operator overloading in c++

#include<iostream>

using namespace std;

class complex

{

private :

int a,b;

public:

void set\_data(int x, int y)

{a=x;b=y;}

void show\_data()

{

cout<<"a = "<<a<<" \nb = "<<b<<endl;

}

complex add(complex c2)

{

complex temp;

temp.a=a+c2.a;

temp.b=b+c2.b;

return temp;

}

complex operator +(complex c2)

{

complex temp;

temp.a=a+c2.a;

temp.b=b+c2.b;

return temp;

}

};

int main()

{

complex c1,c2,c3,c4;

c1.set\_data(3,4);

c2.set\_data(6,7);

c3=c1.add(c2);

c4=c1.operator +(c2);

c4=c1+c2;

//ab kya hum es me add funtion hata kr sum likh sakate hai ha likha skaate hai magar jaha add function declare kiya hai waha bhi change karana padega

//or ab kya es ka badal ke + rkh sakate hai nhi rkh skaate kyu ki ek rule hai function me nam pe hum alphabet , digit and underscore ke alawa kuch or nhi laga sakate toh + allowed nhi hai

//but c++ yeh kahata hai ki agar kisi operator sysmbol ko as na function name use karana chahate hai toh kr sakate hai but shart etani hai ki hum uske pahale operator keyword likhe

//but yad rahe hum un hi operator ko as an function name use kr sakate hai jo c language me ek valid opeartor tha

//lekin jab hum function ka nam aisa koe operator symbol rakhate hai toh ek or khas bat samane ati hai aise kisi function ko hum bina . lagaye bhi call kr sakate hai

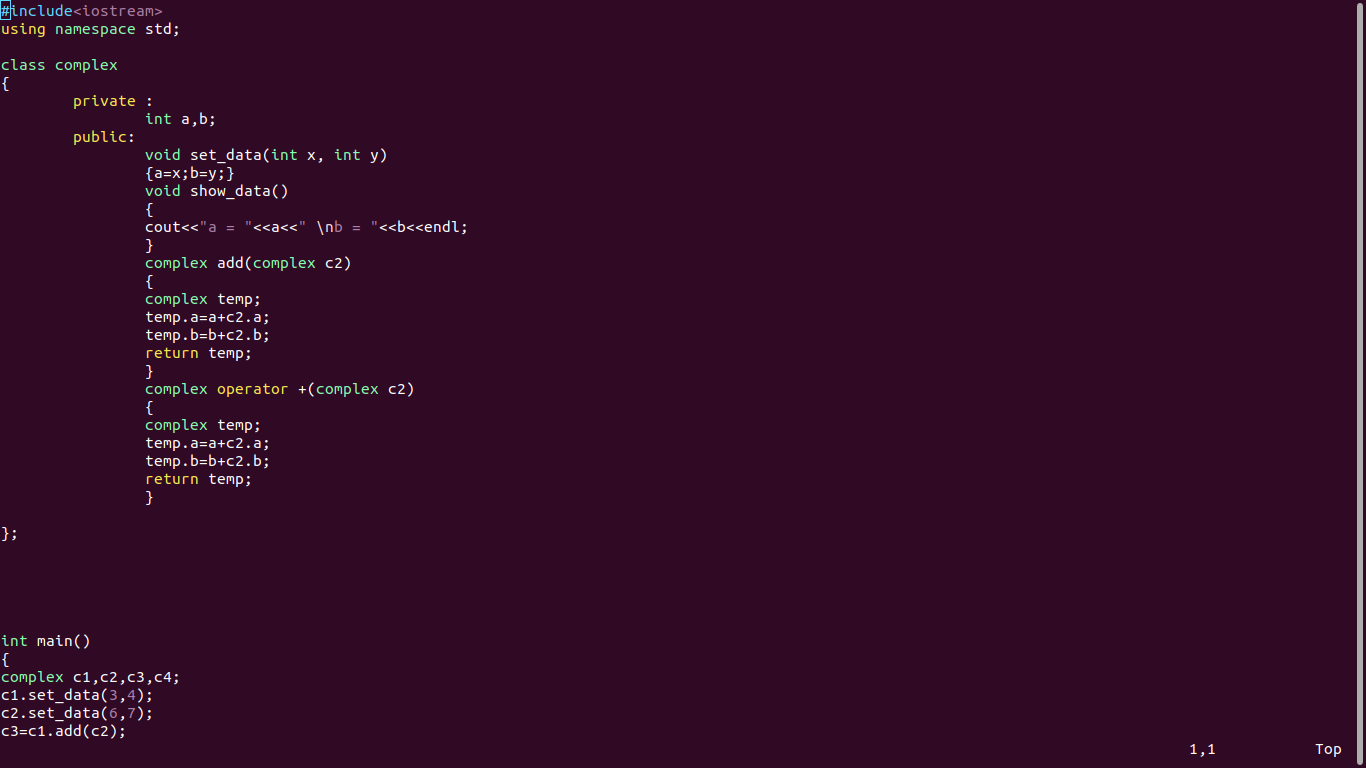
//yani c1 ke bad . and operator bhi na likhe , or paranthesis bhi na lagaye tabhi call hoga

