面向对象程序设计
实验辅导

闫哲（yanzhe@china.com.cn）

2005-3-8

北京大学

课程介绍

- 目的：帮助大家顺利通过实验考试
- 形式：知识回顾、考题分析、思路总结
- 安排：
  - 明确考试要求
  - 重点知识回顾
  - 典型考题分析
  - 编程环境使用调试
  - 常见问题总结

考试要求

- 语法
  - C++是程序设计语言
  - 语法是基础
  - 熟悉各种语法成分
  - 掌握三种程序控制结构
  - 达到熟练运用程度

考试要求

- 面向对象概念
  - 培养面向对象程序设计思想
  - 理解面向对象中的重要概念
  - 掌握面向对象机制的实现
  - 完成面向对象程序设计

考试要求

- 算法
  - 程序 = 算法 + 数据结构
  - 算法和数据结构→类
  - 掌握常用算法
  - 具备初步算法设计能力

考试要求

- 上机环境编程
  - 实际编程的平台
  - 集成开发工具
  - 熟练掌握各种基本操作
  - 结合提示信息，改正程序错误
  - 掌握基本的程序调试方法

重点知识回顾

- 过程性语言语法：
  - 数据类型（字符串）
  - 表达式和运算
  - 控制结构（顺序、分支、循环）
  - 函数（声明、调用）
  - 输入输出（**cin**、**cout**）
  - 数组
  - 指针
  - 引入库函数

重点知识回顾

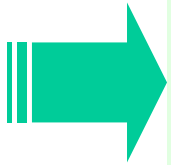
- 类和对象
  - 对象
  - 类
  - 成员属性和方法
  - 访问权限
  - 构造函数
  - 对象的使用

```
class Person{
    private:
        int id;
        int energy;
    public:
        Person();
        int exercise();
        int disease();
        int getEnergy();
};
```

重点知识回顾

- 友元函数

```
class Complex{
private:
    float real, image;
public:
    Complex(float r, float i);
    float getReal();
    float getImage();
    void Show();
};
Complex Add(Complex& c1,Complex& c2){
  float r=c1.getReal()+c2.getReal();
  float i=c1.getImage()+c2.getImage();
  return Complex(r,i);
}
```



```
class Complex{
private:
    float real, image;
public:
    Complex(float r, float i);
    float getReal();
    float getImage();
    void Show();
    friend Complex Add(Complex& c1,Complex& c2);
};
Complex Add(Complex& c1,Complex& c2){
    float r=c1.real+c2.real;
    float i=c1.image+c2.image;
    return Complex(r,i);
}
```

重点知识回顾

- 运算符重载

```
class Number{
private:
    float num;
public:
    Number(float n);
    void Show();
    friend Number operator+
    (Number& n1, Number& n2);
};
Number operator+(Number& n1, Number& n2){
    float n=n1.num+n2.num;
    return Number(n);
}
```

```
class Number{
private:
    float num;
public:
    Number(float n);
    void Show();
    Number operator+(Number& n2);
};
Number Number::operator+(Number& n2){
    float n=num+n2.num;
    return Number(n);
}
```

