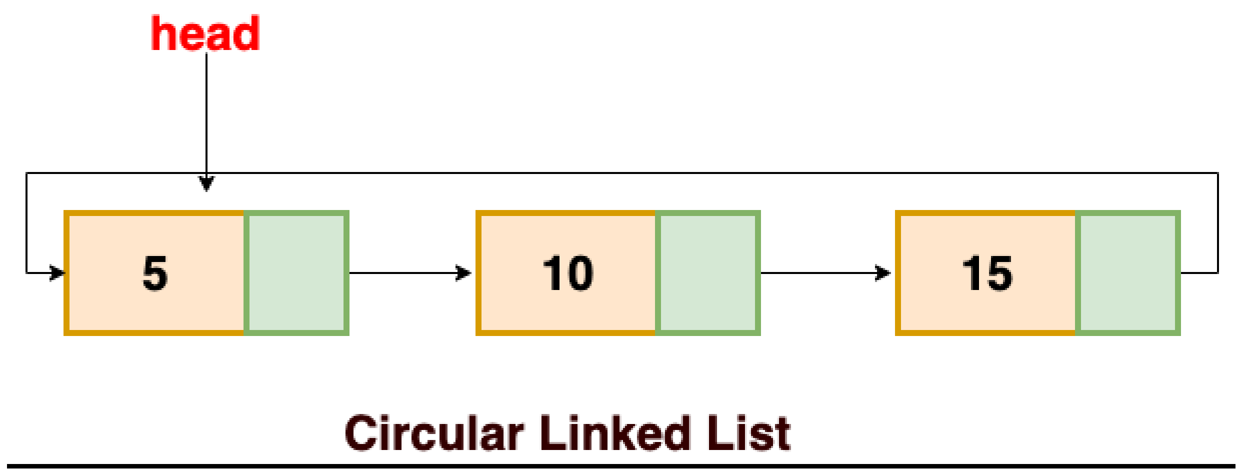
Circular Linked List

Circular Linked List: A circular linked list is either a singly or doubly linked list in which there are no **NULL** values. Here, we can implement the Circular Linked List by making the use of Singly or Doubly Linked List. In the case of a singly linked list, the next of the last node contains the address of the first node and in case of a doubly-linked list, the next of last node contains the address of the first node and prev of the first node contains the address of the last node.



Advantages of a Circular linked list

- Entire list can be traversed from any node.
- Circular lists are the required data structure when we want a list to be accessed in a circle or loop.
- Despite of being singly circular linked list we can easily traverse to its previous node, which is not possible in singly linked list.

Disadvantages of Circular linked list

- Circular list are complex as compared to singly linked lists.
- Reversing of circular list is a complex as compared to singly or doubly lists.
- If not traversed carefully, then we could end up in an infinite loop.

- Like singly and doubly lists circular linked lists also doesn’t supports direct accessing of elements.

Applications of Circular linked list

- Circular lists are used in applications where the entire list is accessed one-by-one in a loop. Example: Operating systems may use it to switch between various running applications in a circular loop.
- It is also used by Operating system to share time for different users, generally uses Round-Robin time sharing mechanism.
- Multiplayer games uses circular list to swap between players in a loop.

C program to create and traverse Circular Linked List

```
#include <stdio.h>

#include <stdlib.h>

struct node {

   int data;

   struct node * next;

}*head;

void createList(int n);

void displayList();

void main()

{

   int n, data, choice=1;
```

```
    head = NULL;

    while(choice != 0)

    {

        printf("CIRCULAR LINKED LIST PROGRAM\n");

        printf("1. Create List\n");

        printf("2. Display list\n");

        printf("0. Exit\n");

       printf("Enter your choice : ");

scanf("%d", &choice);


        switch(choice)

        {

            case 1:

                printf("Enter the total number of nodes in list: ");

                scanf("%d", &n);

                createList(n);

                break;

            case 2:

                displayList();

                break;

            case 0:
                break;

            default:

                printf("Error! Invalid choice. Please choose between 0-2");

        }



        printf("\n\n\n\n\n");

    }

getch();

}

void createList(int n)

{

    int i, data;

    struct node *prevNode, *newNode;



    if(n >= 1)

    {

        head = (struct node *)malloc(sizeof(struct node));



        printf("Enter data of 1 node: ");

        scanf("%d", &data);
```

```
   head->data = data;

   head->next = NULL;


   prevNode = head;


for(i=2; i<=n; i++)

   {

       newNode = (struct node *)malloc(sizeof(struct node));


       printf("Enter data of %d node: ", i);

       scanf("%d", &data);


       newNode->data = data;

       newNode->next = NULL;

       prevNode->next = newNode;

       prevNode = newNode;

   }


   prevNode->next = head;


   printf("\nCIRCULAR LINKED LIST CREATED SUCCESSFULLY\n");
```

```
    }

}



void displayList()

{

    struct node *current;

    int n = 1;



    if(head == NULL)

    {

        printf("List is empty.\n");

    }

    else

    {

        current = head;

        printf("DATA IN THE LIST:\n");



        do {

            printf("Data %d = %d\n", n, current->data);



            current = current->next;
            n++;

        }while(current != head);

    }

}
```

