

## 第9讲

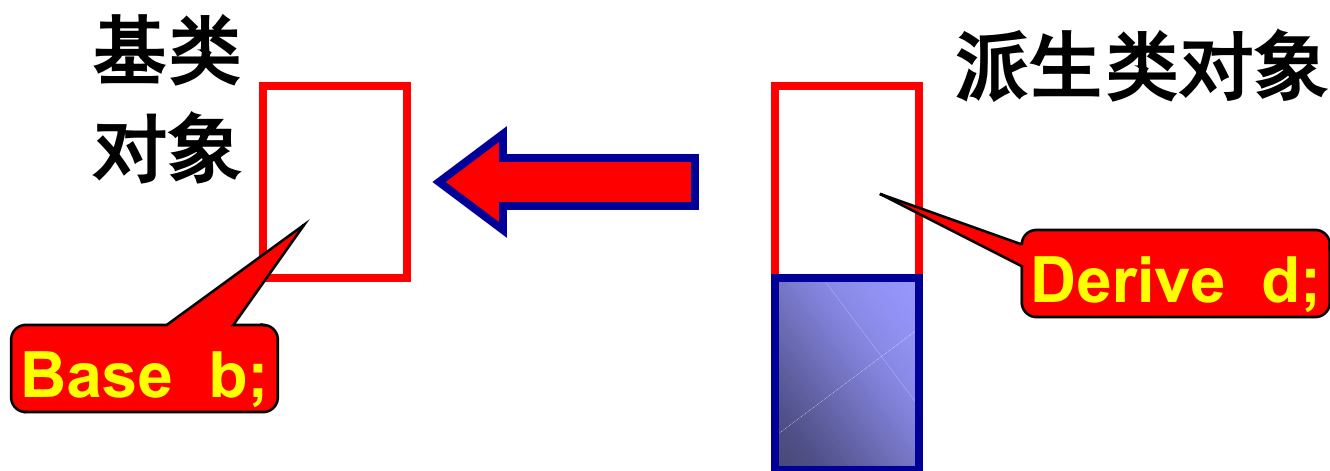
# 多态 ——虚函数

主讲人：赵文彬



# 赋值兼容规则

相互之间能否赋值？



可以将派生类对象的值赋给基类对象。

$b = d;$  只是将从基类继承来的成员赋值。  
反之不行

# 赋值兼容规则

Dot的对象空间

x
y
Dot(x,y)
Dot(&dot)
Show()



Line的对象空间

```
Dot dot;
Line line;
dot=line;
line=dot;
```

非法

x	
y	
Dot(x,y)	
Dot(&dot)	
Show()	
d1	x
	y
	Dot(x,y)
	Dot(&dot)
	Show()
d2	x
	y
	Dot(x,y)
	Dot(&dot)
	Show()
Line()	
Showl()	

从基类继承

基类对象

# 赋值兼容规则

- 可以将一个派生类对象的地址赋给基类的指针变量。

`Base *basep;`

基类指针

`basep=&b;`

基类对象

`basep = &d`

派生类对象

基类对象

`basep`

`Base b;`

`basep`只能引用  
从基类继承来  
的成员。

派生类对象

`basep`

`Derive d;`

# 赋值兼容规则

- 派生类对象可以初始化基类的引用。

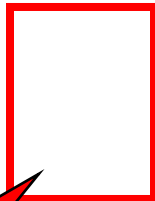
**Derive d;** **派生类对象**

**Base &basei=d;**

**基类引用**

basei只能引用  
从基类继承来  
的成员。

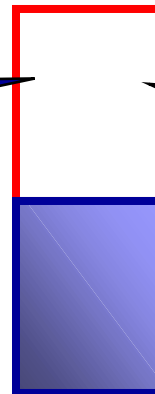
基类对象



**Base b;**

派生类对象

**别名basei**



**Derive d;**

```

class A{
public: int x;
      A(int a=0) { x=a;}
};
class B{
public: int y;
      B(int a=0) { y=a;}
};
class C:public A,public B{   int z;
public: C(int a,int b,int m): A(a), B(b) { z=m; }
      void Show() {   cout<<"x="<<x<<"\t";   cout<<"y="<<y<<"\t";
      cout<<"z="<<z<<"\n";   }
};
void main(void)
{
    A  a1(100);      B  b1(200);      C  c1(10,20,50);
    cout<<"a1.x="<<a1.x<<endl;      cout<<"b1.y="<<b1.y<<endl;
    c1.Show();      a1=c1;  b1=c1; //派生类对象向基类对象赋值
    cout<<"a1.x="<<a1.x<<endl;      cout<<"b1.y="<<b1.y<<endl;
    A *p;  p=&c1; //基类指针指向派生类对象  p->Show();
}
    
```

```

a1.x=100
b1.y=200
x=10  y=20  z=50
a1.x=10
b1.y=20
请按任意键继续...
    
```

错误!

