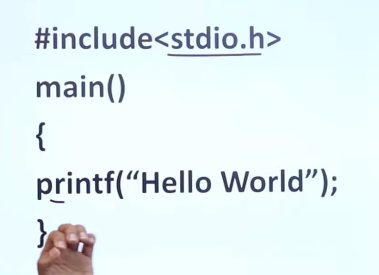
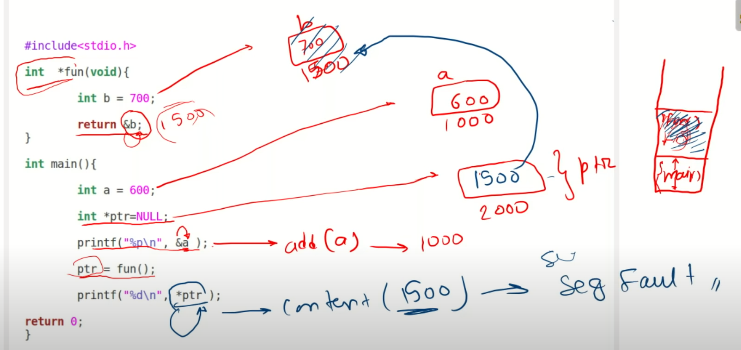
**C Programming**

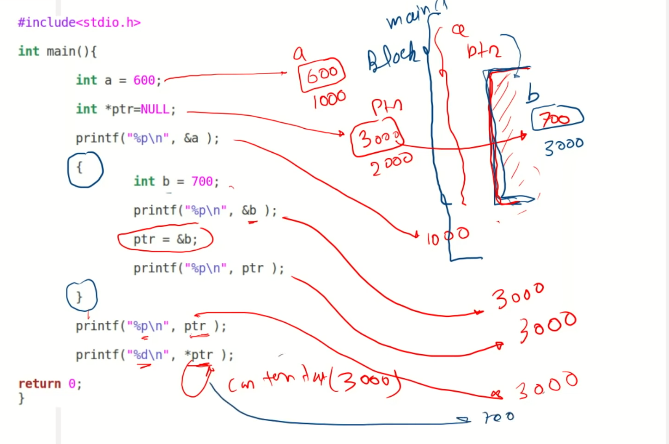
 f means format, so print in format.

Dangling Pointer

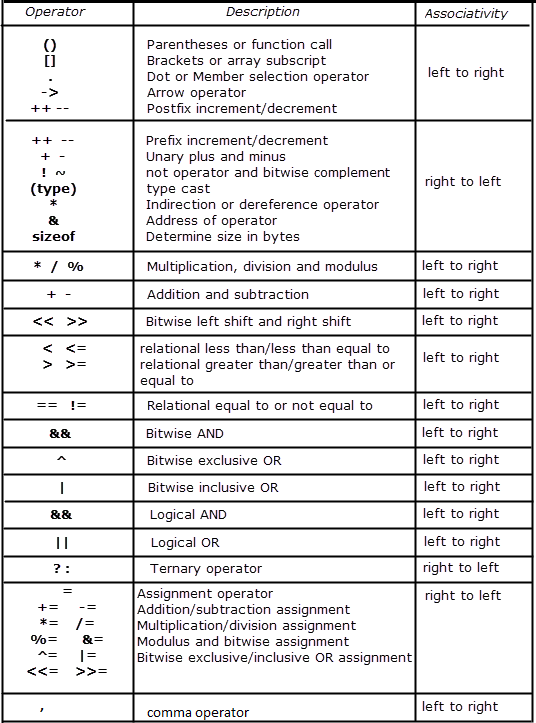


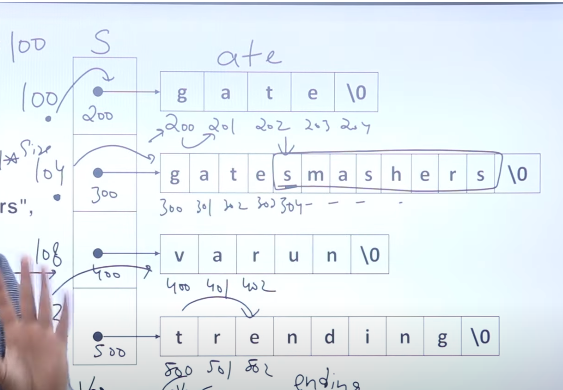
Ptr=fun()

