Assignment - 27 A Job Ready Bootcamp in C++, DSA and IOT MySirG

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. ==

Sol – 1.

#include<iostream>

using namespace std;

class Complex

{

private :

int r=0,i=0;

public :

Complex(){}

Complex(int real,int imag)

{

r=real;

i=imag;

}

Complex operator+(Complex C)

{

Complex temp;

temp.r=r+C.r;

temp.i=i+C.i;

return temp;

}

Complex operator-(Complex C)

{

Complex temp;

temp.r=r-C.r;

temp.i=i-C.i;

return temp;

}

Complex operator\*(Complex C)

{

Complex temp;

temp.r=r\*C.r;

temp.i=i\*C.i;

return temp;

}

int operator==(Complex C)

{

if(r==C.r&&i==C.i)

return 1;

return 0;

}

int getreal()

{

return r;

}

int getimg()

{

return i;

}

};

int main()

{

Complex C1(9,0),C2(0,9),C3;

int i;

C3=C1+C2;

i=C2==C1;

cout<<"C3\nreal : "<<C3.getreal()<<" imag : "<<C3.getimg()<<"\nC1 and C2 are equal : "<<i;

return 0;

}

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

Sol – 2.

#include<iostream>

using namespace std;

class Complex

{

private :

int r=0,i=0;

public :

Complex(){}

Complex(int real,int imag)

{

r=real;

i=imag;

}

Complex operator++()

{

Complex temp;

temp.r=++r;

temp.i=++i;

return temp;

}

Complex operator--()

{

Complex temp;

temp.r=--r;

temp.i=--i;

return temp;

}

Complex operator++(int needless)

{

Complex temp;

temp.r=r++;

temp.i=i++;

return temp;

}

Complex operator--(int needless)

{

Complex temp;

temp.r=r--;

temp.i=i--;

return temp;

}

void showdata()

{

cout<<"Real : "<<r<<" Imag : "<<i<<endl;

}

};

int main()

{

Complex C1(5,4);

(++C1).showdata();

(--C1).showdata();

C1++.showdata();

C1.showdata();

C1--.showdata();

C1.showdata();

return 0;

}

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

friend function.

Sol – 3.

#include<iostream>

using namespace std;

class Complex

{

private :

int r=0,i=0;

public :

Complex(){}

Complex(int real,int imag)

{

r=real;

i=imag;

}

void showdata()

{

cout<<"Real : "<<r<<" Imag : "<<i;

}

friend Complex operator+(Complex,Complex);

};

Complex operator+(Complex X,Complex Y)

{

Complex temp;

temp.r=X.r+Y.r;

temp.i=X.i+Y.i;

return temp;

}

int main()

{

Complex C1(5,4),C2(7,5),C3;

C3=C1+C2;

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.

Sol – 4.

#include<iostream>

using namespace std;

class Time

{

private :

int hr=0,min=0,sec=0;

public :

int operator==(Time X)

{

int tots1=hr\*3600+min\*60+sec;

int tots2=X.hr\*3600+X.min\*60+X.sec;

if(tots1==tots2)

return 1;

return 0;

}

friend ostream& operator<<(ostream &,Time);

friend istream& operator>>(istream &,Time &);

};

istream& operator>>(istream &obj,Time &T)

{

z: cout<<"Enter hour,minute and seconds : ";

obj>>T.hr>>T.min>>T.sec;

T.min=T.min+T.sec/60;

T.sec%=60;

T.hr=T.hr+T.min/60;

T.min%=60;

if(T.hr>24||T.hr<0||T.min<0||T.sec<0)

{

cout<<"Invalid Time!\nTry again\n";

goto z;

}

return obj;

}

ostream& operator<<(ostream &obj,Time T)

{

obj<<T.hr<<"hr "<<T.min<<" min "<<T.sec<<" sec"<<endl;

return obj;

}

int main()

{

Time T1,T2;

cin>>T1>>T2;

cout<<T1<<T2;

if(T1==T2)

cout<<"Times are Same";

else

cout<<"Times are not Same";

return 0;

}

5. Consider following class Numbers

class Numbers

{

int x,y,z;

public:

// methods

};

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

Sol – 5.

#include<iostream>

using namespace std;

class Number

{

private :

int x=0,y=0,z=0;

public :

void operator-()

{

x=-x;

y=-y;

z=-z;

}

void display()

{

cout<<"x : "<<x<<" y : "<<y<<" z : "<<z;

}

void acc()

{

cout<<"Enter x,y,z : ";

cin>>x>>y>>z;

}

};

int main()

{

Number N;

N.acc();

-N;

N.display();

}

6. Create a class CString to represent a string.

a) Overload the + operator to concatenate two strings.

b) == to compare 2 strings.

