作业三 221900180 田永铭

概念题:

1. 派生类从基类那里继承了什么？派生类不能从基类继承什么？

答:派生类继承了基类的所有成员,包括成员变量和成员函数,但不能继承基类的构造函数、析构函数和赋值操作符重载函数。

2. 基类的友元是否是派生类的友元？基类是某个类的友元时，派生类是否也是该类友元？

答：如果在派生类中没有显式说明，则基类的友元不是派生类的友元;

如果基类是另一个类的友元，而该类没有显式说明，则派生类也不是该类的友元。

3. C++中 protected 类成员访问控制的作用是什么？

缓解了封装与继承的矛盾。

4.

（1）D 构造函数 D:: D（）call B 的构造函数 B:: B（），B 的构造函数 call A 的构造函数 A:: A（），A return 给 B，B return 给 D。

（2）D 默认的拷贝构造函数 const D&d call B 的 构造函数 const B&b，B 的构造函数 call A 的构造函数 const A&a，A return 给 B，B return 给 D。

（3）调用派生类 D:: ~D（），call C 的 C:: ~C（），此后，D 又 call B 的 B:: ~B（），B call A 的 A:: ~A（）,依次 return。

编程题:

1.以下为通关代码:

```
#include<iostream>
using namespace std;

// TODO: implement the following class
//       you can add any pulic interfaces if needed,
//       but do not expose any data member public

class Date {
    friend class Time;
    friend class ExtTime;
  private:
    int year;
    int month;
    int day;
  public:
    Date():year(2222),month(2),day(22){}
    Date(int y, int m, int d):year(y),month(m),day(d){}
    void set(int y, int m, int d)
    {
```

```
      year =  y;
      month = m;
      day = d;
    }
    void display()
    {
      cout<<year<<"年"<<month<<"月"<<day<<"日"<<endl;
    }
    void increase()
    {
      day++;
      int leap = 0;
      if((year % 4 == 0 && year % 100 != 0) || (year % 400 ==
0)) leap = 1;
      int days[] = {31,28,31,30,31,30,31,31,30,31,30,31};
      if(leap) days[1]++;
      if(day > days[month-1])
      {
        day = 1;
        month++;
      }
      if(month > 12)
      {
        year++;
        month = 1;
      }
    }
};

class Time {
  friend class ExtTime;
  private:
    int hour;
    int minute;
    int second;
    Date dates;
  public:
    Time():dates(),hour(2),minute(2),second(2){}
    Time(Date & date, int h, int m, int s):dates(date),hour(h)
,minute(m),second(s){}
    void set(Date & date, int h, int m, int s)
    {
```

```
      dates = date;
      hour = h;
      minute = m;
      second = s;
    }
    void display()
    {
      cout<<dates.year<<"年"<<dates.month<<"月"<<dates.day<<"日
"<<hour<<"点"<<minute<<"分"<<second<<"秒"<<endl;
    }
    void increment()
    {
      second++;
      if(second >= 60)
      {
        minute++;
        second = 0;
        if(minute >= 60)
        {
          hour++;
          minute = 0;
          if(hour >= 24)
          {
            dates.increase();
            hour = 0;
           }
        }
      }
    }
    bool equal(Time &other_time)
    {
      return (dates.year == other_time.dates.year && dates.mon
th == other_time.dates.month && dates.day == other_time.dates.
day
          && hour == other_time.hour && minute == other_time.m
inute && second == other_time.second);
    }
    bool less_than(Time &other_time)
    {
      if(dates.year < other_time.dates.year) return true;
      if(dates.year > other_time.dates.year) return false;
      else
```

```
      {
        if(dates.month < other_time.dates.month) return true;
        if(dates.month > other_time.dates.month) return false;
        else
        {
          if(dates.day < other_time.dates.day) return true;
          if(dates.day > other_time.dates.day) return false;
          else
          {
            if(hour < other_time.hour) return true;
            if(hour > other_time.hour) return false;
            else
            {
              if(minute < other_time.minute) return true;
              if(minute > other_time.minute) return false;
              else
              {
                if(second < other_time.second) return true;
                if(second >= other_time.second) return false;
              }
            }
          }
        }
      }
    }
};

class ExtTime {
  private:
    Time time;
    int zones;
  public:
    ExtTime():zones(1),time(){}
    ExtTime(int zone, Date & date, int h, int m, int s):time(d
ate,h,m,s),zones(zone){}
    void set(int zone, Date & date, int h, int m, int s)
    {
      zones = zone;
      time.dates.year = date.year;
      time.dates.month = date.month;
      time.dates.day = date.day;
      time.hour = h;
```

```
      time.minute = m;
      time.second = s;
    }
    void display()
    {
      cout<<"第"<<zones<<"时区";
      time.display();
    }
    void increment()
    {
      time.increment();
    }
    bool equal(ExtTime &other_times)
    {
      return zones == other_times.zones && time.equal(other_ti
mes.time);
    }
    bool less_than(ExtTime &other_times)
    {
      if(zones < other_times.zones) return true;
      else if(zones > other_times.zones) return false;
      else return time.less_than(other_times.time);
    }
};
```