swap的函数原型：

```
template <class T> void swap ( T& a, T& b )
{
  T c(a); a=b; b=c;
}
```

函数原型超级简单吧，这里我们就不做过多的解释啦，下面我们还是通过一个简单的示例来熟悉熟悉它的使用吧。

程序示例：

```
/*****************************************************************
 * Copyright (C) Jerry Jiang
 *
 * File Name   : swap.cpp
 * Author      : Jerry Jiang
 * Create Time : 2012-3-24 4:19:31
 * Mail        : jbiaojerry@gmail.com
 * Blog        : http://blog.csdn.net/jerryjbiao
 *
 * Description : 简单的程序诠释C++ STL算法系列之十五
 *               变易算法 : 元素交换swap
 *
 *****************************************************************/

#include <iostream>
#include <algorithm>
#include <vector>
#include <iterator>

using namespace std;

int main ()
{

 int x = 10, y = 20;                        // x:10 y:20
 swap(x, y);                               // x:20 y:10

 vector<int> first (4, x), second (6, y);  // first:4x20 second:6x10
 swap(first, second);                      // first:6x10 second:4x20

 cout << "first contains:";
 //使用一般的iterator方式输出first
 for (vector<int>::iterator it=first.begin();  it != first.end(); ++it)
 {
  cout << " " << *it;
 }
 cout << endl;

 cout << "second contains: ";
 //使用copy()来实现second的输出
 copy(second.begin(), second.end(), ostream_iterator<int>(cout, " "));
 cout << endl;

 return 0;
}
```

上面示例程序十分简单，只是为了巩固前文中copy算法的使用，我在程序中采用了两种方式进行输出，好了，泛型算法swap我们就不再废话了，现在我们来看看本文中的重头戏吧。

二、容器中的成员函数swap

在容器vector中，其内存占用的空间是只增不减的，比如说首先分配了10,000个字节，然后erase掉后面9,999个，则虽然有效元素只有一个，但是内存占用仍为10,000个。所有内存空间在vector析构时回收。

一般，我们都会通过vector中成员函数clear进行一些清除操作，但它清除的是所有的元素，使vector的大小减少至0，却不能减小vector占用的内存。要避免vector持有它不再需要的内存，这就需要一种方法来使得它从曾经的容量减少至它现在需要的容量，这样减少容量的方法被称为“收缩到合适（shrink to fit）”。（节选自《Effective STL》）如果做到“收缩到合适”呢，嘿嘿，这就要全仰仗“Swap大侠”啦，即通过如下代码进行释放过剩的容量：

```
vector< T >(X).swap(X)    //shrink to fit 收缩到合适
```

```
vector< T >().swap(X)    // 清除X并最小化它的容量，可以理解为交换技巧的变体
```

下面我们通过一个简单的示例来show一下：

```
/*****************************************************************
 * Copyright (C) Jerry Jiang
 *
 * File Name   : swap.cpp
 * Author      : Jerry Jiang
 * Create Time : 2012-3-24 4:19:31
 * Mail        : jbiaojerry@gmail.com
 * Blog        : http://blog.csdn.net/jerryjbiao
 *
 * Description : 简单的程序诠释C++ STL算法系列之十五
 *               成员函数swap实现容器的内存释放
 *
 *****************************************************************/

#include <iostream>
#include <algorithm>
#include <vector>
#include <iterator>

using namespace std;

int main ()
{
 int x = 10;
 vector<int> myvector(10000, x);

 //这里打印仅仅是元素的个数和容量
 cout << "myvector size:"
   << myvector.size() << "," << myvector.capacity()
   << endl;

 //swap交换函数释放内存：vector<T>().swap(X);
 //T:int ; myvertor代表X
 vector<int>().swap(myvector);  // 这里是清除myvector并最小化它的大小

 //两个输出仅用来表示swap前后的变化
 cout << "after swap :"
   << myvector.size() << ", " << myvector.capacity()
   << endl;

 return 0;
}
```

swap交换技巧实现内存释放思想：vector()使用vector的默认构造函数建立临时vector对象，再在该临时对象上调用swap成员，swap调用之后对象myvector占用的空间就等于一个默认构造的对象的大小，临时对象就具有原来对象v的大小，而该临时对象随即就会被析构，从而其占用的空间也被释放。

```
std::vector<T>(X).swap(X)

作用相当于：

{

std::vector<T>  temp(X);

temp.swap(X);

}
```

注意：并不是所有的STL容器的clear成员函数的行为都和vector一样。事实上，其他容器的clear成员函数都会释放其内存。比如另一个和vector类似的顺序容器deque。

****

****

更新申明：

这篇博文是我六年前在阅读《提高c++性能的编程技术》和《Effective STL》两本书籍的时候做的一些笔记，文章只代表个人当时的理解，并不一定都是完全正确的，由于工作原因后面几年基本停止更新了，但最近常用邮箱中收到了一些读者的人身攻击性和辱骂的邮件，本人十分反感这类伪技术喷子，希望大家本着技术人的精神，辩证地阅读和理解网络上的技术文章，更不要做伪技术喷子，多谢！