## Assignment – 27

**A Job Ready Bootcamp in C++, DSA and IOT**

***Operator overloading and friend function***

1. Define a class Complex with appropriate instance variables and member functions.

Define following operators in the class:

a. +

b. -

c. \*

d. ==

#include <iostream>

using namespace std;

class Complex

{

private:

int a,b;

public:

Complex operator+(Complex A)

{

Complex temp;

temp.a=a+A.a;

temp.b=b+A.b;

return temp;

}

Complex operator-(Complex B)

{

Complex temp;

temp.a=a-B.a;

temp.b=b-B.b;

return temp;

}

Complex operator\*(Complex C)

{

Complex temp;

temp.a=a\*C.a;

temp.b=b\*C.b;

return temp;

}

int operator==(Complex y)

{

if(a==y.a&&b==y.b)

return 1;

else

return 0;

}

void setData(int x, int y)

{

a=x;

b=y;

}

void showData()

{

cout<<"Real = "<<a<<" Img = "<<b<<endl;

}

};

int main()

{

Complex c1,c2,c3,c4,c5;

int c;

c1.setData(5,8);

c2.setData(4,2);

c3=c1+c2;

cout<<"\t\tAddition..........."<<endl;

c1.showData();

c2.showData();

c3.showData();

cout<<"\t\tSubtraction........."<<endl;

c5=c1-c2;

c1.showData();

c2.showData();

c5.showData();

cout<<"\t\tMultiplication......"<<endl;

c4=c3\*c2;

c2.showData();

c3.showData();

c4.showData();

c=c4==c3;

if(c){

c4.showData();

cout<<" is egual to";

c3.showData();

}

else{

c4.showData();

cout<<" is not equal to";

c3.showData();

}

return 0;

}

2. Write a C++ program to overload unary operators that is increment and decrement.

#include <iostream>

using namespace std;

class Operator

{

private:

int a,b;

public:

Operator operator++()

{

Operator temp;

temp.a=++a;

temp.b=++b;

return temp;

}

Operator operator--()

{

Operator temp;

temp.a=--a;

temp.b=--b;

return temp;

}

void setData(int x, int y)

{

a=x;

b=y;

}

void showData()

{

cout<<"Real = "<<a<<" Img = "<<b<<endl;

}

};

int main()

{

Operator c1,c2,c3;

c1.setData(5,8);

cout<<"\t\tWithout any changes......"<<endl;

c1.showData();

cout<<"\t\tAfter Increment......"<<endl;

c2=++c1;

c2.showData();

cout<<"\t\tAfter Dicrement......"<<endl;

c3=--c1;

c3.showData();

return 0;

}

3. Write a C++ program to add two complex numbers using operator overloaded by a friend function

#include <iostream>

using namespace std;

class Complex

{

private:

int a,b;

public:

friend Complex operator+(Complex, Complex );

void setData(int x, int y)

{

a=x;

b=y;

}

void showData()

{

cout<<"Real = "<<a<<" Img = "<<b<<endl;

}

};

Complex operator+(Complex A, Complex B)

{

Complex temp;

temp.a=A.a+B.a;

temp.b=A.b+B.b;

return temp;

}

int main()

{

Complex c1,c2,c3;

c1.setData(5,8);

c2.setData(4,2);

c3=c1+c2;//=operator+(c1,c2);

c1.showData();

c2.showData();

c3.showData();

return 0;

}

4. Create a class Time which contains:

- Hours

- Minutes

- Seconds

Write a C++ program using operator overloading for the following:

1. = = : To check whether two Times are the same or not.

2. >> : To accept the time.

3. << : To display the time.

**Output -**

#include <iostream> 

using namespace std;

class Time

{

private:

int H,M,S;

public:

int operator==(Time t)

{

if(H==t.H&&M==t.M&&S==t.S)

return 1;

else

return 0;

}

int operator>>(Time t)

{

if(H>=t.H)

{

if(M>=t.M)

{

if(S>=t.S)

return 1;

}

return 1;

}

else

return 0;

}

friend void operator<<(ostream &output, Time &t);

friend void operator>>(istream &input , Time &T);

};

void operator<<(ostream &output, Time &t)

{

output<<"OPERATOR<< called\n\n";

output<<"Hour : "<<t.H<<endl<<"Minute : "<<t.M<<endl<<"Second : "<<t.S<<endl;

}

void operator>>(istream &input , Time &T)

