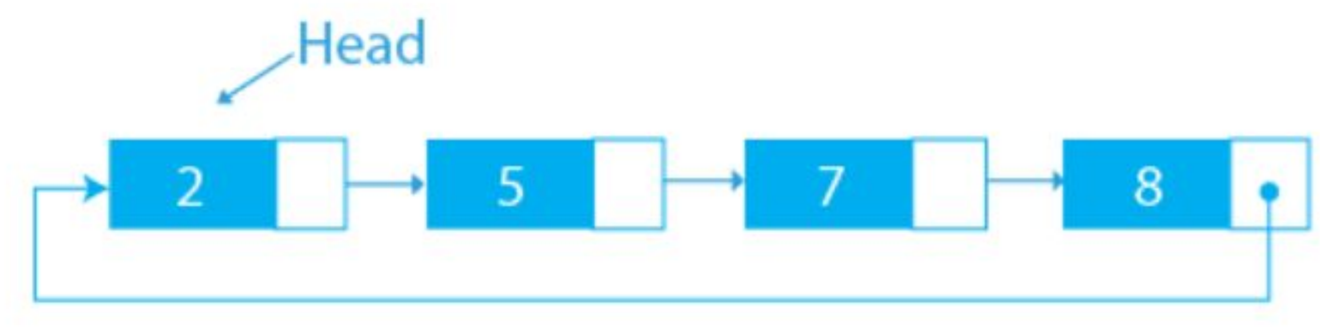
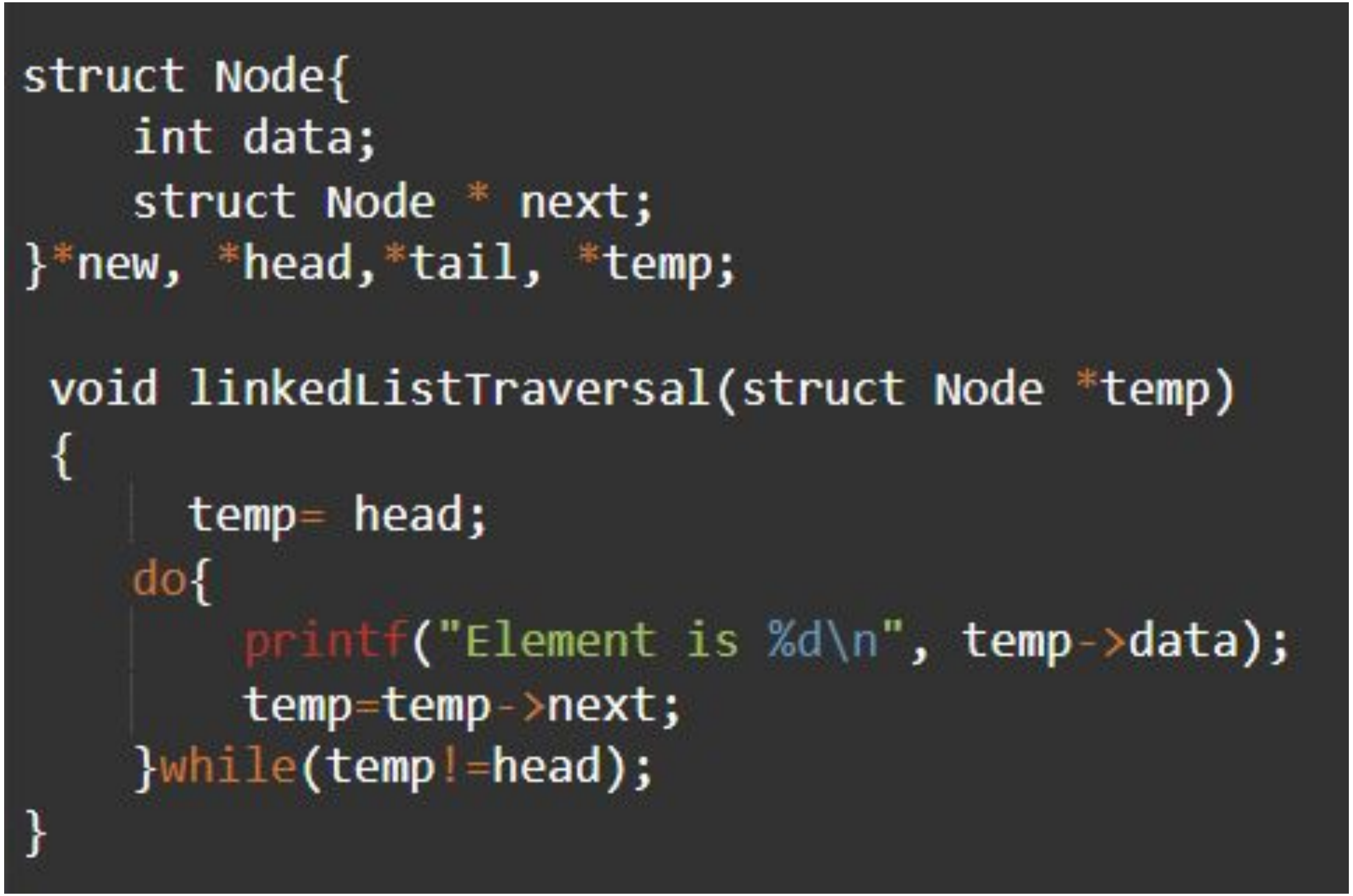
CLL

