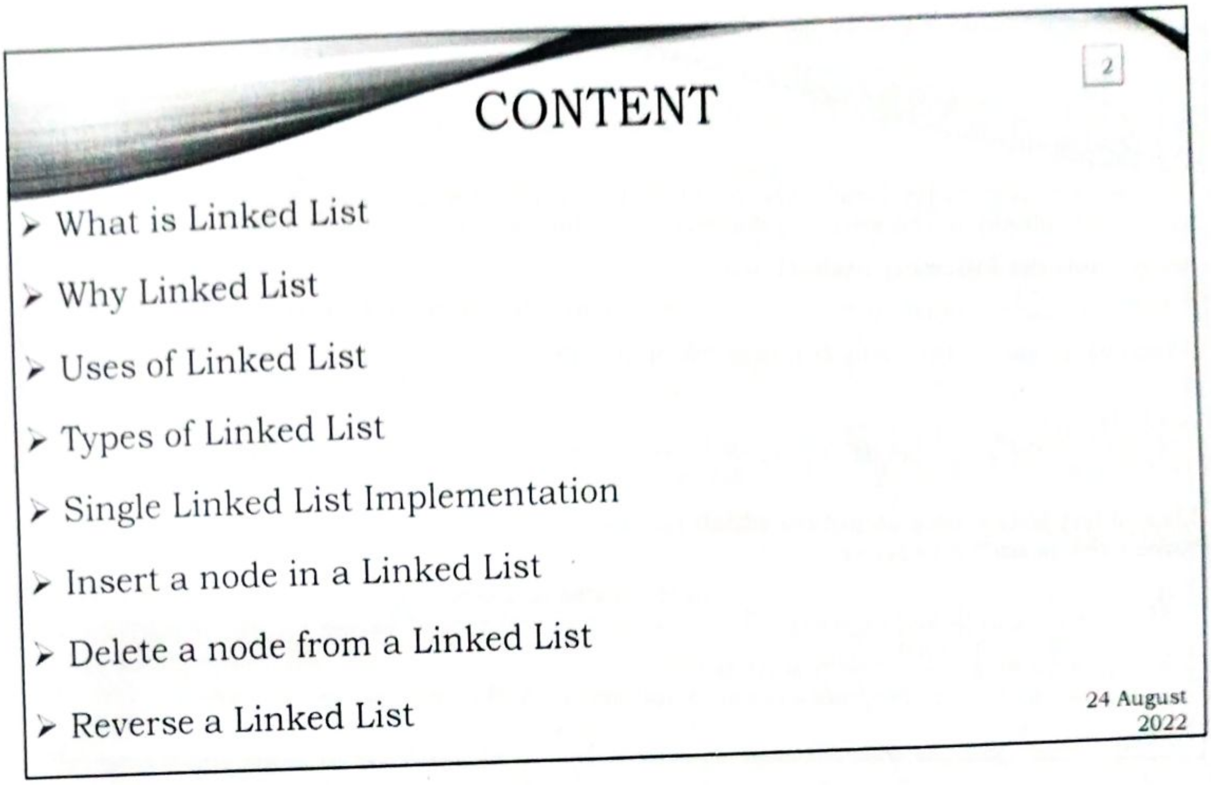
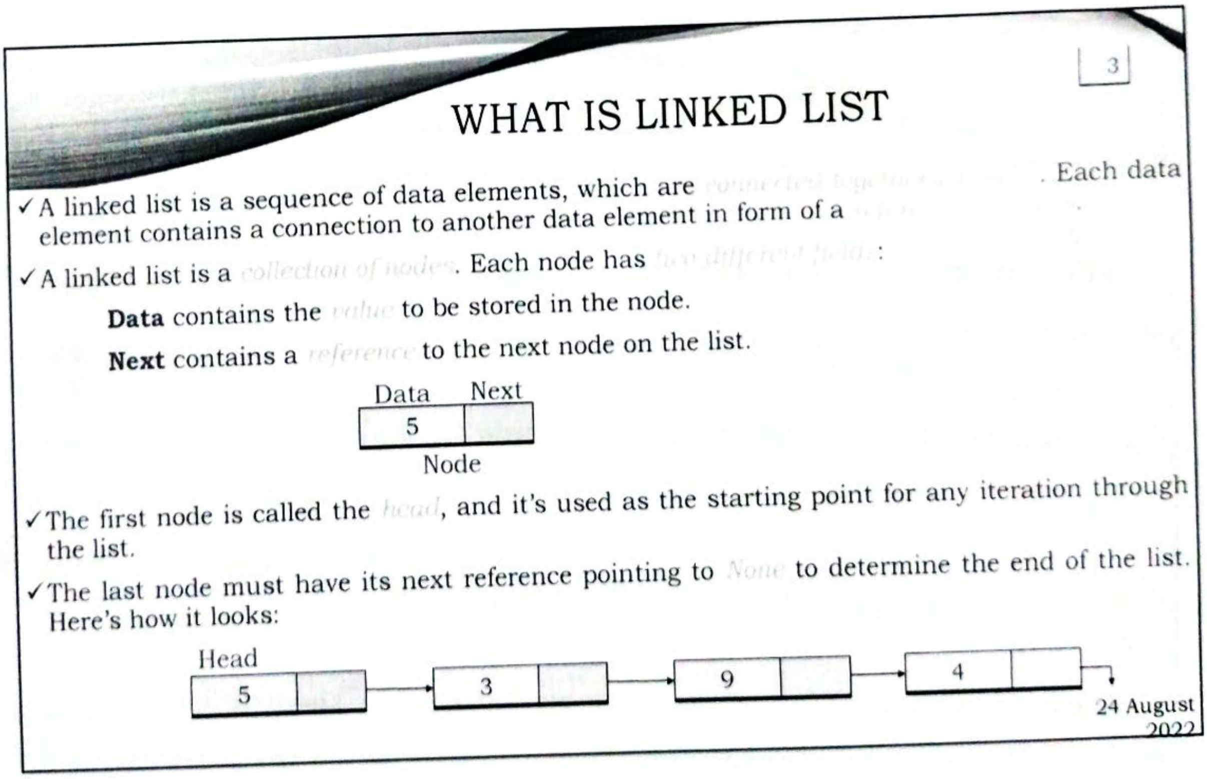


