原 简单的程序诠释C++ STL算法系列之十八：transform

前篇我们已经了解了一种区间元素交换swap_ranges函数，现在我们再来学习另外一种区间元素交换transform。该算法用于实行容器元素的变换操作。有如下两个使用原型，一个将迭代器区间[first，last)中元素，执行一元函数对象op操作，交换后的结果放在[result，result+（last-first)）区间中。另一个将迭代器区间[first1，last1)的元素*i，依次与[first2，first2+（last-first）)的元素*j，执行二元函数操作binary_op(*i,*j)，交换结果放在[result，result+（last1-first1)）。

函数原型：

```
template < class InputIterator, class OutputIterator, class UnaryOperator >
  OutputIterator transform ( InputIterator first1, InputIterator last1,
                             OutputIterator result, UnaryOperator op );

template < class InputIterator1, class InputIterator2,
           class OutputIterator, class BinaryOperator >
  OutputIterator transform ( InputIterator1 first1, InputIterator1 last1,
                             InputIterator2 first2, OutputIterator result,
                             BinaryOperator binary_op );
```

参数说明：

first1, last1
指出要进行元素变换的第一个迭代器区间 [first1,last1)。
first2
指出要进行元素变换的第二个迭代器区间的首个元素的迭代器位置，该区间的元素个数和第一个区间相等。

result
指出变换后的结果存放的迭代器区间的首个元素的迭代器位置
op
用一元函数对象op作为参数，执行其后返回一个结果值。它可以是一个函数或对象内的类重载operator()。
binary_op
用二元函数对象binary_op作为参数，执行其后返回一个结果值。它可以是一个函数或对象内的类重载operator()。

程序示例：

```
/*****************************************************************
 * Copyright (C) Jerry Jiang
 *
 * File Name   : transform .cpp
 * Author      : Jerry Jiang
 * Create Time : 2012-4-29 22:22:18
 * Mail        : jbiaojerry@gmail.com
 * Blog        : http://blog.csdn.net/jerryjbiao
 *
 * Description : 简单的程序诠释C++ STL算法系列之十八
 *               变易算法 : 区间元素交换 transform
 *
 *****************************************************************/
#include <iostream>
#include <algorithm>
#include <vector>
using namespace std;

int op_increase (int i) { return ++i; }
int op_sum (int i, int j) { return i+j; }

int main () {
  vector<int> first;
  vector<int> second;
  vector<int>::iterator it;

  // set some values:
  for (int i=1; i<6; i++) first.push_back (i*10); //  first: 10 20 30 40 50

  second.resize(first.size());     // allocate space
  transform (first.begin(), first.end(), second.begin(), op_increase);
                                                 // second: 11 21 31 41 51

  transform (first.begin(), first.end(), second.begin(), first.begin(), op_sum);
                                                 //  first: 21 41 61 81 101

  cout << "first contains:";
  for (it=first.begin(); it!=first.end(); ++it)
    cout << " " << *it;

  cout << endl;
  return 0;
}
```

****

****