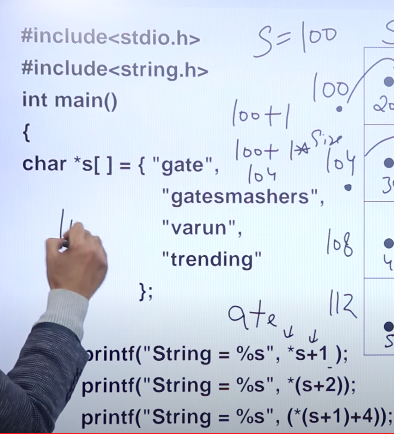
After the function was called it returns the value of B after that the stack becomes empty and due to that weld out \*ptr is printed there is no content left that is the reason segment fault is present. This is called dangling pointer to an unallocated memory.



The address of B will be printed then the address of B will be given to the pointer and the pointer will be printed after coming out of the scope when pointer is again printed as the pointer is storing the address of B it will give 3000 and when it is dereferenced it will give the value 700 this is called dangling pointer pointing out of scope.

s





s=200. \*s=g,

printf(“string=%s”\*s+1)=ate

s+2=400, printf(“string=%s”\*(s+2))= varun

printf(“string=%s”(\*(s+1)+4))=ding

ACII Value

65 A 97 a

C++ OOPs

Int main{

Rectangle \*p;

P=new Rectangle;

Rectangle \*p= new rectangle(); // creating object in heap

() can or cannot be there.

}

Data member should be made private i.e. Data hiding.

r.length is not possible because it’s a private member. U cant set values

Just use class function.

Property Function -

SetLength() =Mutator

Getlength() = Accessor

Area(), Perimeter() = facilitators

Int isSquare() = returns yes or no so it’s a inspector.

Inheritance indicates the code reusability. Encapsulation and abstraction are meant to hide/group data into one element. Polymorphism is to indicate different tasks performed by a single entity.

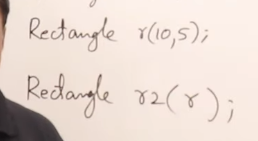
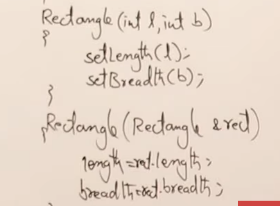
As Java supports usual declaration of data variables, it is partial implementation of OOP. Because according to rules of OOP, object constructors must be used, even for declaration of variables.

Constructor doesn’t have a return type

Void constructor(){} is wrong

Constructor(){}

Copy Constructor.

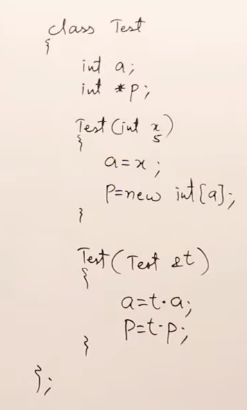
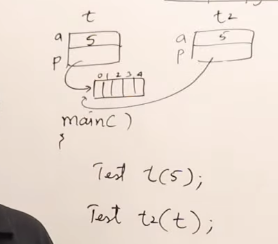
 

Non parameterized, parameterized, Copy constructor

When u make all these constructor with the same name its called constructor overloading.

Deep Copy Constructor.

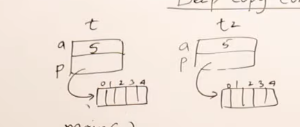
Problem with copy constructor.

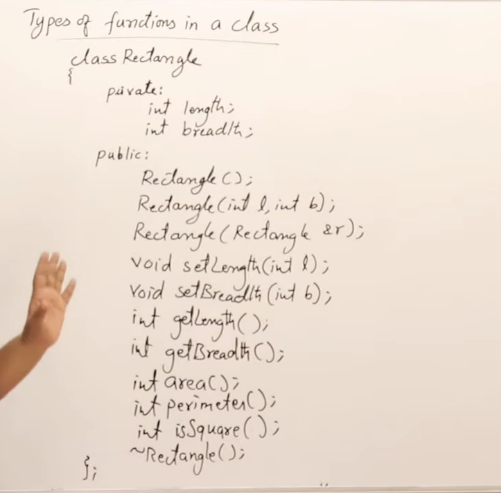
 ’

The copy constructor’s p pointer will be pointing to the same array only

S instead we write.

