virtual function in c++

agar hum pointer parent class ka banata hu toh wo us ke child class ke object ka address contain kr sakata hai

as we earlier know that pointer jis type ka hota hai wo usi type ka variable contain karata hai , or hum janate hai ki nayi class banana mtlb naya data type banana

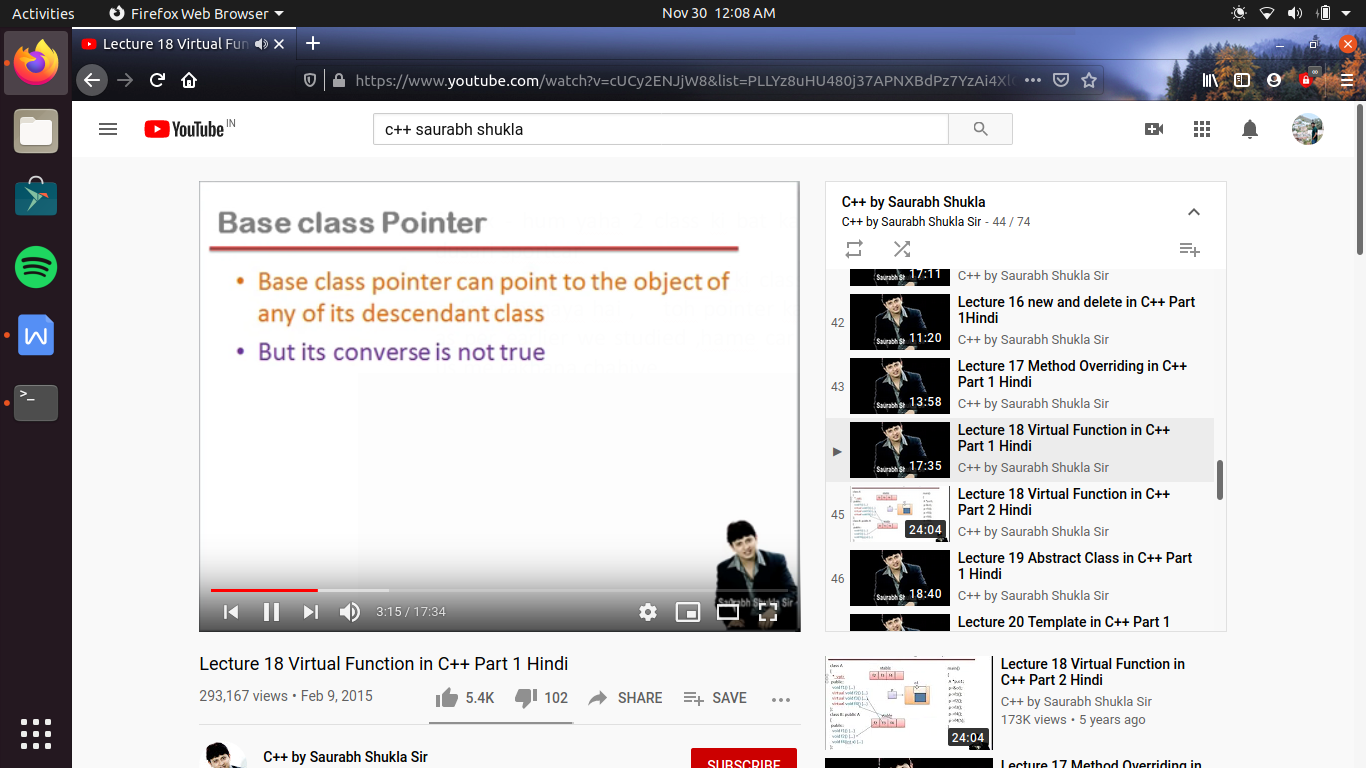
for ex - hum yaha 2 class ki bat karate hai ek hai car dusari sportcar

ab man le hum ne car nam ki class hai us ka hum ne pointer banaya hai , toh pointer ka type hua car , toh as per earlier we studied ,hame car class ka hi address us me rakhana chahiye ,

but yaha rule yeh bhi hai ki agar hum ne parent class ka pointer banaya hai toh wo us ke child class ko bhi point kr sakata hai , or sirf child class nhi , child class ki child class fir us kii bhi child class toh use bolate hai decedent classes(that means vanshag)

toh ek class ke neeche inheritance ke chalate kayi sari classes hai , toh parent class ka pointer apane neeche wali kisi bhi class ko point kr sakata hai , pr es ka ultha sahi nhi hoga

yani child class ka pointer , parent class ko point nhi kr sakata



#include<iostream>

using namespace std;

class A

{

public:

virtual void f1() //virtual function , to tell compiler that its has to perform late binding

