**Experiment No. 4**

**Title:** **Implementation of operator overloading**

**Batch: B2 Roll No.: 16010421091 Experiment No.: 4**

**Aim**: Write a C++ program to create a class 'TIME' to store a time value considering 24 hrs time format, in terms of hours(hrs) and minutes(mins). Provide appropriate empty and parameterized constructor. Define function display( ) to display time in form of hrs:mins. Overload following operators as directed.

(a) + : to add two time values – use member function

(b) - : to calculate difference between two time values – use friend function

(c) prefix increment(++) : to add 1 minute to time value – use member function

(d) prefix decrement(-- ): to subtract 1 minute from time value - use friend function.

While calculating the final time value for each of the above function, perform minutes to hour conversion if the minute value is greater than or equal to 60 minutes

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Resources needed: Text Editor, C++ compiler**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

### Theory:

**Operator Overloading**

Polymorphism, a Greek term, means the ability to take more than on form. An operation may exhibit different behavior is different instances. The behavior depends upon the types of data used in the operation. For example, consider the operation of addition. For two numbers, the operation will generate a sum. If the operands are strings, then the operation would produce a third string by concatenation. The process of making an operator to exhibit different behaviors in different instances is known as operator overloading.

For defining an additional task to an operator, we must mention what is means in relation to the class to which it (the operator) is applied. The operator function helps us in doing so. The syntax of declaration of an Operator function is as follows:

return-type operator operator\_symbol(argument list);

Operator overloading can be defined using a member function or a friend function.

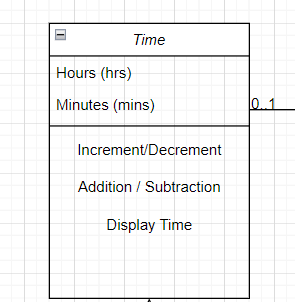
For a Unary Operator Y with operand x, Y x can be interpreted as either

x.operator Y ( ) using member function or operator Y (x) using friend function.

For a Binary Operator X with operands ‘a’ and ‘b’, a X b can be interpreted as either

operator X (b) using member function or operator X (a, b) using friend function .

Class Diagram:



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Results: (Program with snapshot of output)**

