燕山大學
YANSHAN UNIVERSITY

C++面向对象程序设计
实验指导书

（7）多态性

燕山大学软件工程系

目　录

实验 07 多态性

1.1 时间安排 2 学时

本实验安排 2 个实验课时。

1.2 实验目的和要求

1、 掌握将运算符重载为成员函数与非成员函数的区别。
2、 掌握静态编联与动态联编的概念、区别及实现方式。
2、 掌握利用虚函数实现动态多态的方法。
3、 掌握利用纯虚函数与抽象类实现动态多态的方法。

1.3 实验内容 I (调试、理解、体会、掌握)

（1）定义 Cls 类；将运算符 “++”、“--”重载为 Cls 类的成员函数，将运算符“<<”重载为类的非成员函数；在 main 中对类进行实现。观察运行结果，理解多态。

```
#include<iostream>
using namespace std;
class Cls
{
public:
        Cls(double _x=0);
        Cls(const Cls &a);
        Cls &operator=(const Cls &a);
        Cls operator+(const Cls &op2);
        Cls operator-(const Cls &op2);
        Cls &operator++();
        Cls operator++(int);
        Cls &operator--();
        Cls operator--(int);
private:
         int *px;
friend ostream & operator<<(ostream &os, Cls s);
friend istream & operator>>(istream &is, Cls &s);
};
//构造函数
Cls::Cls(double _x)
{
        px=new int;
        *px=_x;
        cout<<"构造函数"<<endl;
}
//复制构造函数
Cls::Cls(const Cls &a)
{
        px=new int;
        *px=*(a.px);
        cout<<"复制构造函数"<<endl;

}
```

```
//重载赋值符
Cls &Cls::operator=(const Cls &a)
{
	if(&a!=this)
	{
		*px=*(a.px);
	}
	cout<<"重载赋值符="<<endl;
	return *this;
}
Cls Cls::operator+(const Cls &op2)
{
	cout<<"重载+"<<endl;
	return Cls(*px+*(op2.px));
}
Cls Cls::operator-(const Cls &op2)
{
	cout<<"重载-"<<endl;
	return Cls(*px-*(op2.px));
}
//重载前置“++”
Cls &Cls::operator++()
{
	(*px)++;
	cout<<"重载+ +(前置)"<<endl;
	return *this;
}
//重载后置“++”
Cls Cls::operator++(int)
{
	Cls old =*this;//调用复制构造函数
	(*px)++;
	cout<<"重载+ +（后置）"<<endl;
	return old;
}
//重载前置“--”
Cls &Cls::operator--()
{
	(*px)--;
	cout<<"重载- -（前置）"<<endl;
	return *this;
}
//重载后置“--”
Cls Cls::operator--(int)
{
	Cls old=*this;
	(*px)--;
	cout<<"重载- -（后置）"<<endl;
return old;
}
//重载“<<”符号，注意第二个参数不能是引用
ostream & operator<<(ostream &os, Cls s)
{
	os<<*(s.px);
	return os;
}
//重载“>>”符号
```

```
istream & operator>>(istream &is, Cls &s)
{
        is>>*(s.px);
        return is;
}

int main(int argc ,char **argv)
{
        Cls a,b(1.0),c(2.0) ;//a:用默认值调用构造函数，b、c 调用构造函数
        Cls d(b),e=c;//拷贝构造函数
        //cout<<a<<b<<c;
//      cout<<"1 a:"<<a<<" b:"<<b<<" c:"<<c<<" d:"<<d<<" e:"<<e<<endl;
//      a=b;//调用赋值符
//      cout<<"2 a:"<<a<<" b:"<<b<<endl;
//      a=b+c;
//      cout<<"3 a:"<<a<<" b:"<<b<<" c:"<<c<<endl;
//      a=b-c;
//      cout<<"4 a:"<<a<<" b:"<<b<<" c:"<<c<<endl;
        a=b++;
        cout<<"5 a:"<<a<<" b:"<<b<<endl;
        a=++b;
        cout<<"6 a:"<<a<<" b:"<<b<<endl;
        a=b--;
        cout<<"7 a:"<<a<<" b:"<<b<<endl;
        a=--b;
        cout<<"8 a:"<<a<<" b:"<<b<<endl;
        return 0;
}
```

（2）找到“实验指导书 06 类的继承与派生”中 1.3 实验内容 I 中的第(1)题(公司雇员)的代码，将公司雇员 Employee 类中的 pay 函数声明为虚函数，即

```
float pay(float hoursWorked) const;
```

