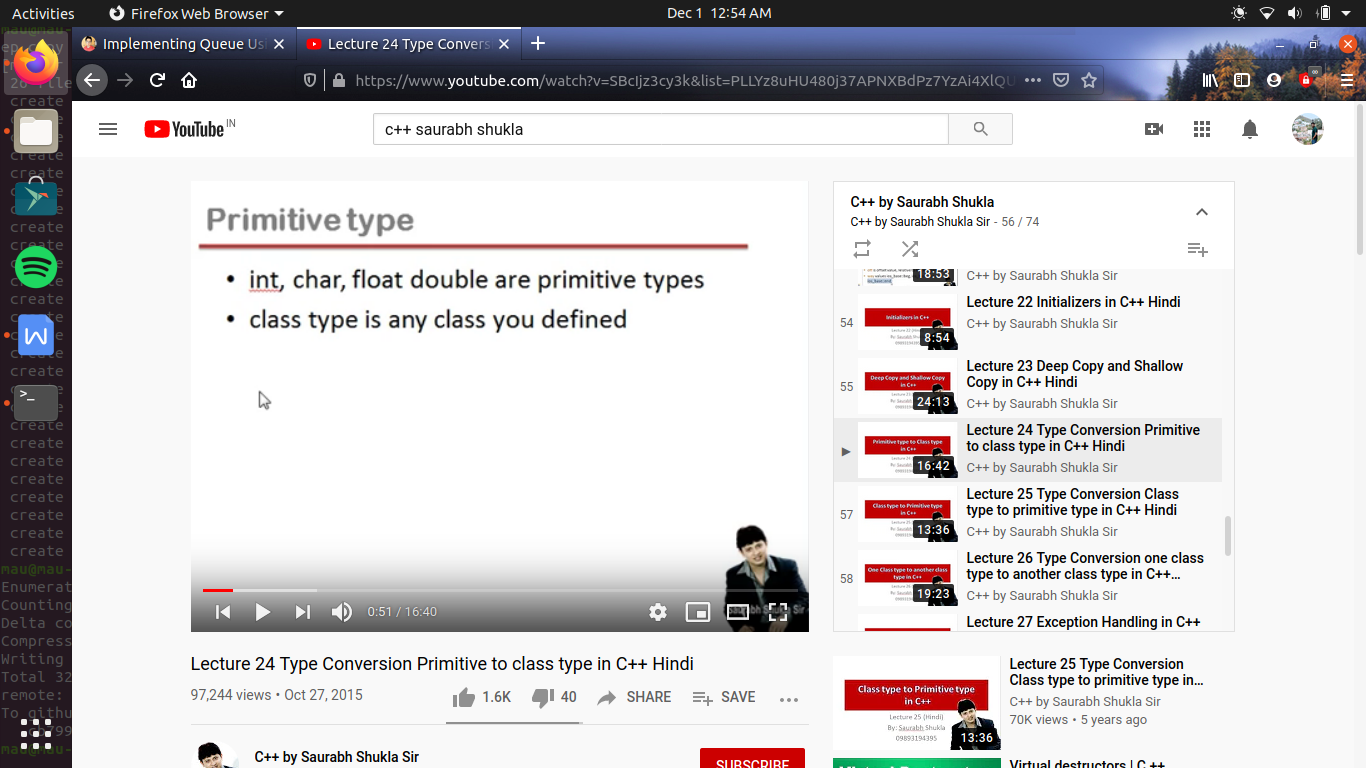
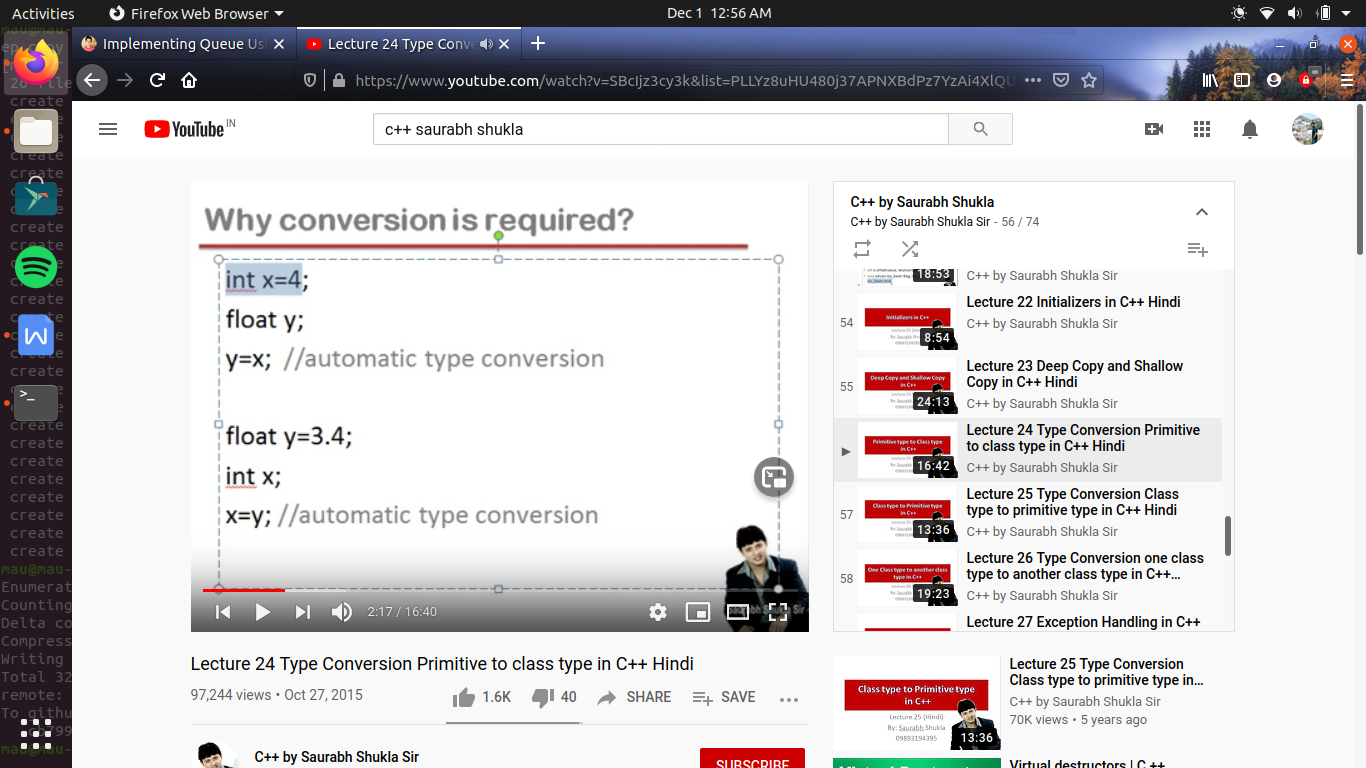
conversion b/w primitive type and class type in c++