CLL



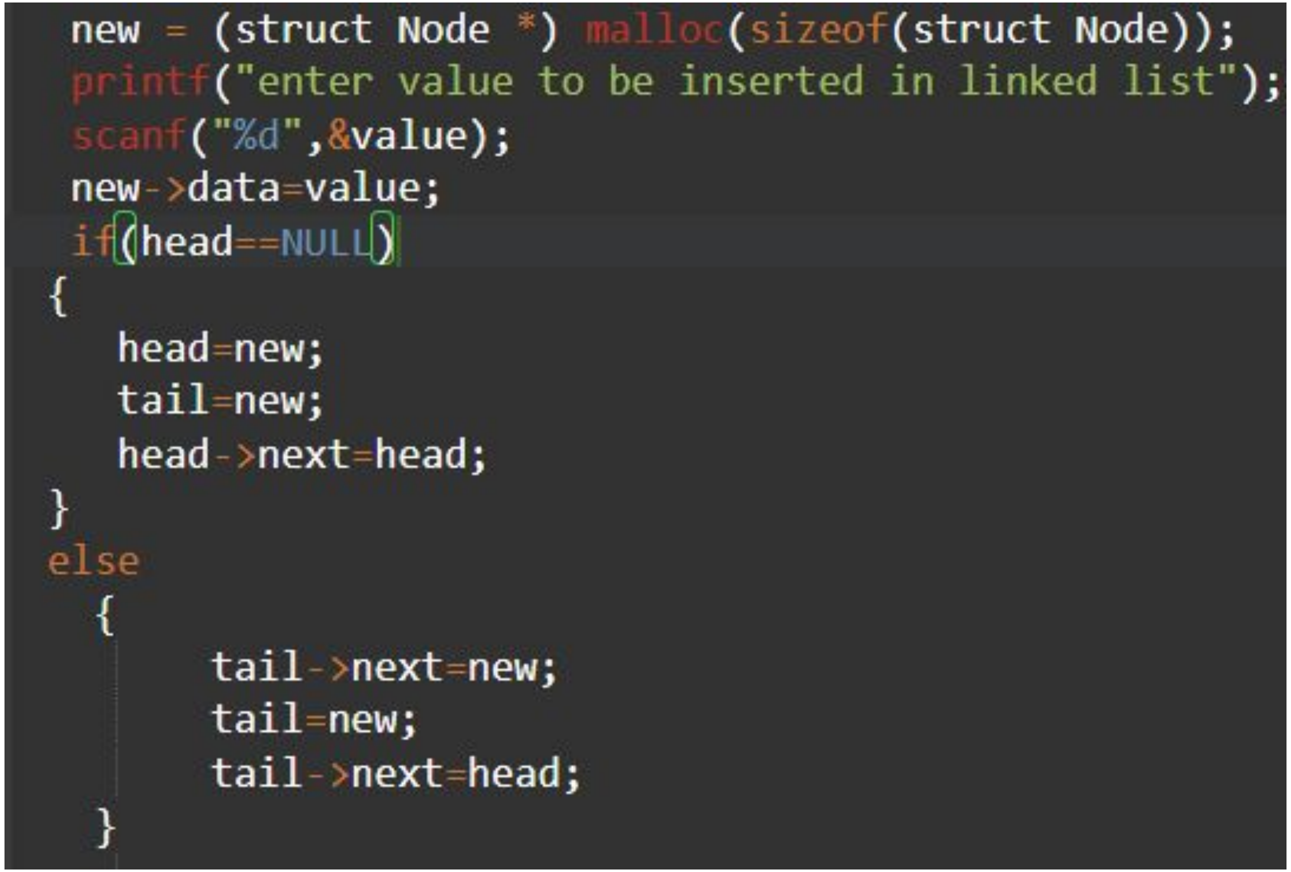
There is no NULL at the end. The last node, instead of pointing to NULL, points to the first node.
A singly linked list or a doubly linked list can be converted to a circular linked list.



