C++ by Saurabh Shukla

PAGE No.
DATE / /

My learnings / Index

1. Introduction to C++
2. Identifiers
3. Input / Output
4. Reference variables
5. Func'n in C++
6. Structure in C++
7. Classes and objects
8. Static Members
9. Constructor
10. Destructor
11. Operator overloading
12. Friend function
13. Inheritance
14. Cons and Destructor in Inheritance
15. this pointer
16. new and delete
17. Method overloading
18. Virtual function
19. Abstract Class
20. Template
21. File Handling
22. tellg and tellp
23. seekg and seekp
24. Initializers
25. Deep copy and Shallow copy
26. Type conversion primitive to class type
27. Conversion class to primitive type
28. One class to another class type

PAGE No.
DATE / /

29. Exception Handling
30. Dynamic Constructor
31. Namespace in C++
32. Virtual destructor
33. Nested class
34. Introduction to STL
35. STL Containers
36. Array in STL
37. Pair in STL
38. Tuple in STL
39. Vector class
40. List class
41. Map class
42. String Class
43. String Class

Lec-1 ॐ गणेशाय नमः ॥

卐

PAGE No.
DATE / /

⇒ C++ follows bottom up approach

C	C++
Subset of C++	Super set of C
procedure ori. prog. lang.	OOP Language
Top down	Bottom up approac

What's OOPS ?
Revolves around the concept of "Object".
To know intact of variable, we use class.

→ 5 Principles of OOP's
- Encapsulation → (Ek objectse related saasic cheez ka ek grp means encaps.)
- Data Hiding =
- Abstraction
- Polymorphism
- Inheritance

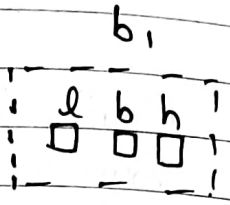
→ Class is an blueprint of an object.
(Example- Ghar ka naksha hai wo class like usse future mein real cheez develop hogi so waise)
→ Object is large memory block which contains several variables but, variable is single data memory block.
→ Class making is a new data-type (user-def / non-primitive / Cutsomized)
→ Class is a means to achieve encapsulation.
→ Object is run-time entity → memory isko milegi → consumes ram → class has 0 memory
→ Object is an instance of class.

Ex-

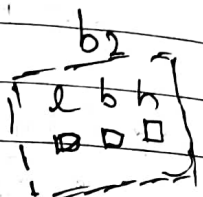
Datatype

```
class Box
{
    int l,b,h;
    void set dimension (int x, int y, int z)
    {....}
    void show dimension ()
    {...}
};
```

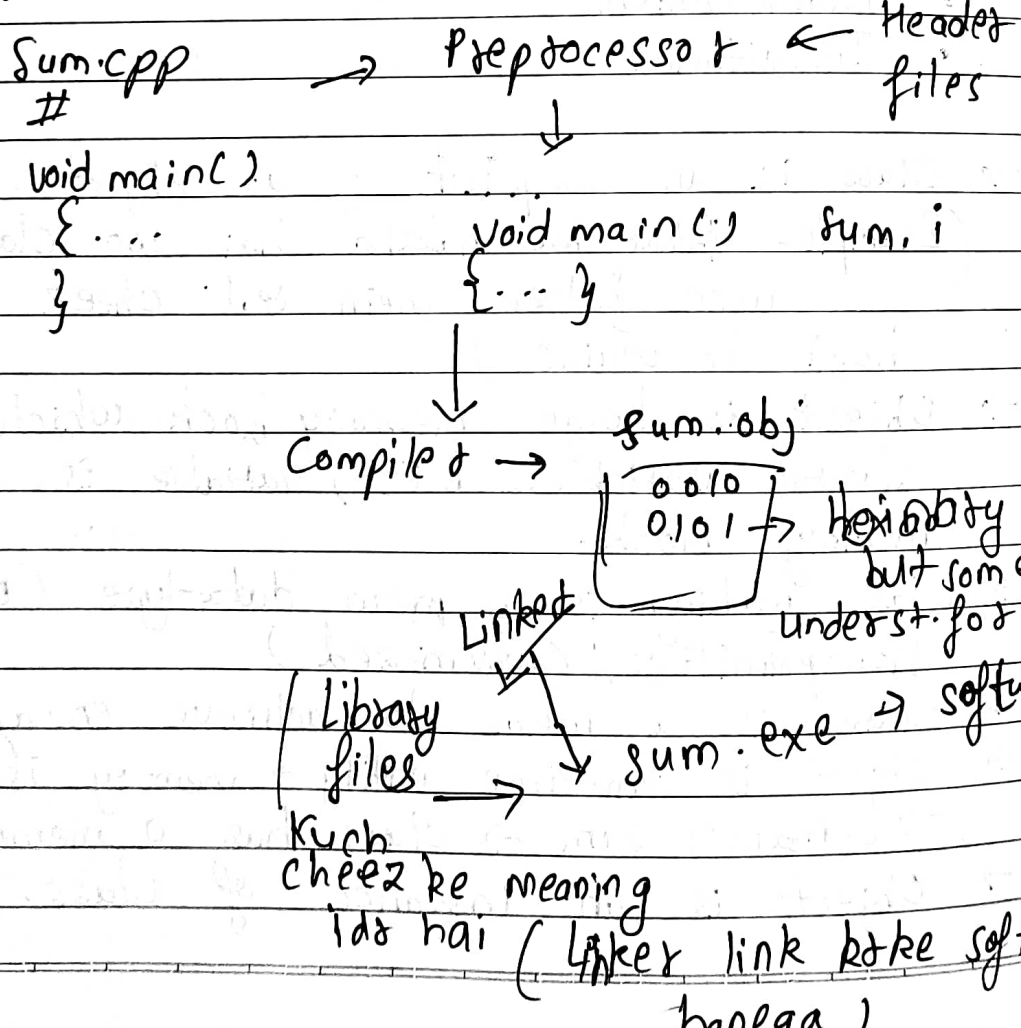
- box b1 ;
  ↑ object



- box b2 ; same as memory of b1 but l, b, h will de diff.



Software development in C++ :



Lec-2

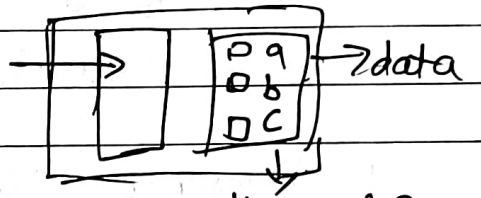
Constant → Data = information = constant

Primary
- Integer → 23, -341, 0, 5
- Real → 3.4, -0.06, 3.0
- Character → 'a', 'A', '+', '2', '_'
  - length should be one with single qoute ''

'-3' its a integer
'mob no' is not a character

"Bhavesh" its string always in double " ".

Program consumes memory in RAM



(Un memory location ke naam hai jaha par hum apne prog. ke data store krenge)

Variables:
→ Variables are the names of memory locations where we store data.
→ Variable name is any comb^n of alphabet (a to z or A to Z), digit (0 to 9) and underscore (_).
→ Valid var. name can't start with digit.

1 byte = 8 bits

Data types

Data type	Example	Memory size	
• int	int a,b ;	2 bytes	integer
• char	char ch='a';	1 byte	char
• float	float k = 3.45 ;	4 bytes	real
• double	double d1 ;	8 bytes	real
• void			

output
cout is a predefined object. [func( ) ← this denotes func]
The operator << is called the insertion or put to operator.

cin is also predefined.
The oper. >> is extraction or get from operator.

endl is a manipulator and it's declared in iostream.h .

