**非变动性算法**

**一. 概述**  
非变动性算法既不改变元素次序，也不改变元素值。

|  |  |
| --- | --- |
| count() | 返回元素个数 |
| count()\_if() | 返回满足某一条件的元素个数 |
| min\_element() | 返回最小值 |
| max\_element() | 返回最大值 |
| find() | 搜索等于某个值的第一个元素 |
| find\_if() | 搜索满足某个条件的第一个元素 |
| search\_n() | 搜索具有某特性的第一段n个连续元素 |
| search() | 搜索某个子区间第一次出现位置 |
| find\_end() | 搜索某个子区间最后一次出现位置 |
| find\_first\_of() | 搜索等于"某数个值之一"的第一元素 |
| adjacent\_find() | 搜索连续两个相等的元素 |
| equal() | 判断两区间是否相等 |
| mismatch() | 返回两个序列的各组对应元素中，第一对不相等元素 |
| lexicographical\_compare() | 判断某一序列在"字典顺序"下是否小于另一序列 |

**二. 元素计数**

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/lwbeyond/article/details/7325702)

1. //计算区间[beg, end)中元素值等于value的元素个数
2. difference \_type
3. count (InputIterator beg, InputIterator end, const T& value)
4. //计算区间[beg, end)中令op(elem)元素判断结果为true的元素个数
5. difference \_type
6. count\_if (InputIterator beg, InputIterator end, UnaryPredicate op)

//计算区间[beg, end)中元素值等于value的元素个数

difference \_type

count (InputIterator beg, InputIterator end, const T& value)

//计算区间[beg, end)中令op(elem)元素判断结果为true的元素个数

difference \_type

count\_if (InputIterator beg, InputIterator end, UnaryPredicate op)

**三. 最小值和最大值**

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/lwbeyond/article/details/7325702)

1. //返回区间[beg, end)中最小值的位置,注意返回的是位置，你要通过 operator\* 来得到值
2. InputIterator
3. min\_element (InputIterator beg, InputIterator end)
4. //Op用来比较两个元素，op(elem1, elem2)
5. InputIterator
6. min\_element (InputIterator beg, InputIterator end, CompFunc op)
7. //返回区间[beg, end)中最大值的位置
8. InputIterator
9. max\_element (InputIterator beg, InputIterator end)
10. InputIterator
11. max\_element (InputIterator beg, InputIterator end, CompFunc op)

//返回区间[beg, end)中最小值的位置,注意返回的是位置，你要通过 operator\* 来得到值

InputIterator

min\_element (InputIterator beg, InputIterator end)

//Op用来比较两个元素，op(elem1, elem2)

InputIterator

min\_element (InputIterator beg, InputIterator end, CompFunc op)

//返回区间[beg, end)中最大值的位置

InputIterator

max\_element (InputIterator beg, InputIterator end)

InputIterator

max\_element (InputIterator beg, InputIterator end, CompFunc op)

**四. 搜索元素  
1. 搜索第一个元素**

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/lwbeyond/article/details/7325702)

1. //返回区间[beg, end)中第一个元素值等于value的元素位置，如果没有找到，两种形式都返回end
2. InputIterator
3. find (InputIterator beg, InputIterator end, const T& value)
4. InputIterator
5. find\_if (InputIterator beg, InputIterator end, UnaryPredicate op)

//返回区间[beg, end)中第一个元素值等于value的元素位置，如果没有找到，两种形式都返回end

InputIterator

find (InputIterator beg, InputIterator end, const T& value)

InputIterator

find\_if (InputIterator beg, InputIterator end, UnaryPredicate op)

**2. 搜索前n个连续匹配值**

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/lwbeyond/article/details/7325702)

1. //返回区间[beg, end)中第一组连续count个元素值全等于value的元素位置
2. //注意如果没有找到匹配的元素或count个数不满足，都会返回end
3. InputIterator
4. search\_n (InputIterator beg, InputIterator end,
5. Size count, const T& value)
6. //返回区间[beg, end)中第一组连续count个元素造成op(elem, value)为true的元素位置
7. InputIterator
8. search\_n (InputIterator beg, InputIterator end,
9. Size count, const T& value, BinaryPredicate op)

