OOP (2)

(Abs/Rel path)?

endl → combination of → '\n' / flush

Bad performance dont use flush:

Data store in buffer flush.

(..) current Folder ka parent

ifs.eof
ifs.good

while (!ifs.eof())

char msg[80];
ifs.getline(msg, 80)

if (ifs.eof())
break;

(cin.ignore)

getline(msg, 80, '\n')

Student S {12, "Ali", 3.4}

struct Student {
int roll;
char name[50];
float marks;
};

ifstream ifs;
ifs.open("text.txt", ios::in)

ios::out)

ios::in | ios::out

Both reading writting both ← File

ios::in | ios::app

(3)

10, —— 20, . . — 30,

10 20 30

```
ifs >> s.roll;
ifs.ignore();
ifs.getline(name, 5);
ifs >> s.marks
```

111 | |
221 | |

12 fail bit()
8 badbit() eof() good()

EOF → -1   eof → when read EOF open.

ff = 11111 → Trues compliment = -1

Stream extraction operator >>

اگر >> use کریں تو اگر end کو eof پتا

تو آخری read کرے گا پرنٹ کر دیتا ہے

ABC → ifs >> c;

cout << c; → ABC

```
while (ifs >> c)
{
  cout << c;
}
```

ate → at end

trunc → trunc

ifstream ifs;

ifs.open(" " ; ios::in | ios::ate)

This Dont move pointer :-

ifs.peek();

(char)ifs.peek();

10, — 20, . . — 30,
10 20 30

(3)

ifs >> s.roll;
ifs.ignore();
isf.getline(name, 5)
ifs >> s.marks;

111 | |
221 | |
| |

12 failbit()
8 badbit() eof() good()

EOF → -1 eof → when read EOF

ff = 11111 → Twos compliment = -1
Stream extraction operator →>>
اگر eof کو end کر کے use >> اگر
تو آخری read کیا پرنٹ کرے گا
ABC → ifs >> c;
cout << c; → ABC

while (ifs >> c)
{
cout << c;
}

ate → at end
trunc → trunc
ifstream ifs;
ifs.open("", ios::in || ios::ate)

This Dont move pointer :-
↓ ifs.peek();
(char) ifs.peek();

\r → carriage return
\n → linefeed

'26' nexed. it

(4)

ifs.tellg() → tells index of pointer in file

ifs.seekg(0) → pointer returns to zero index.

ifs.seekg(2+ifs.tellg()) → beginning

ifs.seekg(-2, ios::end) | ios::beg

2 → point 2 of end

(2, ios::cur) current of pointer

ifs.seekg(0, ios::end)
cout << ifs.tellg();

c='26'

ABCD

AB

eof()
~~good()~~
stream.clear();

26 → 65 → A

windows does not behave well for 26. It treats it as a EOF e.g. whenever 26 in char is hit the compiler will on eof(). but

AB CD

ch=26; AB 26 CD

while(ifs>>ch) → output AB

{cout << ch;

}

remove(__); delete file

ifs>> rollno;
ifs.ignore()
ifs>> name; ifs.getline(name, size)

ifs>>gpa

Lecture 3:- OOP

preprocessor director # define

file obj
↓
Linker → " load into memory
↓
ذمہ داری۔ جب ہم call func
کرتے ہیں تو func میں کونسی
body call ہو گی decide linker کرتا

Function Binding:
ایک func کے against call func کو لنک کرنا

```
#define abc 100
int main( )
{
    cout << abc;   → output: 100
    return 0;
}
```

#include "myMath.h"

linker all files کو جوڑتا ہے

Header file will not be part of code before include.

(6)

Linker always find in cpp file

so, we have to write prototype in header file and body in cpp files.

Reason

~~Header file~~ If you are using headerfiles in multiple files And Headerfiles includes prototype and body It through error due to repetation of code.

