Date: _____ Day: M T W T F S

Object Oriented Programming (Concepts)

Roll No: BSEF24A024

Lab Timings: 9:45 ~ 1:00

Date: _____ Day: M T W T F S

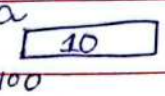
Lecture : 01

Alias:-

- Easy Syntax
- Isn't complete replacement of pointers

Syntax:-

1. int &b=a; (b is an alias to int)
2. int *&q=p;
3. int &c=b;



→ cout << b; (Har woh cheez jo a k sath ho sakti woh b k sath ho sakti)

* cout * << *b; (error coz int a=10 derefer nhi ho sakta)

* alias behaves as an const pointer. At the time of declaration alias bnana zaroori hai.

Example:-

My swap using alias.

```
void swap(int &a, int &b)
{
    int temp=a;
    a= b;
    b= temp;
}
```

```
int main()
{
    int x=10;
    int y= 11;
    swap(x,y);
}
```

Date: _____ Day: M T W T F S

Example:-

```
int x = 901;
int & yes()
{
    int * p = &x;
    return *p;
}
int main()
{
    int & z = yes();
    z = 67;
    cout << x;
    return 0;
}
```

Console:-
67

Imp:-

```
int a = 10;
int *p = &a;
int &x = *p;
```

Prototype:-

int hello (int); (is tarah bhi de saktay)

- Agar aik function kisi bhi file me na ho aur prototype dekr call ki jaye toh linker error ayega-

Date: _____ Day: M T W T F S

* extern int num; (To access global variables in other files)

Usage of Static Example:-

```
int generateToken( )
{
    Static int token = 0;
    token++;
    return token;
}
```

Note:- Local static variable tab use kro jab reset na krna ho.

Adding Header Files-

1. # is preprocessor directive. (Works before compilation)
2. 2 files bnani hain aik .cpp aur aik .h.
3. <iostream> == "iostream"
4. "MyMath.h" coz ye usi folder me hogi na k compiler walay folder me.
5. "MyMath.cpp" se global level pr do same chezein hongi toh error ayega. (Never include a cpp file)

Note:-
"iostream" pehle current folder search krega aur phr jahan compiler installed hai wahan.

Date: Day: M T W T F S

- Header files me ↑ only prototypes aur .cpp files me only definition.
- Header files me object file nhi banay ga.

header.h

```
int a=10;
```

hello.cpp=

```
extern int a;
```

(can't be accessed error will be produced)

* Header file cpp file me include krni hai-

a.h a.cpp
a.h

b.h
a.h

b.cpp
b.h
a.h

(error nhi ayega agr sirf prototypes hongi)

Defines-

Greetings

```
#define "Hello"
#define Integer int
```

(iski jagah ye ayega)

Note:- #define se sirf aik dafa he ho payega include.

Date: _____ Day: M T W T F S

→ # define My_Math_H (Header file me likhna hai)

*#ifndef My_Math_H :

_____

_____

_____

#endif

Note:- Check krega k define hai ya nhi

Syntax of Header Files (Acording to Sir):-

```
#ifndef abc.h
#def abc.h
    (Yahan sari prototypes ayein gi)
#endif
```

Works	Not Work
int a[4]={1,2,3,4}	int a[4]={1,2,3,4}
int * const &p=a;	int * &p=a;

* The actual typeof int a[4] (only a) is int *const. (Same is the case for other datatypes)

Date: Day: M T W T F S

Address Of a Function:-

cout << generateToken; → No ending braces

This will generate an "(erro)" address which will be of the function. If we include a header file of different functions then it'll produce different address possibly because of different compilation unit. (in a)

Pointer of a function:-

returntype (*functionName_ptr)() = &functionName

→ To call it we do:-

cout << functionName_ptr();

Example:-

* int (*generateToken_ptr)() = &generateToken;
* cout << generateToken_ptr();

Note:-

You can't make an alias of an alias. (Will learn the double alias concept in OOP)

Note:- int * setNoe → const int * const setNoe;

remember

Date: Day: M T W T F S

Lecture : 02

Structs :- (C-structs)

attributes.

cin >> x1 >> y1; → point type nam

cin >> x2 >> y2; → point type nam.

Declaration :-

Before main :- → user defined type (iskay objects banayin gay)

struct Point → first letter capital

{

int x;

int y;

};

int main()

{

Point a; → 8 bytes. x, y

Point b; → x, y → 8 bytes.

a.x = 13;

cin >> a.x;

b.x = 19;

cin >> b.x;

}

Dot Operator :-

.;

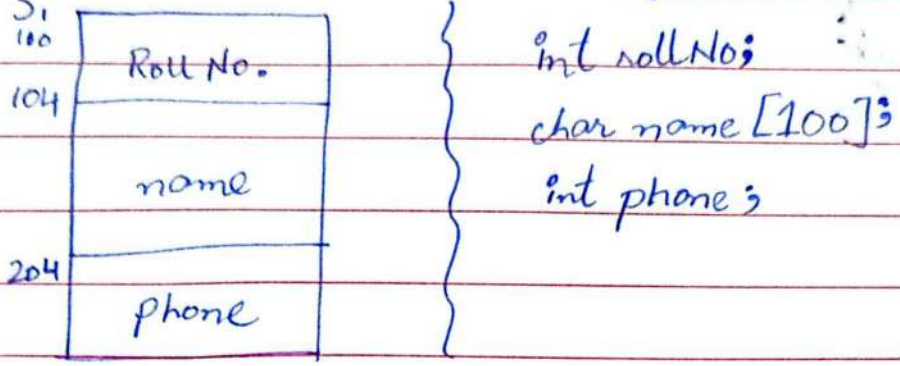
a.x

↓ must be attribute of that struct.

must be variable of a structure type

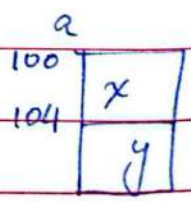
Date: _____ Day: M T W T F S

* Struct k variables ko objects kehte hain.



* If (a == b) (can't do it because they are structs)
* No relational, logical and arithematic operator can be used.
* & and = only these works with structs.

Addressing:-



cout << &a;

a = b;

must be of same type

```
a.x = b.x;
a.y = b.y;
```

ye hoga.

Date: _____ Day: [M][T][W][T][F][S]

Passing Struct To a Function:-

→ float calcDifference (Point s, Point t);
  * calcDifference (a, b);

* &a ki type (Point *) hai.

→ Point * p = &a; → (*p).x = 91;
  cout << (*p).x;

* . ki priority * se zyada hai.

* void inputPoint (Point * p);
  - inputPoint (&a);
* void inputPoint (Point &p);
  - inputPoint (a);

* Abhi struct k sath const nhi lagana.
→ Point a = {3,4}; (can be initialized like array)

* Simple arrays can be passed by value using struct.
* Structure arrays can be
#

Date: _____ Day: M T W T F S

→ Data getOlderDate(Date a, Date b);

* ⊙→ Member Selection Operator.

* (*p).x == p→x; (will only work on pointer, not alias)

Note:-

Name your predicate function (which returns a bool value) as a question Like use isLeapYear rather than checkLeapYear.

Date: Day: [M][T][W][T][F][S]

(Lecture: 03)

→ Byte Alignment (Write on Google)(Wikipedia)

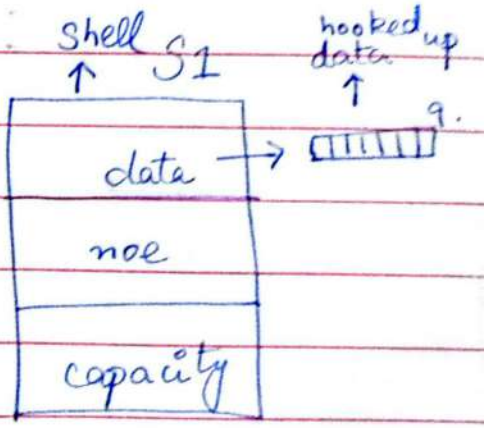
```
*void copy( void * destination, void * source, byte count)
 {
        char * d = (char *) destination;
        char * s = (char *) source.
        for (int i=0; i< byte Count; i++)
                d[i] = s[i];
 }
```

→ copy(&s, &x)
, 108

Functions:-

1. memcpy (memory copy).
2. memmove (memory move).

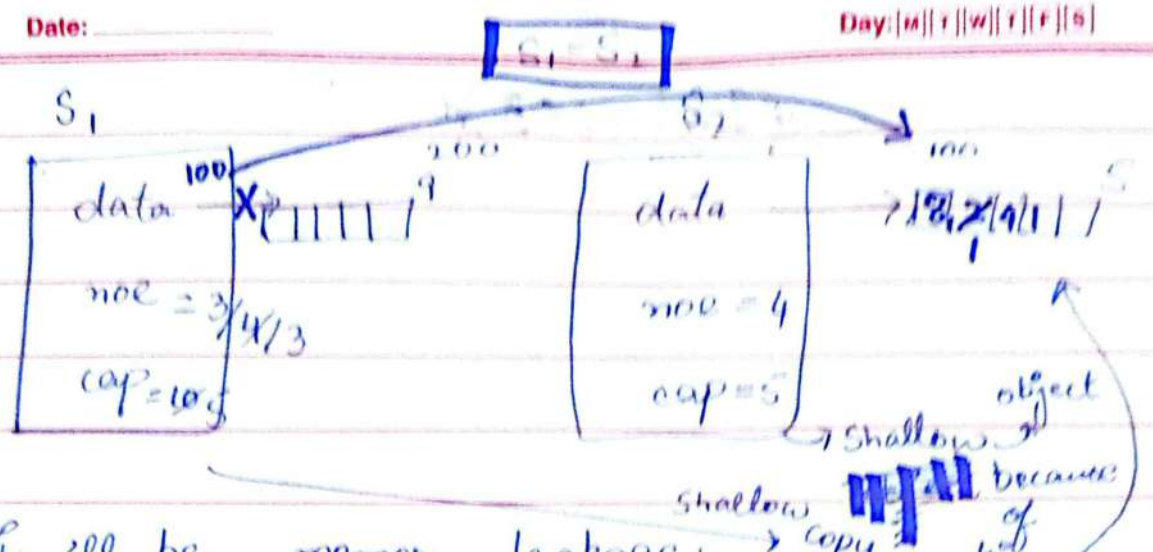
```
struct Set
 {
        int * data;
        int noe;
        int setCapacity;
 }
```



→ S1 = S2;

→ on next page.

Date: Day: |M| T |W| T |F| S



* There'll be memory leakage.
* Data will be inconsistent. (Not deep copy)
* data2 will be dangling if S1.data is deallocated.

→ It's better to use alias wherever possible.

Examples-

```
void createSet (int *& Set &x, int N);
{
    x.capacity = N;
    x.noOfElements = 0;
    x.data = new int [N]; [x.capacity]
}

    x.capacity = 0;
    x.noOfElements = 0;
    x.data = nullptr;
    if (N <= 0)
        return.
```

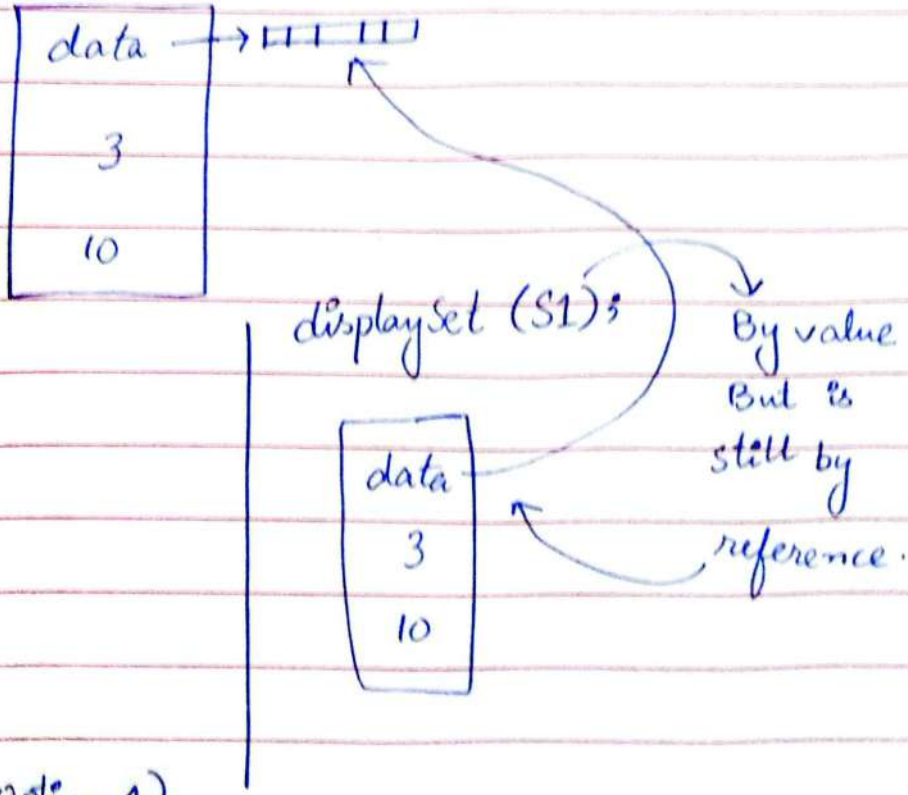
Null Objects-

S

data	0
noe	0
cap	0

Important:

Set S_3



Imp:- (option 1)

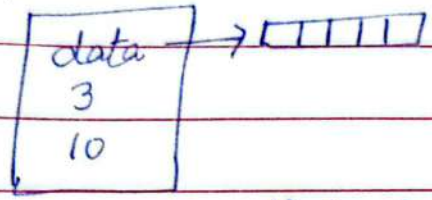
→ Accept the shallow copy fact or

→ Don't change the object attributes (strictly prohibited)

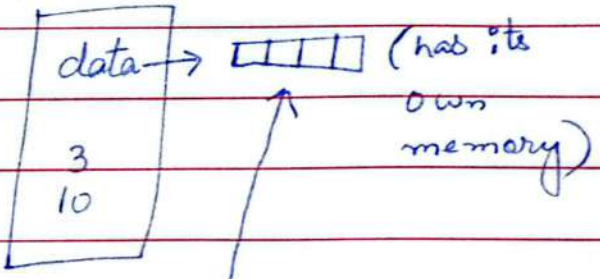
Date: _____ Day: M T W T F S

(option 2)

→ Set S_4



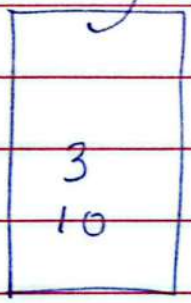
displayset(S_4)



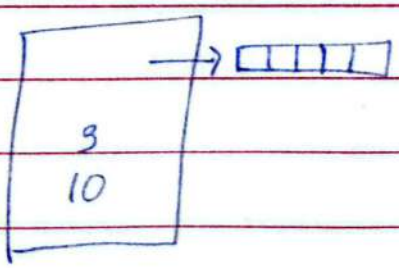

```
*Set CreateClone(Set x)
{
    Set cl;
    createSet(cl, x.cap);
    cl.Noe = x.Noe;
    for(int i=0; i<x.Noe; i++)
        cl.data[i] = x.data[i]
    return cl;

}
```

x



cl

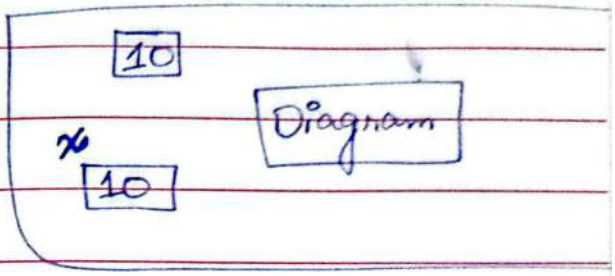


Date: _____ Day: M T W T F S

→ return jo cheez hoti uska kuch naam nhi hota;

Imp:-

```
cout << f() + 2;
int x = f();
int f()
{
    return 10;
}
```



→ createClone deep copy kr rah hai.