Reference Variable :-

int x = 5;

int &y = x; ← ref. variable
y++
cout << x;

x : [5+1] 4e30d ← base address

output - 6

जल / पानी
meaning same.

→ Reference means address
→ Reference variable is an internal pointer
→ declaration of ref. variable is preceded with '&' symbol (but do not read it as 'address of')

&y = x // read as y assign add. of x

→ must be initialized during declaration
→ can be initialized with already declared variables only.
→ Can't be updated (means y ke andar x ka ref. will be stored till y is mortal)

```
{   int x = 6;
    int &y = x;
     y++ ;
    cout << x << endl;
    return 0
}   Output x=7
```

Lecs Functions :-
→ function is block of code performing a unit task.
→ Func^n has name, return type and arguments.
→ They are predefined and user-defined.
→ Predefined functions are declared in header files and defined in library files.

Ex-

```
#inclu <iostream>
void main()                    ⟶
{                    arguments
   void fun();   ⟶ Function declaration
   cout<<" you are fun of main ";
   fun();        ⟶ Function call
}
void fun();
{
  cout<<" You are fun ";
}
```

Globally declare

void has no return type

} ⟶ Function Definition

~~Declaration~~

PAGE No.
DATE / /

Declaration - (Prototype)

Return Type function name (argumentList);

→ Func^n needs to declared before use (like variables) as its bottom-up.
→ Can be declared either globally or locally.
→ Fun^c def^n is a block of code.

Formal arguments
x - Ordinary variables of any type (call by value) *x
*x - Pointer variables (call by address)
&x - Reference variables (call by reference)

&x = a // read as x assign def of a or add.

Benefits of Function :-
1. Easy to use 2. Easy to modify
3. Avoids rewriting of same code 4. Easy to debug 5. Better memory utilization

Func^n in prog. is to save memory (means 64Kb ki 16 bit architecture ko agar 100 kb ki prog. run krni hai so we use func^n toh uss samay sirf func^n ko memory allot hogi then woh release hogi after execution)

Disadvantage - Time consumes alot before run time for small fun^c
Sol^n - Small fun^c creating everytime will take everytime space joh like relative hai but time jyada.

To eliminate this we use inline function.

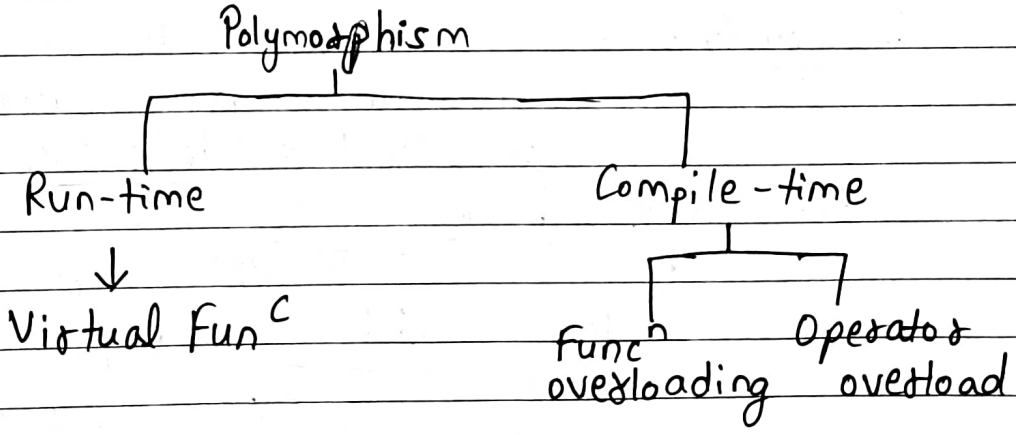
(Compile ke pehle woh separate func^c dikhega but compiler usse expanded line invoke krega yeh hum nhi compiler krega but small func^c hatega its not necessary ki hoga it's not a command)

→ It's a request
→ This benefit reduces as func^n grows in size.
→ So it might ignore request

Exception - func^n with loops, switch, goto
- func^n with recursion
- Func^n containing static variable

(Default Argument are important to ensure that compiler error na ho.)

Function Overloading:



→ Ek se jyada func^c jinke name same ho but job (func^c) alag honge (When a func^n is overloaded with a set of job is called Func^c overload)

PAGE No.
DATE / /

→ Imp point in overloading fun^c is fun^c name same but argument should not be same.

[Note - Agar same hai argument aur return type then kya run hoga? → No, Error (Compile-time)]

Early binding - is imp in fun^c overloading.

First rule - 1. Exact Match (Same fun^c with arg.)
2. No exact match then char → int
Promoted → float → double
3. If no promote, type conversion
primitive type can be converted to each other.

Imp - Primitive type - char, short, int, float, long, double, bool
Derived type - array, pointer
enum → enumeration
user-defined - structure, class, etc

Structure :- 1. used to create data type
2. way to group variables
3. Collec^n of dissimilar elements

Ex- struct book → data type
{
int bookid;
char title[20];
float price;
};

- No memory used as book data type bana jab use krenge variable ke liye tab consume hoga ram

(if before } ; main $func^n$ then global defn) but only can be used when locally declared.

(if written alone no memory used if with variable then 4 bytes)

0 = x = operator ke left side kabhi const. nhi aaskta

For string to char array mein assign then we use func^n

strcpy (b2.title, "C++ easy") ;
↓ array ↓ string

b2.bookid = 101 ;
↓
. ke baad member variable of (b2) for compiler.

⇒ return can only return one value.

→ To avoid misuse of data, we use access specifiers i.e. private, public, protected

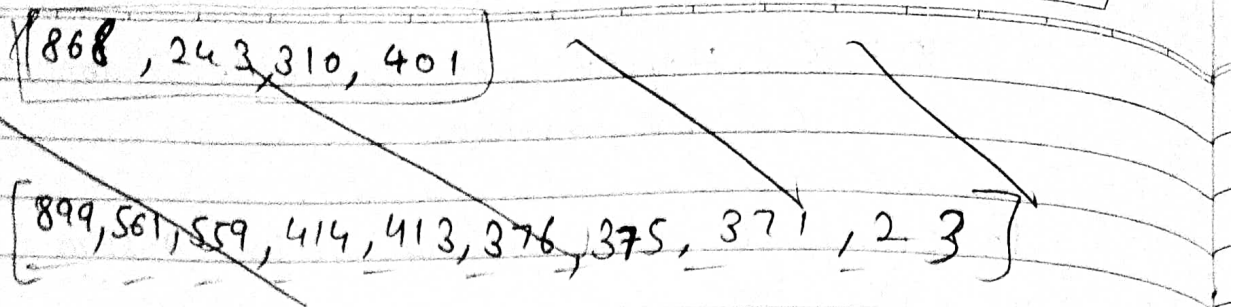
public → structure ke bahr use hoga (members)
private → bahr se accessible nhi hoga

The only difference bet^n structure and class is that,
- the members of class are by default private
- while members of structures are by default public.