↓

```
virtual float pay(float hoursWorked) const;
```

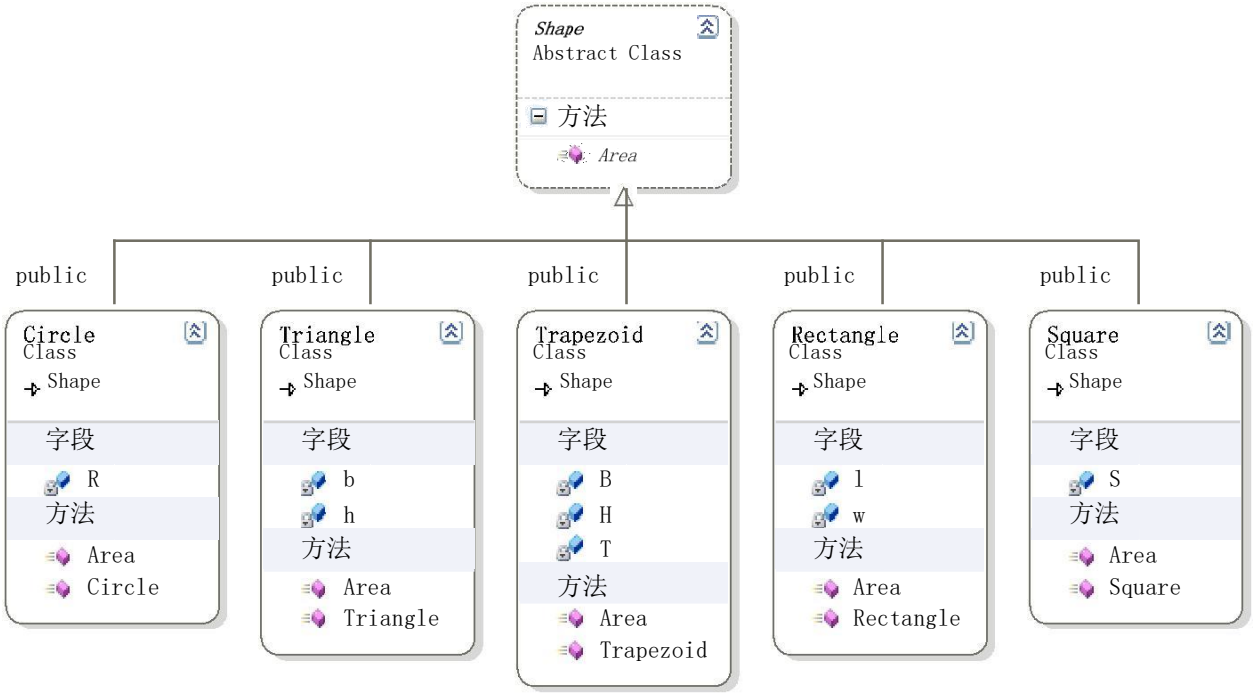
程序的其他部分不动，再观察其运行结果：可以发现三个 Display 函数均调用了各自对象的 pay 函数。注：虚函数实现的基础是类型兼容原则，属于动态绑定。

（3）用 c++ 定义一个车(Vehicle)基类，有 Run，Stop 等成员函数，由此派生出自行车(bicycle)类，汽车(motorcar)类，从 bicycle 和 motorcar 派生出摩托车(motorcycle)类，他们都有 Run，Stop 等成员函数。编写完整并用主函数测试。体会虚函数的作用。

```
#include<iostream>

using namespace std;

class Vehicle{

public:
```

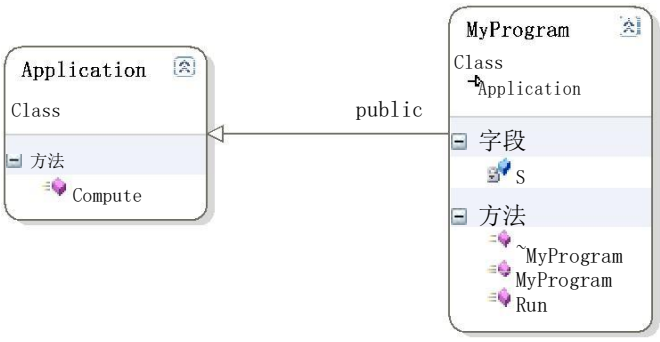
```
    virtual void Run(){cout<<"Vehicle::Run called\n";}
    virtual void Stop(){cout<<"Vehicle::Stop called\n";}

};
class motorcar:public Vehicle{
public:
    void Run(){cout<<"motorcar::Run called\n";}
    virtual void Stop(){cout<<"motorcar::Stop called\n";}
};
class bicycle :public Vehicle{
public:
     virtual void Run(){cout<<"bicycle::Run called\n";}
     void Stop(){cout<<"bicycle::Stop called\n";}
     };
class motorcycle:public bicycle,public motorcar{
public:
     void Run(){cout<<"motorcycle::Run called\n";}
     void Stop(){cout<<"mototrcycle::Stop called\n";}
};
void main(){
     Vehicle ve;
     bicycle bi;
     motorcar mo;
     motorcycle mocy;
     Vehicle *pclass=&ve;
     pclass->Run();
     pclass->Stop();
     pclass=&bi;
     pclass->Run();
     pclass->Stop();
     pclass=&mo;
     pclass->Run();
     pclass->Stop();
}
```