Header file → only prototype
Name same of cpp file
Body here

→ project 4

Readerfile

ifndef xyz → file name including extention
define xyz

Prototypes

end if

: MY_MATH.H
my_Header → Header file

#ifndef MY_SETAPP
define MY_SETAPP

in cpp

#include "myHeader.h"

→ Prototypes

#endif

Storage Class

register
auto
extern
Static → within 1 file

Excess variable from another cpp file

Where Excess:- extern int b
↳ excess value from that file

Excess only in file
→ Static int a;

Static void g();

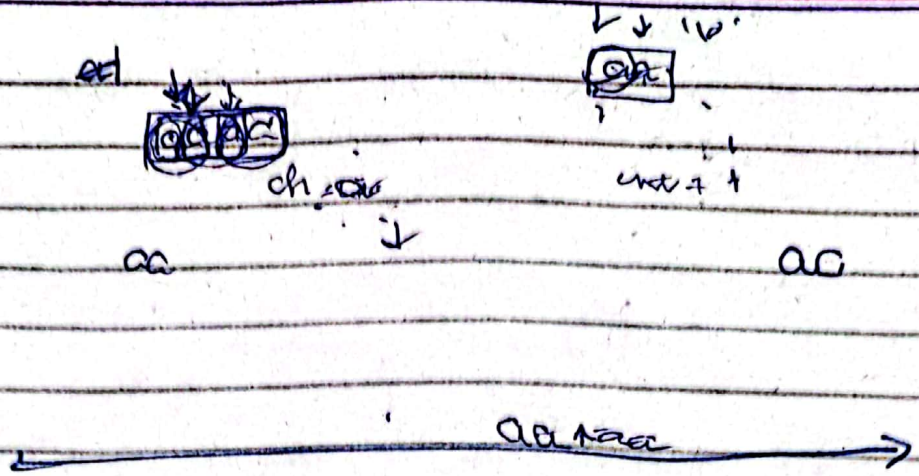
```
void f()
{
  Static int k=80;
}
```

Lifetime ∞ ∴ program ∝ f

Local variable: Declare in body of function
↳ Accessed from that function only
↳ Lifetime of function tell.

Global variable :-
↳ External to function declared
↳ ~~difference~~ Lifetime till termination of function
↳ Access anywhere.

8



Static Variable :- Declare with in function.
↳ can be accessed from that function only.
↳ Lifetime till program terminate.
↳ Keyword "static" for declaration

```
void add( )
{
  static int num
        num++
}       num = 10
```

whenever function call num value is increment

1st num = 10	remain in memory after function closes
2nd num = 11	
3rd num = 12	
4th num = 13	

Lecture 4 - Object Orientation

- Abstraction → Types (2) → functionality, (1) Data
- Encapsulation
- Information Hiding
- Aggregation | Composition | Association
- Inheritance
- Polymorphism. ایک obj کی diff Types ہو

Object → Attributes, functionalities

private:
اس سے نیچے جو بھی function کے میں access ہو
نہیں ہو سکتا دوسرے funct میں

Public:
Accessor ← Getter | Setter → mutator

```
void setNumerator (int n)   → mutator
{
  numerator = n;
}
```

for denominator

```
if (d != 0)
  denom = d;
```

```
int getNum() → Accessor
{
  return numerator;
}
```

Predicate function: returns boolean

13

Always make data members private.

In Class everything in it is private by default.

In struct by default everything is public

ADT → abstract data Type

Object : set of attributes having related values.

State: difference of attributes.

constructor → initialization
→ resource Allocation.

```
class Time {
    int hour;
    int min;
    int sec;
    Time() {
        hour = 10;
        min = 20;
        sec = 50;
        cout << "default";
    }
    Time(int h, int m, int s) {
        hour = h
        sec = s
        min = m
        cout << "custom";
    }
```

default constructor

This function is called when ever an object is created

Time t1, t2 (10, 20, 40)

output : default custom

here t1 default constructor is called and t2 custom.

OOP (11)

If we remove Time() empty default constru is not present so th will get error.