kisi bhi language me keyword ke roop me jo data type hame milate hai use bolate hai primitive data type

jab hum class banate hai toh hum naya data type define karate hai or use bolate hai hum non primitive data type , ya simply bole class type



now why we need conversion?



as x here int type and y here float type

toh

agar hum y me x ki value assign karege

y=x;

toh pahale toh yeh hona nhi chahiye kyu ki dono data alag alag category ke hai , lekin dono hi primitive type hai , toh ese leye x or y ke beech yeh assignment work karega

work ese leye karata hai kyu ki data type ka internally conversion ho jata hai , yani x jo int type ka hai , wo automatically convert ho jayega y ke type me or y me assign ho jayega toh y me jata hai 4.0

ab ek or example deekhate hai

ab ki bar hum x me y ki value rkh rakhe hai toh yaha pr bhi koe error nhi ayegi or automatically type conversion ho jayega , toh y ki value pahale convert hogi int type me or use ke bad x me assign hoga

toh obvious si bat hai ki es me data loss hoga, kyu ki conversion me decimal or us ke bad wali chiz ignore hogayegi or us ke bad wali value x me assign ho jayegi i.e 3

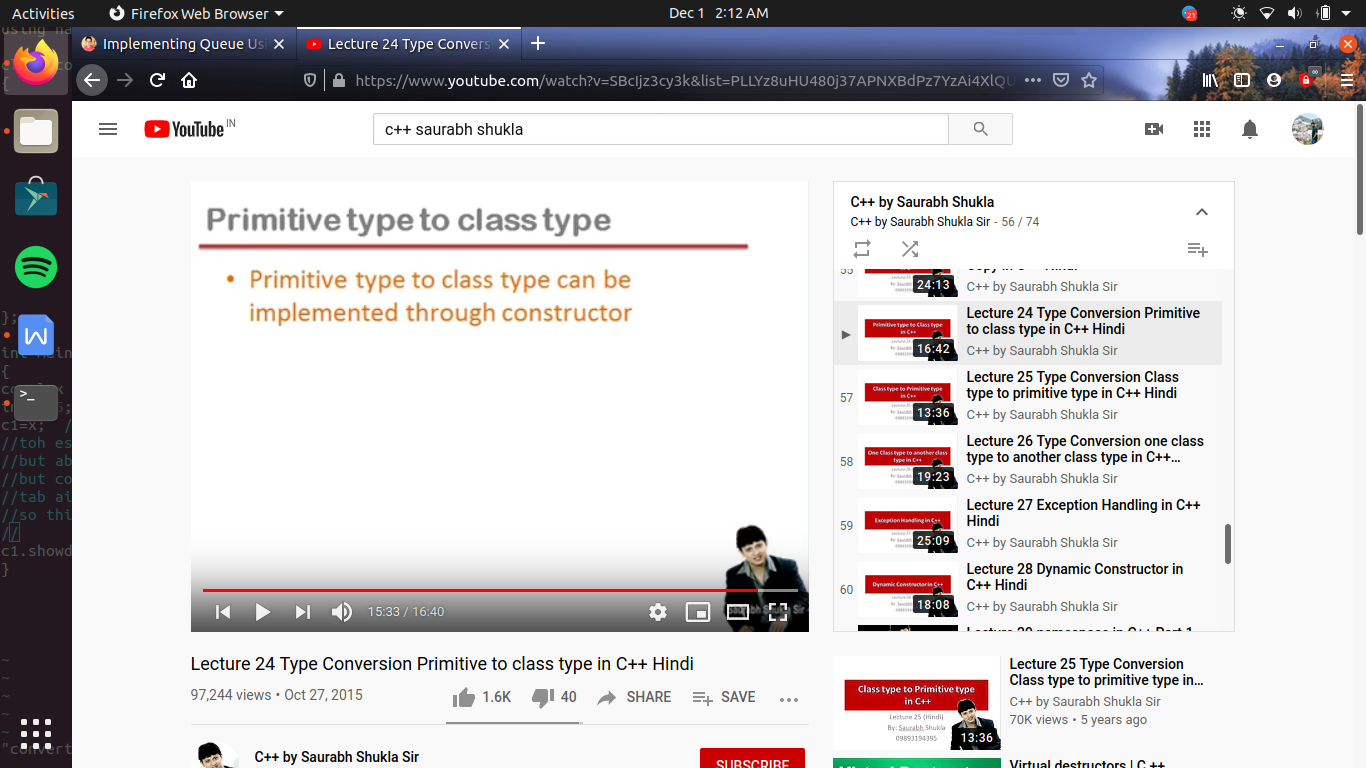
toh agar hum ek primitive type se dusare primitive type me conversion karawate hai toh automatically conversion ho jata hai , hame us me kuch bhi karane ki jarurat nhi padati

lekin agar ek ho primitive type or dusara ho class type toh ayegi error

as yaha hum complex type ki variable me int type ki value assign kr rahe hai , or yeh tabhi ho payega jab hum int type ko convert kar de complex class me

toh ese leye pdata padat hai hame type conversion

primitive type to class type



#include<iostream>

using namespace std;

class complex

{

private:

int a,b;

public :

complex() {} //default constructor

complex(int x) //this constructor call hoga jab object me dusare type ki value assign hogi

{a=x; b=0;}

void setdata(int x, int y)

{a=x;b=y;}

void showdata()

{cout<<"\n a="<<a<<"\n b="<<b<<endl; }

};

int main()

{

complex c1;

int x=5;

c1=x; //es me error ayegi kyu ki x ek int type ki value hai or c1 , complex type ka hai toh yeh convert nhi ho pa raha

//toh es ke leye hum ek constructor banayege,

//but ab tak hum janate the ki constructor tabhi call hota hai jab object banata hai

//but constructor tabhi call hote hai jab bhi object me dusare type ki value assign hoti hai

//tab aise me object ke leye wo constructor call ho jayega jo int type ki value leta hai

//so this line means now // c1.complex(x);

