Post mids

2-05-2024

Static variables:-

local variable is not able to maintain its state.

```
void test()
{
    int s=0;
    cout << s;
    s++;
}
```

```
static int s=0 —①
cout << s;
s++;
```

- ☆ static variables goes to data segment - in block scope
- ☆ the variables that goes into data segment can not be ~~executed~~ destroyed throughout the life of code

```
//0 test();
//1 test();
//2 test();
```

→ in first iteration statement ① will execute.

→ in this iteration st ① will not be executed because it is a static variable and its state is fixed in Data segment. only initialize one time

→ static variables are always initialized with 0 by default.

→ scope of local/static variable is within that function.

→ scope ⇒ local

→ life ⇒ global

→ Static variables are declared in the class. They are data members of class.

→ Static data member is not a part of your object

```
int a, b;
static int s;
static float f;
```

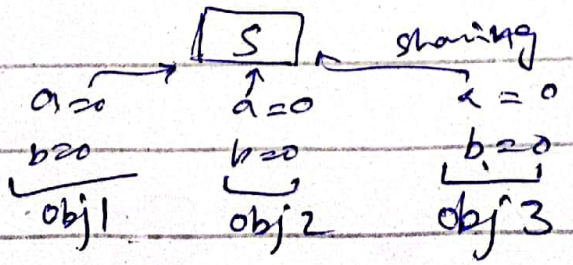
```
cout<< sizeof (obj1);
}
```

→ All the objects can share a static variable

→ It is just like a global variable for your objects.

→ It will create own its own wether the objects will create or not.

→ Static variable create its own space.



→ Circle class. It ~~is Also~~ have to be create a static variable so that it

can be shared by all objects

→ This pointer have don't have any link with the static variable.

It is just declared not defined.

outside the class.

int Algebra :: S; // Defination

Here it will be initialized with zero.

```
cout << obj.1 a; 0
obj1.S++;
cout << obj.2 ; 1          S is shared by all
obj2.S++;      2           objects.
cout << obj1.S; 2
```

output

012.

static variable is called "class data member".

a=0, b=0 → instance data members.

```
int main(){
    cout << Algebra :: S << endl;
```

How to access static data member?

Algebra :: S;

Q: How to check that how many objects are created ~~or~~?

→ Use static data member.

→ increment it in all constructors.

→ Decrement in destructor.

```
sabsy pehlay yeh 1 hua hoga
void test (){    Algebra obj1(1,3), obj2(1,2), a, b;
                                         calls. }
1 ← Algebra g1;      // global function
int main (){
    test ();                     // 5
    cout << Algebra :: s << endl;   // 1.
```

All the functions we have written are ~~non~~ none static M.F.

This pointer doesnot exist in static member function

```
static int getS()
{
    ~~int~~ a = 5;     → this will also be a error.
    return s;
}
```

Here we have created private S.V and we are accessing it in main function

```
cout << Algebra :: getS() << endl;

static void setS(int s)
{                    → works as this pointer for SV
  Algebra :: S = s;
}
```

Compare two arrays with static function

Because there is no need of left hand side object - we only pass arrays in parameter list.

```
static bool compareArrays(const Algebra ar1[], const Algebra ar2[], s1, s2)
```

Define → outside the class.

Declare → inside the class.

~~Public~~ Private data variables can be accessed within static functions by declaration of object within that function.

6-05-2024

Array class:-

→ Pointer as a data member-

when we create array, we will create it dynamically so data member will be pointer and size of array.

```
class Array {
    int *p;
    int size;
public:
    Array : size{5}
    {
error// *p = 88;
```

→ we are trying to allocate an addressed memory. Runtime error, program will crash.

```
    p = new int [size];
    for(int i=0; i<size; i++)
    {
        p[i]=0;
    }
```

Better practice is to store some values in our array otherwise it will store some garbage values.

When we are creating simple objects we just call simple destructor but in case of array class we are creating dynamic array on heap so we have to deallocate the heap memory. p and size will destroy but

array creates memory leaks.