→ Class ke under ka bana hua type ka variable is an object which uses memory.
noun. verb

Imp Structure mein func^n bhi declare kar skte

PAGE No.
DATE / /



func bhi member hoga aur apne structure ke baaki logo ko vo directly access krega w/o (b2.bookid)
aur main ka func^n Ka variable '.' use krega.

```
noun.verb();
```

data security - bahr se accesss nhi hona chahiye private krne se data directly specify nhi hoga but use hotha hai

And c++ mein encapsulation ke tehat hume kisi member ko func se call krna chahiye

Lec07

Classes and Objects :-

Difference bet^n class and structure is basically :-

Members of
- structures are default public
- class are by default private

Imp

PAGE No.
DATE / /

Membership Label is important so classname::

bahr declare krke bhi it's a member function
inline keyword

For member func to be defined outside class syntax :-

returntype classname :: funcname (parameter)
{
- - - -
- - -
}

operator ke opetand

Class is an description of object.
→ Obj. is an instance of a class.
↓ example

Class mein a,b → instance of class member
(member variable)

Functions bhi instance member function hote hai
aage will see.

Object ki state uske methods se change honge
↓ matlb
obj. ke member variables ke values ka collection
ek value change hui toh obj ki state will be
altered
Only instance member func se obj. ka state
change krenge

Instance of obj

	member variable	member functions
also said as	attributes, data members, fields, properties	methods, procedures, actions, operations, services.

Lec 08

Static Members :-

Static Local variables - by default initialized to zero
- life throughout the program

static int x → hamesha nhi banega if fun^c gets called

static keyword use hoga

Static Member Variables - (Independent of object)
→ Declared inside the class body
→ Also known as class member variable
→ they must be defined outside the class.
→ They belong to whole class not any object.
→ There will be only one copy of static member variable for the whole class.
→ Any obj. can use the same copy of class variable
→ Class name ke saath use kr skte hai.

float Account :: roi = 3.5 ;

membership (variable) label

Static member function can be called without an object.

→ They are qualified with keyword static.
→ They are also called class member func.
→ they can be invoked with or w/o object.

classname :: funcname (pa) ;

or when object is there

obj. funcname (parameter) ;

Lec09 Constructor :-

→ is a member func of a class.
→ name of constructor is same as the name of class.
→ no return type, so don't use return keyword
→ must be an instance member func, that is, it can't be static.

Syntax - Classname ( ) { ... }

How to call constructor -

→ It's implicitly called whenever object is created. → user don't need to call it.
→ used to solve problem of initialization.

Problem of initialisation → constructor tabhi call hoga jab constructor (obj) banega fir initialise kr skte hai → member variable → obj ki value OOPS → mistake reduce hoga isse → toh sahi meaning mein isne object banadiya

PAGE No.
DATE / / 1)

Part 2 Object banega toh constructor call hi hoga fir, compiler bhi bana skta hai compile-time mein. (without values) [obj ( ) ;]

If you have made constructor, then compiler nhi banayega constructor (ek bhi banaya toh mai nhi banaunga) ⇒ compiler bolega

Default Constructor → made by compiler (no arguments)

parameterised → made by user / me.

Part 3// Compiler has two choices - 1. Default 2. Copy constructor GN

Case 1: Agar koi bhi nhi bana hai toh compiler dono banayega

Case 2: Agar default maine banaya toh compiler copy wala banayega

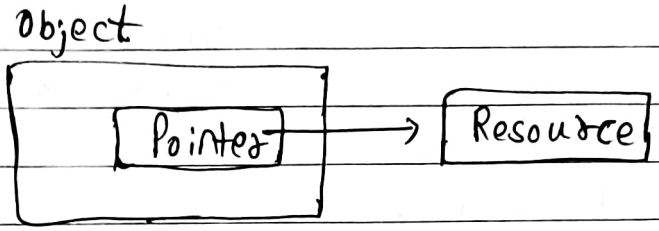
Case 3. Agar dono maine toh compiler bolega "well well well done" !

— x — x —

DESTRUCTOR :-

→ It is an instance member function of a class.
→ The name of the destructor is same as the name of a class but preceded by tilde (~).
→ It can never be static
→ Has no return type
→ Takes no argument (No overloading is possible)

It is defined to release resources allocated to an object.



OPERATOR OVERLOADING :-

Lec 11
Part 1

→ When an operator is overloaded with multiple jobs, it's known as operator overloading.
→ It is a way to implement compile-time polymorphism.

→ Any symbol can be used as func name
  → If it's a valid operator in C language.
  → If it's preceded by "operator" keyword.

→ But we can't overload sizeof, ?: operator.

Binary $c_1 + c_2$ Left wala caller obj hoga
Unary $-c_1$ operand hi caller obj hoga

3

Overloading of unary operator :-

$f(z) = u + iv$
$u_x = v_y$
$u_y = -v_x$
$f'(z) = u_x + iv_x$
$\int f(z)\,dz$
=

No argument means unary.
Overload of ++ operator;
pre increment
Syntax -

```
classname operator++ ()
    { classname var;
      var.x = ++x;   return (var); }
```

post increment -

```
classname operator++ (int)
    { classname var;
      var.x = x++;
      return (var); }
```

Lec 12 Friend function :-

→ Friend func^n is not a member func^n of a class to which it is a friend.
→ It is declared with friend keyword.
→ Defined outside the class to which it is friend.
→ It can access any members of the class to which it is friend.
→ Can't access members directly
→ No caller object
→ Should not be defined with membership label.

obj.var use krenge to make meaningful

PAGE No.
DATE / /

Friend func^n is not a threat to security.

Part 2 Friend func^n is not a member fun^c hence, it does not require's a access specifier.

It can become friend to more than one class.
Hamesha class ke bahr define hoga so, we can't define but declare inside.

Part 3 Agar friend func^n use kr ke krenge overload toh humein ek extra argument lagega.

Part 4 Unary operator as a friend fun^c overloading.

For overloading same as member function we need to declare as friend and then we have to pass earlier caller obj as argument now

Part 5 << - insertion operator
>> - extraction operator

friend ostream& operator << (ostream &, complex);
friend istream& operator >> (istream &, Complex&);

Part 6 class - member func^n to friend of another class
for this use scope resolution operator (::)

PAGE No.
DATE / /

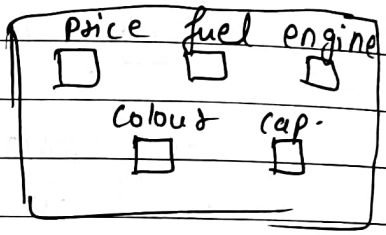
Lec 13
Part 1

INHERITANCE :-

→ Class is used to describe properties and behaviours of an object.
→ It has diff. property names and values.
→ But till now memory doesn't gets alloted we have to declare an object.
→ Obj ke behaviours / actions / method → Class ke function ko hi obj kay

CAR

Properties	Methods
price	set Price ( )
fuel type	set fuel type ( )
engine	set engine ( )
colour	set colour ( )
capacity	set capacity ( )
	get –''–
	get –''–
	–''–

