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.month == other\_time.dates.month && dates.day == other\_time.dates.day

          && hour == other\_time.hour && minute == other\_time.minute && 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(date,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\_times.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);

    }

};