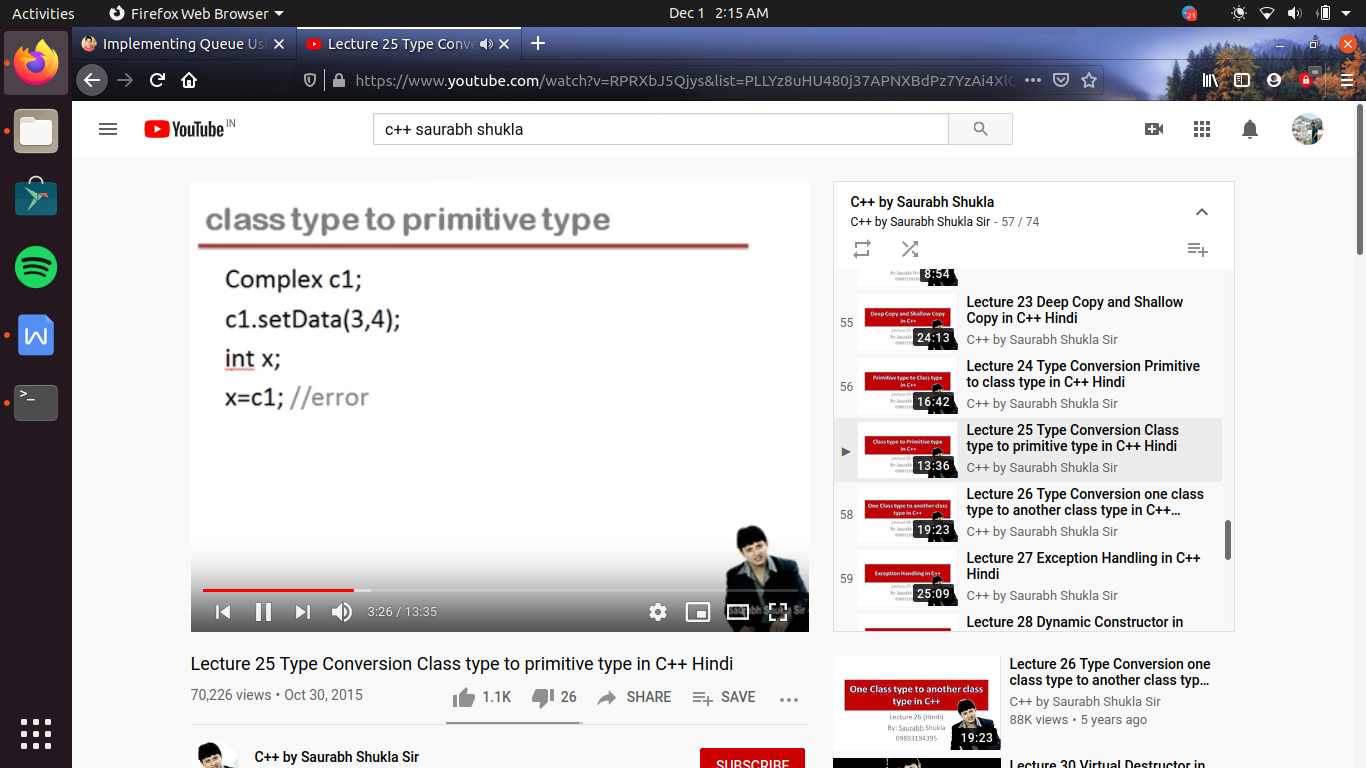
//

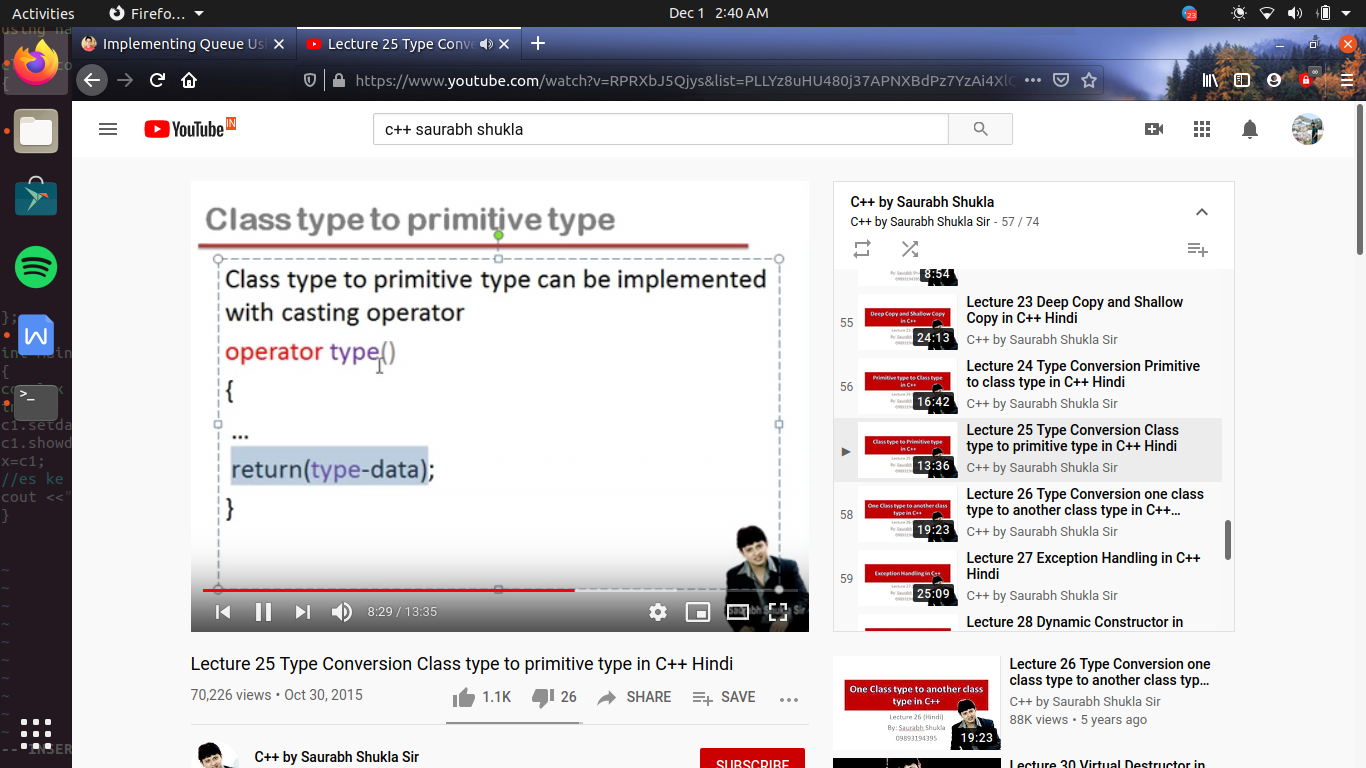
c1.showdata();

}



class type to primitive type ->





#include<iostream>

using namespace std;

class complex

{

private:

int a,b;

public :

complex() {} //default constructor

void setdata(int x, int y)

{a=x;b=y;}

void showdata()

{cout<<"\n a="<<a<<"\n b="<<b<<endl; }

operator int() //yaha int () means yeh function int type ki value return karega

{

return(a); //as c1 object ne call kiya hai casing operator ko toh a ki value yaha kis ki hogi , c1 object ki

}

//i.e operator keyword ke bad likhate hai hum wo data type jo casting operator karega return

};

int main()

{

complex c1;

int x;

c1.setdata(3,5);

c1.showdata();

x=c1; //toh agar hame non-primitive type se primitive type me convert karana padata hai toh hum use karate hai casting operator

//es ke leye hum operator keyword ka use karate hai

//toh ese hum es tarah se samjh sakate hai x=c1.operator int();

//yaha pr c1 ne casing operator ko call kiya, or ab casting operator jo return karega wo assign ho jayega x me

