DHA SUFFA UNIVERSITY
Department of Computer Science
Object Oriented Programming
CS-1002L
Spring 2020

Lab 05

Operator Overloading with its types

Objectives

- To learn about operator overloading on objects and different ways of overloading operators on object.

Operator Overloading

C++ permits us to add two variables of user-defined types with the same syntax that is applied to the basic types. This means that C++ has the ability to provide the operators with a special meaning for a data type. The mechanism of giving special meanings to an operator is known as operator overloading. The general syntax of an operator function is as follows:

Syntax

```
returnType operator op(arg_list)
{
     function body
}
```

Ways of overloading operator

1. ***non-static member function***

   Member function which read member variables of an object (on which the function is called) directly and perform operations. Uses accessor of parameter's object if needed.

Example 5.1

```
//FileName:Time.h
// '+' operator Overloading using non static member function
#ifndef TIME_H
#define TIME_H
class Time
{
     public:
            //Declaring default constructor and destructor
            Time();
            ~Time();

            //Declaring parameterized constructor
            Time(const int& hours, const int& minutes,
                 const int& seconds);

            //Declaring accessors
            int getHours() const;
            int getMinutes() const;
            int getSeconds() const;

            //Declaring member function
            void print() const;

            //Declaring + operator overloading
            Time operator+(const Time& t);

      private:
            //Declaring member variables
            int hours;
            int minutes;
            int seconds;

};
#endif
```

```
//FileName:Time.cpp
#include "Time.h"
#include <iostream>

//Defining default constructor and destructor
Time :: Time() : hours(0), minutes(0), seconds(0) {}

Time :: ~Time(){}

//Defining parameterized constructor
```

```
Time :: Time(const int& hours, const int& minutes,
             const int& seconds)
                  : hours(hours), minutes(minutes),
                    seconds(seconds) {}

//Defining print function to print the member variables
void Time :: print() const {
    std::cout << "Time is " << hours << ":" << minutes
              << ":" << seconds << std::endl;
}

//Defining accessors
int Time :: getHours() const {
    return hours;
}

int Time :: getMinutes() const {
    return minutes;
}

int Time :: getSeconds() const {
    return seconds;
}

//Overloading + operator
Time Time :: operator+(const Time& t){

    int newHours, newMins, newSecs;

    newHours = (this->hours + t.getHours());
    newMins = (this->minutes + t.getMinutes());
    newSecs = (this->seconds + t.getSeconds());

    if(newSecs >= 60) {
        newSecs %= 60;
        ++newMins;
    }

    if(newMins >= 60) {
        newMins %= 60;
        ++newHours;
    }

    if(newHours >= 24) {
        newHours %= 24;
    }

    Time updatedTime(newHours, newMins, newSecs);

    return updatedTime;
}
```

```
//Filename:main.cpp
#include<iostream>
#include "Time.h"

int main() {

    Time t1(13, 45, 23);
    Time t2(6, 23, 41);

    Time t3 = t1 + t2;
    t3.print();
}
```

Output

```
Time is 20:9:4
```

Similarly we can overload '-', 'x', '/', '%' operators.

Lab Task 5.1

Overload '-' operator using non static member function on class Time.

2. ***non-member friend function***

   Friend function which read the member variables directly of all the objects (i.e., object on which it is called as well as object(s) in parameter) and perform operations.

Example 5.2

```
//FileName:Time.h
// '-' operator overloading using friend function
#ifndef TIME_H
#define TIME_H
class Time
{
      public:
            //Declaring default constructor and destructor
            Time();
            ~Time();

            //Declaring parameterized constructor
            Time(const int& hours, const int& minutes,
                  const int& seconds);

            //Declaring member function
            void print() const;

            //Declaring - operator overloading
           //using friend function
            friend Time operator-(const Time& t1,
                                  const Time& t2);
      private:
            //Declaring member variables
            int hours;
            int minutes;
            int seconds;
};
#endif
```

```
//FileName:Time.cpp
#include "Time.h"
#include <iostream>

//Defining default constructor and destructor
Time :: Time() : hours(0), minutes(0), seconds(0) {}

Time :: ~Time(){}

//Defining parameterized constructor
Time :: Time(const int& hours, const int& minutes,
             const int& seconds)
                   : hours(hours), minutes(minutes),
                     seconds(seconds) {}

//Defining print function to print the member variables
void Time :: print() const {
    std::cout << "Time is " << hours << ":" << minutes << ":"
              << seconds << std::endl;
}

Time operator-(const Time& t1, const Time& t2){

    int newHours, newMins, newSecs;

    newHours = (t1.hours - t2.hours);
    newMins = (t1.minutes - t2.minutes);
    newSecs = (t1.seconds - t2.seconds);

    if(newSecs < 0) {
        newSecs += 60;
        --newMins;
    }

    if(newMins < 0) {
        newMins += 60;
        --newHours;
    }

    if(newHours < 0) {
        newHours += 24;
    }

    Time updatedTime(newHours, newMins, newSecs);

    return updatedTime;
}
```

```
//Filename:main.cpp
#include<iostream>
#include "Time.h"

int main() {

    Time t1(13, 45, 23);
    Time t2(6, 23, 41);

    Time t3 = t1 - t2;
    t3.print();
}
```

Output

```
Time is 7:21:42
```

Similarly we can overload ‘+’, ‘x’, ‘/’, ‘%’ operators.