P=new int [a]

 This is a deep copy constructor.

Use scope resolution operator to write the function definitions outside the class.

If u write any function inside the class only it is the part of the main() function only.

Int rectangle :: Perimeter(){} – now it’s a separate function.

Rectangle::rectangle()

Rectangle::rectangle(int l, int b).

In c++ we can write data and function in struct not in c. Since C is a procedural programming language, the data and functions stay separate in it

Empty class ki size = 1Byte

Dynamic Memory allocation

Int \*i=new int;

Hero \*b=new hero;

Default wala constructor bnana hi hoga agr hum baaki constructors banate hain tb.

Agr static allocation k liye destructor automatically call hota hai

Hero a;

Dyamic k liye manually call krna padega.

Hero \*b=new hero;

Delete b (likhne s ab destructor isk liye bhi call hojayega.)

Static keyword this belongs to the class not the object, directly call krna hai isko.

Static functions me “this->” keyword ni hota. They can access static member only. Kisi aur ko ni.

New keyword and Delete keyword

    // Allocate Heap memory

**int**\* array = **new** **int**[10];

    // Deallocate Heap memory

**delete**[] array;

inheritance -

class Human{}; => Parent

class Male : public Human{} => child

Types of Inheritance

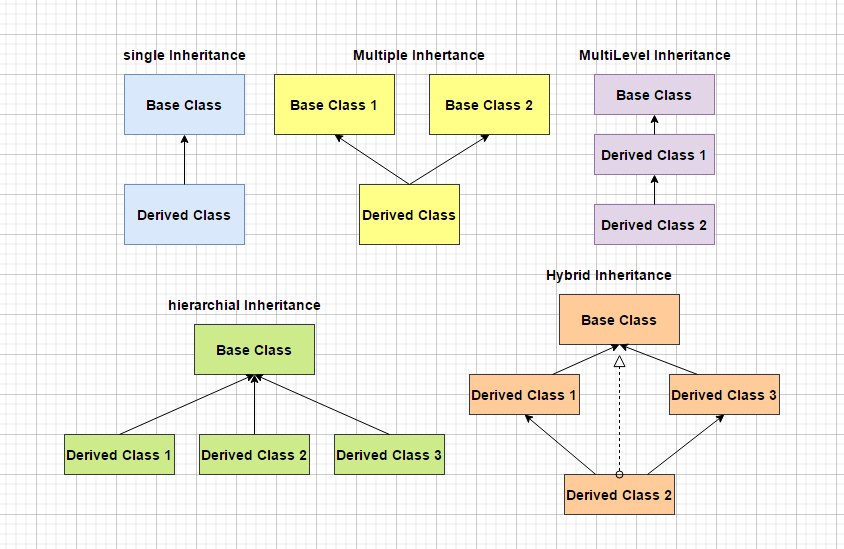
Single,

Multiple,

Multilevel,

Hybrid,

hierarchical.



Polymorphism –

Compile Time (static Polymorphism)– Function(method) and Operator Overloading

Operator Overloading) - :: \* . ?: sizeof() cant be overloaded.

Run time (Dynamic Polymorphism) –

Function overriding ka mtlb hota hai ki agr Parent class aur child class dono m same function hai toh child class ka function call hoga.

Abstraction means Implementation hiding done using access modifiers.

In Operator Overloading – c=a+b, u r calling +(Function) in reference to a and u r passing b as an argument and putting it in c.