**#include <iostream>**

**using namespace std;**

**class Time**

**{**

**int hrs, mins;**

**public:**

**Time()**

**{**

**hrs = mins = 0;**

**}**

**Time(int i, int j)**

**{**

**hrs= i;**

**mins= j;**

**}**

**Time operator + (const Time &t2)**

**{**

**Time temp;**

**temp.hrs = hrs + t2.hrs;**

**temp.mins = mins + t2.mins;**

**return temp;**

**}**

**Time operator ++ ()**

**{**

**Time temp;**

**temp.mins = ++mins;**

**return temp;**

**}**

**friend Time operator -- (Time & op1, int not\_used);**

**friend Time operator - (Time t1, Time t2);**

**void Display();**

**};**

**Time operator -- (Time & op1, int not\_used)**

**{**

**Time temp = op1;**

**op1.mins--;**

**return temp;**

**}**

**Time operator - (Time t1, Time t2)**

**{**

**Time temp;**

**temp.hrs = t1.hrs - t2.hrs;**

**temp.mins = t1.mins - t2.mins;**

**return temp;**

**}**

**void Time::Display()**

**{**

**int addhr;**

**if(mins>=60)**

**{**

**addhr = mins/60;**

**hrs = addhr + hrs;**

**mins = mins - addhr\*60;**

**}**

**cout << hrs<< ":";**

**cout << mins<< ":"<<endl;**

**}**

**int main()**

**{**

**Time t1(12, 22);**

**t1.Display();**

**++t1;**

**t1.Display();**

**t1--;**

**t1.Display();**

**Time t2(23,45);**

**Time t3 = t1 + t2;**

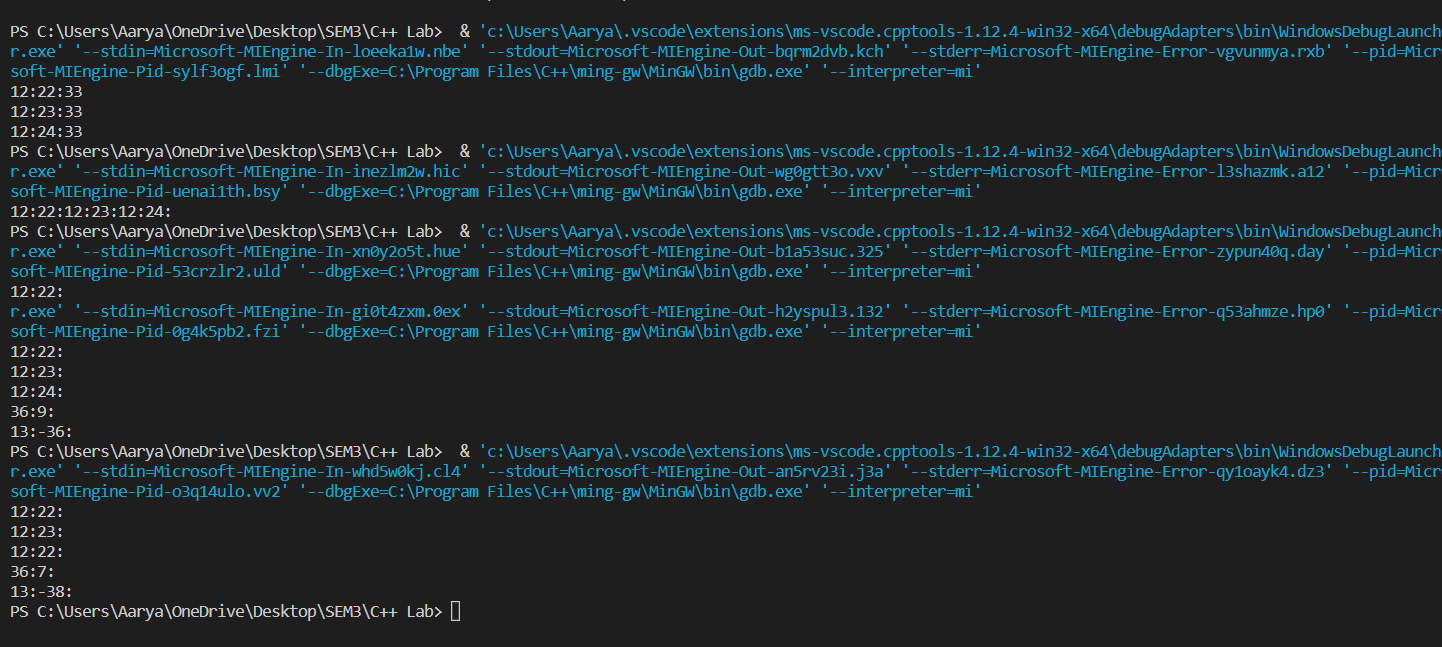
**t3.Display();**

**Time t4 = t3 - t2;**

**t4.Display();**

**return 0;**

**}**

****

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Test Cases (minimum 5 test cases required):**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sr. No.** | **Sample Input** | **Sample Output** | **Description** | **Test Case Type (general/special)** | **Pass/Fail** |
| **1.** | **12:22++** | **12:23** |  | **General** | **Pass** |
| **2.** | **12:23++** | **12:24** |  | **General** | **Pass** |
| **3.** | **12:24+23:45** | **36:9** |  | **General** | **Pass** |
| **4.** | **36:9 – 23:45** | **13:-38** |  | **General** | **Pass** |
| **5.** | **36:9--** | **36:8** |  | **General** | **Pass** |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Questions:**

1. List the rules for operator overloading.

In C++, following are the general rules for operator overloading.

1) Only built-in operators can be overloaded. New operators can not be created.  
2) [Arity of the operators](http://en.wikipedia.org/wiki/Arity)cannot be changed.  
3) Precedence and associativity of the operators cannot be changed.  
4) Overloaded operators cannot have default arguments except the function call operator () which can have default arguments.  
5) Operators cannot be overloaded for built in types only. At least one operand must be used defined type.  
6) Assignment (=), subscript ([]), function call (“()”), and member selection (->) operators must be defined as member functions

1. Write function definition and function call for overloading postfix increment (++) operator for TIME class using
   1. member function
   2. friend function

Member:

Time operator ++ (Time & op1, int not\_used)

{

    Time temp = op1;

    op1.mins--;

    return temp;

}

Friend:

friend Time operator ++ (Time & op1, int not\_used);//In Class

Time operator ++ (Time & op1, int not\_used)

{

    Time temp = op1;

    op1.mins--;

    return temp;

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Outcomes:**

Fundamentals of Operator Overloading, Restrictions On Operators Overloading, Operator Functions as Class Members vs. as Friend Functions, Overloading, <> Overloading Unary Operators, Overloading Binary Operators.

**Conclusion: (Conclusion to be based on the outcomes achieved)**

We can conclude that we have learnt how to implement operator overloading..

**Grade: AA / AB / BB / BC / CC / CD /DD**

Signature of faculty in-charge with date

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**References:**

**Books/ Journals/ Websites:**

1. E Balagurusamy, Object oriented Programming with C++, Tata McGraw-Hill, 8th Edition September 2020
2. Herbert Schildt, C++: The Complete Reference, McGraw Hill Education, 4th edition, July 2017
3. Jeff Langr, Modern C++ Programming with Test-Driven Development : Code Better,Sleep Better, O′Reilly, 1st edition, November 2013
4. <https://docs.microsoft.com/en-us/cpp/cpp/?view=msvc-170>