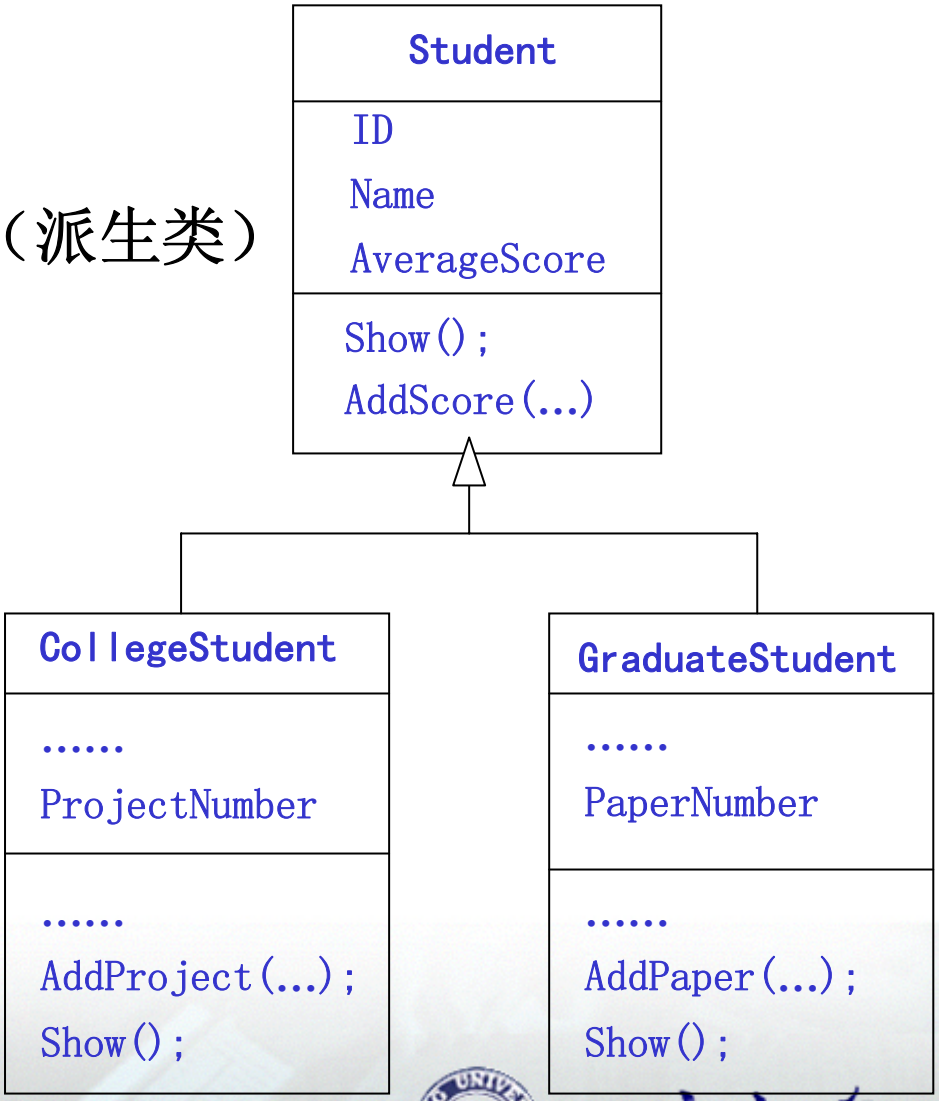
重点知识回顾

- 运算符重载——输入、输出

```
class Number{
    ......
    friend ostream& operator<<(ostream& o, Number& n);
    friend istream& operator>>(istream& i, Number& n);
}
ostream& operator<<(ostream& o, Number& n){
    o<<“The Number is ”<<n.num<<“.”<<endl;  return o;
}
istream& operator>>(istream& i, Number& n){
    cout<<“Please input the number: ”;
    i>>n.num;  return i;
}
main(){
    Number mynumber;
    cin>>mynumber;          cout<<mynumber;   }
```

重点知识回顾

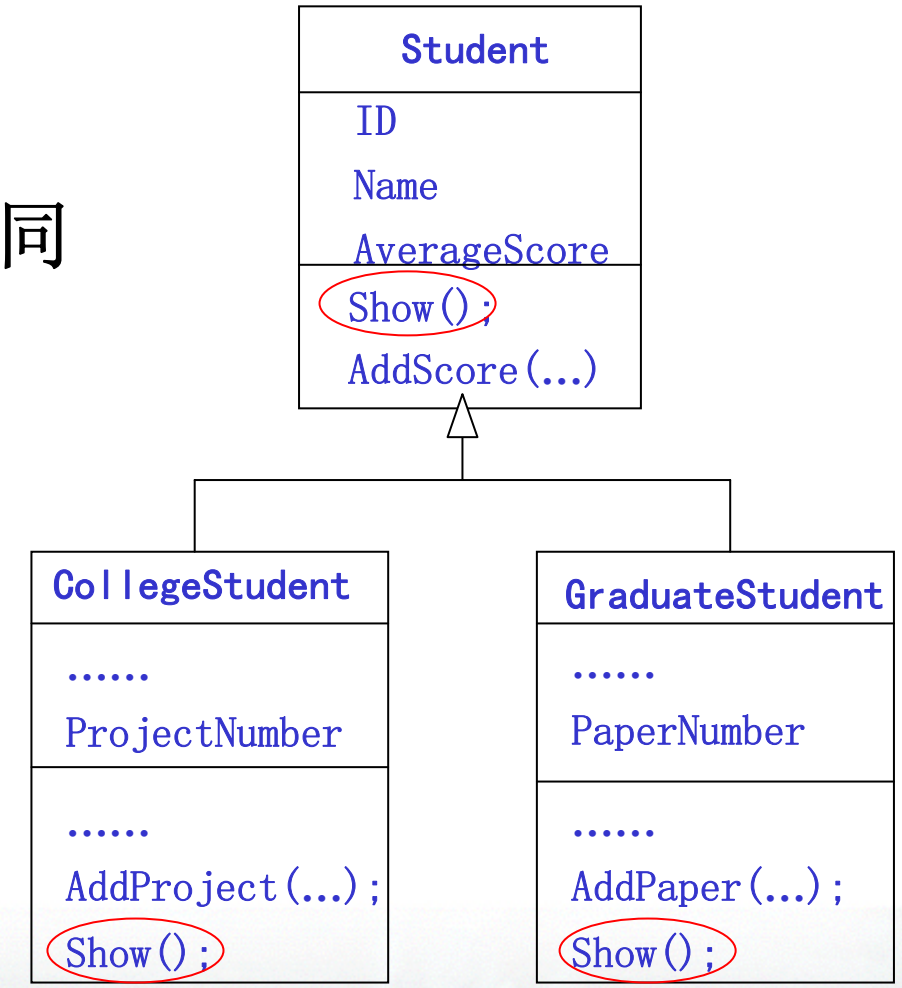
- 继承
  - 父类（基类）←→子类（派生类）
  - 访问权限
  - 构造函数
  - 函数重载
  - 构造类层次，
    合理安排属性和方法



重点知识回顾

- 虚函数和多态
  - 不同类型执行方法不同
  - 多态的条件：
    - 基类中声明**virtual**
    - 派生类中覆盖定义
    - 调用通过指针或引用

```
void ShowStudent(Student& stud)
{
    stud.Show();
}
```



编程环境

- 编程环境 **Microsoft Visual C++ 6.0**
  - 启动**VC**
  - 新建项目（**Win32 Console Application**）
  - 新建**.h**文件、**.cpp**文件
  - 编程
  - 编译、执行
  - 运行调试

例题讲解

- 题型特点
  - 题目明确说明需要构造的类及其主要属性、方法
  - 考查的知识点相对集中
  - 实践性强——实验考试的共同特点

例题讲解

- 解题思路
  - 审题：明确题目要求和考点
  - 设计：根据题目要求设计类及其属性、方法以及类之间的关系，考虑**main**函数的执行过程
  - 实现：面向对象概念、**C++**语法
  - 调试：编译、执行、调试
  - 测试：完成最终程序