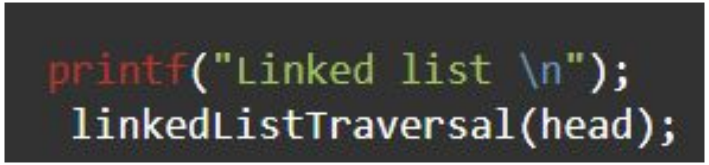
```c
struct Node{
    int data;
    struct Node * next;
}*new, *head,*tail, *temp;

 void linkedListTraversal(struct Node *temp)
 {
      temp= head;
    do{
        printf("Element is %d\n", temp->data);
        temp=temp->next;
    }while(temp!=head);
}
```

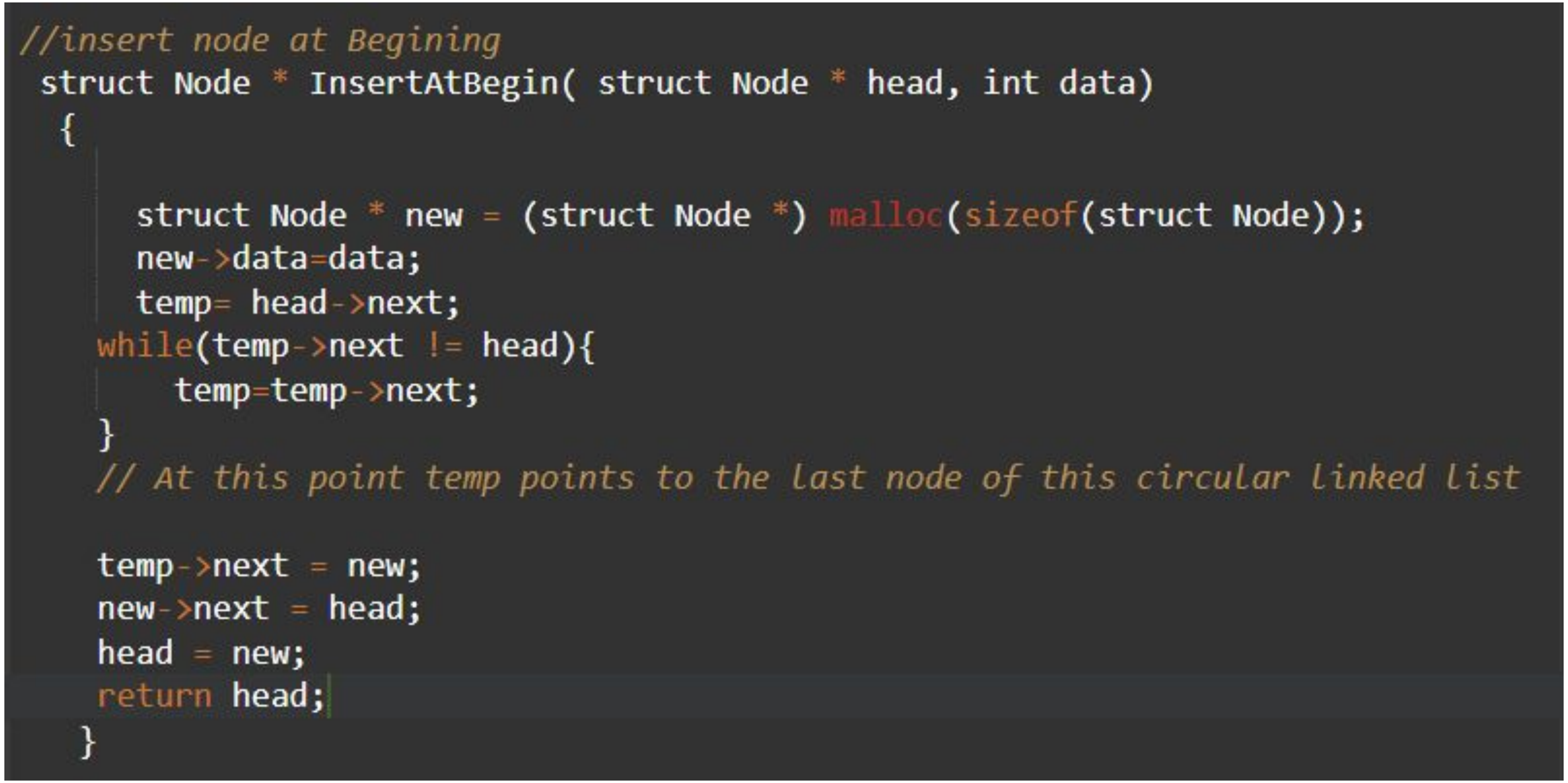
CLL creation



```
  new = (struct Node *) malloc(sizeof(struct Node));
  printf("enter value to be inserted in linked list");
  scanf("%d",&value);
  new->data=value;
  if(head==NULL)
 {
    head=new;
    tail=new;
    head->next=head;
 }
 else
   {
        tail->next=new;
        tail=new;
        tail->next=head;
   }
```



```
 printf("Linked list \n");
  linkedListTraversal(head);
```