→ display me deallocate krna zaroori hai.

* By default, har jagah pehla method use krein gay.

Note:-

To avoid unnecessary padding list your struct attributes in descending order.

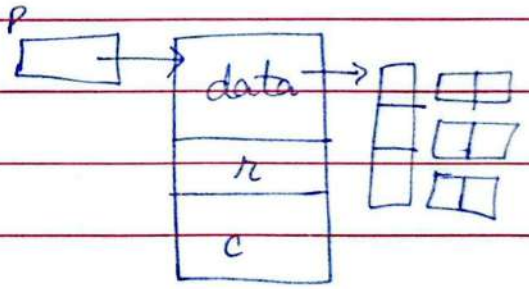
Date: _____ Day: M T W T F S

(Lecture: 04)

→ exit(0) → The program will be terminated.

```
float divide (int n, int d)
{
    if (d!=0)
        return n/d;
    cout << "error";
    exit(0);
}
```

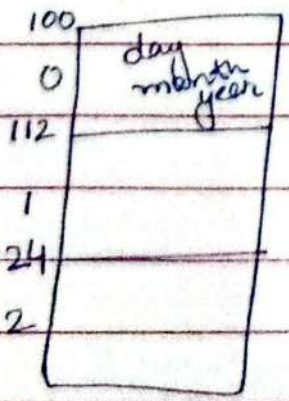
* Date * p = new Date;
* delete p;



- createMatrix (*p, 3, 5);

Struct Arrays-

Date d[3];




```
d[1].month = 11;
inputDate(&d[1]);
```

Date: _____ Day: M T W T F S

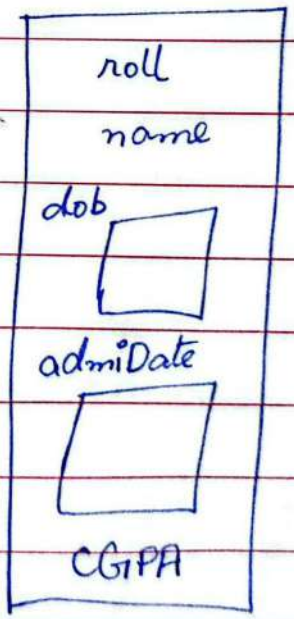
* m[0] = add (m[1], m[2]);

→ Primitive types are language provided and base. (Aren't made by other datatypes)
- int
- float
- string
- char and others etc.

→ struct Student
{
int roll;
char name[30];
Date dob;
Data admiDate;
float CGPA;

};

∴ Student S ∴



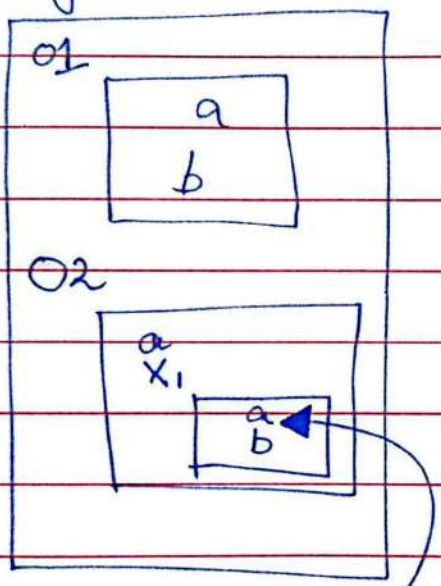
* S.dob.month = 11;
Solved L→R

* inputDate(S.dob);

Date: _____ Day: M T W T F S

→ struct Watch
{
    Date d;
    Time t;
};

→ Z obj;



* obj.02.X1.a;

~~dayInaMonth[12] = {~~
~~Month~~
~~int~~
~~daysInaMonth[Mar];~~

Date: _____ Day: M T W T F S

→ int addRecord(———)
{
const int SUCCESS = 1;
const int DUPLICATE = 2;
const int FULL = 0;
} → make them global.

Enum :-

→ outside main();

enum GameStatus { WIN, IN-PROGRESS, DRAW };
WIN = 0, IN-PROGRESS = 1, DRAW = 2 - - - -

inki type GameStatus hai.

* GameStatus :: DRAW.

→ Gamestatus itself is a type and all inside curly braces are identifiers of const int.

Date: Day: M T W T F S

(Lecture: 05)

→ Jab main khtm hota ya exit(0); hota tou heap ki memory bhi khtm hojati hai. (End of Program).

Object:-

→ has an identity.

Collection of attributes with interrelated functionality.

* Paradime == "طریقہ کار"

→ Abstraction
- Functionality.
- Data.

* Advantages of Functions (Have to Google it).

* _addRational (int n_1, int d_1, int n_2, int d_2, _);

* Rational add (Rational a, Rational b);

→ C++ Structs are totally different.

* object states = Values in the attributes of an object.

State of an object

Date: Day: M T W T F S

* Human → Inheritance (Qualities inherit krna)
  - Teacher
  - Student → Student inherit qualities from Huma.
    - Graduate
    - Undergraduate

1° Aggregation → class is an aggregation of chairs etc.

2° Composition → Class is a composition of chairs, dice etc.

3° Association → Class is an association of chairs, dice etc.

→ All three are different in programming.

* Polymorphism:- Different behaviour on same instruction

→ Example:-

```
f(2,3);
f(3);
f(2,3,1);
```

→ Executes according to number of Arguments. (Polymorphic Behaviour)

Date: _____ Day: M T W T F S

→ Building Blocks of OOPs-

1. Encapsulation.
2. Aggregation.
3. Composition.
4. Association.
5. Inheritance. → Will do it after mids.
6. Polymorphism

* Noun aur us se related functionalities sath honi chahiye.

→ Information Hiding (Outer World ko kya dikhana hai aur kya nhi dikhana)

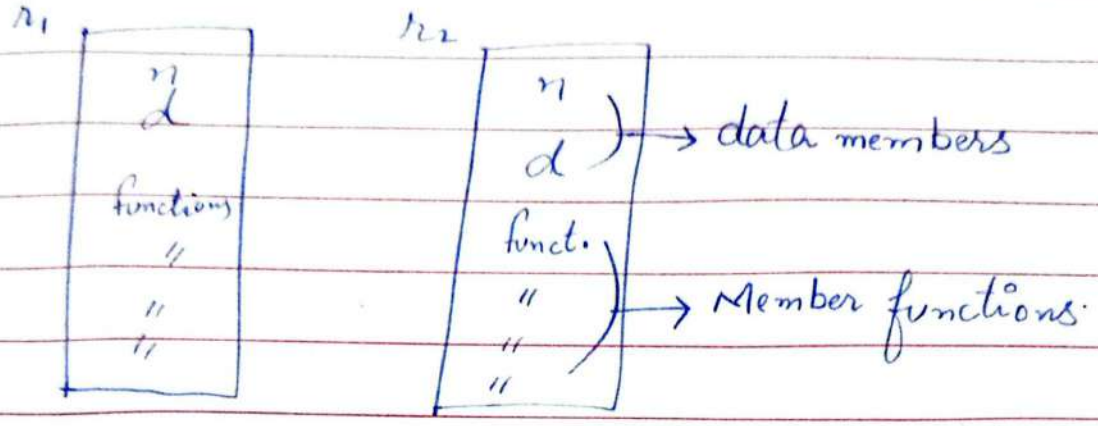
o Struct Rational { → In C++ Structs.

```
int n;
int d;
void f1();
//
//
//
};
```

multiple functions

Date: Day: M T W T F S

* Rational r1, r2;



* printRato ki copy har object me pari hogi.
(Memory waste hogi because mutliple copies, instead of only one copy).

→ r1.printRational(r1);

→ void printRational()

```
{
    cout << n << d;
}
```

→ Jis object ka printRational call hoga usi k liye reduce chle ga.

→ r1.add(r2);

→ object stale ko Invalid nhi honay dega.

Date: Day: M T W T F S

```
o Struct Rational {
    private:
        int n;
        int d;
    public:
        1f();
        2f(n);
};
```

* always make data private for Data Abstraction.

o The functions names that access private data.

- Getter. / - Accessor / - Wrapper.
- Setter / - Mutator

↓

Generic Name

> void getDay();
> void setDay();

→ Date me day, month etc k liye alag alag getter, setter sahi nhi.

(32, 4, 2024)

not set ↓ set ↓ set

→ 1° Struct objects is by default public.
2° class objects is by default private.

Date: Day: [M][T][W][T][F][S]

* changing private attributes doesn't affect user, it affects the functions of members. (Data Abstraction).

→ ADT == Abstract Data Type.

→ Setter tab invalid kre jab object valid state me milay.

→ Abhi assume kr rahay hain k object valid state me mila hai.

In C++.

```
class Date{
    //  //
    //  //
};
```

* classes me cin nhi krna na he cout.

* int Rational :: input (a, b);
  - Now its a member function of Rational class.
  (class name)

* char * and const char * me length ka concept nhi hota.

Date: Day: M T W T F S

(Lecture: 06)

→ function ka interface uski prototype hai.

- As long as you don't change the interfale, the user isn't affected.
- The public part is the interface of a class.
- Data private isliye rakhein take liberty rahe k representation change kr sakein (Data abstraction)
- Har cheez access k liye getter/setter use krein (follow nhi krna but dimaag me bitha lein) (is se agar private variables change krein gay toh sirf mutator/accessor change hongay bus)

• Constructor:-

— A fuction that executes automatically. (for every class object)

- In Public:

```
Time();   ← exact name of class
```

no return type.

- In cpp:-

```
Time::Time() {
    cout << "Hello";
}
```

hours = 0;
minutes = 0;
seconds = 0;

Date: Day: M T W T F S

* hours = minutes = seconds = 0;

* constructors can't be called by '.' operator.

→ Constructor < Initialization.
Resource Allocation.

• During Declaration.
- Time t1, t2(4,5,10);

** Default Constructor :-
A constructor which have no parameters.

➡ Object invalid state me hargiz hand over nhi krna.

* Constructors k andar setter ko call nhi krna because setters ye assume krke likhe gate k default set hota hai.

- First set to default than do anything further in specifications of user.

{
hours = minutes = hours = 0;
setTime(h, m, s);
}

Date: _____ Day: M T W T F S

→ Time (int h=0, int m=0, int s=0);

* Har woh constructor jo without argument execute hojaye default constructor hai.

* Constructor public me isliye rakhte hain coz object automatically call wohi krega jo public hoga.

→ Acc. to language, har class k liye aik constructor lazmi hoga.

→ Agar constructor nhi likhte to language/Compiler khud aik empty default constructor daal deta hai. (In 2010)(Is a concept)

Uniform Initialization:-

Jis saal k andar language standarize ki jati hai uska naam etc [C++10, C++13, C++14] rakh dia jata hai.

* Structs C++ me isliye aye take agr Cme code likha ho toh woh effect na ho. (Backward Compatible)

Date: Day: M T W T F S

- Time t1(1,3,5);
- Time t2();
  - is a prototype not an object that is made.

- For this issue :-

1. Time t1{};
2. Time t2{2,3,50};] preferable syntax;
3. int*p= new int[5] {1,2,3,4,5};

Direct Initialization :-

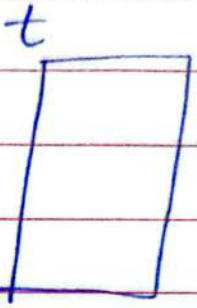
```
Class Time {
  private:
    int hour=5;
    int min=8;
    int second=48;
};
```

* Agar constructor bhi likha hai sath toh first direct initialization then default constructor.
* Nik bhi constructor toh khod se constructor nhi ayega.

Date: _____ Day: M T W T F S

* Aesi class ka bhi object ban jata hai jiskay na members or na member function hons

- TicTacToe t;



Deligate Constructor :- (only for constructors)

- Aik se zyada constructors execute ho saktay hain:

```
o void initialize();
  {
     10K;
  }
     A(int)
     {
        int.
     }

     A()
     {
        int.
     }
```

Date: Day: M T W T F S

- But Now; (concept of Delegation)

then ye ↖ A(int) : A{} → First ye hoga A()

{

}

{

10K;

}

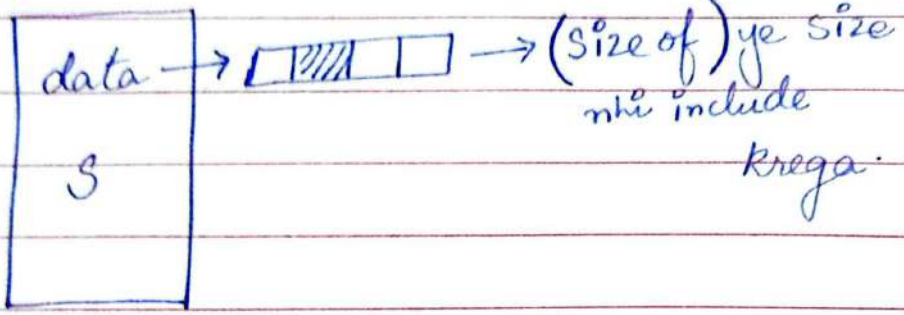
- Time (int h) : Time {h}; → ye bhi ho sakta hai
  - have to write it in cpp file.

