Circular Linked List

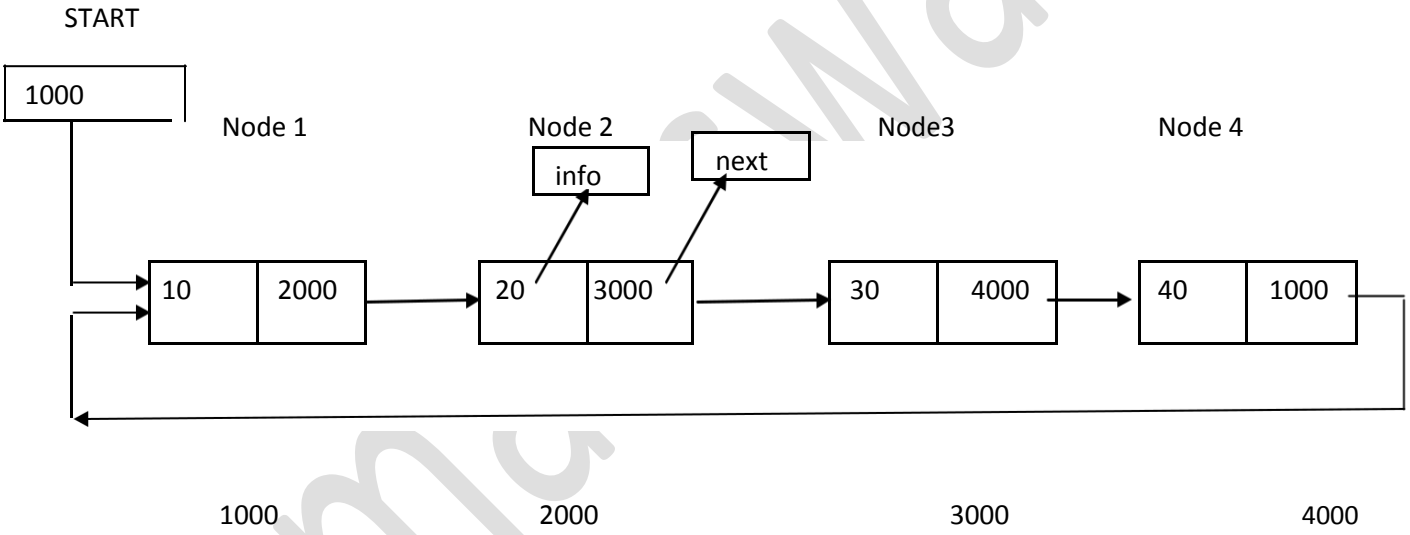
What is Circular Linked List?

In single linked list, every node points to its next node in the sequence and the last node points NULL. But in circular linked list, every node points to its next node in the sequence but the last node points to the first node in the list.

A circular linked list is a sequence of elements in which every element has a link to its next element in the sequence and the last element has a link to the first element.

***That means circular linked list is similar to the single linked list except that the last node points to the first node in the list.**

Example:



Operations

In a circular linked list, we perform the following operations...

- Insertion
- Deletion
- Display

Before we implement actual operations, first we need to setup empty list. First perform the following steps before implementing actual operations.

Step 1 - Include all the starter files which are used in the program.

Step 2 - Declare all the user defined functions.

Step 3 - Define a Node structure with two members info and next

Step 4 - Define a Node pointer 'start' and set it to NULL.

Step 5 - Implement the main method by displaying operations menu and make suitable function calls in the main method to perform user selected operation.

Insertion

In a circular linked list, the insertion operation can be performed in four ways. They are as follows...

- Inserting At Beginning of the list
- Inserting At End of the list
- Inserting At Specific location in the list
- Inserting After a Given Node

Inserting At Beginning of the list:-

The following steps can be followed to insert a new node at the beginning of the circular linked list.

Step 1 - Create a newNode with given value.

Step 2 - Check whether list is Empty (start == NULL)

Step 3 - If it is Empty then, set start = newNode and newNode→next = start.

Step 4 - If it is Not Empty then, define a Node pointer 'temp' and initialize with 'start'.

Step 5 - Keep moving the 'temp' to its next node until it reaches to the last node (until 'temp → next == start').

Step 6 - Set 'newNode → next =start', 'start = newNode' and 'temp → next = start'.

```
void insert_at_beginning()
{
        struct node *temp,*ptr;
        int item;
        temp=(struct node *)malloc(sizeof(struct node));
        if(temp == NULL)
        {
                printf("\nMemory is not allocated\n");
        }
```

```
        else
        {
                printf("\nEnter value to insert\n");
                scanf("%d",&item);
                temp->info=item;
                temp->next=NULL;
                if(start == NULL)
                {
                        start = temp;
                        start->next=start;
                }
                else
                {
                        ptr=start;
                        while(ptr->next != start)
                        {
                                ptr=ptr->next;
                        }
                        temp->next = start;
                        ptr->next=temp;
                        start = temp;
                }
        }
}
```


Inserting At End of the list:-

We can use the following steps to insert a new node at end of the circular linked list.

