SIDDHARTHA TEKRIWAL

13MA20041

OBJECT ORIENTED SYSTEMS DESIGN

Books Recommended -

(1) Grady Booch ; object oriented analysis and design with applications , Addison - Wesley Pale Co.

(2) James Rumbaughetcd , object oriented modelity and design. Prentice - Hall Inc. Pub.

(3) Deitel and Deitel , C++ , how to program . Prentice - Hall Inc. Co.

(4) Deitel and Deitel , Java , how to program . Prentice - Hall Inc. Co.

---

① New way of thinking about programming. (object + message) (entity) (method) model.

② Very near to model of reality. (operator + operand) dependency

(i) New data types + existing data types.

(ii) Provides flexibility to change. (objects are independent)

(iii) Easy - for reusability.

(iv) programming effort is reduced.

(v) Flexible.

Three principles of software development

① Modularity

(i) Vertical (ii) Horizontal

② Abstraction - abbreviation for a more detailed explanation.

③ Concept of information hiding with the recognition of structure.

Technological Advancements Leading to OOPs

① Advances in computer architecture including software support for them.

② Advances in programming languages such as Simula, Small talk, C++, objective, Java. ③ Advances in programming methods.-

Benefits of OOPs -

(1) Reduces the total life cycle software cost by increasing the programmer's productivity. and minimizing the maintenance cost.
(2) Resists corruption of useful information.
(3) Modifications and extensions are performed more easily.
(4) Understandability is simpler as data and methods are localized.
(5) Provides a mechanism for providing model of reality.

Limitations of our approach -

(1) No data abstraction and information hiding.
(2) Inadequate to problem domains and inherent concurrency.
(3) All the data used by them are global.

Software measurement performance evaluation -

(1) Structured programming → operator + operand model
(2) OOP → objects + message model → Objects contains attributes and methods. Each object can be classified as: (i) Actor (ii) Agent (iii) Server

(1) Correctness → Programs should meet their specifications correctly.
(2) Resilience and Reliability or Robustness → Programs should work even in abnormal conditions.
(3) Maintainability → Easy to maintain and easy to extend for any changes.
(4) Interoperability → Program should be easy to port to other systems or should be compatible with other system.
(5) Reusability and generalization.
(6) Efficiency.
(7) Verifiability
(8) Security
(9) Integrity
(10) Friendliness.
(11) Describability.
(12) Understandability.

Software Development Process - (Models)

1. Object Model
2. Dynamic Model
3. Functional Model

Object Model - describes all types of objects and their inter-relationships.
It is a graph in which nodes are the objects and edges are their inter-relationships.

Dynamic Model - describes the changes over the objects overtime. For occurence of different events, how objects states are changing. This can also be described by a graph in which nodes are states (of all the objects) and edges are the transitions which takes by different events.

Functional Model - Data-flow-diagraph is also a graph which indicates how computation within the object and outside object is carried out.

SOFTWARE DEVELOPMENT PROCESS:

(i) Identification of objects and their inter-relationships. Thus, identifying objects as actors, agents and servers role played by them in the model of reality. If several similar types of objects appear, they are classified by using class concept. | classes + objects.

(ii) Identify operations suffered by and required of each object. This specifies the behaviour of each object.

Behaviour → (I) Static → attributes of the objects.

(II) Dynamic → specifying the constraints (say on time/space) that are to be observed on the changes.

(iii) Establish the visibility of each object in relation to other objects by identifying the static dependencies among objects. ← Topology of the software application.

(iv) Establish the interface of each object. (v) Implementation.

Fundamentals of OOP -

1. Encapsulation (Data hiding)
2. Abstraction
3. Modularity.
4. Hierarchy (Concepts of information hiding with the Recognition of structure).
5. Typing (classification) → class (Anonymous)
6. Concurrency (property that distinguishes an active object from non-active)

Objects in OOP are created in two forms -

1. Decomposition
2. Composition

Two views of an object -

1. External
2. Internal

Two methods of computation -

1. Overleaved - parallel computation
2. Interleaved - part by part computation.

7. Persistence (objects continue to remain even after their creator ceases to exist)

1. Identification of objects and its attributes by recognizing major actors, agents and servers. If several objects of similar type, define class for them (object model) / subclasses.
2. Identifying the behaviour of each object by recognizing the operations, it requires to perform and it suffers from the messages (methods) of other objects. (i) static (ii) dynamic.

(ii) dynamic → constraints which are to be placed on occurrence of different events.

(3) Establishing the visibility of each object in relation to other objects by identifying their state dependencies ⇒ topology of the object model.

(4) Establish the interface of each object, by using same representation or different representations.

(5) Implementation.

OBJECTS -

An object is a self-entity whose behaviour is characterized by the action it performs and requires to perform by other objects. All objects have-

(1) State - attributes and their representation.

Ex. Multi-window system → message window / command window / text window.

(i) size (ii) its contents (which may be an object)

(2) Behaviour - the methods carried over by the windows.

Ex. Open / Close / read / write / insert .....

Message by other objects requesting the operations to be carried out. Message is another method with parameter specifying the

~~Operators that carry out operations on objects.~~

(1) Constructor - That changes the state of the object. Objects are defined as an instance of a class. These creates the objects. The constructors can be overloaded and they should have the same name as class name. They do not return anything.

Destructor - e.g.. ~ Student ( )

(2) Selector - An operation that evaluates the current state of the object.

(3) Iterator - That permits all parts of the object are to be visited and return objects.

① actor ② agent ③ server.

Reusability - can be achieved by using primitive operations. These operations are those who have access to the entire object and its representation in an efficient manner.

Class - It is a template or blueprint denoting similar but unique objects.

Important properties/characterics of OOP approach -

① Inheritance
② Polymorphism < Function - with the same name but with different function
Operator - with several meaning.

⇒ new datatypes / existing datatypes.

---

① name < same name may be given for different objects.
aliases → different names may be given for the same object (all names will be mapped to one object)

② restricted visibility - In the worst case, each object can see each other. It is the responsibility of the application developer to decide what objects should be visible so that to limit the no. of objects utilized, which leads to the better understanding of any part of the system and also limits scope of change.

(i) External view - objects and functions to be utilized by other objects.

(ii) Internal view - Implementation.

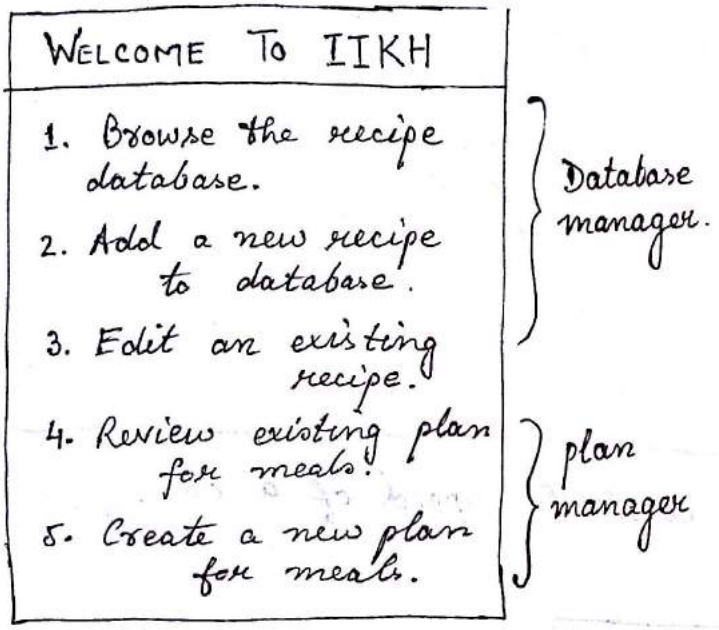
Ex. ① Dog (name, color, breed, ...) ← attributes
(barking(), wagging-tail, fetching()) ← behaviour.

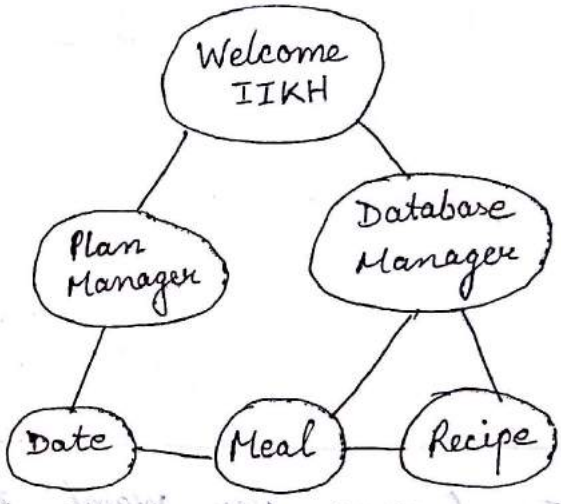
② Bicycle (frame size, wheel size, gears type, break type, material)
(move(), repair(), shift(), ...)

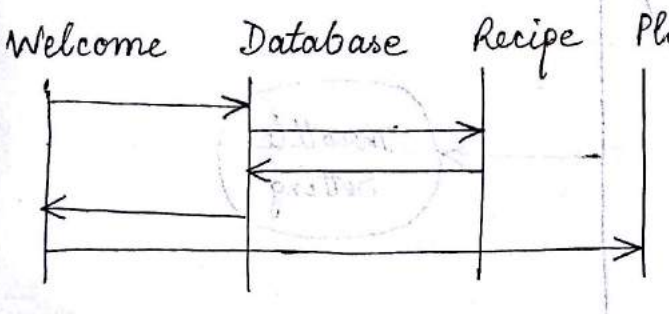
③ Lamp (ON/OFF) (turnon / turnoff)

IIKH ≡ Interactive Intelligent Kitchen Helper.

Suppose we want to plan for our menu for a meal/ for a day/ for weekly plan/ for monthly plan by sitting at a terminal. It has a list of recipes, the ingredients used, as well as the method of preparation. A user can change the existing recipe, provide a new recipe with required ingredients and its method of preparation. You can have a date wise print out of your menu. The system is intelligent in the sense that indicate when to order/ how much to order/ whom to order, etc...



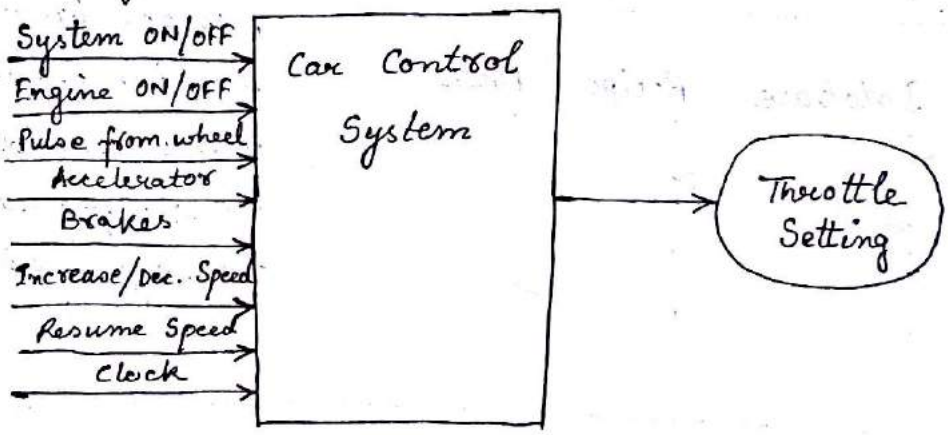




Message browser()
Message display()
return from display()
return from browser()
message make plan()
⋮

Ex. Suppose you want to control speed of a car.

The Block diagram -



(1) System ON/OFF → If ON, the System should maintain speed.

(2) Engine ON/OFF → The System will be active only when Engine is o

(3) Pulse from wheel → A pulse is sent for every revolution of th wh

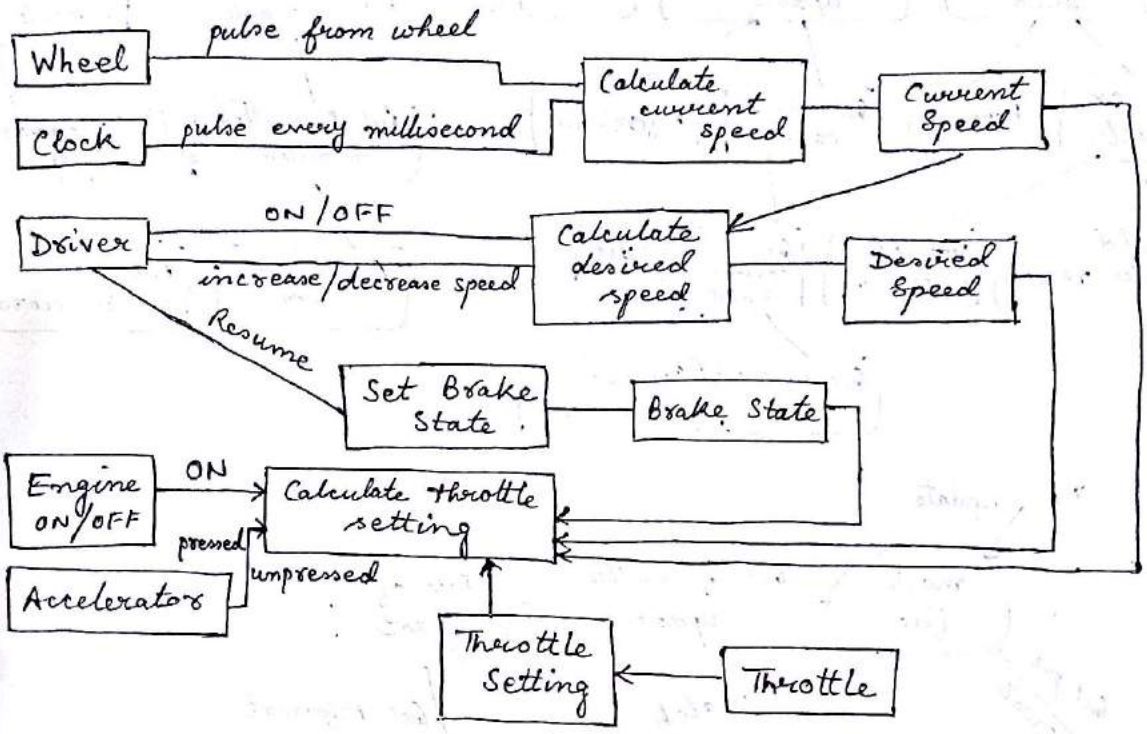
(4) Accelerator → It indicates how far accelerator is pressed.

(5) The Break is ON when break is pressed. The car control syste temporarily reverts to manual control if the brake is presse

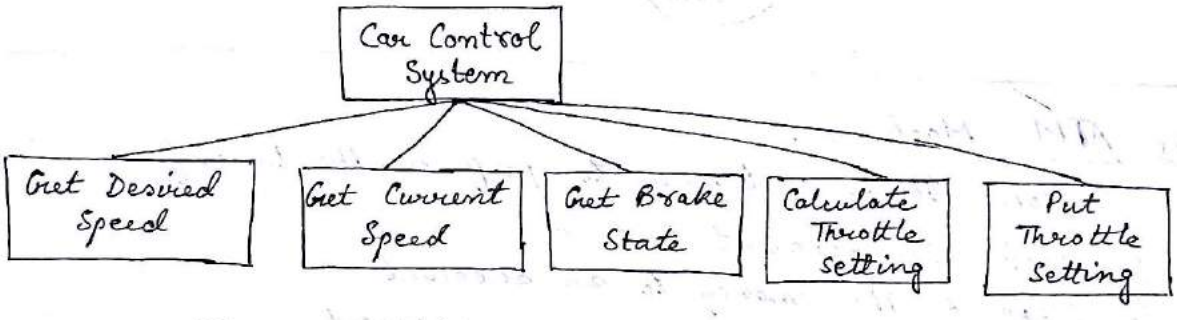
⑥ Increase/Decrease Speed → Increases and Decreases speed. It is applied when the system is ON.

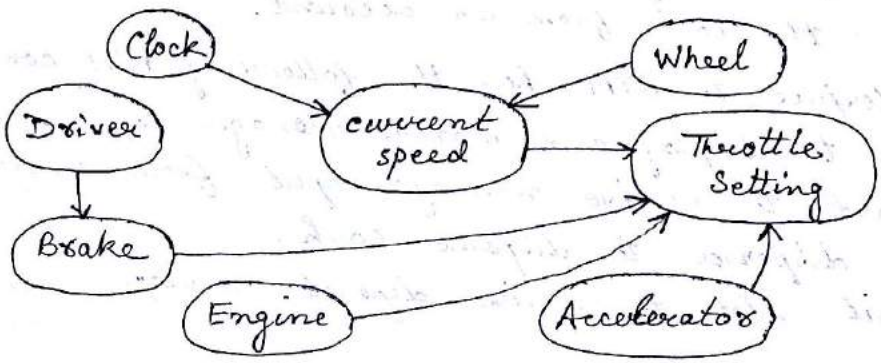
⑦ Resume Speed → Resume the last maintained speed.

⑧ Clock → Timing pulse every millisecond.

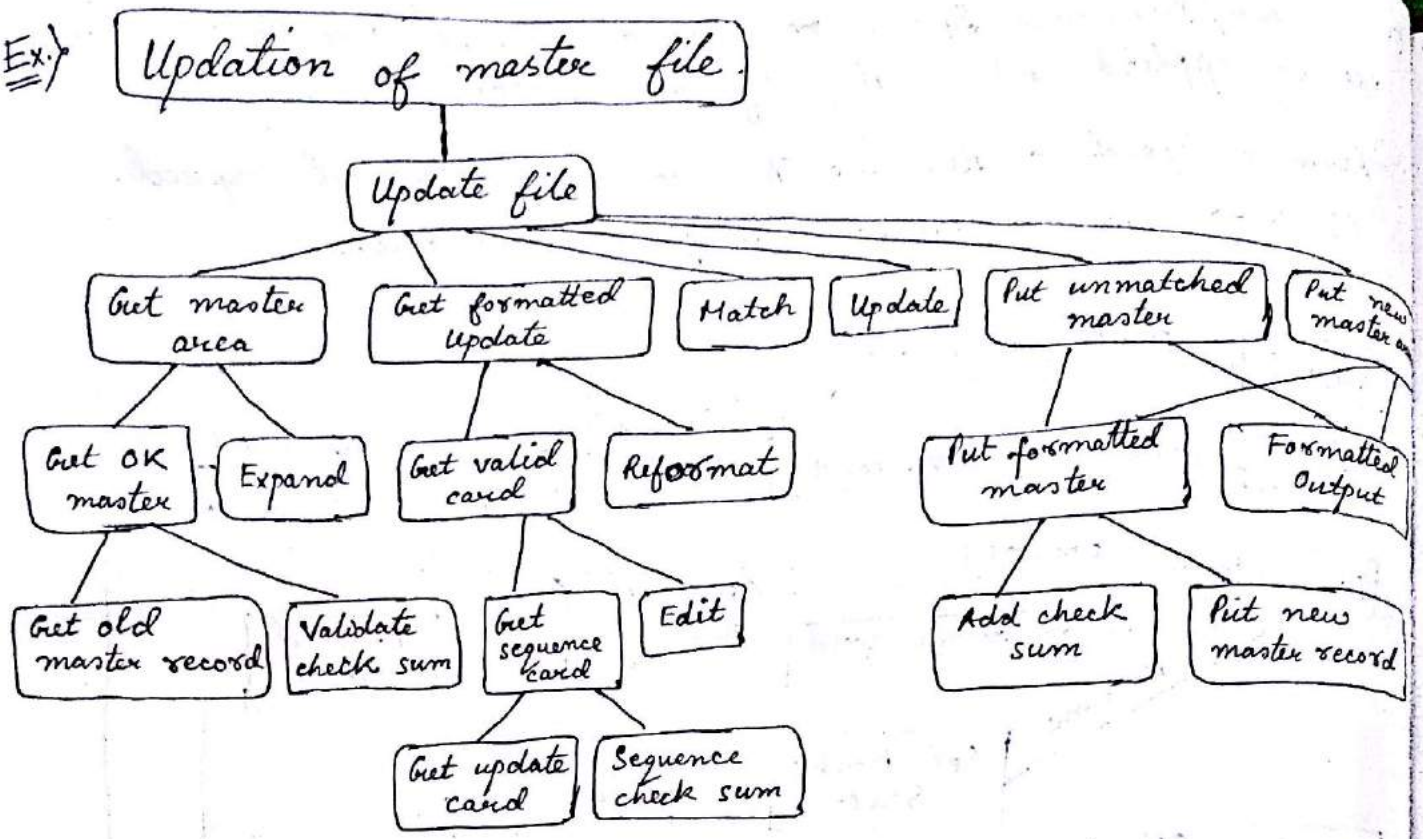


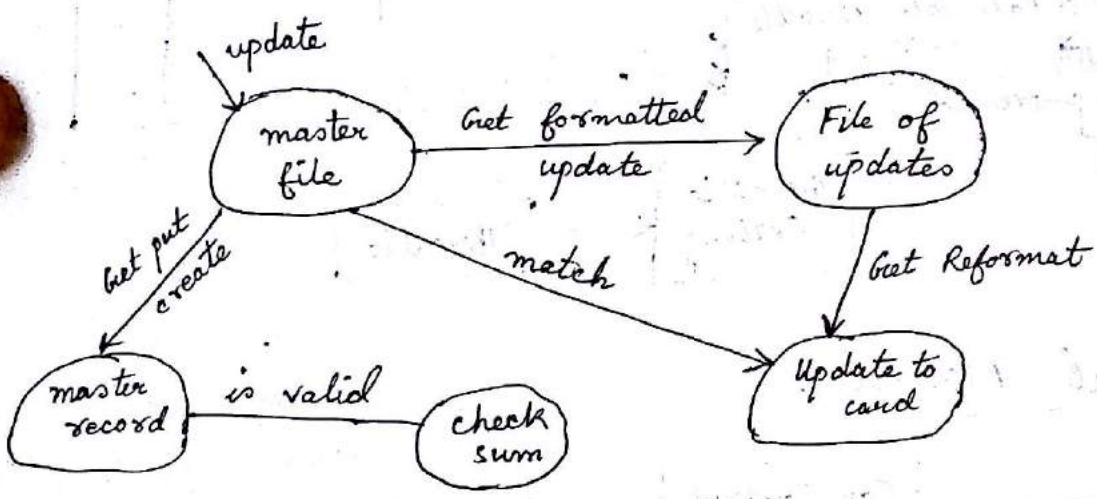
Procedural Model :





Ex.)