{

cout<<"OPERATOR>> called\n\n";

cout<<"Enter Hour : ";

input>>T.H;

cout<<"Enter Minute : ";

input>>T.M;

cout<<"Enter Second : ";

input>>T.S;

}

int main()

{

Time t1,t2;

int t,x;

cout<<"Enter first Time"<<endl;

cout<<"---------------------"<<endl<<endl;

cin>>t1;

cout<<"First Time"<<endl;

cout<<t1;

cout<<"Enter second Time"<<endl;

cout<<"---------------------"<<endl;

cin>>t2;

cout<<"Second Time"<<endl;

cout<<t2;

t=t1==t2;

cout<<"---------------------"<<endl<<endl;

if(t)

cout<<" Time is equal "<<endl;

else

{

x=t1>>t2;

if(x)

{

cout<<"...Greater time is.................";

cout<<t1;

}

else{

cout<<"...Greater time is................\n";

cout<<t2;

}}

return 0;

}

5. Consider following class Numbers

*class Numbers*

*{*

*int x,y,z;*

*public:*

*// methods*

*};*

Overload the operator unary minus (-) to negate the numbers.

#include <iostream>

using namespace std;

class Number

{

private:

int x,y,z;

public:

void operator-()

{

x = -x;

y = -y;

z = -z;

}

void setData(int a, int b, int c)

{

x=a;

y=b;

z=c;

}

void showData()

{

cout<<x<<" "<<y<<" "<<z<<endl;

}

};

int main()

{

Number N1,N2;

N1.setData(5,10,15);

cout<<"\nBefore negation\n";

N1.showData();

-N1;

cout<<"\nAfter negation\n";

N1.showData();

return 0;

}

6. Create a class CString to represent a string.

a) Overload the + operator to concatenate two strings.

b) == to compare 2 strings.

#include <iostream>

#include<cstring>

using namespace std;

class Cstring

{

private :

char str[20];

public :

void inputData()

{

cin>>str;

cout<<endl;

}

void showData()

{

cout<<str<<endl;

}

Cstring operator+(Cstring s)

{

Cstring s1;

strcpy(s1.str , str);

strcat(s1.str , s.str);

return s1;

}

int operator==(Cstring s)

{

int i;

i=strcmp(str,s.str);

if(i!=0)

return 1;

else

return 0;

}

};

int main()

{

Cstring s1,s2,s3;

int r;

cout<<"Enter first string"<<endl;

s1.inputData();

cout<<"Enter second string"<<endl;

s2.inputData();

cout<<endl;

s1.showData();

s2.showData();

cout<<"-------After + 2 string-------"<<endl;

s3=s1+s2;

s3.showData();

r=s1==s2;

if(r==1)

cout<<"Not equal";

else

cout<<"Equal";

return 0;

}

7. Define a C++ class fraction

*class fraction*

*{*

*long numerator;*

*long denominator;*

*Public:*

*fraction (long n=0, long d=0);*

*}*

Overload the following operators as member or friend:

a) Unary ++ (pre and post both)

b) Overload as friend functions: operators << and >>.

**Output**

****#include <iostream>

using namespace std;

class Fraction

{

private:

int Numerator, Denominator;

public:

Fraction()

{

Numerator=0;

Denominator=0;

}

Fraction operator++()

{

Fraction t;

t.Numerator=++Numerator;

t.Denominator=++Denominator;

return t;

}

Fraction operator++(int dummy)

{

Fraction t;

t.Numerator=Numerator++;

t.Denominator=Denominator++;

return t;

}

friend void operator<<(ostream &output, Fraction &t);

friend void operator>>(istream &input , Fraction &T);

};

void operator<<(ostream &output, Fraction &t)

{

output<<t.Numerator<<"/"<<t.Denominator<<endl;

}

void operator>>(istream &input , Fraction &T)

{

cout<<"Numerator = ";

input>>T.Numerator;

cout<<"Denominator = ";

input>>T.Denominator;

}

int main()

{

Fraction f1,f2,f3,f4;

cout<<"Enter first fraction value\n";

f1.inputData();

f3=++f1; // you can write only ++f1 and also write (++f1).display();

cout<<"After post\_increment\n";

// cout<<"++f1 : "<<++f1<<endl;

f3.display(); // you can write here f1.showdata when u write ++f1;

f4=f1++; // you can write only f1++ and also write (f1++).display();

cout<<"After post\_increment\n";