Sol – 6.

#include<iostream>

#include<cstring>

using namespace std;

class CString

{

private :

char str[20];

public :

void getstr()

{

cout<<"Enter a string\n";

cin>>str;

}

void display()

{

cout<<str<<endl;

}

CString operator+(CString &X)

{

CString temp;

strcpy(temp.str,str);

strcat(temp.str,X.str);

return temp;

}

int operator==(CString X)

{

if(strlen(str)==strlen(X.str))

{

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

{

if(str[i]==X.str[i])

continue;

else

return 1;

}

return 0;

}

return 1;

}

};

int main()

{

CString a,b,c;

a.getstr();

b.getstr();

a.display();

b.display();

c=a+b;

c.display();

if(a==b)

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 >>.

Sol – 7.

#include<iostream>

using namespace std;

class Fraction

{

private :

long num=0,den=0;

public :

Fraction(){}

Fraction(long numerator,long denominator)

{

num=numerator;

den=denominator;

}

Fraction operator++()

{

Fraction temp;

temp.num=++num;

temp.den=++den;

return temp;

}

Fraction operator++(int useless)

{

Fraction temp;

temp.num=num++;

temp.den=den++;

return temp;

}

friend ostream & operator<<(ostream &,Fraction);

friend istream & operator>>(istream &,Fraction &);

};

ostream & operator<<(ostream &obj,Fraction F)

{

obj<<"Fraction is "<<F.num<<"/"<<F.den<<endl;

return obj;

}

istream & operator>>(istream &obj,Fraction &F)

{

cout<<"Enter numerator and denominator : ";

obj>>F.num>>F.den;

return obj;

}

int main()

{

Fraction F1,F2(5,10);

cin>>F1;

F2=(++F1);

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.

Sol – 8.

#include<iostream>

using namespace std;

class Matrix

{

private :

int a[3][3];

public :

Matrix(){}

Matrix(int n,int b,int c,int d,int e,int f,int g,int h,int i)

{

a[0][0]=n;

a[0][1]=b;

a[0][2]=c;

a[1][0]=d;

a[1][1]=e;

a[1][2]=f;

a[2][0]=g;

a[2][1]=h;

a[2][2]=i;

}

void accval()

{

cout<<"Enter elements of (3 X 3) matrix\n";

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

{

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

{

cin>>a[i][j];

}

}

}

void display()

{

cout<<endl;

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

{

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

{

cout<<a[i][j]<<"\t";

}

cout<<endl;

}

cout<<endl;

}

Matrix operator-()

{

Matrix M;

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

{

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

{

M.a[i][j]=-a[i][j];

}

}

return M;

}

};

int main()

{

Matrix M1(5,6,7,9,8,4,1,2,3),M2;

cout<<"Before";

M1.display();

cout<<"After";

(-M1).display();

M2.accval();

M1=(-M2);

M1.display();

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).

Sol – 9.

#include<iostream>

#include<cstring>

using namespace std;

class Mystring

{

private :

char str[100];

public :

Mystring(){}

Mystring(char \*p)

{

strcpy(str,p);

}

void accstr()

{

cout<<"Enter a string : ";

fgets(str,100,stdin);

}

void display()

{

cout<<"String is : "<<str<<endl;

}

void operator!()

{

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

{

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

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

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

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

}

}

};

int main()

{

Mystring M1("Bacho KO bachane VaLa balveer"),M2;

M1.display();

!M1;

M1.display();

M2.accstr();

!M2;

M2.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).

Sol – 10.

#include<iostream>

using namespace std;

class Matrix

{

private :

int a[3][3];

public :

Matrix(){}

Matrix(int n,int b,int c,int d,int e,int f,int g,int h,int i)

{

a[0][0]=n;

a[0][1]=b;

a[0][2]=c;

a[1][0]=d;

a[1][1]=e;

a[1][2]=f;

a[2][0]=g;

a[2][1]=h;

a[2][2]=i;

}

void accval()

{

cout<<"Enter elements of (3 X 3) matrix\n";

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

{

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

{

cin>>a[i][j];

}

}

}

void display()

{

cout<<endl;

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

{

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

{

cout<<a[i][j]<<"\t";

}

cout<<endl;

}

cout<<endl;

}

Matrix operator+(Matrix X)

{

Matrix M;

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

{

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

{

M.a[i][j]=a[i][j]+X.a[i][j];

}

}

return M;

}

};

int main()

{

Matrix M1(5,6,7,9,8,4,1,2,3),M2,M3;

M2.accval();

M3=M1+M2;

M3.display();

return 0;

}