Types of Linked Lists - Implementation Guide

Introduction

A linked list is a linear data structure in which elements, called nodes, are connected using pointers. Each node consists of two parts:

1. **Data**: The value stored in the node.
2. **Pointer**: A reference to the next (or previous) node in the sequence.

Below are the various types of linked lists and their corresponding implementation in C.

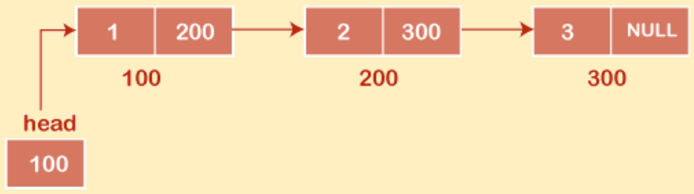
---

1. Singly Linked List

Structure:

A singly linked list has nodes containing:

- **Data**
- **Pointer to the next node**



Code Example:

```
#include <stdio.h>
#include <stdlib.h>

// Define the structure of a node
struct Node {
    int data;
    struct Node* next;
};

// Function to create a new node
```

```
struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

// Function to traverse and print the linked list
void printList(struct Node* head) {
    struct Node* temp = head;
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}

int main() {
    struct Node* head = createNode(1);
    head->next = createNode(2);
    head->next->next = createNode(3);

    printf("Singly Linked List: \n");
    printList(head);

    return 0;
}
```

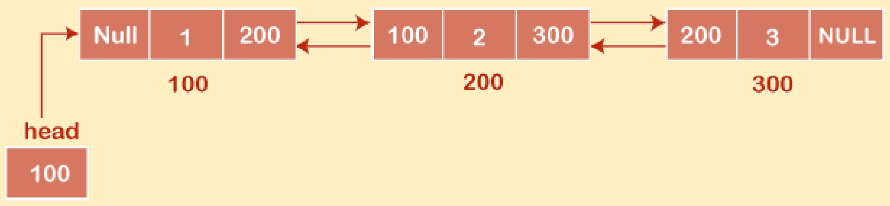
---

2. Doubly Linked List

Structure:

A doubly linked list has nodes containing:

- **Data**
- **Pointer to the next node**
- **Pointer to the previous node**



Code Example:

```
#include <stdio.h>
#include <stdlib.h>
```

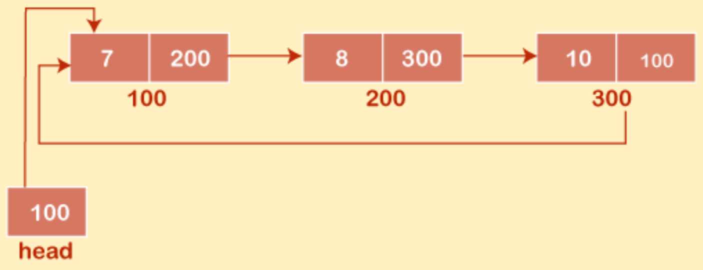
```c
// Define the structure of a node
struct Node {
    int data;
    struct Node* next;
    struct Node* prev;
};

// Function to create a new node
struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;
    newNode->prev = NULL;
    return newNode;
}

// Function to traverse and print the linked list
void printList(struct Node* head) {
    struct Node* temp = head;
    while (temp != NULL) {
        printf("%d <-> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}

int main() {
    struct Node* head = createNode(1);
    struct Node* second = createNode(2);
    struct Node* third = createNode(3);

    head->next = second;
    second->prev = head;
    second->next = third;
    third->prev = second;

    printf("Doubly Linked List: \n");
    printList(head);

    return 0;
}
```

3. Circular Linked List

Structure:

A circular linked list has nodes where:

- The last node points back to the first node.