{}

};

class B :public A

{

public :

void f1() //function overriding

{}

};

int main()

{

A \*p, a1;

B obj;

p=&obj; //yaha hum pointer jo parent class ka hai, child class ke object ka address contain kr raha hai

obj.f1(); // run f1() contain in class B beause hum class B ke object f1() ko call kar raha hai

//lekin jab hum parent class ke pointer ko call karate hai , or us ke through methods ko call karate hai toh kahani thodi different ho jati hai

p->f1();

//yaha pr hum ne parent pointer ke through f1() function ko call kiya

//ab yaha pr f1() ko call koe kisi class ki object se nhi kr rahe hai balaki pointer se kr rahe hai

//or yeh bhi hum janate hai function ki call ko dhek kr usaki sahi version ko bind karane ka kam compiler karata hai or es ko early binding kahate hai

//ab early binding abhi bhi hogi jab hum pointer ke dawara call kr rahe hai

//but ab problem kya hai ki jab early binding hoti hai , or compiler yeh kam kr raha hai toh ese yeh nhi malum ki pointer me addressn kis ka rakha hai

//ese leye compiler ke sahi version ko dhudane ke leye yeh nhi jan payega ki pointer ke andar a1 ka address hai ya obj ka

//kyu ki compile time pe address nhi pata chalge ki address kya hai , kyu ki address tabhi allocate hote hai jab program run kar raha hota hai

//toh compiler time pe toh , pointer ka type hi keval samajh me a skata hai , kyu ki pointer ka type decleration me fix ho chuka tha ,

//or yaha pointer ka type hai A

//es ka mtlb early binding karate time compiler pointer ke type ko deekhega na ki us ke content ko

//toh funtion ko agar call object ke dawara kiya jata hai toh object ke type ko deekhate hai

//but agar pointer ke dawara funtion ko call kiya jata hai toh pointer ke type ko deekahte hai

//toh ese wajah se kyu ki pointer A type ka hai toh compiler A class me jayega or dhudega ki waha koe f1() nam ka function hai kya or agr hai toh

//wo use ke sath bind krdega or ese wajah se es bar parent class wala f1() chl jayega

//ab yaha yeh toh galat hai na kyi ki pointer ke andar toh hum ne class B ke object ka address contain karawaya , pr compiler ese nhi deekhega

//or run kar dega parent class wale f1() ko

//toh yeh overriding ki problem hai kyu ki hum overriding karate hi ese leye hai ki jo latest definition ho wo run kare pr yaha aisa nhi ho raha

//toh es ka solution yahi hai ki early binding na hone diya jaye kyu ki early binding se hi , compiler pointer ka type dhek kr parent class ke f1()

//function ko bind kar raha hai

//toh hona yeh chahiye ki late binding ho, i.e run time pe binding ho i.e run type pe binding kiya jaye

//run time pe ese leye kyu ki tabhi pata chl payega ki pointer me content kya hai

//toh ab yeh bat compiler ko kaise samajhaye ki early binding nhi karana hai or ese late binding hone dena hai

//late binding ko hum dynamic binding bhi bolate hai

//toh aisa karane ke leye hum jo parent me function bana rahe hai usb ke peeche virtual keyword likh dete hai