C program to insert a new node in circular linked list

```
#include <stdio.h>

#include <stdlib.h>

struct node {

    int data;

    struct node * next;

}*head;

void createList(int n);

void displayList();

void insertAtBeginning(int data);

void insertAtN(int data, int position);

void main()

{

    int n, data, choice=1;



    head = NULL;
```

```
while(choice != 0)
{
            printf("CIRCULAR LINKED LIST PROGRAM\n");
    printf("1. Create List\n");
    printf("2. Display list\n");
    printf("3. Insert at beginning\n");
    printf("4. Insert at any position\n");
    printf("0. Exit\n");
    printf("Enter your choice : ");

    scanf("%d", &choice);

    switch(choice)
    {
        case 1:
            printf("Enter the total number of nodes in list: ");
            scanf("%d", &n);
            createList(n);
            break;
        case 2:
            displayList();
            break;
        case 3:
            printf("Enter data to be inserted at beginning: ");
            scanf("%d", &data);
            insertAtBeginning(data);
            break;
        case 4:
            printf("Enter node position: ");
            scanf("%d", &n);
            printf("Enter data you want to insert at %d position: ", n);
            scanf("%d", &data);
            insertAtN(data, n);
            break;
        case 0:
            break;
        default:
            printf("Error! Invalid choice. Please choose between 0-4");
    }
```

```
        printf("\n\n\n\n\n");
    }


getch();
}
void createList(int n)
{
    int i, data;
    struct node *prevNode, *newNode;


    if(n >= 1)
    {
        head = (struct node *)malloc(sizeof(struct node));


        printf("Enter data of 1 node: ");
        scanf("%d", &data);


        head->data = data;
        head->next = NULL;


        prevNode = head;
        for(i=2; i<=n; i++)

        {

            newNode = (struct node *)malloc(sizeof(struct node));



            printf("Enter data of %d node: ", i);

            scanf("%d", &data);



            newNode->data = data;

            newNode->next = NULL;

            prevNode->next = newNode;



            prevNode = newNode;

        }

        prevNode->next = head;



        printf("\nCIRCULAR LINKED LIST CREATED SUCCESSFULLY\n");

    }

}
```

```
void displayList()
{
    struct node *current;
    int n = 1;

    if(head == NULL)
    {
        printf("List is empty.\n");
    }
    else
    {
        current = head;
        printf("DATA IN THE LIST:\n");

        do {
            printf("Data %d = %d\n", n, current->data);

            current = current->next;
            n++;
        }while(current != head);
    }
```

```
}


void insertAtBeginning(int data)

{

    struct node *newNode, *current;



    if(head == NULL)

    {

        printf("List is empty.\n");

    }

    else

    {

        newNode = (struct node *)malloc(sizeof(struct node));

        newNode->data = data;

        new current = head;

        while(current->next != head)

        {

            current = current->next;

        }

        current->next = newNode;
        head = newNode;


    printf("NODE INSERTED SUCCESSFULLY\n");
  }
}
void insertAtN(int data, int position)
{
  struct node *newNode, *current;
  int i;


  if(head == NULL)
  {
    printf("List is empty.\n");
  }
  else if(position == 1)
  {
    insertAtBeginning(data);
  }
  else
  {
    newNode = (struct node *)malloc(sizeof(struct node));

    newNode->data = data;

    current = head;

    for(i=2; i<=position-1; i++)

    {

        current = current->next;

    }

    newNode->next = current->next;

    current->next = newNode;



    printf("NODE INSERTED SUCCESSFULLY.\n");

  }

}
```