Module 16: Operator Overloading

Learning Objectives

1. Design and implement an operator overload member function.
2. Design and implement a copy assignment operator member function.

Process Skills

1. Information processing. Extract structural patterns from sample code.
2. Critical thinking. Use prior knowledge to interpret and explain related concepts.

Please fill in the roles for each member of your team. Take a look at the description of each role to see its responsibilities. If there are only three people in the group, please assign the same person to the **Presenter** and **Reflector** role. It is a good idea to select roles that you have not recently taken.

Team name: _____ Date: _____

Role	Team Member Name
Manager. Keeps track of time and makes sure everyone contributes appropriately.	
Presenter. Talks to the facilitator and other teams.	
Reflector. Considers how the team could work and learn more effectively.	
Recorder. Records all answers and questions and makes the necessary submission.	

For virtual activities: Once you select your roles, change your Zoom name using the format and example below.

Format: *Group X: First name, Last name initial / Role*

Example: *Group 1: Paul I / Presenter*

Model 1. Operator Overloading (9 min) Start time: _____

C++ allows us to define overloaded operators for our classes. You can find links to the common operator overloads at the bottom of this website: https://en.cppreference.com/w/cpp/language/operators. This code below shows how to overload the + operator, whose member function declaration is shown below. Take note that T refers to the class containing the overloaded operator.

```
T T::operator+() const;
```

```
// duration.h
class Duration {
 public:
  Duration() : Duration(800, 900) {}
  Duration(int start_time, int end_time)
      : start_time_(start_time), end_time_(end_time) {}

  int StartTime() const {
    return start_time_;
  }
  int EndTime() const {
    return end_time_;
  }

  void SetStartTime(int start_time) { start_time_ = start_time; }
  void SetEndTime(int end_time) { end_time_ = end_time; }

  Duration operator+(const Duration& other) const {
    int new_start = start_time_;
    int new_end = end_time_;

    if (other.StartTime() < start_time_) {
      new_start = other.StartTime();
    }
    if (other.EndTime() > end_time_) {
      new_end = other.EndTime();
    }
    Duration new_duration(new_start, new_end);
    return new_duration;
  }
```

```
 private:
  int start_time_;
  int end_time_;
};
```

```
// main.cc
#include <iostream>
#include <memory>
#include "duration.h"
int main() {
 Duration dur1(900, 1000);
 Duration dur2(1100, 1130);
 // We use our operator+ overload to allow this operation
 Duration sum = dur1 + dur2;
 std::cout << sum.StartTime() << " - " << sum.EndTime() << "\n";

 return 0;
}
```

Screen output:
900 - 1130

1. Which operator did we overload in the example above? Place a check (✓) beside your answer.

   a. + ✓
   b. -
   c. <
   d. =

2. If we want to overload the < operator to allow us to perform a statement like `dur1 < dur2`, which operator overload would we implement? Place a check (✓) beside your answer.

   a. operator+
   b. operator-
   c. operator< ✓
   d. operator=

3. We see the member function declaration for operator< below. What class should we use for T if we want to create this operator overload for Duration?

```
bool T::operator <(const T2 &b) const;
```

Duration

4. Create a < operator overload for the Duration class given its member function declaration below. A Duration is < another Duration when its end time is less than the other duration's start time. For example, a duration with a start and end time from 800 - 900 is < another with 1000 - 1100. A duration with a start and end time from 800 - 900 is not < another with 830 - 930.

```
bool T::operator <(const T2 &b) const;
```

```
bool Duration::operator<(const Duration &other) const {
  return end_time_ < other.StartTime();
}
```

🛑 **STOP HERE AND WAIT FOR FURTHER INSTRUCTIONS**

Model 2. Copy Assignment Operator (15 min) Start time: _____

The member function declaration below shows the copy assignment operator overload. It is called when we assign a T object to a T variable.

```
T& T::operator =(const T2& b);
```

```
// member function added to the Duration class
Duration& operator=(const Duration& other) {
  start_time_ = other.StartTime();
  end_time_ = other.EndTime();
  return *this;
}
```

```
// main.cc
#include <iostream>
#include "duration.h"
int main() {
  Duration dur1(900, 1000);
  // We use our operator= overload to redefine this operation
  Duration dur3 = dur1;
  dur3.operator=(dur1);
  std::cout << dur3.StartTime() << " - " << dur3.EndTime() << "\n";
  return 0;
}
```