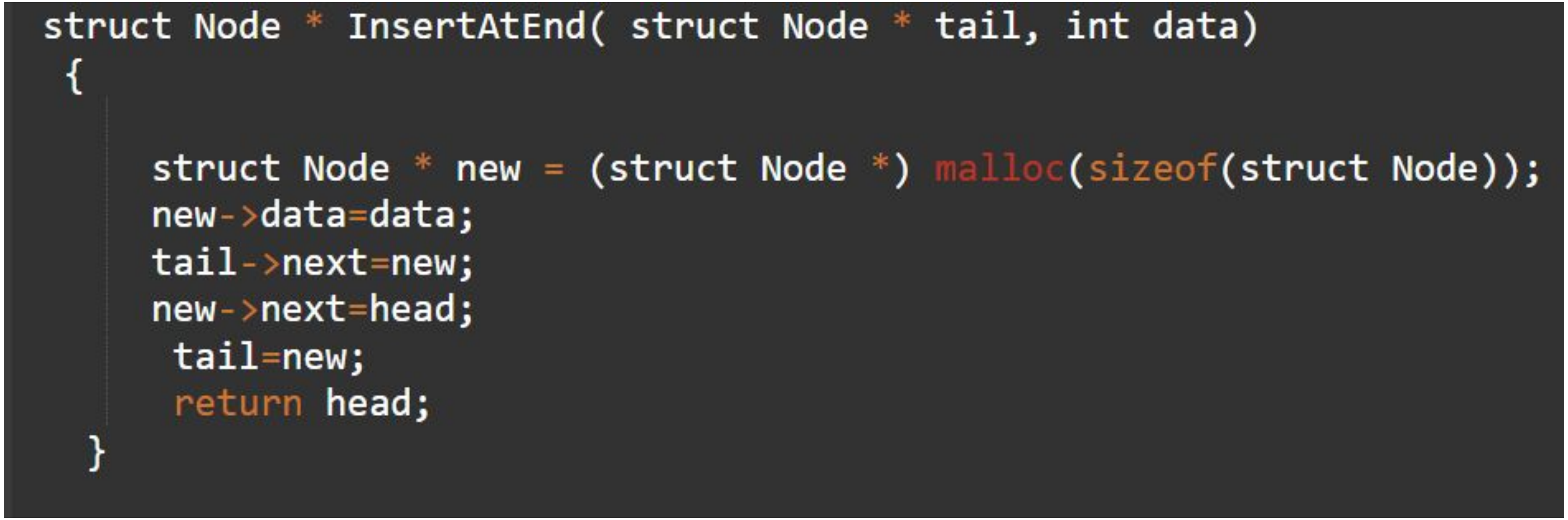


```c
//insert node at Begining
 struct Node * InsertAtBegin( struct Node * head, int data)
  {

      struct Node * new = (struct Node *) malloc(sizeof(struct Node));
      new->data=data;
      temp= head->next;
    while(temp->next != head){
        temp=temp->next;
    }
    // At this point temp points to the last node of this circular linked list

    temp->next = new;
    new->next = head;
    head = new;
    return head;
   }
```