2

CONTENT

- ➢ What is Linked List
- ➢ Why Linked List
- ➢ Uses of Linked List
- ➢ Types of Linked List
- ➢ Single Linked List Implementation
- ➢ Insert a node in a Linked List
- ➢ Delete a node from a Linked List
- ➢ Reverse a Linked List

24 August 2022



3

WHAT IS LINKED LIST

✓ A linked list is a sequence of data elements, which are connected together via links. Each data element contains a connection to another data element in form of a reference pointer.

✓ A linked list is a collection of nodes. Each node has two different fields:

Data contains the value to be stored in the node.

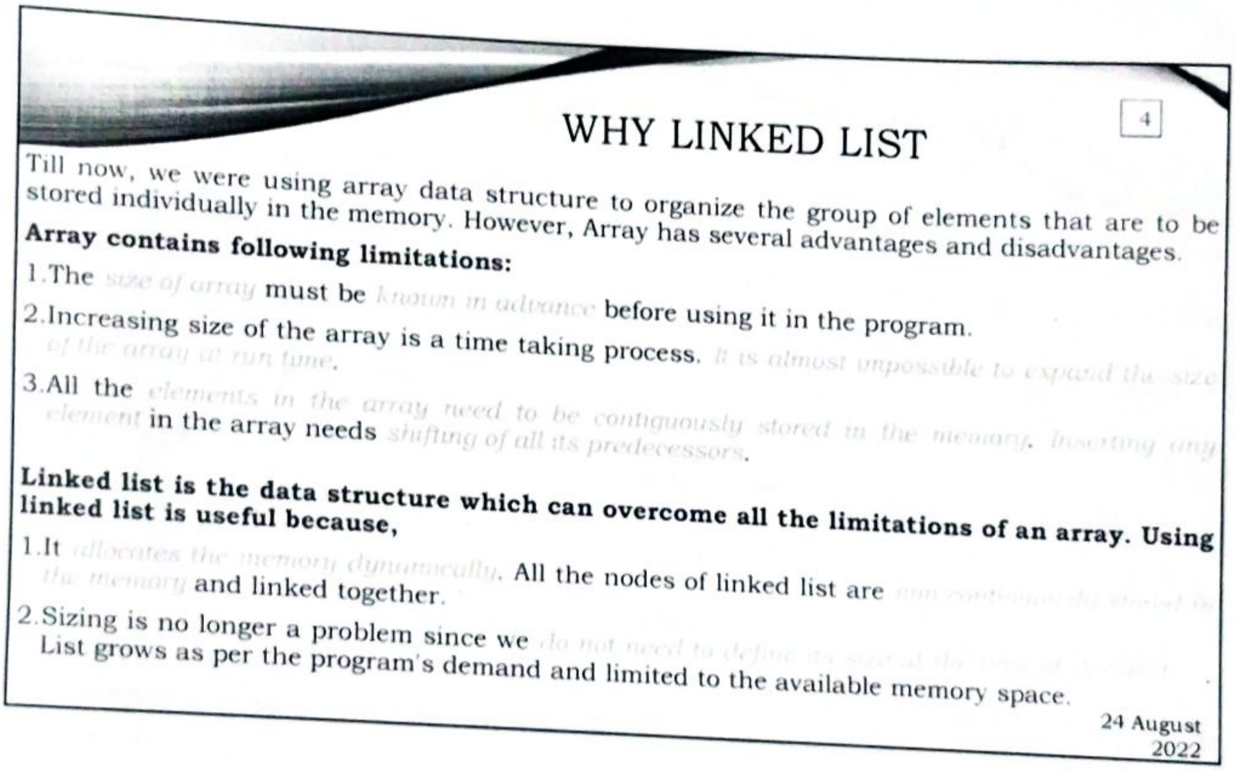
Next contains a reference to the next node on the list.

✓ The first node is called the head, and it's used as the starting point for any iteration through the list.

