**一. 概述**

用来处理数值的算法，需要加上头文件 **#include<numberic>**

|  |  |
| --- | --- |
| accumulate() | 组合所有元素(求总和，求乘积...) |
| inner\_product() | 组合两区间内的所有元素 |
| adjacent\_difference() | 将每个元素和其前一元素组合 |
| partial\_sum() | 将每个元素和其先前的所有元素组合 |

**二. 加工运算后产生结果**

**1. 对序列进行某种运算**

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/lwbeyond/article/details/7332197)

1. //计算initValue和区间[beg, end)内所有元素的总和
2. //也就是： initValue + a1 + a2 + a3 +...
3. T
4. accumulate (InputIterator beg, InputIterator end,
5. T initValue)
6. //也就是：initValue op a1 op a2 ...
7. //比如 op 是 乘，则为：initValue \* a1 \* a2 \* a3...
8. T
9. accumulate (InputIterator beg. InputIterator end,
10. T initValue, BinaryFunc op)

//计算initValue和区间[beg, end)内所有元素的总和

//也就是： initValue + a1 + a2 + a3 +...

T

accumulate (InputIterator beg, InputIterator end,

T initValue)

//也就是：initValue op a1 op a2 ...

//比如 op 是 乘，则为：initValue \* a1 \* a2 \* a3...

T

accumulate (InputIterator beg. InputIterator end,

T initValue, BinaryFunc op)

**2. 计算两序列内积**

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/lwbeyond/article/details/7332197)

1. //返回[beg, end)区间和beg2为起始的区间的对应元素组的内积
2. //initValue + (a1\*b1) + (a2\*b2) + (a3\*b3) + ...
3. T
4. inner\_product (InputIterator1 beg1, InputIterator1 end1,
5. InputIterator2 beg2,
6. T initValue)
7. //initValue op1 (a1 op2 b1) op1 (a2 op2 b2) op1 ...
8. T
9. inner\_product (InputIterator1 beg1. InputIterator1 end1,
10. InputIterator2 beg2,
11. T initValue,
12. BinaryFunc op1, BinaryFunc op2)

//返回[beg, end)区间和beg2为起始的区间的对应元素组的内积

//initValue + (a1\*b1) + (a2\*b2) + (a3\*b3) + ...

T

inner\_product (InputIterator1 beg1, InputIterator1 end1,

InputIterator2 beg2,

T initValue)

//initValue op1 (a1 op2 b1) op1 (a2 op2 b2) op1 ...

T

inner\_product (InputIterator1 beg1. InputIterator1 end1,

InputIterator2 beg2,

T initValue,

BinaryFunc op1, BinaryFunc op2)

**三. 相对值和绝对值之间的转换**

**1. 将相对值转换成绝对值**

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/lwbeyond/article/details/7332197)

1. //计算区间[sourceBeg, sourceEnd)中每个元素的部分和,然后写入destBeg
2. //a1, a1+a2, a1+a2+a3, ...
3. OutputIterator
4. partial\_sum (InputIterator sourceBeg, InputIterator sourceEnd,
5. OutputIterator destBeg)
6. //a1, a1 op a2, a1 op a2 op a3, ...
7. OutputIterator
8. partial\_sum (InputIterator sourceBeg, InputIterator sourceEnd,
9. OutputIterator destBeg, BinaryFunc op)

//计算区间[sourceBeg, sourceEnd)中每个元素的部分和,然后写入destBeg

//a1, a1+a2, a1+a2+a3, ...

OutputIterator

partial\_sum (InputIterator sourceBeg, InputIterator sourceEnd,

OutputIterator destBeg)

//a1, a1 op a2, a1 op a2 op a3, ...

OutputIterator

partial\_sum (InputIterator sourceBeg, InputIterator sourceEnd,

OutputIterator destBeg, BinaryFunc op)

**2. 将绝对值转换成相对值**

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/lwbeyond/article/details/7332197)

1. //计算区间[sourceBeg, sourceEnd)中每一个元素前趋差,然后写destBeg
2. //a1, a2-a1, a3-a2, a4-a3, ...
3. OutputIterator
4. adjacent\_difference (InputIterator sourceBeg,
5. InputIterator sourceEnd,
6. OutputIterator destBeg)
7. //a1, a2 op a1, a3 op a2, a4 op a3, ...
8. OutputIterator
9. adjacent\_difference (InputIterator sourceBeg,
10. InputIterator sourceEnd,
11. OutputIterator destBeg,
12. BinaryFunc op)