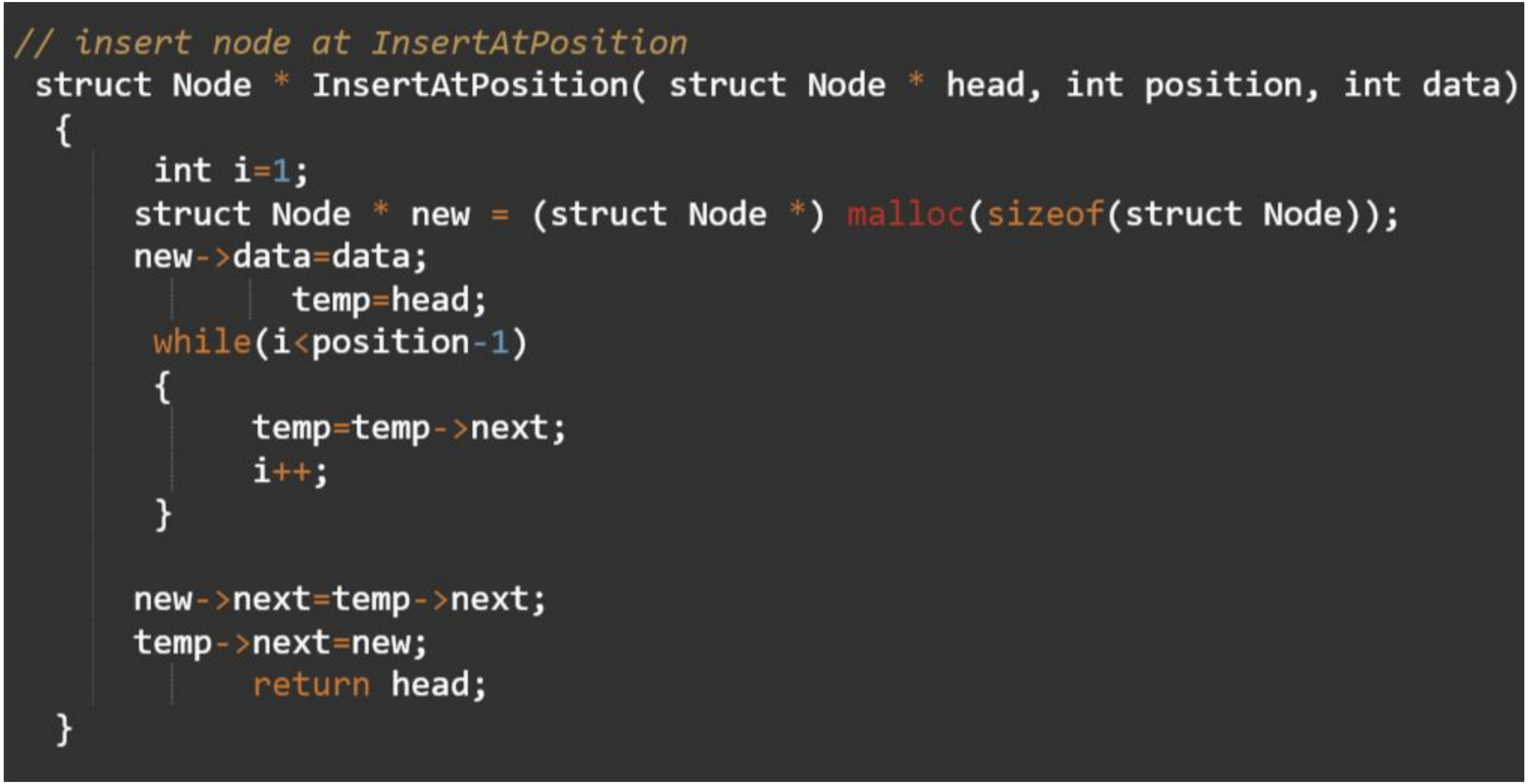
CLL insertion at begining



```
struct Node * InsertAtEnd( struct Node * tail, int data)
 {

     struct Node * new = (struct Node *) malloc(sizeof(struct Node));
     new->data=data;
     tail->next=new;
     new->next=head;
      tail=new;
      return head;
  }
```