Ex. **ATM Machine :**

allows bank customers to perform the basic financial transaction

1. check the account balance..
2. Deposit the money to an account.
3. Withdraw the cash from an account.

A user interface to ATM has the following H/W components

1. A screen to display and manage messages to users.
2. A keyboard to receive message input from users.
3. A cash dispenser to dispense cash.
4. A deposit slot to receive deposit envelope.

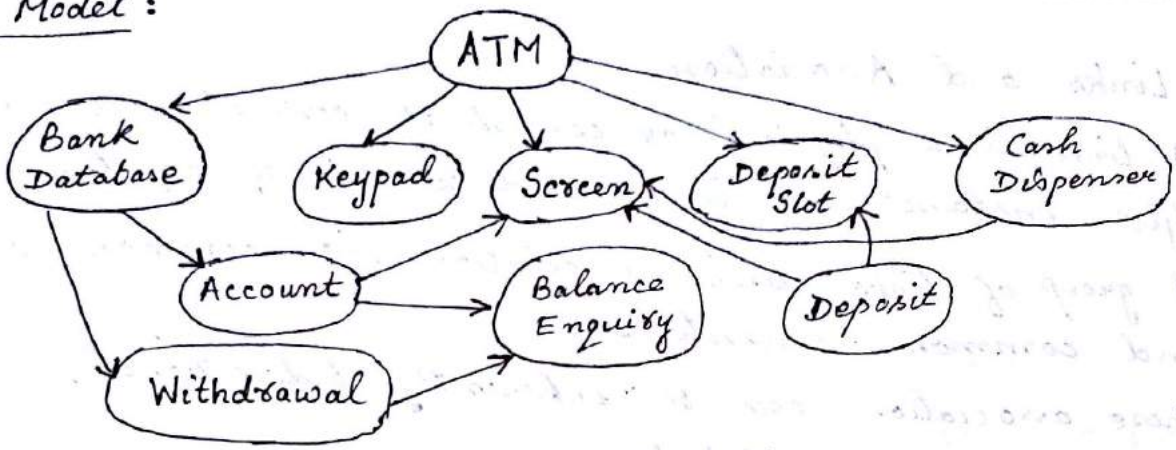
Assumptions : ① Each user has only one account in the bank. When he wants to interact, he inputs his acc no.

② The cash dispenser logins each day by loading it with Rs. 50,000/- in denomination of Rs. 100/-.

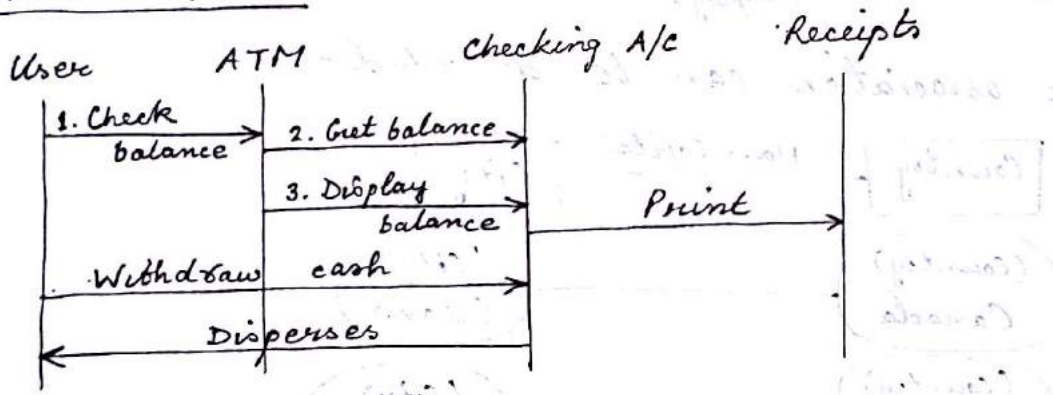
③ When the menu = 2, the available balance should be more than the withdrawal amount. An error reported otherwise.

④ Only one withdrawal is allowed.

Object Model :



Dynamic Model :



SCREEN :

Welcome to ATM

Please enter your a/c no.

Enter your pin.

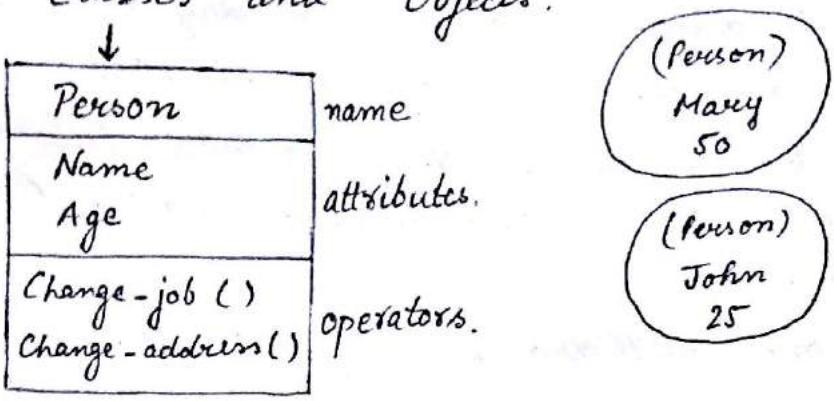
Main Menu :

1. View Balance
2. Withdraw Cash
3. Deposit Cash
4. Exit

Enter Your Choice.

OMT ≡ Object Modeling Technique Notation :

① Classes and Objects.

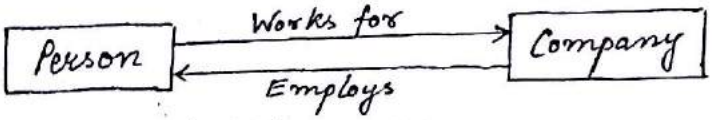


② Links and Association

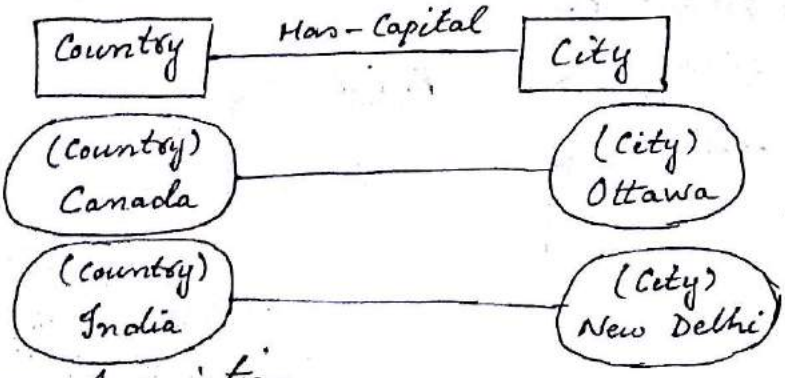
A link is a physical or conceptual connection between object instances. It is an orderied list of objects.

A group of links define association with common structu and common semantics.

These associations can be inherently bidirectional.

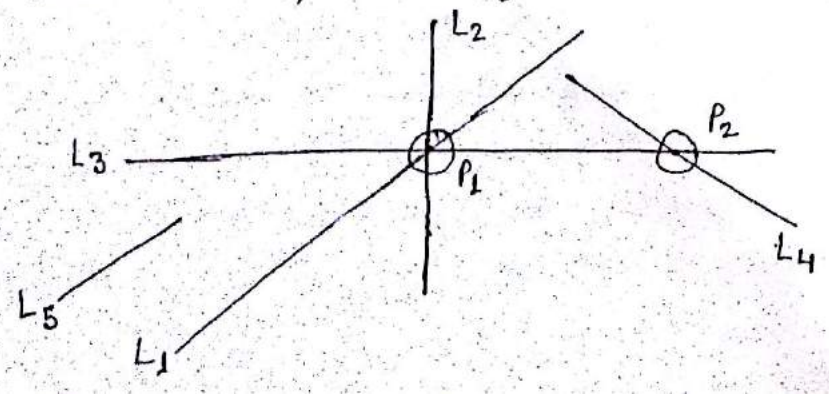


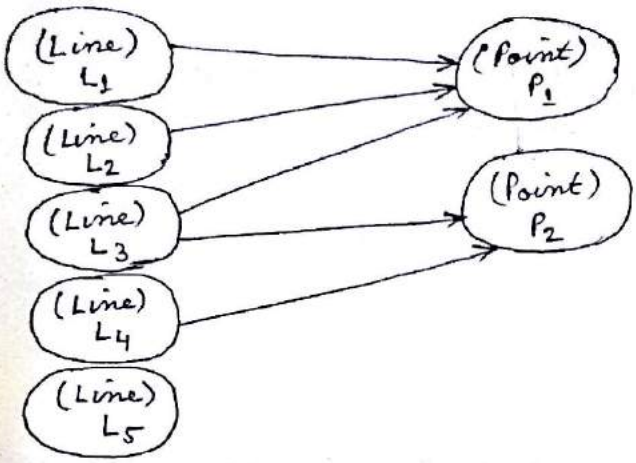
One to one association can be described -



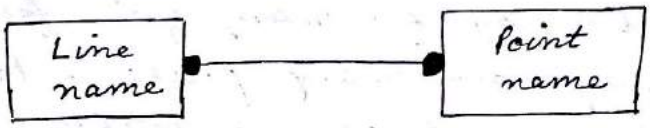
Many to One Association

Ex. Given a line, find all intersecting lines or given an intersection point, find all lines that pass through it.



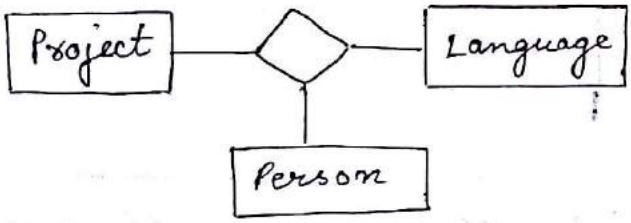


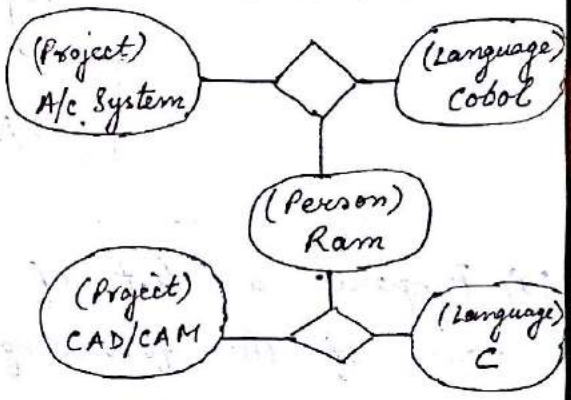
Many to many association



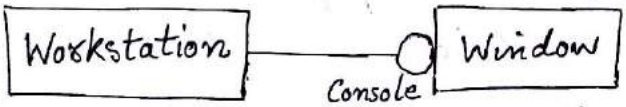
Association can be ternary and denoted by ◇.

Ex. Persons who are programmers use programming language and project.



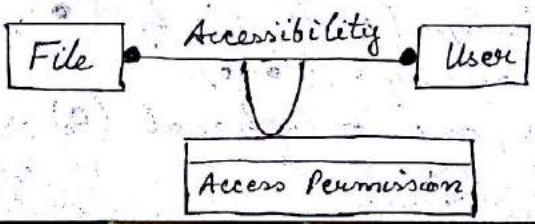


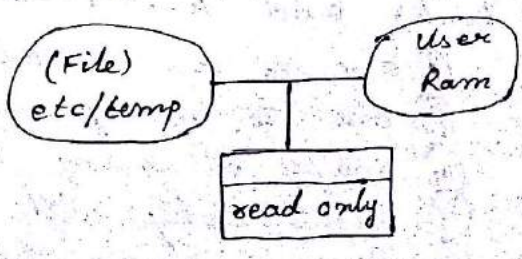
Ex.



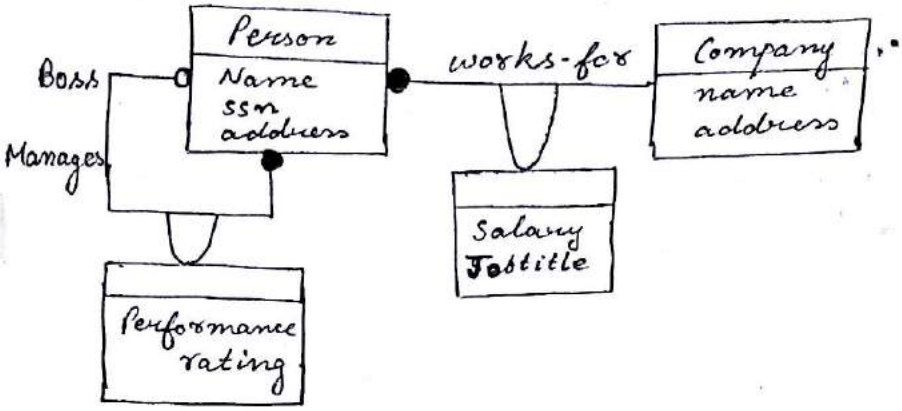
A workstation uses a particular window as its console.

Link attributes → An attribute is a property of an object in a class. Link attributes are the property of association and denoted by a loop.



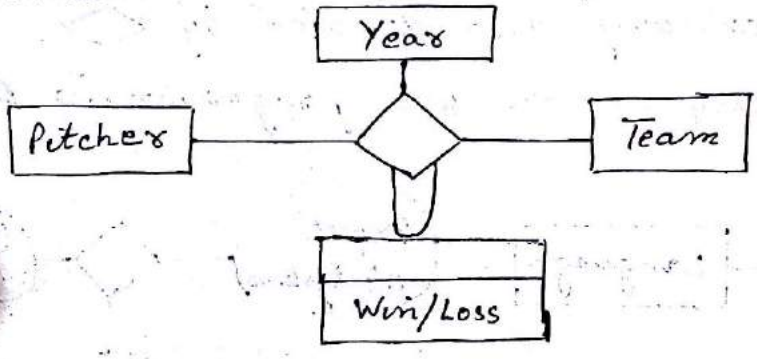


The following diagram shows the link attributes for many to one association.



Following diagram shows link attributes for ternary associat

A pitcher may also play for many years for a given team. A given team may have many pitchers. For each combination a team and years a pitcher has won-loss record



Ex) Prepare a list of objects that you would expect each of the following system to handle.

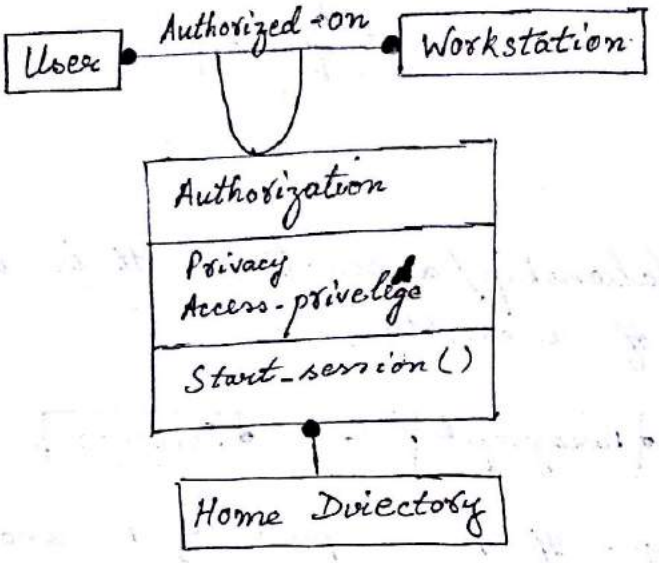
1. A program for laying out a newspaper.
2. A telephone answering machine.
3. A video rental system.

Object modeling techniques notations (OMT) -

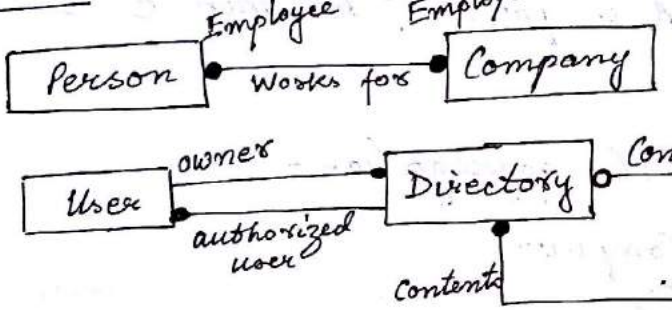
1. classes and objects
2. Links and associations. (◇)
   - (i) one to one relationships
   - (ii) one to many relationships (Multiplicity) (●)$_{2+}$
   - (iii) many to many relationships (●—●)
   - (iv) relationship with zero, one or optional (Q)
   - (v) Ternary association (—◇—)

(vi) Link attributes → properties of a link or association.

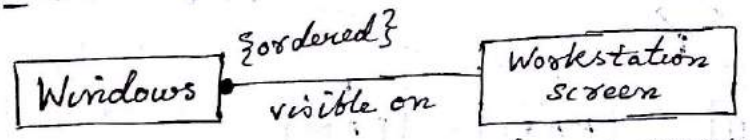
Ex.



Role Names -



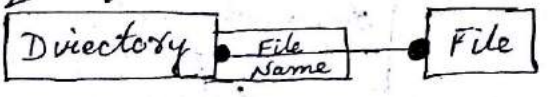
Ordering -



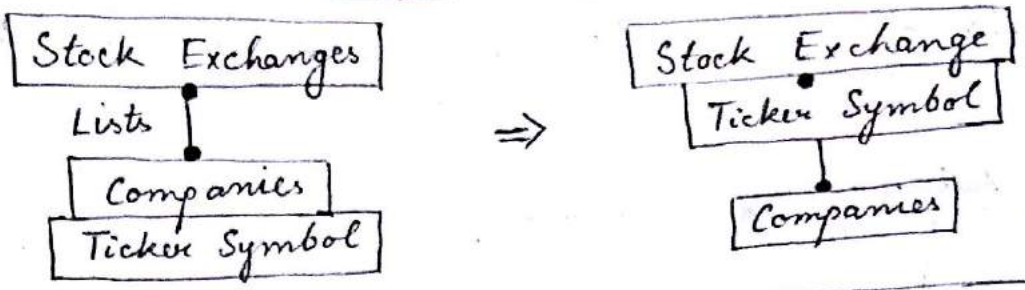
A work-station screen can have several overlapping windows

Qualifications -

A qualification association relates two classes and a qualifier. Denoted by small box containing file name.

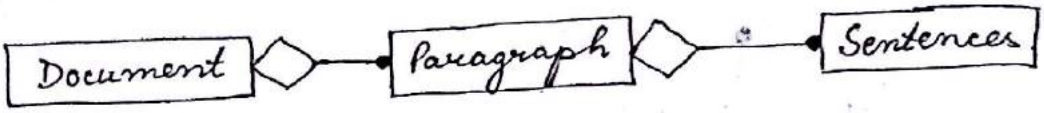
(1) 

(2) A stock exchange lists many companies. However, the stock exchange lists only one company with a given ticker. A company may be listed on many stock exchanges possible under different ticker symbols.



Aggregation -

It is a part of relationship/association. It is denoted by (◇) at the end of assembly.

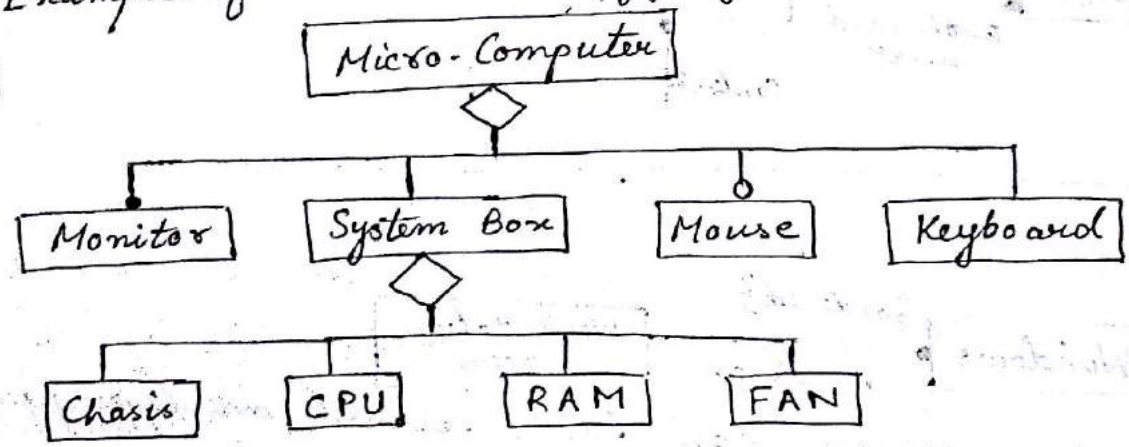


(1) Transitive relationship - If A is part of B and B is part of C, then A is also part of C.