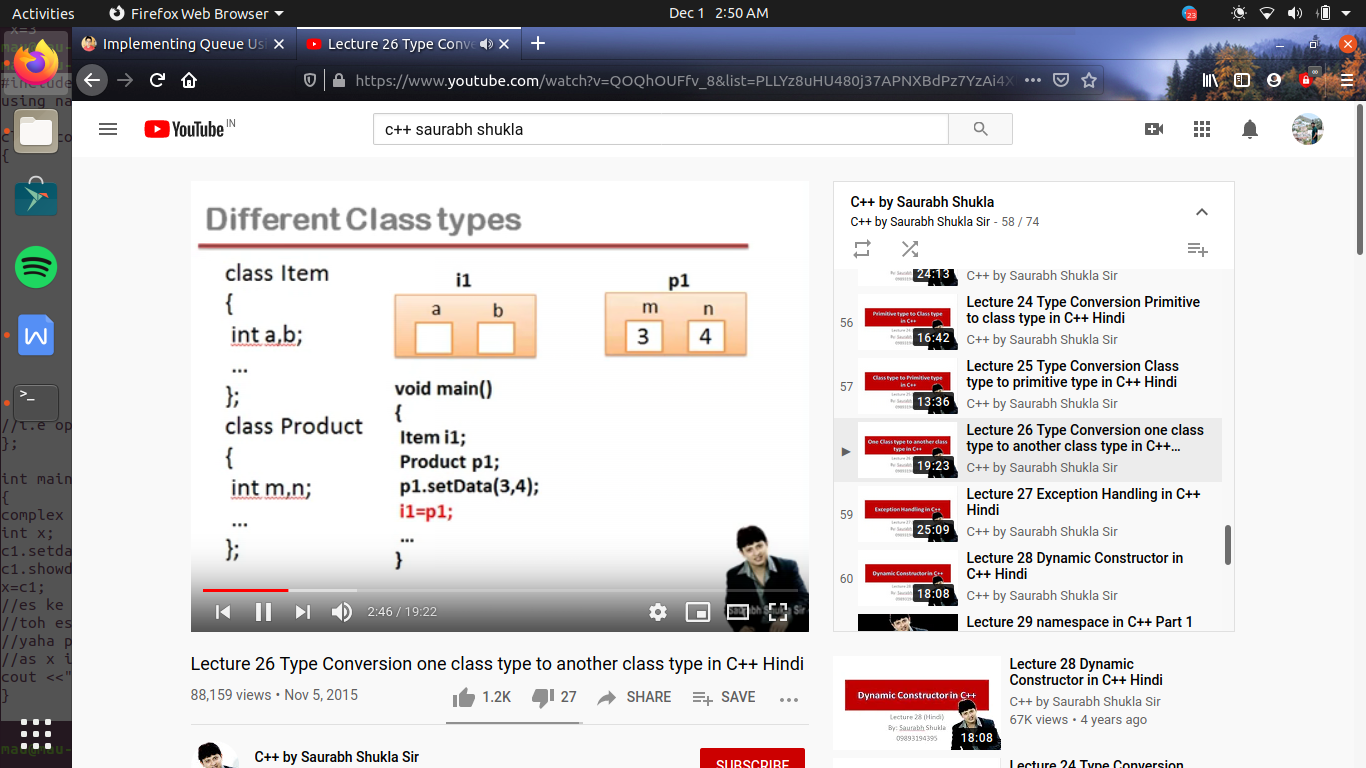
//as x int type ka hai toh casing operator jo return karana chahiye wo bhi return type ki value honi chahiye

cout <<"\n x="<<x<<endl;