Date: Day: M T W T F S

(Lecture: 07)

- Array ADT.

a



- Array ADT se zero size ki array bhi bna saktay hain.

(Array ADT)

- Constructor:-

```
capacity = 0;
data = nullptr;
if(cap<=0)
    return;
capacity = cap;
data = new int[capacity];
```

Destructor:-

- Resource De-allocation.

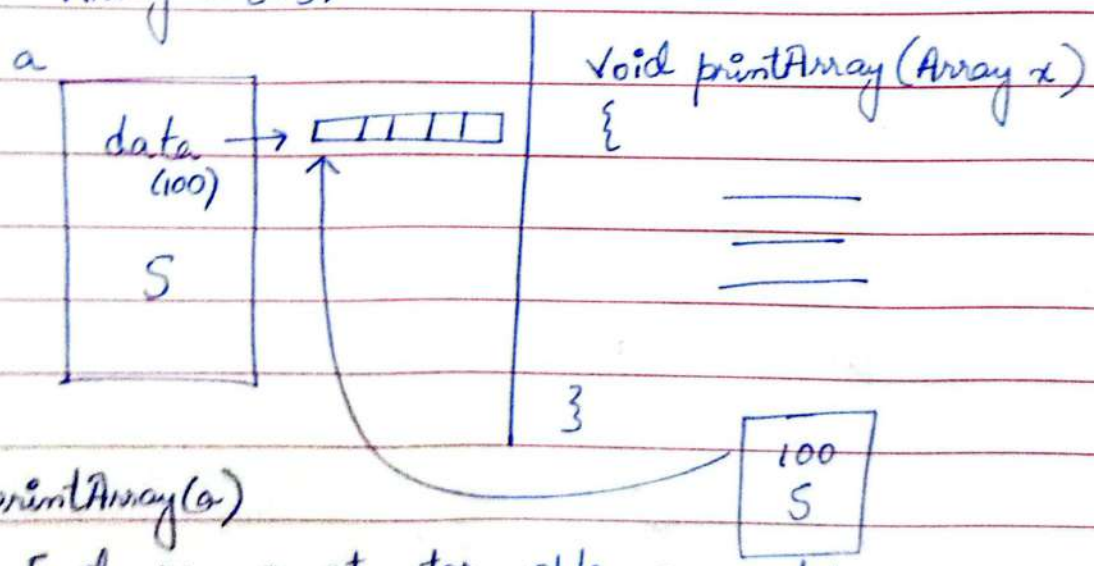
Syntax:- ~Array();

Date: Day: M T W T F S

- Called Automatically when lifetime of object is over.
- Can't give parameter to it. (Parameter less)
- har class k liye aik destructor likha hota hai automatically but its empty.
- Variables jis order me bantay hain uskay reverse order me khtm hotay hain. (Stack Concept)

→ Issue:-

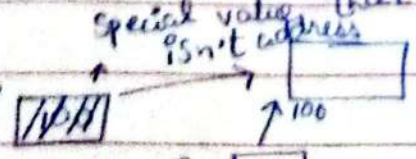
Array a{5};



→ printArray(a)

→ End pr constructor chle ga toh original memory me issue ayega.

- int * q;
- int * q, p = new int;
- q = p
- delete p;



q [100] → yahan abhi bhi dangling hai.

Date: Day: M T W T F S

→ Abhi hum objects by value nhi pass krein gay. (by reference kr saktay hain)

* Destructors can be called mullipple times.
  - Not for constructors as there are other functions that can be used for initialization.

-Destructor:-

a.~Array();

* Jab bhi member function k andar destructor likhein gay toh this→~Array() likhna hai. (only for the calling function)

* i<newCapacity && i<capacity.
  this→~Array();
  newCapacity = capacity;
  a.array = R; new array.

* String se related global function bnane ki zaroorat nhi.

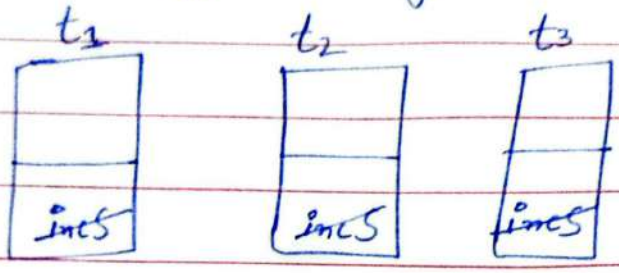
Date: Day: M T W T F S

Lecture # 08

→ t1.incSec(10);

→ incSec t1 per perform hua hai.

→ Memory k andar object sirf attributes per jaate hotay hain.



→ incSec(10) ko ↓ object ka address pass hota hai. calling

→ At Backend: incSec(&t1, 10)

it is a global function but at programmer level we can't treat it as or use it as global function.

* void incSec(Time * this, int inc)
{
this→second = this→second + inc
}

→ At backend this is happening.

* t1.getHour;
getHour(&t1); → khudse aesay nhi kr saktay.

Date: Day: M T W T F S

→ Hum ye Backend wala object ko use kr saktay hain pointer se. Iskay liye this → pointer hai.

→ this → pointer k pass calling object ka address hota hai.

→ There is only single copy of functions.

→ cout << this << "\n";
address of calling function.

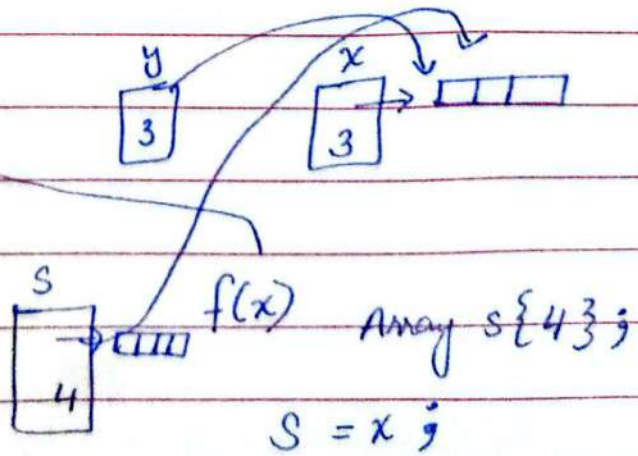
→ Constructors aur Destructors ki bhi aik he copy hoti hai.

→ this → hours = hours; (aesa nhi krna hai).

→ (*this). getlenght (); (functions k andar aesay use kr saktay hain).

Issue:-

```
f(Array y)
{
}
```



Date: _____ Day: M T W T F S

* Abhi sirf ai the type of declaration by value krein gay.

* Array t = x; → abhi itna kaam hoga.
  Array t{x};

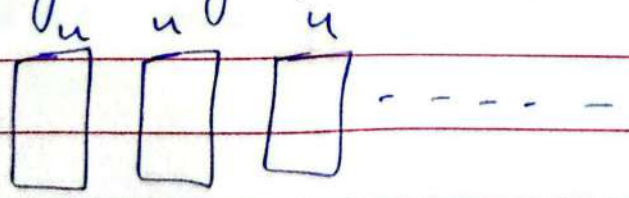
This is copy constructor.

↳ Array (Array & ref) → { data = ref.data; capacity = ref.capacity; }
↳ Array (Array * r)
↳ Array (Array u)

↳ memcpy (this, &ref, sizeof (Array)). Time

* Bitwise, shallow copy, memcpy sab aik he hain.

* Array (Array u)



* Har class me aik copy constructor confirm hota hai. (Isme memcpy chalta hai)

Date: Day: M T W T F S

```
Array (Array & ref)
{
if(! ref.data)
    return;
data = ref.data;
capacity = ref.capacity;

}

String (String & ref) : string()   // const → ye aglay me btayein gay sir.
{
if(ref.IsEmpty())                  If(ref.data == nullptr)
    return;                            return;
size = ref.size
data = new char size;
for(int i=0; i<size; i++)
    data[i] = ref.data[i];
}
find(String substr, int start)
{ int i=0;
  while(data[i] != '\0')
  { int j=0, k=i;
    while(substr.data[j] != '\0' && data[k] != '\0'
          && substr.data[j] == data[k])
    { j++;
      k++;
    }
  }
}
```

Date: _____ Day: M T W T F S

```
if (subStr.data[j] == '0')
        return i;
    i++;
    }
    return -1;
}
```

- If (isE && sub.isE)
  return 0;
- If (// // //)
  return -1;

```
while(s.find(t) != -1)
  {


  }
```

Date: _____ Day: M T W T F S

(Lecture : 09)

→ const < objects.
Data Members.

* class Test
{
int a;
const int b; = 9 i) Direct Initialization.
const char c;
int d;
}

→ Direct Initialization is limited.

* Member Initialization list.

↘ public:
Test(): b(10), c('A').
{}

o Test y(12); → error that what is in b and c?

→ Test(int x): b(x), c(x) → once this part is activated the constness is effective. activated.
{ b=10; (error)
}

Date: _____ Day: M T W T F S

```
main()
{
    int x;
    cin>>x;
    Test t;
    Test y(x); → possible (No error).
}
```

* MIL me kisi attribute ko initialize kr saktay hai.
* MIL me use sequence se initialize hongay jis sequence se class me banay hain.

```
private:
    const char bGroup[4];

→ Test(const char * bg)
  {
      bGroup{ bg[0], bg[1], bg[2], bg[3]}
```

→ if bg = 'B+\0' then there will be error as bg[3] will be error and there will be runtime error.

```
  }
```

Date: _____ Day: M T W T F S

Test (int x) : b{x}, c{x}, d{30}, a{d}: Test()

{

}

- konsay pehle chlein gay ye to pata.
- But MIL hogi ya delegate constructor.

- Const Test t;
- Const Time x;

→ inme const attributes wese he tarqay se hongay.

→ For a const object, non-const ~~(objects)~~ attributes effectiveness start when the constructor's execution is complete.

* const Test t; → koi function use nhi kr payein gay.

→ className * const ~~this~~ this;

→ x ki type hai const time *;

→ const member functions are those that don't have the power to change member functions.

Date: Day: M T W T F S

→ calling object k upar

- int getHour() const;) bas restriction hai.

In it ye the type of this is const classname * const this.

→ change it in definition and prototype.

- It would be a syntax error if constructor / destructor is made constant.

→ Main conflict (this) ka hai.

- logically const int * data hoga.

data = — x.

data[1] = 'A'.

→ void f(...) variable number of arguments.

{

cout << f;

}

- Function Call Overhead? → is se bachne k liye macros bnatay thay.

→ #define max(a,b) (a>b? a:b)

- cout << max(4,5);

not function call it is exchanget with

Date: _____ Day: M T W T F S

y=8; x=5.

cout << max(++x, y++);

cout << y;

↳ it'll be 10.

→ Inline functions :-

```
inline int getMax(int a, int b)
{
    return a>b?a:b;
}
```

it'll be pasted where this is written.

- Only those inline functions will be written with def. in which no control structures are used.
- If functions is there in the class they're inline by default.
- <Stdarg.h>

Date: _____ Day: M T W T F S

```
void f(int size, ...)
{
    va_list list;
    va_start(list, 5);
    int val = va_arg(list, int);
    cout val;
} va_end(list);
```

→ dubara likhein gay toh agli value ajaye gi.

- f(345, 250, 292);

→ Agar 5 ki loop chlayein gay toh agli 2 values me garbage print hogi (runtime error nhi ayega)

- void printf(const char *, ...);

- f(5, 1,2,3,4,5); i/p hr garbage print nhi hongay.

→ Array(int cap, ... =0);

↓ jab const array aye.

Date: Day: M T W T F S

- int & x = 456; (error)
- const char *p = "dsj"; (No error)
- const int & x = 456; (No error)
  ↳ ~~this~~ check krna k kia hoga agr Copy Cont const na ho.

- Temporary value const _ & _ se bind ho sakti hai.

→ String f();
{
String x {"inf"};
return x;
}
int main()
{
String y = f();
}

* (wrote const in para. list of copy constructor)

→ RVO (Return Value Optimization)
(Sirf x ka des. chla aur na x (temp) aur na y ka copy constructor chla)

* (Copy const.) hoga tab agay ka masla hai k chlana hai ya nhi;
* const temporary k liye nhi chle ga agar const likha hai toh.
  ↓ nhi
* Move constructor resources ko move krega.

Date: _____ Day: M T W T F S

- String a{"hello"}
  - const String & b (will receive it)
  - String && b (will not receive it)

- String {"hello"} → is a temporary object, is made on ~~Heap~~ stack.

→ Temporary ka alias nhi banta hai.

- const String & s = String {"Hello"};
  s.find("Hello");
  ↳ error nhi ayega.

→ S = "hello";
  ↳ const char * wala constructor chal jaye ga.

Date: Day: M T W T F S

(Lecture # 10)

Static:-

* Class Level Information: behavior, Data
* Instance Level Information: object
  - reality of something

* C++ is an hybrid language.
  - Can make global function.
  - Also can do object orientation.

* class MyMath {

* Ismi sirf CLI hai ILI nhi.

public:
static int findPower(int, int);
static float findSquare(int); → without creating an object execute kia ja sakta hai static se.
};

→ A class can be made which has no data member. (There objects' size is 1 byte)

→ MyMath m;
m.findPower(2,3);
m.findPower(2,3); } No difference as there are no data member functions.

Date: Day: M T W T F S

→ static isliye use kia take user disturb na ho.

→ MyMath :: findPower(3,5); → object bna kr bhi kr saktay hain but this syntax is more appreciated

→ Jis class me koi data member nhi hoga tou uska this pointer ~~bhi nhi~~ hoga jo 1byte ki taraf ki taraf point kr rah hoga.

* class Student
{
private:
int roll;
float CGPA; ↑ class level Data.
static const char _collegeName[30] = "PUCIT"
}; ↘ ab ye object ka hissa nhi hoga (is ki single copy hogi) (Class Level Information)

→ Student (): _collegeName("PUCIT") → allowed.
{
↓
static ko yahan pr initialize nhi kr saktay.
}

→ static const char * getCollegeName()
{
return collegeName; ↓ Class Level Ber information.
}

Date: _____ Day: M T W T F S

→ static variables k wrapper functions bhi static banain gay.

