第六章 多态与虚函数

主要内容：

1.面向对象的三大特征：封装、多态、继承

2.普通虚函数和虚析构函数，纯虚函数，抽象类和接口类，还有异常处理 隐藏和覆盖，虚函数表

一、多态

1.基本概念

(1)什么是多态

指相同对象接收不同消息或不同对象收到相同消息时产生不同的动作。

(2)静态多态（早绑定）

```
class Rect
{
  public:
  int calcArea(int width);//计算面积
  int calcArea(int width,int height);//名字相同，参数个数不同，互为重载
};
int main()
{
  Rect rect;
  rect.calcArea(10);
  rect.calcArea(10,20);
  return 0;
 }//计算机自动匹配调用，这就叫做静态多态，早绑定。
```

(3)动态多态（晚绑定）

对不同的对象，做着相同的事情，动态多态要异继承和封装为基础

```
class Shape
{
  public:
  double calcArea()
  {
    cout<<"calcArea"<<endl;
    return 0;
  }
};

class Circle:public Shape//圆类
{
  public:
    Circle(double r);
    double calcArea();
  private:
    double m_dR;
 };

 class Rect:public Shape//矩形类
 {
   public:
     Rect(double width,double height);
     double calcArea();
   private:
     double m_dWidth;
     double m_dHeight;
  };
  double Rect::calcArea()
  {
    return m_dWidth *m_dHeight;
  };

  int main()//如何使用
  {
    Shape *shape1=new Circle(4.0);
    Shape *shape2=new Rect(3.0,5.0);
    shape1->calcArea();
    shape2->calcArea();
    .....
    return 0;
   }//没有办法实现，所以引入虚函数virtual
```

二、虚函数

1.基本概念

1.virtual关键字只在类定义中的成员函数声明处使用，不能在类外部写成员函数体时使用，静态成员函数和构造函数不能是虚函数。

2.包含虚函数的类成为“多态类”。

3.多态可以简单的理解为同一条函数调用语句能调用不同的函数，或者说，对不同对象发送同一条消息，使得不同对象有各自不同的行为。

```
class Shape
{
   public:
     virtual double calcArea()//虚函数
     {
        cout<<"calcArea"<<endl;
        return 0;
     }
 };

 class Circle:public Shape
 {
   public:
     Circle(double r);
     virtual double calcArea();//不是必须的，但是最好加上关键字
   private:
     double m_dR;
  };

  class Rect:public Shape
 {
   public:
     Rect(double width,double height);
     virtual double calcArea();//不是必须的，但是最好加上关键字
   private:
     double m_dWidth;
     double m_dHeight;
  };

  int main()//main函数中如何使用
 {
   Shape *shape1=new Circle(4.0);
   Shape *shape2=new Rect(3.0,5.0);
```

```
    shape1->calcArea();
    shape2->calcArea();
    delete shape1;
    delete shape2;
    return 0;
}
```

2.编码示例

```
#include<iostream>
#include<stdlib.h>
using namespace std;
//动态多态，虚函数
//要求：1.定义Shape类，成员函数：calcArea()，构造函数，析构函数
//     2.定义Rect类，成员函数:calcArea()，构造函数，析构函数
//                 数据成员：m_dWidth,m_dHeight
//     3.定义Circle类，成员函数:calcArea()，构造函数，析构函数
//                 数据成员：m_dR
class Shape
{
  public:
    Shape();
    ~Shape();
    virtual double calcArea();//实现了多态，注意要在其他两个也加上virtual关键字
};
Shape::Shape()//构造函数
{
  cout<<"Shape()"<<endl;
}
Shape::~Shape()//析构函数
{
  cout<<"~Shape()"<<endl;
}
double Shape::calcArea()
{
  cout<<"Shape->calcArea()"<<endl;
  return 0;//必须有返回值，没有的话写0
}
//Circle类
class Circle:public Shape
{
public:
  Circle(double r);
  ~Circle();
  virtual double calcArea();
```

```
protected:
  double m_dR;
};
Circle::Circle(double r)
{
  cout<<"Circle()"<<endl;
  m_dR=r;
}
Circle::~Circle()
{
  cout<<"~Circle()"<<endl;
}
double Circle::calcArea()
{
  cout<<"Circle->calcArea()"<<endl;
  return 3.14*m_dR*m_dR;
}
//Rect类
class Rect:public Shape
{
public:
  Rect(double width,double height);
  ~Rect();
  virtual double calcArea();
protected:
  double m_dWidth;
  double m_dHeight;
};
Rect::Rect(double width,double height)
{
  cout<<"Rect()"<<endl;
  m_dHeight=height;
  m_dWidth=width;
}
Rect::~Rect()
{
  cout<<"~Rect()"<<endl;
}
double Rect::calcArea()
{
  cout<<"Rect->calcArea()"<<endl;
  return m_dWidth*m_dHeight;
}

int main()
{
```

```
  Shape *shape1=new Rect(3,6);
  Shape *shape2=new Circle(5);
  shape1->calcArea();
  shape2->calcArea();
  delete shape1;
  shape1=NULL;
  delete shape2;
  shape2=NULL;
  return 0;
}
```

三、虚析构函数

1.多态中存在的问题

需要解决内存泄露问题，为了避免使用父类指针释放子类对象造成的内存泄露

2.虚析构函数的定义

使用virtual关键字去修饰析构函数

```
class Shape
{
  public:
    Shape();
    virtual double calcArea();
    virtual ~Shape();//虚析构函数
 };
```