So,

```
~Array(){
    delete[] p;
}
```

Big three's :-

1) You have to write the destructor when you dynamically create something in your class

We will definitely allocate memory to the pointer in all constructors.

```
Array(int size) : size(size)
{
    p= new int(size);
    for(int i=0 ; i<size; i++)
    {
        p[i]= 0;
    }
}
```

We ~~can~~ also need a constructor where size is passed along with the pre existing array.

```
Array( int a[], const int size) : size(size)
{
    p= new int (size);
    for( int i=0; i<size; i++)
    {
        p[i] = a[i];
    }
}

int main(){
    int a[5] = {1,2,3,4,5};
    Array obj1(a,5);
```

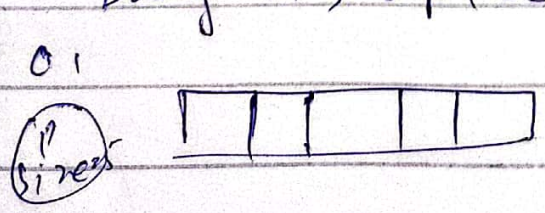
When the default constructor is called (of compiler). it perform two function.

```
This -> size = obj.size;
this -> p = obj.p;
```

Shallow copy constructor

The main mistake is that we are breaking data hiding i.e. a new object is refencing/accessing to the data of another object means data memeber of one object is accessed by another object - Data ~~member~~ hiding and encapsulation is vanished -

Array o1, op(oc)

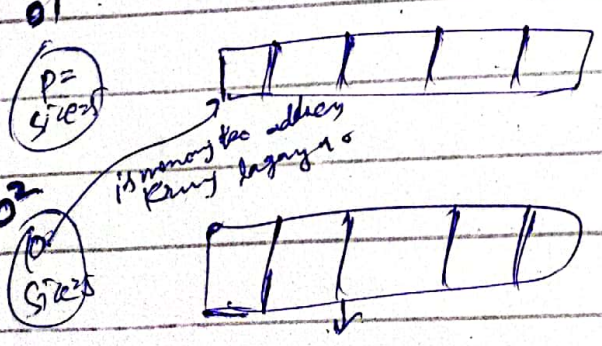


Another error is that the pointer of newly created object gets the address of already created object and when the first any object gets out of scope the pointer of new object gets deleted but the old object's pointer is not deleted that creates dangling pointer -

This is called deep copy constructor.

```
Deep Array (const Array & obj) : size{obj.size}
{
    p = new int [size];
    for (int i=0 ; i< size ; i++)
    {
        p[i] = obj.p[i];
    }
```

Assignment operator :-



when we write in assignment operator.

```
p = obj.p
size = obj.size;
```

Assignment is possible if and only if both objects have same size

or agar koi bhi object out of scope jaye ka to dangling pointer.

```
Array operator = (const Array & obj).   // and we will also check size of both arrays
{
    if (this != &obj && size == obj.size)
    {
        for(int i=0; i<size; i++)
        {
            p[i] = obj.p[i];
        }
    }
    return *this;
}
```

~~3rd~~ Big three:

2) copy constructor (deep copy constructor).
3) definitely ~~write~~ assignment operator overload.

When we have to do assignment (right side data is copied in L.H.S object) irrespective of sizes of arrays. we will have to delete the memory that was allocated to left hand side object and then make the size equal to the size of ~~sig~~ R.H.S object.

```
Array operator = (const array & obj).
{
    if (this != & obj)
    {
    delet [] p;
    this -> size = obj.size;
    p = new int [size];
    // copy array;
    }
```

1 23 45 45

4 6

6-4 = 2+1

in case of int data type we have to return something.

return; → it is control return and is done by void, constructor etc.

8-05-2024

Sub-script operator

Polymorphic behavior of this operator.

cout << a1[3]; // r-value. // getting the value.

L-value it is always contains memory ← a[1] = a; // value is assigned to it.

* index would be a parameter.

* We should make the function constant so that it can also work for constant objects -

* Bound checking so that user can not access the memory greater than size.

returning integer [For right value] - it will work for both constant and non-constant objects

```
int operator [] (const int index) const.
{
    if (index >= 0 && index < size)
    {
        return p[index];
    }
    cout << " Error";
}
```

conditional return.

// If user goes out of bound and here is condition that // we have to return a value (int) than we will exit // the program.

// exit (0); // program terminate.

cout << a[i] = a;

For l-value we will create another function

↓dont write const.

```
int& operator [ ] (const int index)
{
    if(index >=0 && index < size)
        return p[index];
    cout error;
}
```

"int k case mein humain us index ki value return horahi thi is case mein aisa ho raha hai k use index ka new name ban k return ho raha hai main mein again koi change ho ga to function mein aa k us index py bhi change aye ga."

it is just like this:

```
int a=5;
int & r=a;
```

Exception ~~through~~:- throw:

In our case if use wants to access memory greater than size of array we will through exception.

