DATA STRUCTURES
USING "C"

(Circular Linked List)

For

BCA Part-II (Session 2018-21) Students

BY

ANANT KUMAR
MCA, M. Phil, M. Tech.
Faculty Member
Department of Computer Science
J. D. Women's College, Patna

Circular Linked List

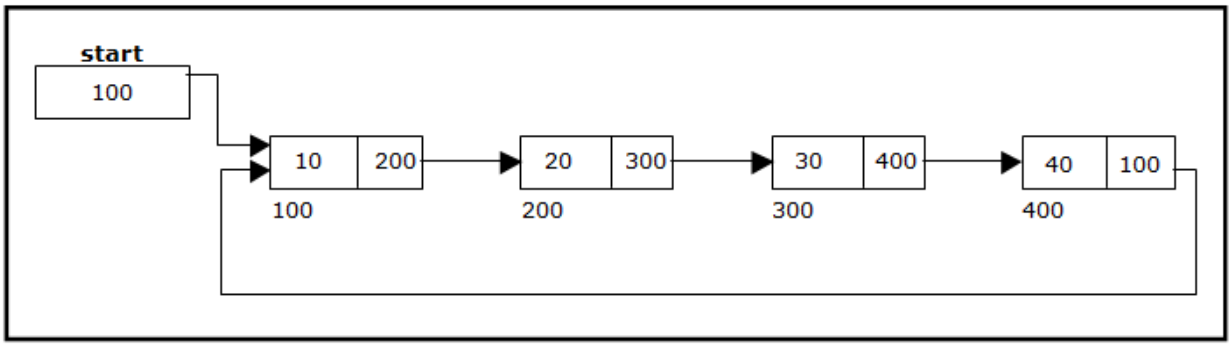
A circular linked list has no beginning and no end. It is necessary to establish a special pointer called *start* pointer always pointing to the first node of the list. Circular linked lists are frequently used instead of ordinary linked list because many operations are much easier to implement. In circular linked list no null pointers are used, all pointers contain valid address.
There are two types of circular linked list:

i. Singly Circular
ii. Doubly Circular

Singly Circular

The basic operations in a singly circular linked list are:

- Creation.
- Insertion.
- Deletion.
- Traversing.



Program:

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

typedef struct cslinklist
{
        int data;
        struct cslinklist *next;
}node;

node *start = NULL;
int nodectr;

node* getnode();
void createlist(int);
void display();
void cll_insert_beg();
void cll_insert_end();
void cll_insert_mid();
void cll_delete_beg();
void cll_delete_last();
void cll_delete_mid();

voidmain(void)
{
    int result, ch, n;
    while(1)
    {
        clrscr();
        printf("\n 1. Create a list ");
        printf("\n\n   ");
        printf("\n 2. Insert a node at beginning ");
        printf("\n 3. Insert a node at end");
        printf("\n 4. Insert a node at middle");
        printf("\n\n   ");
        printf("\n 5. Delete a node from beginning");
        printf("\n 6. Delete a node from Last");
        printf("\n 7. Delete a node from Middle");
```

```
printf("\n\n   ");
printf("\n 8. Display the list");
printf("\n 9. Exit");
printf("\n\n   ");
printf("\n Enter your choice: ");
scanf("%d", &ch);
switch(ch)
{
        case 1 :
                if(start == NULL)
                {
                        printf("\n Enter Number of nodes to create: ");
                        scanf("%d", &n);
                        createlist(n);
                        printf("\nList created..");
                }
                else
                        printf("\n List is already Exist..");
                        break;
                case 2 :
                        cll_insert_beg();
                        break;
                case 3 :
                        cll_insert_end();
                        break;
                case 4 :
                        cll_insert_mid();
                        break;
                case 5 :
                        cll_delete_beg();
                        break;
                case 6 :
                        cll_delete_last();
                        break;
                case 7 :
                        cll_delete_mid();
                        break;
                case 8 :
                        display();
                        break;
```