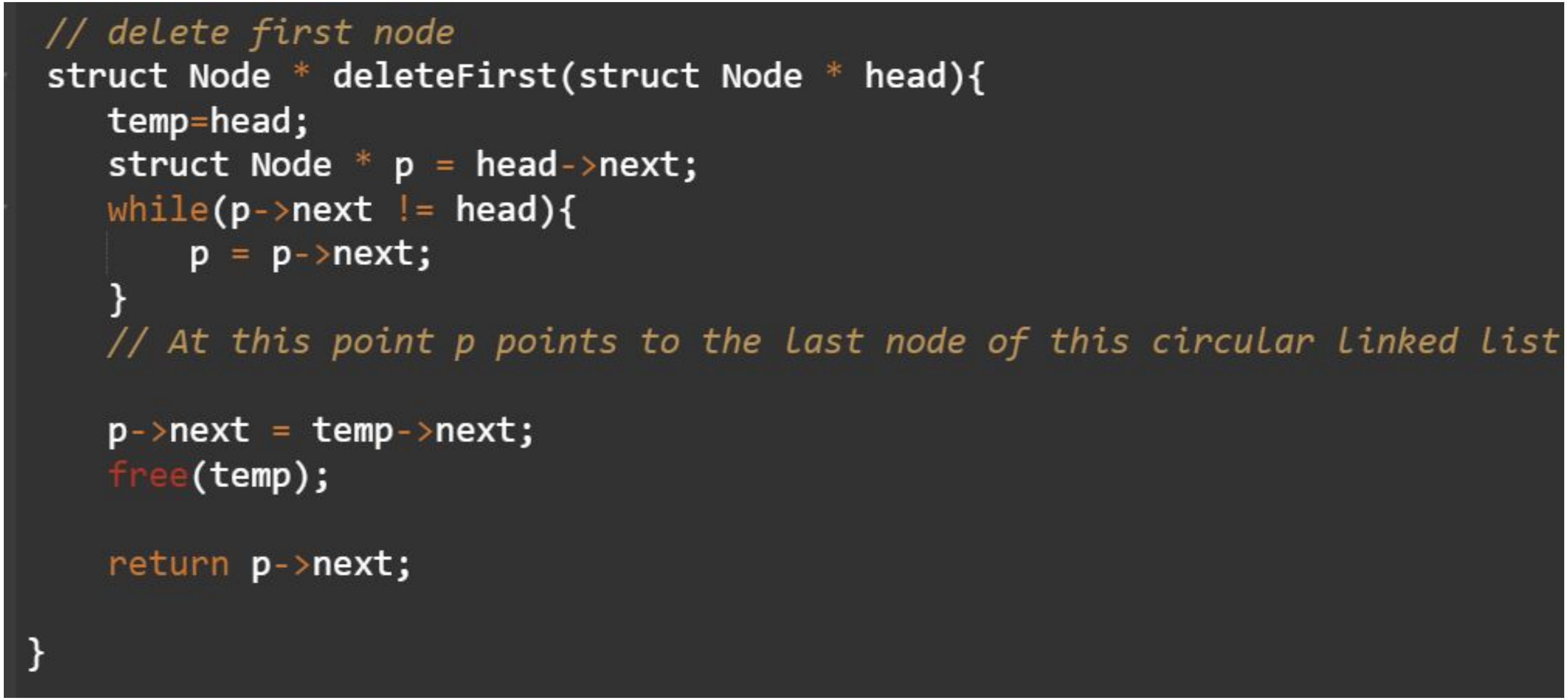
Insert at end



```c
// insert node at InsertAtPosition
 struct Node * InsertAtPosition( struct Node * head, int position, int data)
  {
       int i=1;
      struct Node * new = (struct Node *) malloc(sizeof(struct Node));
      new->data=data;
              temp=head;
       while(i<position-1)
       {
            temp=temp->next;
            i++;
       }

      new->next=temp->next;
      temp->next=new;
            return head;
  }
```

Insert at position



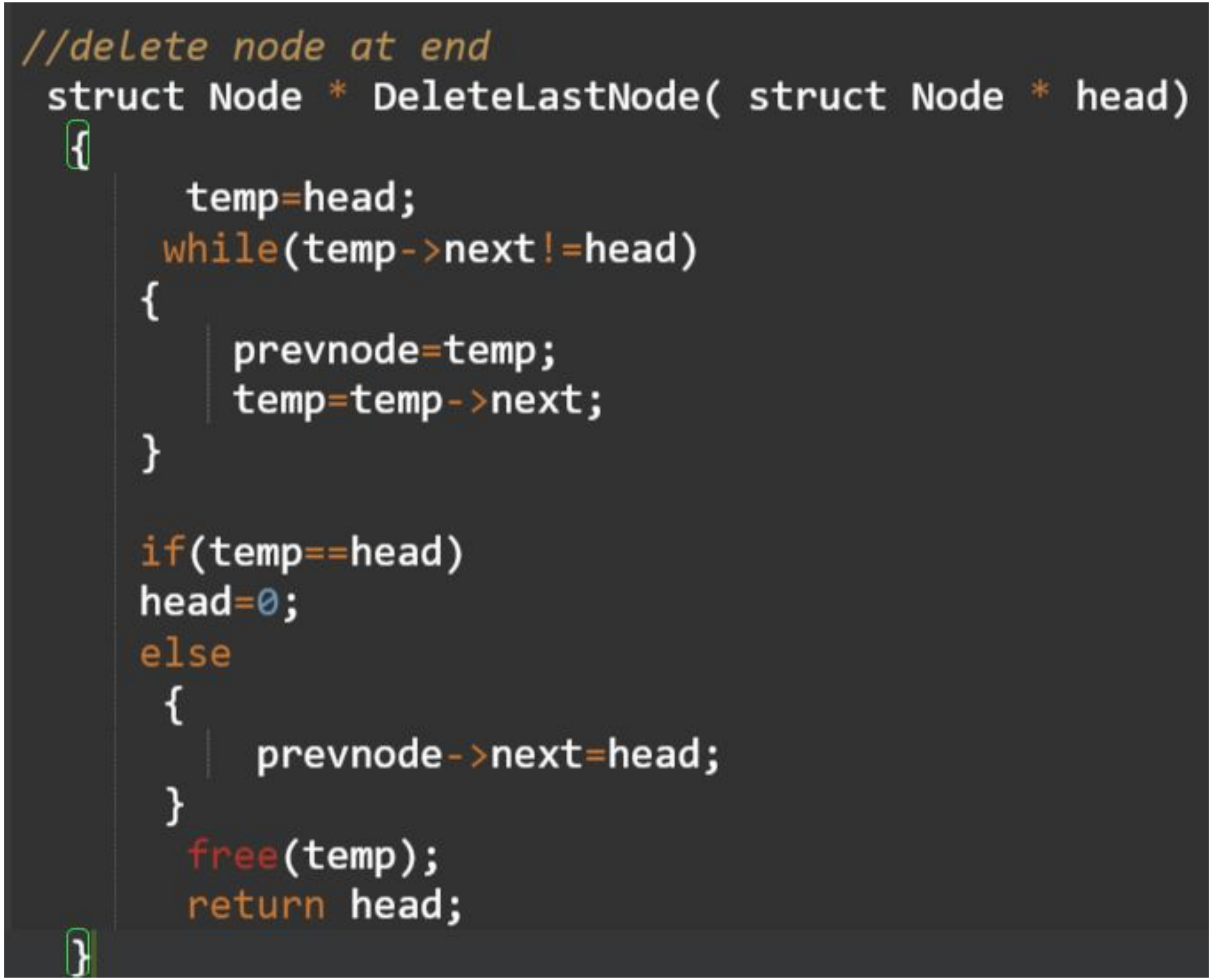
```
 // delete first node
 struct Node * deleteFirst(struct Node * head){
    temp=head;
    struct Node * p = head->next;
    while(p->next != head){
        p = p->next;
    }
    // At this point p points to the last node of this circular linked list

    p->next = temp->next;
    free(temp);

    return p->next;

}
```