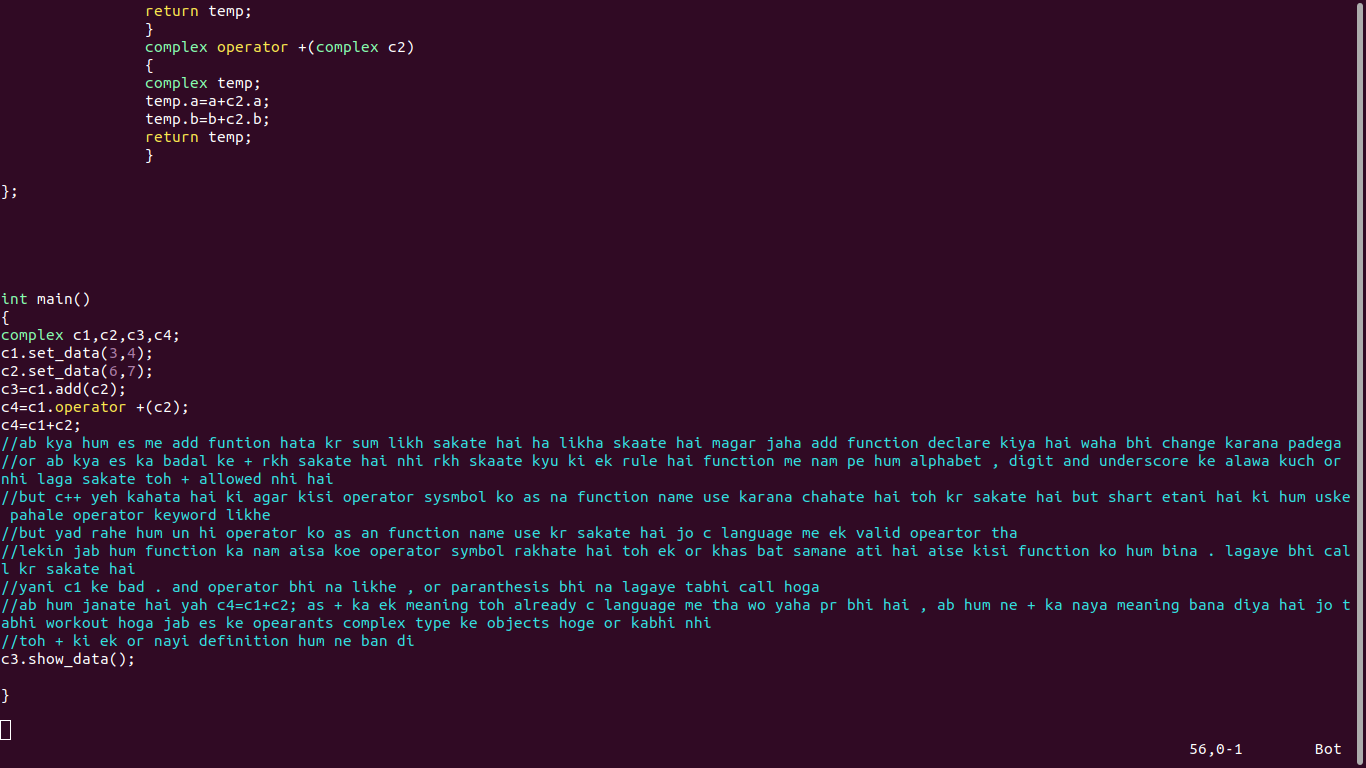
//ab hum janate hai yah c4=c1+c2; as + ka ek meaning toh already c language me tha wo yaha pr bhi hai , ab hum ne + ka naya meaning bana diya hai jo tabhi workout hoga jab es ke opearants complex type ke objects hoge or kabhi nhi

