DATA STRUCTURES AND ALGORITHMS

Data:- Any fact that is stored - existing information (or) knowledge.

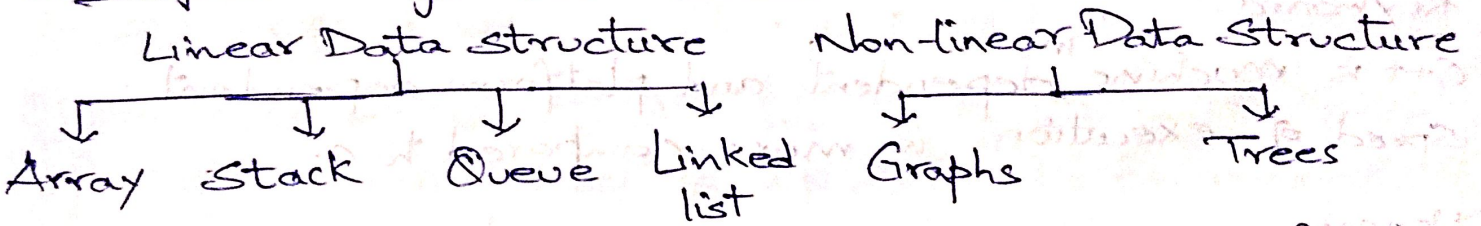
Computer data is the information processed or stored by computer - text documents, images, audio clips, software programs etc.,

Organized or classified data is Information. (Meaningful) -Processed data → -Information.

Data Structure:- A data structure is a scheme for organizing data in the memory of a computer. It is a particular way of storing and organizing data in a computer so that it can be used efficiently.

Different kinds of data structures are suited to diff. kinds of applications.

Classification of Data structures:-



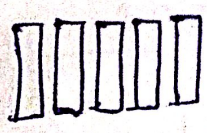
Linear:- Data elements construct a sequence of a linear list. The elements are adjacently arranged to each other and in a specified order. It consumes linear memory space.

The data element are visited sequentially where only a single element can be directly reached.

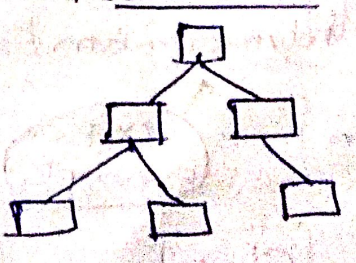
Non-Linear:- Not arranged consecutively, arranged in sorted order. Data elements can be attached to more than one element exhibiting the hierarchial relationship which involves relationship btn child, parent & Grandparent.

~~The tran~~

Linear



Non-Linear



Data structures help you deal with diff. ways of arranging, processing and storing data

C++:-

C	C++
→ Supports function programming	→ Supports ~~opp~~ oops concepts
→ scanf() - input printf() - output	→ cin - input cout - output.
→ can't use inheritance	→ can use inheritance
→ File extension is .c	→ File extension is .cpp

cin :- reads input.

Extraction operator (>>) :- for reading inputs. extracts data from the object cin which is entered using the keyboard.

C++ is machine independent and platform dependent.

Speed of execution is more compared to c.

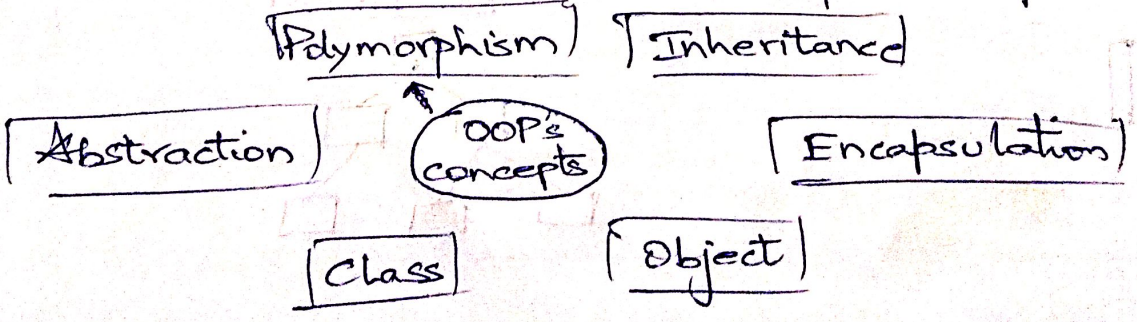
Uses:-

(1) It is used to design Operating system (OS)
OS → interface, communicator b/n user & computer:
Converts the programs into machine understandable language.
OS → set of instructions framed by a programming language - C++.

(2) It is used to design Database, games, graphics...

OOP:- Object Oriented Programming :- ~~aims to im~~

Main aim is to bind together the data and the functions that operate on them so that no other part of the code can access this data except that function.



Class :- The building block of c++ that leads to OOP is a class. It is a user-defined datatype, which holds its own data members & member functions, which can be accessed and used by creating an instance of that class. A class is like a blueprint for an object.

Ex:- Cars :- diff cars in a showroom
common properties :- speed limits, mileage, cost etc,
class has data members and member functions
Data members are the data variables.
Member functions are functions used to manipulate these variables & together these define the properties & behaviour of the object in a class.
Create diff. functions with same variables.
in C, create these variables multiple times
in C++, use class

Object :- It is an identifiable entity, is an instance of class. When class is defined, no memory is allocated but when it is instantiated (Object is created), memory is allocated.

Eg:-
```
class person.
{
  char name[20];
  int id;
 public:
   void getdetails(){}
};
int main()
{
  person p1; → p1 is object of class person.
}
```

Encapsulation :- defined as wrapping up of data and information under a single unit.
Ex:- Company – accounts section, finance section sales section etc.,
Each does its own work.