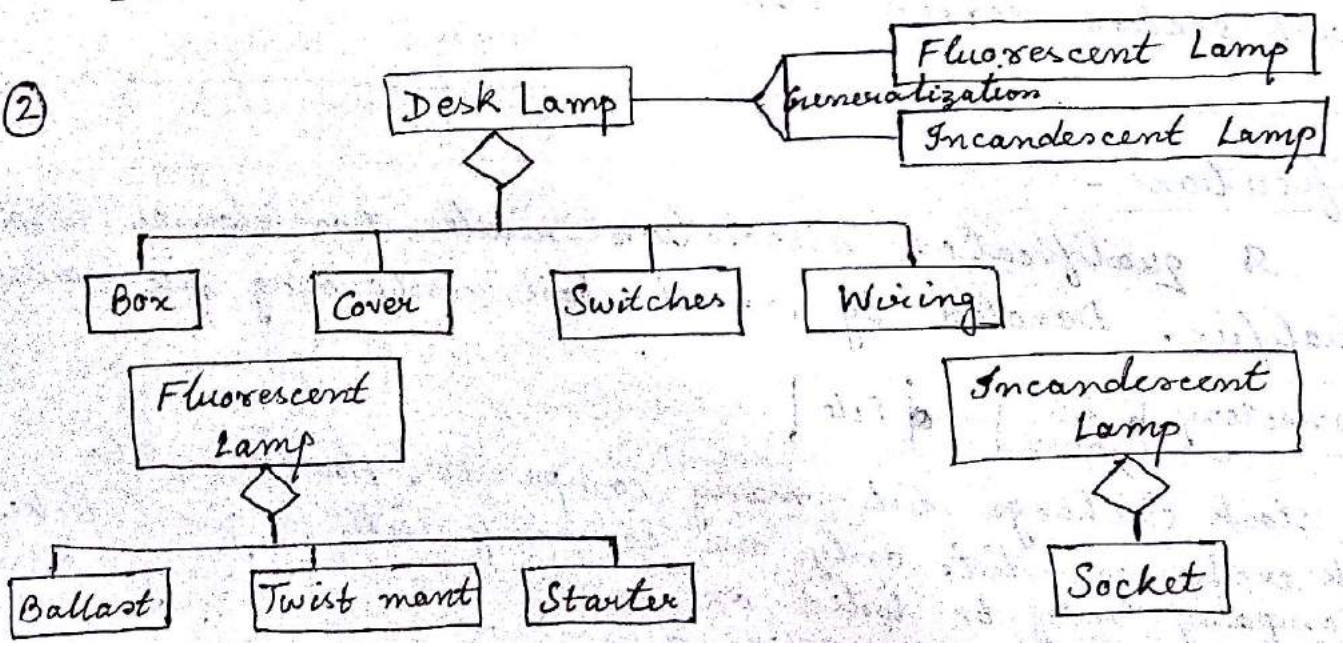
(2) Antisymmetric - If A is part of B, then B cannot be part of A.

* Examples of multi-level aggregation -

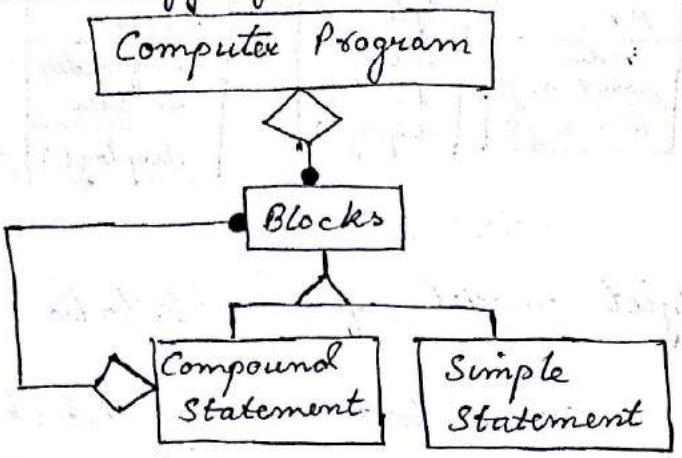
(1)



(2)

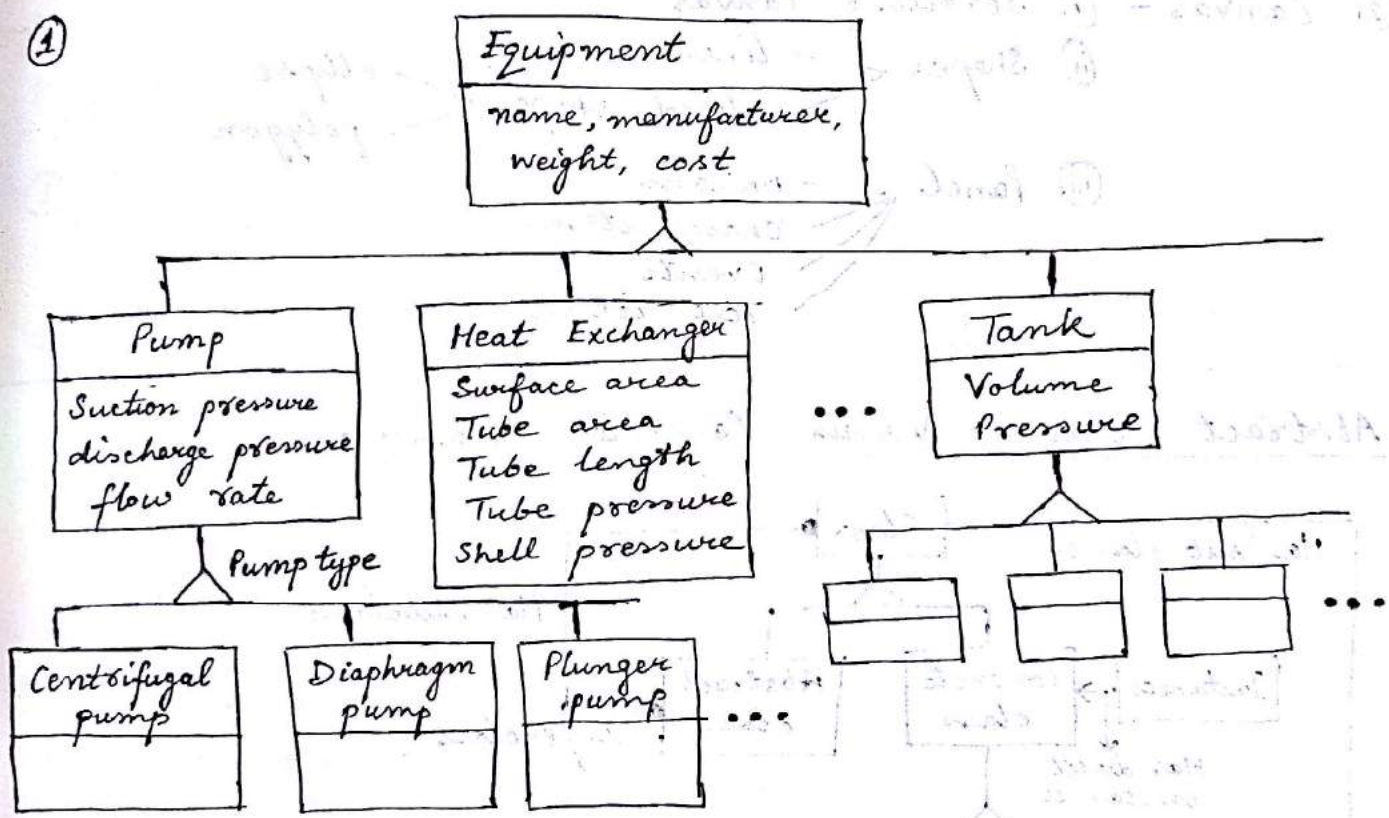


Recursive Aggregate -

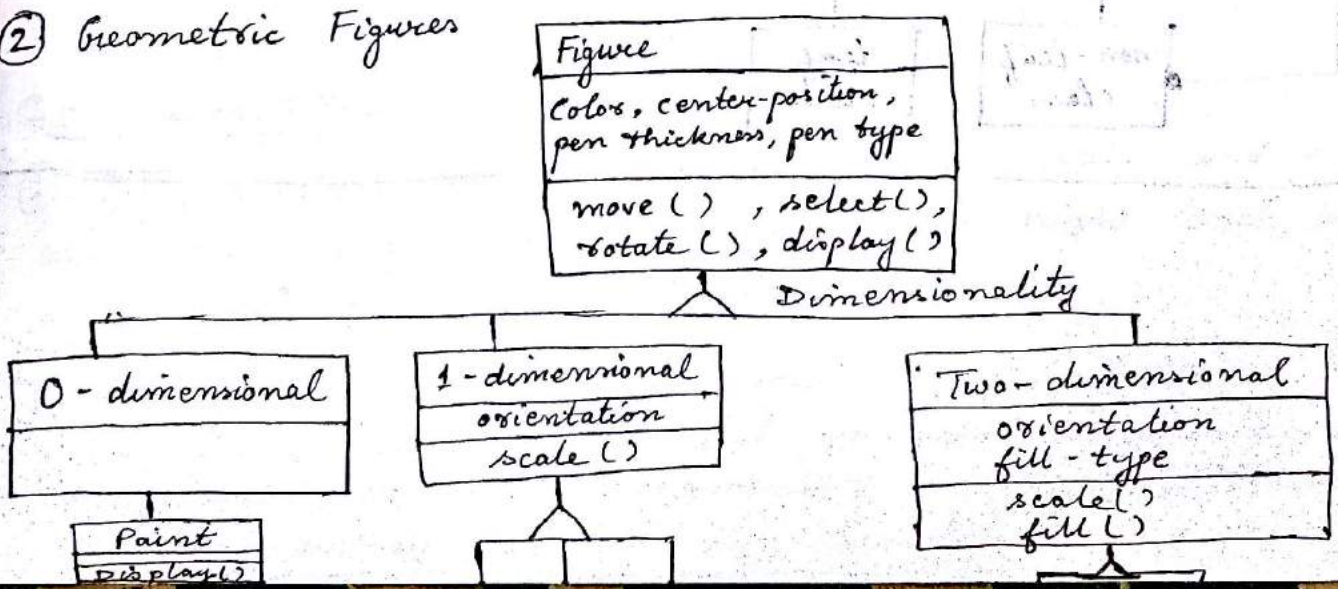


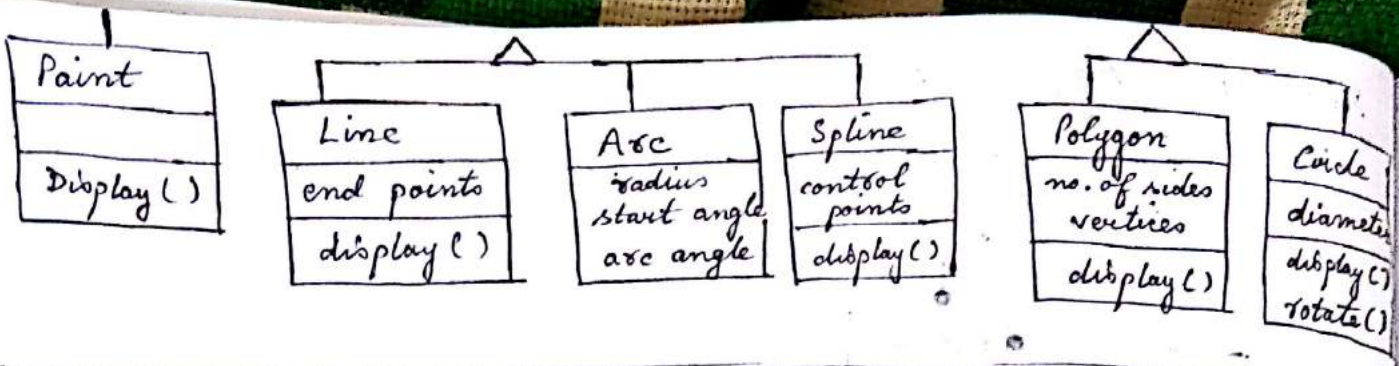
A multilevel inheritance hierarchy -

(1)



(2) Geometric Figures



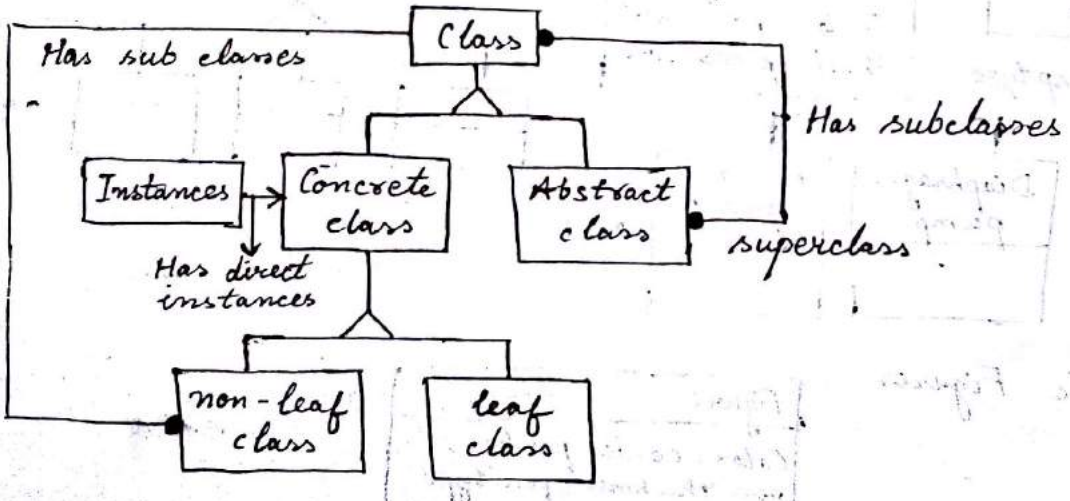


Exercise) Describe an object model for workstation window management system.

(1) Scrolling window - It contains text window which has two dimensional scrolling offset as specified x-offset and y-offset as well as operation scroll ( ).

(2) Canvas - (i) Scrolling Canvas
(ii) Shapes < lines
closed shapes < ellipse, polygon
(iii) Panel < Buttons, choice items, Events, Text items

Abstract Classes Versus Concrete Classes :



C++/C :

(1) register storage class and bit manipulation operators
(2) pointers and ability ~~to create~~ to do arithmetic over them.
(3) can create user defined derived data types using pointers, structures, union and enum.
(4) Single thread of control.

Java :

(1) Simple →
  (i) eliminates many keywords.
  (ii) no preprocessor
  (iii) Large body of application development tools.
    Java Application programming interface (API)
    import java.lang.*
  (iv) Independent functions without overloading.
  (v) no pointers, goto, structures/unions.
    ⇓
    references ⇒ different references can be given to one variable.
  (vi) Multithreading.

C++ Limitations :

(1) No operators to deal strings, lists, sets and arrays declared as a whole. It has no high level data types and operations on them.
(2) No input/output functions (facilities) i.e., there is no read/write statements and no-wired-in file access.
(3) No support for multiprogramming, i.e., no multithreading, no concurrency and, no synchronization is supported.

Java :

② Network Savvy →

(i) Client - server computing.
(ii) provide rich set of tools for internal programming.

→ A user is the client which can activate a network aware application.

(URL's) → universal resource locator through the web-browser. A program is executed on the server to meet the user request (CGI-Bin).

③ Interpreted → Java programs are compiled by a Java compiler which translates it in to byte codes of one or two bytes. These byte codes are executed on JVM which is a s/w used to interpret and produce machine instructions.

It supports just-in time (JIT) compiler.

```
$ Javac A.Java
$ A.class
$ Java A.
```

④ Java is Secure →

Java and C++ :

Consists of alphabets, digits and special symbols. Primary constants, integer, real, boolean, char types, secondary constants, Arrays, pointers, structures, unions and enum.

The variables are declared by identifiers and they can vary. Basically, they refer to memory locations which contains constants. All variables are associated with typ

Four types of instructions -

1. Type declaration instructions.
2. Input / Output instructions.

getchar(), putchar(), scanf(), printf(), gets(), puts().

scanf("<format string>", <list of variables>);
(preceded by &)

printf("<format string>", <list of variables>);

cin, cout, clog, cerr;
>> <<

cin >> x
cout << "String is = " << x;

3. Arithmetical and logical instructions.

arithmetic operators → *, /, %, +, −, =

logical operators → !=, <, >, <=, >=, ==, &&, ||, =.

C++ and Java : Similarities and Differences -

1. No destructor in Java but c++ supports destructor.
   ⇒ Garbage collection and compaction.
2. The constant specifier in Java is finally but C++ supports const. as declaring constant.
3. No memory management in c++, but Java supports memory management features.

Some statements which are similar in C++, Java are:

(i) The continue statement - when it occurs in a loop, control goes to the first statement of the loop.

(ii) Conditional statement and switch - statement exp1 ? exp2 : exp3.

(iii) The switch-case statement -

```
switch (expression)
{
    case 1 : do this ;
    :
    default : do this ;
}
```

(iv) The break statement - If it appears in the loop, the control goes to the first statement after the loop.

(v) The control statements -

(1) Decision → if-then-else

(2) Loop → (i) for-loop
(ii) while-do
(iii) do-while.

for-loop - for (exp 1 ; exp 2 ; exp 3) S_1 ; → where S_1 is a compound or, simple statement.
↓ initialization ↓ conditional expression ↘ increment

The comma operator ⇒ More than one initial expression and increment expression can be specified in the for loop.

while-do - while (logical expression) do S_1 ;
⇓ true

C++ : 0 → false , 1 → true.

Functions and methods -

It has the form:

storage class data-type function-name (formal arg 1, formal arg 2, ... formal arg n)
{
body of the function.
}

default → type of value returned by the function.

Only one value is returned by a function by using a return statement.

A function can be called by function-name (actual arguments);

Passing arguments to the function -

(1) By value (2) By pointers (3) By reference.

↓ copy of the arguments is passed and any alteration is not reflected in the calling function.

Storage class → Scope of the variables is the portion of the program where the variables are used.

① auto
variables defined inside the function are local or automatic to the function. They do not exist if the function terminates.

② Extern
The variables declared external has scope from the point of its declaration to the rest of the program.

```
extern a;
func()
  int a;
(overrides extern a)
```

③ static
They retain their values an takes them wh the function is called again.

④ global.

Ex. ①

```
float a,b,c;
main()
{
  static float a;
  void dummy(void);
  ||
  ||
}
void dummy(void)
{
  static int a;
  int b;
  ____
  ____
}
```

a will change but b and c takes the same value.

Ex. ① Write a C++ program to find the maximum of three programs.

② Write a Java program to find the maximum of three floating point numbers.

---

(1)

```
# include <iostream>
main ()
{
    int a, b, c, d ;
    cin >> a >> b >> c ;
    d = maximum(a, b) ;
    cout << "maximum = " maximum (c, d) << endl;
}
int maximum (int x, int y)
{
    int z ;
    z = (x > y) ? x : y ;
    return (z) ;
}
```

(2)

```
import java.util. Scanner ;
public class MaxPrg.
{
    public void findmax()
    {
        Scanner input = new Scanner (System.in);
        System.out.print ("Input three numbers");
        double num1 = input.NextDouble ();
        double num2 = input.NextDouble ();
        double num3 = input.NextDouble ();
        double result = maximum (num1, num2, num3);
        System.out.println ("Maximum is " + result);
    }
public double maximum (double a, double b, double c)
{
    double m ;
    m = (a > b) ? a : b ;
    return (m > c) ? m : c ;
}
}
public class Max
{
public static void main (String args[])
{
    MaxPrg ob = new MaxPrg ();
    ob.findmax ();
}
}
```

(*) Write a Java Program to find the sum of 3 integers

```
import java.util.Scanner;
public class SumPrg
{       public void sum (~~int a, int b, int c~~)        → take a, b, c as input.
        {       return (a+b+c);
        }
}
public class Sum
{       public static void main ()
        {SumPrg ob = new SumPrg ();
         ob.sum ();
        }
}
```

Arrays -

(1) whose elements are all of the same type and similar characteristics. They share the common name and individual elements are accessed by the name of the array and indices enclosed in square brackets.

⇒ indices represent the dimensionality of the array.

General Form -

storage_class datatype array_name [index 1, index 2, ..., index n]
= { val 1, val 2, ..., val n };

Ex) extern int digit [10] = {1, 2, ..., 10};
static float n [10];

String are an array of characters.

```
char array [4] = "RED";
char array [] = "RED";
```

array [0] = 'R'
array [1] = 'E'
array [2] = 'D'
array [3] = '\0'

(2) Array passing to a function - only the array name and the position from where elements are to be passed.

Ex)
```
main ()
{   int n; float avg;
    float list [100];
    float average ();
    avg = average (n, list);
}
```

float average (int, float []);
float average (int m, float list[]);

③ Multi-dimensional array -

storage-class datatype array-name [exp 1][exp 2]...[exp n];

Pointers - pointers are addresses and a variable which represents them is known as a pointer variable.

datatype *ptr;
↳ type of data pointer is pointing to.

& → address operator.

Ex.) double *xptr, *yptr;
int count = 8;
int *ptrcount;
ptrcount = &count.

Properties - ① dynamic data structures. (shrink, grow)

Ex.) int *x;
x = (int *) malloc (10 * sizeof(int));

Ex.) x = malloc(10 * sizeof(int));
⇒ a block of memory representing a character.

② pointers can be assigned the integer 0 without declaring 0 as an integer. NULL ≡ 0.

③ Operations on pointers. A pointer variable can be assigned to another pointer variable compared using equality, inequality, relational or logical operators provided they represent the same datatype.

④ A pointer may be incremented (++), decremented (--), add an integer (+, +=), subtract (-, -=).

* datatype (*array-name)[exp 1][exp 2] ------- [exp n-1];
* datatype *array-name [exp 2] ----- [exp n];

Passing pointers to functions -

Ex.) consider the program to convert inches to centimeter.

```
#include <iostream>
void main ()
void intocent (double *&);
double var = 10.0;
intocent (var);
```

```
void intocent (double *&v)
{     v *= 2.54;
}
intocent (var);
```

Ex.) `int *p (int a);`
`int (*p) (int a);` — one-dimensional.
`int (*p) (char *a)`
`int (*p) (char (*)a)` — multi-dimensional

---

(1) `int (*p) [10];` → p is a pointer to 10 element int array.

(2) `int (*p)(char *a);` → p is a pointer to a function which accepts argument - pointer to a character array and returns an integer quantity.

(3) `int (*p (char *a)) [10];` → p is a pointer to a function which takes pointer to a character array and returns an integer quantity.