3.virtual关键字在函数中的使用限制

1）普通函数不能是虚函数

2）静态成员函数不能是虚函数

3）内联函数不能是虚函数，会忽略到inline关键字，使其变为一个纯粹的虚函数

4）构造函数不能是虚函数

4.虚析构函数的使用方法

代码示例：

```
#include <iostream>
#include <stdlib.h>
using namespace std;
//动态多态，虚函数
//要求：1.定义Shape类，成员函数：calcArea()，构造函数，析构函数
//      2.定义Rect类，成员函数:calcArea()，构造函数，析构函数
//                    数据成员：m_dWidth,m_dHeight
//      3.定义Circle类，成员函数:calcArea()，构造函数，析构函数
//                    数据成员：m_dR
//      4.定义Coordinate类，成员函数，构造函数，析构函数
//                  数据成员：m_iX,m_iY
class Shape
{
  public:
    Shape();
    virtual ~Shape();//虚析构函数，避免内存泄漏，记得把其他的也加上
    virtual double calcArea();//实现了多态，注意要在其他两个也加上virtual关键字
};
Shape::Shape()//构造函数
{
  cout<<"Shape()"<<endl;
}
Shape::~Shape()//析构函数
{
  cout<<"~Shape()"<<endl;
}
double Shape::calcArea()
{
  cout<<"Shape->calcArea()"<<endl;
  return 0;//必须有返回值，没有的话写0
}
//Coordinate类
class Coordinate
{
public:
  Coordinate(int x,int y);
  ~Coordinate();
private:
  int m_iX;
  int m_iY;
};
Coordinate::Coordinate(int x,int y)
```

```
{
  cout<<"Coordinate"<<endl;
  m_iX=x;
  m_iY=y;
}
Coordinate::~Coordinate()
{
  cout<<"Coordinate"<<endl;
}
//Circle类
class Circle:public Shape
{
public:
  Circle(double r);
  ~Circle();
  virtual double calcArea();
protected:
  double m_dR;
  Coordinate *m_pCenter;
};
Circle::Circle(double r)
{
  cout<<"Circle()"<<endl;
  m_dR=r;
  m_pCenter=new Coordinate(3,5);
}
Circle::~Circle()
{
  cout<<"~Circle()"<<endl;
  delete m_pCenter;
  m_pCenter=NULL;
}
double Circle::calcArea()
{
  cout<<"Circle->calcArea()"<<endl;
  return 3.14*m_dR*m_dR;
}
//Rect类
class Rect:public Shape
{
public:
  Rect(double width,double height);
  ~Rect();
  virtual double calcArea();
protected:
  double m_dWidth;
```

```
  double m_dHeight;
};
Rect::Rect(double width,double height)
{
  cout<<"Rect()"<<endl;
  m_dHeight=height;
  m_dWidth=width;
}
Rect::~Rect()
{
  cout<<"~Rect()"<<endl;
}
double Rect::calcArea()
{
  cout<<"Rect->calcArea()"<<endl;
  return m_dWidth*m_dHeight;
}
int main()
{
  Shape *shape1=new Rect(3,6);
  Shape *shape2=new Circle(5);
  shape1->calcArea();
  shape2->calcArea();
  delete shape1;
  shape1=NULL;
  delete shape2;
  shape2=NULL;
  return 0;
}
```

四、虚函数和虚析构函数的实现原理

1.函数指针

2.虚函数表

3.函数的覆盖与隐藏

4.虚析构函数的实现原理

理论前提：执行玩子类的析构函数就会执行父类的析构函数

5.证明虚函数表指针的存在

编码示例：

```
#include <iostream>
#include <stdlib.h>
using namespace std;
//动态多态，虚函数
//要求：1.定义Shape类，成员函数：calcArea()，构造函数，析构函数
//
//      2.定义Circle类，成员函数:构造函数，析构函数
//                    数据成员：m_iR
//      概念说明：1.对象的大小：数据成员所占的数据大小
//              2.对象的地址
//              3.对象成员的地址
//              4.虚函数指针表
class Shape
{
  public:
    Shape();
    ~Shape();//虚析构函数，避免内存泄漏，记得把其他的也加上
    double calcArea();//实现了多态，注意要在其他两个也加上virtual关键字
};
Shape::Shape()//构造函数
{
  //cout<<"Shape()"<<endl;
}
Shape::~Shape()//析构函数
{
  //cout<<"~Shape()"<<endl;
}
double Shape::calcArea()
{
  cout<<"Shape->calcArea()"<<endl;
  return 0;//必须有返回值，没有的话写0
}

//Circle类
class Circle:public Shape
{
public:
  Circle(int r);
  ~Circle();
protected:
  int m_iR;
};
```

```
Circle::Circle(int r)
{
  m_iR=r;
}
Circle::~Circle()
{
  //cout<<"~Circle()"<<endl;
}
int main()
{
  Shape shape;
  //cout<<sizeof(shape)<<endl;
  int *p=(int *)&shape;
  cout<<p<<endl;
  Circle circle(100);
  int *q=(int *)&circle;
  cout<<q<<endl;
  cout<<(unsigned int)(*q)<<endl;
  //cout<<sizeof(circle)<<endl;
  return 0;
}
```

五、纯虚函数和抽象类

1.纯虚函数

纯虚函数：没有函数体，定义时函数名的后面要加=0;

```
class Shape
{
  public:
    virtual double calcArea()           //虚函数
    {return 0;}
    virtual double calcPerimeter()=0; //纯虚函数
    .....
};
```

2.抽象类

1）抽象类：含有纯虚函数的类，比如上面程序的Shape类

2）抽象类无法实例化对象

3）抽象类的子类也有可能会是抽象类