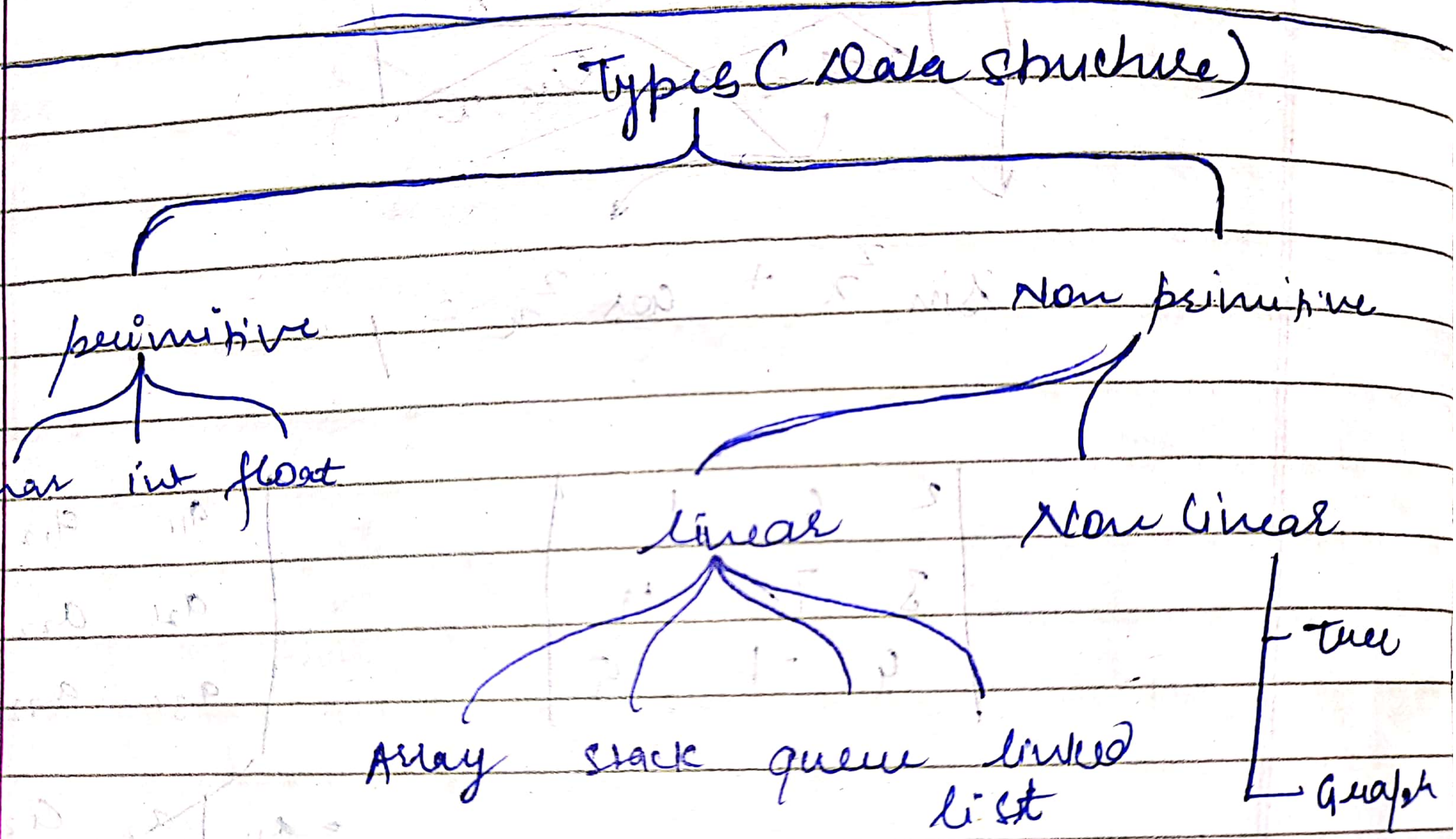
DS



Scanned by CamScanner

# Data Structure And Algorithms.

19-Aug.-19

Data structure and algorithms
↓

1. Data :→ Data is collection of raw facts and set of value.

Information :→ useful ~~use~~ data.
Data after processing.

Structure :→ How we are going to organize/store data in computer memory

Algorithms :→ sequence of statements to perform a particular task.

↳ Koi logic describe karta hai.

Data structure :→

Logical or mathematical way to organise the data in memory.

Types of operation on ] → Data structure.