→ static data members of cpp me globally declare krein gay.

↳ const char Student::_collegeName[30] = "PUCIT"

↓

jab direct initialization ka concept nhi that tab ye ~~kiya~~ krtay thay.

• Static data Member ka naam underscore (_) se start kreingay.

↳ _COLLEGE_NAME → static constant variable.

• ~~Student~~ static const double _INTEREST_RATE = 0.025

```
class Car {  static int autoNumber;
    int chasisNumber;
public:
    Car()
    {
        chasisNumber = autoNum;
        autoNum++;
    }
};
```

At Global Level;
int car::autoNum = 1
↓
Class Level Information.

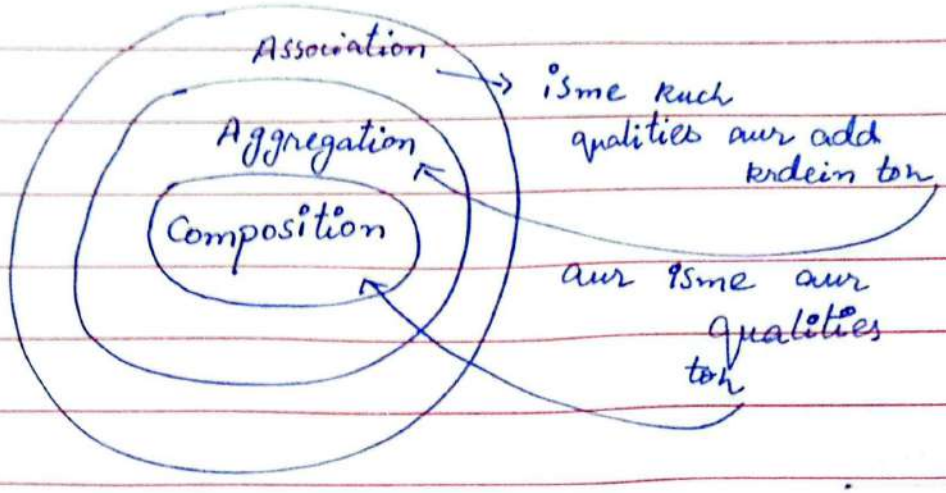
* char arrays nhi bnani ab. String ki class use kreingay.

Date: Day: M T W T F S

(Lecture no: 11)

- Composition / Aggregation / Association / Dependency
  ↓
  objects k darmayan dependency ka relation bhi hota hai.

→ Diagram :-



→ Composition is the core.

* (Composition + Aggregation) == Whole part relationship

→ Lab me koi na ho tab bhi lab hai aur agar teacher students hon toh tab bhi lab → This isn't whole part relationship - It is simple association.

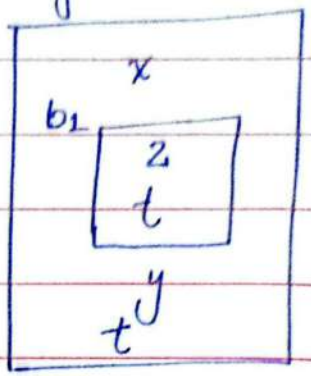
Date: _____ Day: M T W T F S

→ host class

- class A
{ private:
int x;
B b1; → object is aggregated
int y
int t
};

- class B → guest class
{ private:
int z;
int t;
};

A obj



* Kiska constructor pehle chle ga? (A ka ya B ka)
Ans: Guest class ka constructor pehle chle ga. (B ka pehle chle ga)

Reason:-

Agar host class ka pehle chlega toh B uninitialized halat me use hoga.

→ Destructor pehle host class ka chlega phir uskay baad composed objects ka chlega. (object uninitialized halat me nhi dena).

Constructor of A:-

```
A()
{
  b1.___ → ye nhi kr saktay as woh B class ka private part hai
}
```

Date: _____ Day: M T W T F S

```
class Address
{
    int h.NO;
    String address;
    String city;
    String country;
public:
    Address();
    Address(int h, String a, String city, String coun.)
        address(a), address(city), country(coun.),
        h.NO(h)
    {
    - - - - -
    }
```

C O — is order se chlein gay

Note: iskay destructor ki zaroorat he nhi (kuch deallocate krna he nhi) (No Memory allocation in this class)

↓ inka copy constructor chle ga

- Address(const Address &r): hNo(r.hno), ads(r.ads) - - - -
  ↓ default copy constructor aes ay likha hota hai.

```
{
}
```

- Agar MIL na likhein toh phir Address k guest objects k default const. chlein gay.

- Aur agar default na hua us class ka toh syntax error ayega.

→ Address b{a};
↓
iska copy chle ga.

- Address (const Address & r)
{
ads = r.ads → is case me shallow copy hojaye gi.
}

- Char array jab bhi chahiye toh String ka object use krna hai.

→ String getCity()
{
return city;
}

```
main()
{
  a.getCity().display();
}
```

L→R solve hongay.

- s.concatenate(t).concatenate(t).concatenate(t).display();
↓ cascading call.

Date: _____ Day: M T W T F S

→ void setCity (String a)
{
ads.~String();
ads.concatEqual(a);
}

∘ MIL sirf constructors k [redacted] samne aa sakti hai.

→ class Employee
{ float Salary;
int empId;
Address hAddress;
Address offAddress;
}

∴ See the [redacted] execution of constructors & Destructors thoroughly.

→ Agar death relation hai whole part relationship toh composition hai.
→ Otherwise it'll be aggregation.

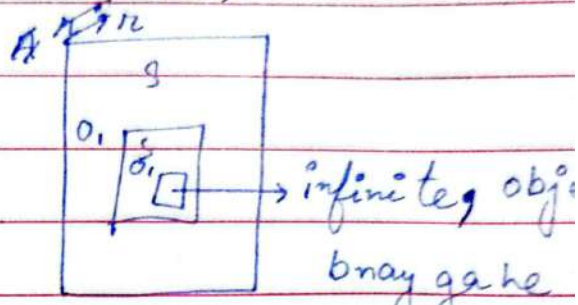
* jo humne is lecture me kia toh woh composition he hai.

* Composition == Strong Aggregation.
Aggregation == Weak Aggregation.

Date: _____ Day: M T W T F

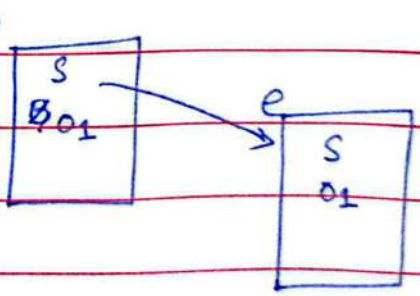
- struct A
{
int s;
A o1;
int y
}

→ aesay nhi ho sakta.
A h;



→ infinite y obj bnay ga he nhi.

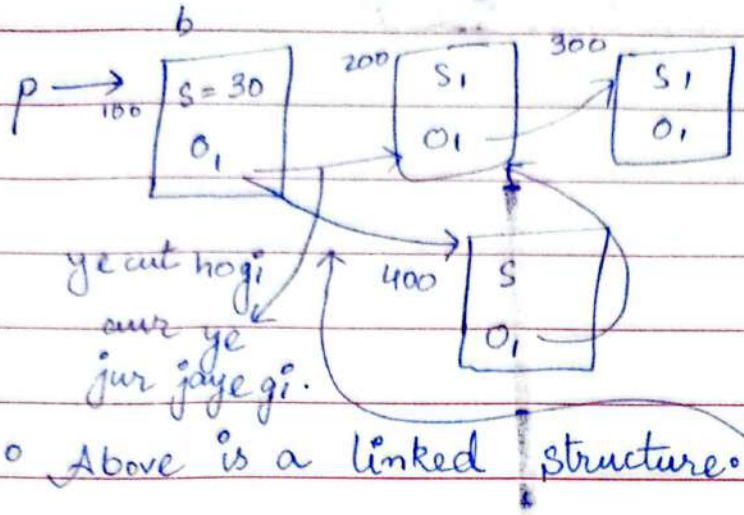
- struct B → aesi class ko self referential class kehte hain.
{ int s;
B * o1; → aesay ho sakta hai.
}



→ b.o1 = &e;
e.o1 = &d;
b.o1 → s → b se e ka s access hojayega.

- A *p = &b;
while(p) → agar guarantee ho k last pr nullptr para hai.
{
cout << p → s;
p = p → o1;
}

Date: _____ Day: M T W T F S



- Above is a linked structure.

→ b.o1 = b1.o1 → o1 → is se

→ is structure k andar resizing bohot he efficient hai.

→ Array ki jesi qualities kisi aur data structure me nhi hai.

Date: Day: M T W T F S

* Design patterns → agley semester me parhaein gay. Interview critical hai

"(Lecture no: 12.)"

```
class Single          → global level pr iskp nullptr se
{ static Single *ptr;   initialize krein gay.
public:    put in private.
   Single()
   {
   }
        Single *
   static createObject();
             ↓
         factory method.
}

static Single * createObject
{
   if(!ptr)
      ptr = new Single();
} return ptr;
```

→ String *s = new String {"def"};

- delete s; → is se lifetime he khtm hojaye ga.

data
size

→ [| | |]

• But s → ~String()
  is se memory deallocate hogi bas.

• String {"Hello"} → stack pr banay ga aur nameless object hai ye.

```
static void freeObject()
{  if(ptr)
      delete ptr;
   ptr = nullptr;
}
```

→ Agar Single pattern wali logic imply krein gay toh sarey constructors + Dest. private aur ye functions provide kreingay.

Date: Day: M T W T F S

* Single d(*s);
  ↓
  is se default copy constructor chle ga aur aik se zyada objects ban jayein gay.

Array of Classes:-

- Time t[5];
  t[0].setTime(3,4,56);
  Time t[5] = { Time(1,2,3), Time(4,9,12), Time(2,3,9), Time(12,12,12), Time(1,2,5)}
  ↓
  saray initialize krein gay.

- String s[3] = {String{}, "Hello", 'c'};
  ↓
  aesay bhi hojaye ga.

↳ s[0].input();

↳ 2D Arrays:-

Array a[3];
a[0].reSize(4);
a[1].reSize(11);
a[2].reSize(8);
a[0].getSet(1)=90;

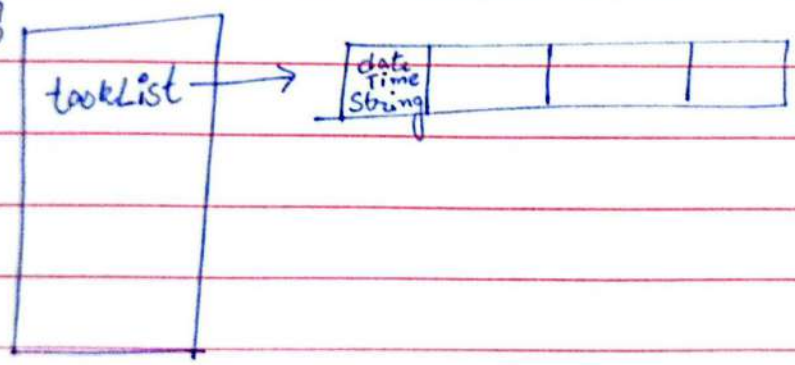
→ jacked array.

Date: Day: [M][T][W][T][F][

↳ for (int i=0; i<3; i++)
{
for (int j=0; j< a[i].getCapacity(); j++)
{
cin >> a[i].getSet(j);
}
}

↳ String *p = new String[3] { "Hello", 'P', 'C' }

→ S



→ addTask(const Task &t)
{
taskList[noOfTasks].updateDate(t.getDate());
}

→ How to see system Time and date.
can google it. (ctime will be used).

Date: _____ Day: M T W T F S

↳ Add Task
↳ Display Task
↳ Display Todays Task

→ These things in Scheduler App class.

Date: _____ Day: M T W T F S

Lecture no: 13

→ Paradime { Procedural ◆ , Object Oriented ◆ }

→ No Memory Leakage in other languages.

→ Integer array ki jagah Array datatype use krni hai.

- Application Programming Interface (API)

→ iskay sath interact krna muskil hai.

→ MFC ki layer

API

↓

iski coding boht he tough hai.

↓

Microsoft Foundation Classes. (Ye easy hai)

* NamedSet → Wrapper class.

```
{
    String name;
    Set s;
public:
};
```

```
NametSet (int cap=0) : s(cap), name('A')
{ }
```

Date: Day: M T W T F S

→ make private constructor.

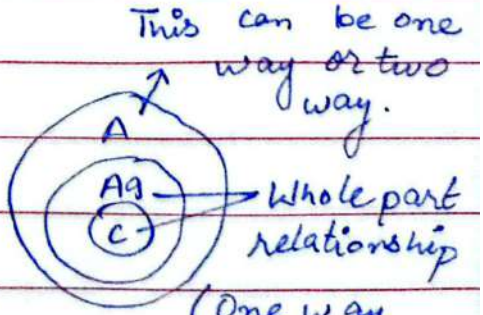
- NamedSet (const set & n) : st (n)
  {}

→ NamedSet calcUnion (NamedSet ns)
{
  Named set y = st.calcUnion(ns.st);
  return y;
}
» Simply return bhi likdein tab bhi sahi hai.

→ class SMatrix	class Matrix
{	{
Matrix m;	Array * m;
public:	int rows;
SMatrix()	int columns;
{}	public:
	}
}	

* Association

Composition Aggregation.

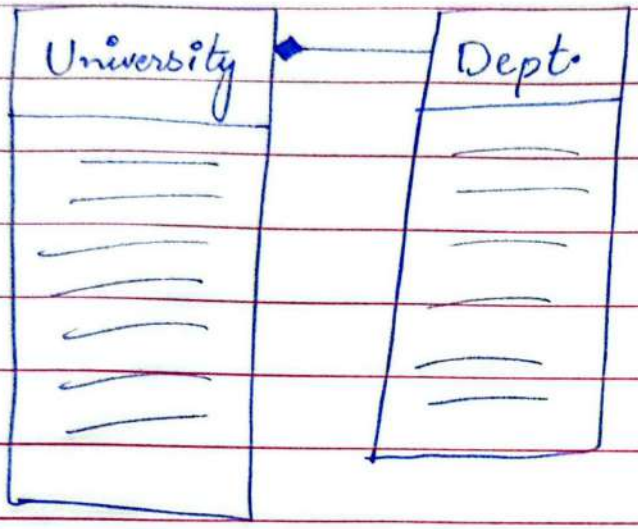


Date: _____ Day: M T W T F S

→ Unified Modeling Language. (UML)

↓

iska aik component hai class diagram.