Language make one default constructor if we make custom construtor dey constructor is not made.

make a reset values function and call it in constructor so that values can be reseted bar bar.

```
Time() {
    resetValues();
}

void resetValues()
{
    hour = 10;
    min = 20;
    sec = 50;
}
```

```
void resetValues(int h, int m, int s) {
    if (h >= 0 && h <= 23)
    {
        hour = h;
    }
    else {
        hour = 13;
    }
}

Time () {
    reset values(13, 20, 45);
}

Time(int h, int m, int s) {
    resetValues(h, m, s);
}
```

or give default values to arguments of function

Functions

isValidHour, min, sec

reset values( )

Constructor Delegation

```
Time(): Time(13, 20, 19) {
}
```

↓ default

↓ custom

جب ایسے کریں گے تو ایک default obj constructor again custom constructor کو call کرے گا۔

(13)

multiple delegation

```
Time(): Time(13, 10, 19): Time(Hello)
{
    cout << "default";
}
```

اُلٹ ڈیلی گیٹ سے نیچے تک execute bodies ہوتی ہیں

State

جب direct initialize کریں تو constructor کی
body بعد میں initialize کرتے

```
Time t1;
Time t2();
```
→ This is function prototype

```
Time t1{};
Time t2{3, 40, 51};
```

```
Time t1, t2;
```
t1, t2 object has only attributes Hour, minute

Write only prototype in class.

```
void Time :: setHour( ) {
}
```

Scanned with CamScanner

(14)

enumeration : 0, 1, 2, 3

```
enum Month = { JAN, FEB, MAR, APRIL.... }
int daysInMonth[12] = { 31, 28, 30, 31.... }

daysInMonth[MAR]
dayInMonth[Month :: MAR];

enum GameStatus = {WIN, DRAW, IN_PROCESS};

GameStatus check(){
        return WIN;
}
```

15

int cell ≡ por - '0'

Destructor → resource deallocation

Arrayj(int 20;

int &getSet(int);

resize set

if (index >= 0 && index < size)

{ return data(index)

exit(0); → return 0 like

↔

Destructor → parameter less
no return Type

~Array();

Array::~Array(){

if(!data)

return

delete [] data

data = nullptr;

size = 0;

}

Constructor:

Making: a b c

deleting destructor: c, b, a

سیدھے سے بنتے ہیں الٹے ختم ہوتے ہیں

```
f() {
    a(5)
}
Array a(3), b(4);
f();
d(6) ✓
```

c	3
c	4
c	5
d	5
c	6
d	4
d	3

```
Array a{3}
a.~Array()
a.~Array()
```

→ c 3, d 3, d 0, d 0

↳ destructor 3 bar called
خود ایک ہم نے دفعہ 2

جب بھی destructor call کروں تو class میں

```
resize(int s)
    if (s <= 0)
    {
        this -> ~Array();
        return;
    }
    resize logic
    this -> ~Array();
    data = temp;
    size = newSize;
```

(19)

Destructor کو بار بار call کرنے کا مسئلہ

```
Array a{5};
Array *p = &a
Array &x = a
```

x. —

p -> —

(*)p. —

Member Function کو ایک اور طریقے سے parameter

```
// Backend
t1.setHour(12);
```

```
setHour(&t1, 12)
void setHour(Time * this, h) {
    this -> hour = h;
}
```

```
Array:: resizeSet( ) {
    Array x(20);
    getSet(1);
```

getSet کو call کرنے کے بعد resize کو call کرنا

```
    x.getSet();
```

copied

String Class trimLeft()

- This pointer
- inline function
→ Pass object by value → copy constructor

constructor ... object ...

```
inline int Set::getSet();
inline int Array::getSet( );
```

Inline ... list?

variable no. of arguments:-

```
void f(...) {

}
f( 1,2,3,4,5);
```

#include <stdarg.h>

```
void ff(int cnt, ... ) {

va_list list;
va_start (list, count);
int value = va_arg (list, int);
int value
for(int i=0; i<=cnt; i++) {

value = va_arg (list, int);
cout << value << endl;
}
va_end (list);
```

Array (int = 0, ...);

va_start (list, s) میں pass کرتے ہیں parameter
یہاں بھی لکھیں گے

String S1 = 345 ; ✓
String S2 {345} ; ✓

Same ہوں گے { String S3 = S1 ; ✓ } copy constructor
String S4 {S1} ; ✓ }

Every class has a copy constructor by default that cop recieves same Type object

```
Time ( Time &x) {        copy constructor
hour = x.hour;
min = x.min;
sec = x.sec;
}
```

default copy constructor:
- member wise copy
- bitwise copy
- Shallow copy → copies address

↓
error دے گا Runtime پر کیونکہ data کے address

Deep Copy

Making copy constructor

```
String ( const String & s) {

    size = s.size;
    data = new Char [size] ;
    Strcopy (data, ref.data); }
```