```
                        case 9 :
                                exit(0);
        }
        getch();
    }
}

node* getnode()
{
        node * newnode;
        newnode = (node *) malloc(sizeof(node));
        printf("\n Enter data: ");
        scanf("%d", &newnode -> data);
        newnode -> next = NULL;
        return newnode;
}

void createlist(int n)
{
        int i;
        node *newnode; node *temp; nodectr = n;
        for(i = 0; i < n ; i++)
        {
                newnode = getnode();
                if(start == NULL)
                        start = newnode;
                else
                {
                        temp = start;
                        while(temp -> next != NULL)
                                temp = temp -> next;
                        temp -> next = newnode;
                }

                newnode ->next=start;
        }
}

void display()
{
```

```
        node *temp;
        temp = start;
        printf("\n The contents of List (Left to Right): ");
        if(start == NULL )
                printf("\n Empty List");

        else
        {
                do
                {
                        printf("\t %d ", temp -> data);
                        temp = temp -> next;
                } while(temp != start);
        }
}

void cll_insert_beg()
{
        node *newnode, *last;
        newnode = getnode();
        if(start == NULL)
        {
                start = newnode;
                newnode -> next = start;
        }
        else
        {
                last = start;
                while(last -> next != start)
                        last = last -> next;
                newnode -> next = start;
                start = newnode;
                last -> next = start;
        }
        printf("\n Node inserted at beginning..");
        nodectr++;
}

void cll_insert_end()
{
```

```
        node *newnode, *temp;
        newnode = getnode();
        if(start == NULL )
        {
                start = newnode;
                newnode -> next = start;
        }
        else
        {
                temp = start;
                while(temp -> next != start)
                        temp = temp -> next;
                temp -> next = newnode;
                newnode -> next = start;
        }

        printf("\n Node inserted at end..");
        nodectr++;
}

void cll_insert_mid()
{
        node *newnode, *temp, *prev; int i, pos ;
        newnode = getnode();
        printf("\n Enter the position: ");
        scanf("%d", &pos);
        if(pos > 1 && pos < nodectr)
        {
                temp = start;
                prev = temp;
                i = 1;
                while(i < pos)
                {
                        prev = temp;
                        temp = temp -> next;
                        i++;
                }
                prev -> next = newnode;
                newnode -> next = temp;
                nodectr++;
```

```
                printf("\n Node inserted at middle..");
        }
        else
                printf("position %d of list is not a middle position ", pos);
}

void cll_delete_beg()
{
        node *temp, *last;
        if(start == NULL)
        {
                printf("\n No nodesexist..");
                getch();
                return;
        }
        else
        {
                last = temp = start;
                while(last -> next != start)
                        last = last -> next;
                start = start -> next;
                last -> next = start;
                free(temp);
                nodectr--;
                printf("\n Node deleted..");
                if(nodectr == 0)
                        start = NULL;
        }
}

void cll_delete_last()
{
        node *temp,*prev;
        if(start == NULL)
        {
                printf("\n No nodesexist..");
                getch();
                return;
        }
        else
```

```
	{
		temp = start;
		prev = start;
		while(temp -> next != start)
		{
			prev = temp;
			temp = temp -> next;
		}
		prev -> next = start;
		free(temp);
		nodectr--;
		if(nodectr == 0)
			start = NULL;
		printf("\n Node deleted..");
	}
}

void cll_delete_mid()
{
	int i = 0, pos;
	node *temp, *prev;
	if(start == NULL)
	{
		printf("\n No nodesexist..");
		getch();
		return;
	}
	else
	{
		printf("\n Which node to delete: ");
		scanf("%d", &pos);
		if(pos > nodectr)
		{
			printf("\nThis node does not exist"); getch();
			return;
		}
		if(pos > 1 && pos < nodectr)
		{
			temp=start;
			prev = start;
```

```
                i = 0;
                while(i < pos - 1)
                {
                        prev = temp;
                        temp = temp -> next ; i++;
                }
                prev -> next = temp -> next;
                free(temp);
                nodectr--;
                printf("\n Node Deleted..");
        }
        else
        {
                printf("\n It is not a middle position..");
                getch();
        }
    }
}
```