Code Example:

```
#include <stdio.h>
#include <stdlib.h>

// Define the structure of a node
struct Node {
    int data;
    struct Node* next;
};

// Function to create a new node
struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

// Function to print the circular linked list
void printList(struct Node* head) {
    if (head == NULL) return;
    struct Node* temp = head;
    do {
        printf("%d -> ", temp->data);
        temp = temp->next;
    } while (temp != head);
    printf("HEAD\n");
}

int main() {
    struct Node* head = createNode(1);
    struct Node* second = createNode(2);
    struct Node* third = createNode(3);
    head->next = second;
    second->next = third;
    third->next = head;

    printf("Circular Linked List: \n");
    printList(head);

    return 0;
}
```

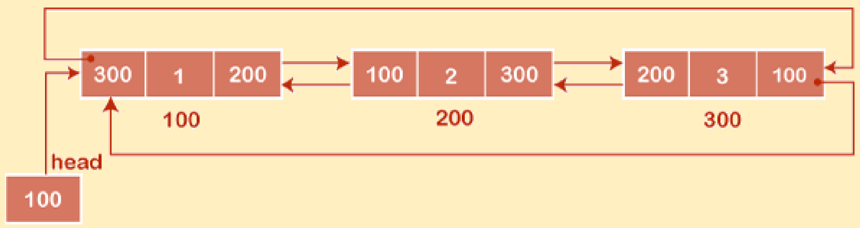
---

4. Doubly Circular Linked List

Structure:

A doubly circular linked list has nodes where:

- The last node points to the first node.
- The first node points back to the last node.



Code Example:

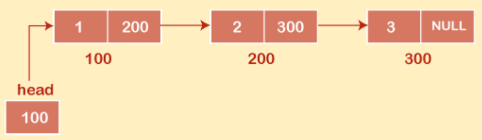
```
#include <stdio.h>
#include <stdlib.h>

// Define the structure of a node
struct Node {
    int data;
    struct Node* next;
    struct Node* prev;
};

// Function to create a new node
struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;
    newNode->prev = NULL;
    return newNode;
}

// Function to print the doubly circular linked list
```

```c
void printList(struct Node* head) {
    if (head == NULL) return;
    struct Node* temp = head;
    do {
        printf("%d <-> ", temp->data);
        temp = temp->next;
    } while (temp != head);
    printf("HEAD\n");
}

int main() {
    struct Node* head = createNode(1);
    struct Node* second = createNode(2);
    struct Node* third = createNode(3);

    head->next = second;
    second->prev = head;
    second->next = third;
    third->prev = second;
    third->next = head;
    head->prev = third;

    printf("Doubly Circular Linked List: \n");
    printList(head);

    return 0;
}
```

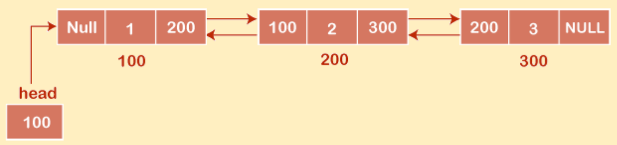
Summary of Differences

Type	Traversal	Pointer to Previous Node	Pointer to Next Node
Singly Linked List	Forward	No	Yes
Doubly Linked List	Forward & Backward	Yes	Yes
Circular Linked List	Forward	No	Yes
Doubly Circular List	Forward & Backward	Yes	Yes

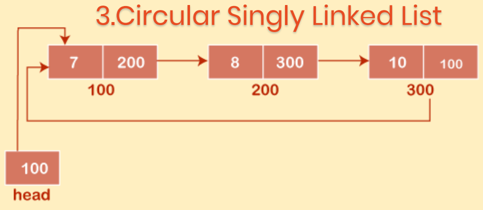
1.Singly Linked List



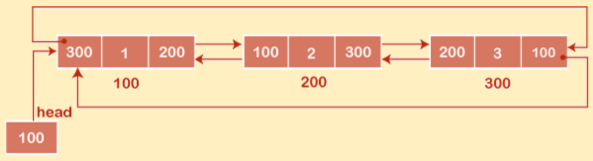
2.Doubly Linked List



3.Circular Singly Linked List



4.Circular Doubly Linked List


300
1
200
100
100
2
300
200
200
3
100
300
head
100