Example:-

```
int divide (int num, int den)
{
    if (den != 0)
        return num/den;
    throw -1234567;   → we can write any string.
}
```

ie:fob

```
int main(){
    cout<< divide(99,0);
}
```

This exception is catched by window.

```
try
{
    cout << divide(99,0)<<endl;
}
catch (const int ex)
{
    cout << ex <<endl;
}
```

The code that may create error is placed in this in main.

what to do if exception occurs.

By writing this code the program will not crash rather show exception we want to show on console.

Method to catch string

```
catch (const char* ex).
```

```
int& Operator[](const int index).
{
    throw "Error Array index out of bound";
}
int main()
{
    Array o1(5).
    try
    {
        o1[6]=99;
    }
    catch (const char* ex)
    {
        cout << ex <<endl;
    }
```

Compiler exceptions

throw runtime_error("Error: ");

object

```
int main()
{
    try
    {
        cout << divide(99,0);
    }
    catch(const runtime_error& ex)
    {
        cout << ex.what() << endl
    }
```

For array class:-

```
int & operator[](const int& index).
{
    throw out_of_range("Error");
}

int main(){
    try{
        cout << a[7] << endl;
    }
    catch(const out_of_range& ex)
    cout << ex << endl;
```

cppreference.com.

Exception:-

It is a class that will catch all type of errors.

we will write only exception in catcher. and it will catch all types of exceptions

When something is not existing in our code it is not a good practice to return null we have to throw exception

13-05-2024

Templates :-

We will just write the procedure once and change the data types on its own.

* They have no existance just like blue print.
* They have no memory.

Syntax :-

```
template < typename t >          (can also write class)
t add (t a, t b)
{
    return a+b;
}
```

// it is a function template.

It will neither create nor execute but it will create other functions.

```
int main()
{
    cout << add (1,2) << endl;
    cout << add (1.1f, 2.2f) << endl;
    cout << add ('a', 'b') << endl;
```

add (1,2);

It will generate a version/instance of a function

```
int add (int a, int b)
{
    return a+b;
}
```

→ It is called template function

This instance will create once based on the nature of call.

Once it is created it will stay in memory and if it is called again then it will not regenerate the versi it call already created version.

add(

↓ at backend + add<t>(v1, v2)

add < int>(1,2)

add <float>(1.1f, 2.2f)

add <char>('a','b')

Compiler is implicitely passing the data type -

But it will start showing error when we pass "two different data types" in parameter list

Solution :-

add <float>(1,1.1f)

↙ float ka template function generate ho jaye ga.

delete[] ...

It all work on compile time.
It will make obj and/exe file bulky.

```
templelate <type name t1 , type name t2)
t2  add(t1 v1 , t2 v2)
{
    return v1 + v2;
}
```

If we write a Template function for array than we will not make unknown type for array size it should be int-

```
templelate<typename . t_array>
void printArray(const t_array a[], const int ar_size)
{
}
```

C++ gives use a functionality that we can use templelate for our class.

So,

```
templelate < typename t>
class Algebra
{
    t a, b;
public:
    Algebra() : a{0} , b {0}
    {
    }
```

```
Algebra(t a, t b)
{
    this->a = a;
    this->b = b;
}
void print()
{
    cout << a << "\t" << b << endl;
}
};
```

It is not a class, it will not create object

→ class tempelate is not executable.

→ class tempelate

template class (int) | template class (float) | tempelate class (char).

```
Algebra<int> o_int;
Algebra<float> o_float;
Algebra<char> o_char;
```

→ it will create whole class again for int of int

Stream insertion (global) function, we have to give type of tempelate otherwise it will give error.

→ Global function class ka part nai hai usay pata he nai hai is ponsi data type pass krni hai

we will define de templete wali line at the start of all global functions

How to split our code?

Class k bahir functions likhain gy.

```
class Algebra {
    int a, b;
public:
    Algebra();
    Algebra(int, int);
    void print();
};
```

How to write in cpp file

return type class name :: function.

header file Algebra.h

```
# pragma once
```

Yahan declare

Algebra.cpp

```
# include "Algebra.h"
# include "iostream"
```

Yahan complete function

15-05-2024

STL and templates are containers, sequence representing arrays that can change size

STL:- standard template library
→ so many class

vector class:-

This class is just like array but its a template

Declaration :-

vector <int> vint; sort of infinite array

Function	return
→ size	current size = capacity
→ max_size	maximum data → it based on the RAM size.