ClassName operator+ (ClassName b) {

ClassName c;

c.a=a+b.a; as we have called ‘b’ as argument and ‘a’ as a reference

return c

#include<iostream>

using namespace std;

class A {

public:

void func() {

cout << " I am A" << endl;

}

};

class B {

public:

void func() {

cout << " I am B" << endl;

}

};

class C: public A, public B {

};

int main() {

C obj;

//obj.func();

obj.A::func() ;

obj.B::func();

return 0;

}

Exception Handling – are some unpredictable circumstances when our program terminates suddenly with some error/issues. These happens during catch running of the program.

Exception handling provides us with the facility to handle these exceptions so that the program keeps running

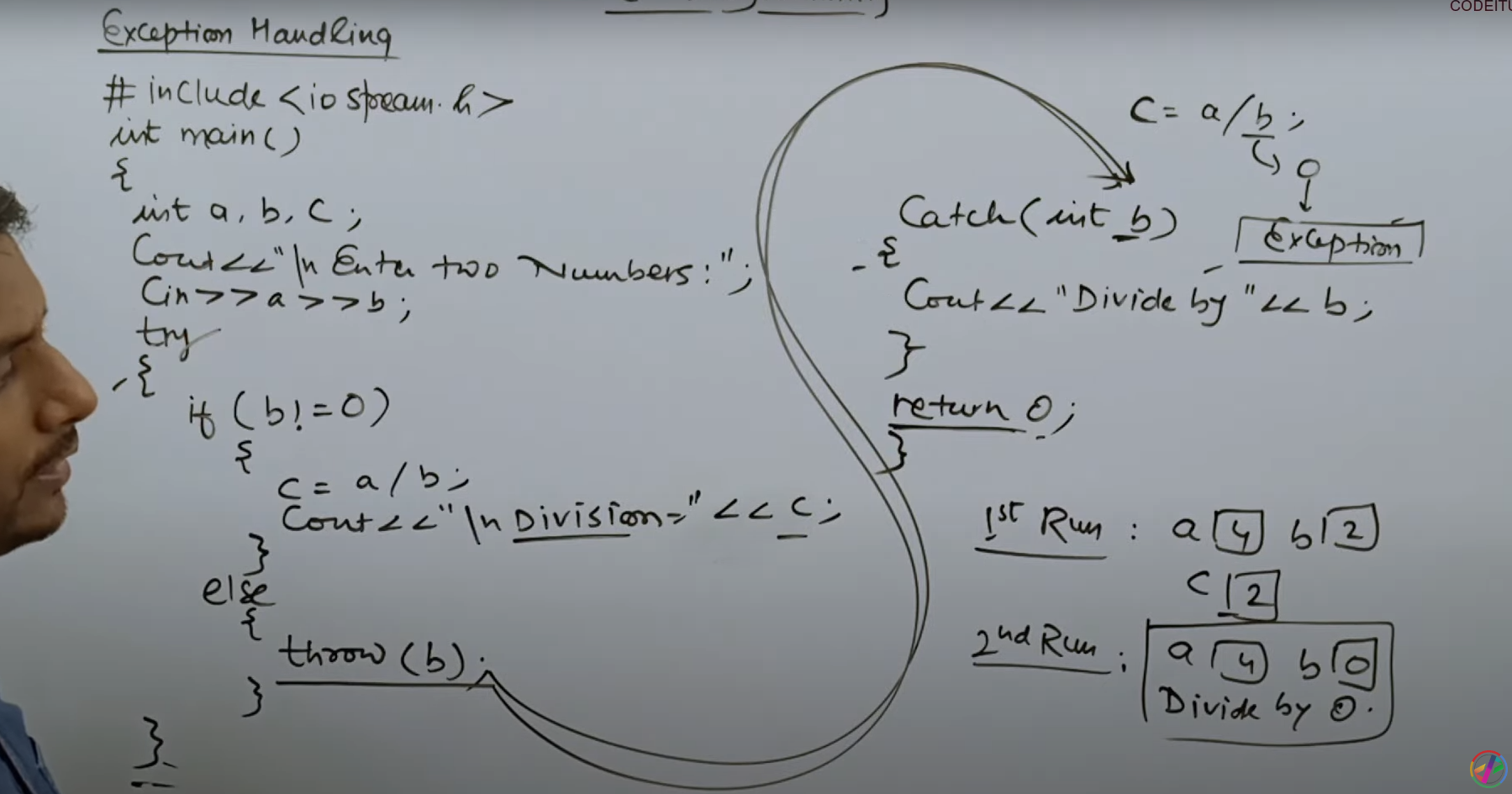
How to perform exception handling

1. Hit the exception
2. Throw the exception
3. Catch the exception
4. Perform the corrective action.

Try () – This block contains code that might contain exception

Catch () – Whenever some exception is found in dry block it is immediately thrown to catch block now catch block will have responsibility to handle the exception.