//toh + ki ek or nayi definition hum ne ban di

c3.show\_data();

}



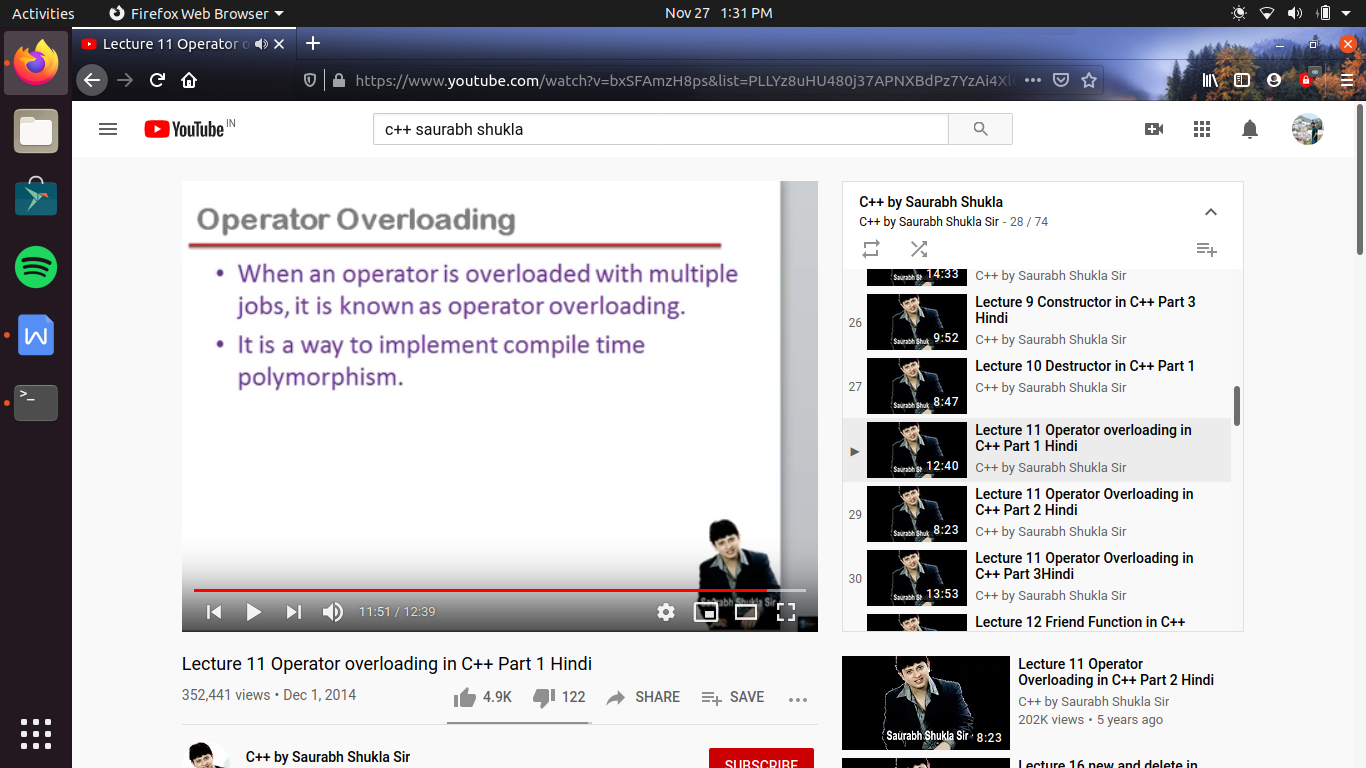


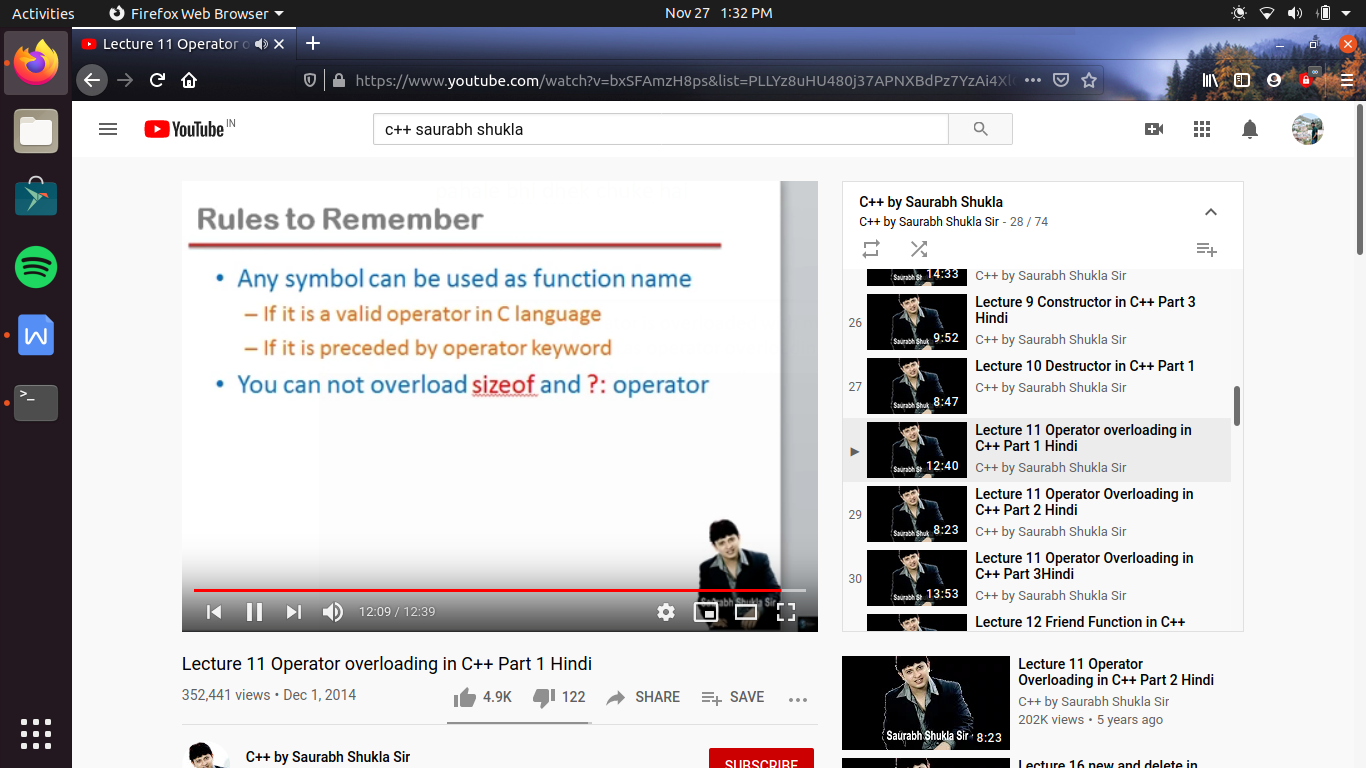
operator overloading -

jab ek operator symbol , ek se jayada kam ke leye bana ho toh hum ese operator overloading kahate hai , jaise hum ne + ki ek or definition bandi, ek jo thi wo primitive type ke add karane ke leye thi or ab dusari jo complex type ko add karane ke leye hai , or jo hamane banae hai

toh es tarah se operator overloaded ho gaya hai multiple job se

yeh polymorphism ko implement karane ka tarika hai , polymorphism ko implement karane ka ek tarika hum pahale bhi dhek chuke hai





but sizeof and conditional opearator ko hum operator overloading ki tarah use nhi kr sakate

+ operator ke binary operator hai , toh hum dhek chuke hai ki binary operator ko kaise over load karate hai

ab hum deekhege how to overload uninary operator :-

#include<iostream>

using namespace std;

class complex

{

private :

int a,b;

public:

void set\_data(int x, int y)

{a=x;b=y;}

void show\_data()

{

cout<<"a = "<<a<<" \nb = "<<b<<endl;

}

complex add(complex c2)

{

complex temp;

temp.a=a+c2.a;

temp.b=b+c2.b;

return temp;

}

complex operator +(complex c2)

{

complex temp;

temp.a=a+c2.a;

temp.b=b+c2.b;

return temp;

}

complex operator -()

{

complex temp;

temp.a=-a;

temp.b=-b;

return temp;

}

};

int main()

{

complex c1,c2,c3;

c1.set\_data(3,4);

c2=c1.operator -(); //c1 ne - operator ko call kiya or us ne or us me koe argument pass nhi kiya and - operator ne jo return kiya wo c2 me assign h

c3=-c1; //es ka meaning bhi same hai upar ki tarah , es pe koe argument pass nhi ho raha es ka mtlb hai yeh ek unary operator hai but agar binary opearator hota toh us me argument pass karate

//or agar yeh binary operator hota , toh us ko 2 operants ki zarurat hoti hai toh waha pe ek operant hota hai caller object or dusara argument

c3. show\_data();

}

//unary opeator ko assign karane ka jayada acha tarika yeh hai c3=-c1;