（4）一个计算某些几何图形的面积之和的例子，以说明纯虚函数及抽象类的应用。




```
# include   <iostream>
using  namespace  std;
class Shape //抽象类
{
public:
      virtual double Area() const=0;   //纯虚函数
};
//三角形 Triangle 类
class Triangle: public Shape
{
      public:
            Triangle(double bottom,double height)
            { b=bottom; h=height; }
            double Area() const
            { return 0.5*b*h; }
      private:
            double b,h;
};
//矩形 Rectangle 类
class Rectangle: public Shape
{
      public:
            Rectangle(double length,double width)
            { l=length; w=width; }
            double Area() const
            {  return l*w;  }
      private:
```

```
        double l,w;
};
//圆 Circle 类
class Circle: public Shape
{
    public:
        Circle(double r)
        { R=r; } double
        Area() const
        { return 3.14*R*R; }
    private:
        double R;
};
//梯形 Trapezoid 类
class Trapezoid: public Shape
{
  public:
Trapezoid(double t, double b, double h)
    { T=t; B=b; H=h; }
    double Area() const
    { return 0.5*(T+B)*H; }
  private:
    double T,B,H;
};
//梯形 Square 类
class Square: public Shape
{
    public:
        Square(double s)
        {   S=s;    }
        double Area() const
        { return S*S; }
    private: double S;
};

class Application
{
    public:
        double Compute(Shape *s [ ],int n) const;
};
```



```
double Application::Compute(Shape *s [ ], int n) const
{ //求数组中各对象的面积和
      double sum=0;
      for(int i=0; i<n; i++)
            sum+=s[i]->Area(); //对各不同类的对象使用统一的接口
      return sum;
}
class MyProgram : public Application
{
public:
      MyProgram();
      ~MyProgram();
      double Run();
 private:
      Shape **S;       //二级指针
};
MyProgram::MyProgram()
{
      S=new Shape *[5]; //声明一个指针数组，并把首地址赋给 S
      S[0]=new Triangle(3.0,5.0); //创建数组元素
      S[1]=new Rectangle(5.0,8.0);
      S[2]=new Circle(8.5);
      S[3]=new Trapezoid(12.0,8.0,6.0);
      S[4]=new Square(6.8);
}
MyProgram::~MyProgram()
{
    //释放内存空间
      for(int i=0; i<5; i++)
            delete S[i];
         delete [ ] S ;
}
double MyProgram::Run()
{
      double sum=Compute(S,5);
      return sum;
}
void main()
{
      MyProgram M;
      cout<<"Area's sum="<<M.Run()<<endl;
}
```

1.5 实验内容 II　(自主完成)

注：将题目的构思过程、源码、运行结果（截图）、心得体会等内容按要求填写，详见实验报告模板。

定义一个基类为哺乳动物类 mammal，其中有数据成员年龄、重量、品种，有成员函数 move()、speak()、eat()等，以此表示动物的行为。由这个基类派生出狗、猫、马、猪等哺乳动物，它们都有各自的行为。编程分别使各个动物表现出不同的行为。

编程思想：

1、为实现动态联编，首先建立 Mammal 抽象类，以此抽象类作为基类，派生 dog、cat、horse、pig 类。其中 Mammal 类数据员有(姓名)name 、(年龄)age、(重量)weight。成员函数 move()、eat()、speak()，定义为纯虚函数：另一个成员函数 display()，声明为虚函数。

2、建立各个派生类 dog、cat、horse、pig。然后建立构造函数为其初始化。再定义函数 move()、speak()、eat()等。

3、main()

函数中建立指向 Mammal 的指针数组，并为各派生类初始化。把指针数组分别指向各个派生类。设计一个循环来显示派生类对象的信息。