Catch block should be below the try block and there can be multiple catch block for a Try block.



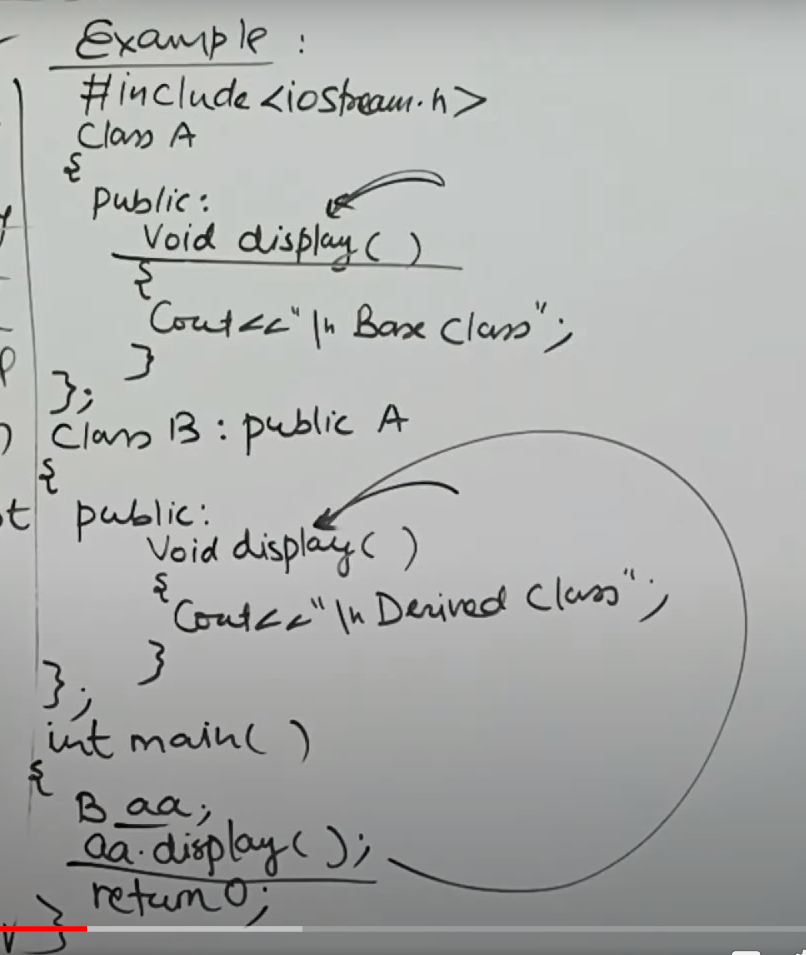
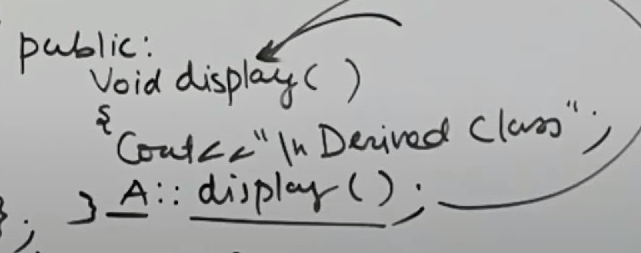
And function overloading we declare more than one function same name and different types of parameters

In function overriding we declare a function in base class and derived class with the same name and same parameters

