**Lab6 Topic: “Operator Overloading and How to add, subtract, divide, …, and to use other operators with objects”**

**Concept/Rule 1**: If object1 and object2 have been instantiated, then the

statement

object1= object2;

automatically calls a default copy constructor.

As a result, both objects point the same location and object1

contains the same value as object2.

**Concept/Rule 2**: If there is an operator = member function of a class, the

statement

object1= object2;

calls the operator = member function and object2 is passed to

the function.

**Concept/Rule 3**: A programmer defined constructor that contains a class object

as a parameter is called a copy constructor. The programmer-

defined copy constructor copies one object to another. See its

function prototype below:

ClassName::ClassName(ClassName ObjectTocopied );

**Concept/Rule 4**: To be able to perform a math operation on objects such as

Object3 = Object1 + Object2; ,

the class needs to overload the + operator by defining a

member function operator + (Classname rightObject ) that

returns Classname as the return type. The function takes the

right object of the + sign from the function call as the

parameter. Based on Object3 = Object1 + Object2; statement,

the operator + is called and Object2 is passed to the function.

**Concept/Rule 5**: To be able to perform a comparison on multiple objects such

as if ( Object1 <= Object2)

cout << “ Object1 ‘s data member(s) contain larger values” ; ,

the class needs to overload the <= operator by defining a

member function operator<= (Classname rightObject ) that

returns Classname as the return type. The function takes the

right object of the <= operator from the function call as the

parameter.

**Concept/Rule 6**: To be able to perform a cin >> object; the class needs to

overload the extraction operator >> by defining a friend

function operator >> (istream para1, Classname para2)

that returns istream as the return type. The function takes two

parameters: the cin and the object from the function call.

**Your Tasks:**

1. Compile the following program, lab6program.cpp. Display a screnshot of the program output.

#include <iostream>

using namespace std;

class Fractionclass

{ friend istream &operator >> (istream &,Fractionclass & ); // function prototype

public:

Fractionclass(); // default construtor

Fractionclass operator + (Fractionclass ); // function prototype.

bool operator < (Fractionclass); // function prototype.

private:

int numerator;

int demominator;

};

Fractionclass::Fractionclass()

{

numerator = 0;

demominator = 0;

}

istream &operator >> (istream &input ,Fractionclass &object)

{cout << "In the operator >> function " << endl;

input >> object.numerator;

input >> object.demominator;

return input;

}

Fractionclass Fractionclass::operator +(Fractionclass rightobject )

{ cout << "Right object passed/ its numerator is " << rightobject.numerator << endl;

cout << "Right object passed/ its denominator is " << rightobject.demominator << endl;

Fractionclass tempobject;

tempobject.numerator = ((this->numerator \* rightobject.demominator) +

(this->demominator\*rightobject.numerator));

tempobject.demominator = (this->demominator\*rightobject.demominator);

cout << tempobject.numerator <<"/" << tempobject.demominator << endl;

return tempobject;

}

bool Fractionclass::operator <(Fractionclass rightobject )

{ cout <<"In the operator < member function "<< endl;

if ( this->numerator == rightobject.numerator && this->demominator == rightobject.demominator)

return true;

else

return false;

}

int main()

{

Fractionclass fraction1, fraction2,fraction3, fraction4;

cout << "Enter a numerator and a demominator for fraction1 : " << endl;

cin >> fraction1;

cout << "Enter a numerator and a demominator for fraction2 : " << endl;

cin >> fraction2;

// cout << fraction1; //To be able to do this, you need to overload the

//insertion << operator.

//Hints: The function is similar to operator >> friend function.

//Use the ostream instead the istream. Use the << instead of >>

//in the function body.The function should display the fraction in

//numerator / demomonator format. For example if fraction1 contains

// 2 as its numerator and 3 as its demominator, then the output is 2/3.

//cout << fraction2;

fraction3 = fraction1 + fraction2;

//cout << fraction3; // If fraction1 contains 2/3 and fraction2 contains 1/5 then

// fraction3 contains 13/15.