✓ The last node must have its next reference pointing to None to determine the end of the list. Here's how it looks:

24 August 2022

24-8



WHY LINKED LIST

4

Till now, we were using array data structure to organize the group of elements that are to be stored individually in the memory. However, Array has several advantages and disadvantages.

Array contains following limitations:

1.The *size of array* must be *known in advance* before using it in the program.

2.Increasing size of the array is a time taking process. *It is almost impossible to expand the size of the array at run time.*

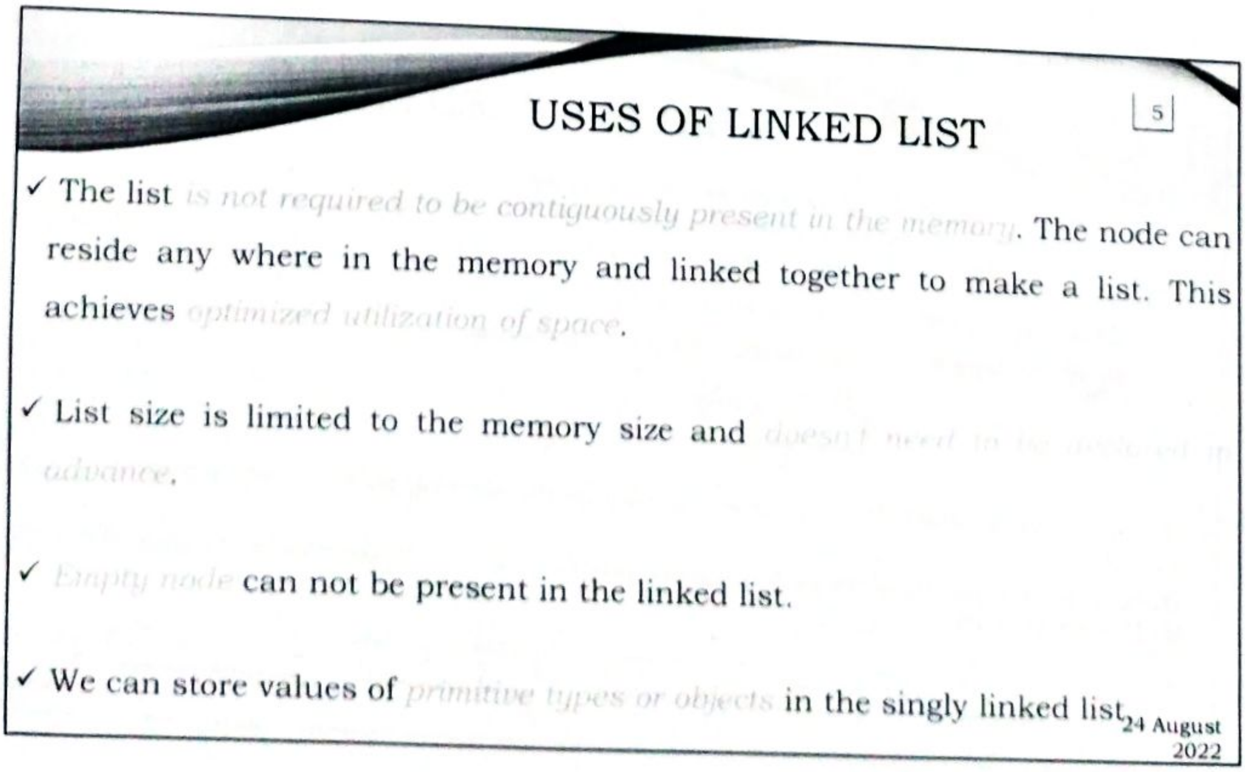
3.All the *elements in the array need to be contiguously stored in the memory. Inserting any element* in the array needs *shifting of all its predecessors.*

