

# day06

## day06

- 一、结构体在C语言在的操作
- 二、友元函数
- 三、运算符重载
  - 1.运算符重载意义
  - 2.运算符重载—成员函数定义
  - 3.运算符重载—全局函数方式
  - 4.运算符重载—输出运算符重载<<
  - 5.运算符重载—输入运算符重载
- 四、lambda表达式
  - 1.lambda表达式的介绍
  - 2.lambda的调用

## 一、结构体在C语言在的操作

结构体和类包括函数一样，但是，结构体里面的函数都是公开的

## 二、友元函数

类当中的方法默认是私有的

```
1  #include <iostream>
2  using namespace std;
3  class car{
4      string color = "黑色";
5      // 标记modifyCar是car的好朋友
6      friend void modifyCar(car & cc);
7  public:
8      string getColor(){
9          return color;
10     }
11 };
12 void modifyCar(car & cc){
13     cc.color = "白色";
14 }
15 int main() {
16     car c;
17     cout << c.getColor() << endl;
18     // 调用友元函数
19     modifyCar(c);
20     cout << c.getColor() << endl;
21     return 0;
```

```
22 }
23 // 运行结果:
24 黑色
25 白色
```

## 友元类

```
1  #include <iostream>
2  using namespace std;
3  class Car{
4      string color = "黑色";
5  public:
6      string getColor(){
7          return color;
8      }
9      friend class SSSS;
10 };
11 class SSSS{
12 public:
13     void modify(Car &c){
14         c.color = "红色";
15     }
16 };
17 int main() {
18     Car c;
19     cout << c.getColor() << endl;
20     SSSS s;
21     s.modify(c);
22     cout << c.getColor() << endl;
23     return 0;
24 }
25 // 运行结果:
26 黑色
27 红色
```

## 三、运算符重载

### 1.运算符重载意义

运算符重载，就是为了，让运算符有更多的特征，可以让运算符匹配更多的类型，让代码看起来更加优雅

运算符重载有两种方式，一种是成员函数，一直是全局函数

### 2.运算符重载—成员函数定义

```

1 // 运算符重载
2 // s1 + s2 谁在前面，就是谁调用的这个运算符重载的函数，谁在后面，谁就是参数
3 返回值类型 operator符号(后一个对象){
4     return 返回值;
5 }

```

```

1 class stu{
2 public:
3     int money;
4     stu(int money):money(money){
5
6     }
7     // 运算符重载
8     // s1 + s2 谁在前面，就是谁调用的这个运算符重载的函数，谁在后面，谁就是参数
9     int operator+(stu & s){
10         return this->money + s.money;
11     }
12 };
13 int main() {
14     stu s1(10);
15     stu s2(20);
16     int a = s1 + s2;
17     cout << a << endl;
18     return 0;
19 }
20 // 运行结果:
21 30

```

用引用，防止数据复制，增加内存，提高运行效率

### 3.运算符重载—全局函数方式

在类中时，只需要传一个参数，当为全局时，需要传两个参数

```

1 int operator+(stu s1,stu s2){
2     return s1.money + s2.money;
3 }

```

### 4.运算符重载—输出运算符重载<<

```

1 class stu{
2 public:
3     string name;
4     int age;
5     stu(){
6     }
7     stu(string name,int age):name(name),age(age){}
8 };
9 // ostream 中将拷贝禁用了，所以必须要用引用
10 void operator<<(ostream& o,stu& s1){
11     cout << s1.name << ":" << s1.age << endl;

```

```

12 }
13 int main() {
14
15     stu s2("李四",20);
16     cout <<s2;
17     return 0;
18 }
19 // 运行结果:
20 李四:20

```

```

1 class stu{
2 public:
3     string name;
4     int age;
5     stu(){
6     }
7     stu(string name,int age):name(name),age(age){}
8 };
9 // ostream 中将拷贝禁用了，所以必须要用引用
10 ostream& operator<<(ostream& o,stu& s1){
11     cout << s1.name << ":" << s1.age << endl;
12     return o;
13 }
14 int main() {
15     stu s1("张三",18);
16     stu s2("李四",20);
17     cout << s1 << s2;
18     return 0;
19 }
20 // 运行结果:
21 张三:18
22 李四:20

```

## 5.运算符重载—输入运算符重载

```

1 class stu{
2 public:
3     string name;
4     int age;
5     stu(){
6     }
7     stu(string name,int age):name(name),age(age){}
8 };
9 // >> 输出运算符重载
10 void operator>>(istream& in,stu& s1){
11     cout << "请输入你的姓名:" << endl;
12     in >> s1.name;
13     cout << "请输入你的年龄:" << endl;
14     in >> s1.age;
15 }

```

```

16 int main() {
17     stu s1("张三",18);
18     cin >> s1;
19     cout << s1.name << endl;
20     cout << s1.age << endl;
21     return 0;
22 }
23 // 运行结果:
24 请输入你的姓名:
25 李四
26 请输入你的年龄:
27 25
28 李四
29 25

```

补

### 1. 赋值运算符

```

1 stu s1("张三",18);
2 stu s2("李四",20); // 走的是有参构造
3 stu s3 = s1;      // 走的是拷贝构造
4 s2 = s1; // 此处用了拷贝赋值运算符

```

默认情况下，类里面有赋值运算符

### 2. 调用运算符()

## 四、lambda表达式

### 1.lambda表达式的介绍

提高代码运行效率，但是阅读性差，理解难度高

实现方法

```

1 // 匿名函数
2 []()->int{};

```

**[]:** 必须要有这是捕获列表

**():** 表示函数参数，如果没有参数，小括号可以不用写

**->:** 连接函数体，可以不写，会推断出来

**int:** 函数的返回值，可以不写，会推断出来

**{}**: 函数体

```

1 []()->void{cout << "你好" << endl;};
2 []{
3     cout << "你好" << endl;
4 };

```

## 2.lambda的调用

### 1. 定义函数指针来接收

```
1 // 定义函数指针来接收
2 void (*p)() = []{cout << "你好" << endl;};
```

### 2. 使用auto来调用

```
1 auto a = []{cout << "你好,lambda" << endl;};
```

### 3. 使用调用运算符 ()

```
1 []{cout << "你好,lambda111" << endl;}();
```

### 使用

```
1 int c = [](int a,int b){return a+b;}(10,20);
2 cout << c << endl;
3 // 运行结果:
4 30
```