简单的程序诠释C++ STL算法系列之四：adjacent_find

C++STL的非变易算法（Non-mutating algorithms）是一组不破坏操作数据的模板函数，用来对序列数据进行逐个处理、元素查找、子序列搜索、统计和匹配。

adjacent_find算法用于查找相等或满足条件的邻近元素对。其有两种函数原型：一种在迭代器区间[first , last)上查找两个连续的元素相等时，返回元素对中第一个元素的迭代器位置。另一种是使用二元谓词判断binary_pred，查找迭代器区间[first , last)上满足binary_pred条件的邻近元素对，未找到则返回last。

函数原型：

```
template<class ForwardIterator>
   ForwardIterator adjacent_find(
      ForwardIterator _First,
      ForwardIterator _Last
      );
template<class ForwardIterator , class BinaryPredicate>
   ForwardIterator adjacent_find(
      ForwardIterator _First,
      ForwardIterator _Last,
            BinaryPredicate _Comp
   );
```

示例代码：

```
/*****************************************************************
 * Copyright (C) Jerry Jiang

 * File Name   : adjacent_find.cpp
 * Author      : Jerry Jiang
 * Create Time : 2011-9-30 22:07:22
 * Mail        : jbiaojerry@gmail.com
 * Blog        : http://blog.csdn.net/jerryjbiao
 * Description :  简单的程序诠释C++ STL算法系列之四
 *                非变易算法 : 邻近查找容器元素adjacent_find
 *****************************************************************/

#include <algorithm>
#include <list>
#include <iostream>

using namespace std;

//判断X和y是否奇偶同性
bool parity_equal(int x, int y)
{
 return (x - y) % 2 == 0 ? 1 : 0;
}

int main()
{
 //初始化链表
 list<int> iList;
 iList.push_back(3);
 iList.push_back(6);
 iList.push_back(9);
 iList.push_back(11);
 iList.push_back(11);
 iList.push_back(18);
 iList.push_back(20);
 iList.push_back(20);

 //输出链表
 list<int>::iterator iter;
 for(iter = iList.begin(); iter != iList.end(); ++iter)
 {
  cout << *iter << "  ";
 }
 cout << endl;

 //查找邻接相等的元素
 list<int>::iterator iResult = adjacent_find(iList.begin(), iList.end());
 if (iResult != iList.end())
 {
  cout << "链表中第一对相等的邻近元素为: " << endl;
  cout << *iResult++ << endl;
  cout << *iResult << endl;
 }

 //查找奇偶性相同的邻近元素
 iResult = adjacent_find(iList.begin(), iList.end(), parity_equal);
 if (iResult != iList.end())
 {
  cout << "链表中第一对奇偶相同的元素为: " << endl;
  cout << *iResult++ << endl;
  cout << *iResult << endl;
 }
 return 0;
}
```

****

****