北京师范大学2018-2019学年第一学期 “C++程序设计”

上机作业八

姓 名	学 号	日 期
袁宇昊	201611130126	2018.11.06

实验目的

- **类的多继承应用。**
- **虚基类的应用。**

实验总结：

请在以下总结实验中发现的问题和解决办法或心得体会。请勿黏贴过多源码。

1. **问题**：实现计时过程可以重复多次

   解决：使用while(1)循环，在循环里判断是否结束

```
while(1)
{
    cout<<"计时结束，还要继续吗（Y/N）？";
    cin>>flag;
    if(flag!='Y')
        break;
}
```

2. **问题**：总是在同一行显示时间和时间的变化。

   解决： 使用'\r'转义字符。

```
void Clock::ShowTime()
{
    cout<<setw(2)<<setfill('0')<<Hour<<":"<<setw(2)<<setfill('0')<<Minute<<":"<<setw(2)
<<setfill('0')<<Second<<'\r';
}
```

3.**问题**：多继承下的构造函数写法：

解决： 使用列表元素的写法：

```
StudentEmployee(string Name,int Age,string Major,int studentID,
                    string Department,int employeeID):
          Person(Name,Age),//调用Person类的构造函数
          Student(Name,Age,Major,studentID),//调用Student类的构造函数
          Employee(Name,Age,Department,employeeID){};//调用Employee类的构造函数
```

4.**问题**：多继承下的二义性问题：

解决：使用虚基类，否则StudentEmployee类会有两套Person类中的变量导致重名：

```
class Person{};
class Employee:virtual public Person{};//虚基
class Student:virtual public Person{};//虚基
class StudentEmployee:public Student,public Employee{};
```

5.**问题**： 日期的进位问题：

有三个小问题：

1.年月日的进位：

因为涉及到润年的问题，所以写出isLeapYear(year)函数判断是否是闰年，再在getMonthDay(year,month)函数中调用之来得到对应年份、对应月份的天数。

```
const int MonthDay[13]={-1,31,28,31,30,31,30,31,31,30,31,30,31};//平年的个月份天数
bool CMyDate::isLeapYear(int year)//判断是否是闰年
{
    if(year%400==0)
        return true;
    if(year%100!=0 && year%4==0)
        return true;
    return false;
}
int CMyDate::getMonthDay(int year,int month)
{
    if(month==2)
        return MonthDay[month]+isLeapYear(year);
    return MonthDay[month];
}
```

再在mydate类中的Next()函数中调用getMonthDay函数：

```
void CMyDate::Next()
{
    Day++;
    if(Day>getMonthDay(Year,Month))//调用getMonthDay()函数
    {
        Day=1;
        Month++;
        if(Month>12)
        {
```

```
            Month=1;
            Year++;
        }
    }
}
```

2.时分秒的进位：这是最开始的写法：

```
void CMyTime::Next()
{
    Second++;
    if(Second>=60)
    {
        Second=0;
        Minute++;
        if(Minute>=60)
        {
            Minute=0;
            Hour++;
            if(Hour>=24)
                Hour=0;
        }
    }
}
```

但是后来为了解决第三个问题，做了更改。

3.年月日时分秒的进位：本来是像像上面的两个进位方法类似地写，但考虑到写出来的Next()函数会有6重if语句的嵌套，过于复杂，而且这个进位和上面的进位特别相似，所以考虑调用上面的Next()函数。

因为这里也是计算下一秒的时间，所以想直接调用时分秒的进位函数，然后再判断是否是这一天的最后一秒，再考虑调用年月日的进位。但是那样的话不能完成，因为年月日是private的变量，继承后直接不可访问了。可以考虑使用友元类的方法可以访问年月日，但我想到了一个更简洁的方法：

让时分秒的Next()函数带一个返回值，返回的是它是否度过了一天：

```
bool CMyTime::Next()//返回值从void变成了bool类型，true表示度过了一天，false表示没有
{
    Second++;
    if(Second>=60)
    {
        Second=0;
        Minute++;
        if(Minute>=60)
        {
            Minute=0;
            Hour++;
            if(Hour>=24)
            {
                Hour=0;
                return true;//hour置0，返回true
            }
        }
```

```
    }
    return false;//返回false
}
```

这样一来，涉及六个时间变量的Next()函数就特别简洁了：

```
void CMyDateTime::Next()
{
    if(CMyTime::Next())
        CMyDate::Next();
}
```

本来会特别复杂的CMyDateTime::Next()函数，只通过改变CMyTime::Next()的返回值就让它变得非常简洁。