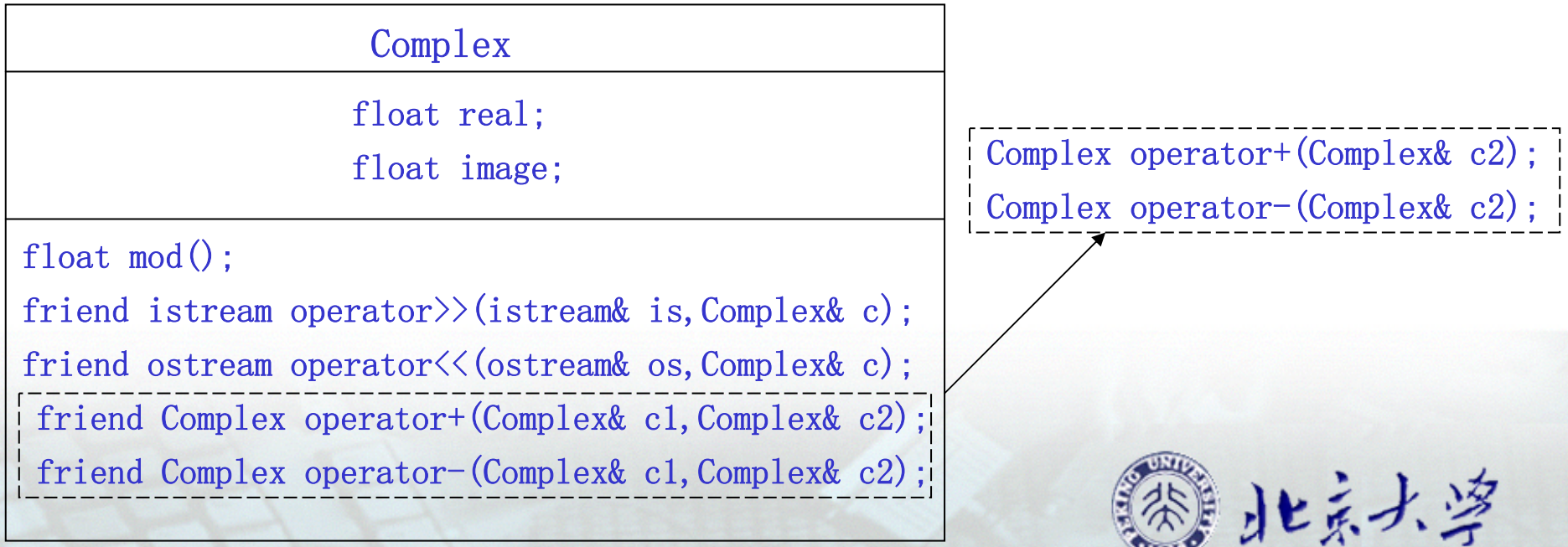
例题讲解1

- 实现一个复数类的程序，要求：
  1. 定义复数类**Complex**，其中包含实部**real**和虚部**image**；
  2. 在复数类中定义方法**mod**计算复数的模，即 $\sqrt{real^2 + image^2}$ 返回值为浮点数，
  3. 重载输入”>>”操作符，使得可以通过**cin**直接读入复数类的对象值；
  4. 重载输出”<<”操作符，使得可以通过**cout**直接输出复数类的对象值；
  5. 重载”+”、”-”运算符，使得复数类之间可以进行加减运算；
  6. 编写**main**函数，测试上述所要求的功能

北京大学

例题讲解1

- 审题：
  - 考查面向对象类的概念、属性和方法
  - 考查运算符重载<<、>>、+、-
  - 考查**main**函数中对象方法的调用
- 设计



例题讲解1

Complex类定义

complex.h头文件

```
class Complex
{
private:
    float real;
    float image;
public:
    Complex();
    Complex(float r, float i);
    float mod();

    friend istream& operator>>(istream& is, Complex& c);
    friend ostream& operator<<(ostream& os, Complex& c);

    friend Complex operator+(Complex& c1, Complex& c2);
    friend Complex operator-(Complex& c1, Complex& c2);
    //Complex operator+(Complex& c2);
    //Complex operator-(Complex& c2);
};
```

例题讲解1

Complex类的实现

构造函数

complex.cpp文件

```
#include<math.h>
#include"complex.h"
Complex::Complex()
{
    real=0;
    image=0;
}
Complex::Complex(float r, float i)
{
    real=r;
    image=i;
}

float Complex::mod()
{
    return sqrt(real*real+image*image);
}
```

例题讲解1

Complex类的实现

操作符重载

complex.cpp文件

```
istream& operator>>(istream& is, Complex& c){
    cout<<"Please input the real of Complex : ";
    is>>c.real;
    cout<<"Please input the image of Complex : ";
    is>>c.image;
    return is;
}
ostream& operator<<(ostream& os, Complex& c){
    if(c.real==0){
          if(c.image==0) os<<"0";
    }else
          os<<c.real;
    if(c.image==0)
          os<<endl;
    else if(c.image<0)
          os<<c.image<<"i"<<endl;
    else
          os<<"+"<<c.image<<"i"<<endl;
    return os;
}
```

```
else if(c.image<0)
  if(c.image==-1)
    os<<“-i”<<endl;
  else
    os<<c.image<<“i”<<endl;
else
  if(c.image==1)
    os<<“+i”<<endl;
  else
    os<<“+”<<c.image
        <<“i”<<endl;
```

例题讲解1

```
/**
 * 运算符+、-重载 实现一 友元函数
 */
Complex operator+(Complex& c1, Complex& c2)
{
    float r=c1.real+c2.real;
    float i=c1.image+c2.image;
    return Complex(r,i);
}
Complex operator-(Complex& c1, Complex& c2)
{
    float r=c1.real-c2.real;
    float i=c1.image-c2.image;
    return Complex(r,i);
}
```

```
/**
 * 运算符+、-重载 实现二 成员函数
 */
Complex Complex::operator +(Complex& c2)
{
    float r=this->real+c2.real;
    float i=this->image+c2.image;
    return Complex(r,i);
}
Complex Complex::operator -(Complex& c2)
{
    float r=this->real-c2.real;
    float i=this->image-c2.image;
    return Complex(r,i);
}
```