(4) `int p (char (*a) [ ]);` → p is a function which takes argument pointer to the character array and returns an integer quantity.

(5) `int p (char *a [ ]);` → p is a function that takes array of pointers of char type and returns an integer quantity.

(6) `int *(*p)(char (*a)[ ]);` → p is a pointer to a function which takes argument a pointer to a character array and returns a pointer to an integer.

(7) `int *(*p)(char *a[]);` → p is a pointer to a function which accepts an array of pointers of char type and returns a pointer to an integer.

(8) `int (*p[10])(void);` → p is an array of 10 pointers to a function, each function takes nothing and returns int.

(9) `int *(*p[10])(char a);` → p is an array of 10 pointers to a function, each function takes a char 'a' as argument and returns integer pointer.

(10) `int *(*p[10])(char *a);` → p is an array of 10 pointers to a function, each function takes a pointer to a character and returns pointer to integer.

Structures and Unions –

A structure is a data structure in which members are of different types.

Its general form is –

```
struct tag ──────→ type of a structure.
{
    member 1;
    member 2;
    ⋮
    member n;
};
```

(1) no storage class is assigned.
(2) member functions are not initialized.

Structure variables are declared as –

```
struct tag var1, var2, ..., var n;
```

```
typedef existing-type, new_type;
```

Ex.)
```
struct part
{
    int modelno;
    int partno;
    float cost;
};
struct part part1, part2;
```

```
static struct part part1 =
        {123, 678, 10.5};
part2 = part1;
```

```
typedef struct {
    int mon
    int day
    int ye
} date

customer.(&acctn
(*pc) → acctno;

customer.lastpay
(*pc).lastpaymen
```

Union –

```
union {
    member 1
    member 2
}
```

Structure passin

(1) By value

Ex.) typedef str

distance d1, d

distance add

d3 = add- le

distance ad

{ distance

d3. in

x = (in

d3. fe

d3. in

return

}

```
typedef struct {
        int month;
        int day;
        int year;
        } date;
```

```
typedef struct {
        int *acctno;
        char *acctype;
        char *name;
        float *balance;
        date lastpayment;
        } customer, *pc = &customer;
```

customer.(&acctno);
(*pc) → acctno;

customer.lastpayment.year;
(*pc).lastpayment.year;

Union -

```
union {
        member 1; → 2 bytes
        member 2; → 4 bytes
        }
```
} 4 bytes. (memory conservation)

Structure passing to functions -

① By value ; ② By reference ; ③ By pointers.

Ex.)
```
typedef struct {
        int feet;
        float inches;
        } distance;

distance d1, d2, d3;

distance add_length (distance, distance);
d3 = add_length (d1, d2);
distance add_length (distance d1, distance d2)
{       distance d3; int x;
        d3.inches = d1.inches + d2.inches;
        x = (int) (d3.inches/12);
        d3.feet = d1.feet + d2.feet + x;
        d3.inches = d3.inches - int(d3.inches/12)*12.0;
        return (d3);
}
```

```
typedef struct distance {
            int feet;
            float inches;
            distance *next;
        };

struct linklist {
        int x;
        linklist *next;
    }
```

Classes and Objects -

C++:

```
class Small {
    private:
        int x;
    public:
        void setdata(int d)
        {
            x = d;
        }
        void display()
        {
            cout << "\n data is" << x << endl;
        }
};

Small s1, s2;
s1.setdata(100);
s2.setdata(500);
```

(inside the class)

```
void setdata(int);
```

(outside the class)

```
Small :: void setdata(int)
{
    x = d;
}
```

```
Small *objptr;
objptr = & s1;

Small &objptr;
objptr = s1;
```

Java:

```
public class Small
{
    private int x;
    public void setdata(--)
    {
        ---
    }
    public void display()
    {
        ---
    }

Small S1 = new Small(100);
Small();  → Constructor
~Small();  → Destructor
```

Objects and Classes -

Ex.) Write a C++ program to print time in universal and standard format.

Universal format is 24 hours clock time format. 1 PM is 13 and 11 PM is 23 with midnight is 0 and noon is 12.

Standard format is to be followed by AM and PM. For example, 1:30:07 PM ⇒ If hour is 0 or 12, it appears as 12 otherwise 1 to 11.

```
#include <iostream>
#include <iomanip>
class Time
{   public:
        Time();
        Time(int, int, int);
        void setTime(int, int, int);
        void setHour(int);
        void setMinute(int);
        void setSecond(int);
        void printStandard();
        void printUniversal();
        int getHour() const;
        int getMinute() const;
        int getSecond() const;
    private:
        int hour;
        int minute;
        int second;
};
Time :: Time()
{   hour = 0; minute = 0; second = 0;
}
Time :: Time(int h, int m, int s)
{   setTime(h,m,s); OR, hour = h; minute = m; second = s;
} OR, hour = setHour(h); minute = setMinute(m); second = setSecond(s);
```

```
void Time :: setTime (int h, int m, int s)
{
    hour = h ; minute = m ; second = s ;
}
void Time :: setHour (int h)
{   hour = h ;
}
void Time :: setMinute (int m)
{
    minute = m ;
}
void Time :: setSecond (int s)
{
    second = s ;
}
void Time :: printUniversal ()
{
   cout << setfill ('0') << setw(2) << hour << ':' <<
           setw(2) << minute << ':' << setw(2) << second <<
                                                  endl;
}
void Time :: printStandard ()
{
    cout << ((hour == 0 || hour == 12) ? 12 : hour % 12) << ':' <<
        setfill ('0') << setw(2) << minute << ':' << setw(2)
        << second << (hour < 12 ? "AM" : "PM")
}
int Time :: getHour() const
{   return hour ;
}
int Time :: getMinute () const
{   return minute ;
}
int Time :: getSecond () const
{   return second;
}
int main ()
{
    Time t ;
00:00:00 ← t.printUniversal();
12:00:00 AM ← t.printStandard();
    t.setTime(13,27,6);
13:27:06 ← t.printUniversal();
1:27:06 PM ← t.printStandard();
```

// check for valid // hour, minute, second

```
t.setTime(99,99,99);
t.printUniversal();  → 00:00:00
t.printStandard();   → 12:00:00 AM
return 0;
```

```
Time sunset ;
Time & objRef = sunset ;
Time & DinnerTime = sunset ;
DinnerTime . hour ;
Time * objptr = & sunset ;
objptr → hour ;
const Time t = (4, 15, 10) ;
```

⊛ Write a program to create a counter class for incrementing the counter, decrementing the counter by using member functions and pointers to access member functions.

⊛ Write a C++ program to input sales figures of 12 months and print out the total amount sale.

```
# include <iostream>
# include <iomanip>
class SalesPerson
{
    public :
        SalesPerson ();
        void getSalesFigures ();
        void setSales (int, double);
        void printAnnualSale ();
    private :
        double Sales [12];
        double totalAnnualSale ();
};
SalesPerson :: SalesPerson ()
{
    for (int i = 0 ; i < 12 ; i++)
        sales [i] = 0.0 ;
}
void SalesPerson :: getSalesFigures ()
{
    double salesFigure ;
    for (int i = 1 ; i <= 12 ; i++)
    {   cout << "Enter Sales Figures" << i << ":" ;
        cin >> salesFigure ; setSales (i, salesFigure);
}   }
```

```
Void SalesPerson :: setSales (int month, double amount)
{
    if (month >= 1 && month <= 12 && amount > 0.0)
        sales [month - 1] = amount;
    else
        cout << "Invalid month or sales figures" << endl;
}
Void SalesPerson :: printAnnualSale ( )
{
    cout << setprecision (2) << fixed << "Total Annual Sales"
         << totalAnnualSale ( ) << endl;
}
double SalesPerson :: totalAnnualSale ( )
{
    double total = 0.0;
    for (int i = 0; i < 12; i++)
        total += sales [i];
    return total;
}
int main ( )
{
    SalesPerson S;
    S.salesFigure ( );
    S.printAnnualSale ( );
    return 0;
}
```

(*) Write a Java program for manipulation of complex numbers using object oriented approach.

```
import java.util.Scanner;
public class Complex
{
    private double a;
    private double b;
    public Complex ( )
    {   a = 0.0;
        b = 0.0;
    }
    public inputComplexNumber ( )
    {
        Scanner input = new Scanner ( );
        System.out.print ("Enter real part : ");
        a = ~~Double.parseDouble~~ input.nextDouble ( );
        System.out.print ("Enter imaginary part : ");
        b = input.nextDouble ( );
    }
```

```
# include <iostream>
# include <string>
class CreateDestroy
{   public:
        createDestroy (int, string);
        ~createDestroy ();
    private:
        int absentID;
        string message;
};
CreateDestroy :: CreateDestroy ( int ID, string messagestring)
{   objectID = ID;
    message = messagestring;
    cout << "object" << objectID << "Constructor runs" << message
                                                << endl;
}
void Create (void);
CreateDestroy first (1, "global before main");
int main ()
{   cout << " main function starts execution";
    CreateDestroy second (2, " local in main");
    static CreateDestroy third (3, " local static in main");
    Create ();
    cout << " main resumes execution" << endl;
    CreateDestroy fourth (4, " local automatic in main");
    cout << " main execution ends" << endl;
    return 0;
}
CreateDestroy :: ~CreateDestroy ()
{   cout << (objectID == 1 || objectID == 6 ? "\n" : " ");
    cout << " Object" << objectID << "Destructor runs" << message
                                                << endl;
}
void Create (void)
{   cout << "Create function execution begins" << endl;
    CreateDestroy fifth (5, " local automatic in create");
    static CreateDestroy sixth ( 6," local static in create");
    CreateDestroy seventh (7," local automatic in create");
    cout << " Create function ends" << endl;
}
```

OUTPUT :

Object 1 Constructor runs (global before main)
main function begins.
Object 2 Constructor runs (local automatic in main)
Object 3 Constructor runs (static local in main)
Create function executes
Object 5 Constructor runs (local automatic in Create)
Object 6 Constructor runs (local static in Create)
Object 7 Constructor runs (local automatic in Create)
Create function ends.
Object 7 is deleted (Destructor called)
Object 5 is deleted (Destructor called)
Object 4 is deleted (Destructor called)
Object 2 is deleted (Destructor called)
Object 6 is deleted (Destructor called)
Object 3 is deleted (Destructor called)
Object 1 is deleted (Destructor called).

POLYMORPHI

① Ope
② Fun

for (i=0;
area

Data Conv

destination b

① Basic
② Basic
③ One u

Ex) Write
type o
(
(

#inclu
const
class D
{ pri

pub

POLYMORPHISM :

1. Operator Overloading.
2. Function Overloading.

```
for (i=0; i< no_of_shapes ; i++)
    area_of_shape[i] = shape[i].compute_area();
```

Data Conversion -

destination ← b = a ; ← source

1. Basic types.
2. Basic type to user defined types and vice-versa.
3. One user-defined type to another user-defined type.

Ex) Write a c++ program to convert a user defined type distance into float type meters.

1. One argument constructor.
2. Overloaded conversion function.

```
#include <iostream>
const float m = 3.28;
class Distance
{
    private:
        int feet;
        float inches;
    public:
        Distance() { feet = 0; inches = 0.0;}
        Distance (int d, float in) { feet = d; inches = in; }
        Distance (float meters)
        { float fltfeet = m * meters;
          feet = int (fltfeet);
          inches = 12.0 (fltfeet - feet)
        }
```

```
void get_distance ()
{
    cout << "\n Enter feet:" ;
    cin >> feet ;
    cout << "\n Enter inches : ";
    cin >> inches ;
}
void show_dist ()
{
    cout << feet << "\'-" << inches << "\"";
}
operator float ()
{
    float fractfeet = inches/12.0 ;
    fractfeet += float (feet) ;
    return (fractfeet/m) ;
}
};
```

```
operator unary_operator_symbol ( )
{
}
```

```
v operator binary_operator_symbol (right hand side operand to operate)
return_datatype {
                }
```

```
void main( )
{
    Distance dist1 = 2.35 ;
    dist1 = 1.00
    Distance dist2 (5, 10.25) ;
    Distance dist3 [redacted] ;
    float meters = float (dist 2);
}
```

Conversion Between Different User-defined Data types -

object a = object b ;
↓ destination ↳ source

So, if the conversion routine is in the source class, we use overloaded conversion function.
So, if the conversion routine is in the destination class, we use overloaded argument constructor.

Ex.) Write a
polar coor
system a
(r, θ
x =
r =

```
# include
# include
class Pola
{ privat
public
};
class Rec
{ privat
d
d
public
};
```

Ex.) Write a C++ program to convert a point in polar coordinate system into cartesian coordinate system and vice-versa.

(r, θ) ; (x, y)

$x = r\cos\theta$; $y = r\sin\theta$

$r = \sqrt{x^2+y^2}$; $\theta = \tan^{-1}(y/x)$

```
#include <iostream>
#include <math>
class Polar
{   private:
        double radius;
        double angle;
    public:
        Polar() { radius = 0.0 ; angle = 0.0;}
        Polar (double r, double a) { radius = r ; angle = a ; }
        double getr() { return radius; }
        double geta() { return angle; }
};
class Rec
{   private:
        double xco;
        double yco;
    public:
        Rec() { xco = 0.0 ; yco = 0.0;}
        Rec(double x, double y) { xco = x ; yco = y ; }

        Rec ( Polar p)
        {   float r = p.getr();
            float a = p.geta();
            xco = r cos(a);
            yco = r sin(a);
        }
        Operator Rec()
        {   double x = radius * cos(angle);
            double y = radius * sin(angle);
            return Rec(x,y);
        }
};
```

```
void main ()
{
    Rec rec;
    polar pol(10.0, 0.785398);
    rec = pol;
}
```

---

(1) For unary operator, no argument.

(2) For binary, one argument representing right hand of the operand of the operator. The left side operand is implicitly used by the compiler.

(3) By using non-static or global functions, all arguments are to be provided.

```
<datatype> operator <symbol> ( ) {
    }
```
} unary.

```
<datatype> operator <binary symbol> (<RHS operand>) {
    }
```

Ex.) Overload ++ (increment operator).

```
# include <iostream>
class Counter
{
    private:
        unsigned int count;
    public:
        Counter ( ) { count = 0; }
        int get_count ( ) { return count; }
        Counter operator ++ ( )
        {
            count ++;
            Counter temp;
            temp.count = count;
            return (temp);
        }
```

```
void main ( )
{
    Counter c1 , c2 ;
    c1++ ;  ++c2 ;
    c2 = c1++ ;
    c2 = ++c1 ;
    cout << " \n c1" << c1.get_count ( ) ;
    cout << " \n c2" << c2.get_count ( ) ;
}
};
```

Ex) Overload + to add two points in polar coordinate sys

$P_1(r_1, \theta_1)$; $P_2(r_2, \theta_2)$

$P_3 = P_1 + P_2$

$x_1 = r_1 \cos\theta_1$; $x_2 = r_2 \cos\theta_2$

$y_1 = r_1 \sin\theta_1$; $y_2 = r_2 \sin\theta_2$

$x_1 + x_2 = r_1 \cos\theta_1 + r_2 \cos\theta_2$

$y_1 + y_2 = r_1 \sin\theta_1 + r_2 \sin\theta_2$

$$r = \sqrt{(x_1+x_2)^2 + (y_1+y_2)^2}$$

$$\theta = \tan^{-1}\{(y_1+y_2)/(x_1+x_2)\}$$

Note : Ternary Oper[illegible] cannot be overloaded. &, * → also cannot be overloaded.

```
Polar operator + (Polar P2)
{   float x = getx() + P2.getx();
    float y = gety() + P2.gety();
    float r = sqrt(x*x + y*y);
    float a = atan(y/x);
    return (Polar(r, a));
}
```

```
Polar()
{   r = 0.0;
    a = 0.0;
}
```

X

```
class Polar
{ private:
    float r;
    float a;
  public:
    Polar() { r = 0.0; a = 0.0; }
    Polar(float r1, float a1) { r = r1; a = a1; }
    void display()
    {   cout << "r = " << r << "a = " << a;
    }
    void get_polar()
    {   cout << "Enter r :"; cin >> r;
    }   cout << "Enter a :"; cin >> a;
```

```
void main ()
{
    Polar P1, P2;
    Polar P3 = P1 + P2;
    display ();
}
};
```

(*) Overload +, <, += for objects of type distance.

```
#include <iostream>
class Distance
{  private:
       int feet;
       float inches;
   public:
       Distance () { feet = 0; inches = 0; }
       Distance (int f, float in) { feet = f; inches = in; }
};
```

← `Distance operator + (Distance d2)`

```
{
    int f = feet + d2.feet;
    float in = inches + d2.inches;
    if (in >= 12.0)
    {   in -= 12.0;
        f++;
    }
    return (Distance (f, in));
}
```

`bool operator < (Distance d2)` d1 < d2 → true, else → false.

```
{
    float lf1 = feet + inches/12.0;
    float lf2 = d2.feet + d2.inches/12.0;
    return (lf1 < lf2) ? true : false;
}
void ~~Distance~~ operator += (Distance d2)
{
    feet += d2.feet;
    inches += d2.inches;
    if (inches >= 12.0)
    {   inches -= 12.0;
    }   feet++;
}
```

Overloading << and >> -

```
#include <iostream>
#include <string>
class PhoneNumber
{
    private:
        String areacode;
        String exchange;
        String line;
};
friend ostream operator << (ostream &, const Phonenumber &);
friend istream operator >> (istream &, Phonenumber &);

ostream & operator << (ostream &output, const Phonenumber & number)
{
    output = '(' << number.areacode << ')' << number.exchange
                 << number.line;
    return output;
}
istream & operator >> (istream &input, Phonenumber &number)
{
    input >> ignore() >> areacode >> ignore() >> exchange >>
             line;
    return 0;
}
```

* friend classes can access private datamembers of each other directly.

Q) Take a string class object. Overload + (concatenation), == (compare) over strings taken as objects as a character away.

Q) Overload functions to find the square of an integer and a float.

Function Overloading -

```
# include <iostream>
void repeatchar (char = '*', int 45);
void main ()
{   repeatchar ();
    repeatchar ('=');
    repeatchar ('+', 30);
}
void repeatchar (char ch, int n)
{   for (int j=0; j<n; j++)
        cout << ch;
    cout << endl;
}
```

```
float fixed_deposit (int k);
int fixed_deposit (float k);
```
} It is not an example of function overloading.

```
# include <iostream>
float add (float x, float y);
class Number
{   private:
        int i, j;
    public:
        Number () { i=0; j=0; }
        int add (int i, int j);
};
int main ()
{   int k, m=3, n=4;
    float z, a=3.5, b=5.5;
    Number nn;
    k = nn.add (m, n);
    z = add (a, b);
    return 0;
}
```

```
int Number :: add (int i, int j)
{   int temp;
    temp = i+j;
    return temp;
}
float add (float x, float y)
{   float temp;
    temp = x+y;
    return (temp);
}
```

POLYMORPHISM :

① Assignment ; ② Initialization.

$a_2 = a_1$; } default copy constructor.
$a_3 = (\,)$; }

alpha $a_4 = a_3$

```
#include <iostream>
class alpha
{
    private:
        int data;
    public
        alpha ();
        alpha (int d) { data = d;}
        void display () { cout << data; }
        alpha operator = (alpha &a).
        {
                data = a.data;
                cout << "Assignment Operator is invoked";
                return (alpha(data));
        }
};
void main()
{
      alpha a1(37);
      alpha a2;
      cout << "\a2 = ";
      a2.display();
      alpha a3 = a2;
      cout << "\n a3 = " ; a3.display();
}
```

$a_1 = 37$;

Assignment operator invoked $a_3 = 37$;

$a_2 = 37$:

$a_3 = 37$;

```
alpha &operator (alpha &)
{

}
```

```
// copy constructor
alpha (alpha &a)
{   data = a.data;
    cout << " copy constructor is called";
}   return (*this);
```

Arrays -

1. Range checking is not possible.
2. Indices are integers for an array of size n.
   (0, 1, 2, ..., n-1) ⇒ other indices cannot be used.
3. Entire array cannot be input/output.
4. Two arrays can not be compared by == or relational operators.
5. When an array is passed to a function, the size of the array is also to be passed.
6. When array can not be assigned to an other array.

```
#include <iostream>
class Array
{friend ostream & operator << (ostream &, const Array &);
friend istream & operator >> (istream &, Array &);

public:
    Array (int = 10);
    Array (const Array &);
    ~Array ();
    int setsize () const;
const Array & operator = (const Array &);
bool operator == (const Array &) const;
bool operator != (const Array &right) const
  {  return ! (*this == right);
  }
int & operator [] (int);
int operator [] (int) const;
private:
    int size;
    int *ptr;
};
```

```
Array :: Array ( int arraysize)
{
    size = ((arraysize > 0) ? arraysize : 10) ;
    ptr = new int [size] :
    for (int i=0 ; i< size ; i++)
        ptr [i] = 0 ;
}
```

Copy Constructor -

membership assignment operator

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

Destructor -

```
Array :: ~Array()
{    delete [] ptr ;
}
int Array :: getSize ()
{   return size ;
}
```

Overloaded Assignment Operator -

```
const Array & Array :: operator = (const Array & right)
{  if ( & right != this)        [Self assignment check]
   {   if( size != right.size)
       {   delete [] ptr ;
           size = right.size ;
           ptr = new int [size] ;
       }
       for (int i=0 ; i< size ; i++)
           ptr [i] = right.ptr [i] ;
   }return 0;
}
```

```
// Overload ==
bool Array :: operator == (const Array &right) const;
{   if (size != right.size)
        return (false);
    for (int i=0 ; i<size ; i++)
        if (ptr[i] != right.ptr[i])
            return (false);
    return (true);
}

// Overloaded >>
//    cin >> *
//     ↑      ↘
//  istream    user operand
//   class       class.

// istream overloading
istream & operator >> (istream & input, Array &a)
{   for (int i=0 ; i < a.size ; i++)
        *(input) >> a.ptr[i];
    return input ;
}
// ostream overloading
ostream & operator << (ostream & output, const Array &a)
{   int i ;
    for (i=0; i<a.size ; i++)
    {   output << setw(12) << a.ptr[i];
        if ((i++) % 4 == 0)
            output << endl ;
    }
    if (i % 4 != 0)
        return output;
}
```

```
// Subscript Operator
int & Array :: operator [] : (int subscript)
{   if ( subscript < 0  || subscript >= size)
    {
        cerr << subscript << "out of range" << endl;
        exit (1);
    }
    return (ptr[subscript]);
}
```

```
* int Array :: operator [] (int subscript) const
{
    if (subscript < 0 || subscript >= size)
    {   cerr << subscript << "out of range" << endl;
        exit (1);
    }
    return ptr [subscript];
}
int main ()
{
    Array integers1 (7);
    Array integers2;
    cout << "Size of integer1 is" << integer1.getSize ()
         << "\n Array after initialization : \n" << integer1;
    cout << "Size of integer2 is" << integer2. getSize() <<
         "\n Array after initialization : \n" << integer2;
```

Output :

```
Size of integer1 is 7
Array after initialization is  0 0 0 0 0 0 0
```

```
    cout << "Enter 17 elements" << endl;
    cin >> integer1 >> integer2;
    Array integer3 (integer1);
}
```

```
#include <iostream>
#include <cstring>
#include <iomanip>
#include <cstdlib>
class String {
   friend ostream & operator << (ostream &, const String &);
   friend istream & operator >> (istream &, String &);
public:
   String (const char *= " ");
   String (const String &);
   ~String ();
   const String & operator = (const String &);      // assignment
   const String & operator += (const String &);     // concatenation
   bool operator !() const;                         // test whether string is empty
   bool operator == (const String &) const;         // check equality
   bool operator < (const String &) const;
   bool operator != (const String &right) const;
   bool operator > (const String &right) const;
   bool operator <= (const String &right) const;
   bool operator >= (const String &right) const;
   char & operator [] (int);
   char operator [] (int) const;
   String operator () (int, int = 0) const;
   int getLength () const;
private:
   int length;
   char [redacted] *sptr;
   void setString (const char *);
};
```

- (1) char *strcpy (char *s1, const char *s2);
- copies the string s2 into the character array s1. The value of s1 is returned.

