70083 ExerciseTypes.CW2

C++ 2

Submitters

---

sf23 Shihan Fu

85/100

Emarking

```
Traffic: Summary for sf23 of v5
--------------------------------

  Comparison with Model Answer:
    Task 1:  1 / 1

Git Repo: git@gitlab.doc.ic.ac.uk:lab2324_autumn/msc_lab2_sf23.git
Commit ID: adbf3
```

```
Detailed Output for test: Task 1
--------------------------------

Task 1

  Compiled OK

  Compilation Standard Output:

g++ -Wall -g -c time.cpp
g++ -Wall -g -c trafficLight.cpp
g++ -Wall -g -c main.cpp
g++ -Wall -g time.o trafficLight.o main.o -o traffic

  Test Passed

```

```
/*
 * @Author: shihan
 * @Date: 2023-11-08 19:08:41
 * @version: 1.0
 * @description: class definition for TrafficLight
 */
/* trafficLights.h - header file for the class trafficLights */

#ifndef TRAFFICLIGHT_H
#define TRAFFICLIGHT_H

#include "time.h"
#include <string>

using namespace std;
/********************** Class TrafficLight **************************/



class TrafficLight {

  public:

    TrafficLight(Time _delay_time,char* _info);
    TrafficLight(Time _delay_time,char* _info,TrafficLight& _traffic_light);

    void carWantsToCross();

    static void setTheTime(Time the_time);

    friend std::ostream& operator << (std::ostream& os, TrafficLight* traffic_light);

    static Time global_time;

  private:

    /* add members and operations to complete the class yourself */
    string color;
    string name;
    string direction;
    Time delay_time;
    TrafficLight *co_light;

};

#endif

```



```
/*
 * @Author: shihan
 * @Date: 2023-11-08 19:08:41
 * @version: 1.0
 * @description: class definition for Time
 */
/* time.h - header file for the class Time */

#ifndef TIME_H
#define TIME_H

#include <iostream>

/********************** Time Class **************************/

class Time {

  public:

    Time();

    Time(int hours, int mins, int secs);

    void add(Time& anotherTime);
    /* adds seconds to seconds, minutes to minutes and
    hours to hours, taking into account that
    a day has 24 hours, an hour has 60 minutes
    and a minute has 60 seconds */

    friend std::ostream& operator << (std::ostream& os, Time& theTime);

    int compareTime(Time& anotherTime);

  private:

    int theHour;
    int theMins;
    int theSecs;

};

#endif

```

If the compareTime function is only going to be called by Time objects, you could have optionally declared it as a private function.