//but es me log kahate hai ki - operator pahale a raha hai

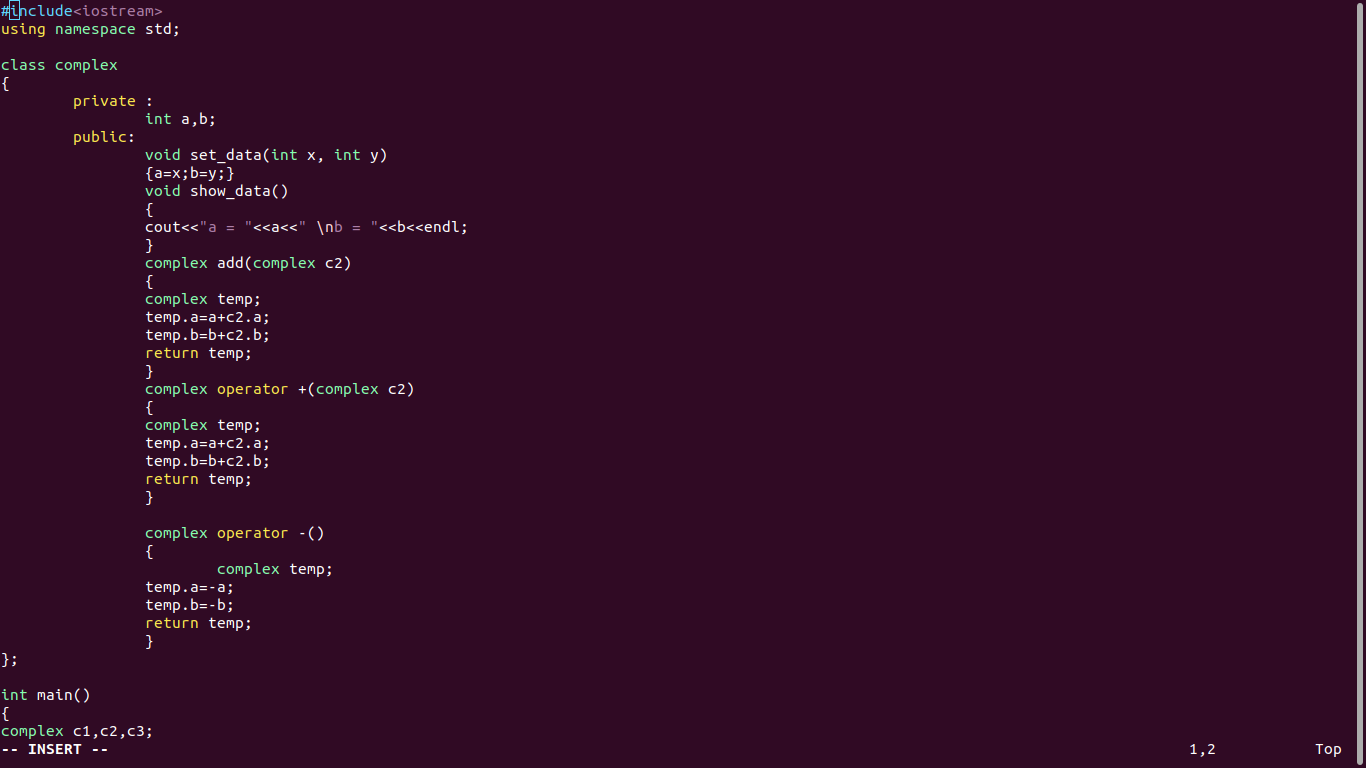
//es ke leye ek rule samajaha padega ki jab hum binary operator ko overload karate hai jaise