(2) char *strncpy (char *s1, char *s2, size_t n);
- copies atmost n characters of string s2 into s1. The value of s1 is returned.

(3) char *strcat (char *s1, const char *s2);

(4) int strcomp (const char *s1, const char *s2);

(5) int strncomp (const char *s1, size.t n);

```
ostream & operator << (ostream &output, const String &s
{   output << s.sptr;
    return output;
}
istream & operator >> (istream & input, String & s)
{   char temp[100];
    input >> setw(100) >> temp;
    s = temp;
    return input;
}
String :: String (const char * s): length ((s!=0 ? strlen(s):0)
{   cout << "constructor" << s << endl;
    setString (s);
}
String :: set-String (const char *String2)
void {  sptr = new char [length+1];
        if (string2 != 0) strcpy (sptr, string2);
        else sptr[0] = '\0';
    }
```

Copy Constructor -

```
String :: String (const String & copy): length (copy.length)
                                          ↳ (initializer)
{   cout << "Copy Constructor" << copy.sptr << endl;
    setString (copy.sptr);
}
String :: ~String ()
{   cout << "destructor" << sptr << endl;
    delete[] sptr;
}
```

Assignment Operator -

```
const String & String :: Operator = (const String &right)
{  cout << "Assignment operator" << endl;
   if (&right != this)
   {   delete [] sptr;
       length = right.length;
       setString (right.sptr);
   }
   else
       cout << "Attempted assignment of a string to itself" <<
                                                        endl;
   return *this;
}
```

Concatenation Overloaded Operator (+=) -

```
const String & String :: operator += (const String &right)
{  size_t newlength = length + right.length;
   char *tempptr = new char [newlength + 1];
   strcpy (tempptr, sptr);
   strcpy ( tempptr + length, right.sptr);
   delete [] sptr;
   sptr = tempptr;
   length = newlength;
   return *this;
}
bool String :: operator ! () const
{    return length == 0;
}
bool String :: operator == (const String &right) const
{    return strcmp (sptr, right.sptr) == 0;
}
char & String :: operator [] (int subscript) // modifiable
{
   if (subscript < 0 || subscript >= length)
   {   cerr << "Subscript" << subscript << "out of range" << endl;
                                                    exit(1);
   }
```

```
    return sptr[subscript];
}
```

Overloaded Substring Operator -

```
String String :: Operator () (int index, int sublength) const
{
    if (index < 0 || index > length || sublength < 0)
        return "";
    int len;
    if ((sublength == 0) || (index + sublength > length))
        len = length - index;
    else
        len = sublength;
    char *tempptr = new char [len+1];
    strcpy (tempptr, & sptr[index], len);
    tempptr[len] = '\0';
    String tempString (tempptr);
    delete [] tempptr;
    return tempString;
}
int String :: getLength () const
{   return length;
}
int main ()
{
    String s1 ("happy");
    String s2 ("birthday");
    String s3;
    cout << "s1 is : \" << s1 << "\" << "s2 is :" << "\" <<
            s2 << "\" << "s3 is \" << boolalpha <<
            "\n\n The result of comparing s1 and s2"
<< "\n" << (s1 == s2) << "s2 != s1" << (s2 != s1) << "s2 > s1"
<< (s2 > s1) << "s2 >= s1" << (s2 >= s1) << "s2 <= s1" << (s2 <= s1)
                                                    << endl;
}
```

Overloading 'new' and 'delete' -

```
ptr = new char [len+1];
delete [] ptr;
```

void * operator new (size_t, numberofbytes); void operator delete (void *);

INHERITANCE :

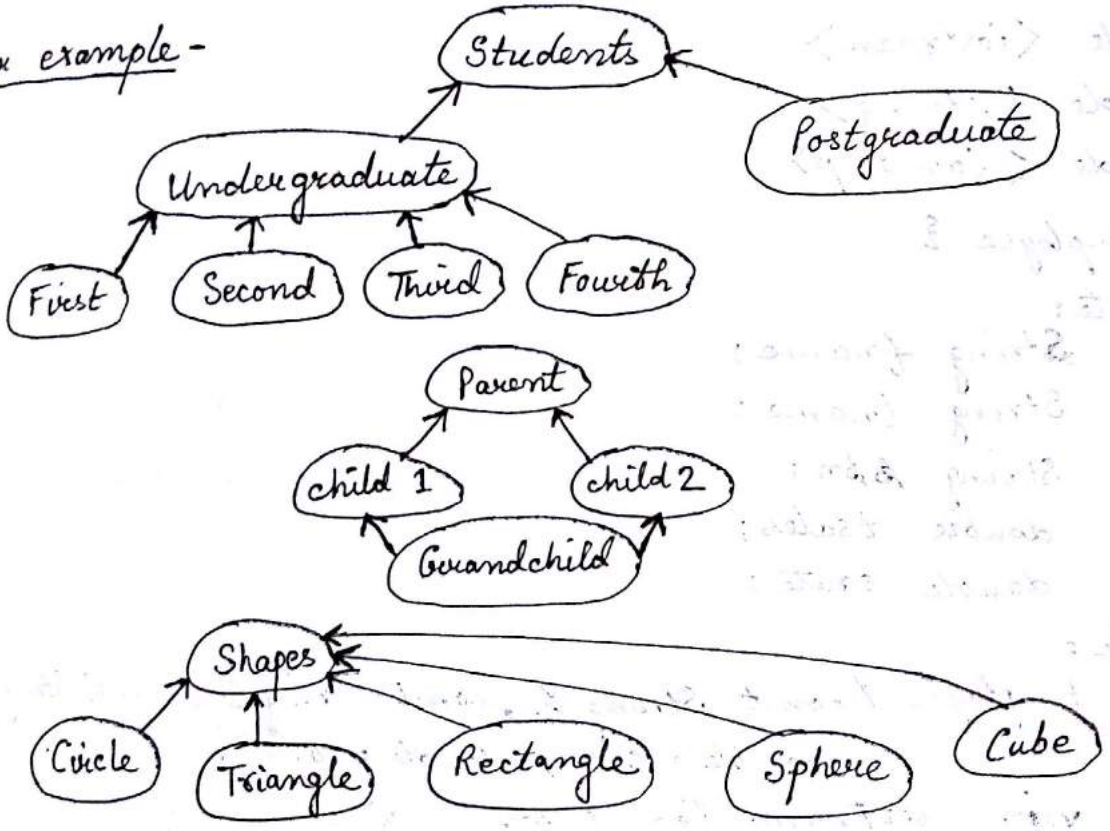
* Inheritance is a (Δ) 'is-a' relationship.
Generalization is a (Δ) 'has-a' relationship.

* Creating a class from an existing class which inherits its properties, creates new behaviour, override existing behaviour. The first class is called the base class and the class derived is known as derived class.

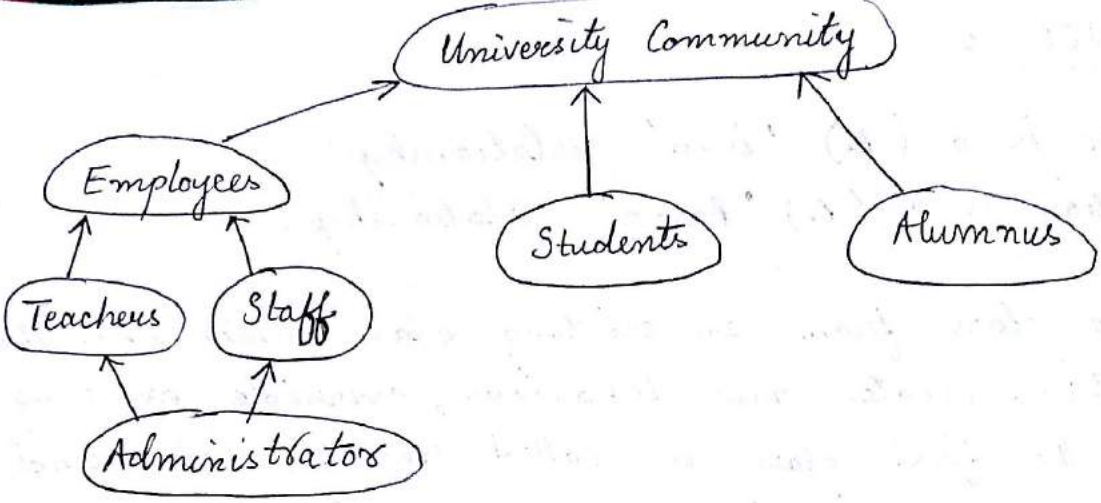
---

1. Direct → Single Level
2. Indirect → Multi Level
3. Multiinheritance → When a class is derived from more than one classes is called multiple inheritance.

For example -



Ex) Consider a university having employees, students, faculty, staff, administrator, alumnus. Draw object model showing inheritance among objects.



Ex) Consider a payroll application involving two types of employees. The first type of employee gets commission on their total sales and the second type of employees gets a basic pay in addition to commission on total sales.

```
#include <iostream>
#include <string>
#include <iomanip>
class Employee {
  private:
        String fname;
        String lname;
        String ssn;
        double tsales;
        double crate;
  public:
        Employee (const String &, const String &, const String &,
                  double = 0.0 , double = 0.0);
        void setFname (const String &);
        String getFname () const;
        void setLname (const String &);
        String getLname () const;
        void setssn (const String &);
        String getssn () const;
        void settSales (double);
        double gettsales () const;
```

```
    void setCrate (double);
    double getCrate () const;
    double earnings () const;
    void print () const;

};
Employee :: Employee (const String & fn, const String & ln,
                      const String & sn, tsales, crate)
                                         ↓double   ↓double
{
    fname = fn;      | setFname (& fn)
    lname = ln;      | setLname (& ln)
    ssn = sn;        | setSsn (& sn)
    tsales = tsales; | setTsales (& tsale)
    crate = crate;   | setCrate (& crat)
}
Employee::
void setFname (const String & fn)
{   fname = fn;
}
Employee::
String getFname () const
{   return fname;
}
Employee::
void setLname (const String & ln)
{   lname = ln;
}
Employee::
String getLname () const
{   return lname;
}
... ssn, tsales, crate.
Employee::
double earnings () const
{   return (tsales * crate);
}
void Employee :: print() const
{
    cout << "Employee-firstname" << fname << '\n' <<
            "Employee-lastname" << lname << '\n' <<
            "Employee-ssn" << ssn << '\n' << "tsales" << tsale
                                  << "crate" << crate;
}
```

```
int main ( )
{
    Employee A ("RAM", "Lal", "222-22-2222", 10000, 0.06);
    A.setTsales (8000.00)
    A.setCrate (0.1);
    cout << A.earnings ( ) << endl;
    return 0;
}
```

⊛ Class BEmployee is inherited from class Employee.

```
class BEmployee : public Employee
{   private:
        double bsalary;
    public:
        BEmployee (const String &, const String &, const String &,
                   double = 0.0, double = 0.0, double = 0.0);
        void setbsalary (double);
        double getbsalary ( ) const;
        double earnings ( ) const;   // overridden
        void print ( ) const;        // overridden
};
BEmployee :: BEmployee (const String & fn, const String & ln,
const String & sn, double tsales, double crate, double bsalary):
            Employee (fn, ln, sn, tsales, crate)
{   setbsalary (bsalary);
}
void BEmployee :: setbsalary (double salary)
{   bsalary = (salary < 0.0) ? 0.0 : salary;
}
BEmployee :: double getbsalary ( ) const
{   return bsalary;
}
```

```
double BEmployee :: earnings () const
{   return (bsalary + Employee :: earnings());
                    OR, get_Crate() * get_Tsales()
}
void BEmployee :: print () const
{   cout << ----------
    OR,
    Employee :: print()
    cout << "bsalary" << getbasesalary();
}
```

Accessing by using pointers -

```
int main()
{   Employee X("Shyam", "Lal", "222-2222", 10000.00, 0.6);
    Employee *Xptr = 0;
    BEmployee Y("Ram", "Lal", "333-3333", 500.00, 0.04, 300.0
    BEmployee *Yptr = 0;
    cout << "Base class object" << X.print();
    cout << "Derived class object" << Y.print();
    Xptr = &X;
    Xptr → print();
    Yptr = &Y;
    Yptr → print();
    Xptr = &Y;
    Xptr → print();
}
```

Virtual functions -

```
virtual print() const
{
}
```

Pure Virtual function -

```
virtual func() = 0;
```

Abstract Classes -

A class is called abstract if it has at least one pure Virtual function.

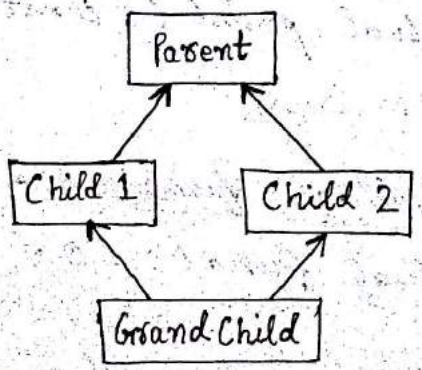
Geometric Shapes -

1. one - dimensional
2. two - dimensional
3. three - dimensional.

```
class Shape
{     virtual draw () = 0;
};
class Triangle :: public Shape();
class Circle :: public Shape();
 :
Shape *ptrarr[100];
for (int j=0 ; j<N ; j++)
     ptrarr[j] ⟶ draw();
```

Ex.) Develop an OOP having two classes Student and Professor using a pure Virtual function is outstanding() for creating a list of students and professors for some awards. A student is selected if he scores more than 9 cgpa. A professor is selected if ~~he~~ publishes more than 100 research papers.

Multiple Inheritance :-




```
class Parent
{   private :
        int basedata;
    public :
        int getdata()
        {   return basedata;
        }
};
class Child1 : public Parent { };
class Child2 : public Parent { };
class GrandChild : public Child1 { };
                   public Child2 { };
```

Friend functions -

When private data part of one class is accessed by member functions or object of other classes, it is possible if both functions or classes are declared as friend of each other.

```
class beta;
class alpha
{   private: int data;
    public: alpha() { data = 3; }
    friend int Gamma(alpha, beta);
};
class beta
{   private: int data;
    public: beta() { data = 7; }
    friend int Gamma(alpha, beta);
    int Gamma(alpha a, beta b)
    {   return (a.data + b.data);
    }
};
```

① Friend functions can appear anywhere.
② Corresponds to more natural form of functions even if they are overloaded.

Friend Functions -

```
#include <iostream>
class Distance
{   private: int feet;
            float inches;
    public: Distance() { feet = 0; inches = 0.0; }
            Distance(float fltfeet)
            {   feet = int(fltfeet);
                inches = 12 * (fltfeet - feet);
            }
            Distance(int flt, float in) { feet = ft;
                                          inches = in; }
```

```
    friend Distance operator + (Distance, Distance);
    friend Distance Square(Distance);
};
Distance Distance :: Square (Distance d)
{
    float fltfeet = d.feet + d.inches/12 ;
    float feetsquare = fltfeet * fltfeet ;
    int f = int (feetsquare)
    float i = 12*( feetsquare - f);
    return Distance (f, i);
}
```

Static functions -

```
class Gamma
{
  private:
    static int total;
    int id;
  public: Gamma () { total ++ ; id = total ; }
    ~Gamma () { total -- ;
                cout << "Destroying id number = " << id;
              }
    static void Showtotal ()
    {  cout << " Total is " << total;
    }
    void showid ()
    {  cout << " Id number is " << id ;
    }
};
void main ()
{
    cout << endl << endl;
    Gamma g1 ;
    Gamma :: Showtotal ();
    Gamma g2, g3 ;
    Gamma :: Showtotal ();
    g1. showid ();
    g2. showid();
    g3. showid ();
    cout << "End of the program";
}
```

Total is 1 ←

Total is 3 ←

g1. showid (); → Id no. is 1
g2. showid(); → " " 2
g3. showid (); → " " 3

```
Destroying id no. = 3
        "     "   2
        "     "   1
```

Templates -

① Generic Programming - It is a form used to represent function overloading for functions which have similar algit but different datatypes. The codes generated are known as template specializations.

(i) Function template	(ii) class template
function template specializations	↳ class is written once and class specializations for different data types are created by compiler

template < Typename/class T > → parameter used as a place holder and can be return type parameter or arguments.

Ex) template < class Elementtype >
template < typename ordertype , filetype >

Ex) Write a function template for finding the maximum of three values.

```
#include <iostream>
template < Typename T >
T maximum ( T value1 , T value2 , T value3 )
{  T maximumvalue = value1 ;
   if (value2 > maximumvalue)
      maximumvalue = value2 ;
   if (value3 > maximumvalue)
      maximumvalue = value3 ;
   return maximumvalue ;
}
```

```
#include <iostream>
template < Typename T >
void printarray (const T *array , int const)
{  for( int i=0 ; i<count ; i++)
      cout << array[i] << " "
   cout << endl ;
}
```

```
int main ()
{
    const int ACOUNT = 5 ;
    const int BCOUNT = 7 ;
    const int CCOUNT = 6 ;
    int a [ACOUNT] = {1, 2, 3, 4, 5}
    double b [BCOUNT] = { 1.1, 2.2, 3.3, 4.4, 5.5} 6.6, 7.7 }
    char c [CCOUNT] = "HELLO"
    cout << "Array a contains" << endl ;
    printarray (a, ACOUNT)
    cout << " Array b contains" << endl ;
    printarray (b, BCOUNT)
    cout << " Array c contains" << endl ;
    printarray (c, CCOUNT) ;
    return 0;
}
```

(*) Consider a stack class independent of the type of arguments contained in it.

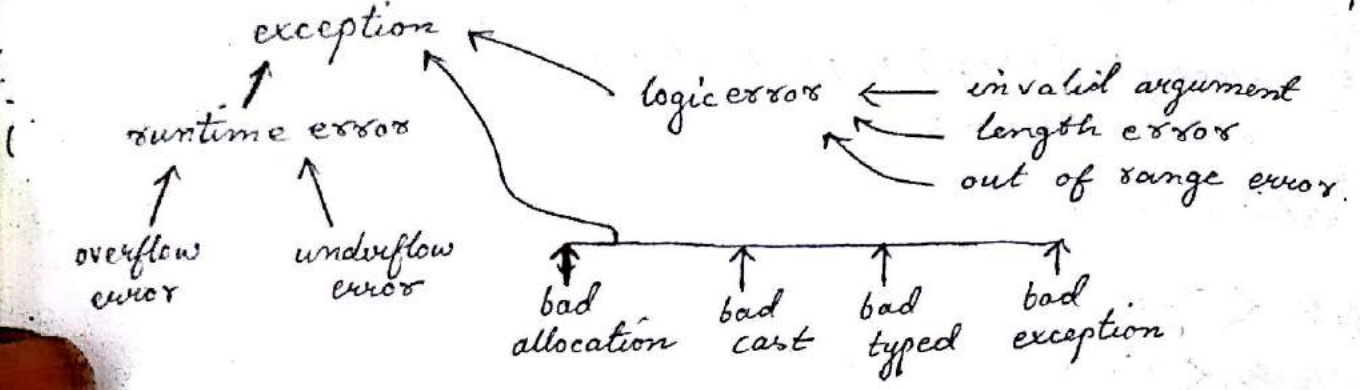
```
template <Typename T>
class Stack
{   public :
        Stack (int = 10) ;
        ~Stack () { delete [] stackptr ; }
        bool push (const T &)
        bool pop (T &)
        bool isEmpty () const
        {   return top == -1 ;
        }
        bool isFull ()
        {   return top = size - 1 ;
        }
    private:
        int size ;
        int top ;
        T *stackptr ;
};
```

```
template <Typename T>
Stack <T> :: Stack (int s): size (s>0 ? s:10)
                top(-1), stackptr new T[size];

template <Typename T>
bool Stack <T> :: push (const T & pushvalue)
{   if (! isFull ())
    {   stackptr [++top] = pushvalue;
        return true;
    }
    return false;
}
template <Typename T>
bool Stack <T> :: pop (T &popvalue)
{   if (! isEmpty ())
    {   popvalue = stackptr [top--];
        return true;
    }
    return false;
}
int main ()
{   Stack <double> doublestack (5);
    double doublevalue = 1.1;
    cout << " Pushing element onto double stack "<<
    while (doublestack.push (doublevalue))
    {   cout << doublevalue <<" ";
        doublevalue += 1.1;
    }
    cout <<"\n Stack is full. Cannot push "<<
        doublevalue << "\n\n popping elements from doublestack";
    while (doublestack.pop (doublestack))
        cout << doublevalue <<" ";
    cout <<" Stack is empty, cannot pop \n";
}
```

Exception Handling -

Ex) Divide by zero,
Failure to allocate memory by new invalid function parameters, array range out of bound exception
<stdexcept>



Exception is an indication of error during the running time of a program. The exceptions are not allowed to occur but the errors are reported to users.