Scanned by CamScanner

→ Memory requird.

→ range

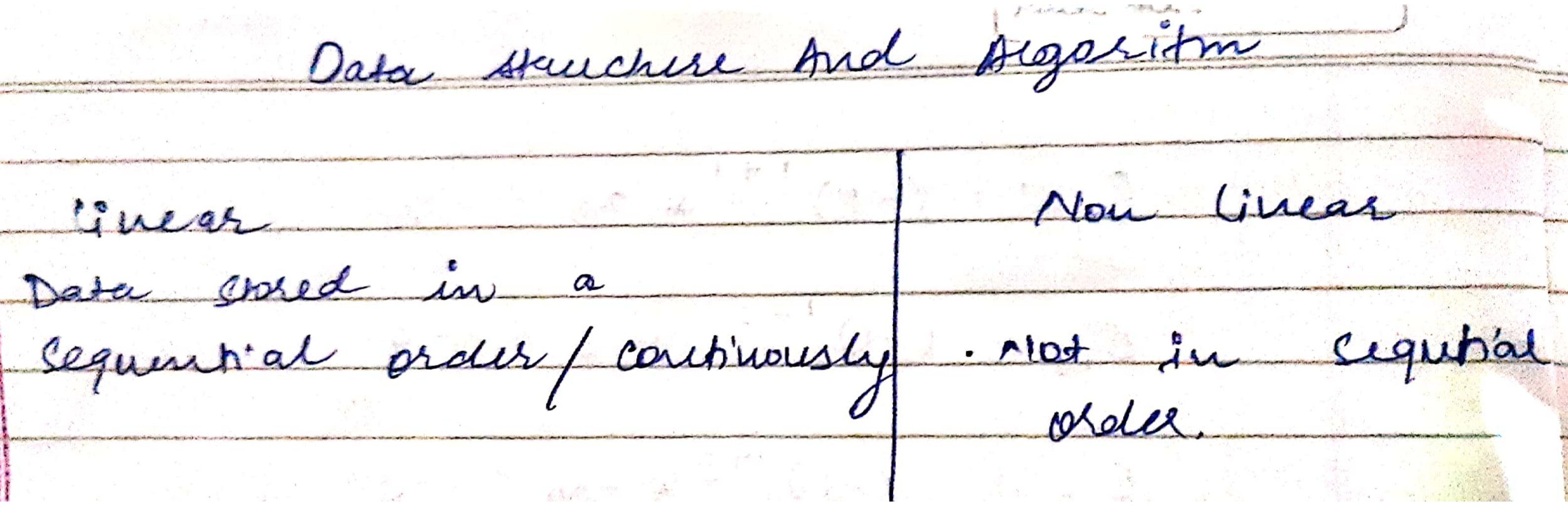
DATE : Data structure through C

PAGE NO. : → Madhu il

→ Schaum's series

Operations on Data base.

1. creation
2. Insersion
3. Deletion
4. searching
5. sorting
6. merging



# Data Structure And Algoritm

| Linear | Non Linear |
|---|---|
| Data stored in a sequential order / continuously | • Not in sequtial order. |

Scanned by CamScanner

Rules to ~~start~~ write algo.

1. begin / start
2. End / exit
3. Variable should be in capital letters.
4. Use declare to declare a variable
5. if cond.
6. repeat for i = _ to _

Scanned by CamScanner

2-19

# DATA STRUCTURE AND ALGORITHM

## Analysis of Algorithm

**Space complaxity**

→ ~~time~~ space seq. to execute an Algo.

**Time complexity**

→ time seq. to execute an Algo

parameters

- → Depends on user.
- → Depends on circumstance.
- → depends on hardware
- → Depenes on process
- → depends on environment + / pretend.

* Almost impossible to find exact time to execute an Algo. because it depends on so many factor.

Time Comp.

- Best case
- Avg.
- Wrost.

Scanned by CamScanner

# SORTING

- Simple Sorting.
- Bubble sorting
- Selection
- Insertion
- quick
- merge

To arrange data in a particular order.