//返回区间[beg, end)中第一组连续count个元素值全等于value的元素位置

//注意如果没有找到匹配的元素或count个数不满足，都会返回end

InputIterator

search\_n (InputIterator beg, InputIterator end,

Size count, const T& value)

//返回区间[beg, end)中第一组连续count个元素造成op(elem, value)为true的元素位置

InputIterator

search\_n (InputIterator beg, InputIterator end,

Size count, const T& value, BinaryPredicate op)

**3. 搜索第一个子区间**

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/lwbeyond/article/details/7325702)

1. //返回区间[beg, end)内和区间[searchBeg, searchEnd)完全吻合的第一个子区间内的第一个元素位置
2. //如果没找到返回 end
3. ForwardIterator1
4. search (ForwardIterator1 beg, ForwardIterator1 end,
5. ForwardIterator2 searchBeg, ForwardIterator2 searchEnd)
6. //子区间的元素和 [searchBeg, searchEnd)内的元素满足op(elem, searchElem)为true
7. ForwardIterator1
8. search (ForwardIterator1 beg, ForwardIterator1 end,
9. ForwardIterator2 searchBeg, ForwardIterator2 searchEnd,
10. BinaryPredicate op)

//返回区间[beg, end)内和区间[searchBeg, searchEnd)完全吻合的第一个子区间内的第一个元素位置

//如果没找到返回 end

ForwardIterator1

search (ForwardIterator1 beg, ForwardIterator1 end,

ForwardIterator2 searchBeg, ForwardIterator2 searchEnd)

//子区间的元素和 [searchBeg, searchEnd)内的元素满足op(elem, searchElem)为true

ForwardIterator1

search (ForwardIterator1 beg, ForwardIterator1 end,

ForwardIterator2 searchBeg, ForwardIterator2 searchEnd,

BinaryPredicate op)

**4. 搜索最后一个子区间**

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/lwbeyond/article/details/7325702)

1. //返回区间[beg, end)内和区间[searchBeg, searchEnd)完全吻合的最后一个子区间内的第一个元素位置
2. //如果没找到返回 end
3. ForwardIterator
4. find\_end (ForwardIterator beg, ForwardIterator end,
5. ForwardIterator searchBeg, ForwardIterator searchEnd)
6. //子区间的元素和 [searchBeg, searchEnd)内的元素满足op(elem, searchElem)为true
7. ForwardIterator
8. find\_end (ForwardIterator beg, ForwardIterator end,
9. ForwardIterator searchBeg, ForwardIterator searchEnd,
10. BinaryPredicate op)

//返回区间[beg, end)内和区间[searchBeg, searchEnd)完全吻合的最后一个子区间内的第一个元素位置

//如果没找到返回 end

ForwardIterator

find\_end (ForwardIterator beg, ForwardIterator end,

ForwardIterator searchBeg, ForwardIterator searchEnd)

//子区间的元素和 [searchBeg, searchEnd)内的元素满足op(elem, searchElem)为true

ForwardIterator

find\_end (ForwardIterator beg, ForwardIterator end,

ForwardIterator searchBeg, ForwardIterator searchEnd,

BinaryPredicate op)

**5. 搜索某些元素的第一次出现地点**

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/lwbeyond/article/details/7325702)

1. //返回第一个既在区间[beg, end)出现，也在区间[searchBeg, searchEnd)中出现的元素位置
2. //如果没找到返回 end
3. ForwardIterator
4. find\_first\_of (ForwardIterator1 beg, ForwardIterator1 end,
5. ForwardIterator2 searchBeg, ForwardIterator2 searchEnd)
6. //返回第一个在区间[beg, end)和区间[searchBeg, searchEnd)
7. //中每个元素在op(elem, searchElem)为true的位置
8. ForwardIterator
9. find\_first\_of (ForwardIterator1 beg, ForwardIterator1 end,
10. ForwardIterator2 searchBeg, ForwardIterator2 searchEnd,
11. BinaryPredicate op)

//返回第一个既在区间[beg, end)出现，也在区间[searchBeg, searchEnd)中出现的元素位置

//如果没找到返回 end

ForwardIterator

find\_first\_of (ForwardIterator1 beg, ForwardIterator1 end,

ForwardIterator2 searchBeg, ForwardIterator2 searchEnd)

