第4讲

对象的实例化和清理

主讲人：赵文彬

本次课主要内容

- 构造函数
- 析构函数
- 对象数组
- 对象指针

构造函数

```
class time
{
};
```

```
class time
{ public:
  int hour;
  int minute;
  int sec;
};
time t1={14,56,30};
```

private?或Protected?

构造函数

- 构造函数
  - 处理对象的初始化
  - 一种特殊的成员函数
  - 在建立对象时自动执行
  - 函数名必须与类名相同
  - 不返回任何值，不能指定包括void在内的各种返回值类型
  - 功能由用户定义

构造函数

```
#include <string>
#include <iostream>
using namspace std;
void main( )
{class stud     // 类定义开始
 { private:
    int num;
    char name[10];
    char sex;
  public:
stud(int n,char xm[ ],char s )
   {num=n;
    strcpy(name,xm);
    sex=s;}
void display( )
{ cout<<"num"<<num;
   cout<<"name"<<name;
    cout<<"sex"<<sex<<endl;
 }
 };  // 类定义结束
stud  s1(10010,”wang”,’M’)
s1.display( );
}
```

构造函数可以不带参数，也可以带参数

构造函数

- 说明
  - 构造函数是在类对象进入其作用域时被调用。
  - 构造函数不需要，也不能被用户调用。

```
stud.student( )
```

  - 如果用户自己没有定义构造函数，C++系统会自动生成一个构造函数，只是该构造函数的函数体是空的，也没有参数，不执行初始化操作。
  - 在构造函数的函数体中除了可以对数据成员赋初值，还可以包含其他语句，但为了保持程序的清晰，一般不提倡加入其他与初始化无关的内容。

构造函数

```
//"rect.h"文件
#pragma once
class rect
{ private:
        double len,wid;
  public:
        rect(double,double);
        double area();};
//"rect.cpp"文件
#include "rect.h"
rect::rect(double l,double w):len(l),wid(w) {}
double rect::area()
{return len*wid;}
```

```
//"main.cpp"文件
#include "rect.h"
#include <iostream>
using namespace std;
void main()
{  rect r1(3,4);
   cout<<"Its area is
"<<r1.area()<<endl;
   rect r2(1.2,3);
   cout<<"Its area is
"<<r2.area()<<endl;
}
```

```
rect::rect(double l,double w)
 {len=l;wid=w;}
```

构造函数

- 构造函数的重载

```
#include <iostream>
using namespace std;
class rect
{public:
        rect();
        rect(int l,int w):len(l),wid(w) { }
        int area();
  private:
        int len;
        int wid;};
rect::rect()
{       len=10;
        wid=5;
}
```

```
int rect::area()
{return len*wid;}
void main()
{       rect r1;
        cout<<r1.area()<<endl;
        rect r2(1,2);
        cout<<r2.area()<<endl;
}
```

构造函数

默认参数的构造函数

```
#include <iostream>
using namespace std;
class rect
{public:
        rect(int l=10,int w=5);
        int area();
  private:
        int len;
        int wid;
};
rect::rect(int l,int w)
{       len=l;
        wid=w;
}
```

```
int rect::area()
{return len*wid;}
void main()
{       rect r1;
        cout<<r1.area()<<endl;
        rect r2(7);
        cout<<r2.area()<<endl;
        rect r3(4,9);
        cout<<r3.area()<<endl;
}
```

构造函数

- 说明
  - 指定默认参数应在类定义中声明构造函数时进行，而不能只在定义构造函数时指定。
  - 一个类只能有一个默认构造函数。
  - 如果构造函数的全部参数都指定了默认值，在定义对象时可以给一个或几个实参，也可以不给实参。
  - 一个类中定义了全部是默认参数的构造函数后，不能再定义重载构造函数。

构造函数

- 复制构造函数
  - 一般形式

类名 对象2（对象1）

```
rect r2(r1);
```

复制构造函数的形式为：

```
rect::rect(const rect & a)
{
    len=a.len;
    wid=a.wid;
}
```