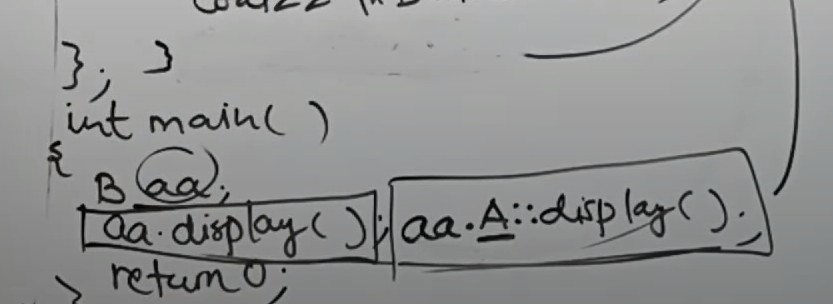
Functional loading is a concept of compile time polymorphism and function over writing is a concept of runtime polymorphism

Function overloading occurs without inheritance and function over writing occurs when child classes inherited from parent class.

Function Overriding –

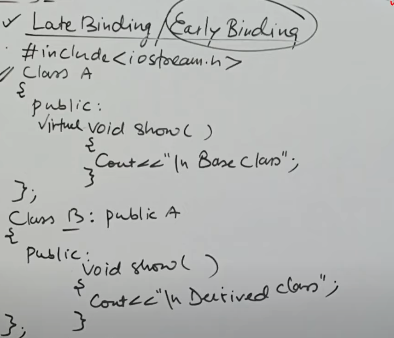
 

Derived Class Derived class, Base class



aa.A::display() – base class

Virtual Function –



Early binding ka mtlb hota hai upr k teeno cases me we know kya call hone wala hai.

Late binding –

A\* ptr;

**A aa;**

Ptr= &aa

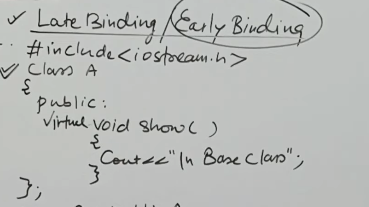
Ptr->show(); Base class ka call hoga.

A\* ptr;

**B aa**;

Ptr= &aa

Ptr->show(); **Base class ka hi call hoga.**



Just add virtual to the base class function-

A\* ptr;

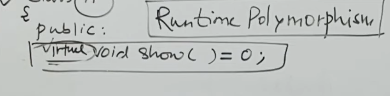
**B aa**;

Ptr= &aa

Ptr->show(); **Derived class ka hi call hoga.**

**Run time m decide hoga ab kaun sa call hone wala hai – Run time Polymorphism.**

**Pure virtual function –**

****

Abstract Base class bni hi isliye hai taaki uss se derived classes banai jaaa ske and Pure virtual function likhne k baad harr ek derived class m ek function of same name hona chahiye.

Friend Function –

class Complex{

int a, b;

**friend Complex sumComplex(Complex o1, Complex o2);**

public:

void setNumber(int n1, int n2){

a = n1;

b = n2;

}

// Below line means that non member - sumComplex funtion is allowed to do anything with my private parts (members)

void printNumber(){

cout<<"Your number is "<<a<<" + "<<b<<"i"<<endl;

}

};

**Complex sumComplex(Complex o1, Complex o2){**

Complex o3;

o3.setNumber((o1.a + o2.a), (o1.b+o2.b));

return o3;

}

int main(){

Complex c1, c2, sum;

c1.setNumber(1, 4);

c1.printNumber();

c2.setNumber(5, 8);

c2.printNumber();

**sum = sumComplex(c1, c2);**

sum.printNumber();

return 0;

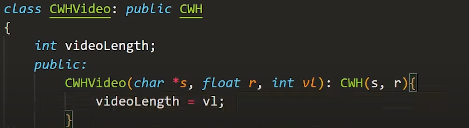
}

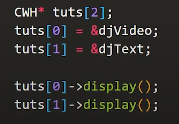
That is not in the scope of the class and health cannot be called from the object of that class.

Friend Function between 2 classes

friend int Calculator ::sumRealComplex(Complex, Complex);

CWH is a constructor called during inheritance.





Virtual functions cannot be static the access by object pointers virtual functions can be friend of another classes

Dynamic cost operator is used for conversion of Poly Morphic types

Type ID operator is used for identifying the exact type of an object

Type\_info – hold the typr imformation returned by typeid operator.