→ We can access vector just like array by subscript operator.

→ at function is ~~more~~ better than [] operator

By default 0 size hoga

```
cout << vint.size() << endl;
vint(5, -1)   → all complete array will be initialized with (-1).
```

↗ not write "5" why? don't know

```
for( int i =0; i< vint.size(); i++ )
```

Actual purpose of for each loop is for container
so for For each loop -

↙ copy contructor slow process.

```
for (int i : vint)
    cout << i << endl;
for (int& i : vint)
{
    cout << i << endl;
    i = 99;
}
```

change in vint → is sy bachay K liay const banao

```
void print (const vector<int> & v)
{
    for (const int& i : v)
        cout << i << "\t";
}
```

void print
for (const int& i:

Auto :-

Automatically determine the type of vector

```
for (const auto i : vint)
```

it can also get by reference.

```
vector<int> iv(vint);
```

copy constructor.

```
try
{
    cout << vint.at(5) << endl;
```

[5].

a[4] a[0]

When we use subscript operator this function will not throw exception wether we use try catch - So we use at function // gives value of index with exception handling -

Front return 1st element of array ~~otherwise Throw exception~~

back return last element of array ~~otherwise Throw exception~~

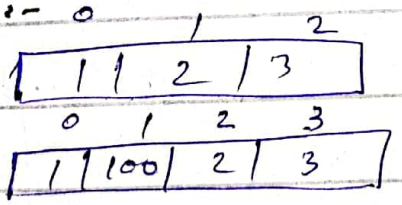
How to insert data in vector.

→ Push back:- add new element at the start. ~~end~~ of vector -

grows dynamically even we don't pass the size of array. (size increase automatically with function)

→ pop-back:- remove last element of array. (size decrease).

→ insert:-



we said insert 100 at 1 location

iterator gives the address it is a pointer We need iterator to insert value -

begin return pointer/iterator to start/begin

end " " " to end.

vector<int>::iterator it = vint.begin();

↓ member of vector class

→ while(it != vint.end())

{

cout << *it << endl;

it++;

}

int i=0;

→ for(vector<int>::iterator i = vint.begin(); i != vint.end(); i++)

cout << *i << endl;

→ vint.insert(vint.begin(), 100);

output 100 1 2 3

vint.insert(vint.begin()+2, 100)

output 1 2 100 3

logic-

yeah size+1 ki array banaye ga purana data copy karaye ga or iss index pay value insert karaye ga

a[4]

erase;

vint.erase(vint.end()-1); v(5,-1).

vint.erase(vint.begin()+1);

→ Vector can hold objects of our class too.

vector<Algebra> v(5, Algebra());

for (auto & obj : v)
    obj.print();

Yahan wahi function chalain gy jo hum ny apni class mein include krwaye hain.

void printvecto (const vector<algebra> & v).
{ for (const auto & obj : v)
    obj.print();
}

20-05-2024

How we can use one class in another class?

→ A function is using the object of a class.
It is a relationship b/w 2 class.

→ It may be one to one, one to many (array class) and many to many.

Containership :-

→ has_a

composition

It is a strong binding among the object

Aggregation -

It is a weak binding among objects.

If the life cycle of contained object depends upon the container it is strong binding.

→ container build ho contained object bhi ban jaye — container destroy honey pe contained object by destroy ho jaye - (Strong binding).

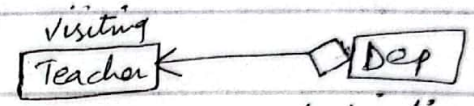
→ The life cycle of contained object does not depend on container (weak binding)

Example:- CPU and moniter (strong binding)
Room and chair/fan (weak binding).

has-a relationship is represented by diamond

filled → composition

empty → aggregation

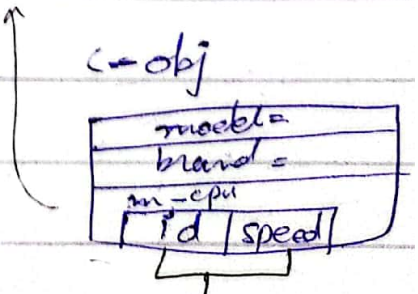


weak binding

```
class CPU{
        int id;
        float speed;
};
class Computer
{
        int model;
        string brand;
        CPU m-cpu;
}
int main()
{   computer c-obj;
```

Def. CPU
Def. computer
Des. computer
Des. CPU

Sub sy pehlay model initialize hoga phir brand phir usky baad wo CPU ka object banaygi iska CPU class k construction to call hoga or phir computer class ka construction call hoga-



These are not visible to c-obj c-obj can not acces the data members of

Destruction wiel be in reverse order.

