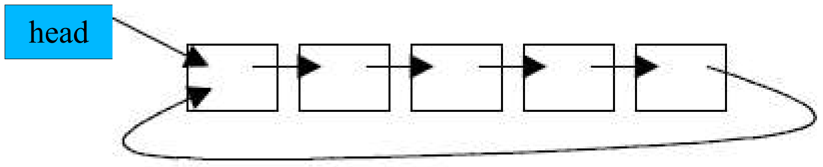
Lecture 7

Circular Linked List

A linked list in which last node's ptr contains the address of first node is called as circular linked list.



It is very easy to build a circular linked list and performing operations like insertion and deletion.We will start with building a circular linked list.

Build a Circular Linked List

Step1: Create a node, call this node as head. Store null in head.

```
node *head = null;
```

Step2: Create another node, call this node as temp; Allocate memory to temp.

```
node *temp =  (void *)malloc(sizeOf(node));
```

Step3: Ask the user to enter value in temp.

```
printf("Enter value");
scanf("%d",temp->value);
```

Step4: Make head point to temp.

```
head = temp;
```

Step5: Since this is the first node, it's link should point to itself.

```
temp->link = head;
```

Step6: Start a loop, which repeats itself as long as ans == 'y'

```
while(ans == 'y') {
```

Step7: Create another node, call this node as temp1. Allocate memory for temp1. Ask the user to enter value.

```
temp1 = (node *)malloc(sizeOf(node));
```

```
                printf("Enter value");
                scanf("%d", temp1->data);
```

Step8: We want tomake temp1 as the last node. To make temp1 as last node, current last node (which i s temp) link should contain the address of temp1.

```
                temp->link = temp1;
```

Step9: temp1 is the second node and also last node. That means, temp1's link should store the address of temp.

```
                temp1->link = head;
```

Step10: Make temp point to temp1.

```
                temp = temp1;
```

Step11: Ask the user, if he wish to add more node.

```
                printf("Do you wish to add more nodes");
                scanf("%c", &ans);
            }
```

Here is the complete code for Building a circular linked list.

```
void main
{
        Structure node
        {
                int data;
                node *link;
        };

        node *head;
        node *temp;
        node *temp1;
// ans is a character variable with default value 'y'
        char ans = 'y';

        head = null;
// create and allocate memory to 1st node
        temp = (node *) malloc (sizeOf(node));

// ask the user to enter value
        printf("Enter value");
        scanf("%d", temp->value);
// // Since temp is the first node, head should point to the first
// node. i.e. head contains the address of first node.
                head=temp;

//temp is the first and last node, last node link part
// always store the address of first node
        temp->next = head;

// Start a while loop, which will iterate as long as ans == 'y'
        while (ans == 'y') {
// create and allocate memory for temp1
        temp1 = (node *) malloc(sizeOf(node));

// ask the user to enter value
        printf("Enter value");
        scanf("%d", temp1->value);

// We want to make temp1 as the last node. i.e., temp's link
//(which is currenly last node) should contain the address of temp1.
        temp->link = temp1;

// now temp1 is the last node, so temp's link should contain store
// the address of first node.
        temp1->link = head;

//Move temp to last node i.e temp1
        temp = temp1;

// Ask the user whether he want to add more nodes
        printf("Do you wish to add more nodes");
        scanf("%c", &ans);
}
```

Printing a Circular Linked List

We have a circular linked list, and head points to the first node, we want to print this linked list.

Step1: Make temp point to the first node.

```
                temp = head;
```

Step2: Start a do while loop, which iterates as long as temp != head

```
do {
```

Step3: print the data at temp

```
printf(“%d”, temp->data);
```

Step4: Move temp to next node.

```
temp = temp->next;
```

Step5: Check the condition of while loop

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

Below is a program for printing a ciruclar linked list

```
//Make temp point to the first node.
        temp = head

// Start a do-while loop
        do
        {
// Print the data
                printf(“%d”,temp->data);
// Move temp to the next node
                temp = temp->next;
        }while (temp!=head);
```