例题讲解1

Complex对象调用main函数设计

complex.cpp文件

```
void main(){
    Complex c1,c2;
    cout<<"Please input Complex c1:"<<endl;
    cin>>c1;
    cout<<"Your Complex c1="<<c1;
    cout<<“mod(c1)="<<c1.mod()<<endl;
    cout<<"Please input Complex c2:"<<endl;
    cin>>c2;
    cout<<"Your Complex c2="<<c2;
    cout<<“mod(c2)="<<c2.mod()<<endl;
    Complex c3=c1+c2;
    cout<<"The Complex c1+c2="<<c3;
    c3=c1-c2;
    cout<<"The Complex c1-c2="<<c3;
    cout<<"Thank you for using me."<<endl;
}
```

例题讲解1

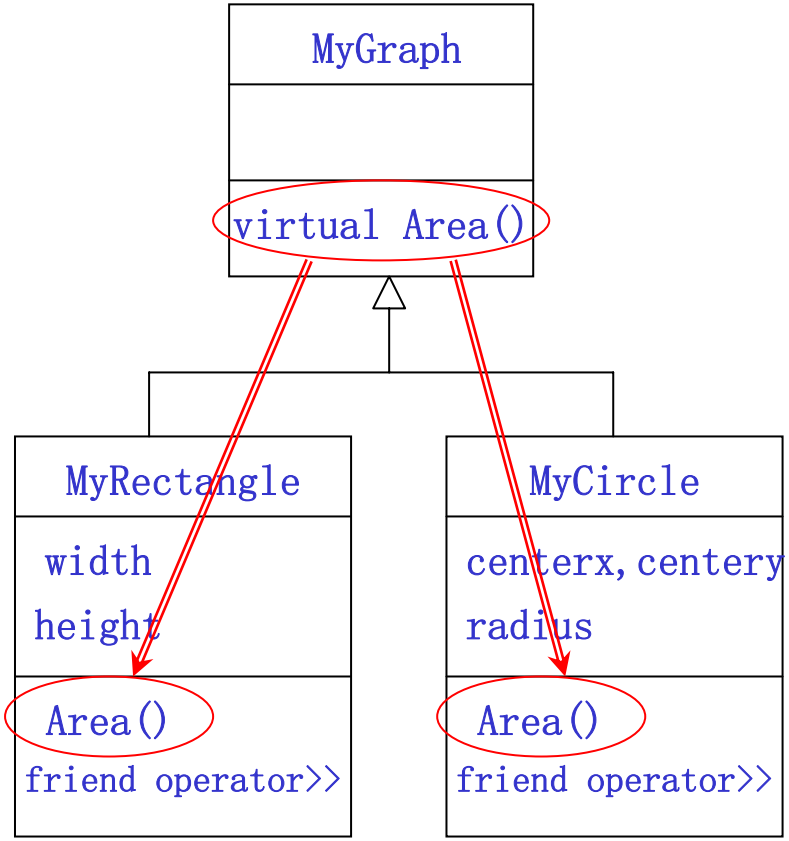
- 编译、执行、调试
  - 编译报错→定位错误改正
  - 执行错误→查看相关位置代码或者跟踪调试
- 总结：
  - 审题：明确题目要求，识别考查点
  - 设计：应用面向对象概念，设计程序中的类，同时考虑**main**函数中对象使用过程
  - 编程：符合面向对象概念和**C++**语法
  - 调试：根据结果修正程序

例题讲解2

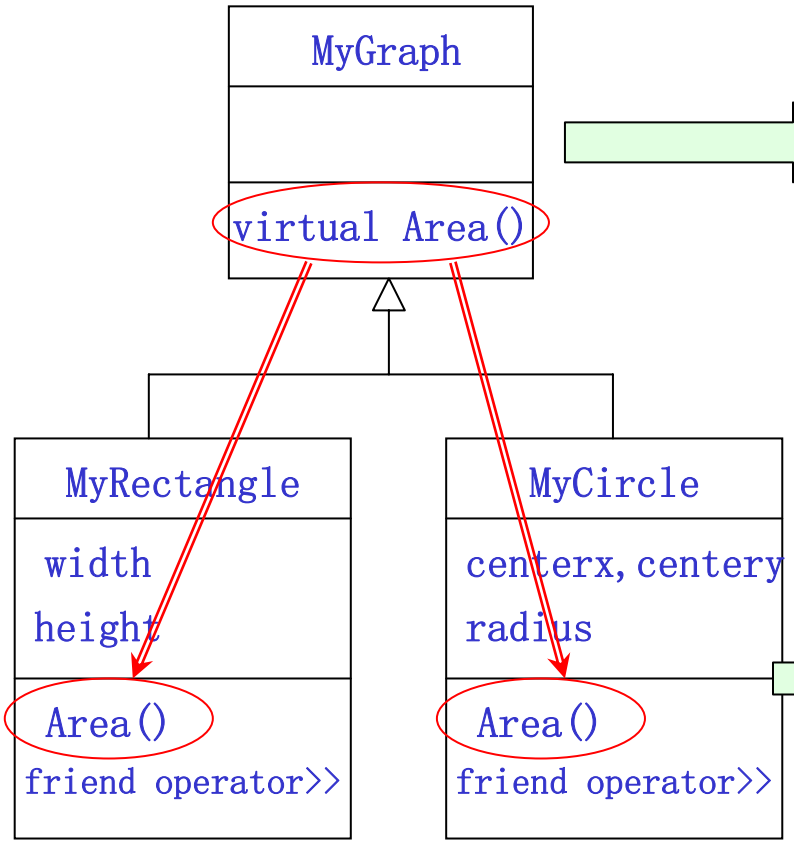
- 实现一个可以操作图形的程序，要求：
  - 定义基类**MyGraph**，至少包含纯虚函数**Area**，计算图形面积；
  - 从基类**MyGraph**中派生矩形类**MyRectangle**和圆形类**MyCircle**，其中矩形信息包括矩形的长和宽，圆形信息包括圆形的圆心和半径，具体实现上述纯虚函数**Area**，计算矩形和圆形的面积；
  - 重载输入”>>”操作符，使得可以通过**cin**输入矩形类和圆形类的对象值；
  - 在**main**函数中，体现面向对象的多态特性，即定义指向**MyGraph**的指针，根据用户选择输入矩形或者圆形的对象值，并且调用**Area**计算其面积。

