第9章 统一接口 不同实现—多态性

1、多态性的基本概念	2、派生类对象替换基类对象
3、虚函数的定义	4、抽象类的定义
5、宠物类的设计	6、运算符重载
7、日期类对象判断大小	8、分数类对象运算符重载

分数类的抽象描述

- 数据成员：分子与分母
  - 都是整型变量：int a,b;
  - 分子、分母定义成公有、私有、保护理论上都可以
  - 但根据抽象封装原理，数据成员应该定义为私有成员
- 函数成员：设置数据、输出分数、+、==、求最大公因数、求负等
  - void set(int aa,int bb);//设置分子分母
  - void show();//显示分数
  - Fraction add(Fraction b);//加一个分数
  - Fraction operator+(Fraction b);//运算符+重载
  - Bool operator==(Fraction b);//运算符==重载
  - Fraction operator-(); //求负运算符-重载
  - int divisor(int p,int q);//求最大公约数

分数类的定义

```
class Fraction
{
private:
        int a;//分子
        int b;//分母
        int divisor(int p,int q);//求最大公约数
public:
        Fraction(){ a=0;b=1; }  //无参构造函数
        Fraction(int,int);                    //有参构造函数
        void set(int aa,int bb);//设置分子分母
        void show();//显示分数
        Fraction add(Fraction b);//加一个分数
        Fraction operator+(Fraction u);          //运算符+重载
        bool operator==(Fraction u);  //运算符==重载
        Fraction operator-();                    //求负运算符重载
        ~Fraction(){ };             //析构函数
};
```

类外定义构造、设置与显示函数

```
Fraction::Fraction(int x,int y)        //有参数构造函数
{
        set(x,y);
}
void Fraction:: set(int aa,int bb)    //设置分子、分母
{
        a=aa;
        if (bb!=0) //分母有效性检验
                b=bb;
        else
        {
                a=0;
                b=1;
        }
}
void Fraction::show()          //显示分数
{
        cout<<a<<"/"<<b;
}
```

类外定义相加成员函数

```
//分数相加，本类对象加u
Fraction Fraction::add(Fraction u)
{
        int tmp;
        Fraction v;

        v.a=a*u.b+b*u.a;   //分子
        v.b=b*u.b;//分母
        tmp=divisor(v.a ,v.b);//计算分子、分母的公约数
        v.a=v.a/tmp;//约去公约数
        v.b=v.b/tmp;//约去公约数
        return v;          //返回结果
}
```

类外定义运算符+重载

```
//分数相加，本类对象加u
Fraction Fraction::operator+(Fraction u)        //运算符+重载
{
        int tmp;
        Fraction v;

        v.a=a*u.b+b*u.a;   //分母
        v.b=b*u.b;//分子
        tmp=divisor(v.a ,v.b);//计算分子、分母的公约数
        v.a=v.a/tmp;//约去公约数
        v.b=v.b/tmp;//约去公约数
        return v;          //返回结果
}
```

类外定义运算符==重载

```
bool Fraction::operator==(Fraction u)   //运算符==重载
{
        float x,y;
        x=(float)a/b;
        y=(float)u.a/u.b;
        if(x==y)
                return true;
        else
                return false;
}
```

类外定义求负运算符–重载

```
Fraction Fraction::operator - ()     //求负运算符重载
{
        a=a*(-1);
        return *this;
}
```

类外定义求最大公因数成员函数

```
int Fraction::divisor(int p,int q)
{
        int r;
        if(p<q)
        {
                int tmp;
                tmp=p;
                p=q;
                q=tmp;
        }
        r=p%q;
        while(r!=0)
        {
                p=q;
                q=r;
                r=p%q;
        }
        return q;
}
```

测试主函数

```
int main()
{
        Fraction f1,f2,f3;//声明类的三个对象
        f1.set (4,-5);      //设置分数1
        f2.set (3,6);       //设置分数2
        f1.show();                  //显示分数1
        cout<<" + ";                //显示加号
        f2.show ();                 //显示分数2
        f3=f1+f2;           //计算分数和
        cout<<" = ";        //显示等号
        f3.show ();                 //显示分数的和
        f1.set(6,-20);
        if(f1==f3)          //测试相等运算符==重载
                cout<<"\nf1==f3"<<endl;
        else
                cout<<"\nf1!=f3"<<endl;
        f2=(-f2); //测试求负运算符-重载
        f2.show();
        return 0;
}
```