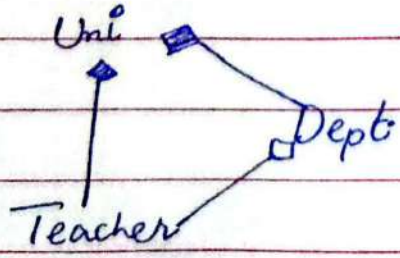
- University is composed of Departments.

~~University~~

→ Whole ◆—— path (composition)

→ Whole ◇—— path (Aggregation)

- This thing consists of this. (90% ye sahi hojayega).



Date: _____ Day: M T W T F S

(Lecture no: 14)

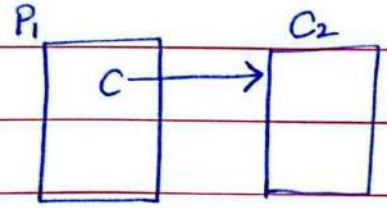
Implementations-

→ Car c1, c2, c3;

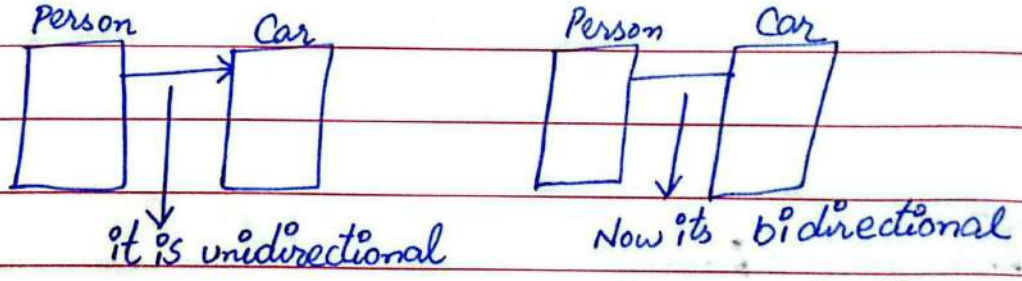
→ Person p1, p2, p3;

- p1.assignCar(&c2);

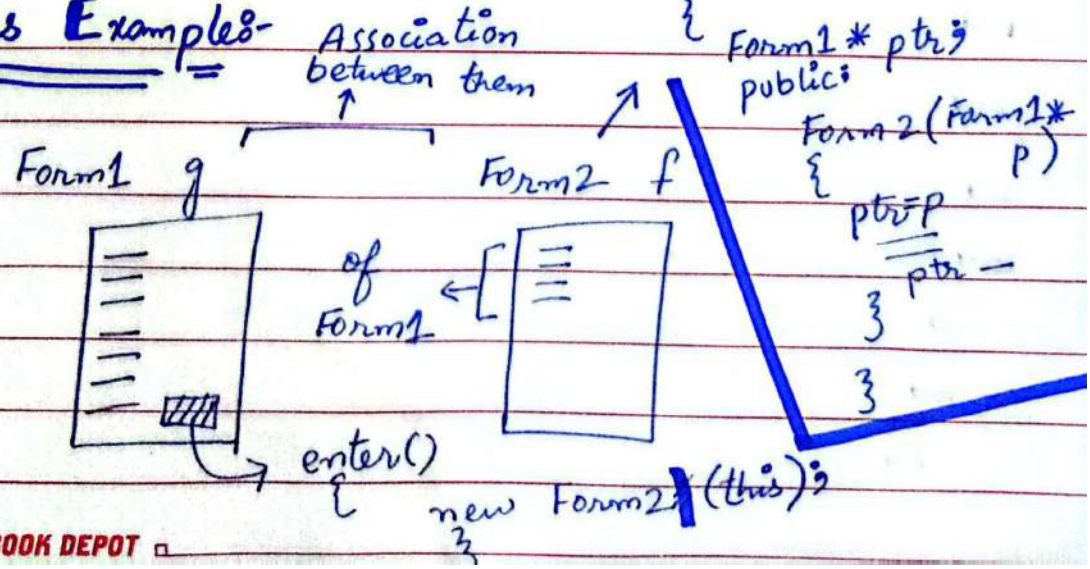
```
Person
{
    Car *c = nullptr;
public:
    void assignCar(car *cr)
    {
        c = cr;
    }
};
```



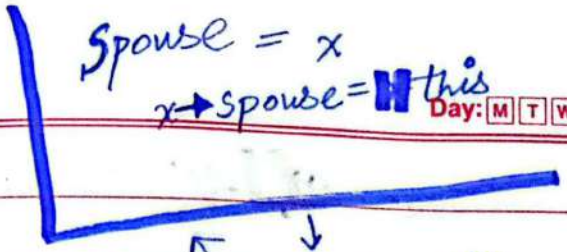
- Association between Car and Person.



Forms Example-



Date: Day: M T W T F S



Two way relation between

```
Person
{
    Person * Spouse = nullptr;

}
```

- Person p1, p2, p3, p4;
  → p1.setSpouse(&p2); ∴ P2P Association.

```
→ void setSpouse(Person * x)
  {   if(x == this)
          return;
      Spouse = x;
      x → Spouse = this;
  }
```

Identifying Nouns:-

- Question ✓
- Quiz
- questionList ✗
- CorrectOption ✗
- noOfQuestion ✗
- DifficultyLevel ✗
- QuestionNo

Date: _____ Day: M T W T F S

→ Question

```
{
    String questionStatement;
    String options[4];
    int correct option;
    int difficultyLevel;
}
```

∴ Question aur QuestionBank ka composition ka relation hai.

→ QuestionBank

```
{
    Question **q;      // q/list
    int noOfQuestions;
    int capacity;
}
```

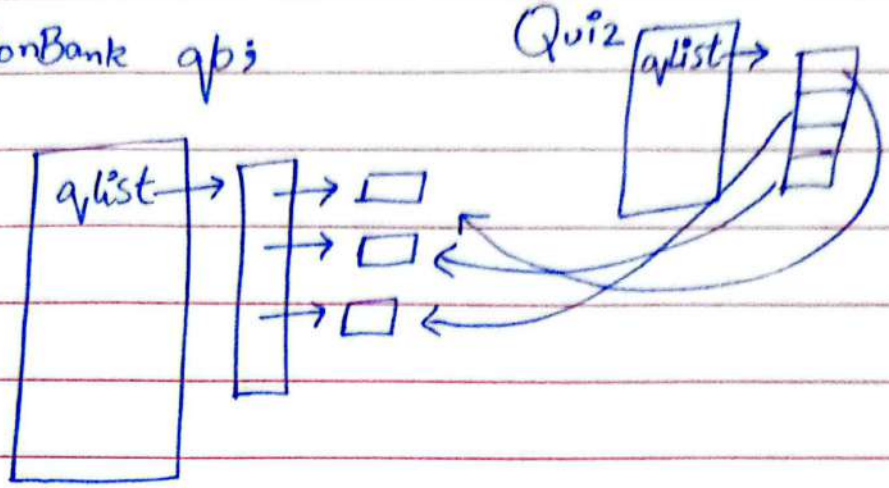
∴ Quiz aur question ka whole part relationship hai. (Aggregation)

→ Quiz

```
{
    Question **q;
    int size;
public:
    addQuestion(Ques *S)
    { q[noQ]=S;
    } noQ++;
}
```

Date: _____ Day: M T W T

- QuestionBank qb;



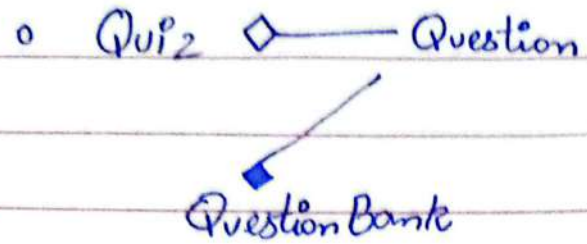
→ Question q;
qb.addQuestion(q);

- Quiz * q = qb.generateQuiz(5);

→ Quiz * generateQuiz (int qCount);
{
int ind. = rand() % noOfQuestions; → questions must be unique.
Quiz *p = new Quiz;
p → addQuestion (—);
qlist[ind.]

* If there is sharing in Aggregation then weak Aggregation.

Date: _____ Day: M T W T F



→ University u;

u.addDept(____);

u.addTeacher(____);

u.assignDept(eId, DId);

→ Coding pov and theoratical pov.

↓ Qb, Q k bagair bann gaya na toh (aggregation)

↓ Qb ka kia faida jab Q nhi hai (composition)

Date: _____ Day: M T W T F S

(Lecture: 15)

Operator Overloading:-

→ feature provided by language, not OOP

Array a {10, 20, 30, 40}
a[1] = 30;
a.getset(1) = 30;

→ all operators are functions that are defined for primitive types.

→ can't change precedence / default associativity rule.

→ Ternary (?:) → can't define Meaning of this for our objects.

→ Kuch aur bhi hain woh google krne / book se dekhne.

→ Binary :-

>	+	+=
<	-	-=
>=	/	%=
<=	*	*=
==	%	/=
!=		

Date: _____ Day: M T W T F S

- operator +, operator %

→ r+t; → r.operator+(t);

calling object

ye Rational ki class me hoga.

Rational operator +(Rational b);

- Rational Rational :: operator +(Rational b).

→ can't change arity of operators. → no. of operators it takes.

→ String a, Date x;

a+x; → function String ki class me hoga.

x+a; → function Date ki class me hoga.

- cin >> a;

object of type istream.

can define operations for user defined types but not for primitive types.

→ C# me hai k jitnay ~~[illegible]~~ relational ~~[illegible]~~ operators hain woh bool return kreingay lazmi.

Date: _____ Day: M T W T F S

→ a+b → concatenate.
→ a+=b → concatEqual.

o copyConst → why Shallow, bit and memberwise copy?

Rational & operator = (const Rational &);
↓ calling object ka address return krta hai.
↓ By default har class me ye hota hai.

→ r = [t=S] ;
→ (r=t) = S ;
↓ agar value return kreingay toh S kisko assign hoga.
isliye alias return kr rahay hain.

o In copy constructor :-
→ *this = ref;

//code nhi likhein gay.

Array & operator = (const Array & ref);
{ if(this == &ref) return *this;] → self assignment check.
this → ~Array();
// copy constructor.
return *this;
}

a = a;
↓ issue ayega. ↑ to solve this issue

Date: _____ Day: M T W T F S

Index:-

a[1];
↓
calling object.

→ simply bas likhna hai bakion ki tarah. no special thing.

→ Jo constructor one parameter lekr execute hojaye woh assignment constructor, type cast constructor etc

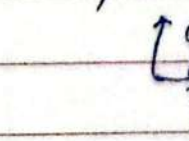
→
```
int * operator[] (int index)
{
    If (int >= 0 && index < rows)
        return data[index];
    exit(0);
}
```

Dry Run:-

m[1];

Matrix m;

/* data



m[1][2]
↓
ye * to int return krega

ye language Provided hai toh chal jayega
↑
→ *int[2]

→ ye return hoga (int *)

Date: Day: M T W T F S

→ () → function call operator.

- m[1][2]
  m(1,2) → aisi classes k objects ko function objects kehte hain.

→ int & operator () (int r, int c)
{ if (check)
  return data[r][c];
  exit(0);
}

* unary ki priority zyada hai.

- ++a +b; → Rational & operator ++ ()
  {
  num += 1
  den += 1
  return *this.
  }

a+++b;

Rational operator++ (int)
{ Rational x(*this);
  num += 1;
  den += 1;
  return x;
}

Date: _____ Day: M T W T F S

(Lecture : 16)

Operator Overloading :-

- cin belongs to class istream.
- cout belongs to class ostream.

- 1+x ?
- cin >> x; ?
- cout << x ; ?

Note:- GO tab krni jab majbur non like here

→ Member Overloading.
→ Global Overloading. (discouraged)

- Find the operators which can't be overloaded globally? (Google) (=, (), [], →, ., ::, ?:)

- x+y → 1 parameter for MO.
- x+y → 2 parameters for GO.

Global Over. :-

```
class String
{};
String operator+(const String &,
                 const String &);
```

Date: _____ Day: M T W T F S

cin >> x :-

& ↑ yahan by value krte toh error ata.

istream operator >> (istream & abc, String & z);

{

z = input();

return abc / cin;

}

Note:-

istream & x = cin;

x >> a;

↙ iski copy ya object nhi bna saktay (restriction hai)

ostream& operator << (ostream&, const String &

→ bagair majburi k Global overloading ki taraf nhi jana hai.

- C# me GO ya global functions ki option nhi hai agr krna toh as member define kro aur static bnao.

- friend functions / classes.

Date: _____ Day: M T W T F S

→ Date y; int z;
String x {"637"};
z = (int) x; → unary operator.
z = x;

Overloading Typecast Operator:-

~~explicit~~ String::
operator int ()
{
→ convert to integer
return 316;
}

Example:-

String x {"214"};
int y = (int) x
↳ explicitly typecast.

→ Agar explicit likhein gay return type me toh implicit typecasting nhi hogi.

o Date :: operator String ()
{
||
}

Date: Day: M T W T F S

granted
• "Friendship is always (given)* never taken"

Friend Functions / Classes :-

```
class A                    void f()  (Not a function of A)
{ friend class B;          {          This has access to
  friend void f();                    private data of class
}                          }          A.
```

↓

jisko friend bnana uski prototype likh deni (priv. & pub. dono me likh saktay hain) (convention ye hai kay top of the line likhna hai.

→ istream aur ostream wali prototypes String class k andar friend krdein phir functions uskay data members ko access kr paye gy.

→
```
class A
{
  friend void B::x();
}
```

Note: isko samajhna but kaam krtay time

use nhi krna.

Forward Declaration.