}

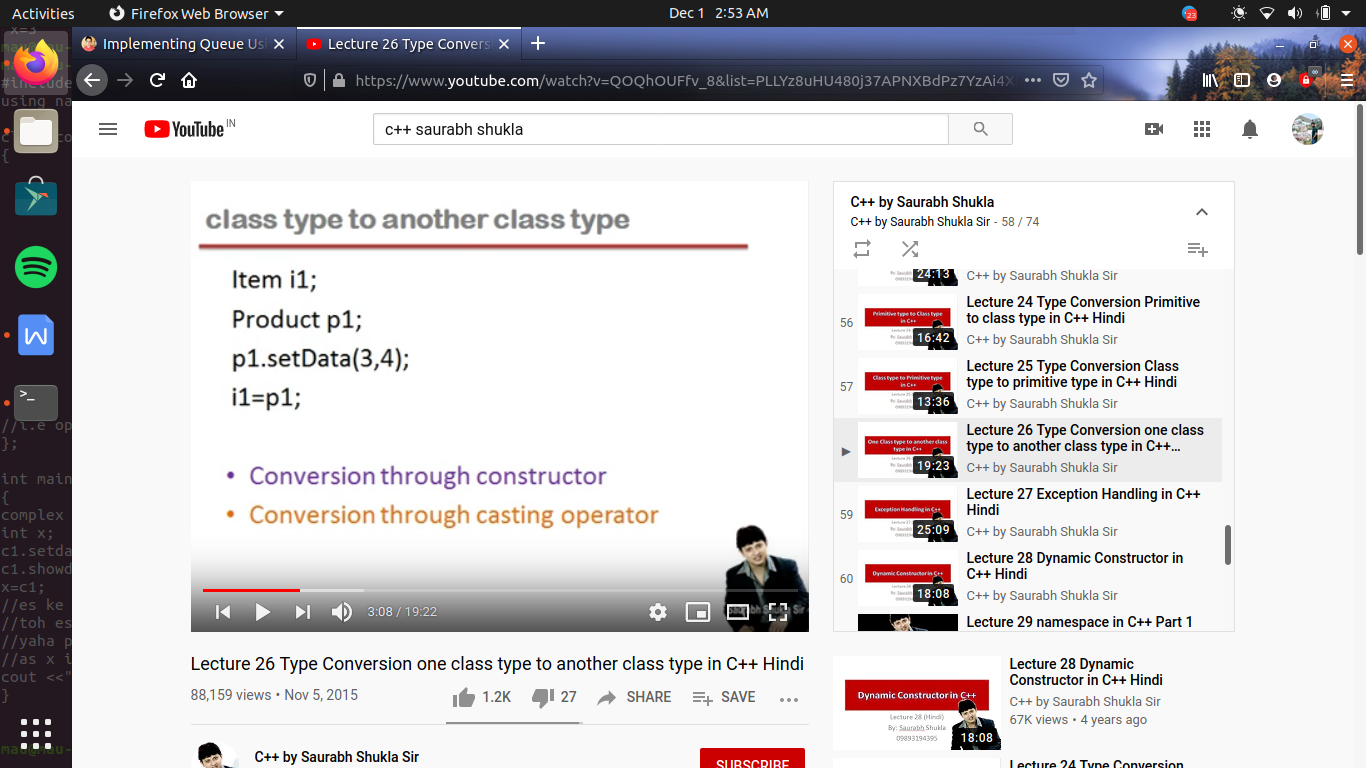


one class type to another class type conversion ->



toh yeh kam karane ke hamare pass 2 tarike hai ya toh hum yeh kam constructor ke through kr le or ya toh hum yeh kam

casting operator ke through kr le



ab agar hum constructor ke through conversion kr rahe hai toh , hame kis class ke leye constructor banana chahieye

toh agr constructor ke through conversion ke rahe hai toh assign ke left me jo object hota hai us ke leye constructor call hoga i.e here constructor item call me banaa hoga

or agar hum casting operator ke through conversion karwana chahate hai toh assignment operator ke right side jo object hoga toh us ke leye casing opearator chalega

/\* using constructor \*/

#include<iostream>

using namespace std;

class product

{

private:

int m,n;

public:

void setdata(int x, int y)

{m=x;n=y;}

void showdata()

{cout<<"\n m="<<m<<"\n n="<<n<<endl; }

int getM()

{return m;}

int getN()

{return n;}

};

class item

{

private:

int a,b;

public :

item () {} //default constructor

void showdata()

{cout<<"\n a="<<a<<"\n b="<<b<<endl; }

item(product p) // ab yaha pr kyu ki product nam ki class bad me bani hai pr hum ese pahale use kar rahe hai toh es ke leye hame ese pahale declare karana hoga

{

a=p.getM(); // as yaha kyu ki m or n ek private member hai product toh ab agar hum ese use kr rahe hote product class me hi toh koe error nhi ati but ab hum use kar rahe hai ese dusari class me toh error ayegi

//toh yeh value lene ke leye hum product class me function banayege // jo ki m or n ki value return kr dega

//ab kyu ki yeh function hum public me banayege toh hum ese item class se bhi access kr lege

b=p.getN(); // ab yeh fir se error dega ki getN() function ka decleration bad me kiya hai toh acha yahi hoga ki hum pura product class hi es ke upar define kr de

}

};

/\*

class product

{

private:

int m,n;

public:

void setdata(int x, int y)

{m=x;n=y;}

void showdata()

{cout<<"\n m="<<m<<"\n n="<<n<<endl; }

int getM()

{return m;}

int getN()

{return n;}

};

\*/

int main()

{

item i1;

product p1;

p1.setdata(3,5);

p1.showdata();

i1=p1;

// as when we are doing conversion through constructor , toh hum ese samjh sakate hai as i1=i1.item(p1); //as i1 call constructor and p1 pass as a argument

// pr agar hum sirf ek constructor banana dete hai or usme argument pass kar dete hai

// pr jab i1 object banate time constructor call hoga , or ab kyu ki i1 banate time koe argument pass nhi kar raha hai