This is a strong binding.

computer destroy hua to sath he CPU ko bhi destroy Krdy ga.

computer K data ko print Kr rahay hein-

```
void print() const
{   cout << model << "\t" << brand << endl;
    this -> m_cpu . printCPUinfo();
}
```

"its" ← this → m_cpu . printCPUinfo();
data member of our class.

Kyun K jab woh constructor dM ko initialise Krwa raha hoga to body mein any sy process slow hoga time zyada lagay ga is liay MIL mein he sary declare kr dain gy takay Subky liay aik sath he call jaye initialize ho jay.

We can not ~~use~~ initialize an object inside the constructor of another class.

It should be written in member initialization list.

→ Agar yeah nai bhi likha tha to bhi woh apni class K constructor mein jo K initialize ho raha tha.

→ How to call a parameterized constructor for CPU class.

Humain CPU ka object bhi pehlay paraya paray ga.

Computer(int _model, string _brand, int _id, float int _speed).
: model{_model}, brand{_brand}, m_cpu{_id, sp

Agar CPU ka default constructor na likha ho or computer ko // sy initialize kra hai to hum CPU K parameterized ko default values pass kr dain gy.

contained object.

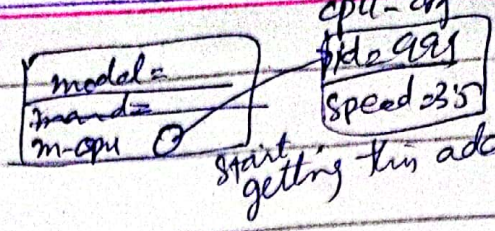
Whenever a pointer is placed in container and new object is formed on heap of (contained object). and deleted in destructor. It doesn't becomes an aggregation because the life cycle of contained object is now depends upon the life cycle of container, so it will also be a composition.

Computer(——, ——, CPU* ref_CPU)
: ——, ——, CPU(ref_CPU).

1.5
8|12
8
40
40

1.5
8|12
8
40

int main(){



CPU cpu_obj{999, 3.5f};
computer c(10, "intel" & cpu_obj);

~~Here~~

Now

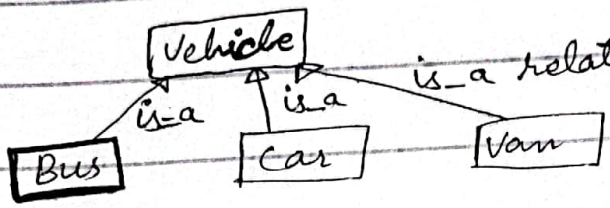
void test(CPU *t) - global function outside the class.
{
computer -
@ @
}
main
CPU obj;
computer( );
~~test~~(obj)
obj.printCpu();

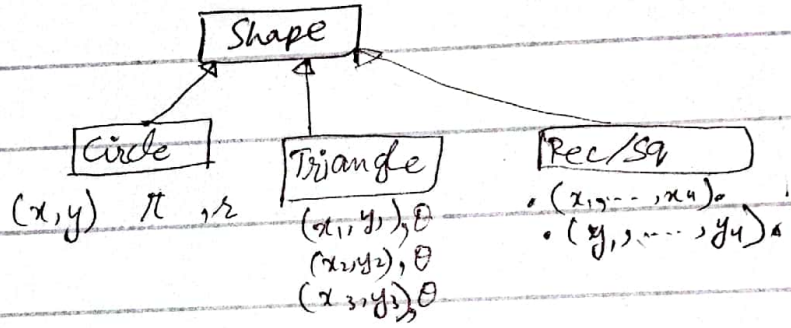
Here computer is destroyed after the ~~obj~~ test function but CPU is not destroyed it is example of aggregation - It is destroyed when main ended.

