构造析构

通过学习构造析构函数来完成对对象的数据成员进行初始化和清理工作

一、构造函数

1、知识点介绍

1、构造函数，它是一种特殊的函数，主要用来在创建对象时初始化对象，即为对象的成员变量赋初始值。

2、构造函数的定义

1、构造函数名和类名相同
2、构造函数没有返回值类型，和返回值
3、构造函数可以重载，需要满足函数重载的条件

```
class student
{
 public:
    student(){}//无参(默认)构造
    student(int a){}//有参(带参)构造
}
```

3、构造函数的调用时机

1、在创建一个新的对象的时候会调用
调用无参构造
```
student stu;
student *p=new student;
```
调用带参构造
```
student stu(1);
student *p=new student(1);
```

4、构造函数的特点

1、如果一个类中没有显示的给出构造函数，系统会自动地给出一个缺省的(隐式)什么都不干的构造函数
2、如果类中有多个构造函数，那么通常会有不同的参数列表和函数体
3、如果用户提供了无参(有参)构造，那么系统就不再提供默认构造
4、类中如果只有有参构造，没有无参构造，那么就不能调用默认构造的方式初始化对象，想用这种方式初始化对象那么就提供无参构造

5、拷贝构造(一种特殊的构造)

1、拷贝构造是一中特殊的构造函数，用自身这种类型来构造自身.

示例：

```
student stu1;
student str2=stu2;//在这一句就会调用拷贝构造
```

拷贝构造的定义：

1、没有写拷贝构造，系统会提供一个隐式的拷贝构造，而这个拷贝构造的操作，我们可以理解为，是用‘=’号一个一个的将存在于对象中的数据成员赋值给新创建的对象中。

2、自定义拷贝构造

类名(const 类名& 引用名){}

这里的const不加也可以，但是这里是拷贝，也就是复制的意思，所以加上，防止被改

在函数体里面通常的做法是逐一拷贝传进来的对像的成员，赋值给新的对象，当然也可以自己定义内容

示例：

```
class Student
{
    int id;
public:
    Student(const Student&stu)
    {
        this->id=stu.id;
        //把传进来对象的id赋值给当前调用对象的id
    }
}
```

拷贝构造的调用

1、在用同种类的的对象，去初始化另一个对象的时候，注意是在定义对象初始化，不是赋值

```
Student stu1;
Student stu2=stu1;//显示调用拷贝构造
Student stu3(stu2);//隐式调用拷贝构造
Student *pStu=new Student(stu3);
```

2、在函数传参时，函数的形参是类对象

```
void fun(Student stu){}
fun(stu1);//在函数调用传参时调用拷贝构造
```

3、如果一个函数的返回值类型时对象，在函数调用结束，返回对象的时候调用拷贝构造

```
Student fun1(Student stu){return stu;}
//这里在函数调用的时候会调用两次拷贝构造，函数传参一次，函数返回时一次
```

拷贝构造的问题

1、浅拷贝

拷贝构造我们不写，系统也会提供一个默认的拷贝构造，而这个拷贝构造的操作，我们可以理解为，是用‘=’号一个一个的赋值的，我们将之称为，浅拷贝，因为在用指针的时候就可以能会出现问，因为我们知道两个同等类型的指针之间用‘=’号赋值，是两个指针的地址指向同一个内存，那么就可能会存在一个问题，就是两个对象的指针都指向同一个内存，那么如果其中一个对象把该内存释放了，就会导致另外一个对象的指针变成野指针。

2、深拷贝

也就是在有上述的问题的时候才需要深拷贝，而深拷贝做的事情也就是自己定义拷贝构造，给新的对象的指针申请内存，来存内容，而不是两个对象的指针指向同一个内存

示例：

```
class Student
{
    char *name;
public:
    Student(char *name)
    {
        this->name=new char[strlen(name)+1];//加1是为了保存'\0'
        strcpy(this->name,name);
    }
    Student(const Student& stu)
    {
        //这个自定义拷贝构造函数做的事，其实就是给新对象的指针申请内存来存数据，如果有其他成员，那么也要记得赋值
        this->name=new char[strlen(stu.nema)+1];//加1是为了保存'\0'
        strcpy(this->name,stu.name);
    }
}
```

1、什么时候要自己写拷贝构造
类中有动态申请内存的时候，必须要写

二、析构函数

1、知识点介绍

析构函数和构造函数一样，也是一种特殊的函数，主要的作用是在对象结束声明周期时，系统自动调用析构函数，来做一些清理工作，比如上面，在对象中有申请内存，那么时需要自己去释放内存的，这个释放内存的操作就可以写在析构函数中，在对象死亡的时候自动调用析构函数释放内存，那么这种就不要担心忘记释放内存了

2、析构函数的定义

1、函数名与类名相同，在前面加上一个~
~Student(){}
2、没有返回值类型和返回值，也没有参数
3、如果类中没有自己写析构函数，那么系统将会给出一个隐式什么都不干的析构函数

3、析构函数的调用时机

1、析构函数可以主动通过对象调用，析构函数必须是公有属性下
2、在对象死亡时，析构函数会主动调用他的析构函数

4、析构函数的特点

1、析构函数做的事是对对象做一些清理工作，主动调用析构函数，并不会释放对象
2、一个类只有一个析构函数

三、this指针

1、知识点介绍

1、this指针是系统自动生成，且隐藏，我们看不到定义，但是可以使用
2、this指针并不是对象本身的一部分，它的作用域在类的内部。当类的普通函数在访问类的普通成员的时候，该this指针总是指向调用者对象。

2、this指针的使用

1、必须在类中使用，在类外是使用不了的
2、this->成员名;或者(*this).成员名；　　表示调用者的某个成员
3、return this;　　表示返回当前调用者对象的地址
4、return *this;　　表示返回当前调用者对象

3、this指针在代码中的表现

1、在类中函数的形参和类中成员同名

```
void MyClass::fun(int sum)
{
    this->sum=sum;
}
```

这样我们就能通过this指针指向sum，来表示this指向的这个sum是当前对象的sum。如果是sum=sum;那么这两个sum都是表示的形参