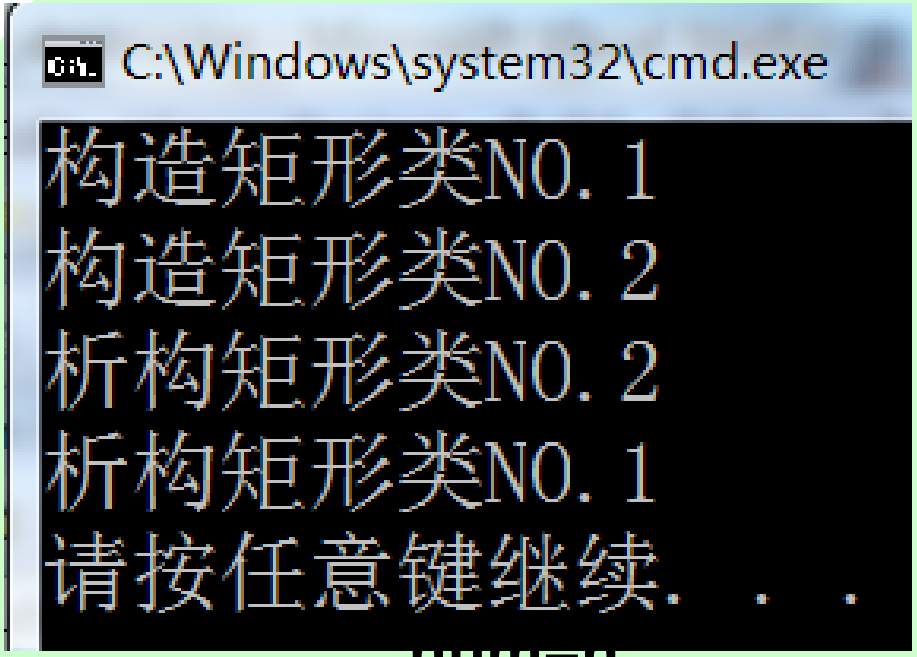
两种方式均调用复制构造函数

类名 对象2=对象1　　rect r2=r1;

析构函数

- 作用
  - 在撤销对象占用的内存之前完成一些清理工作，而非删除对象。
  - 执行用户希望在最后输出一次使用对象之后所执行的任何操作，比如有关信息。
- 说明
  - 函数名为类名前加“~”(取反运算符)
  - 没有函数参数，不返回任何值，没有函数类型
  - 不能被重载
  - 一个类可以有多个构造函数，却只能有一个析构函数。

析构函数




```
#include "rect.h"
void main()
{
        rect r1(10,5,1);
        rect r2(5,1,2);
}
```

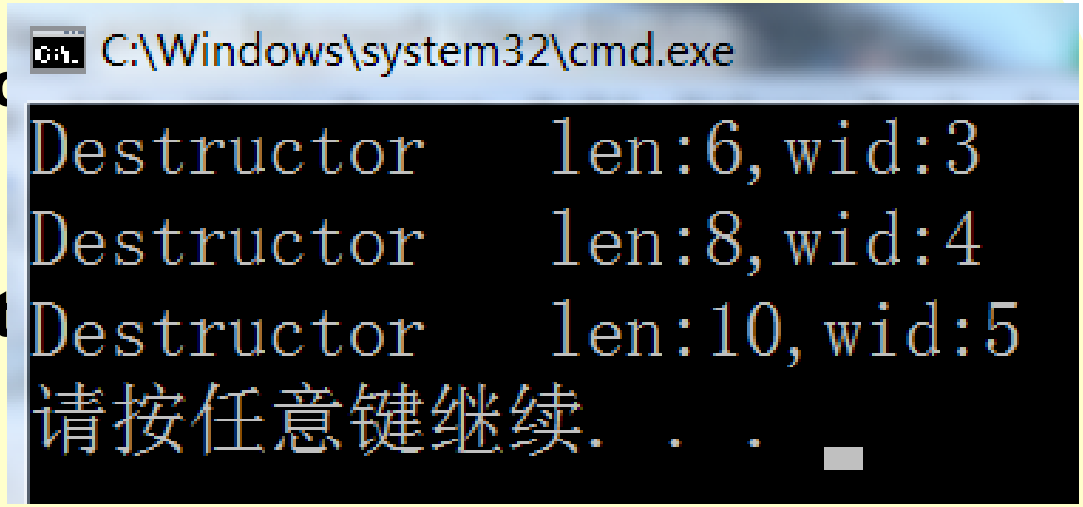
```
        num=n;
        cout<<"构造矩形类"<<"NO."<<num<<endl;   }
~
{
private:
        int len,wid,num;};
```

