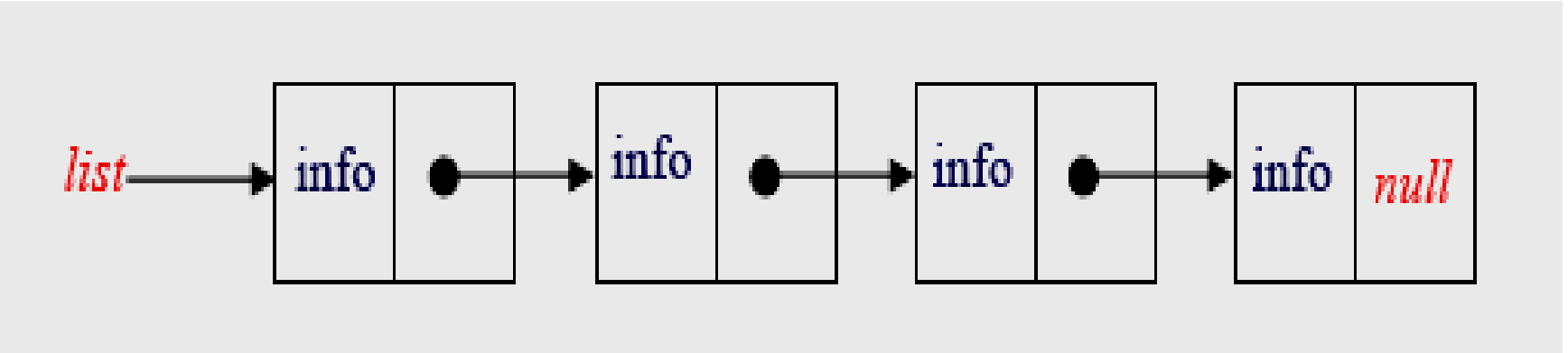
Circular Linked List

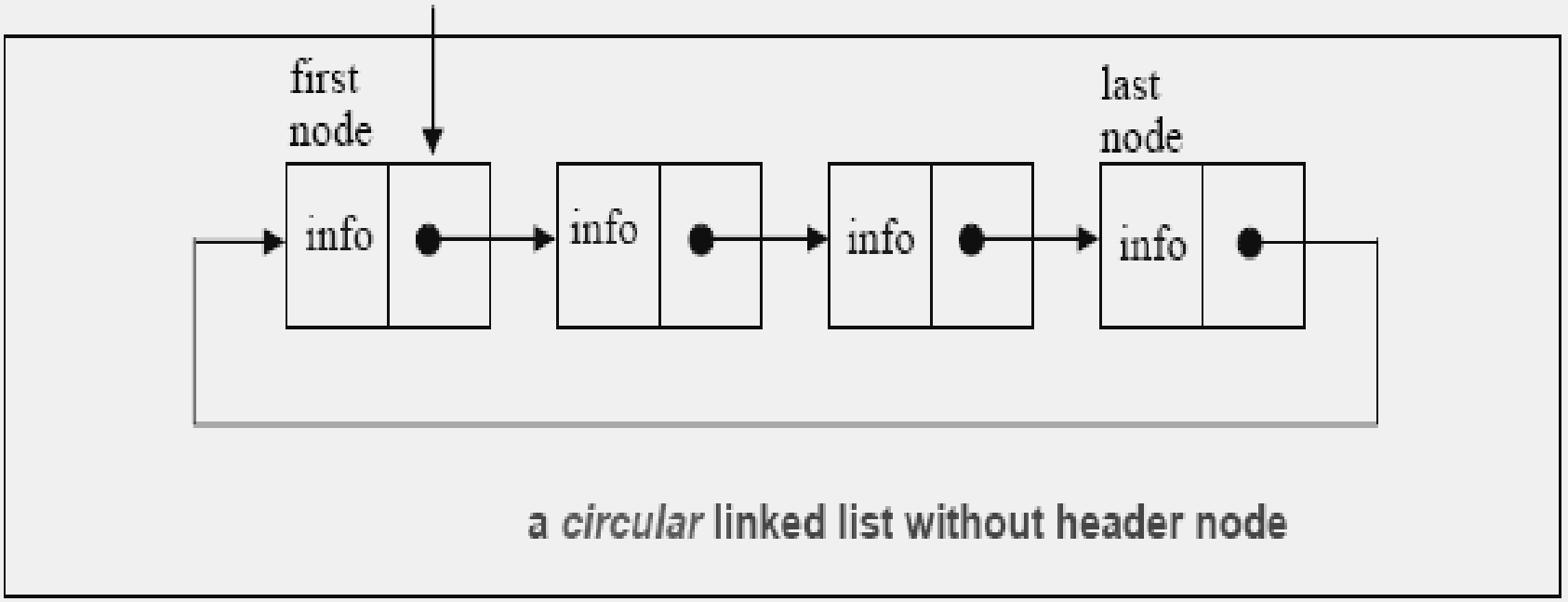
Circular linked lists can be used to help the traverse the same list again and again if needed.