例题讲解2

- 审题：
  - 考查类、属性和方法
  - 考查继承的概念
  - 考查运算符重载
  - 考查虚函数和多态
- 设计：
  - 虚函数的重载
  - 在类层次中构造多态
  - **main**函数中多态的体现



例题讲解2




```
class MyGraph
{
 public:
    MyGraph();
    virtual float Area()=0;
};
```

```
class MyCircle : public MyGraph
{
 private:
    float centerx,centery,radius;
public:
    MyCircle();
    MyCircle(float x,float y,float r);
    float Area();
    friend istream& operator>>
          (istream& is, MyCircle& circle);
};
```

例题讲解2

虚函数重载
(计算图形面积)
mygraph.cpp文件

在MyGraph类声明中有

```
    virtual float Area()=0;
```

定义纯虚函数

```
......
float MyRectangle::Area()
{
    return length * width;
}
float MyCircle::Area()
{
    return 3.1416 * radius * radius;
}
```

例题讲解2

运算符重载
(检查输入合法性)
mygraph.cpp文件

```
istream& operator>>(istream& is, MyRectangle& rect){
    float l,w;
    cout<<"请输入矩形的长: ";  is>>l;
    //判断输入的合法性
    while(l<=0){
          cout<<"非法输入：矩形的长必须是正数！"<<endl;
          cout<<"请输入矩形的长: ";  is>>l;
    }
    rect.length=l;
    cout<<"请输入矩形的宽: ";  is>>w;
    //判断输入的合法性
    while(w<=0){
          cout<<"非法输入：矩形的宽必须是正数！"<<endl;
          cout<<"请输入矩形的宽: ";  is>>w;
    }
    rect.width=w;
    return is;
}
```

例题讲解2

main函数过程实现
(界面友好、
多态的实现)
mygraph.cpp文件

```
void main(){
    MyGraph* pGraph;  MyRectangle rect;  MyCircle circle;  int choice;
    while(true){
        cout<<"该程序提供如下功能："<<endl;
        cout<<"  1. 输入矩形并计算面积；"<<endl;
        cout<<"  2. 输入圆形并计算面积；"<<endl;
        cout<<"  3. 退出。"<<endl;
        cout<<"请选择功能（1-3）："; cin>>choice;
        switch(choice){
         case 1: cin>>rect;  pGraph=&rect;  break;
         case 2: cin>>circle; pGraph=&circle; break;
         default:          break;
        }
        if(choice==1||choice==2) cout<<"图形面积为"<<pGraph->Area()<<endl;
        else  break;
    }
    cout<<"谢谢使用本系统！"<<endl;
}
```

例题讲解2

- 编译、执行、调试
- 总结

```
switch(choice){
 case 1: cin>>rect;
         rect.Area();
         break;
 case 2: cin>>circle;
         circle.Area();
         break;
 default: break;
}
```

不能体现多态性!

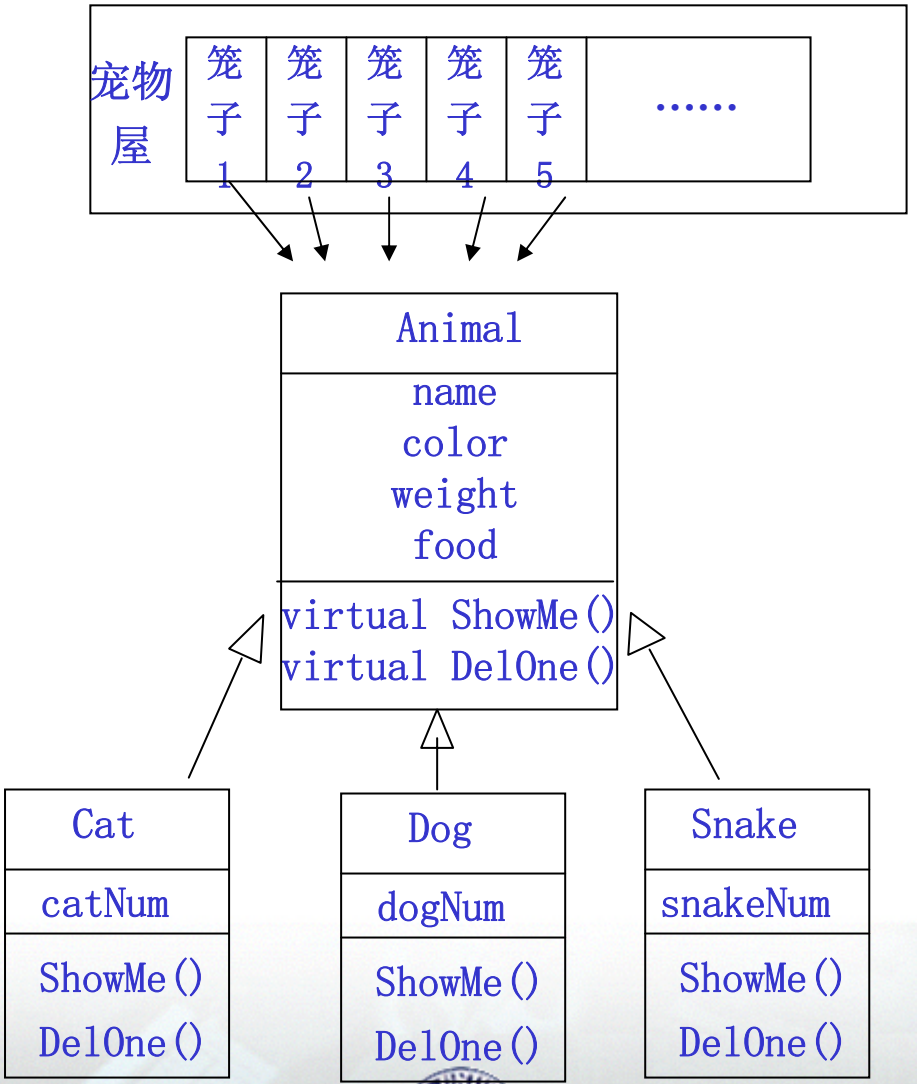
```
switch(choice){
 case 1: cin>>rect;
         pGraph=&rect;
         break;
 case 2: cin>>circle;
         pGraph=&circle;
         break;
 default: break;
}
if(choice==1||choice==2)
    cout<<"图形面积为"<<pGraph->Area()<<endl;
else
    break;
```

