**一. for\_each()算法**

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/lwbeyond/article/details/7325261)

1. for\_each (InputIterator beg, InputIterator end, UnaryProc op)

for\_each (InputIterator beg, InputIterator end, UnaryProc op)

1. 对区间[beg, end)中的每个元素elem调用，op(elem)。

2. op的任何返回值都会被忽略。

3. for\_each()返回op仿函数的一个副本。

4. 调用op的次数为[beg, end)的个数次。

**二. for\_each()例子**

**1. 一般函数作为for\_each()的参数**

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/lwbeyond/article/details/7325261)

1. #include "algostuff.hpp"
2. using namespace std;
3. // function called for each element
4. void print (**int** elem)
5. {
6. cout << elem << ' ';
7. }
8. **int** main()
9. {
10. vector<**int**> coll;
11. INSERT\_ELEMENTS(coll,1,9);
12. // for\_each 对每个 elem 将调用 print(elem)
13. for\_each (coll.begin(), coll.end(), // range
14. print); // operation
15. cout << endl;
16. }

#include "algostuff.hpp"

using namespace std;

// function called for each element

void print (int elem)

{

cout << elem << ' ';

}

int main()

{

vector<int> coll;

INSERT\_ELEMENTS(coll,1,9);

// for\_each 对每个 elem 将调用 print(elem)

for\_each (coll.begin(), coll.end(), // range

print); // operation

cout << endl;

}

**2. 仿函数作为for\_each()的参数**

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/lwbeyond/article/details/7325261)

1. #include "algostuff.hpp"
2. using namespace std;
3. // function object that adds the value with which it is initialized
4. template <c1ass T>
5. c1ass AddValue
6. {
7. private:
8. T theValue; // value to add
9. public:
10. // 构造函数，初始化theValue
11. AddValue (const T& v) : theValue(v) {
12. }
13. // the function call for the element adds the value
14. void operator() (T& elem) const { //不好理解？看成 void Fun(int& elem) const {},好理解了吧！！
15. elem += theValue;
16. }
17. };
18. **int** main()
19. {
20. vector<**int**> coll;
21. INSERT\_ELEMENTS(coll,1,9);
22. // 每个 element 加 10
23. for\_each (coll.begin(), coll.end(), // range
24. AddValue<**int**>(10)); // 这个怎么理解？？？？
25. //首先 AddValue(10)是一个构造函数，构造完后theValue＝10，然后再调用operator(elem)！！
26. PRINT\_ELEMENTS(coll);
27. // 每个 element 加 第一个值
28. for\_each (coll.begin(), coll.end(), // range
29. AddValue<**int**>(\*coll.begin())); // operation
30. PRINT\_ELEMENTS(coll);
31. }

#include "algostuff.hpp"

using namespace std;

// function object that adds the value with which it is initialized

template <c1ass T>

c1ass AddValue

{

private:

T theValue; // value to add

public:

// 构造函数，初始化theValue

AddValue (const T& v) : theValue(v) {

}

// the function call for the element adds the value

void operator() (T& elem) const { //不好理解？看成 void Fun(int& elem) const {},好理解了吧！！

elem += theValue;

}

};

int main()

{

vector<int> coll;

INSERT\_ELEMENTS(coll,1,9);

// 每个 element 加 10

for\_each (coll.begin(), coll.end(), // range

AddValue<int>(10)); // 这个怎么理解？？？？

//首先 AddValue(10)是一个构造函数，构造完后theValue＝10，然后再调用operator(elem)！！

PRINT\_ELEMENTS(coll);

// 每个 element 加 第一个值

for\_each (coll.begin(), coll.end(), // range

AddValue<int>(\*coll.begin())); // operation

PRINT\_ELEMENTS(coll);

}

**3. 利用for\_each()的返回值**

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/lwbeyond/article/details/7325261)

1. #include "algostuff.hpp"
2. using namespace std;
3. //仿函数，用来处理平均值
4. c1ass MeanValue
5. {
6. private:
7. **long** num; // number of elements
8. **long** sum; // sum of all element values
9. public:
10. // 构造函数
11. MeanValue () : num(0), sum(0) {
12. }
13. // 调用
14. // - process one more element of the sequence
15. void operator() (**int** elem) {
16. num++; // increment count
17. sum += elem; // add value
18. }
19. // 重载double类型运算符，没什么道理，只是返回平均值
20. operator **double**() {
21. return static\_cast<**double**>(sum) / static\_cast<**double**>(num);
22. }
23. };
24. **int** main()
25. {
26. vector<**int**> coll;
27. INSERT\_ELEMENTS(coll,1,8);
28. // 返回应该是仿函数，但是被转换成了double，这个是隐式转换
29. **double** mv = for\_each (coll.begin(), coll.end(), // range
30. MeanValue()); // operation
31. cout << "mean value: " << mv << endl;
32. }