A situation, finance section may need data about Sales section, in that case, he is not allowed to directly access the data of Sales section. Contact & request for data.

Encapsulation leads to data abstraction or hiding. Data of any of the above sections are hidden from the other.

Abstraction:- Displaying only essential information & hiding the details.

Ex:- Project :- Display only the abstract - limited info.

Polymorphism:- having many forms. Ability of a message to be displayed in more than one form.

A person at sametime have diff. characteristic - an employee, father, son etc.,

Operator overloading :- The process of making an operator to exhibit diff. behaviours in diff instances.

Function overloading :- Using a single function name to perform different types of tasks.

Ex:-
```
int main()
{
sum1 = sum(20, 30);
Sum2 = sum(20,30,40);
}
```

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

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

Inheritance :- Capability of a class to derive properties and characteristics from another class.

Subclass :- (Derived class) :- The class that inherits properties from another class

Super class (Base class) :- The class whose properties are inherited by subclass.

Reusability :-

Algorithm :- finite set of instructions to perform a particular task.

characteristics of an algorithm :-

(1) Input - Zero or more quantities are externally supplied.
(2) Output - Atleast one quantity is produced.
(3) Definiteness - Each instruction is clear & unambiguous.
(4) Finiteness - The algorithm terminates after a finite no. of steps.
(5) Effectiveness - it must be effective.

It should be easy to understand.
It should be clear.

Development of an Algorithm :-

Problem Statement, Model Formulation (structure), Algorithm design (designing), Algorithm correctness, Implementation (manufacture), Algorithm Analysis, Program Testing, Documentation (checking)

Efficiency of Algorithms :-

The performance of algorithms can be measured on the scales of time and space.

Performing a task in the min. possible time - Time complexity.

An algorithm that consumes or needs limited memory space for its execution - space complexity.

Algorithm Analysis :-

Efficiency of an algorithm can be analyzed at two diff. stages, before & after implementation.

A Priori Analysis :- Theoretical analysis of an algorithm, measured by space & time complexity, processor speed are constant & have no effect on the implementation.

A Posterior Analysis :- Empirical analysis of an algorithm. The algorithm is implemented using programming language & executed on computer machine. In this analysis, actual statistics like running time & space required are collected.

A Priori	A Posteriori
without experience	requires experience
Algorithm	Program
Language Independent	Language Dependent
Hardware Independent	Hardware Dependent
Time & Space functions	Watch time & bytes

Analysis of algorithm :- Process of analyzing the problem-solving capability of the algorithm in terms of time & size required. Time or performance

Worst case :- max. no. of steps taken on any instance of size a.

Best case :- min. no.

Avg. case :- avg. no.

Time analysis:-

Frequency count (or) step count:-

It specifies how many times a statement is executed.

(1) For comments, declarations S.C=0.

(2) Return, Assignments – S.C=1
(we can return value) (assign)

(3) Ignore lower order exponents when higher order exponents are present.

Ex:- $3n^4 + 4n^3 + 10n^2 + n + 100$ → (Ignore) $3n^4 = O(n^4)$

(4) Ignore constant multipliers.

Ex:-
```
int sum(int a[], int n)
{
    s=0 ------------------> Assignment -> S.C=1
    for(i=0; i<n; i++)
        s=a.s + a[i]
    return s; ------------> Return -> S.C=1
}
```

↳ Let n=3, i=0

0<3 → 1
1<3 → 2
2<3 → 3
3<3 → false

(condition checked) including failure case

4 times (n+1) times

The inner statement executed – 3 times (n times)

Body will be executed ⇒ 1, n+1, n, 1 → Higher order exponential.

⇒ 2n+3
= 2n = O(n)

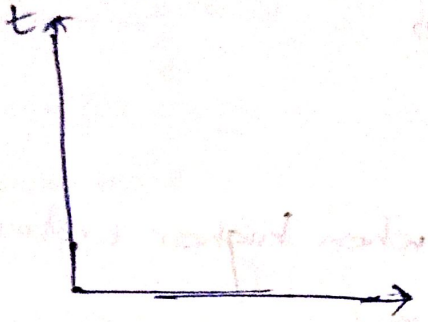
(ignore constant multipliers)

Asymptotic Notations:-

(1) Big oh Notation (2)

(1) Big oh:- Mainly represents upperbound of alg's runtime
→ To calculate max. amount of time taken by the alg.
→ Worst case
→ Max. time → High time complexity.

Def:- Let f(n), g(n) be two non-negative functions, then f(n) = O(g(n)) if there exists two +ve constants c, n_0 such that $f(n) \leq c * g(n)$, $\forall n > n_0$.



f(n) = 3n+2, g(n) = n:

$f(n) \leq c * g(n)$

$3n+2 \leq c*n$ Let n=1, c=4

$5 \leq 4$ ←

n=2	$8 \leq 8$
n=3	$11 \leq 12$
n=4	$14 \leq 16$

Big omega (Ω) :- to represent lowerbound of algorithm.

Polynomial notation :-
Polynomial algorithms include quadratic algorithms $O(n^2)$, cubic algorithms $O(n^3)$.
represents an algorithm whose performance is directly proportional to square of size of the data set.
Exponential notation :- $O(2^n)$ describes an algorithm whose growth doubles with each addition to the data.
Exp. ↑ces more (doubles) than poly.

Avg., Best, worst case complexities :-
Time complexity is dependept on parameter ~~with~~ associated with the I/O instances of the problem.
Eg:- According to no. of instances, time complexities ↑ces.