第7章　数据的抽象与封装

1、实体、对象与类的概念	2、类的定义	3、对象声明与引用	4、私有、公有与保护
5、日期类的设计	6、两种程序设计思想	7、汽车类的设计	8、几何图形圆类的设计
9、构造函数的定义	10、重载构造函数	11、析构函数的定义	12、整数翻译函数
13、实际意义的析构函数	14、Person类的设计	15、对象与指针	16、this指针

OOP的封装

- 在类的定义中不难看出对实体数据的封装
  - 将数据成员和函数成员集成封装在“类体”中
  - 数据成员私有化，外界不能随便读写
  - 函数成员公有化，外界可以调用(运行)
  - 用类去声明各种对象，程序对对象进行相关处理
  - 核心是将实体设计成“统一体”去进行各种处理
- C++还提供了许多封装的方法
  - 构造方法
  - 析构方法

已设计出的钟表类的共性

```
class    Clock   //例7-1描述钟表类
{
private:  //数据成员一般为私有成员
            int  Hour;
            int  Minute;
            int  Second;
public://函数成员一般为公有成员
            void Set(int h,int m,int  s);    //设置时间
            {
                        Hour=h;  Minute=m;  Second=s;
            }
            void Run();                    //钟表运转
            void Report_Time();        //报时响铃
            void Show_Time( );          //显示时间
            int Get_Hour( );             //得到小时数
            int Get_Minute( );          //得到分钟数
            int Get_Second( );          //得到秒数
};
```

已设计出的类的共性

```
class Date
{
private:
        int year,month,day;
public:
        void init(int,int,int );
        void print_ymd();
        void print_mdy();
        int get_year() { return year; }
        int get_month() { return month; }
        int get_day() { return day; }
};
void Date::init(int  yy,int  mm,int  dd)
{
        month=(mm>=1&&mm<=12) ? mm:1;
        year=(yy>=1900&&yy<=9999) ? yy:1900;
        day=(dd>=1&&dd<=31) ? dd:1;
};
```

已设计出的分数类的共性

```
class Fraction
{
private:
        int a;//分子
        int b;//分母
        int divisor(int p,int q);          //求最大公约数
public:
        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
};
```

已设计出的汽车类的共性

```
class automobile
{
private:
        char type[20];                  //汽车型号
        char color[20];                 //汽车颜色
        float price;                    //价格
        int carry_weight;               //载重量
        int carry_customer;             //载客量
public:
        void set_data(char *t,char *c,float pri,int cw,int cc);   //初始化或修改数据成员
        void movecar(int k);                                      //汽车水平运动k步
        void horming(int num);                                    //汽车鸣笛
        void downcar();                                           //汽车垂直向下运动
        void play_mp3(char *ps);                                  //播放歌曲
        char *show_type(){return type;}                           //取出汽车型号
};
```

已设计出的点类的共性

```
class Point
{
        int x, y;       //点的x和y坐标
public:
        void InitPoint( int, int );  // 设置坐标
        int GetX() { return x; } // 取x坐标
        int GetY() { return y; } // 取y坐标
        void Print();                   //输出点的坐标
};
```

已设计出的圆类的共性

```
class Circle
{
private:
        double radius;
        Point Center;
public:
        void InitCircle( double, Point);   //设置圆类数据
        double GetRadius();//取半径
        Point GetCenter();          //取圆心
        double Area();                  //计算面积a
        void Print();             //输出圆心坐标和半径
};
```

已设计出的类的共性

- 数据成员都是私有化成员，外界不能直接访问
- 都有一个成员函数，用来设置数据成员
- 该函数可以在需要时调用，达到随时修改数据成员
- 数据成员值读出必须通过相关成员函数

构造函数的定义

- 在声明一个对象后，紧接着要给对象初始化
- 对象初始化实质上就是对所有数据成员赋值
- 如果对象中某个数据成员未赋值，则该数据成员的值不确定，那么该对象的值不完整
- 构造函数（Constructor）用于创建一个对象，提供了初始化该对象的一种简便手段
- 注意在前面所有类的定义中都有一个成员函数完成初始化数据成员，这个函数就可以当成构造函数

构造函数的定义

- 构造函数的语法格式：

  〈类名〉(〈参数表〉)

  {

  〈函数体〉

  }

- 注意事项：
  - 构造函数的函数名必须与类名相同
  - 构造函数没有返回值
  - 其功能是将对象中的所有数据成员进行初始化，一般由一系列赋值语句构成
  - 由系统在声明对象时自动调用

在日期类中增加构造函数

```
#include<iostream>
using namespace std;
class Date
{           int year,month,day;
public:
            Date(int y=1900,int m=1,int d=1)
            {
                        year = y;  month = m;  day = d;
            }
            void init(int,int,int );
            void print_ymd();
            void print_mdy();
};
void Date::init(int  yy,int  mm,int  dd)
{           month=(mm>=1&&mm<=12) ? mm:1;
            year=(yy>=1900&&yy<=9999) ? yy:1900;
            day=(dd>=1&&dd<=31) ? dd:1;
};
```

日期类构造函数测试

```
void Date::print_ymd()
{       cout<<year<<"-"<<month<<"-"<<day<<endl;}
void Date::print_mdy()
{       cout<<month<<"-"<<day<<"-"<<year<<endl;}
int main()
{
        Date date1;                     //创建一个日期类对象，并初始化
        date1.print_ymd();              //显示初始化数据的情况
        date1.init(2006,3,28);          //修改数据成员
        date1.print_ymd();              //显示修改数据的情况
        Date date2(2013,11,26);         //再创建一个日期类对象，并初始化
        date2.print_mdy();
        date2.init(2006,13,38);
        date2.print_ymd();
        return 0;
}
```

在日期类中取消构造函数

```
#include<iostream>
using namespace std;
class Date
{           int year,month,day;
public:
/*          Date(int y=1900,int m=1,int d=1)
            {
                        year = y;  month = m;  day = d;
            }*/
            void init(int,int,int );
            void print_ymd();
            void print_mdy();
};
void Date::init(int  yy,int  mm,int  dd)
{           month=(mm>=1&&mm<=12) ? mm:1;
            year=(yy>=1900&&yy<=9999) ? yy:1900;
            day=(dd>=1&&dd<=31) ? dd:1;
};
```

日期类无构造函数测试

```
void Date::print_ymd()
{       cout<<year<<"-"<<month<<"-"<<day<<endl;}
void Date::print_mdy()
{       cout<<month<<"-"<<day<<"-"<<year<<endl;}
int main()
{
        Date date1;                     //创建一个日期类对象，并初始化
        date1.print_ymd();              //显示初始化数据的情况
        date1.init(2006,3,28);          //修改数据成员
        date1.print_ymd();              //显示修改数据的情况
        Date date2(2013,11,26);         //再创建一个日期类对象，并初始化
        date2.print_mdy();
        date2.init(2006,13,38);
        date2.print_ymd();
        return 0;
}
```

有无构造函数的区别

- 在前面章节中的程序声明变量
  - int a=0; 或int a(0);
  - struct Date today={2013,3,19};
  - 或struct Date today(2013,3,19);
  - 以上为变量声明的同时赋初值，即都允许初始化
- 对于无构造函数的类，声明变量(对象)不允许初始化
  - 例如 CLOCK alarm={10,53,11}; ×
  - CLOCK alarm(10,53,11) ×
- 有构造函数就允许初始化
  - 例如: Date birthday(1998,12,12); √
  - 但Date birthday={1998,12,12}; ×

感谢收看！