⇒ **3 cases:-** (important)

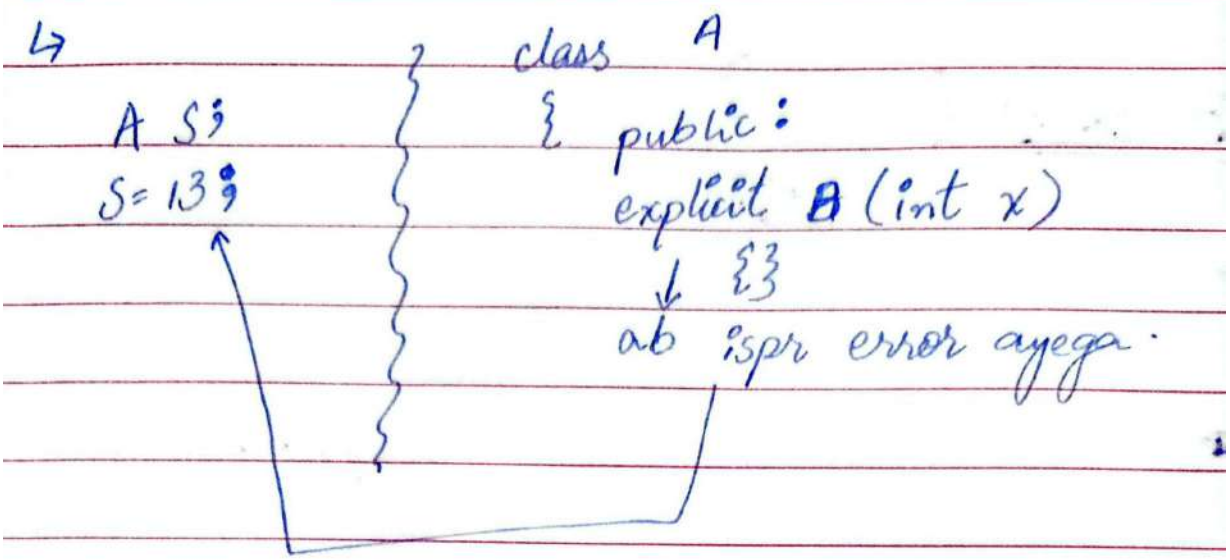
- 2 HF's me 2 classes me friend bnaya.
- 2 HF's hain aik me GF aur aik me class.
- kya 2 classes ke 2 specific functions aik dusray ko friend bna saktay hain.

Conversion Constructor:-

```
String x{"Hello"};
x = 13;
x = String{13};
```

- 1 argument constructor, conversion constructor, type cast constructor. (Sab aik he hain)
- ```
  String x{"Hello"};
  x + "abc";
  ```
  ↳ yahan const char* wala constructor chle ga.
  ```

Date: _____ Day: M T W T F



Situation:-

(1 arg. const which takes Y) ← X ← a = b; → Y (X ka typecast ~~se~~ overload)

a = b; → assign no in X

1. Operator Overload.
2. Constructor. → 2. Typecast Overload

a = (X) b;

1. Arg Constructor. → if I remove this its 2. TO 3. Oper-Overload. giving linker error.

String S; ↓ nhi chle ga as

if (S) { }

if (10) operator bool ()

REHMANI BOOK DEPOT

Generalization.

Date: (Inheritance) Day: M T W T F S

(Lecture: 17)

- Address (String ads, String city, String country)
- Person (int id, String bGroup, Address hAds)
- Teacher (double salary, String specialization, Person per)

Address:-

```
Address (String ads = "###", String city = "###",
         String country = "###")
{
    address.concatEqual(ads);
    cityName.concatEqual(city);
    countryName.concatEqual(country);
}
~Address()
{
    [illegible];
}
String getAds() { return address; }
String getCity() { return city; }
String getCountry() { return country; }
void setAds (String ads)
{
    address.~String();
    address.concatEqual(ads)
}
```

∴ Same for city and country.

Person:-

Person (int id, String bGroup, Address hAds)

→ Person()
{
id = 00;
}

→ ~Person()
{}

→ ~~void~~ int getid() { return id }

~~void~~ String getbGroup() { return bGroup }

Address() { return hAds } gethAds

→ void setid (int idNo.)
{ id = idNo.; }

→ void setbGroup (String bG = "A+")
{
bGroup.~String();
bGroup.concatEqual();
}

Date: _____ Day: M T W T F S

→ void setHouseNo (Address house)
{ hAds = house → ye bhi ho sakta.
hAds.ads = house.ads
hAds.city = house.city
hAds.country = house.country
}

→ hAds.ads.String()
hAds.city.String()
hAds.country.String()

Zarrorat nhi, just for understanding.

Teacher :-

Teacher (double salary, String specialization, Person p)

Inheritance :-

child / Derive class ↑ parent / Base class ↑

class Person : public Address
{
int id;
String bGroup;
}

perfect case ↑

Inheritance perfect feature hai agr kisi class k features as it chahiye (Then Don't use whole part). (Agar aik bhi cheez nhi chahiye toh 10 me se 9 chahiye tab bhi whole part relationship).

→ Agar aesa kreingay toh certain assumption k ~~sath~~ (error k chance usme bhi hain)

Date: Day: M T W T F

- Person → Address — this is one way relationship.
  is-a relationship
- Base (int a=1, b=2) → aesay nhi ho sakta hai.

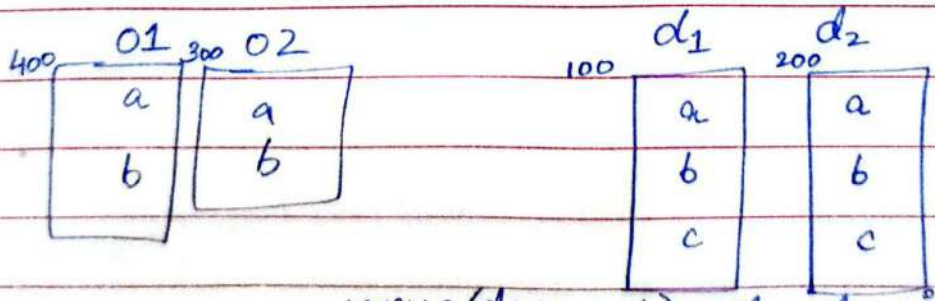
→ Sirf public part access kr saktay inheritance se.

→ inheritance copy paste nhi kr rah hai.

→ Base k this k type aur hai aur derive k this ki koi aur type hai. (object layout)

Diagram:-

Base o1, o2; Derive d_1, d_2



reverse (down cast) allowed nhi, only upcast allowed hai

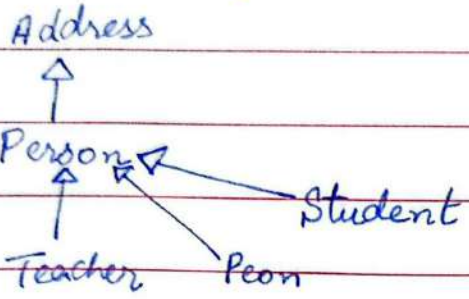
* Base class reference can always refer to derive class objects (Key of inheritance)
* derive ka kuch hissa exact base jesa hota hai.

Date: _____ Day: M T W T F S

→ Base *p = &d1

- p is pointing to ~~1000~~ 100

- Inheritance / Generalization.



more generalization (less specialization)

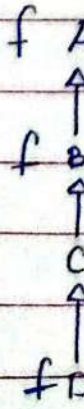
less generalization (more specialization)

definition not recommended

- 2 functions agar Base aur derive me hain toh derive wala call hoga.
- koi cheez likhtay woh pehle child me dekhe ga agar usme nhi toh uskay parent k pass and so on.

agar Base wala krna toh
Base:: test() → imp
d.Base::test()

- Base *p = &d;
  p→test(); ∴ ab base wala call hoga.



C obj;
c.f(); // B wala call hoga.
c.A::f(); // Ab A wala.

MIL me initialize appreciated hai-

Date: Day: M T W T F S

Teacher apna office address kese rakhay ga?

→ ab class ka naam likhein gay.

→ Derive (): Base {1, 2}..

{}

Date: _____ Day: M T W T F S

(Lecture: 18)

Copy Constructor of Base & Derive:-


```
B(const B& r):A(r)
{}
operator = (const A& r)
{
    A::operator = (r);
}
```

→ Array — private: int a; → protected: getA(); setA();

Sorted Array → Sort()

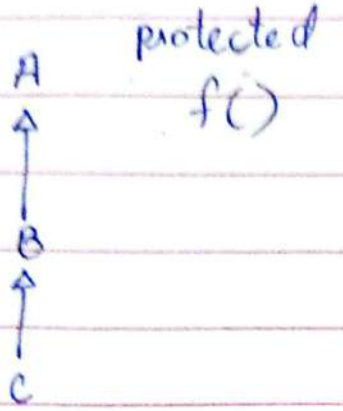
→ inme implementation copy ki but logical sense hai.

Access Modifiers:-

Agar Inheritance na hon toh ① private, ② public, ③ protected do-some hain.

Date: Day: M T W T F S

→ protected = me aur mere child bas.



→ B O1;
O1.f(); X
A O2;
O2.f(); X

→ jo cheez . / laga kr access ho woh class ka interface hai.

Types of Inheritance:-

1) private.
2) public.
3) protected. dunya sirf isko mantay hain.

→ jo kaam krna woh public inheritance se krna hai.

Date: _____ Day: M T W T F S

→ Kuch bhi na likhain toh by default private.

```
class B : private A
{
}
```

∴ a1 A public: f();
        protected: g();

b1 B : public A
   void f()
   { A::f(); }

Main:-
A oa;
oa. ____

→ ye private inheritance me bhi hoga agar class k andar ha

B ob;

a1
b1

ob.g();
ob.f();

→ A * p = & ob;
p → f(); ← main me nhi hoga.

Base / Derive	priv.	pub.	prot.
priv.	X	private	private
pub.	X	pub.	prot.
prot.	X	prot.	prot.

Date: Day: M T W T F S

→ jesi marzi inheritance ho private part kisi class ka kabhi bhi nhi access ho sakta hai. (Important)

→ A
↑
B : private A
↑
C (is me A nhi hai woh sirf B K pass hoga)

(Ye kaam pehle bhi hota tha)

• Constructor Inheritance:-

public:

using Base::Base; → public constructors ab access ho payein gay.

→ jiska control Khud sambhal lein gay woh nhi krega phir.

→ - friendship inherit nhi hoti (check krna hai)
- friendship is always granted.

te: _____ Day: M T W T F S

A f()
f(int)
↑
B
f(int, int)
↓
isne A k f's ko hide krdia hai.

↳ - function overloading
- // Hiding
- // overriding

Solution: B.A::f(2);

using Base::f; → ab hide nhi hongay.

↳ private:
↳ phr access nhi hoga.

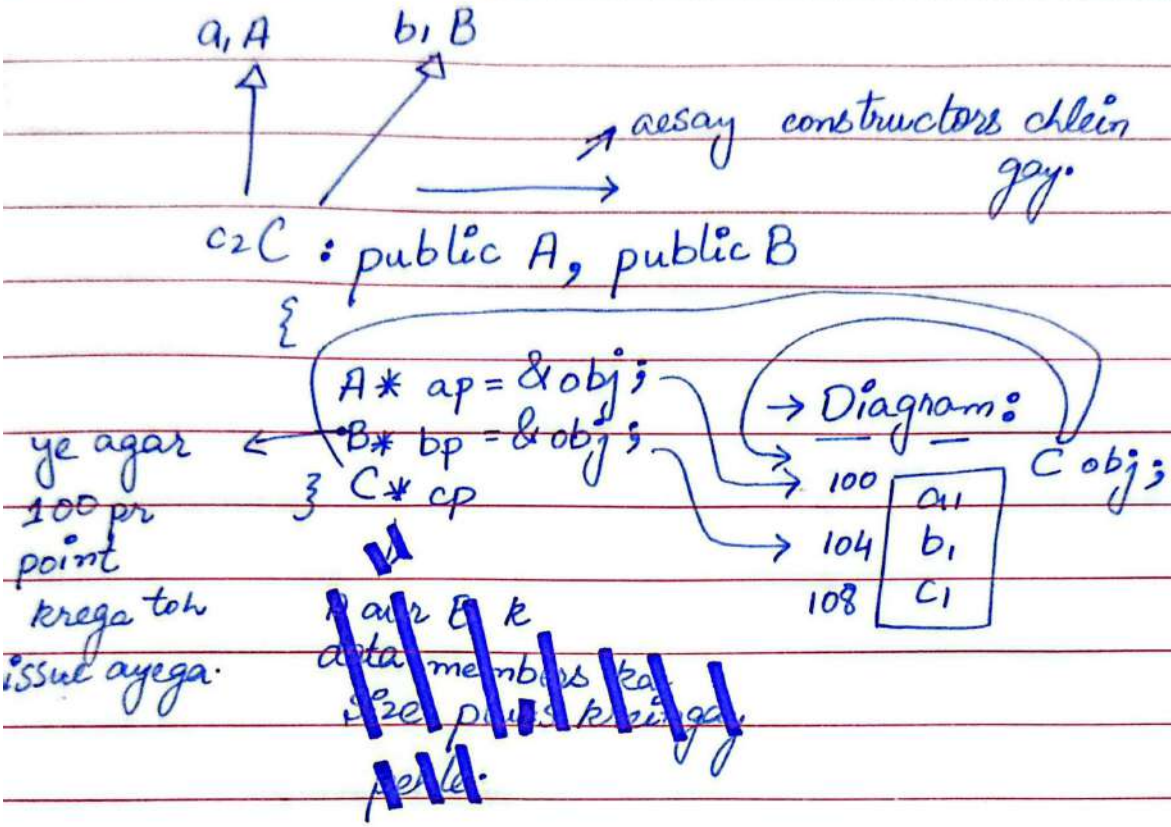
Set Set calcUnion(NamedSet & r)
↑
→ is-a
NamedSet

↳ NamedSet calcUnion(NamedSet& rs)
{ NamedSet x(Set::calcUnion(rs));
return x;
}

Polymorphism :-
A *p = new B; → new A; Inkay bis aab
p→f(); se chlega.

Date: _____ Day: M T W T F S

• Multiple Inheritance-



→ Agar aik function same naam ki 2 Base classes me hogta toh error dega (prototypes puri baad me dekhein gay)

→ C obj;
B * p = (A*)(B*)&obj; // it'll change value of a.
p → b = 9236;

Date: Day: M T W T F S

(Lecture : 19)

Polymarphism-

Agar function (Base walay k peechay) virtual likhein gay toh tab B wala f() call hoga.

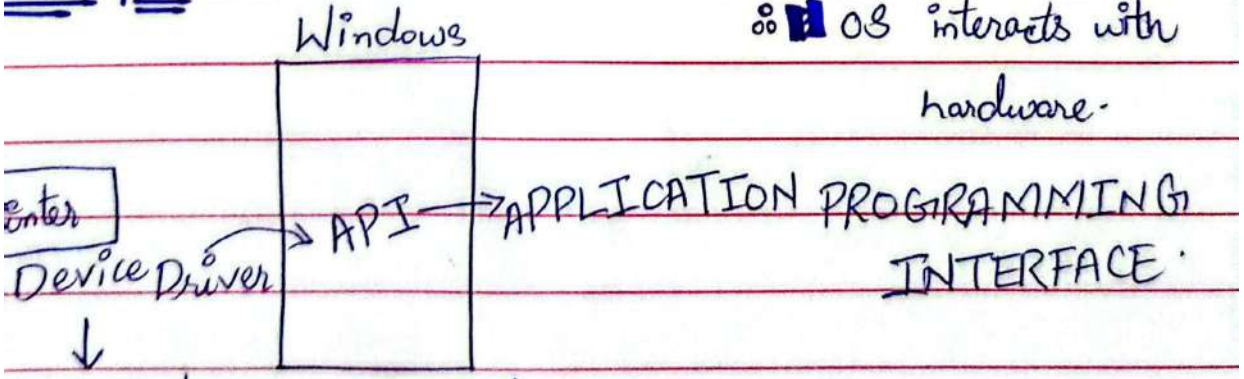
```
new A;
A * p = new B;
p → f();
p → g();
```

Note:-

```
f()  A
g()  ↑
{
f();
}
f()  B
```

B 01;
01.g();
↓
isme A wala f() call hoga.

Example-



∴ OS interacts with hardware.

∴ printer can interact with Windows.

→ Windows GIF → Global functions se bani hai.

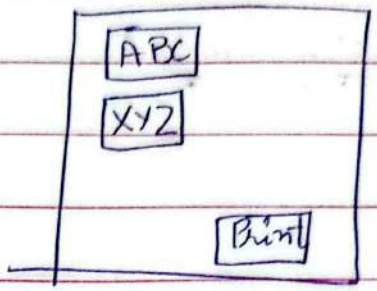
Date: Day: M T W T F S

:: API se interact krna tough tha
toh MFC ki layer daal di upar.

(part of Windows API) → assume.

