**Friend Function and Friend Class**

**LAB #** **07**

**Fall 2019**

**CSE208L Object Oriented Programming Lab**

Submitted by: **Shah Raza**

Registration No. : **18PWCSE1658**

Class Section: **B**

“On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work.”

Student Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Submitted to:

**Engr. Sumayyea Salahuddin**

January 6, 2020

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

**Objectives of the Lab:**

Objectives of the lab are to:

# Understand the difference between a regular function and a friend function.

# Explain the concept of friend Function.

* Develop a friend Function.

# Activity # 01

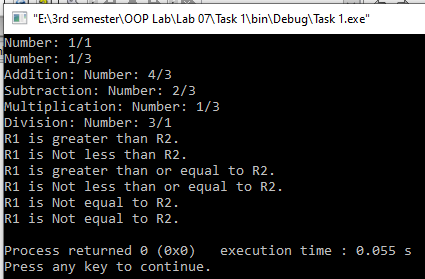
**Title:**

Create a class RationalNumber and some Friend Functions.

**In C++**

**Source code: Output:**

#include <iostream>



using namespace std;

class RationalNumber

{

private:

int num;

int den;

public:

RationalNumber(): num(0),den(0){}

RationalNumber(int a, int b)

{

if(b<=0)

cout<<"Denominator can not be 0 or negative.";

else

{

num=a,den=b;

}

int n=num,d=den;

while(n!=d)

{

if(n>d)

n-=d;

else

d-=n;

}

num/=n;

den/=n;

}

void ShowRN()

{

cout<<"Number: "<<num<<"/"<<den<<endl;

}

friend RationalNumber operator+(RationalNumber ob1, RationalNumber ob2);

friend RationalNumber operator-(RationalNumber ob1, RationalNumber ob2);

friend RationalNumber operator\*(RationalNumber ob1, RationalNumber ob2);

friend RationalNumber operator/(RationalNumber ob1, RationalNumber ob2);

friend bool operator>(RationalNumber ob1, RationalNumber ob2);

friend bool operator<(RationalNumber ob1, RationalNumber ob2);

friend bool operator>=(RationalNumber ob1, RationalNumber ob2);

friend bool operator<=(RationalNumber ob1, RationalNumber ob2);

friend bool operator==(RationalNumber ob1, RationalNumber ob2);

friend bool operator!=(RationalNumber ob1, RationalNumber ob2);

};

RationalNumber operator+(RationalNumber ob1, RationalNumber ob2)

{

RationalNumber temp;

temp.num = (ob1.num\*ob2.den)+ (ob1.den\*ob2.num);

temp.den = ob1.den\*ob2.den;

return temp;

}

RationalNumber operator-(RationalNumber ob1, RationalNumber ob2)

{

RationalNumber temp;

temp.num = (ob1.num\*ob2.den)- (ob1.den\*ob2.num);

temp.den = ob1.den\*ob2.den;

return temp;

}

RationalNumber operator\*(RationalNumber ob1, RationalNumber ob2)

{

RationalNumber temp;

temp.num = ob1.num\*ob2.num;

temp.den = ob1.den\*ob2.den;

return temp;

}

RationalNumber operator/(RationalNumber ob1, RationalNumber ob2)

{

RationalNumber temp;

temp.num = ob1.num\*ob2.den;

temp.den = ob1.den\*ob2.num;

return temp;

}

bool operator>(RationalNumber ob1, RationalNumber ob2)

{

if (ob1.num/ob1.den>ob2.num/ob2.den)

return true;

else

return false;

}

bool operator<(RationalNumber ob1, RationalNumber ob2)

{

if (ob1.num/ob1.den<ob2.num/ob2.den)

return true;

else

return false;

}

bool operator>=(RationalNumber ob1, RationalNumber ob2)

{

if (ob1.num/ob1.den>=ob2.num/ob2.den)

return true;

else

return false;

}

bool operator<=(RationalNumber ob1, RationalNumber ob2)

{

if (ob1.num/ob1.den<=ob2.num/ob2.den)

return true;

else

return false;

}

bool operator==(RationalNumber ob1, RationalNumber ob2)

{

if (ob1.num/ob1.den==ob2.num/ob2.den)

return true;

else

return false;

}

bool operator!=(RationalNumber ob1, RationalNumber ob2)

{

if (ob1.num/ob1.den!=ob2.num/ob2.den)

return true;

else

return false;

}

int main()