先构造的后析构，后构造的先析构

析构函数

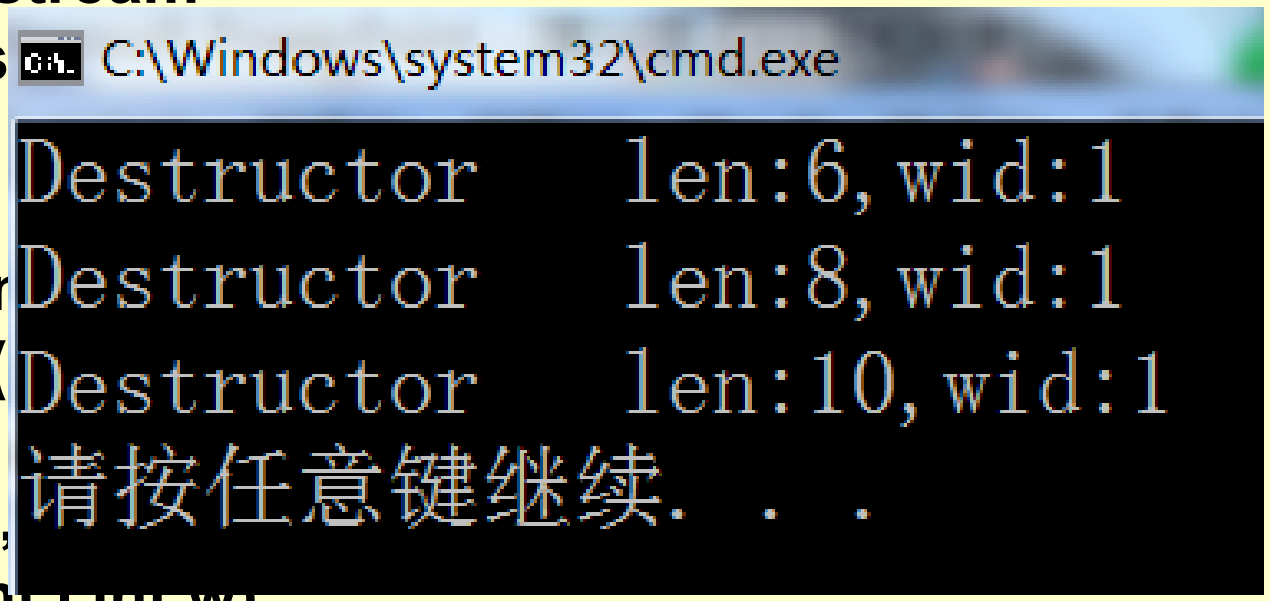
- 说明
  - 如果用户**没有定义**构造函数/析构函数，C++编译系统会**自动生成**一个构造函数/析构函数，但只是徒有名称和形式，实际上**什么操作都不进行**。
  - **构造函数**在**建立**对象时调用，**析构函数**在**撤销**对象占用内存前调用。

```
#include <iostream>
using namespace st
class Rect
{public:
        Rect(int l=2,int
        ~Rect();
 private:
        int len,wid;};
Rect::
Re
{cou                                    <<wid<<endl;
};
v
{
        Rect r[3]={Rect(10,5),Rect(8,4),Rect(6,3)};
}
```



建立对象数组时，分别调用构造函数，对每个元素初始化，每个元素的实参分别用括号包起来，不会混淆

```
#include <iostream>
using names
class Rect
{public:
        Rect(ir
        ~Rect(
 private:
        int len,
Rect::Rect(int l,int w)
{len=l;wid=w;}
Rect::~Rect()
{cout<<"Destructor   len:"<<len<<",wid:"<<wid<<endl;
};
void main()
{
        Rect r[3]={10,8,6};
}
```



三个实参分别作为每个元素的第一个实参

对象指针

- 指向对象的指针

```
#include <iostream>
using namespace std;
class Rect
{
public:
        int len,wid;
        Rect(int l,int w)
        {len=l;wid=w;}
        int Area()
        {return len*wid;}
};
```

```
void main()
{
        Rect r1(10,5), *p;
        p=&r1;
        cout<<(*p).len<<endl;
        cout<<p->wid<<endl;
        cout<<p->Area()<<endl;
}
```

同结构体

```
#include <iostream>
using namespace std;
class Rect
{public:
        Rect(int l=10,int w=5)
        {len=l;wid=w;}
        int Area()
        {return len*wid;}
 private:
        int len,wid;};
void main()
{       Rect r[3]={Rect(2,1),Rect(3,2),Rect(4,2)};
        cout<<"The area of r[0] is "<<r[0].Area()<<endl;
        cout<<"The area of r[1] is "<<r[1].Area()<<endl;
        cout<<"The area of r[2] is "<<r[2].Area()<<endl;
}
```

怎样区分使用的是哪个对象元素的数据成员？

对象指针

- this 指针
  - 每个成员函数都包含一个特殊的指针—this指针。
  - this指针的值是当前被调用的成员函数所在的对象的起始地址。
  - this指针是隐式使用的，作为参数传递给成员

```
int Rect::Area()
{return len*wid;}
```

```
int Rect::Area()
{return this->len*this->wid;}
```

小结

- 构造函数
- 析构函数
- 对象数组
- 对象指针
  - this指针

小结

- 作业
  - 先定义一个点类，类名为point，用课堂教师演示的方式添加类，即类的定义要在头文件中，另外有一个描述类成员函数实现的cpp文件，还有一个主函数的文件。
    - 将其三维坐标定义为私有成员，通过构造函数为其初始化，并在构造函数和析构函数中有输出语句，以便于从运行结果看出构造函数析构函数的运行。
    - 写三个构造函数用于重载，包含一个默认构造函数。
    - 定义一个对象指针，并通过该指针完成对点对象坐标的输入和输出。
    - 定义对象数组，观察构造函数和析构函数调用的顺序。