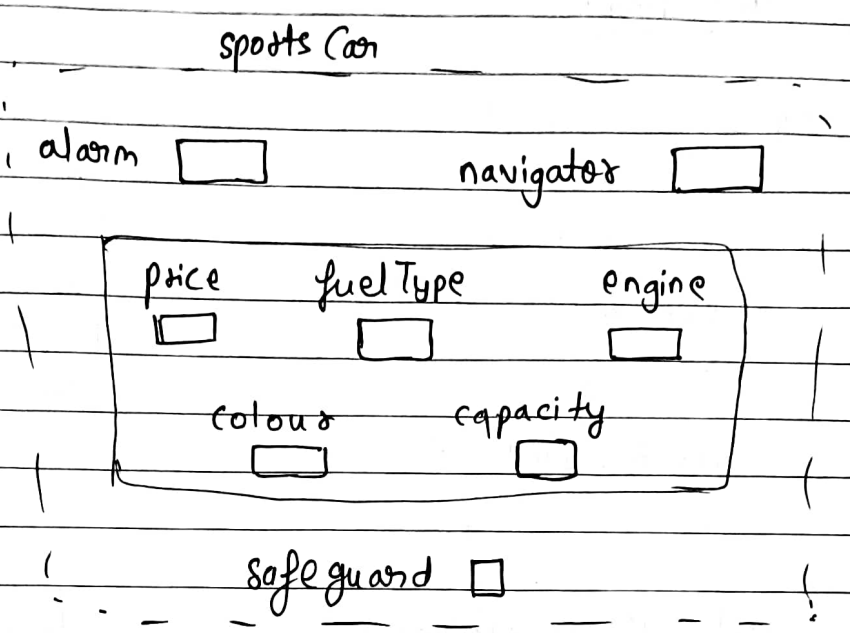


For sports car - we have to declare more methods and properties. If we have to declare for sports then car ka representation kahan hojayega

Second method mein hum dusre class mein naye properties declare kange lekin fir encapsulation rule out hojayega
↓ ek entity ke related info ek hi obj mein rakho

PAGE No.
DATE / /

Third - We have to link the two classes but it's unidirectional like ↑ bottom to up sports car mein dono classes use hoga.



The process of inheriting properties and behaviours of existing class into a new class is Inheritance.

Existing class = old = parent = base class

New class = child = derived class

Syntax :-

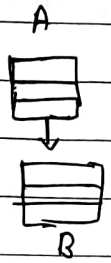
```
class Base_class
{ ------ };

class Derived_class : visibility_mode Base_class
{ ------
  ---
};
```

visibility_mode → private, public or protected

Hamesha derived class banate samy pehla wala use hoga

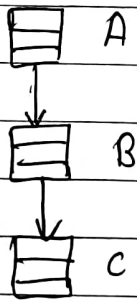
Types of Inheritance -

1) Single Inheritance



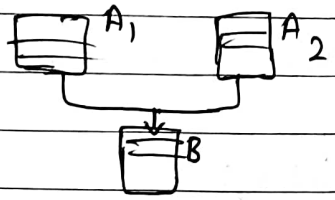

```
Class A
  {... };
class B : public A
  { .... };
```

2) Multilevel Inheritance




```
class A { .. };
Class B : public A
  { .... };
Class C : public B
  { .... };
```

3) Mutiple Inheritance

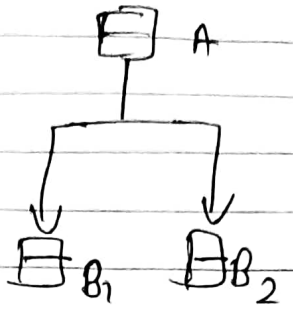



```
Class A1{ };
Class A2{ };
class B : public A1, public A2
    {... };
```

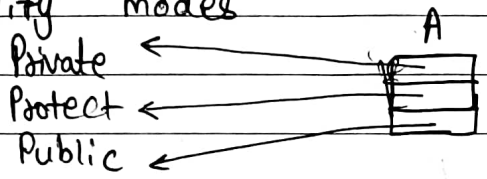
4) Hierarchical Inheritance

Imp

```
Class A
{
};
class B1 : public A
{
};
class B2 : public A
{
};
```



Visibility modes

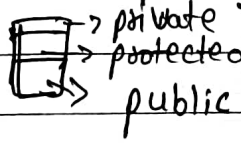


Types of users of a class

- User 1 will create object of your class
- User 2 will derived class from your class

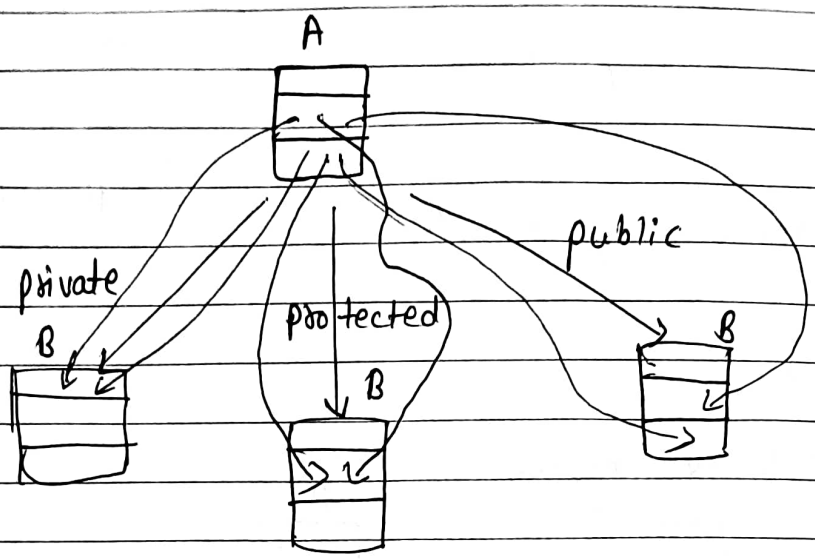
Availability V/s Accessibility

Koi bhi object banane pe memory sabko milegi private variable ko bhi aur public ko bhi but accessible only public hoga so, kabhi agar koi user obj banake call krega toh accessible public hoga but available saare members honge



Protected → User 1 use nhi kr payega
→ User 2 karega access by object

Private → No user can access



Case 1: Class B ke users ko class A ke public, protected, / private nazar aayenge aur woh unhe access nhi kar paayenge.

Lec 13
Part 3

Is-a- relationship

Jaha pe is a sentence hoga waha pe inheritance hoga.

Table is-a furniture.
Student is-a person.
Banana is-a fruit.

Association
→ Aggregation
→ Composition
→ Inheritance

Decide parent class by generalization or derived by specification.

derived class ke object mein saare members honge and parent mein sirf uske members.

Is a relation is always implemented as public inheritance.

Why?

→ If I derive a class from parent class and I want to establish is-a relationship bet^n them then it means the object of derived class can access public members of parent class so, if i use private or protected then, public members of parent class will become available but not accessible so I have to use public keyword only for is-a relationship.

When we will use private or protected inheritance ? →

for example stack if i assign one index value (0, 34)
- and then insert a value using insert func^n by taking public access then it will throw error and hence i have to use private to avoid being inserted a value directly at second index.

Lec 14

Constructor in inheritance :-

Constructor is invoked implicitly when an object is created.
Parent ka cons -> parent ka obj
lekin child mein dono property hai

Child class ka cons, hi dono ko call krega pehle
(call hua)
parent class ka const. chalega and then it came back to himself.
Constructor's working order (In inheritance)
From parent to child
Call order -> child to parent

Child class ka cons. krega parent ko call for argument we need to write own constructor.

Destructor :-

Execution order -> child to parent
Calling order same

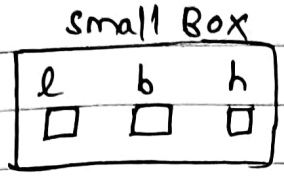
Child Body mein code rhega for parent destructor

Lec 15

THIS POINTER IN C++

PAGE No.
DATE / /

→ A pointer contains address of an objects is called Object Pointer.