$j=0$

$i<n-1-i$ /

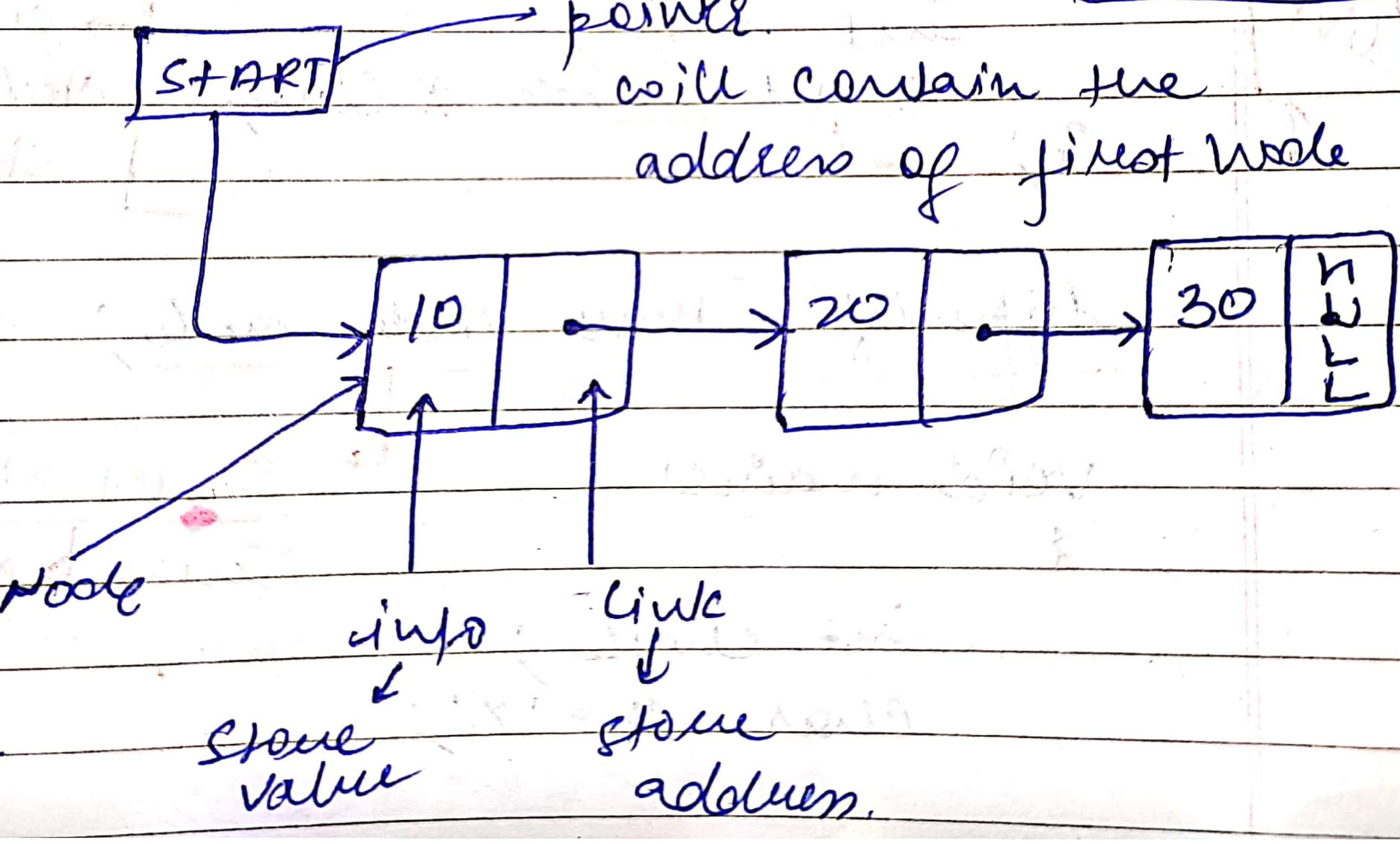
$[j]$ $[j+1]$

| Linked List | Array |
|---|---|
| Dynamic Memory | Static Memory alloc |
| No need of shifting in insertion & deletion | shifting in insertion & deletion. |
| difficult to implement and maintain due to pointer. | |
| ONE DIRECTIONAL | |
| more memory will be needed by variables because of pointers | |
| No Direct acces | Direct acces. |

Scanned by CamScanner

Linklist is a collection of nodes where each node will contain two pars first part will contain value and Second part will contain address of another node.