A circular list is very similar to the linear list where in the circular list the pointer of the last node points not NULL but the first node.

Single Linked List



Circular Single Linked List



Circular Linked List

In a circular linked list there are two methods to find the first node .

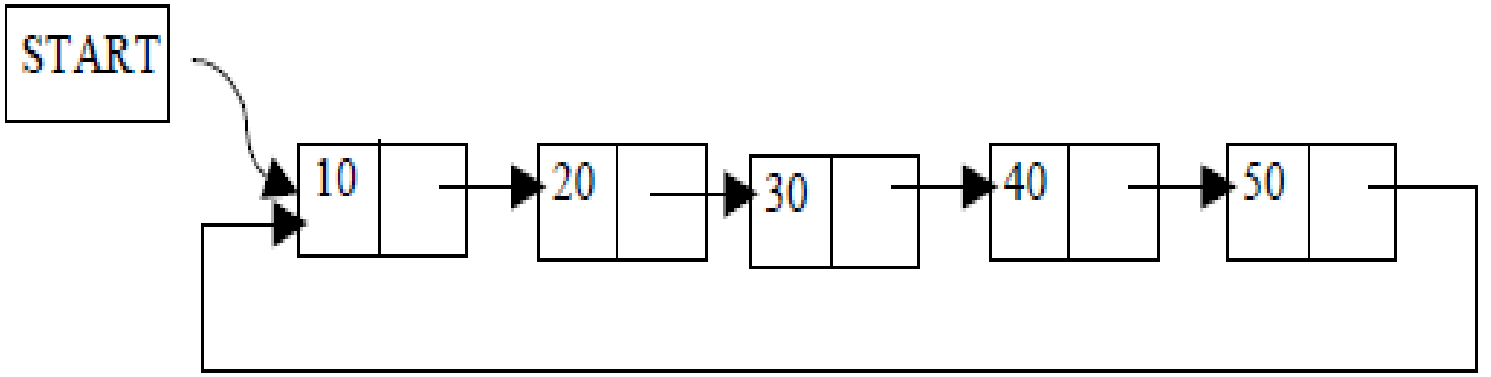
Last node address part points the first node or

head pointer *contains* first node address like in Single link list.

No node in a circular linked list contains *NULL*

Circular Single Linked List

A circular linked list is one, which has no beginning and no end.



Circular Linked list

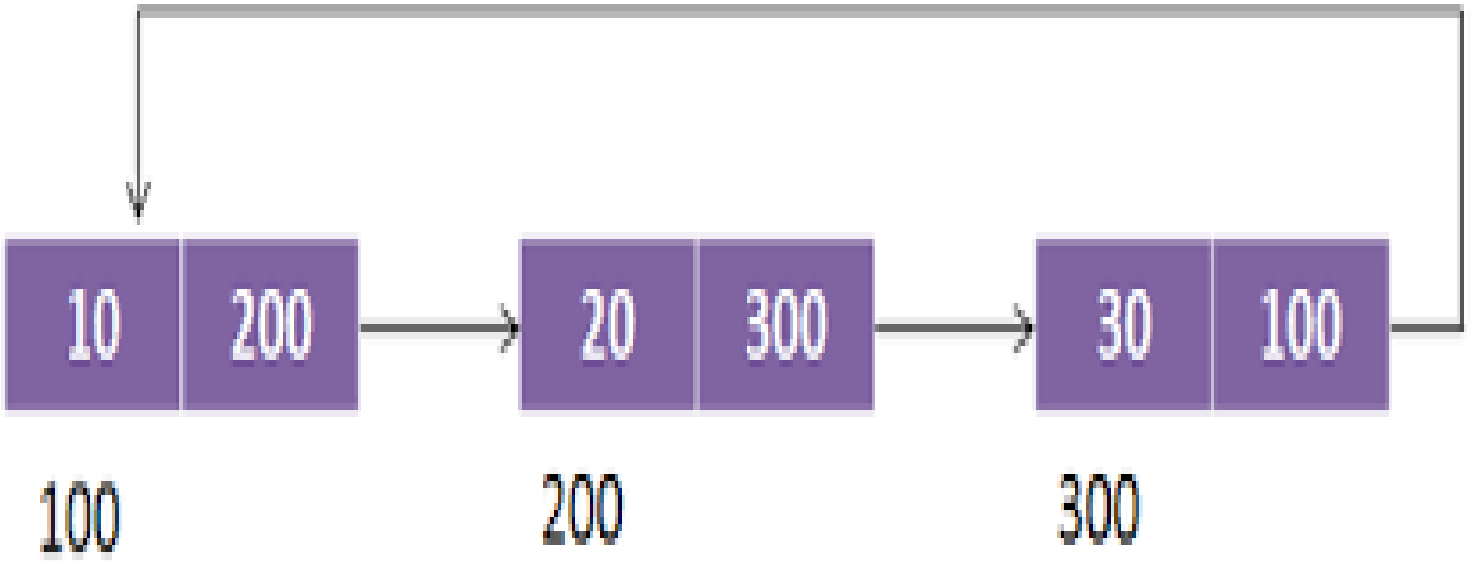
```
typedef struct node
{
int data;
struct node *link;

}NODE;
```