It would have been great to see some docstrings here. (-5 marks)

```cpp
/*
 * @Author: shihan
 * @Date: 2023-11-08 19:08:41
 * @version: 1.0
 * @description:
 */
/* main.cpp - for MSc Computing C++ Assessed Exercise 2. */

#include <iostream>
#include "trafficLight.h"
#include "time.h"

using namespace std;

int main() {
  cout << "=========================\n"
       << "Roads open in Sleepy Town:\n"
       << "=========================\n";

  /* (a) initialise the time: */
  // First, the global clock is initialised to 0:0:0

  Time timeZero(0, 0, 0);
  TrafficLight::setTheTime(timeZero);         // event (a) completed


  /* (b) create a pair of traffic lights with delays 15 minutes and 5
         minutes respectively:                                        */
  cout << "\nA pair T1 and T2 of (slow) collaborating lights is set up:\n";
  char T1Name[] = "T1 (North South)";
  char T2Name[] = "T2 (East West)";
  Time delayT1(0, 15, 0);
  Time delayT2(0, 5, 0);
  TrafficLight LightT1(delayT1, T1Name);
  // should be &LightT1
  TrafficLight LightT2(delayT2, T2Name, LightT1);   // event (b) completed


  /* (c)-(f) begin the simulation with 4 car crossings: */
  LightT1.carWantsToCross();              // event (c) completed
  LightT2.carWantsToCross();              // event (d) completed
  LightT1.carWantsToCross();              // event (e) completed
  LightT2.carWantsToCross();              // event (f) completed


  /* (g) create another pair of traffic lights with extra long delays of
    6hrs, 15mins, 44secs and 14hrs, 5mins, 57secs respectively:          */
  cout << "\nA new pair T3 and T4 of (very slow!) collaborating lights is now ↗
set up:\n";
  char T3Name[] = "T3 (North South)";
  char T4Name[] = "T4 (East West)";
  Time delayT3(6, 15, 44);
  Time delayT4(14, 5, 57);
  TrafficLight LightT3(delayT3, T3Name);
  TrafficLight LightT4(delayT4, T4Name, LightT3);   // event (g) completed


  /* (h)-(m) continue the simulation with 6 more car crossings: */
  LightT3.carWantsToCross();              // event (h) completed
  LightT3.carWantsToCross();              // event (i) completed
  LightT4.carWantsToCross();              // event (j) completed
  LightT4.carWantsToCross();              // event (k) completed
  LightT3.carWantsToCross();              // event (l) completed
  LightT4.carWantsToCross();              // event (m) completed

  cout << "\n==================================\n"
```



```cpp
       << "Roads close forever in Sleepy Town.\n"
       << "==================================\n";

  return 0;
}
```

```
/*
 * @Author: shihan
 * @Date: 2023-11-08 19:08:41
 * @version: 1.0
 * @description: class implementation for TrafficLight
 */

#include "trafficLight.h"
#include "time.h"
#include<iostream>
#include <string>

using namespace std;

/* initialization for static variable */
Time TrafficLight::global_time=Time();

/* TrafficLight constructor */
TrafficLight::TrafficLight(Time _delay_time, char* _info):delay_time(_delay_time){
    color = "red";
    string info = _info;
    std::string delimiter = " ";
    name = info.substr(0, info.find(delimiter));
    direction = info.substr(info.find(delimiter), info.length()-1);
}

/* TrafficLight constructor */
TrafficLight::TrafficLight(Time _delay_time,char* _info,TrafficLight& _traffic_light):delay_time(_delay_time), co_light(&_traffic_light){
    color = "red";
    string info = _info;
    std::string delimiter = " ";
    name = info.substr(0, info.find(delimiter));
    direction = info.substr(info.find(delimiter), info.length()-1);
    // combine both lights
    _traffic_light.co_light=this;
}

/* carWantsToCross: the logical implementation of changing light colours */
void TrafficLight::carWantsToCross(){
    cout << endl;
    // case 1. both lights are red, can directly change this light
    if(color == "red" && co_light->color == "red"){
        // output sample: ***  at 0:0:0 a car wants to cross light T1 (North South),, with colour: red
        cout << "***  at "<<global_time << " a car wants to cross light " << name
        << direction << ", with colour: " << color <<endl;
        // change the light's color to yello and green
        global_time.add(delay_time);
        // output sample: at 0:15:0 T1 (North South) changes colour to yellow
        color = "yellow";
        cout << "     at "<< global_time << " "<< name << direction
        << " changes colour to " << color <<endl;
        // output sample: at 0:30:0 T1 (North South) changes colour to green
        global_time.add(delay_time);
        color = "green";
        cout << "     at "<< global_time << " "<< name << direction
        << " changes colour to " << color <<endl;

    }

    // case 2. the light is red, the other one is green
    else if(color == "red" && co_light->color == "green"){
        cout << "***  at "<<global_time << " a car wants to cross light " << name
        << direction << ", with colour: " << color <<endl;
        // co-light, to yellow
        global_time.add(co_light->delay_time);
        co_light->color="yellow";
        cout << "     at "<< global_time << " "<< co_light->name << co_light->direction
        << " changes colour to " << co_light->color <<endl;
        //the-light, to yellow
        global_time.add(delay_time);
        color="yellow";
        cout << "     at "<< global_time << " "<< name << direction
        << " changes colour to " << color <<endl;
        //co-light, to red
        global_time.add(co_light->delay_time);
        co_light->color="red";
        cout << "     at "<< global_time << " "<< co_light->name << co_light->direction
        << " changes colour to " << co_light->color <<endl;
        //the-light, to green
        global_time.add(delay_time);
        color="green";
        cout << "     at "<< global_time << " "<< name << direction
        << " changes colour to " << color <<endl;
    }

    // case 3. the light is green
    else if(color == "green"){
        cout << "***  at "<<global_time << " a car wants to cross light " << name
        << direction << ", with colour: " << color <<endl;
    }

}

/* set the static variable: global_time to specific time */
void TrafficLight::setTheTime(Time the_time){
    TrafficLight::global_time = the_time;
}

/* overwrite the cout function */
std::ostream& operator << (std::ostream& os, TrafficLight* traffic_light){
    os << "traffic light name: " << traffic_light->name << ", direction: " << traffic_light->direction;
    return os;
}
```

