**Data Structures**

* It is a data organization , management and storage format that enables efficient access and modification.
* Or , a data-structure is a particular way of organizing data in a computer so that it can be effectively used .

**Abstract data types**

* It is made up of primitive data-types , but operation logics are hidden
* Special kind of data-types whose behavior is defined by a set of values and set of instructions.
* Ex :- Lists , stack , queue

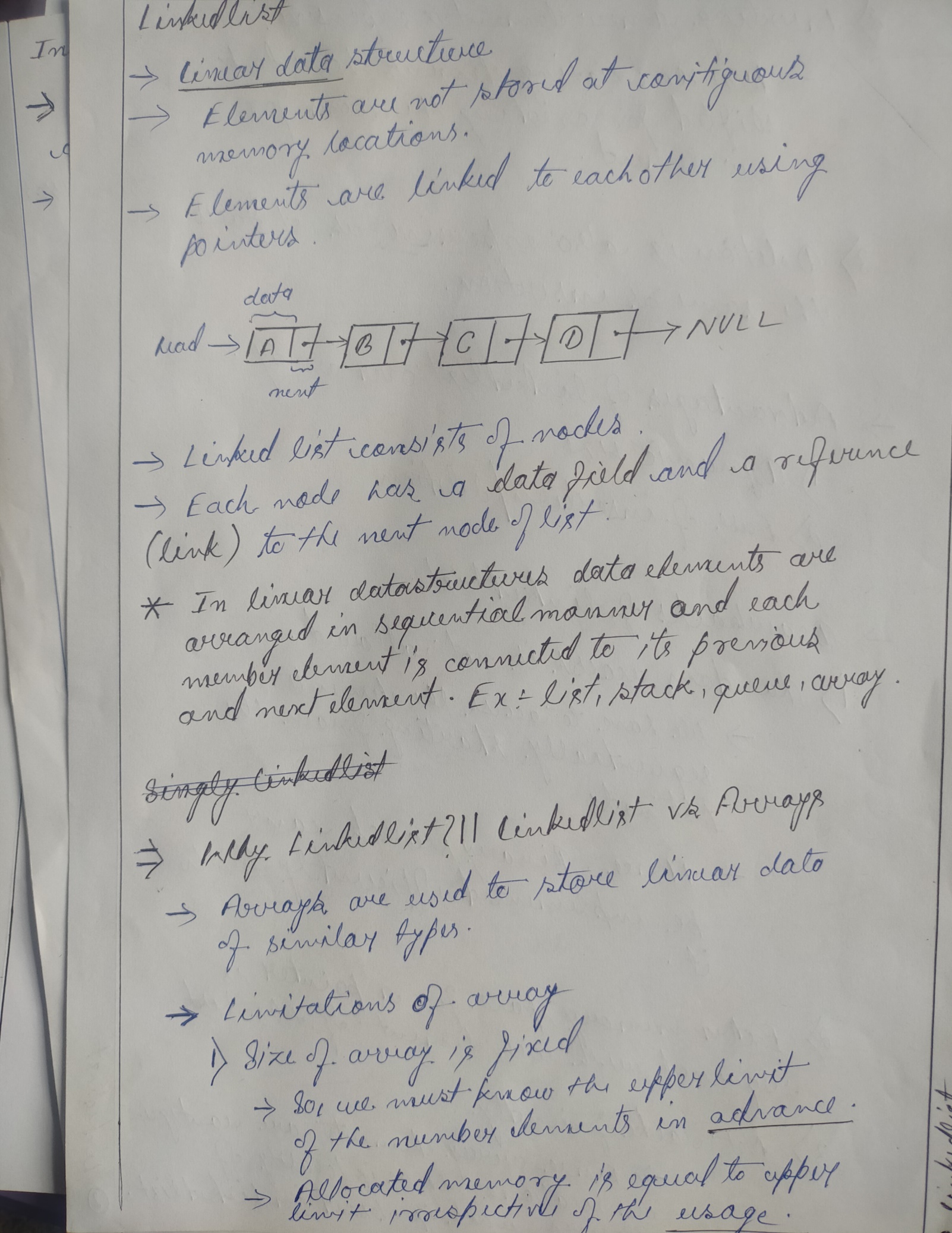
**Linked lists**

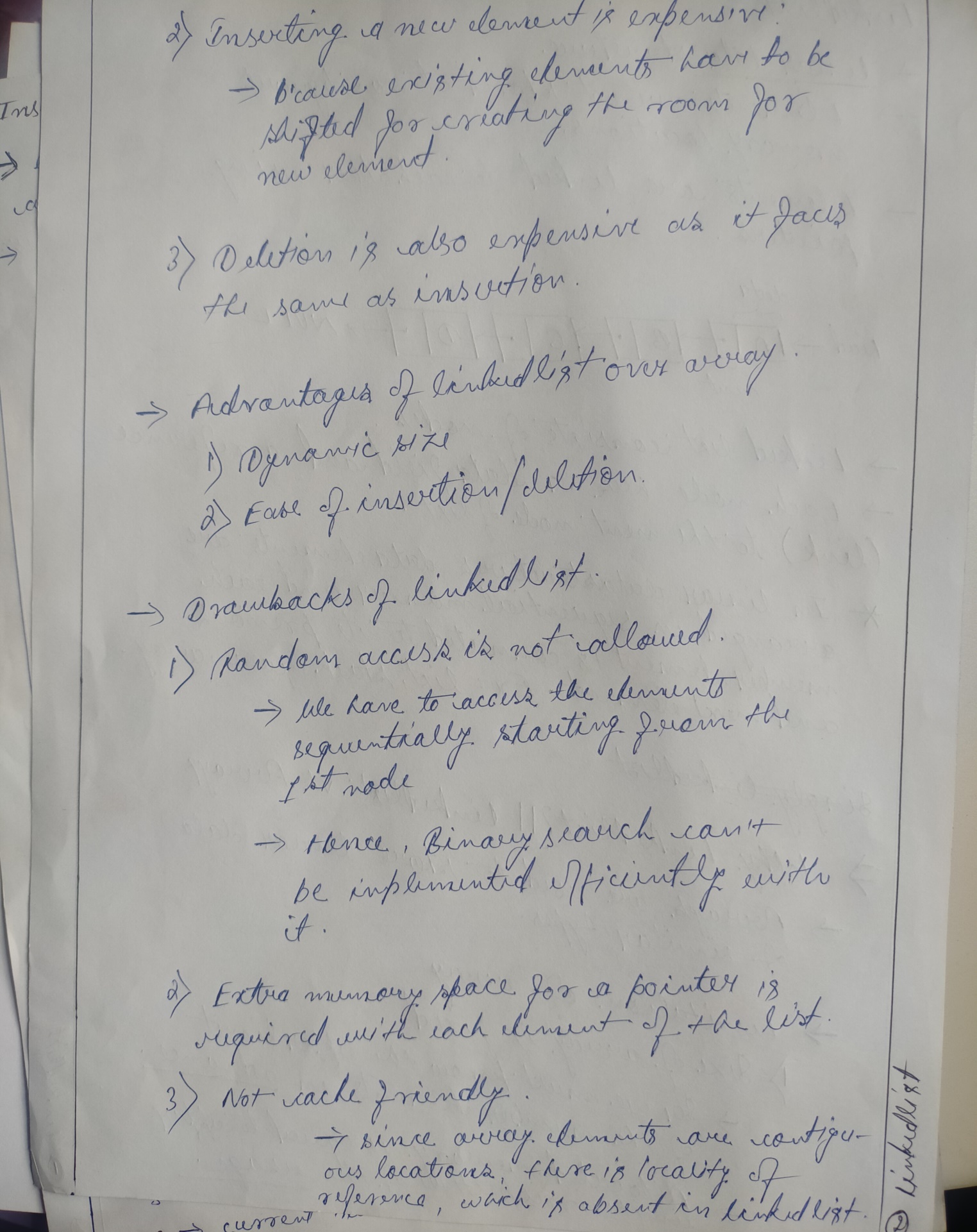
Types of linked lists

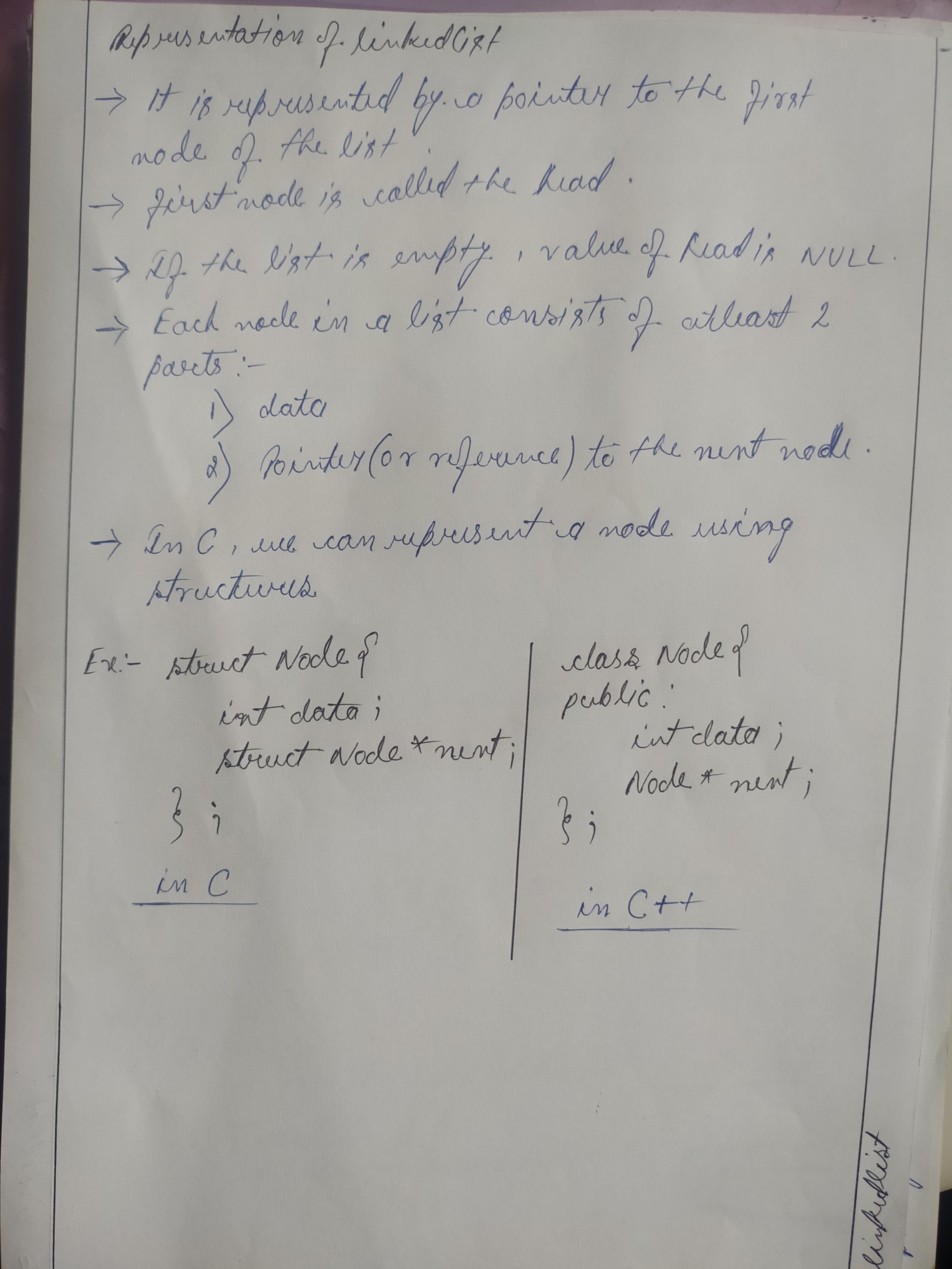
1. Singly Linked list
2. Doubly Linked list
3. Circular Linked list