→ use hoga for pointers to assign values

whereas, we use . for objects.

Whenever pointer will call a function then, members will be assigned where pointer is assigning.

This pointer → keyword ek block mein scope { };

this is a local object pointer in every instance member function containing address of the caller object.

this pointer can not be modify.

It is used to refer called object in member function.
[Jab tak static keyword add nhi kroge it's a instance member function.]

Calleobj ka address this pointer store krega.

this se nameconflict dusr hogya.

Lec16

PAGE No.
DATE / /

New and Delete in C++

SMA v/s DMA

SMA - Static Memory Allocation (No relation with Static keyword)
DMA - Dynamic memory allocation

Eg -

```
int x;
float y;
Complex c2;
Student s1;
```

} -> Declaration statements

Memory kitni lagi ye compile time pe decide hoga, after compile name ke jagah pe memory hoga as decided in compile time. (as per variable name)

{ var; } -> func khtm variable khtm

⇒ Static memory allocation ke help se banne wale variables ki life jitni hai utni rhegi, when executed we can't destroy both.

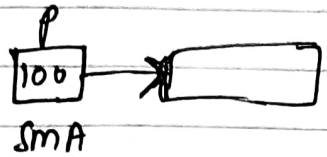
If we know, how much variable then only it's preferred. (No flexibility using SMA)

If variable, (memory nhi pata then we use DMA (At run time, we make variables without informing compiler here, we have new keyword.

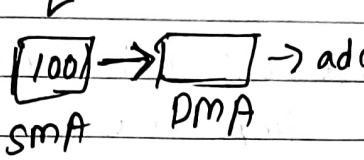
PAGE No.
DATE / /

Syntax ↑ keywood

int *p = new datatype → DMA variable only address
no name



float *q = new float



We can even make arrays using them

float *q = new float [5] (dynamic)

```
int x ;
cin >> x ;
int * p = new int [x];
```

delete (only DMA variable's memory release kare ko use kate bcz if we don't, still they use the space)

delete p; → pointer name → DMA

delete []p; → for arrays

(Dynamically created variables should be stored in pointer (address store)]

DMA variables memory throughout the program.

Lec 17

PAGE No.
DATE / /

METHOD OVERRIDING :-

- Method Overloading
- Method Overriding
- Method Hiding

Method means function.

Func^n same, argument same - Overriding

Func^n same, argument diff - Hiding

Early Binding :- jiska obj hai usko call hoga
object ka type imp hai

For overloading of func^n, func^n ke saare versions ek hi scope or ek hi class mein hone chahiye.

17/01/22

Lec 18

VIRTUAL FUNCTION

→ Base class pointer can point to the object of any of its descendant class.
↳ (Vansaj)

→ But its converse is not true.

For calling func^n using pointer we use [p → func^n] (arrow →)

For late (dynamic) (run-time) binding, we use virtual func^n.

Isse use ↵ krne se pointer ke content ko aadhaar maana jayega and hum ek class mein bhut virtual (pointers) use kar skte hai.

PAGE No.
DATE / /

Part II

If in the class there is one virtual func^n then compiler will declare a variable within the class, it will not be present in child base it will inherit.

*_vptr

Compiler will make static array (pointers ke liye)

Vtable → address stored by vptr.

1	2	3	4

↳ fun^n ka address store hoga

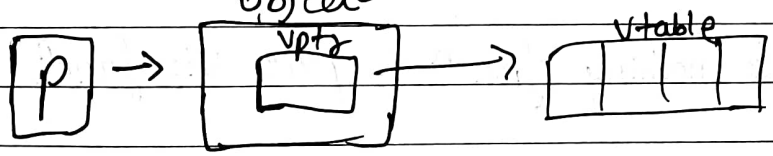
EB → Compile-time (Compiler will decide)
LB → Run-time

In early binding, pointer ka type imp. hai

Eg- A *p , p->f1();

In late binding, we should consider pointer kisko point kr rha hai.

To avoid overriding error, we use virtual func^n



```
while (col <= row)
cout << count << " ";
col++;
count++;
```

count = (3)

row = 3 (4)

3 <= 3

(3) (4) (5)

(5)

```
q <= 1        (2)
(1)
2 3
```

PAGE No.
DATE / /

Lec 19

Part I :-

A do nothing func^n is a pure virtual function.

Syntax- Type func( ) = 0 this will make compiler understand it's do nothing func^n and iski koi def^n nhi hogi outside of inside class.

Rule-1. Iska obj nhi banega (do nothing func^n ka) use child class

Rule-2. Agar aap usi func^n ko obj se call krenge toh aapko overide krna padega for calling via object

virtual do nothing func^n → pure vir. func^n

Abstract Class- If a class has a pure virtual func^n and has no object for instance in method.

If a class has pure vir func^n then, we have to use (base) child class for calling func^n's and use parent properties.

PAGE No.
DATE / /

Two Advantages :- (Pure virtual func^n)

1. Class ko abstract kiya
2. Child class ko pressure dekhe hai ki, ye func^n define krna hi hoga. Parent ke respect mein no meaning.

[> A class containing pure virtual function is an abstract class
> we can't instantiate abstract class.]

lec 20 Part I

TEMPLATES IN C++ :-

→ The keyword template is used to define func^n template and class template.
→ It is a way to make your func^n or class generalize as far as data type is concern.

To avoid func^n overloading and using func^n again and again for diff. values of set variables we use template.

Function template is also known as generic function.

template < class type > type functionname (type arg1, ...)
↳ placeholder (user will decide what to put)

for generalisation, it sees argument's type.
We see that we can create two placeholders at a time.

50x 68x20 188

PAGE No.
DATE / /

Part II

Class Template (also known as Generic class)

template < class type > class class_name { ... };

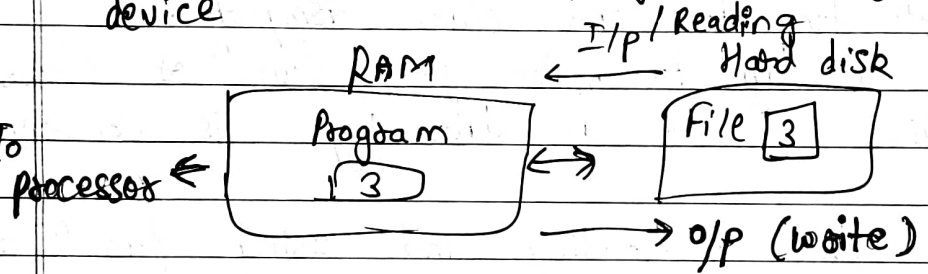
Lec 21 Part I

File Handling in C++ :-

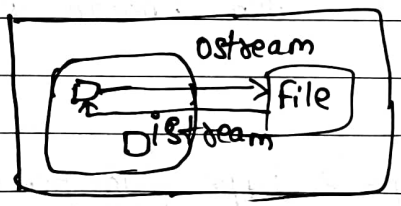
Data Persistence

- Whenever the program ends, memory of ram gets released.