// cout<<"++f1 : "<<f1++<<endl;

f4.display();// you can write here f1.display() when u write f1++;

return 0;**UPDATED FROM HERE**

Fraction f1,f2;

cout<<"\n f1 : ";

cout<<f1;

cout<<"\n f2 : ";

cout<<f2;

cout<<"\nEnter first fraction value\n";

cin>>f1;

cout<<"\n f1++ : ";

f1++;

cout<<f1;

cout<<"\n ++f1 : ";

++f1;

cout<<f1;

cout<<"\n\nEnter second fraction\n";

cin>>f2;

f2=++f1;

cout<<"\n f2 = ++f1";

cout<<"\n f1 : ";

cout<<f1;

cout<<"\n f2 : ";

cout<<f2;

f2=f1++;

cout<<"\n f2 = f1++";

cout<<"\n f1 : ";

cout<<f1;

cout<<"\n f2 : ";

cout<<f2;

return 0;

}

}

8. Consider a class Matrix

*Class Matrix*

*{*

*int a[3][3];*

*Public:*

*//methods;*

*};*

Overload the - (Unary) should negate the numbers stored in the object.

Output –



**#include <iostream>**

**using namespace std;**

**class matrix**

**{**

**private :**

**int arr[3][3];**

**public:**

**void inputData();**

**void showData();**

**void operator-();**

**};**

**void matrix::inputData()**

**{**

**int i,j;**

**for(i=0;i<3;i++)**

**for(j=0;j<3;j++)**

**cin>>arr[i][j];**

**}**

**void matrix::showData()**

**{**

**int i,j;**

**for(i=0;i<3;i++)**

**{**

**cout<<endl;**

**for(j=0;j<3;j++)**

**cout<<arr[i][j]<<"\t";**

**}**

**}**

**void matrix::operator-()**

**{**

**int i,j;**

**for(i=0;i<3;i++)**

**{**

**for(j=0;j<3;j++)**

**{**

**arr[i][j]=-arr[i][j];**

**}**

**}**

**}**

**int main()**

**{**

**matrix a;**

**cout<<"\t\tEnter 3 X 3 Matrix\n\n";**

**a.inputData();**

**a.showData();**

**-a;**

**cout<<endl;**

**cout<<" After Negation"<<endl;**

**a.showData();**

**return 0;**

**}**

9. Consider the following class mystring

*Class mystring*

*{*

*char str [100];*

*Public:*

*// methods*

*};*

Overload operator “!” to reverse the case of each alphabet in the string

(Uppercase to Lowercase and vice versa)

#include <iostream>

using namespace std;

class mystring

{

private :

char str[100];

public:

void input()

{

cin>>str;

}

void display()

{

cout<<str;

}

void operator!();

};

void mystring::operator!()

{

int i,j;

for(i=0;str[i];i++)

{

if(str[i]>=65&&str[i]<=90)

{

str[i]=str[i]+32;

}

else if(str[i]>=97&&str[i]<=122)

{

str[i]=str[i]-32;

}

}

}

int main()

{

mystring s;

cout<<"Enter a string\n";

s.input();

!s;

cout<<"Overload operator “!” to reverse the case of each alphabet in the string(Uppercase to Lowercase and vice versa).\n";

s.display();

return 0;

}

*10.Class Matrix*

*{*

*int a[3][3];*

*Public:*

*//methods;*

*};*

Let m1 and m2 are two matrices. Find out m3=m1+m2 (use operator

overloading).

Output -

#include <iostream>

using namespace std;

class matrix

{

private:

int arr[3][3];

public:

void operator+(matrix );

void inputData();

void showData();

};

void matrix::inputData()

{

for(int i =0 ; i<3 ; i++)

{

for(int j=0 ; j<3 ; j++)

{

cin>>arr[i][j];

}

}

}

void matrix::showData()

{

for(int i =0 ; i<3 ; i++)

{

cout<<endl;

for(int j=0 ; j<3 ; j++)

{

cout<<arr[i][j];

}

}

}

void matrix::operator+( matrix c)

{

int mat[3][3];

for(int i=0;i<3;i++)

{

for(int j=0;j<3;j++)

mat[i][j]=arr[i][j]+c.arr[i][j];

}

cout<<"\nAddition of two matrix\n";

for(int i=0;i<3;i++)

{

cout<<endl;

for(int j=0;j<3;j++){

mat[i][j]=arr[i][j]+c.arr[i][j];

cout<<mat[i][j]<<" ";

}

}}

int main()

{

matrix c1,c2,c3;

cout<<"Enter first matrix\n";

c1.inputData();

cout<<"Enter second matrix\n";

c2.inputData();

c1+c2;

return 0;

}