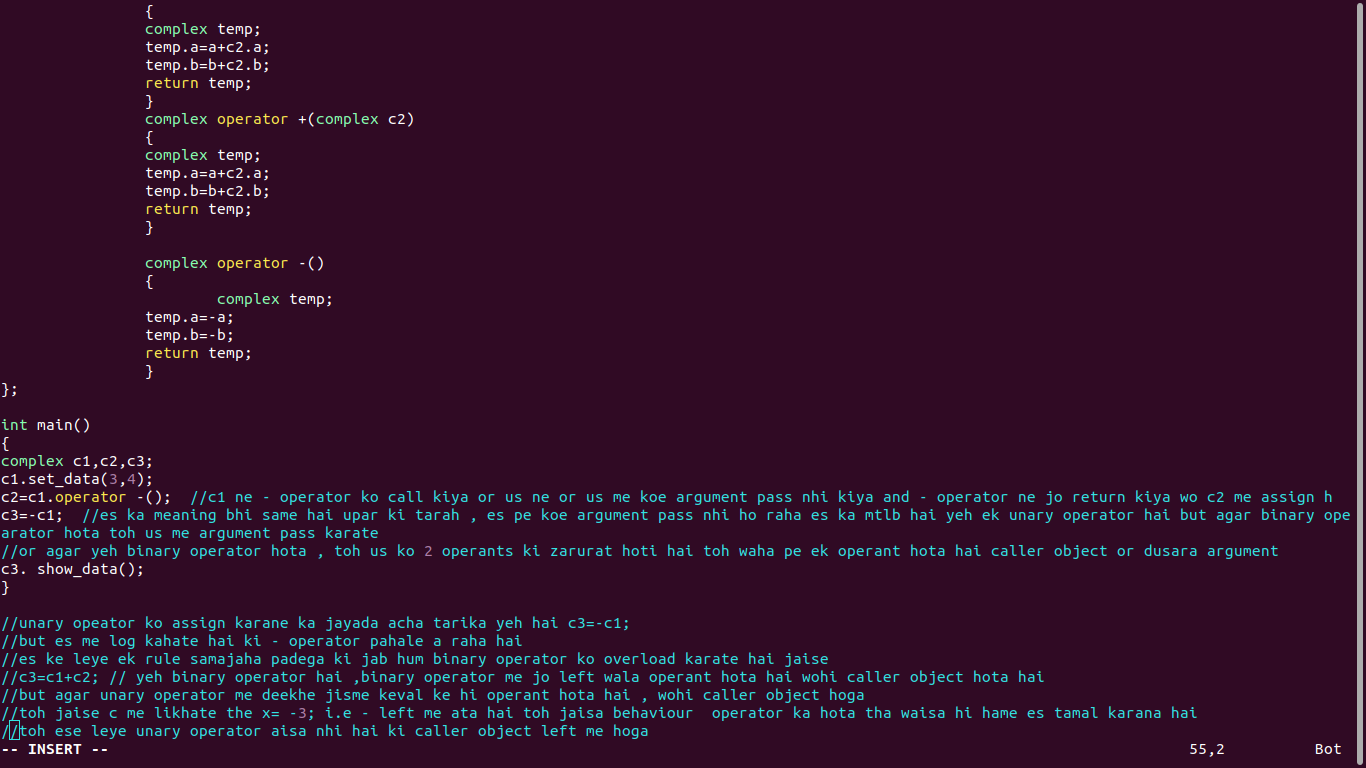
//c3=c1+c2; // yeh binary operator hai ,binary operator me jo left wala operant hota hai wohi caller object hota hai

//but agar unary operator me deekhe jisme keval ke hi operant hota hai , wohi caller object hoga

