

70083 ExerciseTypes.CW2

C++ 2

Submitters

sf23

Shihan Fu

85 / 100

Emarking

Traffic

TestSummary.txt: 1/1

Shihan Fu - sf23:v5

```
1: Traffic: Summary for sf23 of v5
2: -----
3:
4:   Comparison with Model Answer:
5:     Task 1:  1 / 1
6:
7: Git Repo: git@gitlab.doc.ic.ac.uk:lab2324_autumn/msc_lab2_sf23.git
8: Commit ID: adbf3
```

Traffic

results_iconv.txt: 1/1

Shihan Fu - sf23:v5

```
1: Detailed Output for test: Task 1
2: -----
3:
4: Task 1
5:
6:   Compiled OK
7:
8:   Compilation Standard Output:
9:
10: g++ -Wall -g -c time.cpp
11: g++ -Wall -g -c trafficLight.cpp
12: g++ -Wall -g -c main.cpp
13: g++ -Wall -g time.o trafficLight.o main.o -o traffic
14:
15:   Test Passed
16:
17:
```

```

1:  /*
2:  * @Author: shihan
3:  * @Date: 2023-11-08 19:08:41
4:  * @version: 1.0
5:  * @description: class definition for TrafficLight
6:  */
7:  /* trafficLights.h - header file for the class trafficLights */
8:
9:  #ifndef TRAFFICLIGHT_H
10: #define TRAFFICLIGHT_H
11:
12: #include "time.h"
13: #include <string>
14:
15: using namespace std;
16: /***** Class TrafficLight *****/
17:
18:
19:
20: class TrafficLight {
21:
22: public:
23:
24:     TrafficLight(Time _delay_time, char* _info);
25:     TrafficLight(Time _delay_time, char* _info, TrafficLight& _traffic_light);
26:
27:     void carWantsToCross();
28:
29:     static void setTime(Time the_time);
30:
31:     friend std::ostream& operator << (std::ostream& os, TrafficLight* ↵
traffic_light);
32:
33:     static Time global_time;
34:
35: private:
36:
37:     /* add members and operations to complete the class yourself */
38:     string color;
39:     string name;
40:     string direction;
41:     Time delay_time;
42:     TrafficLight *co_light;
43:
44: };
45:
46: #endif
47:

```

```

1:  /*
2:  * @Author: shihan
3:  * @Date: 2023-11-08 19:08:41
4:  * @version: 1.0
5:  * @description: class definition for Time
6:  */
7:  /* time.h - header file for the class Time */
8:
9:  #ifndef TIME_H
10: #define TIME_H
11:
12: #include <iostream>
13:
14: /***** Time Class *****/
15:
16: class Time {
17:
18: public:
19:
20:     Time();
21:
22:     Time(int hours, int mins, int secs);
23:
24:     void add(Time& anotherTime);
25:     /* adds seconds to seconds, minutes to minutes and
26:     hours to hours, taking into account that
27:     a day has 24 hours, an hour has 60 minutes
28:     and a minute has 60 seconds */
29:
30:     friend std::ostream& operator << (std::ostream& os, Time& theTime);
31:
32:     int compareTime(Time& anotherTime);
33:
34: private:
35:
36:     int theHour;
37:     int theMins;
38:     int theSecs;
39:
40: };
41:
42: #endif
43:

```

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)

```

1:  /*
2:  * @Author: shihan
3:  * @Date: 2023-11-08 19:08:41
4:  * @version: 1.0
5:  * @description:
6:  */
7:  /* main.cpp - for MSc Computing C++ Assessed Exercise 2. */
8:
9:  #include <iostream>
10: #include "trafficLight.h"
11: #include "time.h"
12:
13: using namespace std;
14:
15: int main() {
16:     cout << "=====\n"
17:         << "Roads open in Sleepy Town:\n"
18:         << "=====\n";
19:
20:     /* (a) initialise the time: */
21:     // First, the global clock is initialised to 0:0:0
22:
23:     Time timeZero(0, 0, 0);
24:     TrafficLight::setTheTime(timeZero);          // event (a) completed
25:
26:
27:     /* (b) create a pair of traffic lights with delays 15 minutes and 5
28:        minutes respectively: */
29:     cout << "\nA pair T1 and T2 of (slow) collaborating lights is set up:\n";
30:     char T1Name[] = "T1 (North South)";
31:     char T2Name[] = "T2 (East West)";
32:     Time delayT1(0, 15, 0);
33:     Time delayT2(0, 5, 0);
34:     TrafficLight LightT1(delayT1, T1Name);
35:     // should be &LightT1
36:     TrafficLight LightT2(delayT2, T2Name, LightT1); // event (b) completed
37:
38:
39:     /* (c)-(f) begin the simulation with 4 car crossings: */
40:     LightT1.carWantsToCross(); // event (c) completed
41:     LightT2.carWantsToCross(); // event (d) completed
42:     LightT1.carWantsToCross(); // event (e) completed
43:     LightT2.carWantsToCross(); // event (f) completed
44:
45:
46:     /* (g) create another pair of traffic lights with extra long delays of
47:        6hrs, 15mins, 44secs and 14hrs, 5mins, 57secs respectively: */
48:     cout << "\nA new pair T3 and T4 of (very slow!) collaborating lights is now
set up:\n";
49:     char T3Name[] = "T3 (North South)";
50:     char T4Name[] = "T4 (East West)";
51:     Time delayT3(6, 15, 44);
52:     Time delayT4(14, 5, 57);
53:     TrafficLight LightT3(delayT3, T3Name);
54:     TrafficLight LightT4(delayT4, T4Name, LightT3); // event (g) completed
55:
56:
57:     /* (h)-(m) continue the simulation with 6 more car crossings: */
58:     LightT3.carWantsToCross(); // event (h) completed
59:     LightT3.carWantsToCross(); // event (i) completed
60:     LightT4.carWantsToCross(); // event (j) completed
61:     LightT4.carWantsToCross(); // event (k) completed
62:     LightT3.carWantsToCross(); // event (l) completed
63:     LightT4.carWantsToCross(); // event (m) completed
64:
65:     cout << "\n=====\n"

```

```

66:         << "Roads close forever in Sleepy Town.\n"
67:         << "=====\n";
68:
69:     return 0;
70: }
71:

```

```

1:  /*
2:  * @Author: shihan
3:  * @Date: 2023-11-08 19:08:41
4:  * @version: 1.0
5:  * @description: class implementation for TrafficLight
6:  */
7:
8: #include "trafficLight.h"
9: #include "time.h"
10: #include <iostream>
11: #include <string>
12:
13: using namespace std;
14:
15: /* initialization for static variable */
16: Time TrafficLight::global_time=Time();
17:
18: /* TrafficLight constructor */
19: TrafficLight::TrafficLight(Time _delay_time, char* _
info):delay_time(_delay_time){
20:     color = "red";
21:     string info = _info;
22:     std::string delimiter = " ";
23:     name = info.substr(0, info.find(delimiter));
24:     direction = info.substr(info.find(delimiter), info.length()-1);
25: }
26:
27: /* TrafficLight constructor */
28: TrafficLight::TrafficLight(Time _delay_time, char* _info, TrafficLight& _
traffic_light):delay_time(_delay_time), co_light(&_traffic_light){
29:     color = "red";
30:     string info = _info;
31:     std::string delimiter = " ";
32:     name = info.substr(0, info.find(delimiter));
33:     direction = info.substr(info.find(delimiter), info.length()-1);
34:     // combine both lights
35:     _traffic_light.co_light=this;
36: }
37:
38: /* carWantsToCross: the logical implementation of changing light colours */
39: void TrafficLight::carWantsToCross(){
40:     cout << endl;
41:     // case 1. both lights are red, can directly change this light
42:     if(color == "red" && co_light->color == "red"){
43:         // output sample: *** at 0:0:0 a car wants to cross light T1 (North S
outh),, with colour: red
44:         cout << "*** at "<<global_time << " a car wants to cross light " << /
name
45:         << direction << ", with colour: " << color <<endl;
46:         // change the light's color to yello and green
47:         global_time.add(delay_time);
48:         // output sample: at 0:15:0 T1 (North South) changes colour to yellow
49:         color = "yellow";
50:         cout << " at "<< global_time << " "<< name << direction
51:         << " changes colour to " << color <<endl;
52:         // output sample: at 0:30:0 T1 (North South) changes colour to green
53:         global_time.add(delay_time);
54:         color = "green";
55:         cout << " at "<< global_time << " "<< name << direction
56:         << " changes colour to " << color <<endl;
57:
58:     }
59:
60:     // case 2. the light is red, the other one is green
61:     else if(color == "red" && co_light->color == "green"){
62:         cout << "*** at "<<global_time << " a car wants to cross light " << /