//fraction4 = fraction1 - fraction2;

//cout << fraction4;

if (fraction1 < fraction2)

cout << "Fraction1 is less than fraction2." << endl;

else

cout << "Fraction1 is not less than fraction2." << endl;

cin.ignore();

cin.get();

}

1. Make a change to lab6program.cpp as follows:

-Modify the program so that the operator >> is not a friend function. Eliminate the word “friend” then modify the definition:

istream Fractionclass::&operator >> (same parameters)

{same statements;

}

Does it work?

*No*

Is there any error message?

*Multiple errors , such as illegal token, sytax error, and not able to access private member.*

How much do you know about a friend function?

*That a friend has access to a class’s private members.*

1. Answer the following questions based on the code in main():
   1. Does the statement cin >> fraction1; call the operator >> function?

*Yes*

* 1. If you want to add a statement cout << fraction1; , do you need to

implement the operator << function?

*Yes, after you overload the insertion << operator.*

* 1. Does the statement if (fraction1 < fraction2) call the

operator < function?

*Yes*

* 1. If you want to add a statement if (fraction1 > fraction2) , do

you need to implement the operator > function?

*Yes*

1. Read commented lines in the main program. If you uncomment them,they will not work until you implement appropriate functions in the class. Modify the program so that when you uncomment them, the program works. Save your program as **Assessment6ProjectYourname.cpp.** Display the revised program and a screenshot of the output.
2. State what the concepts and rules that have been proven. Provide evidence.

*Concept/Rule 1 Line 40 –*

Fractionclass tempobject;

tempobject.numerator = ((this->numerator \* rightobject.demominator) +

(this->demominator\*rightobject.numerator));

tempobject.demominator = (this->demominator\*rightobject.demominator);

*Concept/Rule 1 Line 52 –*

Fractionclass tempobject;

tempobject.numerator = ((this->numerator \* rightobject.demominator) +

(this->demominator\*rightobject.numerator));

tempobject.demominator = (this->demominator\*rightobject.demominator);

*Concept/Rule 3 Line 36 –*

Fractionclass Fractionclass::operator +(Fractionclass rightobject )

*Concept/Rule 3 Line 48 –*

Fractionclass Fractionclass::operator -(Fractionclass rightobject )

*Concept/Rule 4 Line 83 –* fraction3 = fraction1 + fraction2;

*And Line 9 –*

Fractionclass operator + (Fractionclass ); // function prototype.

*And Line 36-47 –*

Fractionclass Fractionclass::operator +(Fractionclass rightobject )

{ cout << "Right object passed/ its numerator is " << rightobject.numerator << endl;

cout << "Right object passed/ its denominator is " << rightobject.demominator << endl;

Fractionclass tempobject;

tempobject.numerator = ((this->numerator \* rightobject.demominator) +

(this->demominator\*rightobject.numerator));

tempobject.demominator = (this->demominator\*rightobject.demominator);

cout << tempobject.numerator <<"/" << tempobject.demominator << endl;

return tempobject;

}

*Concept/Rule 6 Line 71&72 –*

cout << "Enter a numerator and a demominator for fraction1 : " << endl;

cin >> fraction1;

*Concept/Rule 6 Line 73&74 –*

cout << "Enter a numerator and a demominator for fraction2 : " << endl;

cin >> fraction2;

*Concept/Rule 6 of the above are possible because of this starting at Line 4,5-*

class Fractionclass

{friend istream &operator >> (istream &,Fractionclass&);//function prototype

*And this at Line 23-27 -*

istream &operator >> (istream &input ,Fractionclass &object)

{cout << "In the operator >> function " << endl;

input >> object.numerator;

input >> object.demominator;

return input;}

**What To Turn In**

Submit:

**Lab6ResultYourname.doc** and **Assessment6ProjectYourname.cpp**

Grading Rubric:

|  |  |
| --- | --- |
| Lab6 Completion of Tasks 1- 3 | 9 points |
| Assessment6ProjectYourname.cpp  done in Task #4 . The program runs.  Answers to Task #5 | 5 points  6 points |