{

RationalNumber r1(1,1),r2(1,3);

r1.ShowRN();

r2.ShowRN();

RationalNumber r3=r1+r2;

cout<<"Addition: ";

r3.ShowRN();

r3=r1-r2;

cout<<"Subtraction: ";

r3.ShowRN();

r3=r1\*r2;

cout<<"Multiplication: ";

r3.ShowRN();

r3=r1/r2;

cout<<"Division: ";

r3.ShowRN();

bool result= r1>r2;

if(result)

cout<<"R1 is greater than R2.\n";

else

cout<<"R1 is Not greater than R2.\n";

result=r1<r2;

if(result)

cout<<"R1 is less than R2.\n";

else

cout<<"R1 is Not less than R2.\n";

result=r1>=r2;

if(result)

cout<<"R1 is greater than or equal to R2.\n";

else

cout<<"R1 is Not greater than or equal to R2.\n";

result=r1<=r2;

if(result)

cout<<"R1 is less than or equal to R2.\n";

else

cout<<"R1 is Not less than or equal to R2.\n";

result=r1==r2;

if(result)

cout<<"R1 is equal to R2.\n";

else

cout<<"R1 is Not equal to R2.\n";

result=r1!=r2;

if(result)

cout<<"R1 is Not equal to R2.\n";

else

cout<<"R1 is Equal to R2.\n";

return 0;

}

# Activity # 02

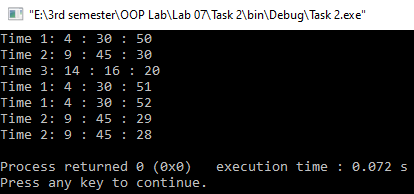
**Title:**

Create a class Time and some friend Functions.

**In C++**

**Source code: Output:**

#include <iostream>



using namespace std;

class Time

{

private:

int hours,mins,sec;

public:

Time(): hours(0),mins(0),sec(0){}

Time(int h, int m,int s)

{

hours=(h>23)?0:h;

mins=(m>59)?0:m;

sec=(s>59)?0:s;

}

void Show()

{

cout<<hours<<" : "<<mins<<" : "<<sec<<endl;

}

friend Time operator+(Time ob1, Time ob2);

friend Time operator++(Time &obj);

friend Time operator++(Time &obj, int NotUsed);

friend Time operator--(Time &obj);

friend Time operator--(Time &obj, int NotUsed);

};

Time operator--(Time &obj, int NotUsed)

{

Time temp;

obj.sec--;

if(obj.sec<0)

{

obj.sec+=60;

obj.mins-=1;

}

if(obj.mins<0)

{

obj.mins+=60;

obj.hours-=1;

}

return temp;

}

Time operator--(Time &obj)

{

obj.sec--;

if(obj.sec<0)

{

obj.sec+=60;

obj.mins-=1;

}

if(obj.mins<0)

{

obj.mins+=60;

obj.hours-=1;

}

return obj;

}

Time operator++(Time &obj, int NotUsed)

{

Time temp=obj;

obj.sec++;

if(obj.sec>59)

{

obj.sec-=60;

obj.mins+=1;

}

if(obj.mins>59)

{

obj.mins-=60;

obj.hours+=1;

}

return temp;

}

Time operator++(Time &obj)

{

obj.sec++;

if(obj.sec>59)

{

obj.sec-=60;

obj.mins+=1;

}

if(obj.mins>59)

{

obj.mins-=60;

obj.hours+=1;

}

return obj;

}

Time operator+(Time ob1, Time ob2)

{

Time temp;

temp.sec=ob1.sec+ob2.sec;

temp.mins=ob1.mins+ob2.mins;

temp.hours=ob1.hours+ob2.hours;

if(temp.sec>59)

{

temp.sec-=60;

temp.mins+=1;

}

if(temp.mins>59)

{

temp.mins-=60;

temp.hours+=1;

}

return temp;

}

int main()

{

Time t1(4,30,50),t2(9,45,30);

cout<<"Time 1: ";

t1.Show();

cout<<"Time 2: ";

t2.Show();

Time t3=t1+t2;

cout<<"Time 3: ";

t3.Show();

cout<<"Time 1: ";

++t1;

t1.Show();

t1++;

cout<<"Time 1: ";

t1.Show();

--t2;

cout<<"Time 2: ";

t2.Show();

t2--;

cout<<"Time 2: ";

t2.Show();

return 0;

}