# 类型兼容

- **里氏代换原则**：任何父类可以出现的地方，都可以将父类替换为子类。
- 子类对象可以赋值给父类对象。
- 子类对象可以初始化父类的引用。
- 子类对象的地址可以赋给指向父类的指针。



# 类型兼容

```
#include <iostream>
using namespace std;
```

```
class CPolygon {
protected:
    int width, height;

public:
    void set_values (int a, int b)
    { width=a; height=b; }
};
```

```
class CRectangle: public CPolygon {
public:
    int area ()
    { return (width * height); }
};
```

```
class CTriangle: public CPolygon {
public:
    int area ()
    { return (width * height / 2); }
};
```

## 内容

```
int main () {  
    CRectangle rect;  
    CTriangle trgl;  
  
    CPolygon * ppoly1 = &rect;  
    CPolygon * ppoly2 = &trgl;  
  
    ppoly1->set_values (4,5);  
    ppoly2->set_values (4,5);  
  
    cout << rect.area() << endl;  
    cout << trgl.area() << endl;  
    return 0;  
}
```

```
C:\Windows\system32\cmd.exe
```

```
20
```

```
10
```

```
Press any key to continue . . .
```

ppoly1或ppoly2可以调用area() 吗?



# 虚函数

- 父类中定义的**虚函数**为其子类提供了一个**通用**的**框架**，**说明了其子类应该具有这个行为**，但是**具体的实现功能是在子类中完成的**。

## 定义格式

```
virtual 函数类型 函数名 (形参表)
{
    函数体
}
```

# 虚函数

```
#include <iostream>
using namespace std;
class CPolygon {
protected:
    int width, height;
public:
    void set_values (int a, int b)
    { width=a; height=b; }
    virtual int area ()
    { return (0); }
};
```

# 虚函数

```
class CRectangle: public CPolygon {  
public:  
    int area ()  
    { return (width * height); }  
};  
class CTriangle: public CPolygon {  
public:  
    int area ()  
    { return (width * height / 2); }  
};
```

# 虚函数

```
int main () {  
    CRectangle rect;  
    CTriangle trgl;  
    CPolygon poly;  
    CPolygon * ppoly;  
    ppoly = &rect;  
    ppoly->set_values (4,5);  
    cout << ppoly->area() << endl;  
    ppoly = &trgl;  
    ppoly->set_values (4,5);  
    cout << ppoly->area() << endl;  
    return 0;  
}
```

```
C:\Windows\system32\cmd.exe  
20  
10  
Press any key to continue
```



# 纯虚函数

- **纯虚函数(pure virtual function)**: 是一个在**基类中说明的虚函数**，它在**该基类中没有定义具体实现**，**要求各派生类根据实际需要定义函数实现**。

```
virtual 函数类型 函数名(参数表)=0;
```

# 纯虚函数

```
#include <iostream>
using namespace std;
class CPolygon {
protected:
    int width, height;
public:
    void set_values (int a, int
b)
    { width=a; height=b; }
    virtual int area () =0;
};
```

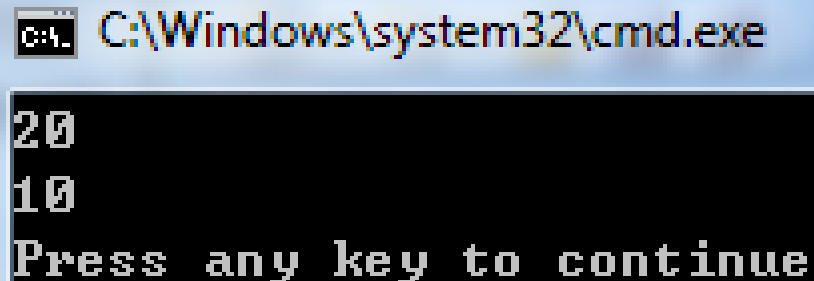
```
class CRectangle: public CPolygon {
public:
    int area ()
    { return (width * height); }
};
```

```
class CTriangle: public CPolygon {
public:
    int area ()
    { return (width * height / 2); }
};
```



# 纯虚函数

```
int main () {  
    CRectangle rect;  
    CTriangle trgl;  
    CPolygon poly;  
    CPolygon * ppoly;  
    ppoly = &rect;  
    ppoly->set_values (4,5);  
    cout << ppoly->area() << endl;  
    ppoly = &trgl;  
    ppoly->set_values (4,5);  
    cout << ppoly->area() << endl;  
    return 0;  
}
```



C:\Windows\system32\cmd.exe

20  
10  
Press any key to continue

# 抽象类

- 如果一个类中至少具有一个纯虚函数，则该类称为**抽象类**。
- **抽象类只能作为其它类的基类使用**，**抽象类不能定义对象**，纯虚函数的实现由派生类给出；
- 派生类仍可不给出所有基类中纯虚函数的定义，继续作为抽象类；如果派生类给出所有纯虚函数的实现，派生类就不再是抽象类而是一个具体类，就可以定义对象。

# 抽象类

- 如果一个类中至少具有一个纯虚函数，则该  
类称为**抽象类**。
  - 抽象类不能用作参数类型、函数返回值或强制类型转换。
  - 可以**定义一个抽象类的指针和引用**。通过抽象类的指针和引用，可以指向并访问各派生类成员，这种访问是具有**多态特征**的。

## 小结

- 兼容赋值原则

- 虚函数

- 纯虚函数

- 抽象类

- 作业

- 定义**动物类**，并派生**鱼类**、**鸟类**和**兽类**。要求动物类包含动物名字，编号，年龄，体重4个属性，各类中除构造函数外，还需要有用于设置描述动物特征的函数，用**虚函数**实现**多态**。