Hence, we have to transfer memory to secondary device

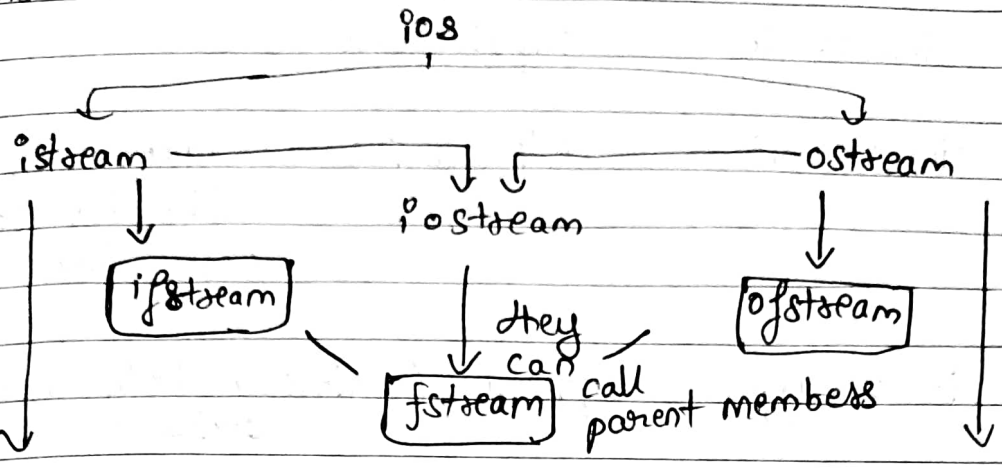


This is known as file Handling.



Variable data to file → output stream (ostream)
file data to variable → input stream (istream)

Streams



cout [ostream ke child class ka obj]
(so he can access ostream membess)

>> → extraction
<< → insertion

To open a file → obj.open("file name");
↳ fout

To close → fout.close();

For reading file data :-
ifstream → obj(fin) ~~obj.~~ fin.open("File");
fin.close();

Part III Opening modes of file :-

fstream can do both reading and writing.

ios :: in input/ read

ios :: out o/p / write (dyaan yeh rahe, toh file ka content erase hoga then new content)

ios :: app append (storing both new and old data)

ios :: ate update (for changing versions of file)

ios :: binary binary

If we open a file obj.open fout.open() then we bring the file from hard disk to RAM.

ofstream → default opening mode → ios :: out

ifstream → default opening mode → ios :: in

fstream →

TEXT MODE V/S BINARY MODE

- Text mode is the default opening mode for binary, pass the argument ios :: binary.

fout << "My name is \n Bhu";

for binary mode → \n (aisa nhi hoga ki new line aaye)

for text mode → it will be treated as new line.

Video 58

tellg and tellp (both are predefined) funcn's

tellg()
→ defined in istream class
→ It's prototype is
return type (int value)
- streampos tellg();
→ returns the posⁿ of the current character in the i/p stream.

tellp()
→ define in ostream class
→ streampos tellp();
→ -11- in o/p stream.

seek → shift

PAGE No.
DATE / /

seekg and seekp in file handling :-

seekg → Defined in istream class.

It's prototype is

- istream & seekg (streampos pos);
- istream & seekg (stream off off, ios_base :: seekdir way);

→ pos is new absolute position within the stream (relative to beginning)

tellg ek index aage ka read krega kya mai waapis se index pe aa skta hu in betn

way values ios_base :: beg, cur, end

seekp → ostream (for writing)

```
ofstream fout
fout << tellp.get(); fout.tellp();
fout.close();
```

PAGE No.
DATE / /

LEC 22 - INITIALIZERS

→ Initializer List is used to initialize data members of a class.
→ The list of members to be initialized is indicated with constructor as a comma separated list followed by a colon.

Const mtlb jo value aayega woh change nhi hoga

Class mein variable ko initialise nhi krenge bas declare krenge
```
const int x;
```

For initialising this, we use initialiser list:

```
class name () : var(value);
```

If we make reference variable
```
int &y;
class name (int&n) : var(value), y(n)
```

```
void main()
{ int m
  Dummy (Class name)
  d1(m);
```

There are condn's where initialization of data members inside constructor doesn't work and Initializer List must be used

→ For initialization of non-static const. data members.
→ For initialisation of reference members.

Lec 23

Deep Copy and Shallow Copy //

How to create a copy of object?

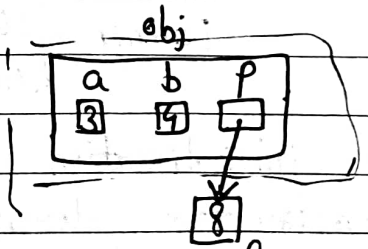
Same class 2nd obj mein 1st ka data copy

- Copy constructor
- Copy Assignment operator (Implicit)

> Creating copy of object by copying data of all member variable as it is. → Shallow Copy

> Deep Copy

Creating an object by copying data of another object along with the values of memory resources resides outside the obj but handled by that object.



obj ke bahr hai ek value

But in mere future, we create another copy of object then bahr ka nhi aayega

By default Shallow Copy but bcz of address in pointer it gets same data

* kyne se woh use memory block ko represent krega joh woh store krega.

If shallow copy was done koi ek object delete hoga then loc^{n} pe data delete hoga and second obj ke pointer ko invalid memory milti.

Lec 24

Primitive to Class type

int, char, float, double are primitive types.

class type is any class you defined.

Examples of auto type conversion

```
int x = 4;
float y;
y=x;
```

O/p = 4.0

```
int y;
float x = 3.4;
y=x;
```

O/p - 3

Total 4 combination → Primitive to Class type
Class type to primitive type
Class type to class type

Class type → defined in program

Lec 24 Part I

Primitive to class type :-

It can be implemented through Constructor.

II Class type to primitive type :-

It can be implemented with casting operator.

Syntax:

```
operator type ()
{
    ....
    return( type-data );
}
```

Lec 26 One class type to another class type

→ Conversion through constructor.
→ Conversion through casting operator.

Constructor assign ke left side waale object, ke block ya class mein banega

Lec 27: EXCEPTION HANDLING

→ Exception is any abnormal behaviour which is run-time error.
→ Exceptions are off beat situation in your program where your program should be ready to handle it with appropriate response.

→ C++ provides a built-in error handling mechanism that is called exception handling.
→ Using exception handling, you can more easily manage and respond to runtime errors.

try, catch, throw

Program statements that you want to monitor for exceptions are contained in a try block.

If any exception occurs within try block, it is thrown (using throw).

The exception is caught, using catch, and processed.

```
try {
  } catch (type1 arg) {
}
  catch (type 2 arg) {
}
  ....
      catch (type N arg) {
}
```

PAGE No.
DATE / /

- When an exception is caught, arg will receive its value.
- If you don't need access to the exception itself, specify only type in the catch clause - arg is optional.
- Any type of data can be caught, including classes that you create.

General form of the throw statement is :

```
throw exception ;
```

Throw → It must be executed either within the try block proper or from any func^n that the code within the block calls.

Lec 28 **Dynamic Constructor**

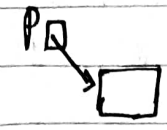
Whenever object is created, automatically constructor is called.
Imp work of constructor is object ko object banao.
(Eg - Bade kehte hai insaan bano)

Inko obj. tab kahenge when they will represent any data

Dynamic → Constructor can allocate dynamically created memory to the object.
Thus, object is going to use memory region, which is dynamically created by constructor.

object → Constructor

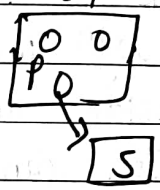
car * p = new car ;



new ki madad se obj banega then normal constructor

Constructor joh aisa space bana raha jisko object ke members dynamically point kar the

eg → A()
{ p = new int ;
*p = 5;
}



Dynamic constructor, obj ke through hone wala access memory ko create krta hai.

