多态性

1 虚函数

2 抽象类

基类和派生类举例

```
class Base { //基类Base定义
public:
   void display() const {
        cout << "Base::display()" << endl;
   }
};
//公有派生类Derived定义
class Derived: public Base {
public:
   void display() const {
        cout << "Derived::display()"<<endl;
   }
};
```

定义基类指针

```
void fun(Base *ptr) { //参数为指向基类对象的指针
  ptr->display();       //"对象指针->成员名"
}
int main() {            //主函数
  Base base;            //声明Base类对象
  Derived derived;  //声明Derived类对象
  fun(&base);         //用Base对象的指针调用函数
  fun(&derived);    //用Derived对象的指针调用函数
  return 0;
}
```

运行结果:

```
Base::display()
Base::display()
```

```
class Base { //基类Base定义
public:
  virtual void display() const {
        cout << "Base::display()" << endl;
  }
};
//公有派生类Derived定义
class Derived: public Base {
public:
  void display() const {
        cout << "Derived::display()"<<endl;
  }
};
```

```
void fun(Base *ptr) { //参数为指向基类对象的指针
  ptr->display();       //"对象指针->成员名"
}
int main() {            //主函数
  Base base;            //声明Base类对象
  Derived derived;  //声明Derived类对象
  fun(&base);         //用Base对象的指针调用函数
  fun(&derived);    //用Derived对象的指针调用函数
  return 0;
}
```

运行结果:

```
Base::display()
Derived::display()
```

8.3 虚函数

- 用virtual关键字说明的函数
- 虚函数是实现运行时多态性基础
- C++中的虚函数是动态绑定的函数
- 虚函数必须是非静态的成员函数
- 虚函数经过派生之后，就可以实现运行过程中的多态。

8.3.1 虚函数成员

- C++中引入了虚函数的机制在派生类中可以对基类中的成员函数进行覆盖（重定义）。
- 虚函数的声明

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

8.4.1 纯虚函数

- 纯虚函数是一个在基类中声明的虚函数，它在该基类中没有定义具体的操作内容，要求各派生类根据实际需要重写该函数，纯虚函数的声明格式为：
- virtual 函数类型 函数名(参数表) = 0;
- 带有纯虚函数的类称为抽象类:

```
class  类名
 {
      virtual 类型 函数名(参数表)=0; //纯虚函数
     …
}
```

8.4.2 抽象类

- 定义：抽象类是带有纯虚函数的类。对于暂时无法实现的函数，可以声明为纯虚函数，留给派生类去实现。
- 作用：通过多态性调用虚函数。
- 注意：
  - 抽象类只能作为基类来使用。
  - 不能声明抽象类的对象。
  - 构造函数不能是虚函数，析构函数可以是虚函数

定义抽象基类Shape(1/6)

```
#include <iostream.h>
class Shape
{public:
    virtual double area() const=0;
    virtual void show() const=0;
};
```

Circle类的定义(2/6)

```
#define PI 3.1416
class Circle :public Shape
{public:
    Circle(double r1=1.0){r=r1;};
    double area()const{return PI*r*r;};
    void show()const{cout<<"I am a Circle:";}
 private:
    double r;
};
```

Rectangle类的定义(4/6)

```
class Rectangle :public Shape
{public:
    Rectangle(double=1.0,double=1.0);
    double area() const;
    void show() const;
 private:
    double length;
    double width;
};
```

Rectangle类的具体实现(5/6)

```
Rectangle::Rectangle(double a, double b)
{  length = a;
   width = b;
}
double Rectangle::area() const
{  return length*width;
}
void Rectangle::show() const
{  cout<<" I am a Rectangle: ";
}
```

抽象基类的例子(6/6)

```
void callArea(Shape &obj)
{  obj.show();
   cout<<"area = "<<obj.area()<<endl;
}
int main()
{  Circle cir(2.5);
   Rectangle rec(2.4,5.3);
   callArea(cir);
   callArea(rec);
   return 0;
}
```

```
I am a Circle: area = 19.635
I am a Rectangle: area = 12.72
```

8.8 小结

- 主要内容
  - 多态性的概念、虚函数、纯虚函数、抽象类
- 达到的目标
  - 理解多态的概念，学会运用多态机制。