//返回第一个在区间[beg, end)和区间[searchBeg, searchEnd)

//中每个元素在op(elem, searchElem)为true的位置

ForwardIterator

find\_first\_of (ForwardIterator1 beg, ForwardIterator1 end,

ForwardIterator2 searchBeg, ForwardIterator2 searchEnd,

BinaryPredicate op)

**6. 搜索两个连续且相等的元素**

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/lwbeyond/article/details/7325702)

1. //返回区间[beg, end)中第一对"连续两个相等元素"中的第一元素的位置
2. InputIterator
3. adjacent\_find (InputIterator beg, InputIterator end)
4. //返回区间[beg, end)中第一对"连续的两个元素使op(elem, nextElem)为true"的第一元素的位置
5. InputIteratoradjacent\_find\_if (InputIterator beg, InputIterator end,
6. BinaryPredicate op)

//返回区间[beg, end)中第一对"连续两个相等元素"中的第一元素的位置

InputIterator

adjacent\_find (InputIterator beg, InputIterator end)

//返回区间[beg, end)中第一对"连续的两个元素使op(elem, nextElem)为true"的第一元素的位置

InputIteratoradjacent\_find\_if (InputIterator beg, InputIterator end,

BinaryPredicate op)

**五. 区间比较  
1. 检验相等性**

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/lwbeyond/article/details/7325702)

1. //判断区间[beg, end)内的元素是否都和以cmpBeg开头的区间内的元素相等
2. **bool**
3. equal (InputIterator1 beg, InputIterator1 end,
4. InputIterator2 cmpBeg)
5. //判断区间[beg, end)内的元素和以cmpBeg开头的区间内对应的元素，
6. //是否都能够使op(elem, cmpElem)为true
7. **bool**
8. equal (InputIterator1 beg, InputIterator1 end,
9. InputIterator2 cmpBeg, BinaryPredicate op)

//判断区间[beg, end)内的元素是否都和以cmpBeg开头的区间内的元素相等

bool

equal (InputIterator1 beg, InputIterator1 end,

InputIterator2 cmpBeg)

//判断区间[beg, end)内的元素和以cmpBeg开头的区间内对应的元素，

//是否都能够使op(elem, cmpElem)为true

bool

equal (InputIterator1 beg, InputIterator1 end,

InputIterator2 cmpBeg, BinaryPredicate op)

**2. 搜索第一处不同点**

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/lwbeyond/article/details/7325702)

1. //返回区间[beg, end)和以cmpBeg开头的区间中第一组两两相异的对应元素
2. pair<InputIterator1,InputIterator2>
3. mismatch (InputIterator1 beg, InputIterator1 end,
4. InputIterator2 cmpBeg)
5. //返回区间[beg, end)和以cmpBeg开头的区间中第一组使op(elem,cmpElem)为false的元素
6. pair<InputIterator1,InputIterator2>
7. mismatch (InputIterator1 beg, InputIterator1 end,
8. InputIterator2 cmpBeg, BinaryPredicate op)

//返回区间[beg, end)和以cmpBeg开头的区间中第一组两两相异的对应元素

pair<InputIterator1,InputIterator2>

mismatch (InputIterator1 beg, InputIterator1 end,

InputIterator2 cmpBeg)

//返回区间[beg, end)和以cmpBeg开头的区间中第一组使op(elem,cmpElem)为false的元素

pair<InputIterator1,InputIterator2>

mismatch (InputIterator1 beg, InputIterator1 end,

InputIterator2 cmpBeg, BinaryPredicate op)

**3. 检验"小于"**

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/lwbeyond/article/details/7325702)

1. //判断区间[beg1,end1)的元素是否小于区间[beg2,end2)的元素
2. **bool**
3. lexicographical\_compare (InputIterator1 beg1, Input Iterator1 end1,
4. InputIterator2 beg2, InputIterator2 end2)
5. //以op(elem1, elem2)比较元素。如果elem1 小于 elem2则返回true
6. **bool**
7. lexicographical\_compare (InputIterator1 begl, InputIterator1 end1,
8. InputIterator2 beg2, InputIterator2 end2,
9. CompFunc op)