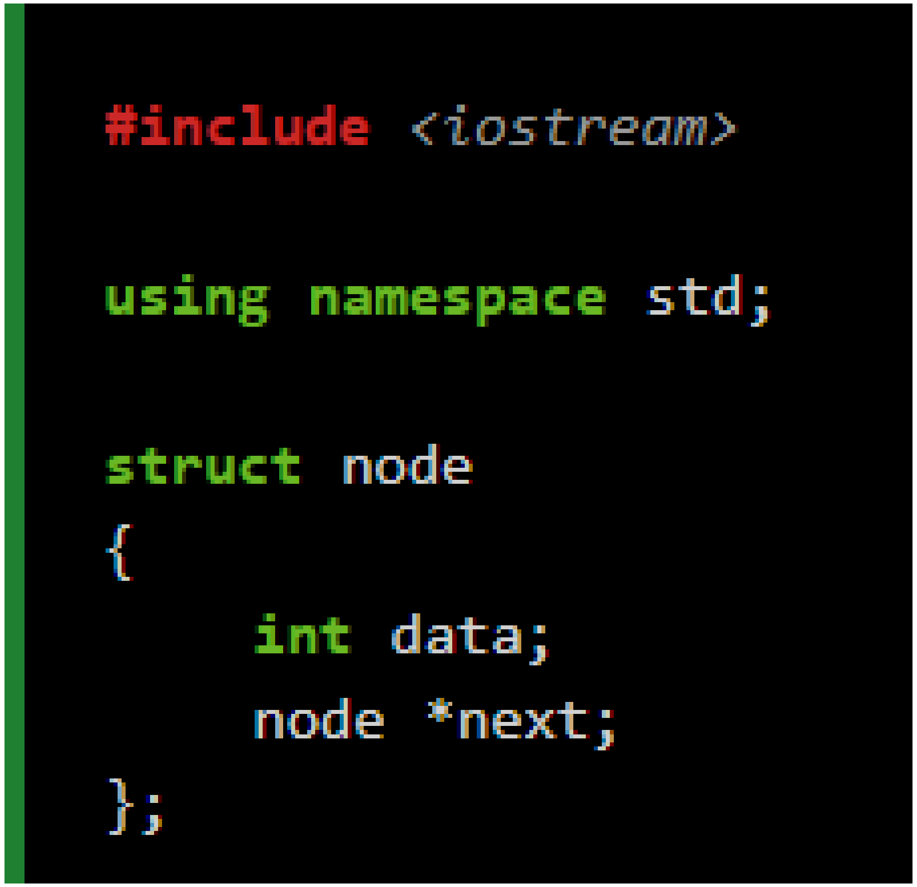
Linked List

Singly Linked List, Circular Linked List , and Doubly Linked List

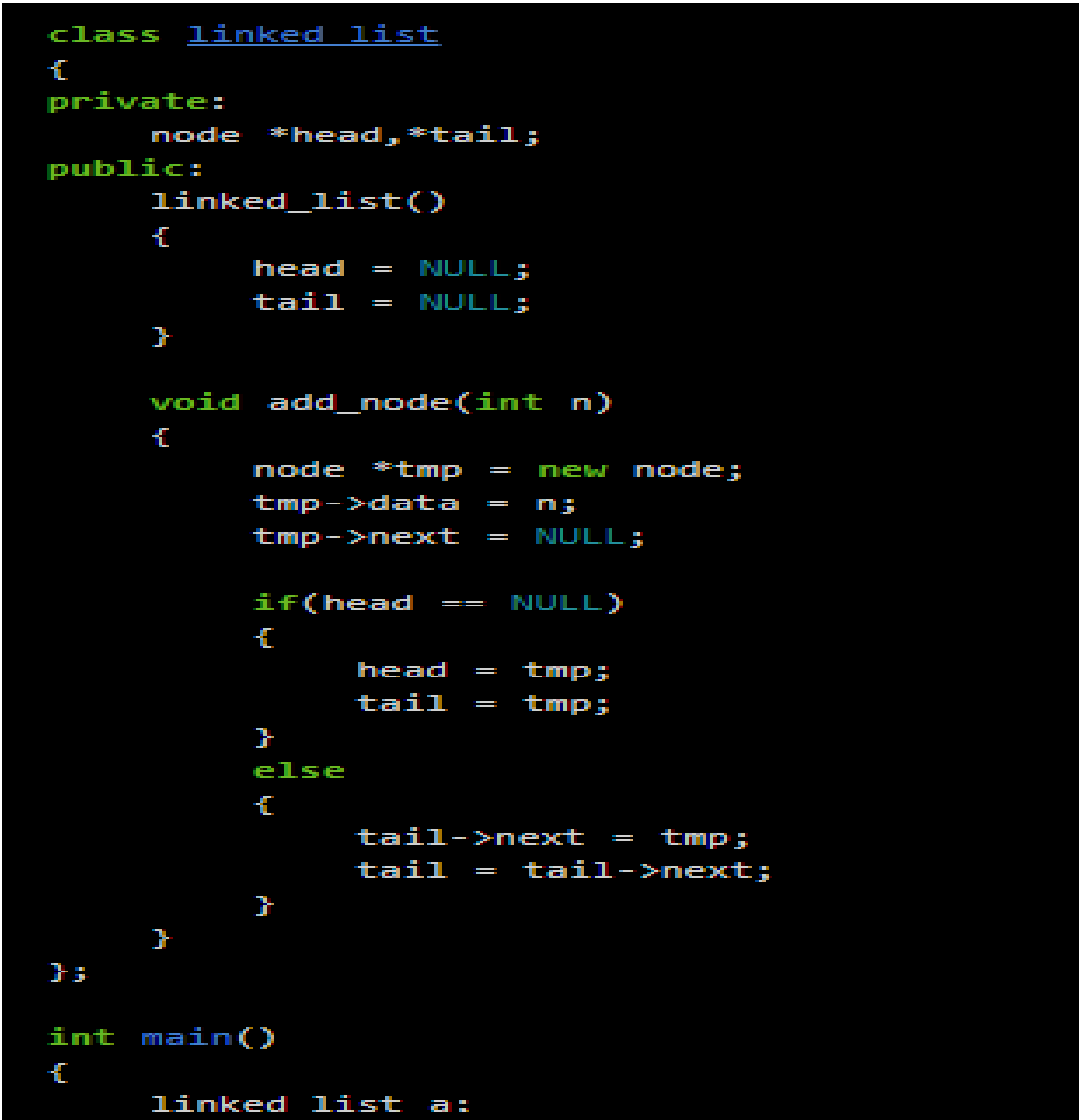
Instructor: Anam Qureshi

Singly Linked List

Structure of a Node



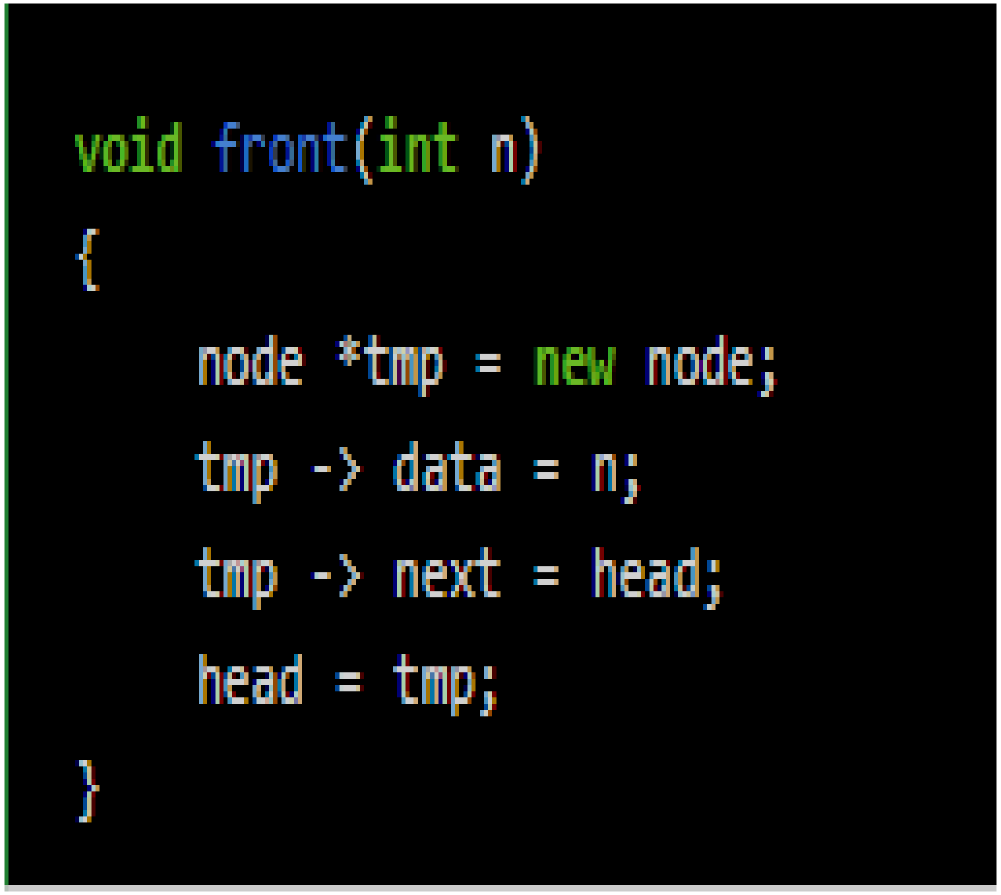
```
#include <iostream>

using namespace std;

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



```
class linked_list
{
private:
        node *head,*tail;
public:
        linked_list()
        {
                head = NULL;
                tail = NULL;
        }

        void add_node(int n)
        {
                node *tmp = new node;
                tmp->data = n;
                tmp->next = NULL;

                if(head == NULL)
                {
                        head = tmp;
                        tail = tmp;
                }
                else
                {
                        tail->next = tmp;
                        tail = tail->next;
                }
        }
};

