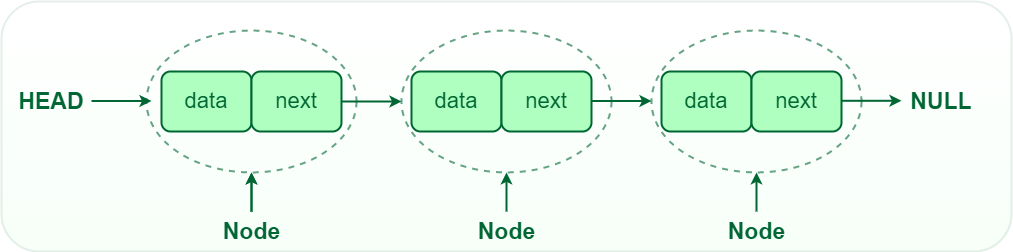
***Step 6: Learning LinkedList***

* **What is LinkedList?**

A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers as shown in the below image:



In simple words, a linked list consists of nodes where each node contains a data field and a reference(link) to the next node in the list.

**Reference:** <https://www.geeksforgeeks.org/what-is-linked-list/>

* **Operations on Linked Lists**

1. [**Insertion**](https://www.geeksforgeeks.org/insertion-in-linked-list/)**:**Adding a new node to a linked list involves adjusting the pointers of the existing nodes to maintain the proper sequence. Insertion can be performed at the beginning, end, or any position within the list.
2. [**Deletion**](https://www.geeksforgeeks.org/deletion-in-linked-list/)**:** Removing a node from a linked list requires adjusting the pointers of the neighbouring nodes to bridge the gap left by the deleted node. Deletion can be performed at the beginning, end, or any position within the list.
3. [**Searching**](https://www.geeksforgeeks.org/search-an-element-in-a-linked-list-iterative-and-recursive/)**:** Searching for a specific value in a linked list involves traversing the list from the head node until the value is found or the end of the list is reached.

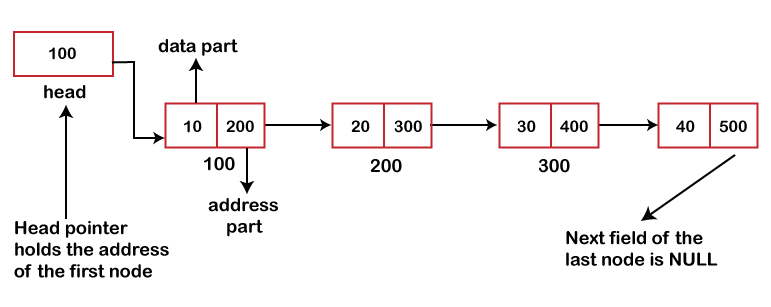
* [**Types of linked lists**](https://www.geeksforgeeks.org/types-of-linked-list/)**:**

There are mainly three types of linked lists:

1. Single-linked list
2. Double linked list
3. Circular linked list

**6.1 Learning Singly LinkedList**

In a singly linked list, each node contains a reference to the next node in the sequence. Traversing a singly linked list is done in a forward direction.

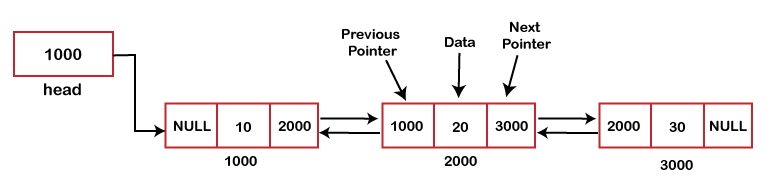
****

* **Characteristics of a Singly Linked List:**
* Each node holds a single value and a reference to the next node in the list.
* The list has a head, which is a reference to the first node in the list, and a tail, which is a reference to the last node in the list.
* The nodes are not stored in a contiguous block of memory, but instead, each node holds the address of the next node in the list.
* Accessing elements in a singly linked list requires traversing the list from the head to the desired node, as there is no direct access to a specific node in memory.

**Program:**

**6.2 Learning Doubly LinkedList**

In a doubly linked list, each node contains references to both the next and previous nodes. This allows for traversal in both forward and backward directions, but it requires additional memory for the backward reference.



* **Advantages of Doubly LinkedList over the Singly LinkedList:**
* A DLL can be traversed in both forward and backward directions.
* The delete operation in DLL is more efficient if a pointer to the node to be deleted is given.
* We can quickly insert a new node before a given node.
* In a singly linked list, to delete a node, a pointer to the previous node is needed. To get this previous node, sometimes the list is traversed. In DLL, we can get the previous node using the previous pointer.
* **Disadvantages of Doubly LinkedList over the Singly LinkedList:**
* Every node of DLL Requires extra space for a previous pointer. It is possible to implement DLL with a single pointer though (See [this](https://www.geeksforgeeks.org/xor-linked-list-a-memory-efficient-doubly-linked-list-set-1/)and [this](https://www.geeksforgeeks.org/xor-linked-list-a-memory-efficient-doubly-linked-list-set-2/)).
* All operations require an extra pointer previous to be maintained. For example, in insertion, we need to modify previous pointers together with the next pointers. For example, in the function for insertions at different positions, we need 1 or 2 extra steps to set the previous pointer.