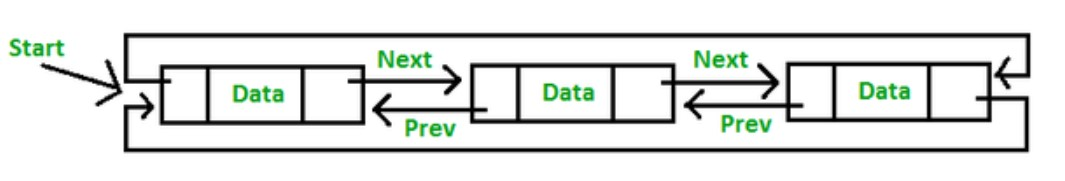
**Experiment no 7: Circular Linked List**

**Aim:** Implementation of Circular Linked List ADT

**Objective:** Circular Linked List can be used to manage the Computing Resources of the computer. Data Structure such as stacks and queue are implemented with the help of circular linked list

**Theory :** Circular Doubly linked list: Circular Doubly Linked List has properties of both doubly linked list and circular linked list in which two consecutive elements are linked or connected by the previous and next pointer and the last node points to the first node by the next pointer and also the first node points to the last node by the previous pointer.



**Algorithm :**

1. Insertion at the Beginning

store the address of the current first node in the newNode (i.e. pointing the newNode to the current first node)

point the last node to newNode (i.e making newNode as head)

2. If the node to be deleted is the only node

free the memory occupied by the node

store NULL in last

3. If last node is to be deleted

find the node before the last node (let it be temp)

store the address of the node next to the last node in temp

free the memory of last

make temp as the last node

**Code :**

#include <stdio.h>

#include <stdlib.h>

struct Node {

int data;

struct Node\* next;

};

void push(struct Node\*\* head\_ref, int data)

{

struct Node\* ptr1 = (struct Node\*)malloc(sizeof(struct Node));

ptr1->data = data;

ptr1->next = \*head\_ref;

if (\*head\_ref != NULL)

{

struct Node\* temp = \*head\_ref;

while (temp->next != \*head\_ref)

temp = temp->next;

temp->next = ptr1;

}

else

ptr1->next = ptr1;

\*head\_ref = ptr1;

}

void printList(struct Node\* head)

{

struct Node\* temp = head;

if (head != NULL)

{

do {

printf("%d ", temp->data);

temp = temp->next;

} while (temp != head);

}

printf("\n");

}

void deleteNode(struct Node\*\* head, int key)

{

if (\*head == NULL)

return;

if ((\*head)->data == key && (\*head)->next == \*head) {

free(\*head);

\*head = NULL;

return;

}

struct Node \*last = \*head, \*d;

if ((\*head)->data == key) {

while (last->next != \*head)

last = last->next;

last->next = (\*head)->next;

free(\*head);

\*head = last->next;

return;

}

while (last->next != \*head && last->next->data != key) {

last = last->next;

}

if (last->next->data == key) {

d = last->next;

last->next = d->next;

free(d);

}

else

printf("Given node is not found in the list!!!\n");

}

int main()

{

struct Node\* head = NULL;

push(&head, 2);

push(&head, 5);

push(&head, 7);

push(&head, 8);

push(&head, 10);

printf("List Before Deletion: ");

printList(head);

deleteNode(&head, 7);

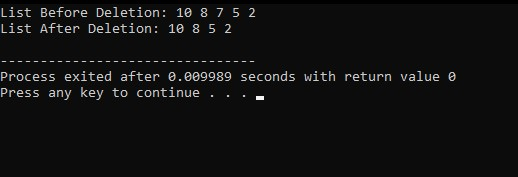
printf("List After Deletion: ");

printList(head);

return 0;

}

**Output :**

****

**Conclusion :**

* It is used in multiplayer games to give a chance to each player to play the game.
* Multiple running applications can be placed in a circular linked list on an operating system. The os keeps on iterating over these applications.