```

A more elegant 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).

```

name
63:         << direction << ", with colour: " << color <<endl;
64:         // co-light, to yellow
65:         global_time.add(co_light->delay_time);
66:         co_light->color="yellow";
67:         cout << " at "<< global_time << " "<< co_light->name << /
co_light->direction
68:         << " changes colour to " << co_light->color <<endl;
69:         //the-light, to yellow
70:         global_time.add(delay_time);
71:         color="yellow";
72:         cout << " at "<< global_time << " "<< name << direction
73:         << " changes colour to " << color <<endl;
74:         //co-light, to red
75:         global_time.add(co_light->delay_time);
76:         co_light->color="red";
77:         cout << " at "<< global_time << " "<< co_light->name << /
co_light->direction
78:         << " changes colour to " << co_light->color <<endl;
79:         //the-light, to green
80:         global_time.add(delay_time);
81:         color="green";
82:         cout << " at "<< global_time << " "<< name << direction
83:         << " changes colour to " << color <<endl;
84:     }
85:
86:     // case 3. the light is green
87:     else if(color == "green"){
88:         cout << "**** at "<<global_time << " a car wants to cross light " << /
name
89:         << direction << ", with colour: " << color <<endl;
90:     }
91:
92: }
93:
94: /* set the static variable: global_time to specific time */
95: void TrafficLight::setTheTime(Time the_time){
96:     TrafficLight::global_time = the_time;
97: }
98:
99: /* overwrite the cout function */
100: std::ostream& operator << (std::ostream& os, TrafficLight* traffic_light){
101:     os << "traffic light name: " << traffic_light->name << ", direction: " << /
traffic_light->direction;
102:     return os;
103: }

```

```

Traffic                                time.cpp: 1/1                        Shihan Fu - sf23:v5

1:  /*
2:  * @Author: shihan
3:  * @Date: 2023-11-08 19:08:41
4:  * @version: 1.0
5:  * @description: class implementation for Time
6:  */
7:
8: #include "time.h"
9: #include<iostream>
10: #include <string>
11:
12: using namespace std;
13:
14: /* Time constructor */
15: Time::Time():theHour(0), theMins(0), theSecs(0){
16: }
17:
18: /* Time constructor */
19: Time::Time(int hours, int mins, int secs):theHour(hours), theMins(mins), theSecs(secs){
20: }
21:
22: /* Time add */
23: void Time::add(Time& anotherTime){
24:     int seconds;
25:     int minutes;
26:     int hours;
27:
28:     seconds=(theSecs+anotherTime.theSecs)%60;
29:     minutes=(theMins+anotherTime.theMins+((theSecs+anotherTime.theSecs)/60))%60;
30:     hours=(theHour+anotherTime.theHour+(theMins+anotherTime.theMins+(theSecs+anotherTime.theSecs)/60)/60)%24;
31:
32:     theSecs=seconds;
33:     theMins = minutes;
34:     theHour = hours;
35: }
36:
37: /* overwrite the cout function */
38: std::ostream& operator << (std::ostream& os, Time& theTime){
39:     os << theTime.theHour << ":" << theTime.theMins << ":" << theTime.theSecs;
40:     return os;
41: };
42:
43: /* Time comparision */
44: int Time::compareTime(Time& anotherTime){
45:     string this_time = to_string(theHour)+":"+to_string(theMins)+":"+to_string(theSecs);
46:     string another_time = to_string(anotherTime.theHour)+":"+to_string(anotherTime.theMins)+":"+to_string(anotherTime.theSecs);
47:     return (this_time < another_time)?-1:1;
48: }
49:
50:

```

```

Traffic                                test.cpp: 1/1                        Shihan Fu - sf23:v5

1:  /*
2:  * @Author: shihan
3:  * @Date: 2023-11-08 19:08:41
4:  * @version: 1.0
5:  * @description: test of classes and basic functions
6:  */
7:
8: #include<iostream>
9: #include "time.h"
10: #include "trafficLight.h"
11:
12: using namespace std;
13:
14: int main(){
15:
16:     // time test
17:     Time timeZero(0, 0, 0);
18:     TrafficLight::setTheTime(timeZero);
19:     cout <<TrafficLight::global_time;
20:
21:     // traffic light test
22:     cout << "\nA pair T1 and T2 of (slow) collaborating lights is set up:\n";
23:     char T1Name[] = "T1 (North South)";
24:     char T2Name[] = "T2 (East West)";
25:     Time delayT1(0, 15, 0);
26:     Time delayT2(0, 5, 0);
27:     TrafficLight LightT1(delayT1, T1Name);
28:     TrafficLight LightT2(delayT2, T2Name, LightT1);
29:
30:     // function test
31:     LightT1.carWantsToCross();
32:     LightT2.carWantsToCross();
33:
34:     return 0;
35:
36: }

```

Nice to see a test function here!

Traffic

makefile: 1/1

Shihan Fu - sf23:v5

```
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.