**Assignment - 28 A Job Ready Bootcamp in C++, DSA and IOT MySirG**

**Operator Overloading, friend operator and this pointers**

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

Overload following operators

a. << insertion operator

b. >> extraction operator

Sol – 1.

#include<iostream>

using namespace std;

class Complex

{

private :

int r=0,img=0;

public :

Complex(){}

Complex(int real,int imag)

{

r=real;

img=imag;

}

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

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

};

istream& operator>>(istream &obj,Complex &C)

{

cout<<"Enter real and imag part : ";

obj>>C.r>>C.img;

return obj;

}

ostream& operator<<(ostream &obj,Complex C)

{

obj<<"Complex number is\nReal : "<<C.r<<" and Imag : "<<C.img<<" i"<<endl;

return obj;

}

int main()

{

Complex C1(4,5);

cout<<C1;

return 0;

}

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

One of the functions should be setData() to set the properties of the object. Make

sure the names of formal arguments are the same as names of instance variables.

Sol – 2.

#include<iostream>

using namespace std;

class Complex

{

private :

int r=0,img=0;

public :

void setdata(int r,int img)

{

this->r=r;

this->img=img;

}

void showdata()

{

cout<<"Complex number is\nReal : "<<r<<" and Imag : "<<img<<" i"<<endl;

}

};

int main()

{

Complex C1;

C1.setdata(5,2);

C1.showdata();

return 0;

}

3. Overload subscript operator [] that will be useful when we want to check for an index

out of bound.

Sol – 3.

#include<iostream>

#include<cstring>

using namespace std;

class Array

{

private :

int a[100];

const int size = 100;

public :

void setval(int n,int index)

{

if(n>=size)

{

cout<<"Array bounding error";

exit(0);

}

a[index]=n;

}

int operator[](int n)

{

if(n>=size)

{

cout<<"Array index out of bound exception";

exit(0);

}

else

return a[n];

}

};

int main()

{

Array a;

a.setval(5,110);

cout<<a[110];

return 0;

}

~~4.~~ Create a student class and overload new and delete operators as a member function

of the class.

~~5.~~ Create a student class and overload new and delete operators outside the class.

6. Create a complex class and overload assignment operator for that class.

Sol – 6.

#include<iostream>

using namespace std;

class Complex

{

private :

int r=0,img=0;

public :

Complex(){}

Complex(int real,int imag)

{

r=real;

img=imag;

}

Complex& operator=(Complex &X)

{

r=X.r;

img=X.img;

return \*this;

}

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

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

};

istream& operator>>(istream &obj,Complex &C)

{

cout<<"Enter real and imag part : ";

obj>>C.r>>C.img;

return obj;

}

ostream& operator<<(ostream &obj,Complex C)

{

obj<<"Complex number is\nReal : "<<C.r<<" and Imag : "<<C.img<<" i"<<endl;

return obj;

}

int main()

{

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

C3=C2=C1;

cout<<C3<<C2<<C1;

return 0;

}

7. Create an Integer class and overload logical not operator for that class.

Sol – 7.

#include<iostream>

using namespace std;

class Integer

{

private :

int a;

public :

Integer(int x)

{

a=x;

}

Integer(){}

void setval(int x)

{

a=x;

}

int operator!()

{

return (-a);

}

};

int main()

{

Integer x=10;

cout<<!x;

return 0;

}

8. Create a Coordinate class for 3 variables x,y and z and overload comma operator

such that when you write c3 = (c1 , c2 ) then c2 is assigned to c3. Where c1,c2,and

c3 are objects of 3D coordinate class.

Sol – 8.

#include<iostream>

using namespace std;

class Coord3d

{

private :

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

public :

Coord3d(int a,int b,int c)

{

x=a;

y=b;

z=c;

}

Coord3d(){}

void display()

{

cout<<"Value of x,y and z = "<<x<<" "<<y<<" and "<<z<<endl;

}

Coord3d operator,(Coord3d C)

{

Coord3d temp;

temp.x=C.x;

temp.y=C.y;

temp.z=C.z;

return temp;

}

};

int main()

{

Coord3d C1(1,3,5),C2(2,4,6),C3,C4(10,15,20);

C3=(C1,C2);

C3.display();

C3=(C2,C1,C4);

C3.display();

return 0;

}

9. Create an Integer class that contains int x as an instance variable and overload

casting int() operator that will type cast your Integer class object to int data type.

Sol – 9.

#include<iostream>

using namespace std;

class Integer

{

private :

int x;

public :

Integer()

{

}

Integer(int a)

{

x=a;

}

operator int()

{

return x;

}

void display()

{

cout<<"Value : "<<x<<endl;

}

};

int main()

{

Integer a;

int x=5,y;

a=x;

y=(int)a;

a.display();

cout<<y;

return 0;

}

10. Create a Distance class having 2 instance variable feet and inches. Also create

default constructor and parameterized constructor takes 2 variables . Now overload ()

function call operator that takes 3 arguments a , b and c and set feet = a + c + 5 and

inches = a+b + 15.

Sol – 10.

#include<iostream>

using namespace std;

class Distance

{

private :

int f,i;

public :

Distance()

{

}

Distance(int feet,int inches)

{

f=feet;

i=inches;

}

Distance operator()(int a,int b,int c)

{

Distance d;

d.f=a+c+5;

d.i=a+b+15;

return d;

}

void display()

{

cout<<"Feet : "<<f<<" Inches : "<<i<<endl;

}

};

int main()

{

Distance d1(5,10);

Distance d2;

d1.display();

d2 = d1(10,20,30);

d1.display();

d2.display();

return 0;

}

11. Create a class Marks that have one member variable marks and one member

function that will print marks. We know that we can access member functions using

(.) dot operator. Now you need to overload (->) arrow operator to access that

function.

Sol – 11.

#include<iostream>

using namespace std;

class Marks

{

private :

int m;

public :

Marks(int mark)

{

m=mark;

}

Marks\* operator->()

{

return this;

}

void printmarks()

{

cout<<"Marks : "<<m<<endl;

}

};

int main()

{

Marks m(5);

m->printmarks();

return 0;

}