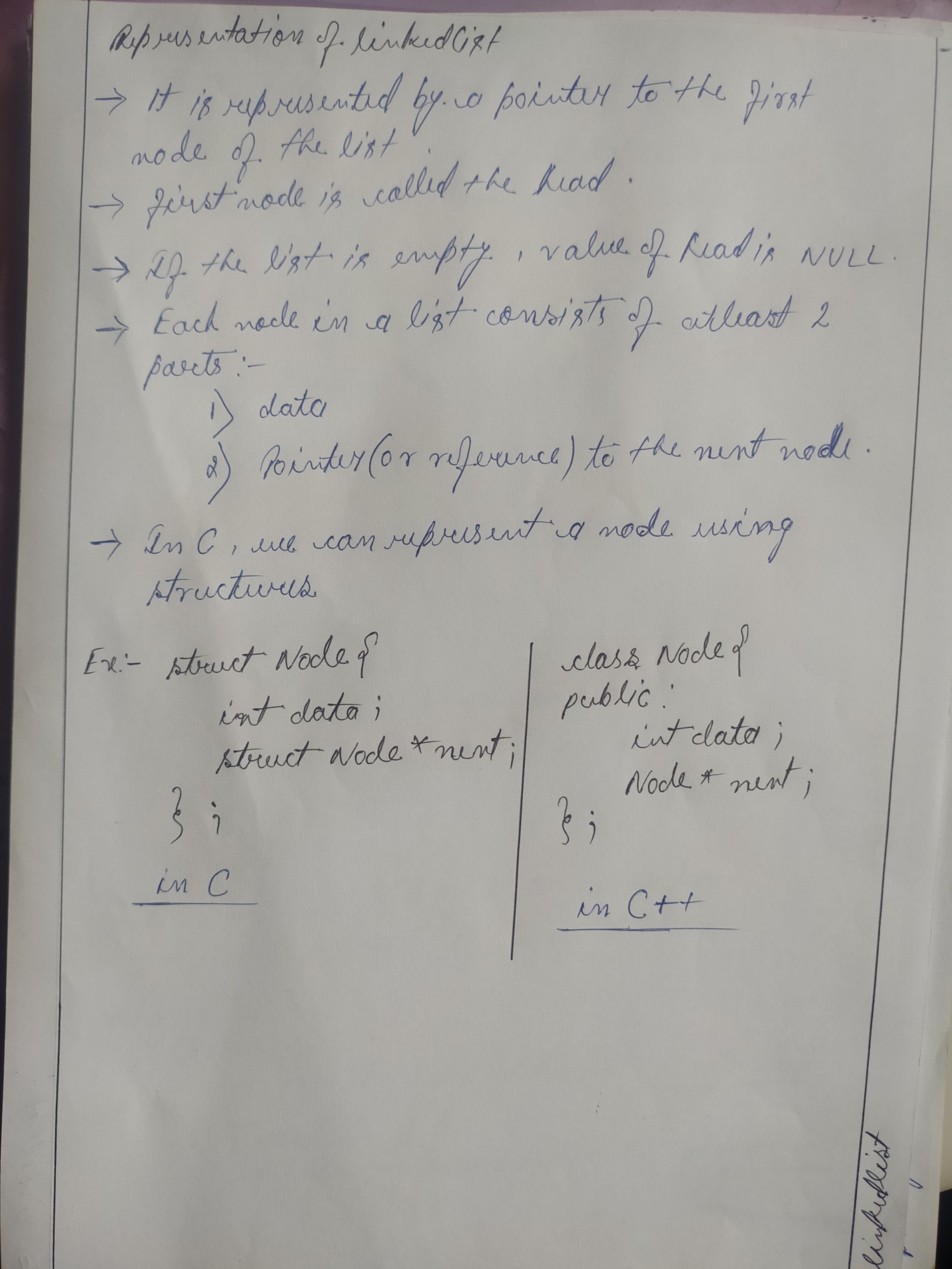
Each type will consist at least following functions to operate on the list-elements

* Create – to create the list (Although , one can create a simple list directly in the main function without any user-defined function)
* InsertatFirst – To insert the element at the beginning of the list
* InsertAfter – To insert the node after a specific node
* InsertatLast – to insert at the last
* DeleteFirst
* DeleteLast
* DeleteElement – Delete the node with specific data
* Reverse
* Traverse – to go through the nodes in order to operate them or print them

****

****

****

****

**Singly Linked list**

* It is the standard type of linkedlist , in which there are two fields in each element one is data and another one is next node pointer .
* Unlike doubly linked list , this is an one-way list .

***typedef struct node{***

***int info ;***

***struct node \* next ;***

***}node;***

**Algorithm create(head , n)**

input : head – pointer to the first node of the list (*initially head = NULL*)

n – number of nodes/elements in the list

1 . if head != NULL then

2 . print(“List is already created”)

3 . return

4 . end if

5 . for i <- 1 to n do

6 . input item

7 . newnode <- getNode()

8 . info[newnode] <- item

9 . next[newnode] <- NULL

10 . if head = NULL then

11 . head <- newnode

12 . else

13 . next[temp] <- newnode

14 . temp <- newnode

15 . end for

16 . return

**void create(node \*\* head , int n)**

{

node \* newnode , \* temp ;

int item , i ;

if(\*head!= NULL)

{

printf(“List created\n”);

return ;

}

for(i=1 ; i<=n ; i++)

{

printf(“Enter item = ”);

scanf(“%d”,&item);

newnode=(node\*)

malloc(sizeof(node));

newnode -> info = item ;

newnode -> next = NULL ;

if((\*head)==NULL)

(\*head) = newnode ;

else

temp -> next = newnode ;

temp = newnode ;

}

return ;

}

**void insertatFirst(node \*\* head , int item)**

{

if((\*head) == NULL)

{

printf(“List Empty\n”);

return;

}

node \* newnode = (node\*)malloc(sizeof(node));

newnode -> info = item ;

newnode -> next = \*head

(\*head) -> newnode ;

return;

}