例题讲解3

- 宠物屋里有12个笼子，每个笼子可以放不同的动物，包括猫Cat、狗Dog和蛇Snake，但同一时刻一个笼子最多能放1只宠物，要求：
  - 实现一个简易的管理系统，可以增加、删除指定笼子中的宠物，查询每个笼子中存放宠物的情况（包括笼子为空的情况），统计宠物的种类和数量；
  - 定义描述宠物屋的类Shelves, 其中有12个笼子用于存放各种宠物；
  - 定义虚拟基类Animal，至少包括纯虚函数ShowMe；
  - 定义派生类Cat、Dog和Snake，具体实现上述纯虚函数ShowMe，显示该宠物的名称、颜色、体重和喜爱的食物；
  - 重载输入“>>”操作符，使得可以通过cin直接读入宠物颜色、体重和喜爱的食物；
  - 编写main函数，测试上述所要求的各种功能。

例题讲解3

- 审题：
  - 考查类、属性、方法
  - 考查继承的概念
  - 考查虚函数和多态
  - 考查运算符重载
  - 考查整体-部分关系
- 设计
  - 整体类**Shelves**
  - 部分类 类层次 **Animal->Cat/Dog/Snake**
  - 设计**main**函数功能界面



例题讲解3

- **main函数中的功能界面：**

  宠物管理：
  - 添加宠物 → 处理流程
    1）输入笼子编号
    2）选择宠物种类
    3）根据宠物种类调用>>输入宠物信息
    4）将该宠物加到指定宠物笼中
  - 删除宠物
  - 查询宠物
  - 统计宠物
  - 退出

- 设计类方法
  - **int Add( ) ? or int Add(int n, Animal* animal) ?**
  - **int Delete( ) ? or int Delete(int n) ?**
  - **int Query( ) ? or int Query(int n) ?**

例题讲解3

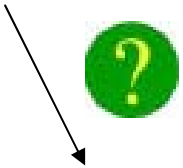
数据类声明
(静态成员的使用)
Animal.h文件

```
class Animal{
 protected:
    char* name; char* color; float weight; char* food;
 public:
    Animal();
    virtual void ShowMe()=0;
    virtual void DelOne()=0;
};
class Cat : public Animal{
 private:
    static int catNum;
 public:
    Cat();
    void ShowMe();
    void DelOne();
    static int getCatNum();
    friend istream& operator>>(istream& is, Cat& cat);
};
```

例题讲解3

管理类声明
Shelves.h文件

```
class Shelves{
 private:
    Animal* shelves[12];
 public:
    Shelves();
    int Add(int n,Animal* newAnimal);
    int Delete(int n);
    int Query(int n);
    int Stat();
};
```



```
int Add();
int Delete();
int Query();
```

北京大学

例题讲解3

数据类实现
animalsys.cpp文件

```
int Cat::catNum=0;
Cat::Cat(){  catNum++; }
int Cat::getCatNum(){  return catNum; }
void Cat::DelOne(){  catNum--; }
void Cat::ShowMe(){
    cout<<"我是一只猫——"<<endl;
    cout<<"  名字："<<name<<endl<<"  颜色："<<color<<endl;
        <<"  体重："<<weight<<endl<<"  食物："<<food<<endl<<endl;
}
istream& operator>>(istream& is, Cat& cat){
    cout<<"您正在录入猫的信息："<<endl;
    cout<<"     名称："; is>>cat.name;
    cout<<"     颜色："; is>>cat.color;
    cout<<"     体重："; is>>cat.weight;
    cout<<"     喜爱的食物："; is>>cat.food;
    return is;
}
```

北京大学

例题讲解3

管理类实现
(检查执行条件)
animalsys.cpp文件

```
Shelves::Shelves(){
    for(int i=0;i<12;i++)   shelves[i]=NULL;
}
int Shelves::Add(int n, Animal* newAnimal){
    if(shelves[n]!=NULL){
          cout<<"您要放入的笼子已满，请重试"<<endl<<endl;
          return 0; }
    shelves[n]=newAnimal;
    cout<<"宠物已经成功放入笼子"<<endl<<endl;
    return 1;
}
int Shelves::Delete(int n){
    if(shelves[n]==NULL){
          cout<<"您要清空的笼子里没有宠物"<<endl<<endl;
          return 0;  }
    shelves[n]->DelOne();
    shelves[n]=NULL;
    cout<<"指定笼子已经成功清空"<<endl<<endl;
    return 1;
}
```

例题讲解3

管理类实现
(多态实现
静态方法调用)
animalsys.cpp文件

```
int Shelves::Query(int n){
    if(shelves[n]==NULL){
        cout<<"您要查询的笼子里没有宠物"<<endl<<endl;
        return 0;
    }
    shelves[n]->ShowMe();
    return 1;
}
int Shelves::Stat(){
    cout<<"12只笼子中有猫"<<Cat::getCatNum()<<"只，";
    cout<<"狗"<<Dog::getDogNum()<<"只，";
    cout<<"蛇"<<Snake::getSnakeNum()<<"只。"<<endl<<endl;
    return 1;
}
```

