简单的程序诠释C++ STL算法系列之十一：search_n

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。

重复元素子序列搜索search_n算法：搜索序列中是否有一系列元素值均为某个给定值的子序列，它有如下两个函数原型，分别在迭代器区间[first, last)上搜索是否有count个连续元素，其值均等于value（或者满足谓词判断binary_pred的条件），返回子序列首元素的迭代器，或last以表示没有重复元素的子序列。

函数原型：

```
template<class ForwardIterator1, class Diff2, class Type>
   ForwardIterator1 search_n(
      ForwardIterator1 _First1,
      ForwardIterator1 _Last1,
      Size2 _Count,
      const Type& _Val
   );
template<class ForwardIterator1, class Size2, class Type, class BinaryPredicate>
   ForwardIterator1 search_n(
      ForwardIterator1 _First1,
      ForwardIterator1 _Last1,
      Size2 _Count,
      const Type& _Val,
      BinaryPredicate _Comp
   );
```

示例程序：

搜索向量容器ivect = {1,8,8,8,4,,4,3}中有三个连续元素为8，没有四个连续元素8，以及有三个连续元素的两倍为16.

```
/*****************************************************************
 * Copyright (C) Jerry Jiang
 * File Name   : search_n.cpp
 * Author      : Jerry Jiang
 * Create Time : 2011-10-11 22:23:47
 * Mail        : jbiaojerry@gmail.com
 * Blog        : http://blog.csdn.net/jerryjbiao
 * Description :  简单的程序诠释C++ STL算法系列之十一
 *                非变易算法 : 重复元素子序列搜索search_n
 *****************************************************************/

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

bool twice(const int para1, const int para2)
{
 return 2 * para1 == para2;
}

using namespace std;

int main()
{
 vector<int> ivect;
 ivect.push_back(1);
 ivect.push_back(8);
 ivect.push_back(8);
 ivect.push_back(8);
 ivect.push_back(4);
 ivect.push_back(4);
 ivect.push_back(3);

 vector<int>::iterator iLocation;
 iLocation = search_n(ivect.begin(), ivect.end(), 3, 8);
 if (iLocation != ivect.end())
 {
  cout << "在ivect中找到3个连续的元素8" << endl;
 }
 else
 {
  cout << "在ivect中没有3个连续的元素8" << endl;
 }

 iLocation = search_n(ivect.begin(), ivect.end(), 4, 8);
 if (iLocation != ivect.end())
 {
  cout << "在ivect中找到4个连续的元素8" << endl;
 }
 else
 {
  cout << "在ivect中没有4个连续的元素8" << endl;
 }

 iLocation = search_n(ivect.begin(), ivect.end(), 2, 4);
 if (iLocation != ivect.end())
 {
  cout << "在ivect中找到2个连续的元素4" << endl;
 }
 else
 {
  cout << "在ivect中没有2个连续的元素4" << endl;
 }

 iLocation = search_n(ivect.begin(), ivect.end(), 3, 16, twice);
 if (iLocation != ivect.end())
 {
  cout << "在ivect中找到3个连续元素的两倍为16" << endl;
 }
 else
 {
  cout << "在ivect中没有3个连续元素的两倍为16" << endl;
 }
 return 0;
}
```