array → | 10 | 20 | 30 |



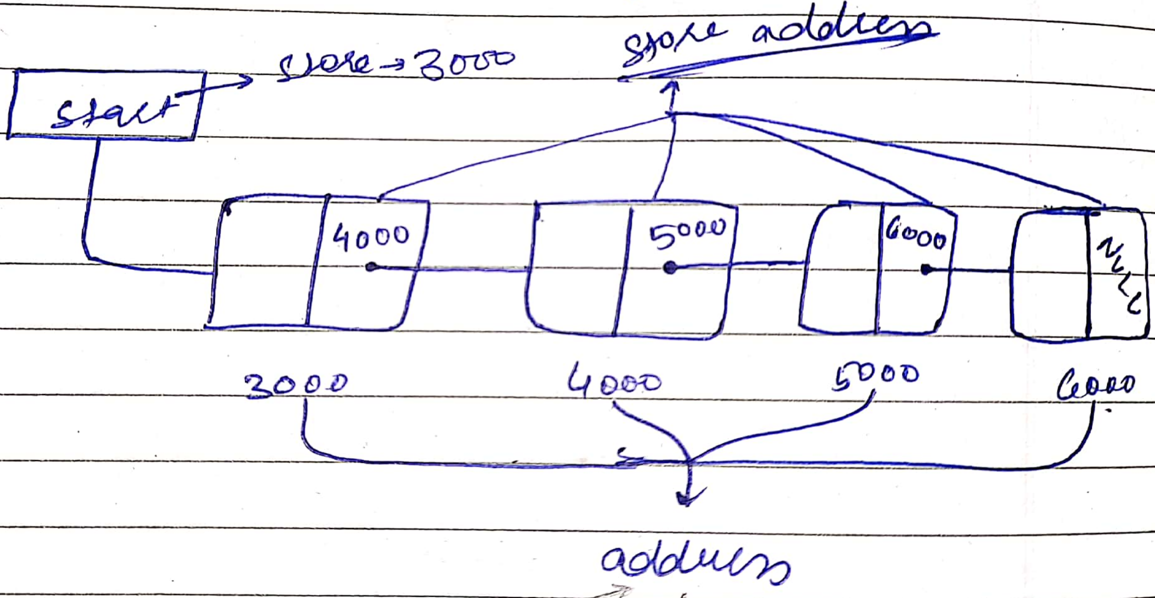
pointer will contain the address of first node

| Static | Dynamic |
|---|---|
| Complile time | Run Time<br>( add or release memory) |
| new } C++<br>delete | C — 1. malloc<br>calloc ] - provide memory ] → runtime<br>2 free() - release memory |

Scanned by CamScanner

pucob in allay.

* Static Memory allocation.



# DSA

Arithmatic Notation / equation

1. infix expression.
2. postfix "
3. prefix "

A+B → infix.
AB+ → post.
+AB → prefix.

A+B * C/D

Priority + Associativity

+ And - lowest
* And / And % ② medium
^ → caret for power — Maximum

* [ all left to right with same priority.

CONVERSION OF INFIX INTO POSTFIX

A+B*C/D     ~~A+BC*D/+~~

~~A+*~~ ~~(ABF C/D/)~~ *

~~A BC*D/+~~

A+BC*/D
A+BC*D/
ABC*D/+

Scanned by CamScanner

Infix to post fix conversion by using STACK

Left mai left parenthes right mai right.

scan left to right

if left parenthesis come sent stack
Operand → postfix.

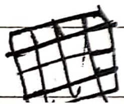
if operator comes check stack if no operator found push ~~stop~~ operator into stack.

else check periority. then compare

Scanned by CamScanner

if top periorit >= pop top operator send it into postfix.

else new will stay into stack.
both will stay.



Add ) to right side

Scan postfix expression from left to right.

if an oprend is encounter then push oprend into stack

If an operator is encounter then pop top 2 elements from stack and perform following operation.

result = [TOP-1] Operator [TOP]

push this result into stack

Repeat Above two steps untill a ~~right~~ ) encounter

if a ) is encounter then pop top elements from stack print as a result.

## QUEUE

It is a linear data structure in which insertion & deletion are performed on different ends.

These ends are known as front & REAR

FRON - it is ~~those~~ used to store position of first element (oldest element) in Queue.

REAR - Is used to store the pos. of last element in Queue

Scanned by CamScanner

Queue works on FIFO principle

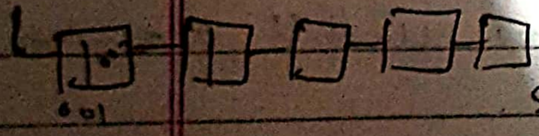
Queue can be impliments by using array and linked list.

## APPlication Of Queue

→ Used to impliment time sharing O/S

→

Types OF QUEUE



Simple Q
Circular
DE-Quee (Double ended)
Periority Q.

Scanned by CamScanner