例题讲解3

main函数实现
(调用管理类方法)
animalsys.cpp文件

```
void main(){
    int choice=1; int n;
    Shelves shelves; Cat* cat; Dog* dog; Snake* snake;
    while(choice!=5){
        cout<<"系统提供如下功能："<<endl;
        cout<<"    1. 添加宠物；"<<"    2. 删除宠物；"<<endl;
        cout<<"    3. 查询宠物；"<<"    4. 统计宠物；"<<endl;
        cout<<"    5. 退出。"<<"请选择功能（1-5）："; cin>>choice;
        switch(choice){
         case 1: ......//见下页
         case 2: cout<<"请选择要删除的笼子编号（1-12）："; cin>>n;
                 shelves.Delete(n-1); break;
         case 3: cout<<"请选择要查询的笼子编号（1-12）："; cin>>n;
                 shelves.Query(n-1);  break;
         case 4: shelves.Stat();   break;
         case 5: break;
         default: cout<<"输入错误。"<<endl<<endl;    }
    }
    cout<<"感谢使用本系统！"<<endl;
}
```

北京大学

例题讲解3

main函数实现
(调用管理类方法)
animalsys.cpp文件

```
//处理Add添加宠物的情况
char animal;
cout<<"请选择要添加的笼子编号（1-12）："; cin>>n;
cout<<"请选择要添加的宠物种类 c(猫)/d(狗)/s(蛇)：";cin>>animal;
switch(animal){
 case 'c': cat=new Cat;     cin>>*cat;
   if(!shelves.Add(n-1,cat))        {
         cat->DelOne();
         delete cat;
   } break;
 case 'd': dog=new Dog;     cin>>*dog;
   if(!shelves.Add(n-1,dog))        {
         dog->DelOne();
         delete dog;
   } break;
 case 's': //省略
 default: cout<<"输入错误。"<<endl<<endl;
}
break;
```

例题讲解3

- 另一种设计和实现：

```
class Shelves{
    . . . . . .
    int Add();
    int Delete();
    int Query();
}
```

1）要求用户输入笼子编号；
2）查看笼子是否已满；
3）要求用户选择宠物种类；
4）根据宠物种类创建宠物对象；
5）将宠物放到指定笼子中

1）要求用户输入笼子编号；
2）检查笼子是否为空；
3）将指定笼子清空

1）要求用户输入笼子编号；
2）检查笼子是否为空；
3）调用pAnimal->ShowMe()显示信息

```
case 1: shelves.Add(); break;
case 2: shelves.Delete(); break;
case 3: shelves.Query(); break;
case 4: shelves.Stat(); break;
case 5: break;
```

例题讲解3

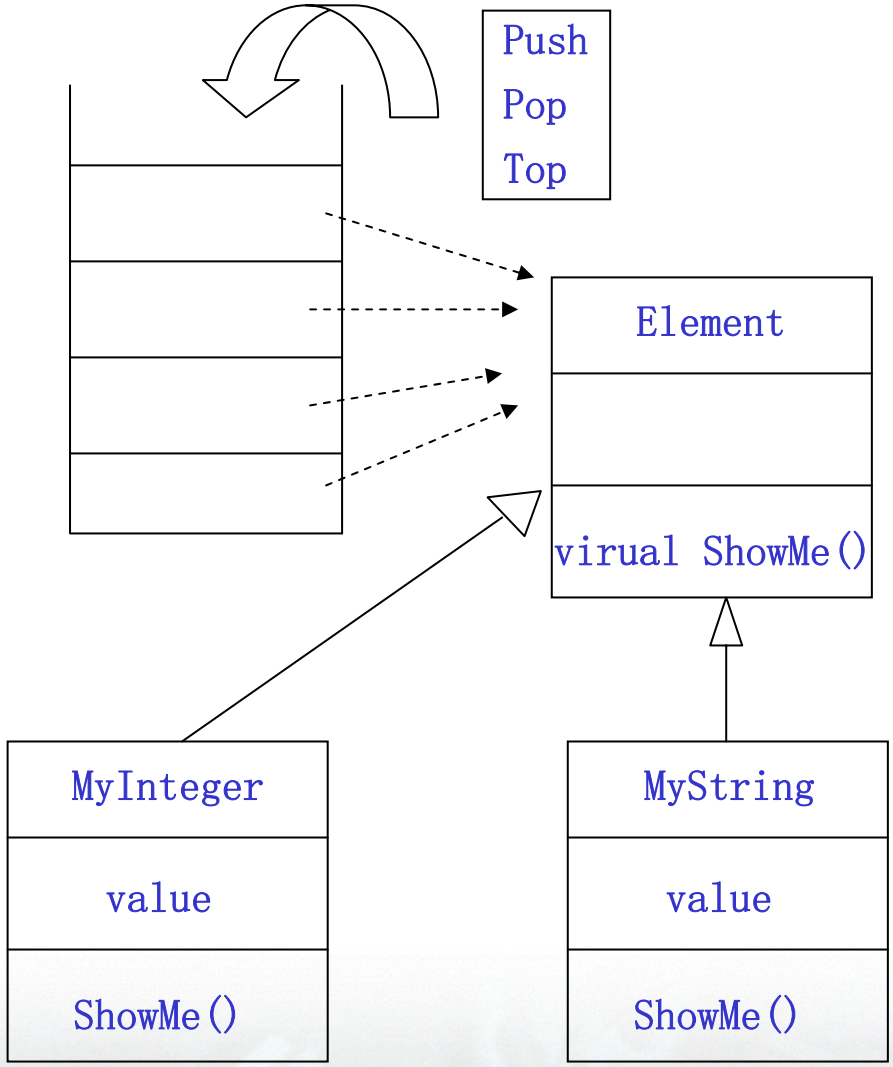
- 总结：
  - 利用继承和虚函数实现多态
  - 管理类和数据类的整理-部分关系
  - 操作界面设计
  - 接口的设计和交互问题归属
  - 综合考查类、继承、多态、运算符重载、指针数组、算法设计等知识掌握程度

