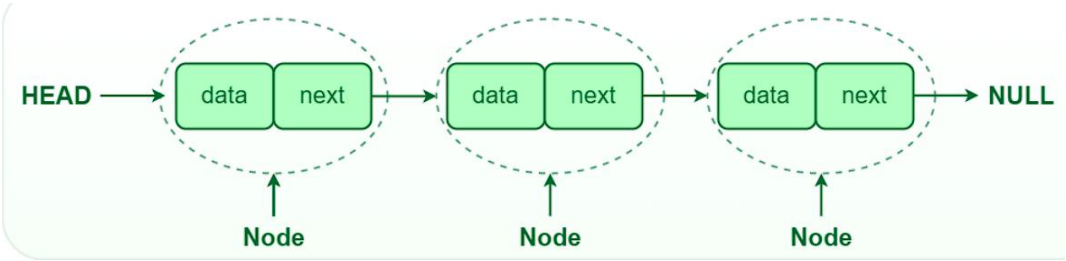
Linked Li

Linked List

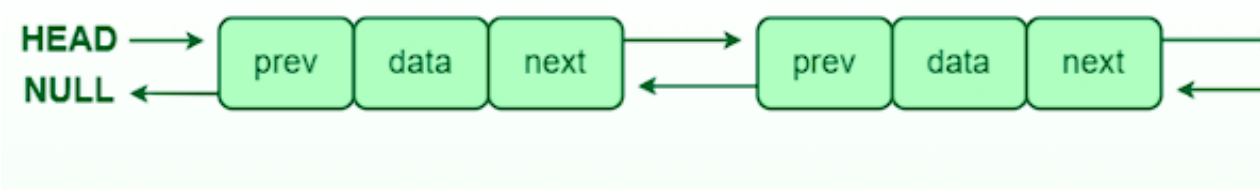
- *A Linked List* is a linear data structure which lo
  node is a different element. Unlike *Arrays*, *Lin*
  contiguous location.

Types of Linked List

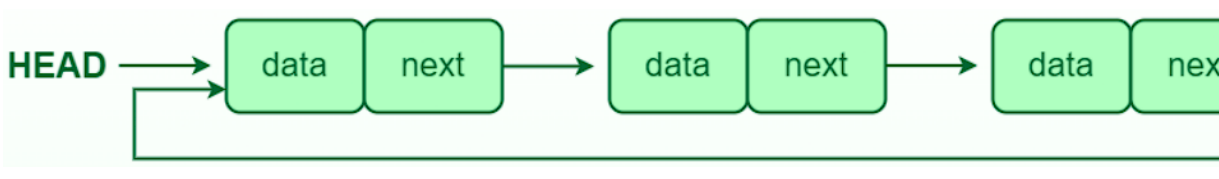
- **Singly linked list**
  - In a singly linked list, each node contains a reference to the next nod forward direction.



- **Doubly linked list**
  - In a doubly linked list, each node contains references to both the nex forward and backward directions.

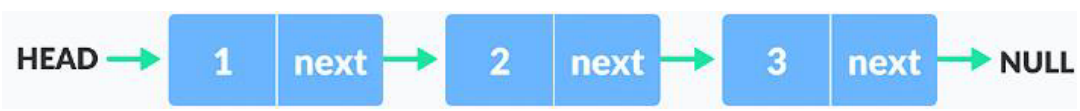


- **Circular linked**
  - In a circular linked list, the last node points back to the head node, cr linked



Linked List representation | Si

```
struct node
{
  int data;
  struct node *next;
};
```




```
/* Ini
struct
struct
struct
struct

/* All
one =
two =
three

/* As
one->
two->
three

/* Co
one->
two->
three

/* Sa
head
```

Insert at the beginning

```
void insertAtBeginning(T data){
      struct node *newNode;
      newNode = malloc(sizeof(struct n
      newNode->data = data;
      newNode->next = head;
      head = newNode;
}
```

Insert at the end

```
void insertAtEnd(T data)
{
        struct node *newNode;
        newNode = malloc(sizeof(struct node));
        newNode->data = data;
        newNode->next = NULL;

        struct node *temp = head;
        while(temp->next != NULL){
                temp = temp->next;
        }

        temp->next = newNode;
}
```

Insert after

```
void insertAfter(T data, T previous)
{
        struct node *newNode;
        newNode = malloc(sizeof(struct node));
        newNode->data = data;

        struct node *currentNode = head;
        while(currentNode != null && currentNode->data !=
                currentNode = currentNode->next;
        }

        if(currentNode != null){
                newNode->next = currentNode->next
                currentNode->next = newNode
        }

}
```

Print List

```
void printList(){
        struct node *temp = head;

        while(temp != NULL) {
                printf(temp->data);
                temp = temp->next;
        }
}
```

Delete at head

```
void deleteAtHead(){
    head = head->next;
}
```

Delete by value

```
void deleteByValue(T data){
        struct node *currentNode = head;
        struct node *prevNode = null;

        if(currentNode.data == data)
                deleteAtHead()

        while(currentNode != null){
                if(currentNode->data == data){
                        prevNode->next = currentNode->
                        break;
                }

        prevNode = currentNode
        currentNode = currentNode->next
        }
}
```

Search

```
int searchlist(T value){
        struct node *temp = head;
        while(temp != NULL) {
        if (temp->data == value) {
                return 1;
        }
        temp=temp->next;
        }
        return 0;
}
```