```
class Printer
{
public:
    virtual void print()
    {}
};
```

Note :- Ctrl+P




```
class ABC: public printer
{
public:
virtual void print()
  {
    // How printer will work.
  }
};
```

→ device driver coding of printer.

→ Important.

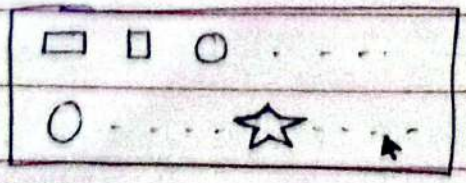
:: Sir said derive me bhi likhdo virtual issue nhi hai

```
void Printer (Printer* p)
{
  p->print();
}
```

→ part of Windows API

↓ connected with print button.

Example 2:-



Date: _____ Day: M T W T F S

```
class Shape
{
    public:
        virtual void drop();
            {}
};
class line : public Shape
{
    public:
        virtual void drop();
            { // How line draw?
            }
};
void Draw (Shape* S)
{
    S -> drop();
}
```

→ Same will be for triangle.

→ This is global function.

Another Solution:-

→ Do simple if else.

→ yahan pr agar new shapes ati hain toh existing code modify krna paray ga.

→ Specific functionality kisi specific

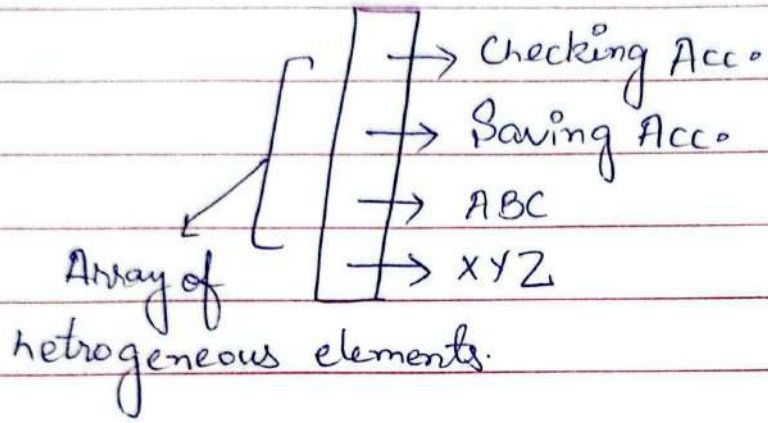
Date: Day: M T W T F S

∴ But Windows to OOP language nhi hai?

∴ print dabanay se object kese poncha?

Shape me dalnay me issue ayega.

- Account * * ac



- debit (Account * ac)

```
{
    ac → debit();
}
```

Why Not Virtual Every Where?

virtual void f(void) ← A a1
void g()
virtual void h()
// // x()

B → virtual void z()
virtual void h() latest
virtual void f() → implementation
virtual void y()
void t()

b1

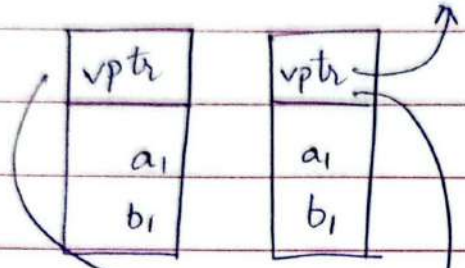
Date: _____ Day: M T W T F S

∴ 1) void (*p) (int, int) = g
2) void (*p)(void) = f

→ To call:-
p(67, 47);

object layout:-

B obj1, obj2

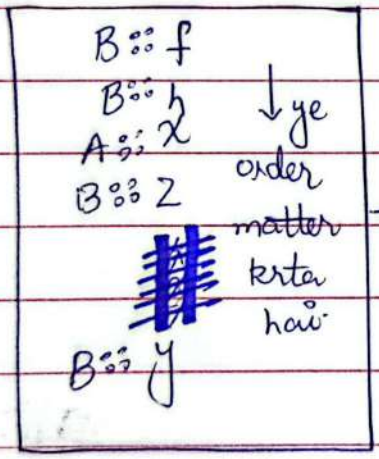


size of this pointer is Machine dependent

∴ vptr har object ka hoga

∴ Vtable har class ka hoga

B class ✓-table:-



↓ye order matter krta hai

→ array of function pointers

→ B class k apnay aur inherited functions k addresses.

QUESTIONS:-

→ Type kya hai p ki?
→ function hai?
→ accesible hai?
→ virtual to nhi hai

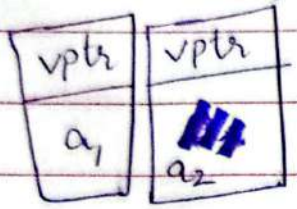
→ A* p = new B;

p → g();
p → h();
p → x();

• Function Binding
↓
Agar faisla runtime pr ho to runtime binding agar compile time pr to compile time binding.

Date: _____ Day: M T W T F S

→ A obj1, obj2;



A v Table

A::f
A::h
A::x

aj ye discuss kia hai-

→ runtime Binding [redacted] == runtime polymorphism.

→ Static and dynamic type of pointer.

↓ at the time of declaration type.

↓ actually jis type ko point kr rah woh-

→ inherited virtuals + self virtuals = location of function.

→ virtual ko agar derive me [redacted] override krtay hain redefine nhi-

→ virtual f() A

↑

B

f()

↙ ye bhi virtual hai (automatically)

Date: _____ Day: M T W T F S

(Lecture: 20)

Debugging:-

↳ F10 → To start debugging.
→ Debug → Windows → Watch → Watch1
→ Autos automatic watch hai jo current function k variable dikhata hai.
→ Ctrl+F5 to stop debugging.
→ F11 → step into
→ Right Click → move to cursor.

- function me gaya bhi ya nhi toh iskay liye Breakpoint lagana hai.

- Breakpoint pr tab rukega jab Start Debugging kreingay.

(Return Value Optimization)

Before C++:-

```
int f()
{
    int a;   <-- =10
    return a;
}
```

a: 10

```
main()
{
    int x = f();
}
```

x: 10

10

Date: Day: M T W T F S

VS 2010:-

temporary nhi banata tha-

→ String (const String &) ↘ temporary
{
}

const & se he bind ho sakta hai bus.

Today:-

String f()
{
String x = 10;
return x;
}

iska copy bhi nhi chla

main()
{
String a = f();
}

x → '1010'

a → '1010'

→ na copy const. aur na. dest. chla.

→ String x;
x = f();] → No RVO (x ka const. chle ga but temp. wala kaam nhi hoga)

→ String x;
f();] → RVO, yahan temp. bheje ga bus.

Date: Day: [M][T][W][T][F][S]

```
main()
{
   String x;
   x = f();
}
   x
  [10]
```

```
String
f()
{
   String a = 10
   return a;
}
   a
  [10]
```

→ String f(String &h)
{
return h;
}

∴ ab nhi hogi RVO because h temporary nhi hai.

// Woh marnay nhi ja rah hai-

```
String f()
{  String a, b;
   int x;
cin >> x;
   if(x)
      return a;
   else
      return b;
}
```

// ab RVO nhi hogi.
// Move constructor chle ga.

Date: _____ Day: M T W T F

C++14:-

- Move constructor kaam krta tha.

```
void mySwap (String &a, String & b)
{
    String temp= move(a);
    a= move(b);
    b= move(temp);
}
```

Date: Day: M T W T F S

(Lecture : 21)

virtual void f()

A
↑
B

f()

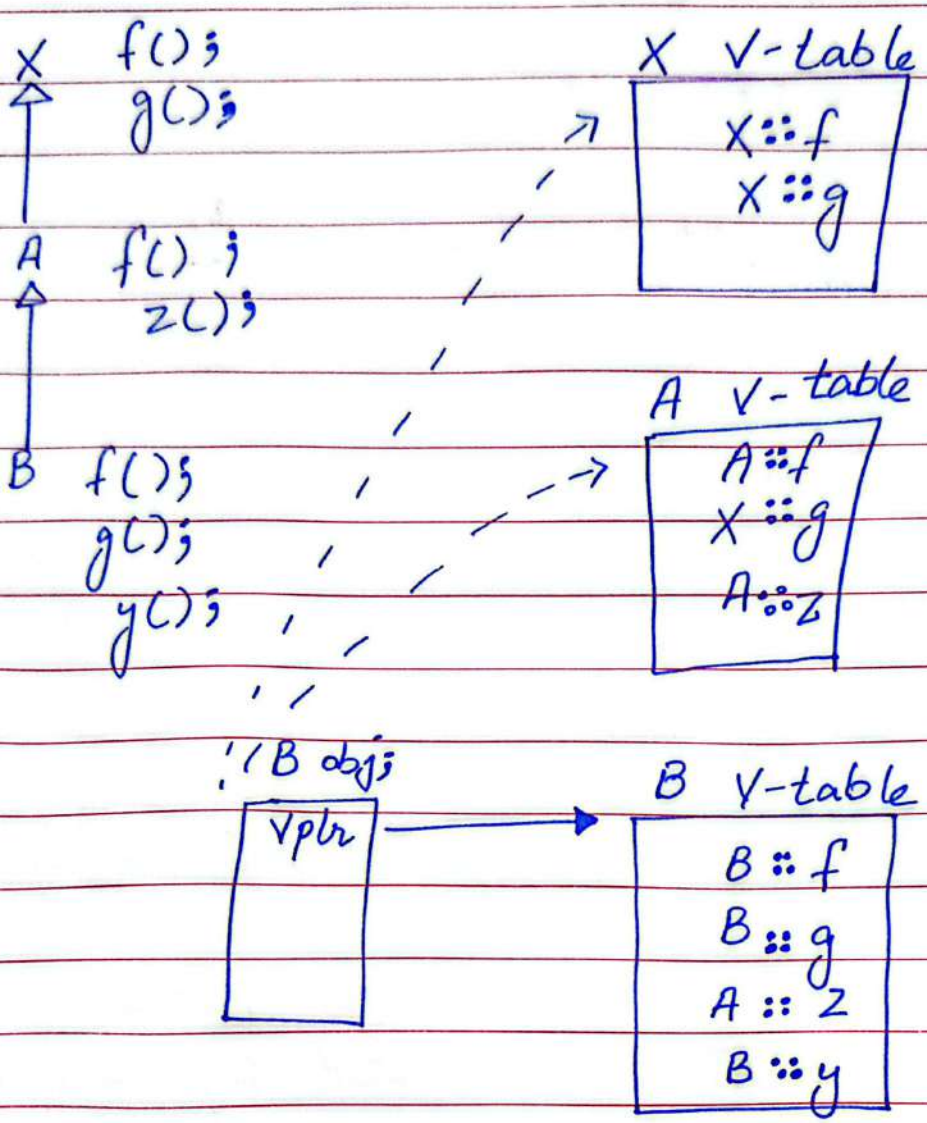
A() { this → f(); }

B()
{
}

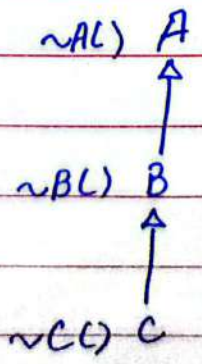
B obj;

* B walay f() ko call nhi jani chahiye.
* constructors me jahan tk initialized hoga wahan wali latest implementation chle gi.
* B se pehle uskay upar wali has class ka V-table bnay ga.
* vptr ki value during construction and destruction change hoti hai.

Date: _____ Day: M T W T F S



Destructors :-



C obj;
obj.~C();
↓
ye base k destructors bhi chlaye ga.

Date: _____ Day: M T W T F S

Note:-

```
A *p = new C;
delete p;
```
] → undefined behaviour

p → [a1 → Sirf ye khtm hoga? ↓ pata nhi
b1
a1]

Solution:-

base k destructor ko virtual krdo.

→ Constructors ko virtual mark nhi kr saktay.

- **Situation for virtual object:-**

```
f (A *p)
{
    B o1(*p); → B (const B & ref)
}
```
↓ Derive can't refer to Base.

How it can be done?:-

Date: _____ Day: M T W T F S

A → virtual A* clone()
{
return new A (*this);
}

B → A* clone()
{
return new B (*this);
}

C → A* clone()
{
return new C (*this);
}

→ Ab hojaye ga yes:-

f (A *p)
{
A* q = p → clone ();
}

Q:- Printer ka object kese ja rah hai?

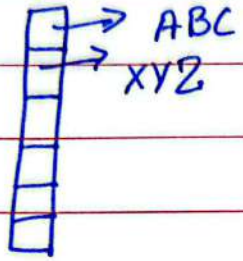
class Printer → In Win. API
{
public:
virtual void print() {}
};

class ABC : printer {
public: void print();
};

XYZ class

Date: _____ Day: M T W T F S

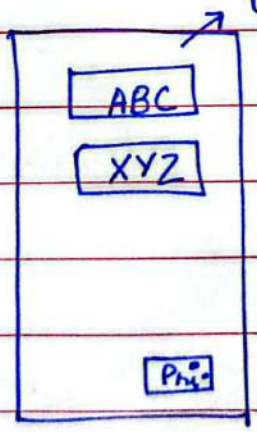
class PrinterList → Win API
{ String printerName;
Print * * pl;



public:
addPrinter(Printer* p);
};

Global:-
PrintList plist;
main();
{
pList.addPrint
(new ABC
}

DropDown GUI:-



* Above aesi heirarchy hai jisme Printer ka object kabhi bnayein gay le nhi.

Date: _____ Day: M T W T F S

Example class:-

```
class Switch → just an interface.
{ public:
   virtual ON();
   virtual OF();
};
```

* Such classes are called Abstract classes.