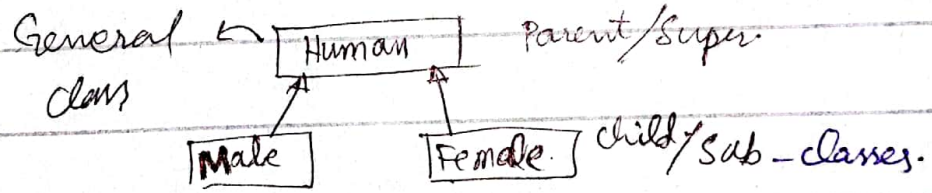
22-05-2024

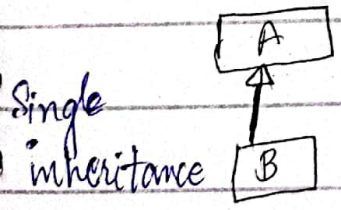
Inheritance:-

All the features of previously created class is utilized in a new class and new features are also added.

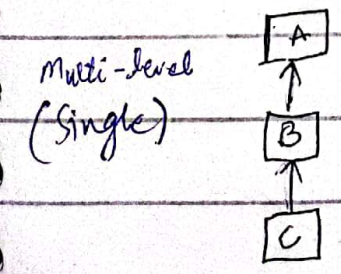




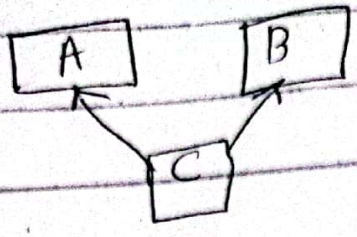




→ B is the child of A



A becomes grandparent of C
C has characteristics of A and B-



Multi inheritance

class B : A Private inheritance

class B : public A

Inherit all the feature of class A to class B publicly.

int main(){

B obj;

Big three:
→ Constructor doesn't inherit
→ destructor doesn't inherit
→ Assignment operator doesn't inherit

Def A

Def B

Def B

Def A

Size of B = 8 (data of a will be inherit in A).

2 data members → child class

da, db → parent class (int type)

Destructor calling is in reverse order.

No containership Containership.

Q2 class B have 2 data members.

number of the class

Search upward

If we don't specify any default control will be private.

Encapsulation doesn't data become member

None of the property can class will be shown to child class

Functions of A class can also be accessible to B class.

obj.A::printA(); obj.printA();

agar hum ny kaha K inheritance to public hai lekin data member ko access kny ki koshish kr rahay hein to woh private he rahay ga woh accessible nai hoga.

void printB() const. ↗ over-riding

{

A::printA(); → because da is inaccessible because it is private.

cout<<"db= "<<db<<endl;

When we write same names of same function in child and parent classes then child class check its own function

→ It becomes recursion.

→ function overriding.

→ Runs time error - stack overflow.

How to fix issue?

A :: print (); // Access the member function of A class.

Due to is_a relationship typecasting is allowed.

we can assign object of child class to parent class.

```
A oa(999);
B ob(7,8);
oa = ob;
```

no error, just warning.

slicing

Parent sirf apny features ko utha/copy

Kry ga Kyun K parents child K features Ko ~~kr~~ nai ly sakta.

A abhi bhi apny kaam kry ga ~~a~~ function bhi apny kaam he kry ga.

B(const B& obj) : db(obj.db) , A(obj).

Yahan py wahi concept hai K child class ka object parent class ko pass kr sakty hain wo sirf apny features ko copy krly ga.

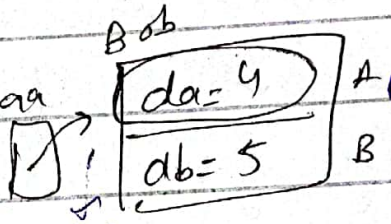
B(const B& obj) : db(obj.db) , A(obj).

29-05-2024

Virtual Functions

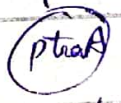
We can assign memory of child class to the pointer memory of parent class.

```
A @a;
B. ob;
C oc;
ob=ob; → slicing only get the DM of A.
```




```
A* ptra;
ptra a= &ob;
```

Here it point to the function of its own type class.



refer to its own type

To get the memory of the object dynamically we have to make function of parent class virtual.