int main()
{
        linked_list a;
```

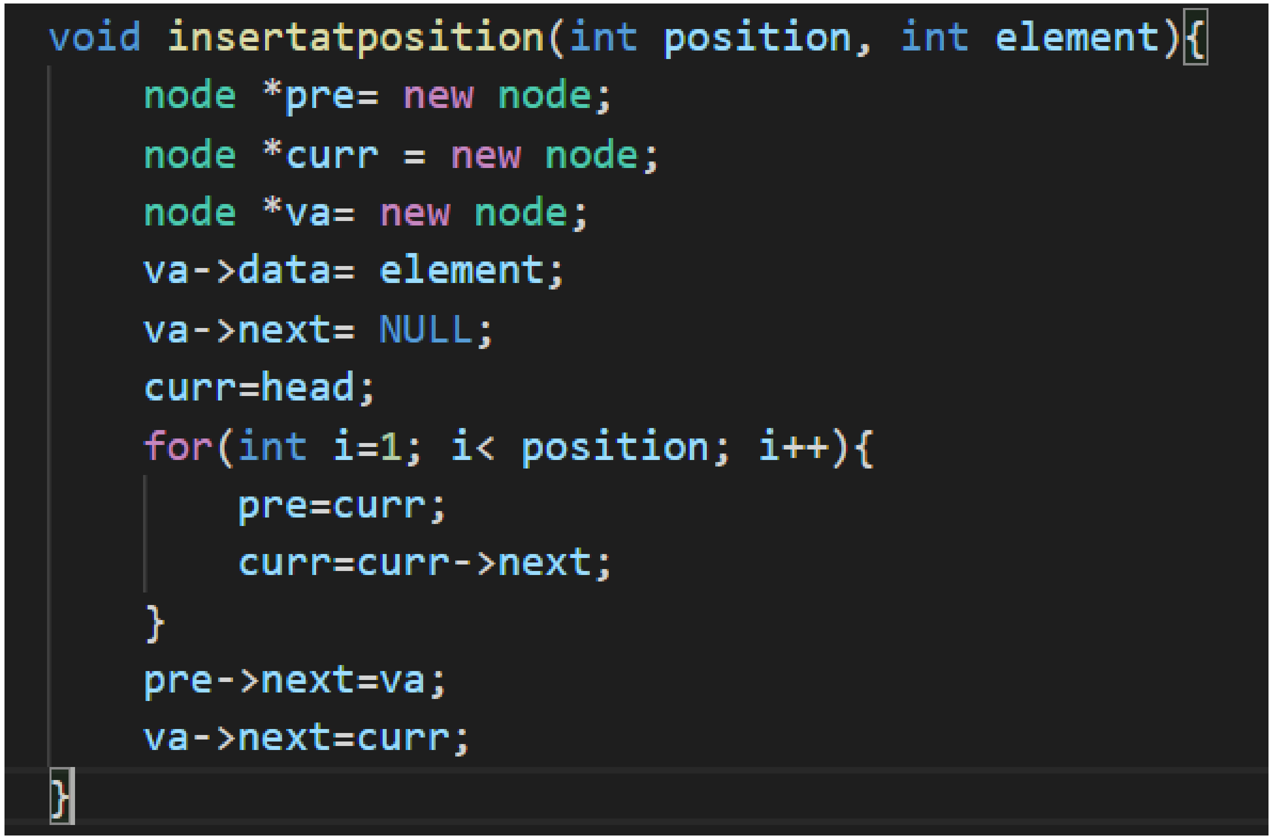
Insert in an empty LL and Insert at end

Insert at the beginning



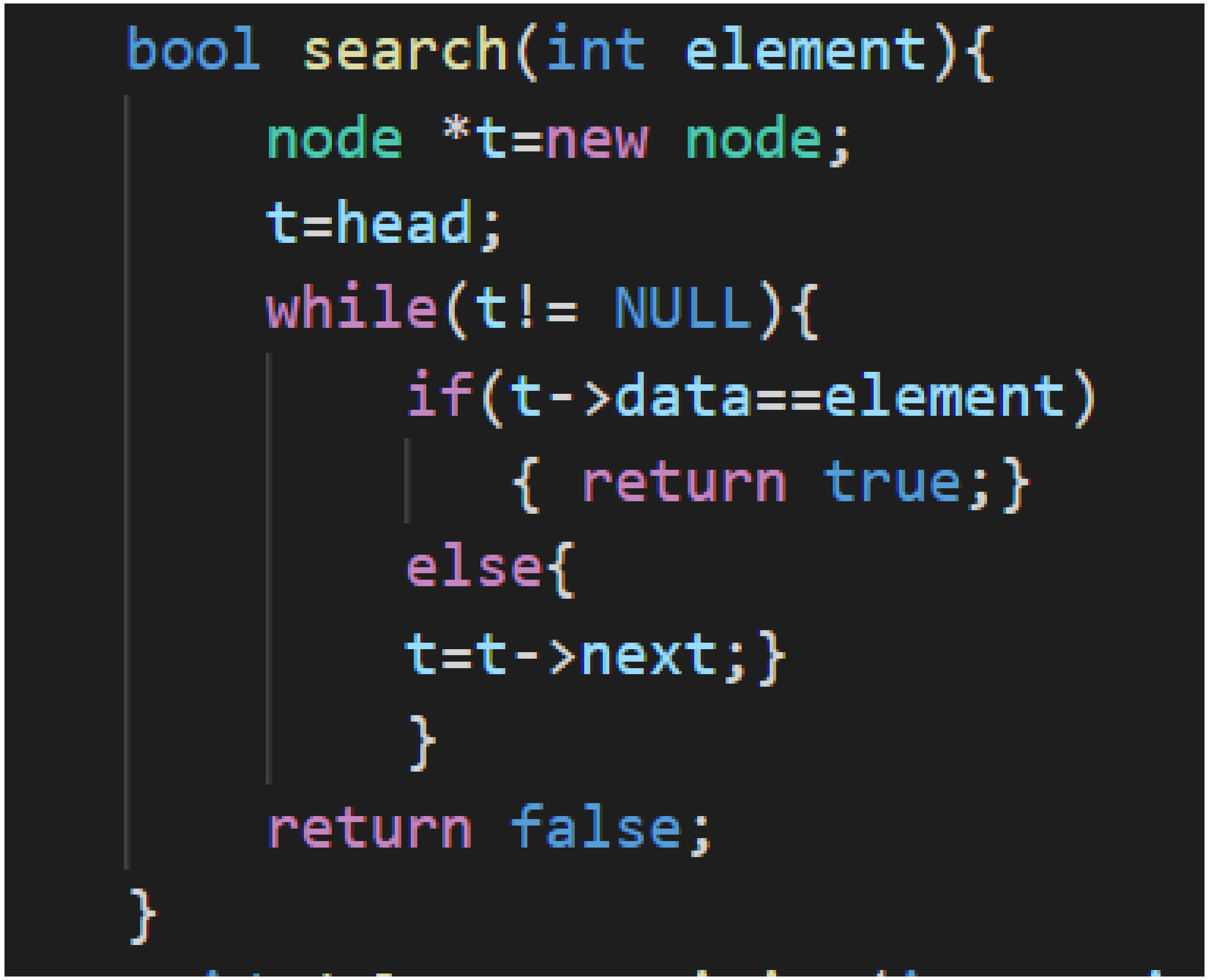
```
void front(int n)
{
    node *tmp = new node;
