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

日期类的抽象描述

- 日期类是所有公元日期的集合
- 每个公元日期的特征和功能可以作为日期类的特征和功能
- 公元日期的特征(数据成员)如下：
  - 年份：**int year;**
  - 月份：**int month;**
  - 日：**int day;**

日期类的抽象描述

- 公元日期的功能(函数成员)：
  - 设置日期：**void init(int y,int m,int d);**
  - 显示日期：分两种格式显示日期值
    - 按年月日格式显示：**void print_ymd();**
    - 按月日年格式显示：**void print_mdn();**
  - 取出年份值：**int get_year();**
  - 取出月份值：**int get_month();**
  - 取出日的值：**int get_day();**
  - 判闰年：**bool IsLeapYear() ;**

日期类的定义

```
class Date
{
private:
        int year,month,day;
        void SetSystemDate();                    //取得系统日期
public:
        void init(int,int,int );                 //设置年月日数据
        void print_ymd();                        //显示年月日
        void print_mdy();                        //显示月日年
        int get_year() { return year; }          //得到年份值
        int get_month() { return month; }        //得到月份值
        int get_day() { return day; }            //得到日值
        bool IsLeapYear();                       //判断是否为闰年
};
```

类外定义成员函数

```
void Date::SetSystemDate()
{
        //取得系统日期
        tm *gm;     //tm是时间(含年月日时分秒)结构体
        time_t t=time(NULL);    //time_t是长整型，t表示 (总秒数)
        gm = gmtime(&t);
        year = 1900 + gm->tm_year;
        month = gm->tm_mon +1;
        day = gm->tm_mday;
}
```

类外定义成员函数

```
void Date::init(int  yy,int  mm,int  dd)
{
        if (yy>=1900&&yy<=9999)
                year = yy;
        else
                {
                        SetSystemDate();
                        return;
                }
        if (mm>=1&&mm<=12)
                month = mm;
        else
                {
                        SetSystemDate();
                        return;
                }
        if (dd>=1&&dd<=31)
                day = dd;
        else
                {
                        SetSystemDate();
                        return;
                }
}
```

类外定义成员函数

```
void Date::print_ymd()
{
        cout<<year<<"-"<<month<<"-"<<day<<endl;
}
void Date::print_mdy()
{
        cout<<month<< “-” <<day<< “-” <<year<<endl;
}
bool Date::IsLeapYear()
{
        if ( year % 400 == 0 || ( year % 100 != 0 && year % 4 == 0 ) )
                return true;
        else
                return false;
}
```

日期类的测试主函数

```
#include<iostream>
#include<time.h>
using namespace std;
int main()
{           Date date1;                    //创建一个日期类对象，但未初始化
            date1.print_ymd();             //显示未初始化数据的情况
            system("pause");
            date1.init(2008,3,28);         //初始化数据成员
            date1.print_ymd();             //按年月日显示日期
            system("pause");
            Date date2;                    //再创建一个日期类对象
            date2.init(2006,13,28);        //初始化错误的日期数据
            date2.print_mdy ();            //按月日年显示日期
            system("pause");
            if (date1.IsLeapYear())        //测试判断闰年函数
                        cout<<date1.get_year()<<"是闰年"<<endl;
            else
                        cout<<date1.get_year()<<"不是闰年"<<endl;
            return 0;
}
```

感谢收看！