



多态与运算符重载-例题

数组类与运算符重载

1. 题目内容与要求

- ① 首先设计一个数组类，其成员变量包括数组名称、数组元素个数和数组变量；
- ② 然后定义成员函数包括构造函数、析构函数以及数组元素显示函数、数组排序函数和两个数组的加法函数等；
- ③ 最后在主函数中定义数组类的对象，并调用这些函数显示数组元素的值、对一个数组进行排序以及计算两个数组的加。

2. 类的分析

数组类
数组标签 数组大小 数组名
构造函数 析构函数 显示函数 排序函数 加法函数

主函数
数组类对象
对象函数调用

3. 类的设计

Array
string name int count int *data
Array() ~Array() print() order() add()

main
Array a1; Array a2;
a1.print(); a2.print(); (a1+a2).print()

4. 数组类

```
class Array // 数组类
{
private:
    string name; // 数组名称
    int count; // 数组大小
    int *data; // 数组变量
public:
    Array(); // 默认构造函数
    Array(string name,const int * data,int count); // 构造函数
    ~Array(); // 析构函数
    void print(); // 显示数组元素函数
    void order(); // 数组排序函数
    Array add(const Array & array); // 数组加法函数
};
```

构造函数

```
Array::Array(){
    name="";
    count=0;
    data=NULL;
}
Array::Array(string name,const int * data,int count){
    this->name=name;
    this->count=count;
    this->data = new int[this->count];
    for(int i=0; i<this->count; i++) {
        this->data[i]=data[i];
    }
}
```

析构函数与显示函数

```
Array::~~Array(){  
    if(data!=NULL) {  
        delete [] data;  
        count=0;  
    }  
}  
  
void Array::print(){  
    cout<<name<<": ";  
    for(int i=0; i<count; i++) {  
        cout<<data[i]<<" ";  
    }  
    cout<<endl;  
}
```


数组排序函数

```
void Array::order() { // 数组排序
    for(int i=0; i<count-1; i++) {
        for(int j=0; j<count-i-1; j++) {
            if(data[j]>data[j+1]) {
                int temp=data[j];
                data[j]=data[j+1];
                data[j+1]=temp;
            }
        }
    }
}
```

加法函数

```
Array Array::add(const Array & array){
    Array temp;
    int max=(count>array.count?count:array.count);
    int min=(count<array.count?count:array.count);
    temp.name=this->name+" "+array.name;
    temp.count=max;
    temp.data=new int[temp.count];
    int i=0;
    for(i=0; i<min; i++) {
        temp.data[i]=data[i]+array.data[i];
    }
    for(; i<count; i++) {
        temp.data[i]=data[i];
    }
    for(; i<array.count; i++) {
        temp.data[i]=array.data[i];
    }
    return temp;
}
```

主函数main程序代码

```
#include <iostream>
using namespace std;
```

<类的定义在此!>

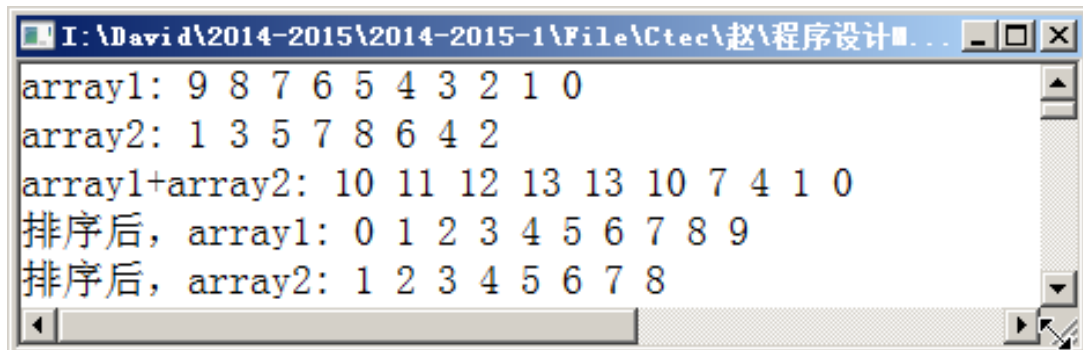
```
int main() // 主函数
{
    <核心代码在此!>
    return 0;
}
```

```
int data1[] = {9,8,7,6,5,4,3,2,1,0};
Array array1("array1",data1,10);
array1.print();
int data2[] = {1,3,5,7,8,6,4,2};
Array array2("array2",data2,8);
array2.print();
```

```
(array1.add(array2)).print();
array1.order();
cout<<"排序后, ";array1.print();
array2.order();
cout<<"排序后, ";array2.print();
```



5. 运行结果



```
I:\David\2014-2015\2014-2015-1\File\Ctec\赵\程序设计...  
array1: 9 8 7 6 5 4 3 2 1 0  
array2: 1 3 5 7 8 6 4 2  
array1+array2: 10 11 12 13 13 10 7 4 1 0  
排序后, array1: 0 1 2 3 4 5 6 7 8 9  
排序后, array2: 1 2 3 4 5 6 7 8
```

6. 程序分析与思考

问题



```
a1.print();  
a2.print();  
a1.add(a2).print();
```

这些调用太繁琐，如何简化？

答案



使用+和<<的运算符
重载函数

详细格式



operator+代替add函数
operator<<代替print函数

修改后的显示函数

```
ostream & operator<<(ostream & out,const Array & array){  
    out<<array.name<<": ";  
    for(int i=0; i<array.count; i++) {  
        out<<array.data[i]<<" ";  
    }  
    out<<endl;  
    return out;  
}
```

修改后的加法函数

```
Array Array::operator+(const Array & array){
    Array temp;
    int max=(count>array.count?count:array.count);
    int min=(count<array.count?count:array.count);
    temp.name=this->name+" "+array.name;
    temp.count=max;
    temp.data=new int[temp.count];
    int i=0;
    for(i=0; i<min; i++) {
        temp.data[i]=data[i]+array.data[i];
    }
    for(; i<count; i++) {
        temp.data[i]=data[i];
    }
    for(; i<array.count; i++) {
        temp.data[i]=array.data[i];
    }
    return temp;
}
```

修改后的主函数调用

```
int data1[] = {9,8,7,6,5,4,3,2,1,0};  
Array array1("array1", data1, 10);  
cout<<array1;  
int data2[] = {1,3,5,7,8,6,4,2};  
Array array2("array2", data2, 8);  
cout<<array2;
```

```
cout<<(array1+array2);  
  
array1.order();  
cout<<"排序后, "<<array1;  
  
array2.order();  
cout<<"排序后, "<<array2;
```



7. 延伸思考

问题



=、-、*、/、...
如何重载？

8. 小结

- ✓ 运算符重载是多态的另一种表现形式，许多基本运算符都可以实现重载。
- ✓ 运算符重载的作用是免去了调用的麻烦。
- ✓ 基本运算符重载的定义格式和使用格式：
 - +运算符重载函数格式如下：operator+、a1+a2
 - =运算符重载函数格式如下：operator=、a4=a3
 -

例题的讲解就到这里!

谢谢!