Step 1 - Create a newNode with given value.

Step 2 - Check whether list is Empty (start == NULL).

Step 3 - If it is Empty then, set start = newNode and newNode → next = start.

Step 4 - If it is Not Empty then, define a node pointer temp and initialize with start.

Step 5 - Keep moving the temp to its next node until it reaches to the last node in the list (until temp → next == start).

Step 6 - Set temp → next = newNode and newNode → next = start.

void insert_at_end()

```
{
        struct node *temp,*ptr;
        int item;
        temp=(struct node *)malloc(sizeof(struct node));
        if(temp == NULL)
        {
                printf("\nMemory is not allocated\n");
        }
        else
        {
                printf("\nEnter value to insert\n");
                scanf("%d",&item);
                temp->info=item;
                temp->next=NULL;
                if(start == NULL)
                {
                        start = temp;
                        start->next=start;
                }
```

```
                else
                {
                        ptr=start;
                        while(ptr->next != start)
                        {
                                ptr=ptr->next;
                        }
                        temp->next = start;
                        ptr->next=temp;
                }
        }
}
```

Inserting After a Given Node

We can use the following steps to insert a new node after a node in the circular linked list.

Step 1 - Create a newNode with given value.

Step 2 - Check whether list is Empty (start == NULL)

Step 3 - If it is Empty then, set start = newNode and newNode → next = start.

Step 4 - If it is Not Empty then, define a node pointer temp and initialize with start.

Step 5 - Keep moving the temp to its next node until it reaches to the node after which we want to insert the newNode (until temp1 → info is equal to location, here location is the node value after which we want to insert the newNode).

Step 6 - Every time check whether temp is reached to the last node or not. If it is reached to last node then display 'Given node is not found in the list!!! Insertion not possible!!!' and terminate the function. Otherwise move the temp to next node.

Step 7 - If temp is reached to the exact node after which we want to insert the newNode then check whether it is last node (temp → next == start).

Step 8 - If temp is last node then set temp → next = newNode and newNode → next = start.

Step 9 - If temp is not last node then set newNode → next = temp → next and temp → next = newNode.

```
void insert_afternode()
{
        struct node *temp,*ptr;
        int item,value;
        temp=(struct node *)malloc(sizeof(struct node));
        if(temp == NULL)
        {
                printf("\nMemory is not allocated\n");
        }
        else
        {
                printf("\nEnter value to insert\n");
                scanf("%d",&item);
                temp->info=item;
                temp->next=NULL;
                if(start == NULL)
                {
                        start = temp;
                        start->next=start;
                }
                else
                {
                        ptr=start;
                        printf("\nEnter the value after which insertion is required\n");
                        scanf("%d",&value);
                        do
                        {
                                        ptr=ptr->next;
```

```
                    }
                    while(ptr!=start && ptr->info!=value);
                    if(ptr == start)
                    {
                            printf("\nNode is not present\n");
                    }
                    else
                    {
                            temp->next=ptr->next;
                            ptr->next=temp;
                    }
            }
        }
}
```

Home work: - Inserting At a specific location

Hint:- The logic for algorithm and program will be same as insert after a given node. Refer, understand and then implement.

Deletion from Circular Linked List

In a circular linked list, the deletion operation can be performed in four ways those are as follows.

- Deleting from Beginning of the list
- Deleting from End of the list
- Deleting a node after a given node
- Deleting a node from specific location

Deleting from Beginning of the list

We can use the following steps to delete a node from beginning of the circular linked list...

Step 1 - Check whether list is Empty (start == NULL)

Step 2 - If it is Empty then, display 'List is Empty!!! Deletion is not possible' and terminate the function.

Step 3 - If it is Not Empty then, define two Node pointers 'temp1' and 'temp2' and initialize both 'temp1' and 'temp2' with start.

Step 4 - Check whether list is having only one node (temp1 → next == start)

Step 5 - If it is TRUE then set start = NULL and delete temp1 (Setting Empty list conditions)

Step 6 - If it is FALSE move the temp1 until it reaches to the last node. (until temp1 → next == start)

Step 7 - Then set start = temp2 → next, temp1 → next = start and delete temp2.

void delete_from_beginning()

```
{
        struct node *temp,*ptr;
        if(start == NULL)
        {
                printf("\nlist is empty\n");
        }
        else
        {
                temp=start;
                ptr=start;
                printf("\nDeleted node is %d",temp->info);
                while(ptr->next != start)
                {
                        ptr=ptr->next;
                }
                start=start->next;
                ptr->next=start;
                free(temp);
        }
}
```

Deleting from End of the list

We can use the following steps to delete a node from end of the circular linked list...