- NODE *head =NULL;

hptr

100



Adding the node at begin in circular Single Linklist

- void add_begin()

  {

  NODE *temp,*temp1;

  temp1=head;

  temp=(NODE *)malloc(sizeof(NODE));

  printf("Enter data\n");

  scanf("%d",&temp->data);

… conti

```
if(head==NULL) // empty Linked list
{
head=temp;  // adding the first node
temp->link=temp; // creating link to itself (circular)
}
```

… conti

```
else
{
while(temp1->link!=head)
temp1=temp1->link;  // find the last node
temp->link=head;
temp1->link=temp;
head=temp;        // update head pointer
}
}
```

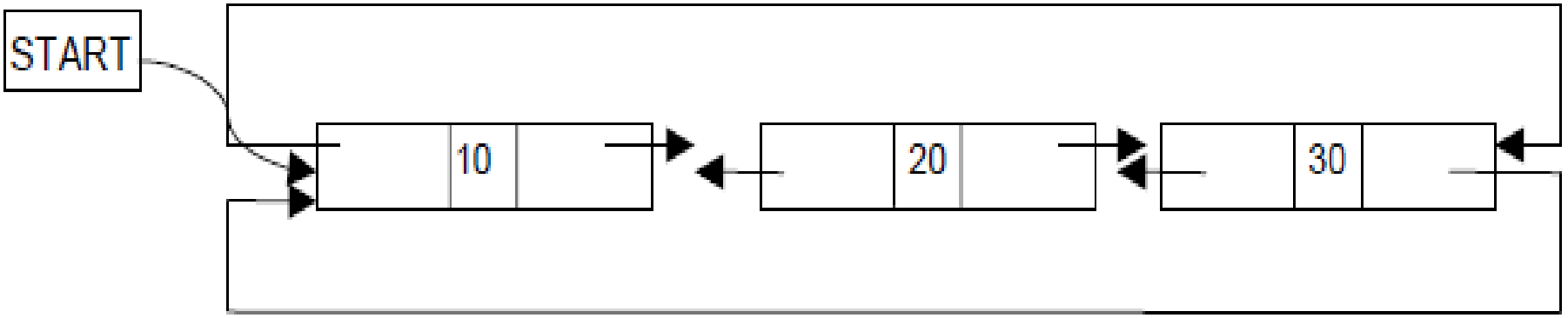
Adding the node at end in circular Single Linklist

```
else
{
while(temp1->link!=head)
temp1=temp1->link;  // find the last node
temp->link =temp1->link;
 temp1->link=temp;
}
```

Printing the nodes of a circular Single linklist

```
void print()
{
NODE *temp;
temp=head;
if(temp==NULL)
printf("\n List is empty\n");
else
{
        while(temp->link!=head) //  until last node
         {
          printf("%d \n",temp->data);
          temp=temp->link;
       }
         printf("%d",temp->data);   // to print the last node data
}
```

Circular Double Linked List



Circular Doubly Linked list

Adding the node at begin in circular Double Linklist

```
if(head==NULL)
{
temp->next=temp->prev=temp;
}
else
{
temp1=head->prev;
temp->next=head;
temp->prev=temp1;
temp1->next=head->prev=temp;
head=temp;
```

- }
-