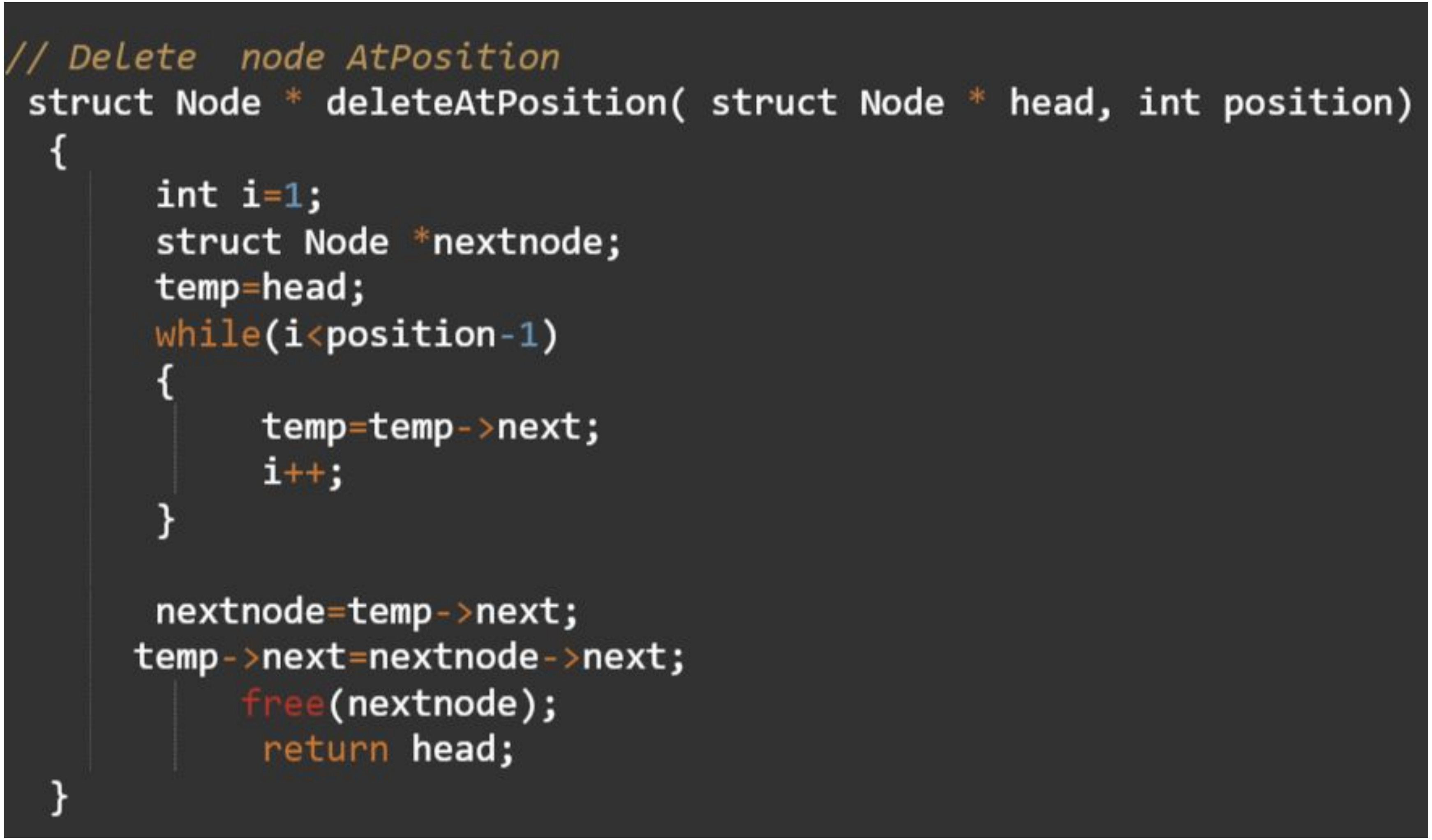
Delete first node



```
//delete node at end
 struct Node * DeleteLastNode( struct Node * head)
  {
       temp=head;
      while(temp->next!=head)
     {
         prevnode=temp;
         temp=temp->next;
     }

     if(temp==head)
     head=0;
     else
      {
          prevnode->next=head;
      }
       free(temp);
       return head;
  }
```