Step 1 - Check whether list is Empty (start == NULL)

Step 2 - If it is Empty then, display 'List is Empty!!! Deletion is not possible' and terminate the function.

Step 3 - If it is Not Empty then, defines two Node pointers 'temp1' and 'temp2' and initialize 'temp1' with start.

Step 4 - Check whether list has only one Node (temp1 → next == start)

Step 5 - If it is TRUE. Then, set start = NULL and delete temp1. And terminate from the function. (Setting Empty list condition)

Step 6 - If it is FALSE. Then, set 'temp2 = temp1 ' and move temp1 to its next node. Repeat the same until temp1 reaches to the last node in the list. (until temp1 → next == start)

Step 7 - Set temp2 → next = start and delete temp1.

```
void delete_from_end()
{
        struct node *temp,*ptr;
        if(start == NULL)
        {
                printf("\nlist is empty\n");
        }
        else
        {
                temp=start;
                while(temp->next != start)
                {
                        ptr=temp;
                        temp=temp->next;
                }
                ptr->next=start;
```

```
            printf("\nDeleted node is %d\n",temp->info);
            free(temp);
        }
}
```

Deleting from a Specific location

We can use the following steps to delete a specific node from the circular linked list...

Step 1 - Check whether list is Empty (start == NULL)

Step 2 - If it is Empty then, display 'List is Empty!!! Deletion is not possible' and terminate the function.

Step 3 - If it is Not Empty then, defines two Node pointers 'temp1' and 'temp2' and initialize 'temp1' with start.

Step 4 - Keep moving the temp1 until it reaches to the exact node to be deleted or to the last node. And every time set 'temp2 = temp1' before moving the 'temp1' to its next node.

Step 5 - If it is reached to the last node then display 'Given node not found in the list! Deletion not possible!!!'. And terminate the function.

Step 6 - If it is reached to the exact node which we want to delete, then check whether list is having only one node (temp1 → next == start)

Step 7 - If list has only one node and that is the node to be deleted then set start = NULL and delete temp1 (free(temp1)).

Step 8 - If list contains multiple nodes then check whether temp1 is the first node in the list (temp1 == start).

Step 9 - If temp1 is the first node then set temp2 = start and keep moving temp2 to its next node until temp2 reaches to the last node. Then set start = start → next, temp2 → next = start and delete temp1.

Step 10 - If temp1 is not first node then check whether it is last node in the list (temp1 → next == start).

Step 11- If temp1 is last node then set temp2 → next = start and delete temp1 (free(temp1)).

Step 12 - If temp1 is not first node and not last node then set temp2 → next = temp1 → next and delete temp1 (free(temp1)).

```
void delete_fromlocation()
{
        struct node *temp,*ptr;
```

```
int loc,count=1;
if(start == NULL)
{
        printf("\nlist is empty\n");
}
else
{
        ptr=start;
        temp=start;
        printf("\nEnter the location from deletion is required\n");
        scanf("%d",&loc);
        do
        {
                ptr=temp;
                temp=temp->next;
                count++;
        }
        while(temp!=start && count!=loc);
        if(temp == start)
        {
                printf("\nNode is not present\n");
        }
        else
        {
                ptr->next=temp->next;
                printf("\nDeleted node is %d\n",temp->info);
                free(temp);
        }
```

```
        }
}
```

Assignment: **Deleting after a given node**

Hint: The logic for algorithm and program will be same as delete from specific location. Refer, understand and then implement.

Displaying content of a circular Linked List

We can use the following steps to display the elements of a circular linked list...

Step 1 - Check whether list is Empty (start == NULL)

Step 2 - If it is Empty, then display 'List is Empty!!!' and terminate the function.

Step 3 - If it is Not Empty then, define a Node pointer 'temp' and initialize with start.

Step 4 - Keep displaying temp → info with an arrow (--->) until temp reaches to the last node

Step 5 - Finally display temp → info with arrow pointing to start → info.

```
void traverse()
{
        struct node *temp;
        if(start == NULL)
        {
                printf("\nList is empty\n");
        }
        else
        {
                temp=start;
                printf("\nValues of LIst are\n");
                do
                {
                        printf("%d\t",temp->info);
                        temp=temp->next;
```

```
            }
            while(temp!=start);
        }
}
```

Data type or Structure of a node in the circular linked list defined as :

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

Assignment: **Write a menu driven program to implement double circular linked list. Mark - 5**

Hint:- It is a combination of double linked list and circular linked list where last node next will contain address of starting node and previous of starting node will contain the address of last node.

"Any Fool Can Write Code that a Computer Can Understand. Good Programmers Write Code that a Human Can Understand".

All The Best

Your's Sam Sir