LEC 29 NAMESPACES :-

Header files mein func^n, variables, class ke decla^n hote hai

Defn in library files (Defn = Coding {.... })

.h
declarations kiye hue group ko namespace kehte hai
{ = }
namespace Myspace
{ }

PAGE No.
DATE / /

Namespace is a container for identifiers.

It puts the names of its members in a distinct space so that they don't conflict with the names in other namespaces or global namespace.

Syntax:- namespace Myjagah { → keyword scope is global
. . . . . }

→ It's defn doesn't terminates with a semicolon like in class defn.
→ defn must be done at global scope, or nested inside another namespace.
→ You can use an alias name for your namespace name, for ease of use. (usf Bukchod)

namespace ms = Myspace;

As it's not a class hence, doesn't has any instance (objects).

[Unnamed namespace can be used too]

Namespaces can be extended →
It can be continued and extended over multiple files, they are not redefined or overridden.

File 1.h
```
namespace my
{ void f1();
}
```

File 2.h
```
namespace my
{ void f2();
}
```

PAGE NO.
DATE / /

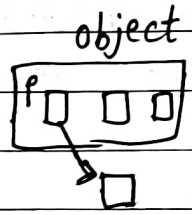
PART 2 → NAMESPACES IN C++

Accessing members of namespace :-

using keyword allows you to import an entire namespace into your program with a global scope.

It can be used to import a namespace into another namespace or any program.

LEC30 VIRTUAL DESTRUCTOR :-



if object gets destroyed then if there's a pointer then we use release memory option.

int *p ; New -
delete -

Base pointer can point to the object of derived class.

int *p = new int p int address banega
↓ □ → □ (dynamic memory)
type

Student ptr = new Student ;

By delete ptr
ptr □ → ☒ (destroy)

virtual ~ classname (){ }

Hence, virtual keyword late binding hoga and pointer ka member content pe

action hoga.
Isliye pehle derived class ke pointer se hote hue pointer ke base mein jaake memory data release krega.

LEC 31 NESTED CLASS IN C++

→ Class inside a class is called nested class.
→ A nested class is a member and a such has the same access rights as any other member.
→ The members of an enclosing class have no special access to members of a nested class; the usual access rules shall be obeyed.

How to make objects of nested class →
For inside the enclosing class it can be within private of enclosed (Syn) Nested class-name obj;

For outside, the nested class should be made public first and then can be made as Syntax →
Enclosed class :: Nested class obj;

Role of access modifier for making use of outside the class.

STL IN C++

STANDARD TEMPLATE LIBRARY

as of any other prog. lang

collecn of predefined classes or objects.

C++ by SShukla →

- [x] Introduction
- [x] STL Containers
- [x] Array in STL
- [ ] Pair in STL
- [ ] Tuple in STL
- [ ] Vector class in STL
- [ ] List class in STL
- [ ] Map class in STL
- [ ] String class in C++ ① and ②

Lec 32 → Introduction to STL

→ powerful set of C++ template classes

→ At the core of the C++ Standard Template Library are following three- well structured components
- Containers
- Algorithms
- Iterators

↳ Containers are used to manage collections of objects of a certain kind.

→ Containers Library in STL provide containers that are used to create data structures like arrays, linked list, trees, etc.

→ These container are generic, they can hold elements of any data types.

Example

vector can be used for creating dynamic arrays of char, integer, float and other types.

Diff^n bet^n dynamic and normal array is we can increase size of dynamic array.

Store → containers
Fiction → algorithms

Algorithms -

→ Algorithms act on containers. They provide the means by which you will perform initialization, sorting, searching, and transforming of the contents of containers.

→ Algorithms library contains built in functions that performs complex algorithms on the data structures.

- One can reverse a range with reverse() func^n, sort a range with sort() func^n, search in a range with binary-search() and so on.

- Algorithm library provides abstraction, i.e. you don't necessarily need to know how the algorithm works.

Iterators :-

→ Iterators are used to step through the elements of collections of objects. These collections may be containers or subsets of containers.

→ Iterators in STL are used to point to the containers.

→ Iterators actually acts as a bridge betⁿ containers and algorithms.

Example :-

→ sort() algorithm have two parameters, starting iterator and ending iterator, now sort() compare the elements pointed by each of these iterators and arrange them in sorted order, thus it does not matter what is the type of the container and same sort() can be used on diff. types of containers.

PAGE No.
DATE / /

push and pop

Lec33 STL Containers →

→ Containes library is a collecⁿ of classes.

→ The Containers are implemented as generic class templates.

→ Containes help us to implement and replicate simple and complex data structures very easily like arrays, list, trees, associative arrays and many more.

→ They can be used to hold different kind of objects. (int, char, float, etc.)

(Template are imp)

Common Containers :-

1. vector : replicates arrays
2. queue : -11- queues
3. stack : -11- stack
4. priority_queue : -11- heaps
5. list : -11- linked list
6. set : -11- trees
7. map : -11- associative arrays

replicate → to copy something exactly

Sequence Containes (data sequence mein store hoga) - like array, linked list, etc.

PAGE No.
DATE / /

Associative Containers
→ sorted data struc. like map, set, etc.

Unordered associative Containers
→ unsorted data structures

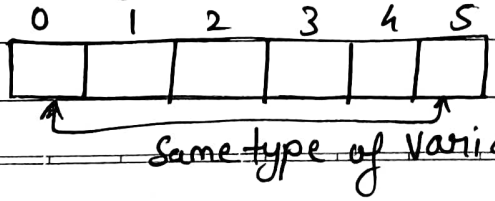
Containers Adapters
→ interfaces to sequence containers

If we use list containers to implement linked list we just have to include the list header file and use list constructor to initialize the list.

```
#include <iostream>
#include <list>
int main()
{
    list<int> mylist;
}
```
↳ type (it can be float, user-defined), etc.

LEC 34 ARRAY in STL

1. Array is a linear collec^n of similar elements.
2. Array in STL provides us the implementation of static array.
3. Use header array
→ #include <array>

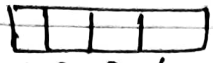


PAGE No.
DATE / /

Syntax :-

array < int, 4 > obj;

type of element ↵ ↳ size of array



int variables

Creating array objects :-

Syntax :- array < object_type, array_size > array_name;

It creates an empty array of object_type with max size of array_size.

Member functions :- (of array template)

1. at()

This func^n (method) returns value in the array at the given range

If the given range is greater than the array size, out_of_range exception is thrown.

2. front and back()

front() method returns the first element in the array.

back() method returns the last element in the array.

PAGE No.
DATE / /

3. fill()
This, func^n assigns the given value to every element of the array.

4. Swap()
This func^n swaps the content of two arrays of same type and same size.

5. size()
This func^n returns the number of element present in the array.

6. begin and end()
begin() func^n returns the iterator pointing to the first element of the array

end() func^n returns an iterator pointing to an element next to the last element in the array.

For displaying an array data of index 5

```
for (int i=0 , i<5, i++ )
  { cout << array[i];
  }
```

For displaying size of an array

```
name_array.size();
```

PAGE No.
DATE / /

LEC 35 Pair in STL in C++

→ Pair is a template class in Standard Template Library.

→ Pair is not a part of container.

