构造函数

首先构造函数的名字与class类名是一样的

```
class Person {
public:
    //构造函数
    //1 没有返回值，也不用写void
    // 2 函数名与类的名称相同
    // 3 可以有参数，可以发生重载
    // 4创建对象时构造函数自动调用，只调用一次
    Person() {
        cout << "构造函数的调用" << endl;
    }
}


//按照类型分类 普通构造函数 拷贝构造函数
class Person {
public:
    Person() {
        cout << "Person的无参构造函数的调用" << endl;
    }

    Person(int a) {
        age = a;
        cout << "Person的有参构造函数的调用" << endl;//构造函数为main函数中的创建对象添加初始属性
    }

    //拷贝构造函数，其他的都是普通构造
    Person(const Person &p) {
        //将传入的对象的所有属性拷贝到自身身上
        age = p.age;
        cout << "Person的拷贝构造函数的调用" << endl;
    }

    ~Person() {
        cout << "析构函数的调用" << endl;
    }
    int age;
};
```

无参构造和有参构造就不说了，用于接收外部实例化的时候的传入参数或者直接默认实例化

主要讲一下拷贝构造函数

拷贝构造函数需要传入一个const Classname &obj

那为什么是const呢，因为我们希望后面的参数也就是那个原始对象不被修改

有& 代表我们传入的只是一个引用，并非那个对象本身

现在来说一下在main函数中实例化一个对象的调用构造函数操作

```
//1 括号法
```

```
//1 括号法
    //Person p1;//默认构造函数调用
    //Person p2(10);//有参构造函数调用
    //Person p3(p2);//拷贝构造函数

    //注意事项
    //调用默认构造函数的时候不要加()
    //若加了括号，可能不会创建类对象，则编译器只会认为是函数的声明
    //Person p1(); 可能只是默认声明了一个函数
//2 显示法
    Person p1;
    Person p2 = Person(10);//调用有参构造
    Person p3 = Person(p2);//调用拷贝构造

    Person(10);// 匿名对象  当前行执行结束后，系统回收匿名对象，即调用构造函数后立即调用析构函数

    //注意事项  不要利用拷贝构造函数构造匿名对象  编译器认为Person(p3) 等价于 Person p3 会重定义
    //Person(p3);
    // 3 隐式转换法
    Person p4 = 10; //相当于写Person p4 = Person(10)
    Person p5 = p4; //拷贝构造函数
```

关于拷贝构造函数，有一些其他的创建方法

```
//1 使用一个已经创建完毕的对象来初始化一个新对象
void test_copycons() {
    Person p10(20);
    Person p11(p10);//拷贝构造
    cout << "p2's age:" << p11.m_Age << endl;
}
//2 值传递的方式给函数参数传值
void do_work(Person p) {
    //...
}
void test_numtrans() {
    Person p13;//直接在函数体内制造
    do_work(p13);
}
// 3 值方式返回局部对象
Person do_work1() {
    Person p14;
    cout << (int*)&p14 << endl;
    return p14;//通过值返回来赋值
}
void test_numback() {
    Person p = do_work1();// 由do_work1函数创造出的p14对象，被等式赋值给了p，p14在赋值前就已经被析构了
    cout << (int*)&p << endl;
}
```

现在来讲一下关于构造函数的调用规则

```
//写了拷贝构造函数，则系统不会提供其它类型的构造函数，其他的如果不写则调用不了
//本身调用默认构造函数，如果写了有参构造函数，系统就不会再提供默认构造函数（但是会有拷贝构造函数），没有默认构造，系统会报错
//如果写了有参构造函数，则一定要在创建对象的时候调用有参构造
```

如果你自己什么构造函数都不写，系统就直接调用了默认构造函数

如果你写了一个有参构造函数，那么系统默认构造函数就被替代了，这个时候你就要自己重新写一个无参构造函数，来实现无参实例构造。

但是这个时候系统还会默认提供给你一个拷贝构造函数，你还是可以用的捏

如果你写了一个拷贝构造函数，那么系统的所有默认构造函数都会被替代，默认就自己写，想用有参构造也自己写

析构函数

```
~Person() {
    cout << "析构函数的调用" << endl;
}
//析构函数
    // 1 没有返回值， 不写void
    // 2 函数名和类名相同，在前面加“~”
    //3 不可以有参数，不可以发生重载
    //4 对象在销毁前会调用一次
```

名字与类名相同，但是前面多加一个~

当这个对象的生命周期结束了之后就会调用这个来销毁对象释放内存