6. Circular Linked List

6.1 Structure

In a circular linked list, the last node points back to the first node (or head).

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

Visual Representation:

```
      ┌──────────────────────────┐
      ↓                          │
HEAD → [10|●] → [20|●] → [30|●] ──┘
```

6.2 Operations

6.2.1 Insert at Beginning

```
void insertAtBeginning(struct Node **head, int value) {
    struct Node *newNode = (struct Node*)malloc(sizeof(struct Node));
    if (newNode == NULL) {
        printf("Memory allocation failed!\n");
        return;
    }

    newNode->data = value;

    // If list is empty
    if (*head == NULL) {
        newNode->next = newNode;  // Points to itself
        *head = newNode;
        return;
    }
```

```
    // Find last node
    struct Node *temp = *head;
    while (temp->next != *head) {
        temp = temp->next;
    }

    // Insert at beginning
    newNode->next = *head;
    temp->next = newNode;
    *head = newNode;
}
```

6.2.2 Display Circular List

```
void displayCircular(struct Node *head) {
    if (head == NULL) {
        printf("List is empty!\n");
        return;
    }

    struct Node *temp = head;
    printf("List: ");

    do {
        printf("%d", temp->data);
        temp = temp->next;
        if (temp != head) {
            printf(" → ");
        }
    } while (temp != head);

    printf(" → (back to %d)\n", head->data);
}
```

6.2.3 Delete Node

```
void deleteNode(struct Node **head, int value) {
    if (*head == NULL) {
        printf("List is empty!\n");
```

```
    return;
}

struct Node *temp = *head;
struct Node *prev = NULL;

// If head node contains the value
if (temp->data == value) {
    // If only one node
    if (temp->next == *head) {
        free(temp);
        *head = NULL;
        return;
    }

    // Find last node
    while (temp->next != *head) {
        temp = temp->next;
    }

    // Delete head
    temp->next = (*head)->next;
    free(*head);
    *head = temp->next;
    return;
}

// Search for the node
prev = *head;
temp = (*head)->next;

while (temp != *head && temp->data != value) {
    prev = temp;
    temp = temp->next;
}

// If value not found
if (temp == *head) {
    printf("Value %d not found!\n", value);
    return;
```

```
    }

    // Delete node
    prev->next = temp->next;
    free(temp);
}
```

6.2.4 Count Nodes

```
int countNodes(struct Node *head) {
    if (head == NULL) {
        return 0;
    }

    int count = 1;
    struct Node *temp = head->next;

    while (temp != head) {
        count++;
        temp = temp->next;
    }

    return count;
}
```

Revision #1
Created 2025-10-27 05:05:07 UTC by DS
Updated 2025-10-27 05:05:20 UTC by DS