Lab Task 5.2

`Overload ‘+’ operator using non-static friend function`

3. ***non-member non-friend function***

   Function which relies only on accessor of object to retrieve value and perform operations.

Example 5.3

```
//FileName:Time.h
#ifndef TIME_H
#define TIME_H
class Time
{
     public:
          //Declaring default constructor and destructor
          Time();
          ~Time();

          //Declaring parameterized constructor
          Time(const int& hours, const int& minutes,
               const int& seconds);

          int getHours() const;
          void setHours(const int& hours);

          //Declaring member function
          void print() const;

     private:
          //Declaring member variables
          int hours;
          int minutes;
          int seconds;
};
#endif
```

```
//FileName:Time.cpp
#include "Time.h"
```

```
#include <iostream>


//Defining default constructor and destructor
Time :: Time() : hours(0), minutes(0), seconds(0) {}

Time :: ~Time(){}

//Defining parameterized constructor
Time :: Time(const int& hours, const int& minutes,
           const int& seconds)
                : hours(hours), minutes(minutes),
                     seconds(seconds) {}

int Time :: getHours() const {
    return this->hours;
}


void Time :: setHours(const int& hours) {
    this->hours = hours;
}

//Defining print function to print the member variables
void Time :: print() const {
     std::cout << "Time is " << hours << ":" << minutes << ":"
           << seconds << std::endl;
}

void operator+=(Time& t1, int hours){

    t1.setHours((t1.getHours() + hours) % 24);
}
```

```
//Filename:main.cpp
#include<iostream>
#include "Time.h"
void operator+=(Time& t1, int hours);
int main() {

     Time t1(13, 45, 23);
     t1 += 4;
     t1.print();
}
```

Output

```
Time is 17:45:23
```

Similarly we can overload '-=', 'x=', '/=', '%=' operators.

Lab Task 5.3

`Overload '-=' operator using non-static friend function`

Overloading Unary Operators

Unary operators act on only one operand. Examples of unary operators are the increment and decrement operators ++ and -- .

Example 5.4

```
//FileName:Time.h
//Postfix operator overloading
#ifndef TIME_H
#define TIME_H
class Time
{
      public:
            //Declaring default constructor and destructor
            Time();
            ~Time();

            //Declaring parameterized constructor
            Time(const int& hours, const int& minutes,
                 const int& seconds);

            //Declaring postfix increment operator
            Time operator++(int);

           //Declaring member function
           void print() const;

      private:
            //Declaring member variables
            int hours;
            int minutes;
            int seconds;
};
#endif
```

```
//FileName:Time.cpp
#include "Time.h"
#include <iostream>

//Defining default constructor and destructor
Time :: Time() : hours(0), minutes(0), seconds(0) {}

Time :: ~Time(){}
```

```
//Defining parameterized constructor
Time :: Time(const int& hours, const int& minutes,
           const int& seconds)
                : hours(hours), minutes(minutes),
                     seconds(seconds) {}

//Defining print function to print the member variables
void Time :: print() const {
      std::cout << "Time is " << hours << ":" << minutes << ":"
           << seconds << std::endl;
}

Time Time::operator++(int){

    Time tempTime(this->hours, this->minutes, this->seconds);
    this->hours = (this->hours + 1) % 24;

    return tempTime;
}
```

```
//Filename:main.cpp
#include<iostream>
#include "Time.h"
int main() {

     Time t1(13, 45, 23);
     Time t2 = t1++;

     t2.print();
     t1.print();
}
```

Output

```
Time is 13:45:23
Time is 14:45:23
```

Lab Task 5.4

`Overload '--' prefix operator using non static member function.`

Overloading Binary Operators

Binary operator overloading acts on two operands. If you want to perform any binary operation like add, subtract, divide, multiply, compare etc. on two objects, you need to call member functions written to perform these operations. (Already covered in Example 6.1 and 6.2)

Overloading Relational Operator

There are various relational operators supported by C++ language like (`<`, `>`, `<=`, `>=`, `==`, etc.) which can be used to compare C++ built-in data types. You can overload any of these operators, which can be used to compare the objects of a class.