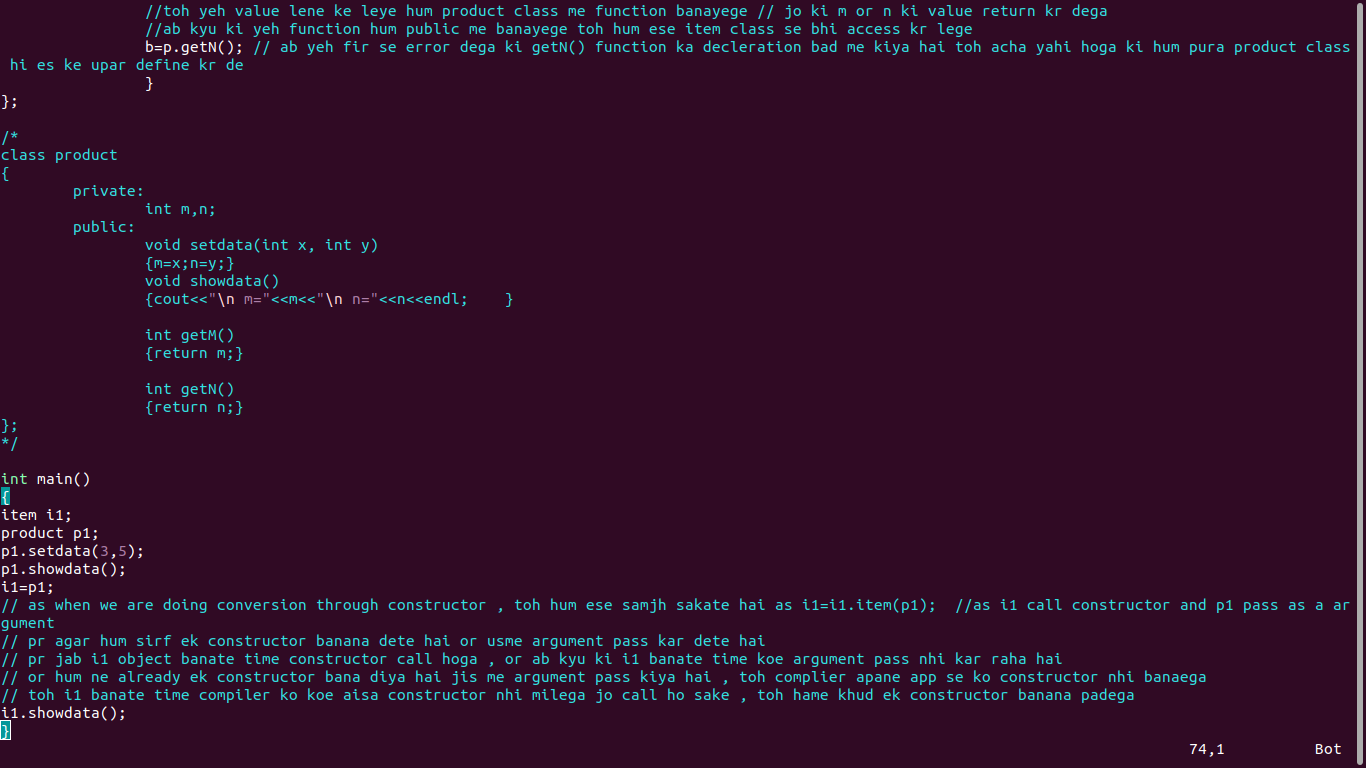
// or hum ne already ek constructor bana diya hai jis me argument pass kiya hai , toh complier apane app se ko constructor nhi banaega

// toh i1 banate time compiler ko koe aisa constructor nhi milega jo call ho sake , toh hame khud ek constructor banana padega

i1.showdata();

}





/\* by using casting operator \*/

#include<iostream>

using namespace std;

class item

{

private:

int a,b;

public:

void showdata()

{cout<<"\n a="<<a<<"\n b="<<b<<endl;}

void setdata(int x,int y)

{a=x;b=y;}

};

class product

{

private:

int m,n;

public:

void setdata(int x, int y)

{m=x;n=y;}

void showdata()

{cout<<"\n m="<<m<<"\n n="<<n<<endl;}

operator item()

{

item temp;

temp.setdata(m,n);

return temp;

}

};

int main()

{

item i1;

product p1;

p1.setdata(3,5);

p1.showdata();

i1=p1; //now as we are converting it as a casting operator

//so on casting operator , hum assign ke right side wali class me casting operator function banate hai

//toh ese hum aise bhi samajh sakaate hai i1=p1.operator();

i1.showdata();

}