例题讲解4

- 请实现一个栈，既可以存放整数，又可以存放字符串。简单的说，栈是一种数据结构，按照后进先出的顺序管理进、出栈的元素。本题要求完成：
  - 实现描述栈的类**Stack**，其中定义了栈的实际大小**Size**，并包括进栈函数**Push**，出栈函数**Pop**和显示栈顶元素的函数**Top**
  - 定义基类**Element**，至少包含纯虚函数**ShowMe**
  - 从基类**Element**中派生整数类**MyInteger**和字符串类**MyString**，具体实现上述纯虚函数**ShowMe**，显示该元素的类型和相应的值
  - 重载输入>>操作符，使可以通过**cin**读入整数和字符串
  - 编写**main**函数，测试以上各种功能，栈中元素是整数或字符串

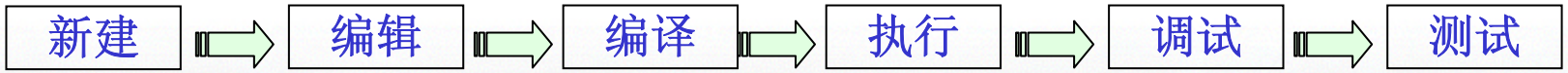
例题讲解4

- 审题
  - 考查类、属性方法
  - 考查继承
  - 考查虚函数和多态
  - 考查组合类
  - 考查>>重载
- 设计
  - 设计类层次
  - 设计实现多态
  - 设计组合类
  - 设计main功能界面



VC编程和调试

- 常用功能：
  - 新建 **project**、**.h file**、**.cpp file**
  - 编辑 **Copy**、**Cut**、**Paste**
  - 创建 **Compile**、**Build**、**Execute**
  - 调试 **Step Into**、**Step Over**、**Step Out**、**Breakpoint**、**AddWatch**
- 熟悉开发流程：



北京大学

VC编程和调试

- 如何面对错误？
  - **Compile报错→语法错误，改正 (Easy)**

    常见语法错误：
    - 关键字、变量名大小写敏感
    - 强类型语言
    - 类的声明末尾需要加分号；

      class A{ . . . . . };
    - **#include** 预处理命令的使用
    - 小心中文逗号陷阱，**VS** ，
    - 大括号 **{ }** 不匹配
    - 类成员的访问权限

VC编程和调试

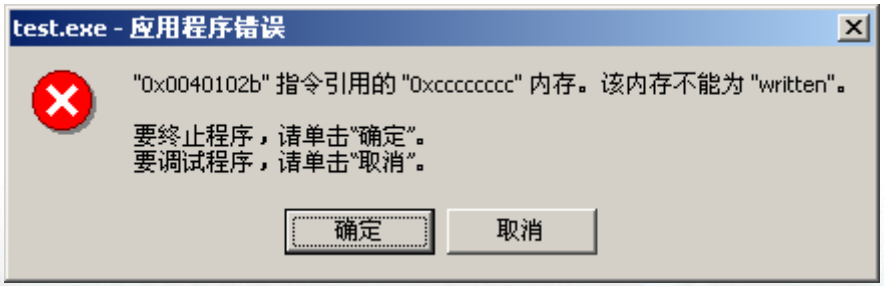
- 常见报错提示：
  - 'cout' : undeclared identifier (#include<iostream.h>)
  - 'var' : undeclared identifier(变量需要先声明再使用)
  - unexpected 'class' 'MyRect' ； syntax error : missing ';' before ':' (class声明后丢失;分号)
  - binary '<<' : no operator defined which takes a right-hand operand of type 'class MyGraph' (or there is no acceptable conversion) (操作符<<重载问题)
  - unexpected end of file found (大括号{ }不匹配)
  - 'color' : cannot access private member declared in class 'MyGraph' (访问权限错误)
  - function call missing argument list (方法调用缺少参数)

VC编程和调试

- 如何面对错误？
  - 无法创建或执行→
    常见原因
    - 类似“**unresolved external symbol _WinMain@16**”等，很可能建立的项目属性不对
    - 类似“**cannot open Debug/xxx.exe for writing**”等，说明**exe**正在使用无法重新生成
  - 对于以上报错的情况，需要
    - 明白错误的含义
    - 定位错误，可能在指定行的前后

VC编程和调试

- 如何面对错误？
  - 运行时出现非法操作→很可能指针操作不当
    常见原因：
    - 指针变量没有初始化，如
      字符串**char* name;　　strcpy(name,”Tom”);**
    - 指针数组的使用要小心
    - 指针变量已经**delete**再直接使用



VC编程和调试

- 如何面对错误？
  - 运行时出现错误结果→调试，找到问题所在
  - 调试的基本步骤：
    - 重现错误，找到出错规律
    - 根据错误结果判断大致错误位置
    - 利用**VC**的调试手段
      断点 **Breakpoint**
      单步跟踪 **Step Into**、**Step Out**、**Step Over**
      查看变量值 **AddWatch**

VC编程和调试

- 如何发现错误？
  - 测试可以保证程序的正确性和健壮性
  - 常用方法：
    - 测试程序提供的各种功能
    - 选择边缘输入
    - 尝试非法输入
- 调试、测试举例

常见问题总结

- 类成员的访问权限
- 继承时的构造函数、函数重载
- 虚函数和多态的正确使用
- 运算符重载（+、>>）的实现方法
- 组合类的管理
- 功能界面的设计
- 如何面对错误
- 常见的语法错误
- 常用的调试手段

评分标准

- 基本要求（及格要求）：
  - 使用面向对象的概念编程；
  - 体现题目规定的知识点，例如继承、重载、多态等；
  - 掌握简单的数据结构和算法，并具备简单的算法设计能力；
  - 能够正确通过编译，能够运行，而且能够得到正确的运算结果；
  - 虽然没有通过编译，但程序编写基本正确，视情况也有可能及格。
- 良好要求：
  - 达到及格的所有要求；
  - 程序风格良好；
  - 运行结果界面友好；
  - 检查必要的输入非法的错误并给出提示，尤其是容易导致程序死机的非法输入；

预祝大家取得好成绩

Questions & Comments

北京大学