**一. 概念：**

函数对象function objects，又名仿函数，是一个定义了operator()操作的对象。

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/lwbeyond/article/details/7323890)

1. class FunctionObjectType {
2. public:
3. void operator() {
4. statements
5. }
6. };

class FunctionObjectType {

public:

void operator() {

statements

}

};

**二. 仿函数当做排序准则：**

程序员经常需要将某些class object以已序的形式放到容器中，然而有时你无法使用一般的operator<来对这些对象排序，这时可以仿函数。

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/lwbeyond/article/details/7323890)

1. #include <iostream>
2. #include <string>
3. #include <set>
4. #include <algorithm>
5. using namespace std;
6. class Person {
7. public:
8. string firstname() const;
9. string lastname() const;
10. ...
11. };
12. /\* 仿函数，按名字排序
13. \* - operator() returns whether a person is less than another person
14. \*/
15. class PersonSortCriterion {
16. public:
17. **bool** operator() (const Person& p1, const Person& p2) const {
18. /\* a person is less than another person
19. \* - if the last name is less
20. \* - if the last name is equal and the first name is less
21. \*/
22. return p1.lastname()<p2.1astname() ||
23. (! (p2.1astname()<p1.lastname()) &&
24. p1.firstname()<p2.firstname());
25. }
26. };
27. **int** main()
28. {
29. //声名 set ，用这个仿函数
30. typedef set<Person,PersonSortCriterion> PersonSet;
31. //create such a collection
32. PersonSet coll;
33. ...
34. //do something with the elements
35. PersonSet::iterator pos;
36. for (pos = coll.begin(); pos != coll.end();++pos) {
37. ...
38. }
39. ...
40. }

#include <iostream>

#include <string>

#include <set>

#include <algorithm>

using namespace std;

class Person {

public:

string firstname() const;

string lastname() const;

...

};

/\* 仿函数，按名字排序

\* - operator() returns whether a person is less than another person

\*/

class PersonSortCriterion {

public:

bool operator() (const Person& p1, const Person& p2) const {

/\* a person is less than another person

\* - if the last name is less

\* - if the last name is equal and the first name is less

\*/

return p1.lastname()<p2.1astname() ||

(! (p2.1astname()<p1.lastname()) &&

p1.firstname()<p2.firstname());

}

};

int main()

{

//声名 set ，用这个仿函数

typedef set<Person,PersonSortCriterion> PersonSet;

//create such a collection

PersonSet coll;

...

//do something with the elements

PersonSet::iterator pos;

for (pos = coll.begin(); pos != coll.end();++pos) {

...

}

...

}

**三. 拥有内部状态的仿函数：**

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/lwbeyond/article/details/7323890)

1. #include <iostream>
2. #include <list>
3. #include <algorithm>
4. #include "print.hpp"
5. using namespace std;
6. class IntSequence {
7. private:
8. **int** value;
9. public:
10. //constructor
11. IntSequence (**int** initialValue)
12. : value(initialValue) {
13. }
14. //''function call''
15. **int** operator() () {
16. return value++;
17. }
18. };<span><span class="comment"></span></span>
19. **int** main()
20. {
21. list<**int**> coll;
22. //insert values from 1 to 9
23. generate\_n (back\_inserter(coll), //start
24. 9, //number of elements
25. IntSequence (1)); //generates values
26. PRINT\_ELEMENTS(coll); //1 2 3 4 5 6 7 8 9
27. //replace second to last element but one with values starting at 42
28. generate (++coll.begin(), //start
29. --coll.end(), //end
30. IntSequence (42)); //generates values
31. PRINT\_ELEMENTS(coll); //1 42 43 44 45 46 47 48 9
32. }

#include <iostream>

#include <list>

#include <algorithm>

#include "print.hpp"

using namespace std;

class IntSequence {

private:

int value;

public:

//constructor

IntSequence (int initialValue)

: value(initialValue) {

}

//''function call''

int operator() () {

return value++;

}

};

int main()

{

list<int> coll;

//insert values from 1 to 9

generate\_n (back\_inserter(coll), //start

9, //number of elements

IntSequence (1)); //generates values

PRINT\_ELEMENTS(coll); //1 2 3 4 5 6 7 8 9

//replace second to last element but one with values starting at 42

generate (++coll.begin(), //start

--coll.end(), //end

IntSequence (42)); //generates values

PRINT\_ELEMENTS(coll); //1 42 43 44 45 46 47 48 9

}

注：**generate/generate\_n** 通过一个数值产生器类生成的数值来初始化一个容器, 通过将产生器函数**gen**返回的值赋给由迭代器指定的[first, last) or [first, first + n)范围内的所有元素， generate和generate\_n算法初始化或重新初始化了一个序列。

以引用方式传递仿函数：

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/lwbeyond/article/details/7323890)

1. generate\_n<back\_insert\_iterator<list<**int**> >, **int**, IntSequence&>
2. (back\_inserter(coll), //start
3. 4, //number of elements
4. seq); //generates values

generate\_n<back\_insert\_iterator<list<int> >, int, IntSequence&>

(back\_inserter(coll), //start

4, //number of elements

seq); //generates values

调用后，seq的值已经改变，下次再调用会用上次的值。

**三. for\_each()的返回值**

for\_each()可以返回其仿函数，这样你可以返回值获取仿函数的状态了。

**for\_each** (InputIterator***beg****,* InputIterator***end****,* UnaryProc***op***)， 对区间[beg, end)，中的每个elem调用op(elem)。

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/lwbeyond/article/details/7323890)

1. MeanValue mv = for\_each (coll.begin(), coll.end(), //range
2. MeanValue()); //operation
3. cout << "mean value: " << mv.value() << endl;

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

MeanValue()); //operation

cout << "mean value: " << mv.value() << endl;

**四. 预定义的仿函数**

#include <functional>

|  |  |
| --- | --- |
| negate<*type*>() | - *param* |
| plus<*type*>() | *param1* + *param2* |
| minus<*type*>() | *param 1* - *param2* |
| multiplies<*type*>() | *param1* \* *param2* |
| divides<*type*>() | *param1* / *param2* |
| modulus <*type*>() | *param1* % *param2* |
| equal\_to<*type*>() | *param1* == *param2* |
| not\_equal\_to<*type*>() | *param1* ! = *param2* |
| less<*type*>() | *param1* < *param2* |
| greater<*type*>() | *param1* > *param2* |
| less\_equal<*type*>() | *param1* <= *param2* |
| greater\_equal<*type*>() | *param1* >= *param2* |
| logical\_not<*type*>() | ! *param* |
| logical\_and<*type*>() | *param1* && *param2* |
| logical\_or<*type*> () | *param1* | | *param2* |

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/lwbeyond/article/details/7323890)

1. typedef set<**int**, greater<**int**> > IntSet;
2. typedef set<**int**, less<**int**> > IntSet;