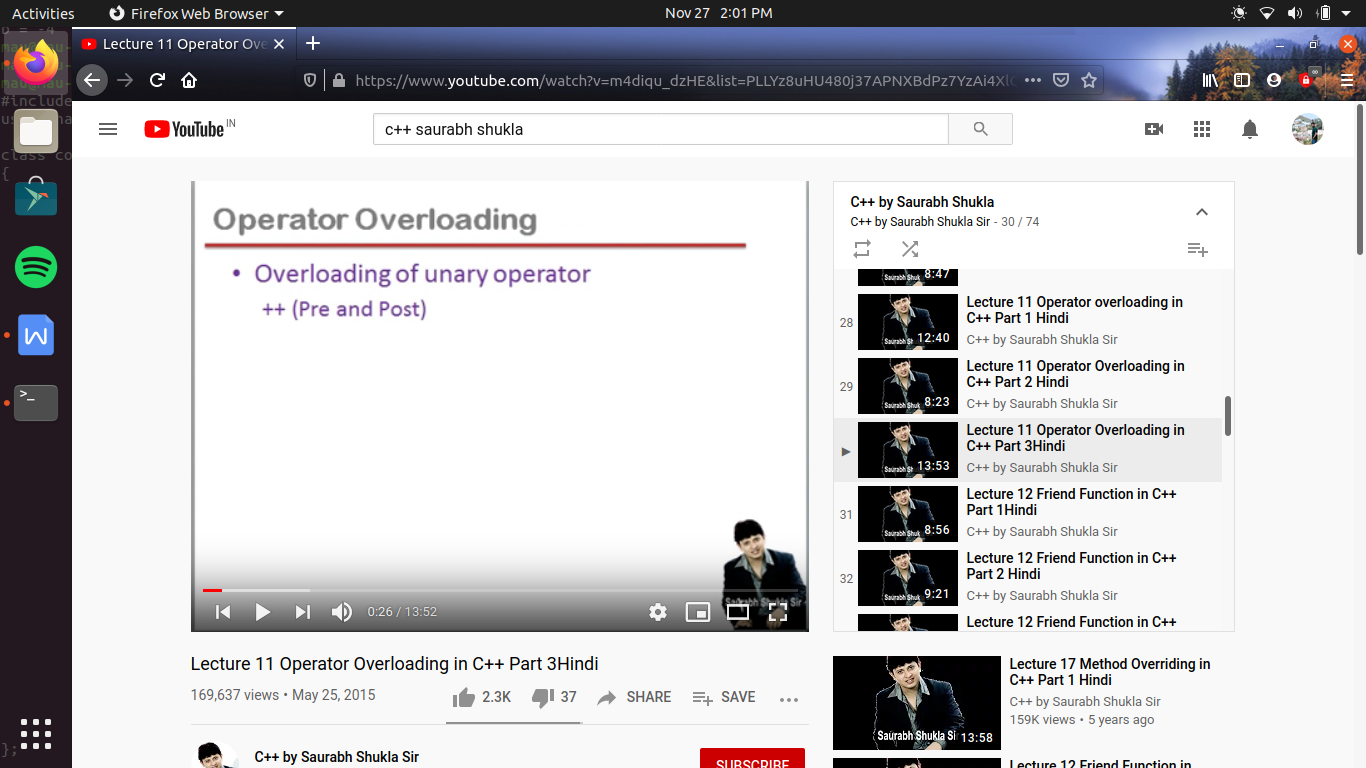
//toh jaise c me likhate the x= -3; i.e - left me ata hai toh jaisa behaviour operator ka hota tha waisa hi hame es tamal karana hai

//toh ese leye unary operator aisa nhi hai ki caller object left me hoga





overloading of increment operator :-



#include<iostream>

using namespace std;

class integer

{

private :

int a;

public:

void set\_data (int x)

{a=x;}

void show\_data()

{cout<<"a = "<<a<<endl;}

integer operator ++() //this is the coding for preincrement

{

integer c1;

c1.a=++a;

return c1;

}

integer operator ++(int) //post-increment

{

integer c2;

c2.a=a++;

return c2;

}

};

int main()

{

integer b1,b2;

b1.set\_data(4);

b1.show\_data();

b2=++b1; //or aagr pre increment hota hai toh pahale increment hota hai fir bad me wo value assign ho jati hai

//here means b2=b1.operator++();

b2=b1++; //post increament means pahale b1 ki value b2 me assign ho jaye fir us me increament ho

//here it means (b2=b1)++; //yani hame es ek leye ek alag function banana padega

//but ek or ++ operator wala function ,or dono me koe argument pass nhi ho rahi toh compiler ko yeh pata nhi chl payega ki kon sa kis time perform karana hai

//ese leye c++ me naya rule aya hai ki agar hum pre increment and post increment alag alag banana chahate hai toh , toh compiler differentiate kr payega

//jab hum post increament wale function me ek int argument pass karege , yeh int argument mention keval karana hai es ka koe use nhi hai

//kyu ki actual me hame koe value pass karani nhi hai , lekin hum yeh int argument pass es leye kr rahe hai ki compiler dono version of increment operator function me difference kr paye

//toh yeh pre-increment and post increment me fark karane ke leye hai

//or aisa hi decrement operator (--)ke sath bhi karana padega

b1.show\_data(); //as here b1 is incremented in pre-increment , and the same value assign to b2 , so both are 5

//here as in post increment value of b1 is not increased, first it assign the value of b1 into , the the value to b2 , so value of b2 remain 5

//then the value increased in b1; //as increment operator in attached to b1; thus value of b1 becames 6

b2.show\_data();

}