```
class
→ Printer
  {
     String printerName;
     public:
        virtual void print()=0;   ← pure virtual
        virtual String getPrinterName();
  };
```

* aisi class jiskay andar aik bhi pure virtual function ajaye. uska object nhi banay ga aur woh abstract class hogi.

A → pure virtual

B → Agar isme pure virtual override nhi hua toh ye bhi pure virtual hogi.

Abstract Class & Pure Virtual Class:-

→ baki languages me same hai.

→ Abstract Class:-

→ jisme mix kaam.

→ Pure Abstract Class:-

→ jisme only pure virtual functions hongay. (implementation koi nhi hogi).

* Baki languages me extends hota, Access Modifiers nhi.
* interface ko implement, class ko extends.
* No interface keyword, backend pr # define hua wah class ki jagah.
* Pure Abstract == interface → rest languages.

∘∘ Cpp have 4 typecaste operators.

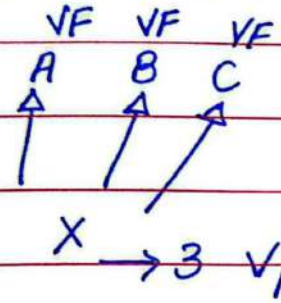
Date: Day: M T W T F S

↑ To save time writing common code.

* pure virtual ki implementation de saktay hain. (Within class nhi outside class)

* MFC → search by yourself.

• Issues → Multiple Inheritance:- → Bus C++ me hai. (aur aik 2 me bhi)



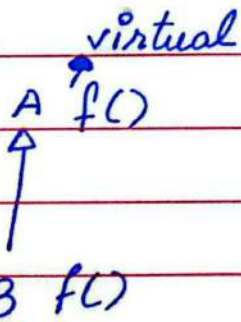
→ 3 Vptr's

→ Scenario:-

∘∘ f(A**p)

{

→ Sirf B's ka f() chle.



// Next page.

∘∘ Agar Chua toh tab bhi typecaste hojaye ga.

}

Type Casting:-

1) () → typical jisme ↑ jisko ↑
2) static_cast< >( )
3) reinterpret_cast< >( )

Date: _____ Day: M T W T F S

4) const_cast <>()
5) dynamic_cast <>()

→ ye valid downcasting krta hai.
↳ isko dynamic type pata hoti hai.

→
```
A *p = new A
       // B

B *q = dynamic_cast <B*> (p);
if(q)
{
    q -> f();
}
```

Agar A hua toh nullptr.

→ function.

→ Runtime Type Identification (RTTI).

⁂ dynamic_cast v-table se type ki info leti hai.

Date: Day: M T W T F S

(Lecture # 22)

```
        void f();
A       void g();
↑       void (*q)(void)
B
```

```
→ B()
  {
    q = g;
  }
```

```
→ A()
  {
    q = f;
  }
```

```
main()
{
    B bObj;
    A aObj;
    A * p = &dbj;
    p → q();  → f() call hoga.
    p = &bObj;
    p → q();  → g() call hoga.
}
```

Object Slicing:-

A aObj; → aObj [a1]

B bObj → bObj [a1 b1]

Date: Day: M T W T F S

∝ bObj = aObj; (Syntax error)
aObj = bObj; ✓

→ aObj me a1 hoga bus.

→ f(A r) → object slice hoga.
{
}

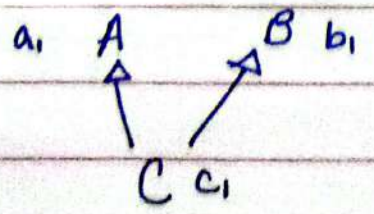
→ f(A &r) → ab slice nhi hoga.
{
}

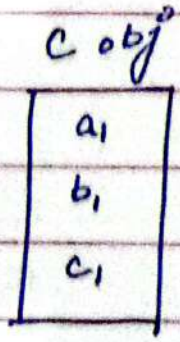
→ Point 1 across compilers allowed nhi (fact mean kr istemal nhi krna)

→ Move constructor ki cheezein dekhni google se? (cppreference.com) → puchna hai sir se

↓
Move != copy → ek hai toh dusra by default nhi hai.

Multiple Inheritance:-

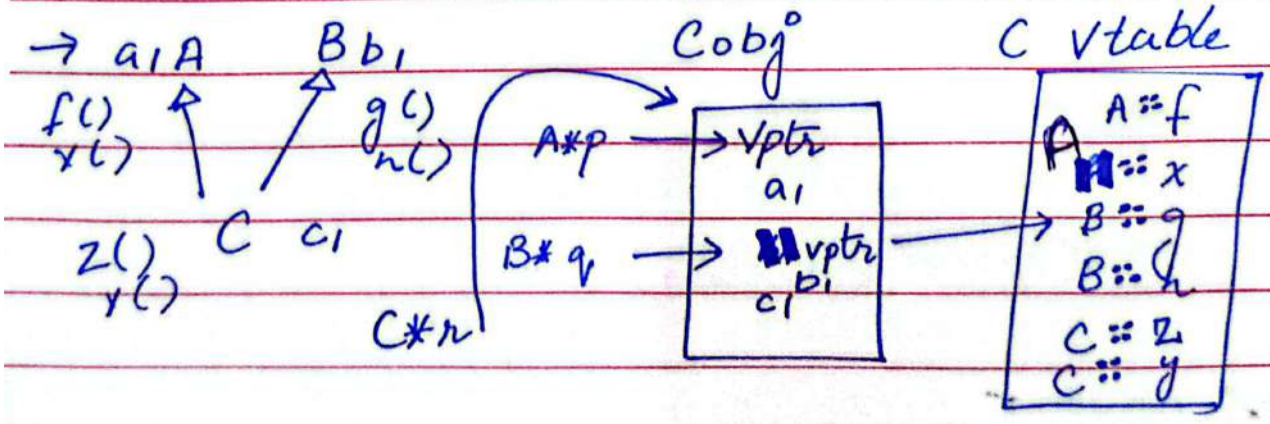




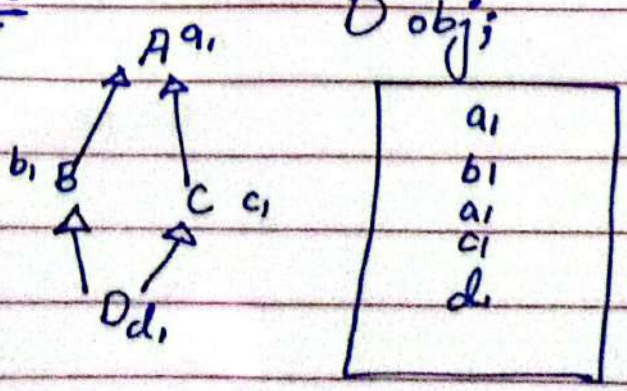
A* = obj → a1
B* = obj → A b1
C* = obj → a1

* Diamond Inheritance Problem ki wajah se multi-inheritance baki languages me nhi.

* Agar class me virtual functions hain aur koi data member na ho toh uska size vptr k size k equal hoga.



Major Issue:-

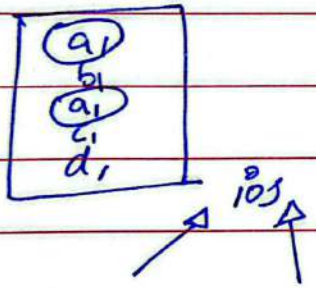


Date: _____ Day: M T W T F S

obj.a1 = 91 ✗
" d1 = 90 ✓
" c1 = 60 ✓
" b1 = 10 ✓

→ obj.B::a1 = 91 → ab chal jayega.

∴ D obj;

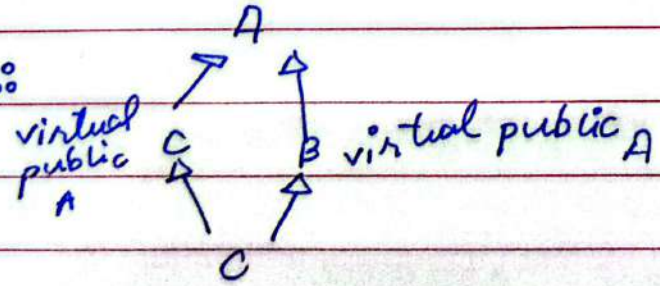


ios

→ istream ostream

ostream

∴



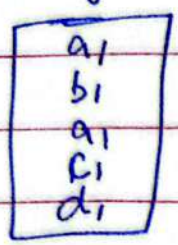
Solution :-

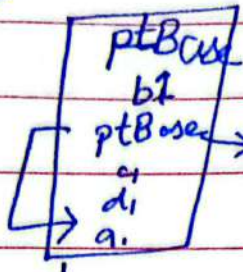
virtual inheritance.

∴ Scratch k mamlay me sahi hai but already implemented unko me nhi change kr sakte.

Date: Day: M T W T F S

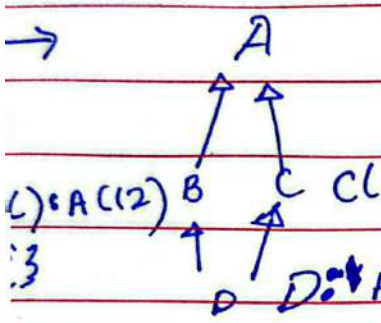
D objs →





Bus pointer yahan point krega.

→



→ #1 issue of discarding diamond inheritance

→ ab a1 aik he hai toh issue hai.

→ Dono cont. me kon chle ga?

→ ye D decide krega

→ Access available hogi toh.

- virtual function can't be made static.

→ template<typename T>

→ void myswap (T &a, T &b)

{

T temp = a;

a = b;

b = temp;

}

Date: _____ Day: M T W T F S

```
→ bool Search(T *arr, int size, T key)
{
    int i=0
    while (i<size && a[i]!=key)
    {
        i++;
    }
    return i==size? -1: i;
}
```

→ class Templates make classes generic class.

- STL → Standard Template Library.
  ↳ built-in bohot sari classes aur functions hain.

Date: Day: M T W T F S

(Lecture : 23)

Templates:-

argument should be a type.

↑ → capital & single letter

```
template<typename T>
void mySwap (T &a, T &b)
{
    T temp = a;
    a = b;
    b = temp;
}
```

convention hai.

function Template se templated functions generate hongay.

```
int main()
{
    int a=1, b=11;
    mySwap<int>(a, b);
}
```

↓ naya function create kro int ko T se replace krdo.

→ same will be mySwap <double>(x,y).

→ mySwap(x,y) → aesay bhi hojaye ga.

↓

dono different type ke hongay toh error dega.

Date: Day: M T W T F S

→ double c=10;
print <int>(c);
↳ ab int wala hoga.

→ Multiple Template parameters.

→ Template apnay aglay function tk bas replace krega.
↑ global

→ funtion templates header me likhnay, alag alag nhi likhnay abhi.

→ main se upar template + prototype.
→ main se neechay template + definition.

Class Template :-

```
template < typename T >
class Array.
```

* classes k case me hamesha explicit argument deduction krni parayi.

* 1) Array <int> a{5};
2) Array <float> b{3};] → 2 classes generate hui hain.

Date: Day: M T W T F S

Backend pr :- different name ki classes hongi.

class Array<int>, class Array<float>

→ Array<int> *p = &a;

→ Array < Array<int>> x{3}; int* data

pehle ye banegi

phir ye.

Array<int> * data.

→ for (int i=0; i<x.getCapacity(); i++)
{ for (int j=0; j<x[i].getCapacity(); j++)
cin >> x[i][j];
}

Dictionary Example:-

class Pair
template <typename T, typename K>
{ public:
T key;
K value;
}

Date: Day: M T W T F S

```
template <typename K, typename T>
class Dictionary
{ public:
    Pair<K,T> * list;

};
```

→ C++ me templates ka syntax itna Synced nhi hai.

→ #include <initializer_list>

```
main()
{
  initializer_list<int> list = {10,20,30,40,50}
  cout << list.size();
} cout << list.begin();
t int *p =
```

↓ ye no. of arguments khud btaye ga.

↓ ye sirf element ka address dega. → constant

```
cout << list.end();
```

↓ akhri element k baad wala address dega. (for comparison).

→ for (const int x : list) — int array pr bhi work krega. any class (agr begin aur end hoga)

```
    cout << x;
```

primitive data types ka pata hoga k start aur end kab?.

Date: _____ Day: M T W T F S

→ auto x = 978;

↓ khud data type decide kre ga.

:

→ {10, 20, 30, 40}

↳ iski type initializer_list hai.

* Array (initializer_list<T> list = {})

{

// changes + initialize;

const. = ↓ (Array<int>)

Array

↑

}

→ main()

{ or Array<Array<int>> = {, , , }

Array<int> a = {1, 2, 3};

↳ const. = ↓ (Array<int>)

Array.

}

→ for (auto ls : list)

{

for (auto value : ls)

{

cout << value ;

}

}

Date: _____ Day: M T W T F S

```
→ T * begin()
  {
      return &data[0];
  }
→ T* end()
  {
      return &data[capacity];
  }
```

Date: _____ Day: M T W T F S

Exception Handling

→ Exception arises during execution of the program. (during runtime).

→ Keywords: try, catch & throw.

Try & Catch:-

A block of code that causes the exception is try block and code to handle exception is written in catch block.

* Throw is used to throw an exception.

→ * Every try should have a catch.
* A try can have more than one catch.

Syntax:-

```
try { throw something }
catch ()
{ }
```