A more elegent way to define the colour data members is to define them as enums rather than strings. (-5 marks)

if you're use "using namespace std;" you don't need the std:: namespace operator for objects from the std library.

Good that you use substr instead of directly assinging the str with =.

This function could have been further abstracted and split into multiple functions (-5 marks).

Traffic trafficLight.cpp: 2/2 Shihan Fu - sf23:v5

git@gitlab.doc.ic.ac.uk:lab2324_autumn/msc_lab2_sf23.git adbf3

```cpp
/*
 * @Author: shihan
 * @Date: 2023-11-08 19:08:41
 * @version: 1.0
 * @description: class implementation for Time
 */

#include "time.h"
#include<iostream>
#include <string>

using namespace std;

/* Time constructor */
Time::Time():theHour(0), theMins(0), theSecs(0){
}

/* Time constructor */
Time::Time(int hours, int mins, int secs):theHour(hours), theMins(mins), theSecs(secs){
}

/* Time add */
void Time::add(Time& anotherTime){
    int seconds;
    int minutes;
    int hours;

    seconds=(theSecs+anotherTime.theSecs)%60;
    minutes=(theMins+anotherTime.theMins+((theSecs+anotherTime.theSecs)/60))%60;
    hours=(theHour+anotherTime.theHour+(theMins+anotherTime.theMins+(theSecs+anotherTime.theSecs)/60)/60)%24;

    theSecs=seconds;
    theMins = minutes;
    theHour = hours;
}

/* overwrite the cout function */
std::ostream& operator << (std::ostream& os, Time& theTime){
    os << theTime.theHour << ":" << theTime.theMins << ":" << theTime.theSecs;
    return os;
};

/* Time comparision */
int Time::compareTime(Time& anotherTime){
    string this_time = to_string(theHour)+":"+to_string(theMins)+":"+to_string(theSecs);
    string another_time = to_string(anotherTime.theHour)+":"+to_string(anotherTime.theMins)+":"+to_string(anotherTime.theSecs);
    return (this_time < another_time)?-1:1;
}


```



```cpp
/*
 * @Author: shihan
 * @Date: 2023-11-08 19:08:41
 * @version: 1.0
 * @description: test of classes and basic functions
 */

#include<iostream>
#include "time.h"
#include "trafficLight.h"

using namespace std;

int main(){

    // time test
    Time timeZero(0, 0, 0);
  TrafficLight::setTheTime(timeZero);
    cout <<TrafficLight::global_time;

    // traffic light test
    cout << "\nA pair T1 and T2 of (slow) collaborating lights is set up:\n";
    char T1Name[] = "T1 (North South)";
    char T2Name[] = "T2 (East West)";
    Time delayT1(0, 15, 0);
    Time delayT2(0, 5, 0);
    TrafficLight LightT1(delayT1, T1Name);
    TrafficLight LightT2(delayT2, T2Name, LightT1);

    // function test
    LightT1.carWantsToCross();
    LightT2.carWantsToCross();

    return 0;

}
```

Nice to see a test function here!

```
1: traffic: time.o trafficLight.o main.o
2:   g++ -Wall -g time.o trafficLight.o main.o -o traffic
3:
4: time.o: time.cpp time.h
5:   g++ -Wall -g -c time.cpp
6:
7: trafficLight.o: trafficLight.cpp trafficLight.h
8:   g++ -Wall -g -c trafficLight.cpp
9:
10: main.o: main.cpp time.h trafficLight.h
11:   g++ -Wall -g -c main.cpp
12:
13: clean:
14:   rm -rf *.o main
```

Nicely broken down makefile.