Screen output:
900 - 1000

5. Which operator are we overloading in Model 2's sample code? Place a check (✓) beside your answer.

   a. +
   b. -
   c. <
   d. = ✓

6. Which variable from the main function is passed as an argument to the operator= member function when this statement runs: `Duration dur3 = dur1;`? Place a check (✓) beside your answer.

   a. dur3
   b. dur1 ✓
   c. Duration

7. Review the body of the operator= member function. What values will be assigned to the start_time_ and end_time_ member values of dur3 when operator= is called? Place a check (✓) beside your answer.

   a. dur1 (other)'s start time and end time ✓
   b. dur3 (other)'s start time and end time
   c. 800 and 1200

Symbol Table after performing the `Duration dur3 = dur1;` assignment

Variable Name	Scope	Type	Memory address	Value
dur1	main()	Duration	0x7fffe9b0cd10	dur1: Duration start_time_ = 900 end_time_ = 1000
dur3	main()	Duration	*before assignment* 0x7fffe9b0cc08 *after assignment* 0x7fffe9b0cc08	dur3: Duration start_time_ = 900 end_time_ = 1000
this	operator=()	Duration*		0x7fffe9b0cc08

8. `this` is a special pointer available to all member functions. They refer to the object whose member function was called. In our example, the operator= member function was called on dur3. Therefore, `this` is a pointer that points to dur3. According to the symbol table, what is the value of `this`? Place a check (✓) beside your answer.

   a. 0x7fffe9b0cd10
   b. 0x7fffe9b0cc08 ✓
   c. 0x7fffe9b0cb06

9. What do we expect to get if we dereference `this` as shown in the operator= member function body: `return *this;`? Place a check (✓) beside your answer.

   a. dur3 ✓
   b. dur1
   c. a newly created Duration object

10. operator= returns a Duration& (Duration reference). Assigning a reference to a variable changes the memory address referenced by the variable. In this example, we are replacing the memory address associated with dur3 after calling the operator= member function (`Duration dur3 = dur1;`). Why is the memory address associated with dur3 (0x7fffe9b0cc08) replaced by the same memory address? Place a check (✓) beside your answer.

    a. We return a reference to *this, which refers to the dur3 object.✓
    b. The symbol table is incorrect, the new memory address should be 0x7fffe9b0cb06
    c. We return a reference to *this, which refers to the dur1 object.

11. Implement a copy assignment operator overload for the Volunteer class below.

```
#include <iostream>
#include <memory>
#include "duration.h"

class Volunteer {
public:
 Volunteer(const std::string& name, double hours_worked)
     : name_(name), hours_worked_(hours_worked) {}

 Volunteer(const std::string& name, double hours_worked,
           int start_time, int end_time)
     : Volunteer(name, hours_worked) {
   schedule_ = std::make_shared<Duration>(start_time, end_time);
 }

// TODO: Place your operator= overload here
 Volunteer& operator=(const Volunteer& other) {
   name_ = other.Name();
   hours_worked_ = other.HoursWorked();
   schedule_ = other.Schedule();
   return *this;
 }




 const std::string& Name() const {
   return name_;
 }
 double HoursWorked() const {
   return hours_worked_;
 }
```

```
  std::shared_ptr<Duration> Schedule() const {
    return schedule_;
  }

  void AssignSchedule(int start_time, int end_time) {
    schedule_ = std::make_shared<Duration>(start_time, end_time);
  }

  void DisplaySchedule() {
    std::cout << schedule_->StartTime() << " - "
              << schedule_->EndTime() << "\n";
  }

private:
  std::string name_;
  double hours_worked_;
  std::shared_ptr<Duration> schedule_;
};
```

Reflector questions

1. What was the most useful thing your team learned during this session?

2. What prior discussion helped you understand operator overloading?

3. What was the most difficult concept to understand in the copy assignment operator discussion?