Example 5.5

```
//FileName:Time.h
//Overloading == operator
#ifndef TIME_H
#define TIME_H
class Time
{
     public:
           //Declaring default constructor and destructor
           Time();
           ~Time();

           //Declaring parameterized constructor
           Time(const int& hours, const int& minutes,
               const int& seconds);

           //Declaring member function
```

```
            void print() const;

             //Declaring == relational operator
             bool operator==(const Time& t) const;

      private:
            //Declaring member variables
            int hours;
            int minutes;
            int seconds;
};
#endif
```

```
//FileName:Time.cpp
#include "Time.h"
#include <iostream>

//Defining default constructor and destructor
Time :: Time() : hours(0), minutes(0), seconds(0) {}

Time :: ~Time(){}

//Defining parameterized constructor
Time :: Time(const int& hours, const int& minutes,
             const int& seconds)
                  : hours(hours), minutes(minutes),
                    seconds(seconds) {}

//Defining print function to print the member variables
void Time :: print() const {
    std::cout << "Time is " << hours << ":" << minutes << ":"
              << seconds << std::endl;
}

bool Time::operator==(const Time& t){
    return t.hours == this->hours
```

```
            && t.minutes == this->minutes
                 && t.seconds == this->seconds;
}
```

```
//Filename:main.cpp
#include<iostream>
#include "Time.h"
int main() {

     Time t1(13, 45, 23);
     Time t2 = t1;

     std::cout << (t1 == t2) << std::endl;
     t1.print();
     t2.print();
}
```

Output

```
1
Time is 13:45:23
Time is 13:45:23
```

Insertion extraction operator

The insertion operator << is the one we use for output, as in:

```
std :: cout << "This is output" << std :: endl;
```

The extraction operator >> is the one we use for input, as in:

```
 std :: cin >> x;
```

Example 5.6

```
//FileName:Time.h
//Overloading insertion and extraction op using friend function
```

```
#include <iostream>
#ifndef TIME_H
#define TIME_H
class Time
{
	public:
		//Declaring default constructor and destructor
		Time();
		~Time();

		//Declaring parameterized constructor
		Time(const int& hours, const int& minutes,
		     const int& seconds);

		//Declaring member function
		void print() const;

		//Declaring stream insertion operator
		friend std::ostream& operator<<(std::ostream& out,
		                                Time& t);

		friend std::istream& operator>>(std::istream& in,
		                                Time& t);

	private:
		//Declaring member variables
		int hours;
		int minutes;
		int seconds;
};
#endif
```

```
//FileName:Time.cpp
#include "Time.h"
#include <iostream>


//Defining default constructor and destructor
```

```
Time :: Time() : hours(0), minutes(0), seconds(0) {}

Time :: ~Time(){}



//Defining parameterized constructor
Time :: Time(const int& hours, const int& minutes,
             const int& seconds)
                  : hours(hours), minutes(minutes),
                    seconds(seconds) {}

//Defining print function to print the member variables
void Time :: print() const {
    std::cout << "Time is " << hours << ":" << minutes
              << ":" << seconds << std::endl;
}

//Overloading stream insertion operator
std::ostream& operator<<(std::ostream& out, Time& t) {

     out << "Time is " << t.hours << ":" << t.minutes
         << ":" << t.seconds;

     return out;
}

//Overloading stream extraction operator
std::istream& operator>>(std::istream& in, Time& t) {

     std::cout << "Enter time in format ->
                hours minutes seconds: " << std::endl;
     in >> t.hours >> t.minutes >> t.seconds;

     return in;
}
```

```
//Filename:main.cpp
#include<iostream>
#include "Time.h"
int main() {

    Time t1;
    std::cin >> t1;
    std::cout << t1;
}
```

Output

```
Enter time in format -> hours minutes seconds:
23 15 45
Time is 23:15:45
```

Lab Tasks

Task 5.5(Home Assignment)
Create a class Student with member variables defined below and overload insertion and extraction operators on Student class.

1. name
2. age
3. rollNumber

Task 5.6(Home Assignment)
Overload += operator on above class and add value to the age variable.

Eg,

Student s(“Ali”, 20, “cs132011”);

s += 3;

s.getAge() -> should print 23

Task 5.7(Home Assignment)

DHA SUFFA UNIVERSITY
Department of Computer Science
Object Oriented Programming
CS-1002L
Spring 2020

Write a class Time Duration which represents time duration. The class should have three fields for hours, minutes and seconds. It should have a constructor to initialize the hours, minutes and seconds. A function printDuration() to print the current time. Overload the following operators: - Plus operator (+) to add two Time Duration objects. (Maintaining minutes and seconds <=60) - Operator < to compare two Time Duration objects. (Overload these Operator using friend function and non member non friend functions)

Submission Instructions

1.Number your solution folders as question number e.g. Q1, Q2, etc. (Q is in upper case)

2.Every folder should contain three files (one header, one implementation and one driver)

3.Create a new folder named cs152abc where abc is your 3 digit roll #. e.g. cs152111.

4.Copy all the project folders into this folder.

5.Now make sure a zip file named cs152abc.zip is created e.g. cs152111.zip

6. Upload the assignment solution on LMS under the assignment named Lab 06 Assignment – XX, where XX is your section name