S3=S1 Shallow Copy

↳ Dont Assign for classes that have data in Shell.

```
string f( ) {
    string n;
    return n
}
main ( )
{
string y = f ( );
return 0;
}
```

Constructor / destructor

OOP

Student
{
const int RollNo
float GPA

const data Member
Const member function
const object

ClassName * const this

~~return this;~~

Here last stage you can change const data member initial

constructor → member initializer list

```
String(): member
{

}
```

~~header~~ | definition پر ہوتا ہے
Member Initializer list

```
Student( ): rollNo(32), b(91)
{

}
```

member data کو پہلے ہی initialize کر دیتا ہے

```
Student (int rno, int v) : rollNo(rno), b(v)
{

}
```

also adding delegate constructor

```
Student (int rno): rollNo(rno), Student( ),
}                                    GPA(13)
```

کیا اگر کیا ہے call تو نہیں delegate کو initialize اور واپس سے delegate کر رہا ہے کیوں ... body ... constructor

Test

{

const int a, b;

}

Test (int x, int y) : b(x), a(b)

{

}

const className * this

a.getSet(1)

getSet(&a, 1);

const Array *

Header file Make const

int getSize() const;

const obj

int String::getSize() const

{

return size;

}

const

const ClassName * const this

Non-const object can call const member function.

اگر ایک non const یا const funct

تو کس کو precedence non کو ملتی ہے۔

int & getset(int);

const int & getset (int) const;

cout << a.getset();

a.getset() = 87; X error

---

int * data

In const member function we cannot change the array but can change data of array

data[2] = 845; ✓

~~int *~~

for const function → that it make.

const int * data;

for not changing

int * const data;

```
class student {
    const int rollNo;
    char name[30];
    float GPA;
}
```

Copy Constructor

destructor کا کام: اگر کوئی capture resources کیں ہیں تو ان کو free کرے گا۔

(24)

Student (const Student &ref): rollNo (ref.RollNo)
{
name =
GPA = ref.GPA;
}

Make in Heap

Array *p = new Array {5, 1, 2, 3}
p → ~Array();
Heap Lifetime we call manually.
Necessary use:
delete p;

اگر Heap پر obj بنا ہو اس کا destructor خود نہیں چلتا۔

Heap پر بنا ہوا obj nameless.

Array {5, 10, 20, 30}

یہ بھی nameless
اس کا control صرف اس لائن تک ہے، اس کے بعد ختم۔

Array x = Array {5, 10, 20, 30};

یہ مرنے جا رہا ہے تو اسی کا نام x رکھ دو
const copy اور ← call کرے گا۔

(25)

```
void f(int &n)
{}

f(34) X error

void f(const int &n)
{}
f(34)
```

یہ temporary بن گیا ہے

جب temporary کو receive کرنا ہو تو اس کو const by reference کے ساتھ alias کو ignore کرنا ہے

```
int &r = 345;   X error
int && r = 345;  ✓
```

یہ صرف temporary کو receive کرے گا

```
Array (const Array &);
Array (Array &&);
```

اگر ڈیٹا صرف پاس کرنا ہے تو یہ چلے گا اور اگر ڈیٹا بدل کے دوسرا رکھوانا ہے

```
Array (Array &&);

Array (const Array &)
```

یہ temp ڈیٹا receive نہیں کر سکے گا

```
Array s = Array {5, 1, 2, 5}
```

Temporary

alias ہو جائے گا variable کی copy نہیں بنے گا۔

string c
static string y

class MyMath.
{ data

Static Objects

```
void f( ){
    static string y( );
}
int main ( ){

    f( );
    f( );
    f( );
}
```

}
int main()
{ mymath m1;
m1.power( )
MyMath::power

constructor AND destructor → ایک بار ہی چلے گا

```
Static string a ( );
void f( ){
    Static string b( );
}
int main( ) {
```

Static n

Check
ایک local بناؤ
ایک func میں static
ایک static global
ایک global

obj static پھر ختم ہونگے obj local پہلے — objects global

Static Data Members & Member Functions

ایسی classes کے obj بن جاتے ہیں جن کے کوئی data members نہ ہوں، ان کا size 1Byte ہے۔