Delete last node



```
// Delete  node AtPosition
 struct Node * deleteAtPosition( struct Node * head, int position)
  {
        int i=1;
        struct Node *nextnode;
        temp=head;
        while(i<position-1)
        {
             temp=temp->next;
             i++;
        }

        nextnode=temp->next;
       temp->next=nextnode->next;
            free(nextnode);
             return head;
  }
```

CLL

Advantage

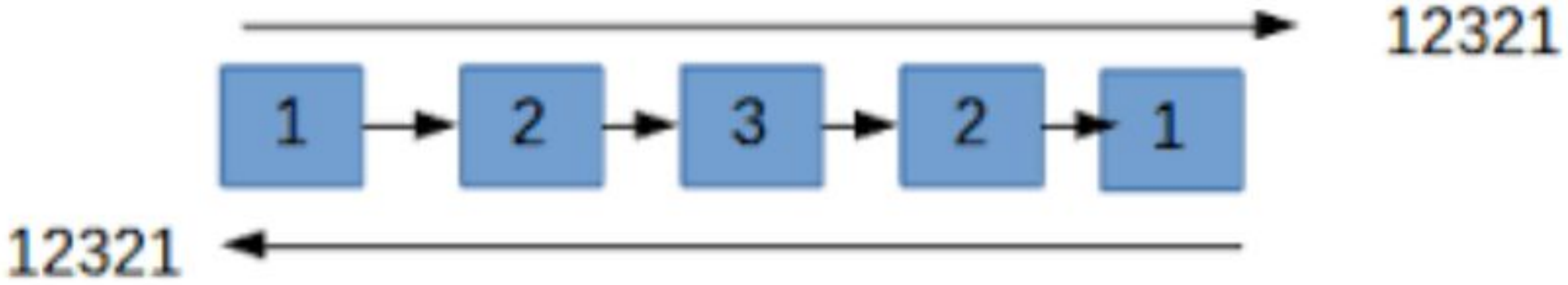
- We can traverse the entire list using any node as the starting point. It means that any node can be the starting point. We can just end the traversal when we reach the starting node again.
- Useful for the implementation of a queue. We don't need to store two-pointers that point to the front and the rear. Instead, we can just maintain a pointer to the last inserted node. By doing this. We can always obtain the front as the next of last.

Disadvantage

- Finding the end of the list is more complex, as there is no NULL to mark the end.
- Reversing of circular list is a complex as compared to singly or doubly lists.
- If not traversed carefully, then we could end up in an infinite loop.
-

You are given a circular doubly linked list that contains integers as the data in each node. These data on each node is distinct. You have developed a special algorithm that prints the three continuous elements of the list, starting from the first element or head of the list and runs for infinite time. For example, if the list is $\{1, 9, 12, 7\}$, then the output of the algorithm will be $\{1, 9, 12, 9, 12, 7, 12, 7, 1, \ldots\}$.

Palindromic Linked List



Practice problems

How to reverse a singly linked list?

How to find the middle element of the linked list

How do you find the length of a singly linked list?

Given a linked list and a value *x*, partition it such that all nodes less than *x* come before nodes greater than or equal to *x*