6.10 函数对象_物联网 / 嵌入式工程师 - 慕课网

> 慕课网慕课教程 6.10 函数对象涵盖海量编程基础技术教程，以图文图表的形式，把晦涩难懂的编程专业用语，以通俗易懂的方式呈现给用户。

10. 函数对象

一、介绍

如果一个类将 () 运算符重载为成员函数，这个类就称为函数对象类，这个类的对象就是函数对象。函数对象是一个对象，但是使用的形式看起来像函数调用，实际上也执行了函数调用，因而得名。

二、实现

```
#include <iostream>


using namespace std;


class Average{
      public:
          int operator()(int a1,int a2,int a3)
          {
              return (a1 + a2 + a3) / 3;
          }
};


int main(void)
{
        Average average;


        cout << average(1,2,3) << endl;


        return 0;
}
```

三、STL 内建函数对象

STL 内建了一些函数对象，用法和一般函数完全相同，使用内建函数对象，需要引入头文件

#include，主要分为以下三类:

- 算术仿函数
- 关系仿函数
- 逻辑仿函数

1. 算术仿函数

```
template<class T> T plus< T > // 加法仿函数

template<class T> T minus< T > //减法

template<class T> T multiplies< T > //乘法

template<class T> T divides< T > //除法

template<class T> T modulus< T > //取模

template<class T> T negate< T > //取反，一元运算
#include <iostream>

#include <functional>


using namespace std;


int main(void)

{

    plus<int> p;

    cout << p(10,20) << endl;


    negate<int> n;

    cout << n(50) << endl;


    return 0;

}
```

2. 关系仿函数

```
template<class T> bool equal_to< T > // 等于

template<class T> bool not_equal_to< T > //不等于

template<class T> bool greater< T > //大于

template<class T> bool greater_equal< T > //大于等于

template<class T> bool less< T > //小于

template<class T> bool less_equal< T > //小于等于
#include <iostream>

#include <functional>

#include <vector>

#include <algorithm>
```

```
using namespace std;


class MyCompare

{

public:

    bool operator()(int v1,int v2)

    {

        return v1 > v2;

    }

};


int main()

{

     vector<int> v;


    v.push_back(10);

    v.push_back(30);

    v.push_back(40);

    v.push_back(20);

    v.push_back(50);


    for (vector<int>::iterator it = v.begin(); it != v.end(); it++)

    {

        cout << *it << " ";

    }

    cout << endl;


#if 0

     template< class RandomIt, class Compare >

     void sort( RandomIt first, RandomIt last, Compare comp );

#endif


    //方法一：自己实现仿函数

    //sort(v.begin(), v.end(), MyCompare());


    //方法二：使用STL内建仿函数  大于仿函数

    sort(v.begin(), v.end(), greater<int>());


    for (vector<int>::iterator it = v.begin(); it != v.end(); it++) {

        cout << *it << " ";

    }

    cout << endl;


    return 0;
```

```
}
```

3. 逻辑仿函数

```
template<class T> bool logical_and< T > // 逻辑与

template<class T> bool logical_or< T > //逻辑或

template<class T> bool logical_not< T > //逻辑非
#include <iostream>

#include <functional>

#include <vector>

#include <algorithm>


using namespace std;


int main()

{

    vector<bool> v;


    v.push_back(true);

    v.push_back(false);

    v.push_back(true);

    v.push_back(false);


    for (vector<bool>::iterator it = v.begin();it!= v.end();it++)

    {

        cout << *it << " ";

    }

    cout << endl;


    //逻辑非  将v容器搬运到v2中，并执行取反操作

    vector<bool> v2;

    v2.resize(v.size());

    //template<class T> bool logical_not<T> //逻辑非

    transform(v.begin(), v.end(),  v2.begin(), logical_not<bool>());


    for (vector<bool>::iterator it = v2.begin(); it != v2.end(); it++)

    {

        cout << *it << " ";

    }

    cout << endl;


    return 0;
```

```
}
```

四、使用函数对象的好处

- 通过函数对象调用 operator()，可以通过内联省略函数的调用开销，比通过函数指针调用函数（不能内联）效率高。

> 使用函数对象虽然我们没有显示加上 inline 关键字，但编译器优化时可能会优化成内联

- 因为函数对象是用类生成的，所以类中可以添加相关的成员变量，用来记录函数对象使用时的更多信息
- 划线
- 写笔记

学习要认真，笔记应当先

公开笔记 0/1000 提交

Sunny_SunshineX

删除 编辑