高级程序设计 HW2

STUNAME: 汪福运

STUID:191300051

T1

1.1构造函数的成员初始化列表的作用

一方面，对于常量和引用数据的成员，即不能够在说明他们时进行初始化，也不能够在构造函数中对他们赋值。只能够在成员初始化列表中对他们进行初始化。

另一方面，在创建包含成员对象的对象时，除了会自动调用本身类的构造函数外，还会自动调用成员对象的构造函数。而通常情况下，自动调用的是成员对象的默认构造函数。如果要调用成员对象类的非默认构造函数，需要在构造函数的成员初始化列中显示的指出。

1.2 C++的成员对象是什么？ 当创建包含成员对象的类的对象时，构造函数的调用顺序是什么样的?

（1）成员对象是一个类的数据成员，它是另一个类的对象。

（2）

先执行成员对象类的构造函数，在执行本对象类的构造函数。

如果含有多个成员对象，则按照它们在本类中的说明顺序执行。

T2 代码编程题

```
#include <stdio.h>
#include <assert.h>

//输出宏
#define COLOR_NONE                    \033[0m
#define FONT_COLOR_RED              \033[0;31m
#define FONT_COLOR_BLUE             \033[1;34m
#define Assert(expr,type) if ((expr)==0) {printf("\033[0;31m%s!!
\033[0m",Errors_string[type]);assert(0);}


//常数宏
#define Minimum 0
#define Maxsecond 60
#define Maxminite Maxsecond
#define Maxhour 24
```

```
16
17
18  //全局常量
19  const char Errors_string[3][10]={ "Herror","Merror","Serror"};
20  enum Errors_type {Herror,Merror,Serror};
21
22
23
24  //类的定义
25  class Time
26  {
27      private:
28      int t_hour,t_minite,t_second;
29
30      void simp()
31      {
32          while (t_second>=Maxsecond)
33          {
34              t_second-=Maxsecond;
35              t_minite++;
36          }
37          while (t_minite>=Maxminite)
38          {
39              t_minite-=Maxminite;
40              t_hour++;
41          }
42          while (t_hour>=Maxhour)
43          {
44              t_hour-=Maxhour;
45
46          }
47      }
48      public:
49      Time(int h=0,int m=0,int s=0)
50      {
51          set(h,m,s);
52      }
53      void set(int h,int m,int s)
54      {
55          Assert(h>=Minimum&&h<Maxhour,Herror);
56          Assert(m>=Minimum&&m<Maxminite,Merror);
57          Assert(s>=Minimum&&s<Maxsecond,Serror);
58
59          t_hour=h;
60          t_minite=m;
61          t_second=s;
62      }
63      void increment()
64      {
65          t_second++;
66          simp();
67
68      }
69      void display()
70      {
71          printf("Time:  %02d:%02d:%02d\n",t_hour,t_minite,t_second);
72      }
73
```

```
    bool equal(Time &other_time)
    {
        return (t_hour==other_time.t_hour&&t_minite==other_time.t_minite&&t_second==other_time.t_second);
    }

    bool less_then(Time &other_time)
    {
        return (t_hour<other_time.t_hour) || \
        (t_hour==other_time.t_hour && \
        t_minite<other_time.t_minite)||\
        (t_hour==other_time.t_hour &&\
        t_minite==other_time.t_minite&&\
        t_second<other_time.t_second);
    }
};




//正常测试
void test1()
{
    Time a;
    Time b(0,2,10);
    a.display();
    b.display();
    printf("a<=b?%d\n",a.less_then(b));
    for (int i=0;i<=(int)100000;i++)
    {
        a.display();
        a.increment();
    }
    a.display();
    b.display();
    printf("a<=b?%d\n",a.less_then(b));
}


//异常测试
void test2()
{
    Time a(-1,2,100);
    // Time b(24,0,0);
    // Time a;
    // a.set(1,61,1);
    // a.set(1,50,-1);
}
int main()
{
    // test1();
    test2();

}
```