



1.4.2 Linked Lists

A linked list is a data structure where data is stored in **nodes**, with each node containing a **pointer** to the next node in the list.

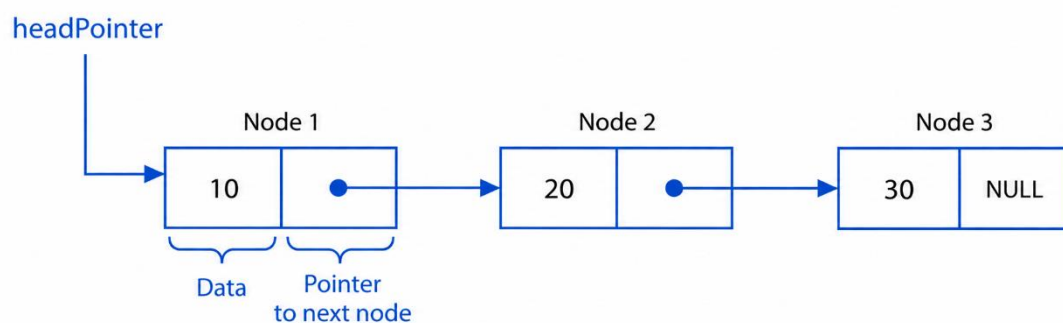
Structure of a Linked List

Each node contains:

- **Data** – the value being stored.
- **Pointer** – the location of the next node.

The **head pointer** stores the location of the first node in the link list.

A **NULL pointer** indicates the end of the link.



Traversing a Link List

1. Start at the node indicated by the head pointer.
2. Access the data.
3. Follow the pointer to the next node.
4. Repeat until the target node is reached or the pointer is NULL.

Adding a Node

A node can be added to a linked list by inserting the new node between two nodes in the Linked List.

- Traverse the Linked List to where the new node should be inserted.
- The new node's pointer is changed to point to the node that should follow after.
- The pointer of the node before is changed to point to the new node.





Removing a Node

1. Traverse the Linked List to the node that is to be removed.
2. Change the pointer of the node before to point to the node that is currently pointed to by the node to be deleted.
3. Add the removed node to the free nodes.

Advantages of using Linked Lists

- Nodes do not need to be stored in consecutive memory locations.
- Nodes can easily be added or removed by changing the pointers.
- The Linked List can grow or shrink dynamically.
- Inserting or deleting nodes does not require the other data to be shifted.

Disadvantages of using Linked Lists

- Each node requires extra memory to store a pointer.
- Elements cannot be accessed directly using an index.
- To find a particular node, the list has to be traversed from the beginning