Syntax of using pair object

```
pair <T1, T2> pair1;
```

Example :

```
pair <String, int> p1;
```

- pair ← template keyword
- String ← data type
- int ← data type
- p1 ← object

Inserting Value via pair :-

```
p1 = make_pair("type1 value", 'type2 value');
```

Accessing pair members :-

```
cout << p1.first << endl;
cout << p1.second << endl;
```

↳ main thing here we need to take care of user-defined data types.

PAGE No.
DATE / /

LEC 36 Tuple in STL in C++ (we have to include a header file #include<tuple>

Just like in pair, we can pair two heterogeneous objects, in tuple we can pair multiple objects.

Syntax of using Tuple object

```
tuple < T1, T2, T3 > tuple 1;
```

Example:

```
tuple < string, int, int > t1;
```

Inserting value

```
t1 = make_tuple ("India", 16, 10);
```

To print it

```
cout << get < pos^n no > (obj as argument);
```

We can compare a pair or tuple by

== != < > <= >= these operators.

LEC37 Vector Class in STL in C++ :-

The most general purpose container is the Vector.

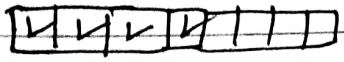
It supports a Dynamic Array.

What is Dynamic Array ?

Continue →

Initially, when array is declared it has its own capacity suppose it has 1, the vector will have 1 but if we enter second value then capacity of vector doubles if 3 then now it's capacity will be 4.
Jab capacity khtm hogi tab capacity will be doubled

capacity = 1 2 4 8



Hence, it's a dynamic array.

→ Array size is fixed, no flexibility to grow or shrink during execution.
→ Vector can provide memory flexibility.

Vector size
Vector being a dynamic array, doesn't needs size during declaration.

Code below will create a blank vector
vector <int> v1; (We must include #include <vector>)

Initialize during declaration :-
vector <int> v1 = {10, 20, 30};
if <string> v1 = {"Bhavesh", "Upadhyay"}

More declaration -
vector <int> v1; → zero length vector → capacity is zero.

PAGE No.
DATE / /

vector < char > CV(5);
- Creates a 5 element char vector

vector <char> CV(4, 'a');
- initializes 4 element char vector with 'a'
↳ Eg → vector <string> V(3, "bru");

Relational operators
== > >=
!= < <=

0	1	2
bru	bru	bru

→ Subscript Operator [ ]
↳ it's also defined for vector.
Basically for printing value of vector array.

vector V1 [i] for (int i = 0; i < capacity ; i++)

→ Push_back ()
push_back is a member function, which can be used to add value to the vector at the end.
Example ⇒ V1. push_back (10);

→ pop_back()
It removes the last element.

for finding capacity → obj (V1. capacity ();)

Even after popping last element the if it's not a multiple of 2 the capacity remains same but values are getting removed.

while (1)
{ coding == life;
}

PAGE No. 04-02-22
DATE 02/:/21

→ size()
Returns the number of elements currently in the vector.

→ clear()
Removes all elements from the vector.

→ at()
this method works same in case of vector as it works for array.
returns the element at i^{th} index in the vector vector_name.

→ front and back()
↳ returns the first element of the vector.
↳ back() returns the last element of the vector.

→ For inserting in bet^n

how much value

vl.insert(it + index, value);

it can be 0, 1, 2

PAGE No.
DATE / /

→ LEC 38 LIST CLASS IN STL (To use it header file #include < list >)

- List class supports a bidirectional, linear list.
- Vector supports random access but a list can be accessed sequentially only.
- List can be accessed front to back or back to front.

For printing data of an list -

we have to use an iterator

```
list < data type > :: iterator obj = Obj of list.begin();
while ( obj != Obj list.end() )
{
    cout << *p;
    p++;
}
```

Useful functions of list class :-

1] size() obj of list.size();

2] push_back() → piche milta and daalna
push_front() → (aage)

3] pop_front() remove krna aage
pop_back() piche

4] sort()
→ Print in ascending order

5] reverse()
→ ulta kardega

6] remove() specific
→ remove specific value (44);
pass this

7] clear()
→ clear (saaf) sab kuch

Calling
obj of list . func();

LEC 39 Map Class :-

→ replace with similar object

#include <map>

→ Maps are used to replicate associative arrays.

Numeric v/s Associative

int a[5];
float a[5];

When we use other than default number to represent array's index it's called Associative

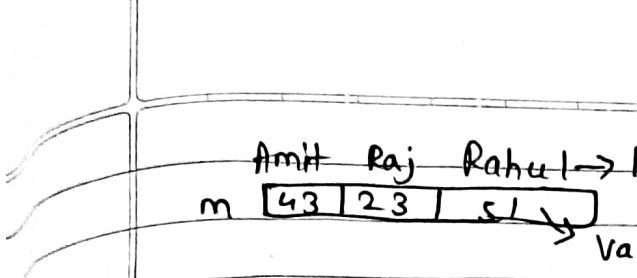
Index →	0	1	2	3	4

A	B	C	D	E
1	2	3	4	5

m["Rahul"]

In arrays, where index is represented by numbers are Numeric arrays.

→ Maps contain sorted key-value pair in which each key is unique and cannot be changed, and it can be inserted or deleted but cannot be altered.



key - value
Raj - 23
Amit - 43

⇒ Value associated with keys can be altered.

Map property :-

maps always arrange its keys in sorted order.

In case the keys are of string type, they are sorted in dictionary order.

(Ascending Order) → sort();

Useful functions :-
at() size() insert() clear() empty() []
↳ if 0 then not empty
1 then empty

⇒ insert()
se assign hoga and print kar skte hai

⇒ Before using strcpy in vscode
#include <cstring>

PAGE No.
DATE / /

LEC 40 String Class in C++ :-

Using null-terminated character arrays are not technically data types.
So, C++ operators can't be applied to them.

The string class is a specialization of a more general template class called basic-string.

Since, defining a class in C++ is creating a new data type, string is derived data type.

This means operators can be overloaded for the class.

String is safe than char array

Careless programmer can overrun the end of an array that holds a null terminated string.

For example, using strcpy()

String class handles such issues.

String is an another container class.

To use string class, #include <string>

It supports constructor such as
- string()
- string (constant char *str)
- string (const string & str)

PAGE No.
DATE / /

Concatenation (Append) (+)

Assignment

Last lecture by C++ God !!

Lec 40 Part 2 - String Class

Insertion and extraction operator :-

<< Insertion (for output)
>> Extraction (for input)

Mixed operations :-

You can mix string objects with another string object or C style string.

C++ string can also be concatenated with character constant.

concatenation ke samay (do type) one side pe dusra type reh skta hai

Useful method (func^n)

1. assign func^n Sx->[obj. assign("arg");]

2. append func^n Sx->[obj. append("bau");]
extra likna

3. insert() specific index number pe value (assign) daalna

4. replace()
obj.replace(index no, length, "cheez value");
Eg- s1.replace(2, 2, "A") to replace
if s1 is hello
then o/p → heAo
Cause 2nd index se uske inclusion ke aage ke 2 values replaced.

5. erase() sab kuch gayab
obj.erase();

6. find() → aage se index dhundo
int i = obj.find("");
if finds then o/p → index no
otherwise zero

7. rfind() → piche se index dhundo
int i = obj.rfind("");

8. compare()
dictionary order → 0 same
→ -1 Aman (s1) Amit (s2) s1.compare(s2)
not dic. → 1 Amit (s2) Aman (s1)

9. c_str()

isse string ko char array mein transfer

string obj = ""; char str[10];

Syntax :- strcpy (str, s1.c_str());

10. size()

obj.size();