(1) Fault Tolerant and robust programs.

Eg. (i) Division by zero
(ii) Array range out of bounds.
(iii) Invalid parameters in the function call.
(iv) Unable to allocate memory by new.

Division by zero - program terminates if integer a is divided by b=0 for floating point division, the program continues with INF or -INF as the value of division. This gives wrong results.

---

* C++ standard class library <stdexcept> contains all types of exceptions.

* Suppose we want to handle Divide By Zero exception for integers m and n (n/m)

```
double quotient (int n, int m)
{  if (m == 0) throw DivideByZeroException ();
   return static-cast <double> (n)/m ;
}
```

Note: ① Throws - to throw the exception which may be any type, a group of types or having exception hierarchies by inheritance.

```
# include <iostream>
# include <stdexcept>
class DivideByZeroException : public runtime_error
{   public:
    DivideByZeroException :: DivideByZeroException () :
            run-time_error ("attempted to divide by zero")
            {
            }
};
int main ()
{   int n1;
    int n2;
    double result;
    while (cin >> n1 >> n2)
    {
        try
        {   result = quotient(n1, n2);
            cout << "result" << result <<
                                endl;
        }
        catch (DivideByZeroException & dividebyzeroexception)
        {   cout << "Exception occured" << dividebyzeroexception
                                .what() << endl;
        }
        cout << "Enter two integers";
                "To terminate the program enter return"
    }
    cout << endl;
    return 0;
}
```

→ virtual function

* when input is a condition, it returns a reference to a pointer whose value is either null or not null giving 0 or 1 (false/true).

Note: More than one catch handler can be used to handle the exceptions.

INPUT / OUTPUT (iostream) -

extraction ↑ >> , insertion ↑ <<

cin , cout , cerr and clog with >> , <<

- cin → input from standard device. (keyboard)
- cout → output from standard device. (monitor)
- cerr → unformatted errors in the keyboard
- clog → formatted errors on the log file.

① Unformatted - A sequence of bytes are transferred from device to memory and vice-versa. High speed, high volume data transfer.

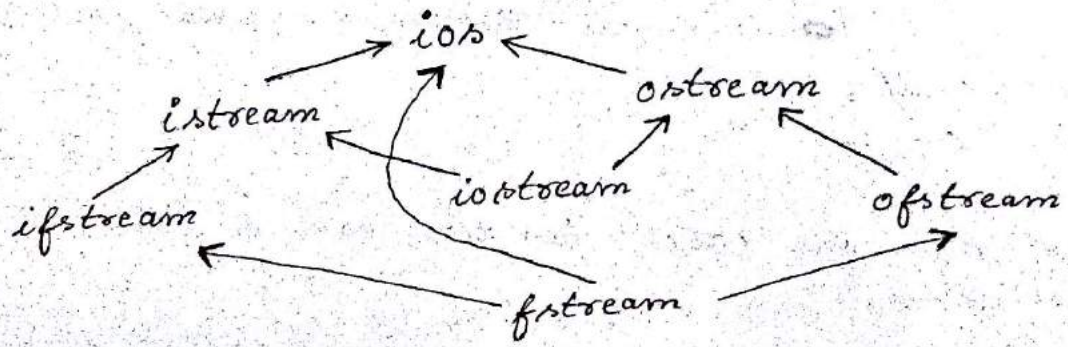
② Formatted - A group of bytes are used having appropriate means say integer, string or floating point quantity used for transfer.

I/O is carried out in terms of streams which are a sequence of bytes. The information contained in these bytes can be interpreted by the program which access them.

For carrying out I/O, one should include iostream header file.

```
# include <iostream>
# include <iomanip>   (manipulators such as setw(), precision(), fixed, internal ... for formatted data)
# include <fstream>
```

↓ for file processing



Stream output (ostream) -

C++ determines datatypes automatically. For example, suppose you want to output char * (it is a pointer to the character string terminated by '\n')

```
char *word = "again";
cout << "String is : " << word;   ⇒ again
```

Suppose we want to output the address of the first character of the string used.

<void *> outputs the address of the first character of the string.

```
cout << "Address of string is : " static-cast <void *> word
                                   (in hexadecimal)
```

put() used to output char

```
cout.put('A')          ⇒ A
cout.put('A').put('$') ⇒ A$
cout.put(65)           ⇒ A
```

Stream input is carried out by cin, >>

It basically skips white space character on input stream and returns a reference to cin object. Unformatted input is carried out by get() member function which inputs a character. [cin.get();]

cin.eof(); ⇒ this would return 0 if end-of-file is encountered on input stream.

cin.getline (letter, size, '$'); → delimiter

cin.get (size);

---

① ignore() member function of istream either reads or discards a designated number of characters or terminates upon entering delimiters such as EOF which forces ignore() to skip to the end of file from reading a file. ex. ignore() >> x

↳ 1 char by default.

② **peek()** → returns the next character from the input stream but does not remove it from the stream.

③ **putback()** → places the previous character obtained by get() back on to the stream.

Unformatted I/O using read(), write() and gcount().

read() inputs some number of bytes to a character array and **write()** outputs some number of bytes from a character array. These bytes are not formatted and null characters if any are displayed.

```
char letter[] = "HAPPY BIRTHDAY"
cout.write(letter, -5);   => write chars from the letter array from the beginning.
cin.read(letter, 10);
```

gcount() gives the no. of chars input/output in the last operation.

Formatted I/O is done by manipulators -

① integral stream base (integers are treated as decimals)
To change the base 10 to hex or octal, we hex or oct.

```
int number = 12;                                    showbase()
cout << number;          => 12                      setbase(10)
cout << hex << number;   =>
cout << oct << dec << number;   => 12
cout << showbase() << number;   => 10 12
cout << showbase() << hex << number;
```

② Floating point precision ⇒ no. of digits to the right of decimal point.

Ex. setprecision(2)

```
double root = sqrt(2.0)
int places = 2;
cout << fixed << precision(places) << root;   (1.41)
```

decimal point should also appear

③ Field Width

setwidth() or width() or setw()

cin.width(5); ⇒ input data into 5 places.

cin >> setw(5) >> x;

cout << setw(12) << x; if x is larger than 12 places, the number will be displayed as it is.

---

① skipws / noskipws

② left

③ right

④ internal ⇒ sign is left justified and magnitude is right justified.

cout << internal << setw(12) << x;

x = ● 12 ;
+ ..... 12 .

⑤ showbase()

⑥ showpoint()

⑦ showpos()

⑧ uppercase ⇒ specify the uppercase letter.
should be used in a hexadecimal number and E should be used in scientific notation for floating point number.

x = 1.94699 cout << scientific << x;

x = 0.194699e01.

cout << nouppercase << scientific << x;

0.194699e01

⑨ fixed.

Stream Manipulators -

① integral stream base - dec, oct, hex
(10) (0) (0x)

- (showbase())

```
int number = 12;
cout << dec << number << endl;
     0xC    012      12
cout << dec << oct << number;
        012
cout << showbase(10) << number;
                        12
```

② Field Width (setwidth() / setw() / width())

```
cin.width(5)
cout << setw(10) << number;
        # # # # # # # # 1 2
```

③ Floating number precision (implies number of digits to the right of decimal point.)

setprecision() / precision()

```
double root = sqrt(2.0);
int places = 2;
cout << fixed;
cout.precision(places)
cout << root << endl;
```

} cout << fixed << setprecision(places) << root << endl;

---

① skipws / noskipws

② left / right (padding elements appear left / right)

③ internal → cout << internal << setw(10) << number;
+ # # # # # # # 1 2

④ showpoint → to show the fixed point.

⑤ showpos → specifies that (+) will proceed the positive number

⑥ scientific → Ex. 1.946 e9 0.1946 e 10

```
float p = 1.946
cout << scientific << setw(10) << p;   # # 0.1946e1
```

⑦ uppercase → cout << scientific << setw(10) << uppercase << left << p;
0.194600 E1

Note: If neither fixed nor scientific notation is specified,

ex. 9.99000 → 9.99
9.9000 → 9.9
9.000 → 9.0

⑧ fill() or setfill()

```
int x = 10000;
cout << right << fill('*') << setw(10) << x;
```

*****10000

```
cout << left << setfill('*') << setw(10) << x;
```

10000*****

⑨ boolalpha / noboolalpha → specifying boolean formal. (0 - false, 1 - true)

```
bool boolvalue = "False";
cout << boolapha << boolvalue;
```

Stream Error State -

The state of the stream may be tested through a number of bits by using rdstate() member function.

① the end of file bit is set when end-of-file (EOF) is encountered on input/output stream.

eof = 1/0 , eof() , cin.eof();

② The fail bit is set when there is a format error, that is a non-digit character appears when reading an integer. failbit() → cin/cout.failbit();

③ The bad bit is set if the device failure occurs.

badbit() → cout.badbit();

cin.rdstate();
cout.rdstate();

④ The good bit is set if badbit, fail bit and eof bits are not set. goodbit();

File Processing - <fstream>, <iostream>, <iomanip>

① create/read/update files.

→ (i) sequential files (ii) random files.

C++ recognizes a file as a sequence of bytes with no structure. The programs which process these files interpret their contents.

A file is a sequence of logically related records. The record is a group of fields where a field can be identified as a key-field.

Ex. employee file → a record for each employee. The record has the form - (acc_no, name, salary, age, ...)

⊛ Creating a Sequential File -

```
#include <iostream>
#include <fstream>
int main()
{
    ofstream outclientfile ("client-file", ios::out);
                                          ⇒ mode
    if (!outclientfile)
                ⇒ overloaded operator
    {
        cerr << "file could not be
             opened" << endl;
        exit(1);
    }
    cout << "Enter clients records
         with EOF when end-of-file" <<
         endl;
    int acct_no;
    char name[40];
    double balance;
    while (cin >> acct_no >> name >> balance)
    {
        outclientfile << acct_no << name << balance << endl;
        cout << '%';
    }
    return 0;
}
```

Mode -

- ios::app → open and append record at the end of file.
- ios::ate → open a file for output and move to the end of the file.
- ios::in → for input
- ios::out → for output.
- ios::trunc → discard the file contents if it appears
- ios::binary → open and create a binary file.

① Suppose records entered are not in order, collect all the records and sort it prior to writing on to the file.

② Possible errors while writing are-
(i) Attempting to open a non-existent file.
(ii) Attempt to open a file for reading or writing without permission
(iii) Opening a file when no disk space is available.

③ Read the file :

⇒ open the file for input

```
ifstream inclientfile ("client.dat", ios :: in) ;
```

OR,

```
ifstream inclientfile;
inclientfile.open ("client.dat", ios :: in);
while ( inclientfile >> acct_no >> name >> balance)
        outline (acct_no , name, balance) ;
void outline ( int acct_no , const String name, double balance)
{
    cout << left << setw(10) << acct_no << setw (13) << name
    << setw(7) << precision (2) << right << balance << endl;
```

getpointer - places the pointer to the record to be input.
putpointer - places the pointer to the position where record is to be written.

seekg() - same as getpointer. seekg (0);
seekp() - same as putpointer. seekp (100);

tellg() , tellp() .

seekg (n, ios :: end); - place the getpointer at the nth byte from the end.

⊛ Random Access Files :

In these files, records are placed in arbitrary order and they are not of fixed length.
read() and write() functions of istream and ostream classes.

read() - inputs a record from the specified memory location into a specified location on the devices.

ofstream outfile (<<) number ⇒ writing the number into the file by using outfile object of the ofstream class.
(<< to be appropriately overloaded)

write() - outfile.write (reinterpret_cast <const char*> & number, sizeof (number))

Ex.) Create a random access file capable of storing at most 100 fixed length records for a company that have at most 100 employees. Each record should have an account number, first name, last name and a balance. The account number is considered as a key-field.

```
#include <iostream>
#include <fstream>
#include <string>
#include <stdlib>
#include <iomanip>

class CustomerData
{
    private:
        int acct_no;
        char fname[15];
        char lname[10];
        double balance;
    public:
        CustomerData (int = 0, string = "", string = "", double = 0.0);
        void setFname (string);
        string getFname () const;
```

```
    void setLname(string);
    string getLname() const;
    void setBalance(double);
    double getBalance() const;
    void setAcct(int);
    int getAcct() const;
};
CustomerData :: CustomerData(int acctval, string Fvalue,
                             string Lvalue, double Bvalue)
{
    setAcct(acctval);
    setFname(Fvalue);
    setLname(Lvalue);
    setBalance(Bvalue);
}
void CustomerData :: setAcct(int acctval)
{
    acct_no = acctval;
}
CustomerData :: getAcct() const
{
    return acct_no;
}
void setFname(string fvalue)
{
    const char *fnval = fnamestring.data();
    int length = fnamestring.size();
    length = (length < 15 ? length : 14);
    strcpy(fname, fvalue, length)
    fname(length) = '\0';
}
int main()
{
    ofstream outcredit("credit.dat", ios::binary);
    if(!outcredit)
    {
        cerr << "File cannot be opened" << endl;
        exit(1);
    }
    CustomerData blankCustomer;
    for(int i=0; i<99; i++)
        outcredit.write(reinterpret_cast<const char *>
                        (&blankcustomer), sizeof(CustomerData));
```

```
cin >> acct_no;
while (acct_no > 0 && acct_no <= 100)
{
    cin >> setw(15) >> fname;
    cin >> setw(10) >> lname;
    cin >> balance;
    Customer.setAcct (acct_no);
    Customer.setFname (fname);
    Customer.setLname (lname);
    Customer.setBalance (balance);
    outcredit.seekp (Customer.getAcctNo() - 1) *
                                    sizeof (CustomerData));
    outcredit.write (reinterpret_cast <const char *>
                    (&Customer), sizeof (CustomerData));
    cin >> acct_no;
}
return 0;
}
```

Linear Algebra

Def: Self adjoint operator :

A linear operator T on an I.P.S. V is said to be self-adjoint if $\langle Tx, y\rangle = \langle x, Ty\rangle \ \forall\ x, y \in V$.

We denote the self-adjoint operator of T by T^*.

(i) A self-adjoint operator on a real I.P.S. is called symmetric operator.

(ii) A self-adjoint operator on a complex I.P.S. is called Hermitian operator.

Example: Clearly $T = \hat{0}$ (Zero Operator), $T = I$ (identity operator) are self-adj operators.

Def: If a linear operator T on an I.P.S. V is such that $T^* = -T$ then, T is called skew-symmetric or skew-Hermitian operator accordingly as V is real or complex IPS.

Theorem: Every linear operator on a finite dimensional complex I.P.S. V can be uniquely expressed as $T = T_1 + iT_2$ where, T_1 and T_2 are self-adj. operators on V.

Proof - Write $T = \left(\frac{T+T^*}{2}\right) + i\left(\frac{T-T^*}{2i}\right)$

$$:= T_1 + T_2$$

where, $T_1 = \frac{T+T^*}{2}$; $T_2 = \frac{T-T^*}{2i}$

$$T_1^* = \left(\frac{T+T^*}{2}\right)^* = \frac{1}{2}\left(T^* + (T^*)^*\right) = \frac{1}{2}\left(T^*+T\right) \quad \left[\because (T^*)^* = T\right]$$
$$= T_1$$

$$T_2^* = \left(\frac{T-T^*}{2i}\right)^* = \frac{1}{2i}\left(T^* - (T^*)^*\right) = -\frac{1}{2i}\left(T^* - T\right) \quad \left[\because (T^*)^* = T\right]$$
$$= \frac{1}{2i}(T - T^*) = T_2$$

Hence, T_1 and T_2 are two self-adj. operators on V, such that $T = T_1 + iT_2$ ——— (1)

Suppose $T = U_1 + iU_2$ where $U_1^* = U_1$ and $U_2^* = U_2$ ——— (*)

Claim - $U_1 = T_1$ and, $U_2 = T_2$.

$T^* = (U_1 + iU_2)^* = U_1^* - iU_2^* = U_1 - iU_2$ $(\because (*))$ ——— (**)

From (*) and (**) we have,

$T + T^* = 2U_1$ and, $T - T^* = 2iU_2$

$\Rightarrow U_1 = \frac{T+T^*}{2} := T_1$ and, $U_2 = \frac{T-T^*}{2i} := T_2$

Hence, the representation in (1) is UNIQUE.

Theorem: Every linear operator T on a finite dimensional complex IPS, V can be expressed UNIQUELY as sum of self-adj. operator and skew-Hermitian operator on V.

Proof - Let $T = \frac{1}{2}(T+T^*) + \frac{1}{2}(T-T^*)$

$= T_1 + T_2$

where, $T_1 = \frac{1}{2}(T+T^*)$; $T_2 = \frac{1}{2}(T-T^*)$

Then clearly, $T_1^* = T_1$ and, $T_2^* = -T_2$

$\therefore T_1$ is self-adj. operator and T_2 is a skew-Hermitian operator.

Suppose $T = U_1 + U_2$, where $U_1^* = U_1$ and, $U_2^* = -U_2$.

Then, $T^* = U_1^* + U_2^* = U_1 - U_2$

$U_1 = \frac{T+T^*}{2} = T_1$ and, $U_2 = \frac{T-T^*}{2} = T_2$.

Therefore, the representation of T is UNIQUE.

Example: Let T be a linear operator on an I.P.S. V. Then $T = \hat{0} \iff \langle Tx, y\rangle = 0 \ \forall\ x, y \in V$.

Proof -

$\Longrightarrow$ Clear

$\Longleftarrow$ Let $\langle Tx, y\rangle = 0 \ \forall\ x, y \in V$.

$\Rightarrow \langle Tx, Tx\rangle = 0$ (by letting $y = Tx$)

$\Rightarrow \|Tx\|^2 = 0$

$\Rightarrow Tx = 0 \quad \Rightarrow T = \hat{0}$

Example: Let T be a linear operator on a finite dim. complex I.P.S. V. Then $T = \hat{0} \iff \langle Tx, x\rangle = 0 \ \forall\ x \in V$.

Proof -

$\Longrightarrow$ Clear

$\Longleftarrow$ Let $\langle Tx, x\rangle = 0 \ \forall\ x \in V$

Let $x, y \in V$. Then, $x + y \in V$.

$\Rightarrow \langle T(x+y), (x+y)\rangle = 0$

$\Rightarrow \langle Tx + Ty, x+y\rangle = 0$

$\Rightarrow \cancel{\langle Tx, x\rangle}^{\,0} + \langle Tx, y\rangle + \langle Ty, x\rangle + \cancel{\langle Ty, y\rangle}^{\,0} = 0$

$\Rightarrow \langle Tx, y\rangle + \langle Ty, x\rangle = 0 \quad \forall\ x, y \in V$ ——— (1)

$\Rightarrow \langle Tx, iy\rangle + \langle T(iy), x\rangle = 0 \quad (y \in V,\ iy \in V)$

$\Rightarrow -i\langle Tx, y\rangle + i\langle Ty, x\rangle = 0$

$\Rightarrow i(\langle Ty, x\rangle - \langle Tx, y\rangle) = 0$

$\Rightarrow \langle Ty, x\rangle - \langle Tx, y\rangle = 0$ ——— (2)

From (1) and (2), $\langle Ty, x\rangle = 0 \quad \forall\ x, y \in V$

$\Rightarrow \langle Ty, Ty\rangle = 0$

$\Rightarrow \|Ty\| = 0 \quad \Rightarrow Ty = 0 \quad \forall\ y \in V$

$\Rightarrow T = \hat{0}$

Example: Let $V = \mathbb{R}^2(\mathbb{R})$ with standard i.p.s. on it.

Define $T : \mathbb{R}^2 \longrightarrow \mathbb{R}^2$ by $T(x,y) = (-y, x)$

$$\langle T(x,y), (x,y) \rangle = \langle (-y,x), (x,y) \rangle$$
$$= -yx + xy .$$
$$\therefore \langle T\alpha, \alpha \rangle = 0 \quad \forall \ \alpha = (x,y) \in \mathbb{R}^2.$$

But $T \neq \hat{0}$

$$\begin{bmatrix} -\lambda & -1 \\ 1 & -\lambda \end{bmatrix}$$
$$\lambda^2 + 1 = 0$$

Def. Orthogonal Complement -

Let W be a subspace of vector space V. Then the orthogonal complement of W, denoted by $W^\perp$ is defined by - $W^\perp = \{ v \in V, \langle v, w \rangle = 0 \ \forall \ w \in W \}$

It is evident that $W^\perp$ is a subspace of V.

Example: Let V be a F.D. I.P.S. and let T be a linear operator on V. Suppose W is a subspace of V which is invariant under T. Then the orthogonal complement of W is invariant under T^*.

Sol.) Let $\beta \in W^\perp$

Claim: $T^*\beta \in W^\perp$

i.e., to prove $T^*\beta$ is orthogonal to every vector is W.

