函数对象

1 定义

若一个类重载了运算符”()”,则该类的对象就成为函数对象

```
class CMyAverage{//函数对象类
public:
        double operator()(int a1,int a2,int a3)
        {
                return (double)(a1+a2+a3)/3;
        }
};
        CMyAverage average;//函数对象
        cout<<average(3,2,3);//average.operator()(3,2,3)
        //输出：2.66667
```

2 函数对象的应用

Accumulate源代码1

```
template<typename_Inputlterator,typename_Tp>
_Tp accumulate(_Inputlterator__first,_Inputlterator__last,_Tp__init)
{
        for(;__first!=__last;++__first)
                __init=__init+*__first;
        return__init;
}//typename等价于class
```

源代码2

```
template<typename_Inputlterator,typename_Tp,typename_BinaryOperation>
  _Tp  accumulate(_Inputlterator__first,_Inputlterator__last,_Tp__init,  _BinaryOperation__b
  {
          for(;__first!=__last;++__first)
                __nint=__binary_op(__init,*__first);
              return__init;
  }
  //调用accumulate时，和__binary_op对应的实参可以是个函数或函数对象
```

3 函数对象的应用示例

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <numeric>
#include <functional>
using namespace std;
int sumSquares(int total,int value)
{
        return total+value*value;
}
template<class T>
void Printinterval(T first,T last)
{//输出区间[first,last)中的元素
        for(;first!=last;++first)
                cout<<*first<<" ";
                cout<<endl;
}
template<class T>
class SumPowers
{
        private:
                int power;
        public:
            SumPowers(int p):power(p){}
            const T operator()(const T&total,const T&value)
            {//计算value的power次方，加到total上
                    T v=value;
                    for(int i=0;i<power-1;++i)
                            v=v*value;
                    return total+v;
            }
};
int main()
{
        const int SIZE=10;
        int a1[]={1,2,3,4,5,6,7,8,9,10};
        vector<int>v(a1,a1+SIZE);
        cout<<"1)";PrintInterval(v.begin(),v.end());
        int result=accumulate(v.begin(),v.end(),0,SumSquares);
        cout<<"2)平方和:"<<result<<endl;
        result=accumulate(v.begin(),v.end(),0,SumPowers<int>(3));
        cout<<"3)立方和:"<<result<<endl;
        result=accumulate(v.bagin(),v.end(),0,SumPowers<int>(4));
```

```
        cout<<"4)4次方和:"<<result;
        return 0;
}
//输出:
//1)1 2 3 4 5 6 7 8 9 10
//2)平方和: 385
//3)立方和:3025
//4)4次方和:25333
```

4 greater的应用

```
#include <list>
#include <iosteam>
using namespace std;
class Myless
{
        public:
        bool operator()(const int&c1,const int&c2)
        {
                return (c1 % 10)<(c2 % 10);
        }
};
template<class T>
void Print(T first,T last)
{
        for(;first!=last;++first)
        cout<<*first<<",";
}
int main()
{
        const int SIZE=5;
        int a[SIZE]={5,21,14,2,3};
        list<int>lst(a,a+SIZE);
        lst.sort(MyLess());
        Print(lst.begin(),lst.end());
        cout<<endl;
        lst.sort(greater<int>());//greater<int>()是个对象
        Print(lst.begin(),lst.end());
        cout<<endl;
        return 0;
}
//输出:21,2,3,14,5
//    21,14,5,3,2
```

5 在STL中使用自定义的“大”，“小”关系

关联容器和STL中许多算法，都是可以用函数或函数对象自定义比较器的。在自定义了比较器op的情况下，以下三种说法是等价的：

1）x小于y

2)op(x,y)返回值为true

3)y大于x

6 例题

写出MyMax模板

```
#include <iostream>
#include <iterator>
using namespace std;
class MyLess
{
        public:
                bool operator()(int a1,int a2)
                {
                if((a1%10)<(a2%10))
                        return true;
                else
                        return false;
                }
                bool MyCompare(int a1,int a2)
                {
                if((a1%10)<(a2%10))
                        return false;
                else
                        return true;
};
template<class T,class Pred>
T MyMax(T first,T last,Pred myless)
{
        T tmpMax=first;
        for(;first!=last;++first)
        if(myless(*tmpMax,*first))
        tmpMax=first;
        return tmpMax;
};
int main()
```

```
{
        int a[]={35,7,13,19,12};
        cout<<*MyMax(a,a+5,MyLess())<<endl;
        cout<<*MyMax(a,a+5,MyCompare)<<endl;
        return 0;
}
//输出：19 12
```