Linked list is the data structure which can overcome all the limitations of an array. Using linked list is useful because,

1.It *allocates the memory dynamically.* All the nodes of linked list are *non-contiguously stored in the memory* and linked together.

2.Sizing is no longer a problem since we *do not need to define its size at the time of declaration.* List grows as per the program's demand and limited to the available memory space.

24 August 2022



USES OF LINKED LIST

5

✓ The list *is not required to be contiguously present in the memory*. The node can reside any where in the memory and linked together to make a list. This achieves *optimized utilization of space.*

✓ List size is limited to the memory size and *doesn't need to be declared in advance.*

✓ *Empty node* can not be present in the linked list.

✓ We can store values of *primitive types or objects* in the singly linked list

24 August 2022

6

TYPES OF LINKED LIST

The following are the types of Linked List:

1. *Single Linked List*
2. *Doubly Linked List*
3. *Circular Linked List*
4. *Doubly Circular Linked List*

Single Linked List: It is the commonly used linked list in programs. If we are talking about the linked list, it means it is a singly linked list. The singly linked list is a data structure that contains two parts, i.e., *one is the data part, and the other one is the address part*, *which contains the address of the next or the successor node*. The address part in a node is also known as a *link field*.

One way linked list or singly linked list can be *traversed only in one direction*. In other words, we can say that each *node contains only next address*, therefore we *can not traverse the list in the reverse direction*.

24 August 2022

7

CONTINUE

Double Linked List: The doubly linked list contains two pointers or reference. We can define the doubly linked list as a linear data structure with *three parts: the data part and the other two address part*. In other words, a doubly linked list is a list that has three parts in a single node, includes *one data part, a reference to its previous node, and a reference to the next node*.

Circular Linked List: A circular linked list is a variation of a singly linked list. The only difference between the ***singly linked list*** and a ***circular linked*** list is that *the last node does not point to any node in a singly linked list, so its link part contains a None value*. On the other hand, the *circular linked list* is a list in which *the last node connects to the first node, so the link part of the last node holds the first node's address.* The circular linked list has no starting and ending node. We can traverse in any direction, i.e., *either backward or forward*.

Double Circular Linked List: It is a *doubly linked list* also because *each node holds the address of the previous node also*. The main difference between the doubly linked list and doubly circular linked list is that *the doubly circular linked list does not contain the* ***None*** *value in the previous field of the node*. As the doubly circular linked contains three parts, i.e., *two address parts and one data part so its representation is similar to the doubly linked list*.

24 August 2022