Let $\alpha \in W$.

Then, $\langle \alpha, T^*\beta \rangle = \langle T\alpha, \beta \rangle$ ($\because$ by the def. of adj. operator)

$= 0$ ($\alpha \in W \Rightarrow T\alpha \in W$; β is orthogonal to every element in W.)

$\Rightarrow \langle \alpha, T^*\beta \rangle = 0$

$\Rightarrow \overline{\langle \alpha, T^*\beta \rangle} = \overline{0}$

$\Rightarrow \langle T^*\beta, \alpha \rangle = 0$

$\therefore T^*\beta \in W^\perp$

Exercise: Let T and S be two self-adj. operators on a F.D.I.P.S. V. Then ST is self-adj. oper. $\Leftrightarrow ST = TS$.

Exercise (2): Let V be a finite dimensional I.P.S. and T be a linear operator $\ni T^2 = T$. Then T is self-adj. $\Leftrightarrow TT^* = T^*T$.

Take $M = \text{Max}\{M', M_1, \ldots, M_{N_0-1}\}$

Then, $|f_n(x)| < M$, $\forall n$ and $\forall x$

---

(4) $f_n \to f$ uniformly on S, $M > 0$ s.t. $|f_n(x)| \leq M \ \forall x \in S, \forall n$.
Let g be cont. on $B(0;M)$ [a disk] and define $h_n(x) = g[f_n(x)]$, $h(x) = g[f(x)]$, if $x \in S$. $h_n \to h$ uniformly on S.

Sol) Given that g is cont. on $\underbrace{B(0;M)}$. Therefore, g is U.C.
(closed and bounded)

Given $\epsilon > 0$, $\exists\ \delta > 0$, s.t. whenever $|x-y| < \delta$ then, $|g(x)-g(y)| < \epsilon$. ——— (1)

For this δ, $\exists\ N \in \mathbb{Z}^+$, s.t.

$$|f_n(x) - f(x)| < \delta \quad , \forall\ n \geq N \quad \forall\ x \in S. \quad \text{——— (2)}$$

From (1) and (2), given $\epsilon > 0$, $\exists\ N$ s.t. for $n \geq N$,

$$|g(f_n(x)) - g(f(x))| < \epsilon \quad \forall\ x \in S \quad \Rightarrow h_n(x) \rightrightarrows h(x) \text{ unif.}$$

---

(5) $f_n(x) = (1/n)\, e^{-n^2x^2}$; $x \in \mathbb{R}$, $n = 1, 2, \ldots$

Prove: $f_n \to 0$ unif. on $\mathbb{R}$ and $f_n' \to 0$ pointwise on $\mathbb{R}$, but that the convergence of $\{f_n'\}$ is not uniform on any interval containing the origin

Proof - $\lim_{n\to\infty} f_n(x) = \lim_{n\to\infty} \frac{1}{n e^{n^2x^2}} = 0$.

$$\lim_{n\to\infty} \text{Sup} |f_n(x) - f(x)| = \lim_{n\to\infty} \text{Sup} \left| \frac{1}{n e^{n^2x^2}} \right| = \lim_{n\to\infty} \frac{1}{n} = 0.$$

$$f_n'(x) = \left(\frac{1}{n}\right)(-n^2)(2x)\, e^{-n^2x^2} = (-2nx)\, e^{-n^2x^2}$$

$$\lim_{n\to\infty} f_n'(x) = \lim_{n\to\infty} \frac{-2nx}{e^{n^2x^2}} = \lim_{n\to\infty} \frac{-2x}{(2nx^2)e^{n^2x^2}} = 0.$$

$$\lim_{n\to\infty} \text{Sup} |f_n'(x) - f'(x)| = \lim_{n\to\infty} \text{Sup} \left| (-2nx)\, e^{-n^2x^2} \right|$$

For non-uniform convergence, $n_k = k$; $x_k = 1/k$.

$$|f_{n_k}'(x_k) - f'(x_k)| = \left| f_k'\left(\frac{1}{k}\right) - 0 \right| = \left| \left(-2k \cdot \frac{1}{k}\right) e^{-k^2 \cdot \frac{1}{k^2}} \right| = +\frac{2}{e}$$

Assignment 4 -

① $f_n(x) = \left\{ \frac{x}{nx+1} \right\}$ converges pointwise for $x \in [0, \infty)$ and uniformly for $x \in [0, k]$ where, k is a positive number. : Prove

Proof - For pointwise convergence,

$$\lim_{n\to\infty} f_n(x) = \lim_{n\to\infty} \frac{x}{nx+1} = \lim_{n\to\infty} \frac{1}{n + 1/x} < \lim_{n\to\infty} \frac{1}{n} < \epsilon$$

$$\lim_{n\to\infty} f_n(x) = 0 .$$

For uniform convergence,

$$\lim_{n\to\infty} \text{Sup} \left| f_n(x) - f(x) \right| = \lim_{n\to\infty} \text{Sup} \left| \frac{x}{nx+1} - 0 \right| = \lim_{n\to\infty} \frac{k}{nk+1} = 0 .$$

Hence, uniformly convergent.

---

② $f_n(x) = \left\{ \frac{nx}{n^2x^2+1} \right\}$; $(-\infty, \infty)$

Sol) For pointwise convergence,

$$\lim_{n\to\infty} \frac{nx}{n^2x^2+1} = \lim_{n\to\infty} \frac{1}{nx + \frac{1}{nx}} = 0 .$$

For uniform convergence,

$$\lim_{n\to\infty} \text{Sup} \left| f_n(x) - f(x) \right| = \lim_{n\to\infty} \text{Sup} \left| \frac{nx}{n^2x^2+1} - 0 \right|$$

For non-uniform convergence, $n_k = k$, $x_k = 1/k$.

$$\left| f_k\left(\frac{1}{k}\right) - 0 \right| = \left| \frac{k \cdot k}{k^2+k^2} - 0 \right| = \left| \frac{1}{2} \right| > \epsilon_0 . \quad \text{(Non-uniform)}$$

---

③ $f_n \to f$ uniformly on S , f_n is bounded on S.

Prove : $\{f_n\}$ is uniformly bounded on S .

Proof : $\left| f_n(x) - f(x) \right| < \epsilon \quad \forall \quad n \geq N(\epsilon)$.

Given $\epsilon = 1$, $\exists$, N s.t. $n \geq N$

$$\left| f_n(x) - f(x) \right| < 1 \Rightarrow \left| f(x) \right| < 1 + \left| f_{N_0}(x) \right| , \quad n \geq N_0$$

Let $\left| f_{N_0}(x) \right| < M_{N_0}$.

For $n = 1, 2, 3, \ldots, N_0 - 1$, $\left| f_1(x) \right| \leq M_1$, $\ldots$ $\left| f_{N_0-1}(x) \right| \leq M_{N_0-1}$

$$\left| f_n(x) \right| < 1 + \left| f(x) \right| \leq 2 + \left| f_{N_0}(x) \right| ; \quad n \geq N_0 .$$

$$= M'$$

$$\int_a^b (f_n - \epsilon_n)\,d\alpha \le \underline{\int_a^b} f\,d\alpha \le \overline{\int_a^b} f\,d\alpha \le \int_a^b (f_n + \epsilon_n)\,d\alpha \quad \text{——— (**)}$$

$$\Rightarrow 0 \le \overline{\int} f - \underline{\int} f \le 2\epsilon_n (\alpha(b) - \alpha(a))$$

But as $n \to \infty$, $\epsilon_n \to 0$. So, $\overline{\int_a^b} f\,d\alpha = \underline{\int_a^b} f\,d\alpha \Rightarrow f \in R(\alpha)$

Note : $\alpha(x) = x$

$$2\epsilon_n \int_a^b d\alpha = 2\epsilon_n \underbrace{[\alpha(b) - \alpha(a)]}_{(b-a)}$$

SERIES OF FUNCTIONS –

$$\sum_{n=1}^{\infty} f_n(x)$$

$S_n = \sum_{i=1}^{n} f_i(x)$ (nth partial sum).

(S_n) is a sequence of function.

$S_n(x) \longrightarrow S(x)$ point-wise convergence

$S_n(x) \longrightarrow S(x)$ uniformly convergent

$$\int \sum_{i=1}^{\infty} f_i(x) = \sum_{i=1}^{\infty} \int f_i(x)$$

Thm: Suppose $|f_n(x)| \le M_n \;\forall\; x \in E \subseteq \mathbb{R}$, $n \ge 1$. Then the series $\sum_{n=1}^{\infty} f_n(x)$ converges uniformly if $\sum_{n=1}^{\infty} M_n$ converges.

Proof – Use Cauchy Criterion for U.C.

$$\left| \sum_{i=m}^{n} f_i(x) \right| \le \sum_{i=m}^{n} M_i \quad ; \quad S_n = \sum_{i=1}^{n} f_i(x)$$

$$|S_n - S_m|$$

(S_n) is unif. Cauchy, $S_n \to S$ unif. $\left| \sum_{i=n}^{m} M_i \right| < \epsilon$; $n, m \ge N_0$.

Ex) $f_n(x) = 2nxe^{-nx^2}$; $x \in [0,1]$

$\lim_{n\to\infty} f_n(x) = 0 = f(x)$ $\forall\ x \in [0,1]$

$\int_0^1 f(x)\,dx = 0$; $\lim_{n\to\infty} \int_0^1 2nxe^{-nx^2}\,dx = \lim_{n\to\infty} \int_0^1 (e^{-nx^2})'(-n)\,dx$

$= \lim_{n\to\infty} -(e^{-n} - 1) = 1$

Thm) Let $\{f_n\}$ be a seq. of continuous functions on $A \subseteq \mathbb{R}$. and $f_n \rightrightarrows f$ unif. on A. Then f is cont. on A.

Proof: $\forall\ \epsilon > 0$, $\exists\ N(\epsilon) \in \mathbb{N}$ s.t. whenever $n \geq N(\epsilon)$

$|f_n(x) - f(x)| < \epsilon/3$ $\forall\ n \geq N(\epsilon)$.

Take $c \in A$.

$|f(x) - f(c)| \leq |f_n(x) - f(x)| + |f_n(x) - f_n(c)| + |f_n(c) - f(c)|$

$\leq \epsilon/3 + |f_n(x) - f_n(c)| + \epsilon/3$ ——— (a)

Since each f_n is cont. at c,

$|f_n(x) - f_n(c)| < \epsilon/3$ whenever $|x - c| < \delta$ ——— (b)

From (a) and (b), we have, given $\epsilon > 0$, $|x - c| < \delta$.

$|f(x) - f(c)| \leq \epsilon/3 + \epsilon/3 + \epsilon/3 = \epsilon$

Note: $f_n \rightrightarrows f \equiv \sup_{x \in E} |f_n(x) - f(x)| \to 0 \equiv \|f_n - f\|_E \to 0$.

Thm: Let $\{f_n(x)\}$ be a sequence of Riemann integrable functions on $[a,b]$ and $f_n \rightrightarrows f$ unif. on $[a,b]$.

Then, (1) $f \in R(\alpha)$

(2) $\lim_{n\to\infty} \int_a^b f_n\,d\alpha = \int_a^b f\,d\alpha$.

Proof: Let $\epsilon_n = \sup_{x \in [a,b]} |f_n(x) - f(x)|$ ($\because \epsilon_n \to 0$ as $n \to \infty$)

$f_n(x) - \epsilon_n \leq f(x) \leq f_n(x) + \epsilon_n$ ——— (2)

Cauchy Criterion for uniform convergence -

Let $\{f_n\}$ be a sequence of bounded function on $\mathbb{R}$.
Then, $f_n \to f$ uniformly on $A \iff$ for each $\epsilon > 0$, $\exists$ a natural number $H(\epsilon)$ s.t. whenever $m, n \geq H(\epsilon)$ then $\|f_m(x) - f_n(x)\|_A \leq \epsilon$.

Proof: ($\Rightarrow$) $f_n \to f$ unif. on A

$$|f_m(x) - f_n(x)| \leq |f_m(x) - f(x)| + |f_n(x) - f(x)|$$
$$\leq \epsilon/2 + \epsilon/2 = \epsilon \quad ; \quad m, n \geq H(\epsilon)$$

($\Leftarrow$) $\|f_m - f_n\|_A \leq \epsilon \quad , m, n \geq H(\epsilon)$

$$\Rightarrow |f_m(x) - f_n(x)| \leq \|f_m - f_n\|_A \leq \epsilon \quad \forall\ m, n \geq H(\epsilon) \quad \text{---- } (*)$$

$\Rightarrow \{f_n(x)\}$ is a Cauchy sequence is $\mathbb{R}$.

Since, $\mathbb{R}$ is complete, $\{f_n(x)\}$ converges to $f(x)$

from $(*)$, letting $n \to \infty$, we get,

$$|f_m(x) - f(x)| \leq \epsilon \quad \forall \quad m \geq H(\epsilon), \ \forall\ x \in A$$

$\Rightarrow (f_m) \to f$ unif. on A.

Consequences of Uniform Convergence -

① $f_n(x) = x^n \quad ; \quad x \in [0,1]$

↓ pointwise limit.

$$f(x) = \begin{cases} 0 \ , & 0 \leq x < 1 \\ 1 \ , & x = 1 \end{cases}$$ Discontinuous function.

} Non-uniform convergence.

② $$f_n(x) = \begin{cases} n^2 x \ , & 0 \leq x \leq 1/n \\ -n^2(x - 2/n) \ , & 1/n \leq x \leq 2/n \\ 0 \ , & 2/n \leq x \leq 1 \end{cases} \qquad n \geq 2$$

↓ pointwise limit

$f(x) = 0 \ , \quad x \in [0,1] \qquad \int_0^1 f(x)\,dx = 0$

$$\lim_{n\to\infty} \int_0^1 f_n(x)\,dx \neq \int_0^1 \lim_{n\to\infty} f_n(x)\,dx$$
$$\lim_{n\to\infty} 1 \neq 0$$

① $f_n(x) = \frac{x}{n}$; $n_k = k$, $x_k = k$ (Let)

$|f_{n_k}(x_k) - f(x_k)| = |1 - 0| \geq \epsilon_0$ — Non-uniform convergence

② $f_n(x) = x^n$; Let $n_k = k$, $x_k = \frac{1}{2^{1/k}}$; $x \in (-1, 1)$

$|f_{n_k}(x_k) - f(x_k)| = \left|\left(\frac{1}{2^{1/k}}\right)^k - 0\right| = \frac{1}{2} \geq \epsilon_0$ — Non-uniform

③ $f_n(x) = \frac{x^2 + nx}{n}$; Let $n_k = k$, $x_k = -k$. $[k \geq 1]$

$|f_{n_k}(x_k) - f(x_k)| = \frac{k^2 + k(-k)}{k} + k = k \geq \epsilon_0$ — Non-uniform

---

Uniform norm / Sup norm –

Suppose $\phi : A \to \mathbb{R}$ be a bounded function.

Define $\|\phi\|_A = \sup \{ |\phi(x)| : x \in A \}$

Note: $\|\phi\|_A \leq \epsilon \iff |\phi(x)| < \epsilon \quad \forall x \in A$

Lemma: A sequence $\{f_n\}$ converges uniformly to f on A (bounded function)
iff $\|f_n - f\|_A \to 0$ as $n \to \infty$.

Proof – ($\Rightarrow$)

$\exists\ H(\epsilon)$, $n \geq H(\epsilon)$, $|f_n(x) - f(x)| \leq \epsilon$

$\Rightarrow \sup_{x \in A} |f_n(x) - f(x)| \leq \epsilon \quad \forall x \in A$.

↳ $\Rightarrow \|f_n - f\|_A \leq \epsilon$

$\Rightarrow \|f_n - f\|_A \to 0$ as $n \to \infty$.

($\Leftarrow$) $\|f_n - f\|_A \to 0$

$|f_n(x) - f(x)| \leq \sup_{x \in A} |f_n(x) - f(x)| = \|f_n - f\|_A \to 0$

$\Rightarrow f_n$ converges to f uniformly.

---

① $f_n(x) = \frac{x}{n}$; $x \in [0, 1]$ $\quad \sup_{x \in [0,1]} |f_n(x) - f(x)| = \frac{1}{n} \to 0$ as $n \to \infty$. ∴ Uniform

② $f_n(x) = x^n$; $x \in [0, 1]$ $\quad \sup_{x \in [0,1]} |f_n(x) - f(x)| = 1$ ∴ Non-uniform

③ $f_n(x) = \frac{x^2 + nx}{n}$; $x \in [0, 4]$ $\quad \sup_{x \in [0,4]} |f_n(x) - f(x)| = \frac{16}{n} \to 0$ as $n \to \infty$ ∴ Uniform

Sequence and Series of Functions -

$\{f_n(x)\}_{n\geq 1}$ $\quad f_i(x) : \mathbb{R} \to \mathbb{R}$

How about the convergence ?

$f_n(x) : A \to \mathbb{R}$; $A \subseteq \mathbb{R}$, $x \in A$.

Fix $x = x_0 \in A$. Consider $(f_n(x_0))_{n\geq 1}$

Suppose $f_n(x_0) \xrightarrow[n\to\infty]{} f(x_0)$ for each $x_0 \in A$.

Then we call the seq. $(f_n(x))$ converges pointwise.

Consider the function $f(x)$, $x \in A$. This $f(x)$ is called the pointwise limit function of the seq $\{f_n(x)\}$.

Def. Let $(f_n(x))_{n\geq 1}$ be a sequence of functions defined on $A \subseteq \mathbb{R}$ and for each $x \in A$

$f_n(x) \to f(x)$ pointwise iff. given $\epsilon > 0$, $\exists$ a natural number $k(\epsilon, x)$ s.t. whenever $n \geq k(\epsilon, x)$, then,

$$|f_n(x) - f(x)| < \epsilon .$$

Ex) ① $f_n(x) = \frac{x}{n} \longrightarrow f \equiv 0$, $x \in \mathbb{R}$

② $f_n(x) = x^n \longrightarrow f = \begin{cases} 0, & -1 < x < 1 \\ 1, & x = 1 \end{cases}$

③ $f_n(x) = \frac{x^2 + nx}{n} \longrightarrow f(x) = x$, $x \in \mathbb{R}$

④ $f_n(x) = \frac{\sin(nx+n)}{n} \longrightarrow f \equiv 0$, $x \in \mathbb{R}$

Uniform Convergence -

Def. Let $(f_n(x))_{n\geq 1}$ be a sequence of functions defined on $A \subseteq R$.

⊛ $f_n(x) \to f(x)$ uniformly on A iff given $\epsilon > 0$, $\exists$ a natural numb $k(\epsilon)$ s.t. whenever $n \geq k(\epsilon)$, then

$$|f_n(x) - f(x)| < \epsilon \ , \ \forall\, x \in A .$$

Non-Uniform Convergence -

⊛ $f_n(x) \not\to f(x)$ uniformly on A iff for some $\epsilon_0 > 0$, $\exists$ subsequence (f_{n_k}) of (f_n) and a seq. $(x_k) \subseteq A$ s.t. $|f_{n_k}(x_k) - f(x_k)| > \epsilon_0$; $\forall k \in \mathbb{N}$

(9) Prove: $\lim_{n\to\infty} \int_0^1 \frac{nf(x)}{1+n^2x^2}\,dx = \frac{\pi f(0)}{2}$; f is cont. on $[0,1]$.

Proof - $\int_a^b f(x)g(x)\,dx = f(c)\int_a^b g(x)\,dx$; $c \in [a,b]$

$$\int_0^1 \frac{nf(x)}{1+n^2x^2}\,dx = \int_0^{1/\sqrt{n}} \frac{nf(x)}{1+n^2x^2}\,dx + \int_{1/\sqrt{n}}^1 \frac{nf(x)}{1+n^2x^2}\,dx = I_1 + I_2 .$$

$$I_1 = f(a_n) \int_0^{1/\sqrt{n}} \frac{n}{1+n^2x^2}\,dx \quad ; \quad 0 \le a_n \le \frac{1}{\sqrt{n}}$$

$$= f(a_n)\left[\tan^{-1}\sqrt{n}\right] \longrightarrow \frac{\pi}{2} f(0) \text{ as } n \to \infty . \quad [0 \le a_n \le 0]$$

Remains to show $I_2 \to 0$ as $n \to \infty$.

$$I_2 = f(b_n) \int_{1/\sqrt{n}}^1 \frac{n}{1+n^2x^2}\,dx \quad ; \quad \frac{1}{\sqrt{n}} \le b_n \le 1 .$$

$$\le M\left[\tan^{-1}(n) - \tan^{-1}(\sqrt{n})\right] \longrightarrow 0 \text{ as } n \to \infty .$$

$\left[-\cot x\right]_{\pi/2}^{\pi}$

$1-(-1) = 2$

(10) (a) $\int_{-\infty}^{\infty} |\sin t|\,dt$; $\lim_{s\to\infty} \int_{-s}^{s} |\sin t|\,dt$ does not exist.

(b) $\int_{-\infty}^{\infty} \sin t\,.\,dt$; $\lim_{s\to\infty} \int_{-s}^{s} \sin t\,.\,dt = 0$

(c) $\int_{-\infty}^{\infty} e^{-x^2}\,dx$; $\lim_{s\to\infty} \int_{-s}^{s} e^{-x^2}\,.\,dx = \lim_{s\to\infty} \int_0^s t^{-1/2} e^{-t}\,dt$

$= \Gamma(1/2) = \sqrt{\pi}$

$x^2 = t$
$\Rightarrow 2x\,dx = dt$
$\Rightarrow dx = \frac{dt}{2\sqrt{t}}$

(d) $\int_{-\infty}^{\infty} x^{1/3}\,dx$; $\lim_{s\to\infty} \int_{-s}^{s} x^{1/3}\,dx = \lim_{s\to\infty}\left[\frac{3}{4}x^{4/3}\right]_{-s}^{s} = \lim_{s\to\infty} \frac{3}{2} s^{4/3} = 0$

③ $f \geq 0$, f is continuous on $[a,b]$; ~~$f \in R[c,1]$ for every~~

$\int_a^b f(x)\,dx = 0$. Prove: $f(x) = 0 \quad \forall \; x \in [a,b]$

Proof : $\sup L(P,f) = \inf U(P,f) = 0 \quad \forall \; P \in P[a,b]$

Let $P = \{a = x_0, x_1, \ldots, x_n = b\}$ be any partition.