اگر کسی func میں this pointer نہ ہو تو اس کو پتا ای نہیں ہوگا اس کو کس obj نے call کیا ہے

اس کا فائدہ ہے کہ obj بنائے بغیر بھی ان func کو call کر سکتے ہیں

static int CalcPower (int, int);

ClassName
in main file — MyMath:: calcPower(2,3)

Static member funct کے ساتھ ہم کوئی بھی This pointer کا استعمال نہیں کر سکتے۔

className * this → x

data member کے آگے static لگانے سے مراد ہے کہ یہ member اب کسی obj کا حصہ نہیں۔

Static data member کے ہم جتنے بھی obj بنا لیں اس کی ایک ہی copy ہو گی۔ جتنے مرضی obj اس class کے بنیں۔

Static member کسی بھی obj کے بننے سے پہلے ہی بن جاتا ہے۔

int car :: StampNo=1; possible ha

```
class car
    static int stampNo;
        int chasisNo;
        int x;
        int y;
    car ( — )
    {
        chasisNo = stampNo;
        stampNo++;
    }
```

Static data member کسی object کا حصہ نہیں ہوتا۔ class level پر ہوتا ہے object or instance → level par ni hota.

Reference count

```
static void f(){
    cout << this;  →  error
}
```

کیونکہ static میں this pointer نہیں ہوتا

اس static func سے non static data member کو Access نہیں کر سکتے

مگر reverse ہو سکتا ہے۔

```
static void f(){
    cout << "hello";
}
main(){
    String::f();
    return 0;
}
```

Static data member k getter and setter b static function hongy.

```
static int getX()
{
    return x;
}
static void setter(int val)
{
    X = val
```

```
main
Time t; SetX(45);
cout << Time::getX();
```

problem?

Linker error...

When use static variable define in Time cpp write globally:-

int Time::x;

we can also do direct initialization
int Time::x = 345;

⟷

header file:

static const int y = 345;
✓ ↳ in x-code
can declare or initilize in cpp file globally.

اگر static ہے تو globally cpp file میں initlize کر لیں۔
اگر static const ہے تو class کے اندر ای کر لیں
const Array کو globally initialize کر سکتے ہیں۔

⟷

data member کے سامنے static لگا دے
وہ class کا حصہ بن جاتا ہے
obj کے attributes سے اس کا کوئی تعلق نہیں۔

⟷

class Array {
static const int x; → ✓ no error

New lecture.

(30) Objects Relationship

whole - part relationship

composition — aggregation

composition → Strong aggregation

aggregation → weak aggregation / Shared

Death Relation

Association

Whole - part is one - way

Association is two - way

Address (): houseNo (0), streetNo (1)

{

}

Address (int hno, int sNO, String city, String area):
houseNO(hnO), streetNo(sNO),
City(cit), area(ar)

{

}

Copy constructor

```
Address(const Address& ref) : houseNo(ref.houseNo ...
{

}
```

Composition → death Relationship
→ host مرا تو guest بھی مرا

ایک class کے اندر اسی class کا obj
as data member نہیں آ سکتا = X

```
String
{
    String *s;   → ✓      Check
    String s;    → ✗
```

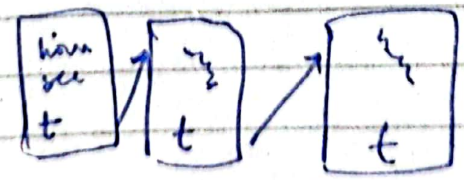
constructor infinite loop میں چلا جائے گا

```
Time {

    Time *t;

}
```

Time t1, t2, t3;



t1.t → t
→ hours
↓
t3.hours

t1.t = &t2;
t2.t = &t3;
t3.t = 0;

t1.t → hours ⇄ t2.hours
same

data member as alias / pointer کی class میں ایک class

بن سکتا ہے یا نہیں؟

alias نہیں بن سکتا

↳ کیونکہ alias کو initialize کرنا ہوتا ہے at the declaration

pointer بن جاتا ہے

→ کیونکہ pointer address کو store کرتا ہے

alias کو نہیں کیونکہ

When constructor is called for construction of alias object as data member it'll stuck in infinite loop so the compiler doesn't allow it.