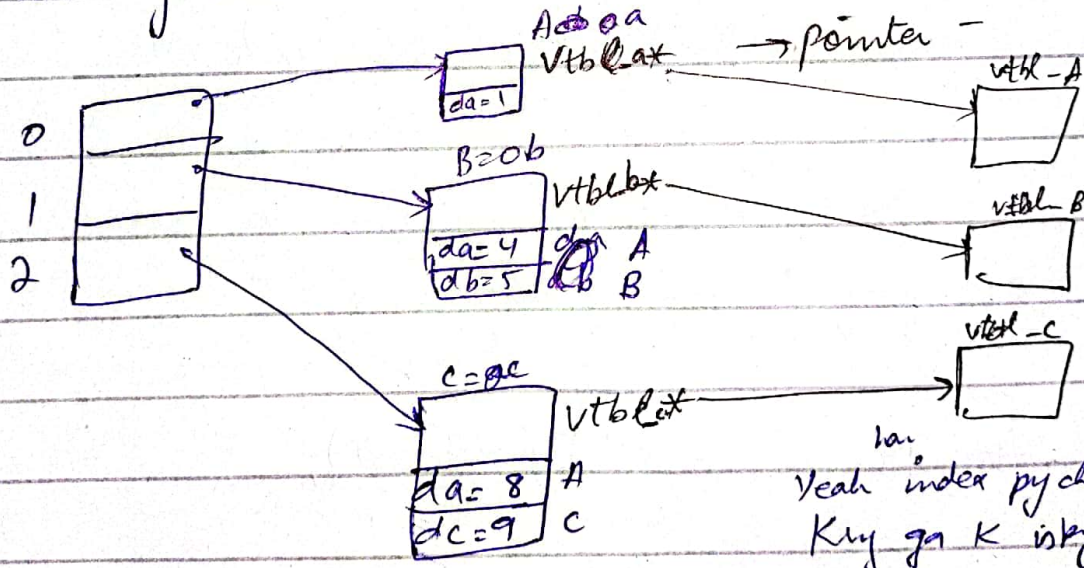
```
virtual void print()

ptra→ print(); // Now it will decide on
                  run time K yeah kis K liay
                  call karay ga.
```

Pehlay jab hum nay A K pointer ko B object ka address dia tha to woh compile time py parent class K fun print function ko call kr raha tha kyn K pointer ki type parent class thi.

Lekin jab hum nay virtual likha parent K print function K sath to woh runtime pay pointer ko jis memory ka reference dia hoga usky liay chalay ga.

A* ptra[3] = {&a, &obj, &oc};

It can place the address of object at every index




```
for (int i=0 ; i<3; i++)
{
    ptra[i] → print();
}
```

→

Yeah index py check kry ga K isky pass kis type K object ka address hai phir print function usi K call hoga kyn K parent ka print function virtual ha

const override:-

→ optional keyword.

→ Means yeah wala function base class sy a rahe hai or woh function virtual hai

void print(int a) const override.

It would be an ~~error~~ ~~that~~ because we have'nt written any parameter in virtual function so it becomes over loading.

→ If we create a virtual function - It would remain virtual in overall code.

→ override likhny sy function ya code mein koi farq nai parey ga.

A* ptr = new B(99,100);

Delete will not delete the B object because B, A sy inherit ho raha jab pointer ~~Call~~ B bana to woh base class K

New thing All function have address

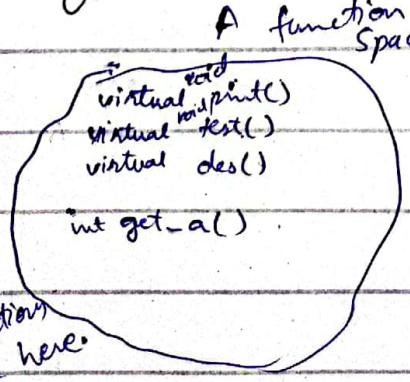
destructor ko call karay ga - Yeah wala print wali functionality ~~kr jaye ga~~. Ki tarha hai - Toh hum destructor ko virtual ~~class~~ banayen gy takay jis class ka object banaya hai uska destructor bhi call ho.

Virtual liking sy class mein aik data member add ho jata hai (agar aik baar bhi virtual likh dia).

go to (11) page. vtbl pointer

The reason is that there are two virtual functions in our child class - The size of child classes increase -