$m_i = \inf_{x \in [x_{i-1}, x_i]} f(x)$; $M_i = \sup_{x \in [x_{i-1}, x_i]} f(x)$; $m_i \geq 0$, $M_i \geq 0$

$\therefore L(P,f) \geq 0$; $U(P,f) \geq 0$

$\sup L(P,f) \not\geq L(P,f)$; $\inf U(P,f) \not\leq U(P,f) \geq 0$

~~$\inf U(P,f) \geq 0$~~

if $f(x) = 0$ then, $\int_a^b f(x)\,dx = 0$.

if $f(x) > 0$ then, $\exists$ at least one $x_0 \in [a,b]$ s.t. $f(x_0) \not> 0$

Since, f is cont. at $x = x_0$ $\exists \; \delta > 0$ s.t. $f(x) \not> 0$, $\forall \; x \in (x_0 - \delta, x_0 + \delta)$

$f(x) \not> 0$, $\forall \; x \in [x_0 - c, x_0 + c] \subseteq (x_0 - \delta, x_0 + \delta)$

Let $m = \inf_{x \in I} f(x) > 0 \quad [\because f(x) > 0]$

$\int_a^b f(x)\,dx \geq \int_{x_0 - c}^{x_0 + c} f(x)\,dx \not\geq m(2c) > 0$.

$\therefore$ $f(x)$ must be zero.

④ $f : (0,1] \to \mathbb{R}$; $f \in R[c,1]$ for every $c > 0$.

$\int_0^1 f(x)\,dx = \lim_{c \to 0} \int_c^1 f(x)\,dx$

Prove : $f \in R[0,1]$

⑤ MVT for integrals.

$\lim_{c \to 0} \sup L(P,f)$; $\sup \lim_{c \to 0} L(P,f)$; $\lim_{c \to 0} \sum m_i \delta_i$; $\lim_{c \to 0}$

⑥ Prove : $\int_0^1 \frac{dx}{x^2 + x^{1/2}}$ is convergent.

Proof : $\frac{1}{x^{3/2}} > \frac{1}{x^2 + x^{1/2}}$; $\int_0^1 \frac{dx}{x^{3/2}}$ is cgt. $\therefore \int_0^1 \frac{dx}{x^2 + x^{1/2}}$ is cgt.

⑦ Prove : $\int_0^1 \frac{\sin x}{x^2}$ is divergent.

Proof : $\forall x \in [0,1]$, $\sin x > 0$, $\sin x < 1$.

$f(x) = \frac{\sin x}{x^2}$; $g(x) = \frac{1}{x}$; $\lim_{x \to 0} \frac{f(x)}{g(x)} = \lim_{x \to 0} \frac{\sin x}{x} = 1$

$\therefore f(x)$ and $g(x)$ converge and diverge together.

$\because \int_0^1 \frac{1}{x}\,dx$ diverges $\therefore \int_0^1 \frac{\sin x}{x^2}$ diverges.

$\lim_{s\to\infty} \int_{-s}^{+a} f(x)\,dx$ exists and finite.

$\lim_{s\to\infty} \int_{a}^{s} f(x)\,dx$ exists and finite.

Adding the above two limits,

$$\lim_{s\to\infty} \int_{-s}^{s} f(x)\,dx = A.$$

$$\therefore \text{C.P.V. of } \int_{-\infty}^{\infty} f(x)\,dx = A.$$

$$\text{C.P.V.} \int_a^b f(x)\,dx = \lim_{\epsilon\to 0^+} \left[\int_a^{c-\epsilon} f(x)\,dx + \int_{c+\epsilon}^{b} f(x)\,dx \right]$$

→ Suppose f is unbounded at $x = c$; $c \in (a, b)$

Assignment 3 -

① $\int_a^b f(x)\,dx = \inf_{P\in\mathcal{P}} U(P,f) = \sup_{P\in\mathcal{P}} L(P,f)$

You can always choose a partition s.t.

$$\Delta\alpha_i = \frac{\alpha(b) - \alpha(a)}{n} \quad ; \quad i = 1, 2, \ldots, n.$$

② Prove: $\lim_{n\to\infty} \frac{(n!)^{1/n}}{n} = e^{-1}$; $\int_0^1 \log x\,dx$; $P = \{1/n, 2/n, \ldots, 1\}$

$f(x) = \log x.$

Proof: $L(P,f) = \sum_{i=1}^{n} m_i \delta_i = \frac{1}{n} \sum_{i=1}^{n} \log(x_i)$

$U(P,f) = \sum_{i=1}^{n} M_i \delta_i = \frac{1}{n} \sum_{i=1}^{n} \log(x_i)$

$$\frac{(n!)^{1/n}}{n} = \frac{\left(n(n-1)\ldots 2\cdot 1\right)^{1/n}}{n^n} = \left(1\left(1-\frac{1}{n}\right)\left(1-\frac{2}{n}\right)\cdots\frac{2}{n}\cdot\frac{1}{n}\right)^{1/n}$$

$$= \exp\left(\frac{1}{n}\sum_{k=0}^{n-1} \log\left(1 - k/n\right)\right)$$

$$\int_0^1 \log(1-x)\,dx = \lim_{\epsilon\to 0^+} \int_0^{1-\epsilon} \log(1-x)\,dx = -1. \qquad f(x) = \log(1-x)$$

$\therefore$ $L(P,f)$ must converge to -1.

$$L(P,f) = \frac{1}{n}\sum_{k=0}^{n-1} \log\left(1 - \frac{k}{n}\right) = -1$$

$$\therefore \lim_{n\to\infty} \frac{(n!)^{1/n}}{n} = e^{-1}$$

Cauchy Principal Value (C.P.V.) -

$$\int_{-\infty}^{\infty} f(x)\,dx = \lim_{\underbrace{M,N \to \infty}_{\text{independent}}} \int_{-M}^{N} f(x)\,dx$$

If $\lim_{R\to\infty} \int_{-R}^{R} f(x)\,dx$ exists, then its value is called C.P.V.

C.P.V. $\int_{-\infty}^{\infty} x^3\,dx = \lim_{s\to\infty} \int_{-s}^{s} x^3\,dx = 0$.

If the integrand is an even function then C.P.V. exists
$\Leftrightarrow$ the integral converges.

Theorem: Suppose f is continuous on $(-\infty,\infty)$ and if $\int_{-\infty}^{\infty} f(x)\,dx$ converges to A, then C.P.V. $\int_{-\infty}^{\infty} f(x)\,dx = A$.

Proof - We need to prove that $\lim_{s\to\infty} \int_{-s}^{s} f(x)\,dx = A$.

f is continuous on $(-\infty,-a]$, $[a,\infty)$, $a \in (-\infty,\infty)$

f is Riemann integrable on $(-\infty,-a]$ and $[a,\infty)$.

Ex) (i) $\int_0^1 \frac{dx}{x}$; (ii) $\int_0^1 \frac{dx}{\sqrt{1-x^2}}$

(i) $F(\epsilon) = \int_\epsilon^1 \frac{dx}{x} = |\ln x|_\epsilon^1 = -\ln \epsilon$; $\lim_{\epsilon \to 0^+} F(\epsilon) = \infty$.

Divergent.

(ii) $F(\epsilon) = \int_0^{1-\epsilon} \frac{dx}{\sqrt{1-x^2}} = \left[\sin^{-1} x\right]_0^{1-\epsilon} = \sin^{-1}(1-\epsilon) - \sin^{-1} 0 = \sin^{-1}(1-\epsilon)$

$\lim_{\epsilon \to 0^+} F(\epsilon) = \sin^{-1}(1) = \pi/2$ Convergent.

Thm) The improper integral $\int_a^b \frac{dx}{(x-a)^p}$ converges for $p<1$ and diverges for $p \geq 1$.

Proof - $\int_{a+\epsilon}^b \frac{dx}{(x-a)^p} = \frac{1}{1-p}$

Proof - (i) $\mu > 1$: $|x^{\mu} f(x)| < M$

So, $|f(x)| < \left|\frac{M}{x^{\mu}}\right|$ ——— (a)

Since, $M > 1$, $\int_a^{\infty} \frac{1}{x^{\mu}} dx$ converges.

By Comparison Test, from (a), we get the integral $\int_a^{\infty} f(x)dx$ converges.

(ii) $\mu \leq 1$: $|x^{\mu} f(x)| > M$; $M > 0$

$|f(x)| > \frac{M}{x^{\mu}}$ if $\mu \leq 1$. $\int_a^{\infty} \frac{1}{x^{\mu}} dx$ diverges.

By Comparison Test, $\int_a^{\infty} f(x)\,dx$ diverges.

~~(ii) If $\mu \leq 1$, s.t. $x^{\mu} f(x)$ f~~

Test for Convergence :

(1) $\int_{-\infty}^{\infty} x^{1/3} dx$; (2) $\int_{-\infty}^{\infty} \frac{dx}{1+x^2}$

(1) $\int_{-\infty}^{\infty} x^{1/3} dx = \int_{-\infty}^{0} x^{1/3} dx + \int_{0}^{\infty} x^{1/3} dx$

$F_1(s) = \int_{-s}^{0} x^{1/3} dx = \left[\frac{x^{4/3}}{4/3}\right]_{-s}^{0} = -\frac{3}{4}(-s)^{4/3}$

$F_2(s) = \int_{0}^{s} x^{1/3} dx = \left[\frac{x^{4/3}}{4/3}\right]_{0}^{s} = \frac{3}{4}(s)^{4/3}$

Divergent

(2) Convergent

Def. Let f be R-integrable in $[a+\epsilon, b]$ for all $\epsilon > 0$ such that $0 < \epsilon < b-a$, but is not R-integrable on $[a,b]$. Let $F(\epsilon) = \int_{a+\epsilon}^{b} f(x)dx$. Then, $\int_a^b f(x)dx$ is said to be convergent to A if $\lim_{\epsilon \to 0^+} F(\epsilon) = A$ otherwise it is divergent.

Def. f is R-integrable in $[a, b-\epsilon]$ $\forall$ $\epsilon > 0$ but not R-integrable on $[a,b]$. $F(\epsilon) = \int_a^{b-\epsilon} f(x)dx$. If $\lim_{\epsilon \to 0^+} F(\epsilon) = A$, then say the integral $\int_a^b f(x)dx$ converges otherwise it diverges

Theorem 2) If $\int_a^\infty f(x)\,dx$ and $\int_a^\infty g(x)\,dx$ are convergent, then $\int_a^\infty (f(x) \pm g(x))\,dx$ are convergent.

Theorem 3) (Comparison Test) If $0 \le f(x) \le g(x)$ $\forall\ x \in [a, \infty)$ and if $f(x)$ and $g(x)$ are Riemann integrable, then

(i) if $\int_a^\infty g(x)\,dx$ converges then $\int_a^\infty f(x)\,dx$ converges.

(ii) if $\int_a^\infty f(x)\,dx$ diverges then $\int_a^\infty g(x)\,dx$ diverges.

Prob.) (1) Show that $\int_a^\infty \frac{1}{e^x+1}\,dx$ converges.

(2) Show that $\int_2^\infty \frac{1}{\sqrt{x^2-1}}\,dx$ diverges.

Sol.) (1) $\frac{1}{e^x} > \frac{1}{e^x+1}$; $\int_a^\infty e^{-x}\,dx$ converges $\therefore \int_a^\infty \frac{1}{e^x+1}$ converges.

(2) $\frac{1}{x} < \frac{1}{\sqrt{x^2-1}}$; $\int_2^\infty \frac{1}{x}$ diverges $\therefore \int_2^\infty \frac{1}{\sqrt{x^2-1}}$ diverges.

---

(3) $\int_1^\infty \frac{\sin^2 x}{x^2}\,dx$ (4) $\int_2^\infty \frac{dx}{\log x}$

Sol) (3) $\frac{\sin^2 x}{x^2} < \frac{1}{x^2}$; $\int_1^\infty \frac{1}{x^2}\,dx$ converges $\therefore \int_1^\infty \frac{\sin^2 x}{x^2}\,dx$ converges.

(4) $\frac{1}{\log x} > \frac{1}{x}$; $\int_2^\infty \frac{1}{x}\,dx$ diverges $\therefore \int_2^\infty \frac{1}{\log x}\,dx$ diverges.

Theorem 4) (μ-Test for Convergence).

Let $f(x)$ be Riemann integrable in $[a, s]$, where $a > 0$ and (fixed) let $s > a$. Then-

(i) If there exist $\mu > 1$ s.t. $x^\mu f(x)$ is bounded for $x > a$, then $\int_a^\infty f(x)\,dx$ converges.

(ii) If $\exists\ \mu \le 1$, s.t. $x^\mu f(x)$ has a positive (negative) lower bound for $x > a$ then, $\int_a^\infty f(x)\,dx$ diverges to ∞ $(-\infty)$.

Improper Integrals -

Def. 1) If $b=\infty$ or, $a=-\infty$, but f is bounded on each point of the interval where it is defined, then $\int_a^b f(x)\,dx$ is called an improper integral of 1st kind.

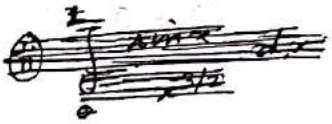
Ex. (i) $\int_1^\infty \frac{1}{x^2}\,dx$ (ii) $\int_0^\infty e^{-x}\,dx$ (iii) $\int_1^\infty \frac{1}{\sqrt{x}}\,dx$ (iv) $\int_1^\infty \sin x\,dx$

Def. 2) If a and b are real numbers and f is unbounded at some point in $[a,b]$ then $\int_a^b f(x)$ is called an improper integral of 2nd kind.

Ex. (i) $\int_0^2 \frac{x}{1-x}\,dx$ (ii) $\int_0^1 \frac{\sin x}{x^{3/2}}\,dx$ (iii) $\int_{-2}^0 \frac{x^{1/2}}{1+x}\,dx$

Def. 3) If the integrand is unbounded at some point of the unbounded interval where it is defined, then $\int_a^b f(x)\,dx$ is called an improper integral of 3rd kind.

Ex. (i) $\int_0^\infty \frac{1}{x^2+x^{1/2}}\,dx$



Convergence of the integral of 1st kind -

Let f be Riemann integrable for every $s > a$ and let $F(s) = \int_a^s f(x)\,dx$. Then, $\int_a^\infty f(x)\,dx$ is said to be convergent to A if $F(s) \to A$ as $s \to \infty$. If $\int_a^\infty f(x)\,dx$ does not converge, we say the integral is divergent.

Theorem 1 : The improper integral $\int_a^\infty \frac{1}{x^p}\,dx \;;\; x > a > 0$ converges for $p > 1$ and diverges for $p \le 1$.

Proof - $F(s) = \int_a^s \frac{1}{x^p}\,dx = \left[\frac{x^{-p+1}}{-p+1}\right]_a^s = \frac{s^{1-p} - a^{1-p}}{1-p}$ ...

(6) $a \in \mathbb{R}$, f is 2 times diff on (a, ∞) and M_0, M_1, M_2 are l.u.b. of $|f(x)|, |f'(x)|, |f''(x)|$. Prove: $M_1^2 \leq 4 M_0 M_2$

For $h > 0$, $f(x+h) = f(x) + h f'(x) + \frac{h^2}{2!} f''(\xi)$; $x < \xi < x+h$

$$|h f'(x)| = \left| f(x) - f(x+h) + \frac{h^2}{2!} f''(\xi) \right| \leq 2M_0 + \frac{h^2}{2} M_2$$

$$\Rightarrow h^2 M_2 - 2h |f'(x)| + 4M_0 \geq 0$$

$\therefore$ Discriminant $\leq 0 \Rightarrow 4 |f'(x)|^2 - 4.4 M_0 M_2 \leq 0$

$$\Rightarrow |f'(x)|^2 \leq 4 M_0 M_2 .$$

Since it is true for all x, therefore it is true for l.u.b.

$$\Rightarrow M_1^2 \leq 4 M_0 M_2 .$$

---

(7) f is diff. on $[a, b]$ with $f(a) = 0$ and $\exists\ A \in \mathbb{R}$ such that $|f'(x)| \leq A |f(x)|$ on $[a, b]$. Prove that $f(x) = 0$ for all $x \in [a, b]$.

If $A = 0$, $|f'(x)| \leq 0 \Rightarrow |f'(x)| = 0 \Rightarrow f'(x) = 0$. [$f$ is constant]

$\because f(a) = 0 \quad \therefore f(x) = 0$ (trivial)

Taking $A > 0$,

$a < x_0 < b$, $[a, x_0]$., $M_0 = \sup_{x \in [a, x_0]} |f(x)|$

$M_1 = \sup_{x \in [a, x_0]} |f'(x)|$

Use MVT in $[a, x_0]$, $f(x) - f(a) = (x-a) f'(c)$; $a < c < x < x_0$.

---

(8) $f(x) = x$; $\rightarrow x$ is a fixed point.

(a) f is diff. ; $f'(t) \neq 1$ for every real t.

Prove: f has at most one fixed point.

(c) $A < 1$ (const) $\ni |f'(t)| \leq A \quad \forall\ t \in \mathbb{R}$

③ $f: [0,\infty] \to \mathbb{R}$ is cont. and diff in $(0,\infty)$, $f(0) = 0$. f' is monotonically increasing. Prove: $f(x)/x$ is monotonically increasing in $(0,\infty)$

$$\left(\frac{f(x)}{x}\right)' = \frac{1}{x^2}\left(x f'(x) - f(x)\right) = \frac{1}{x} f'(x) - \frac{1}{x^2} f(x) > 0 \quad \downarrow$$

[To prove: $f'(x) > \frac{f(x)}{x}$]

Since f is differentiable,

$$f(x) - f(0) = f'(c)(x-0) \quad ; \quad 0 < c < x$$

$$\Rightarrow f(x) < f'(x)\, x \quad ; \quad [\because f' \text{ is monotonically inc.}]$$

$$\Rightarrow f'(x) > \frac{f(x)}{x}$$

---

④ f is cont. on open interval I containing x_0. f' is defined on I except possibly at x_0 and $f'(x) \to L$ as $x \to x_0$. Prove that $f'(x_0) = L$.

$\lim_{x \to x_0} f'(x) = L$ ——— (*)

Using L'Hospital's Rule,

$$f'(x_0) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

$\because f$ is cont $f(x+h) = f(x)$

$\therefore \left(\frac{0}{0} \text{ form}\right)$

$$\Rightarrow f'(x_0) = \lim_{h \to 0} f'(x+h)$$

and, $\lim_{x \to x_0} f'(x) = L \quad \therefore \quad f'(x_0) = L$

---

⑤ Prove: $\lim_{h \to 0} \frac{f(x+h) + f(x-h) - 2f(x)}{h^2} = f''(x)$

Since, f is diff., it is also cont. and hence it has a $\left[\frac{0}{0}\right]$ form. Using L'Hospital's Rule,

$$\lim_{h \to 0} \frac{f'(x+h) - f'(x-h)}{2h} = \frac{1}{2} \lim_{h \to 0} \frac{f'(x+h) - f'(x) - f'(x-h) + f'(x)}{h}$$

$$= \frac{1}{2}\left[f''(x) + f''(x)\right] = f''(x)$$

$$f(x) = \begin{cases} x^{n+1} \sin\left(\frac{1}{x}\right), & x \neq 0 \\ 0, & x = 0 \end{cases}$$

$f(x)$ is n times differentiable but not $(n+1)$ times.

$\therefore f(x) = x^2 \sin\left(\frac{1}{x}\right) \to f''(x)$ does not exist.

Assignment 1 -

(12) $f(x) = 1/x^2$; $A = [1, \infty)$; $B = (0, \infty)$

$|f(x) - f(y)| < M|x-y| \quad \forall\, x, y \in A \longrightarrow$ Lipschitz continuous

$$\Rightarrow \left|\frac{1}{x^2} - \frac{1}{y^2}\right| = \left|\frac{y^2 - x^2}{x^2 y}\right| = \frac{|(y-x)(x+y)}{x^2 y^2} = \left(\frac{1}{xy^2} + \frac{1}{yx^2}\right)|y-x|$$

$$\leq 2|y-x|$$

$\therefore f(x)$ is U.C. on A.

Let $x_n = \frac{1}{\sqrt{n}}$; $y_n = \frac{1}{\sqrt{n+1}}$

$x_n - y_n \to 0$ as $n \to \infty$ but, $|f(x_n) - f(y_n)| = |n+1-n| = 1$

Therefore, $f(x)$ is not U.C. on B.

Assignment 2 -

(1) $f: \mathbb{R} \to \mathbb{R}$ be differentiable ; $|f(x) - f(y)|^2 < |x-y|^2$ $\forall\, x, y \in \mathbb{R}$. Prove that f is constant.

$$\frac{|f(x) - f(y)|}{|x-y|} < |x-y|$$

Take limit as $y \to x$, $0 \leq |f'(x)| \leq 0$

$\therefore f'(x) = 0 \Rightarrow f$ is constant.

---

(2) $f'(x) > 0 \quad \forall\, x \in (a, b)$. Prove f is strictly increasing in (a, b). $g = f^{-1}$. Prove that g is differentiable and, $g'(f(x)) = \frac{1}{f'(x)}$; $x \in (a$

Using mean value theorem,

$$f(b) - f(a) = (b-a) f'(c) \quad ; \quad a < c < b$$

$$\Rightarrow f(b) > f(a)$$

$$g(f(x)) = x$$

$$\Rightarrow g'(f(x))\, f'(x) = 1 \quad \Rightarrow \quad g'(f(x)) = \frac{1}{f'(x)}.$$