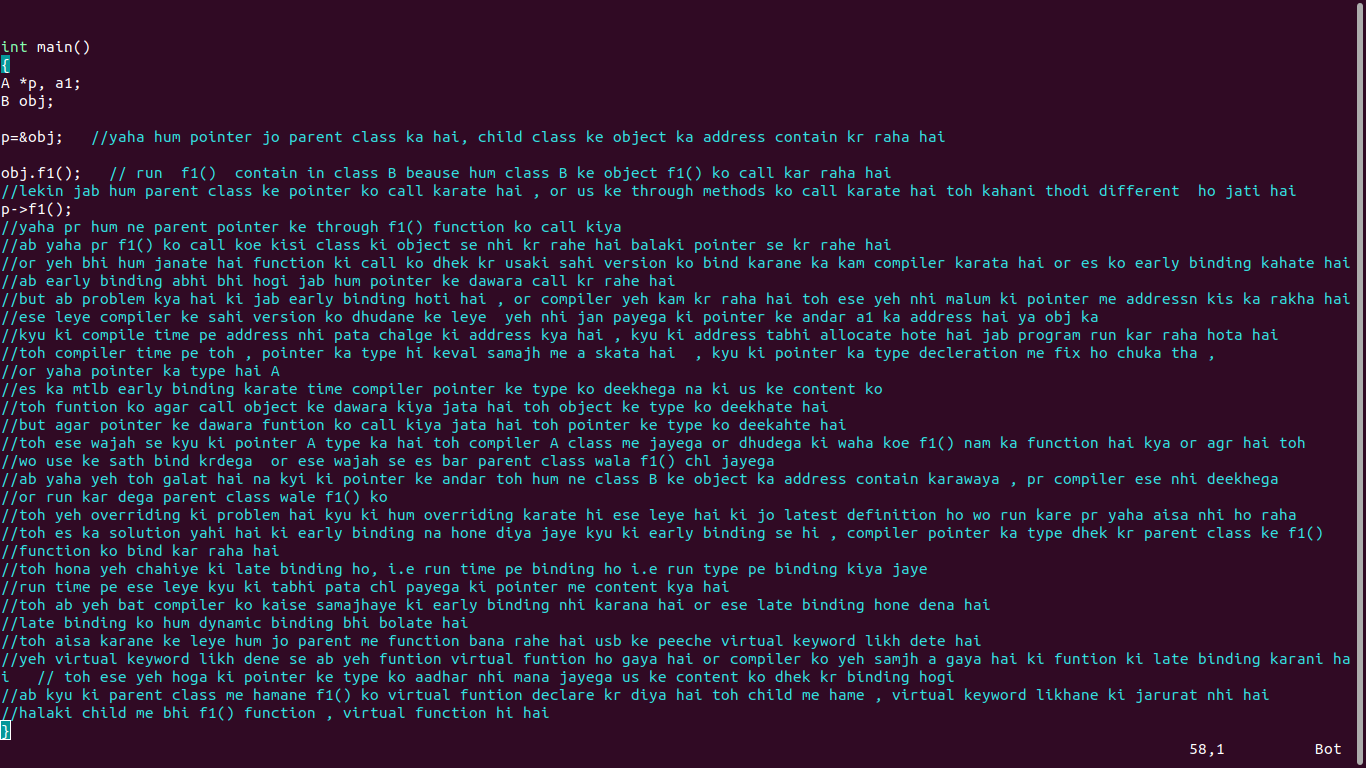
//yeh virtual keyword likh dene se ab yeh funtion virtual funtion ho gaya hai or compiler ko yeh samjh a gaya hai ki funtion ki late binding karani hai // toh ese yeh hoga ki pointer ke type ko aadhar nhi mana jayega us ke content ko dhek kr binding hogi

//ab kyu ki parent class me hamane f1() ko virtual funtion declare kr diya hai toh child me hame , virtual keyword likhane ki jarurat nhi hai

//halaki child me bhi f1() function , virtual function hi hai

}





how virtual function concept works :-

rules ->

pahala concept yeh hai ki jis bhi class me ek bhi virtual function hoga , us class ke leye compiler , apani tarf se ke variable as a member declare kr dega

jaise us ne banala

\*\_vptr;

ab hum es ka sabut deekhane ke leye , hum us class ka size nikal sakate hai , hamesa 2 bytes jayada hi deekhega , toh tab samajh jayege ki es tarah ka ek pointer variable us class ke andar bana hua hai

lekin wo variable us ke child class ke leye alag se nhi banaega , ab yeh kyu nhi banega

kyu wo variable instance member variable hai

toh aagr A class ka object banega toh us ke andar yeh pointer hoga , or jab B class ka object banayege toh bhi toh wo pointer us me hoga kyu ki wo inherit ho jayega parent class se , toh B class me alag se \*\_vptr banane ki kya jarurat hai

ab dusara kam compiler yeh karata hai ki compiler ek array bana deta hai , or yeh array kasia banata hai ,static array

toh yadi hum koe static variable class me banate hai toh hame pata hai ki agar us class ka object banyege bhi nhi tabhi us variable ko memory mil jayegi , or ek hi banega chahe jitane object banale .

static array hai mtlb ek ke andar bahot sari jagah hai, or yhe kaisa array hai , yeh ek pointer ka array hai, mtlb es ke andar sare variables kya hai , pointer variable ,

or yeh function pointer variable hai mtlb en me kisi variable ka address nhi rakha jayega, kisi function ka address rakha jayega i.e keval virtual function ka address hoga