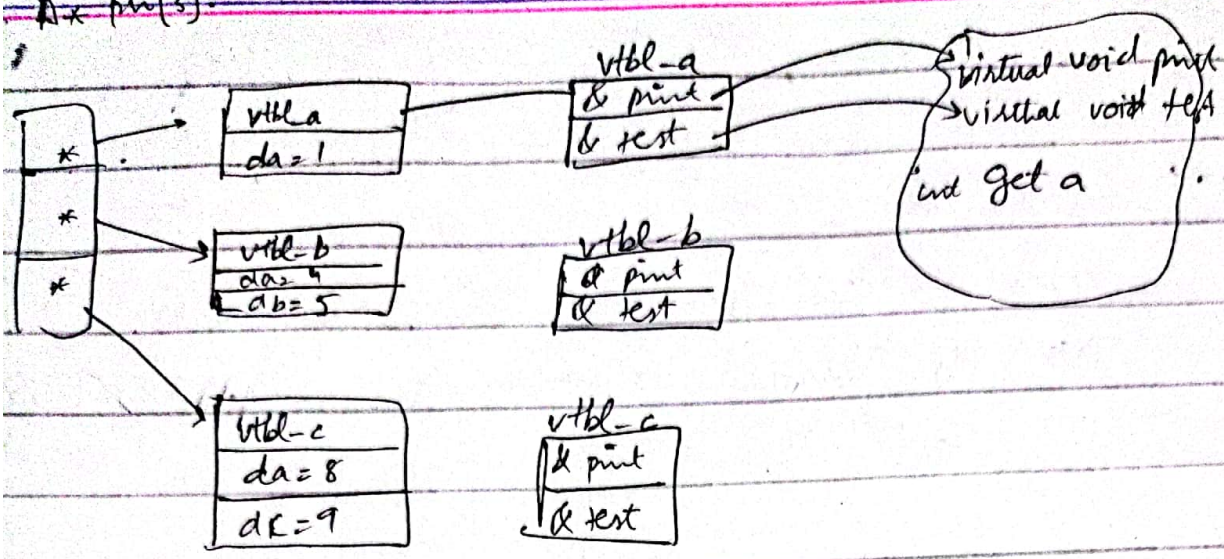
Vtbl pointer address to the virtual table of class (these are arrays).



Vtble mein virtual function K addresses hotay hein -

Vtble is not inherited again ho jata to woh A class ko access kita.

A * ptr[3].

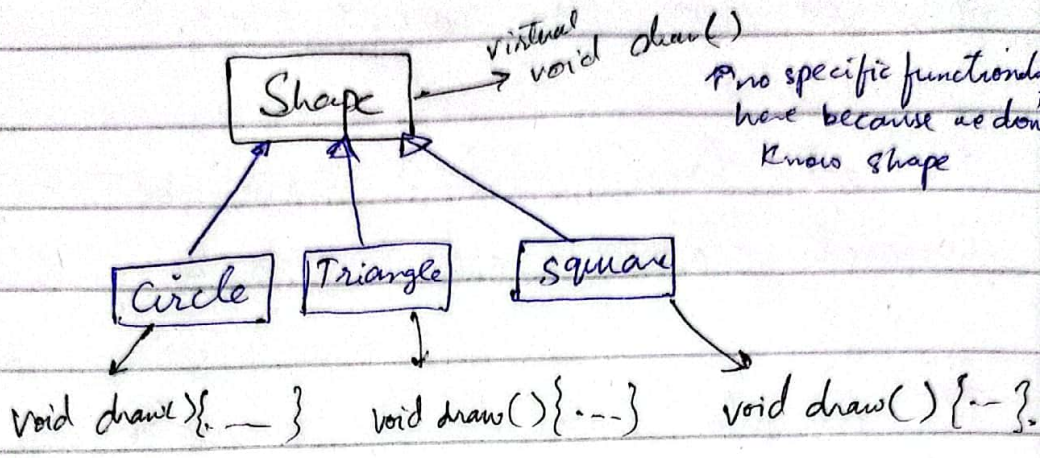


ptr[1] → test().

Steps

1. get the memory type
2. get vtble address.
3. Find the address of test function.
4. call the function of that address.

3-06-2024



When we don't know that what should be the functionality of virtual function.

virtual void draw() = 0 → pure specifier
↑ declaration

it becomes pure virtual function.

→ When it becomes pure virtual function, then child classes have to define that function

Employee class → salary function
→ extend few classes from it ↓ salary function should be there and made it pure virtual

By adding pure virtual function, it becomes abstract class → (The class where object cannot be created) only

pointers can be created and pass the reference of child class to parent class-

Abstract class:- object can't be created
Concrete class:- object can be created.

If we don't provide an overrider in child classes then they also becomes abstract class because all the characteristics of base class is inherited in child class-

```
Vector<A*> va;
B objB;
C objC;
va.push_back(&objb);
va.push_back(objC);

for(A* p : va)
    p->test;

class B: private A
```

When we do private inheritance it becomes composition. Because the characteristics of base class becomes private in child class and can't be ~~accessible~~ ~~used~~ accessed.

Multiple inheritance

↑ Public class

class C : public A : ~~public~~ no public B