or es array ko nam diya jata hai -> vtable

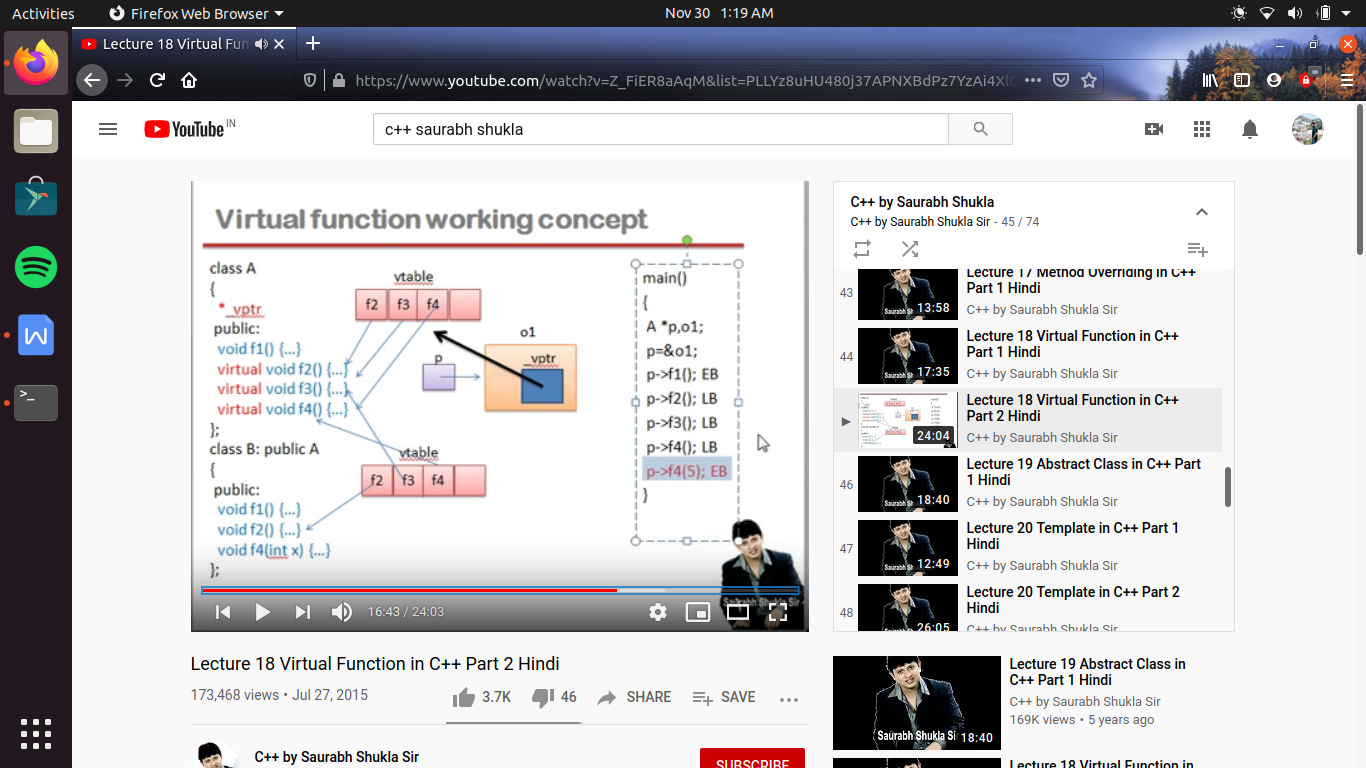
or jis bhi class me virtual function hai us ke leye vtable banega , or uske sabhi decending classes ke leye bhi vtables banegi alag alag , es ka mtlb class A ke leye toh bana hi hai class B ke leye bhi bana hai

ab yeh yeh \*\_vptr contain karata hai es vtable ka address

ab agar A class ka object hoga , toh \*\_vptr , class A ki vtable ka address contain karega

or agar B class ka object banega toh \*\_vptr

class B ke vtable ka address contain karega



ab yaha jaise function bane hai , jis ke me virtual keyword likhe hai wo ho jayega late binding (LB)

ab jaise ki

main()

me hum ne ek pointer banaya p, A class ka , or wo contain kar raha hai , o1 object ka address ,

when pointer calls f2() function , so as pointer contain address of object o1 and o1 have a pointer name \*\_vptr made by compiler , as p points contain address of object

o1 having class A , thus \_vptr points vtable of class A, and by \_vptr ,jis function ki f2() name se entry hogi wo function run ho jayega

as at the last p->f4(5); as vtable me f4() nam ka function hai toh pr pr us me hame arguments chahiye kyu ki main se hum 5 pass kar rahe hai , toh class A me koe aisa function hi nhi hai jo single argument lele , toh es line me ayegi error

or agar hum ne pointer banaya class A ka i.e parent class , or wo contain kr raha hai address of object of child class toh \_vptr ab point karega vtable jo ki class B me ba ni hue hai , or as jo bhi entry hogi vtable me wo un ho jayega