    tmp -> data = n;
    tmp -> next = head;
    head = tmp;
}
```

Insert at any position



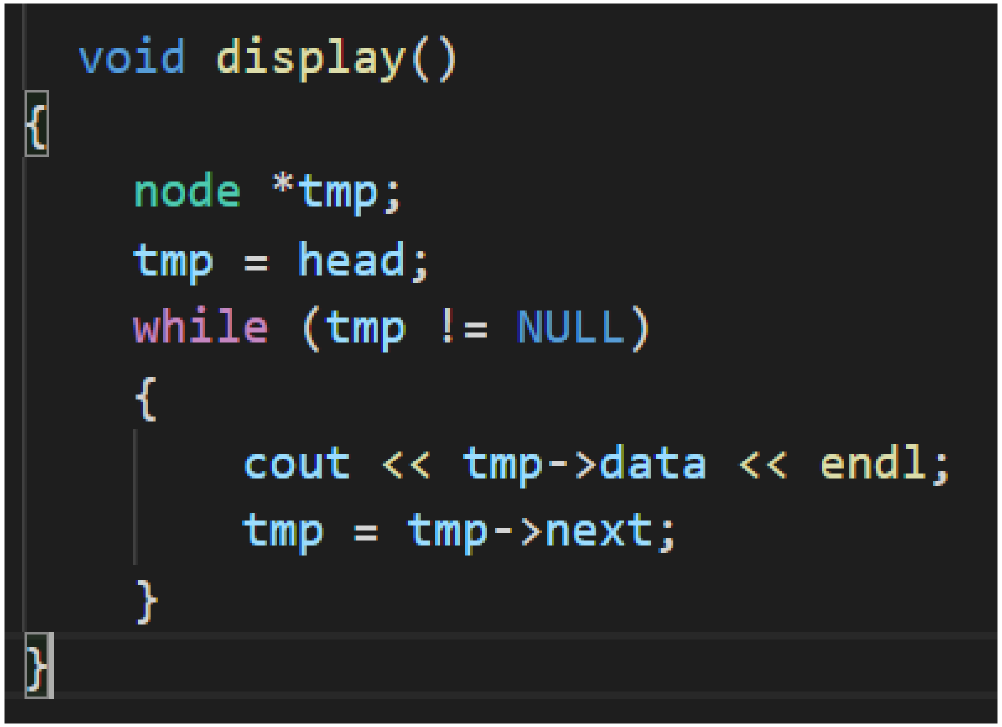
```
void insertatposition(int position, int element){
    node *pre= new node;
    node *curr = new node;
    node *va= new node;
    va->data= element;
    va->next= NULL;
    curr=head;
    for(int i=1; i< position; i++){
        pre=curr;
        curr=curr->next;
    }
    pre->next=va;
    va->next=curr;
}
```

Searching in Linked List



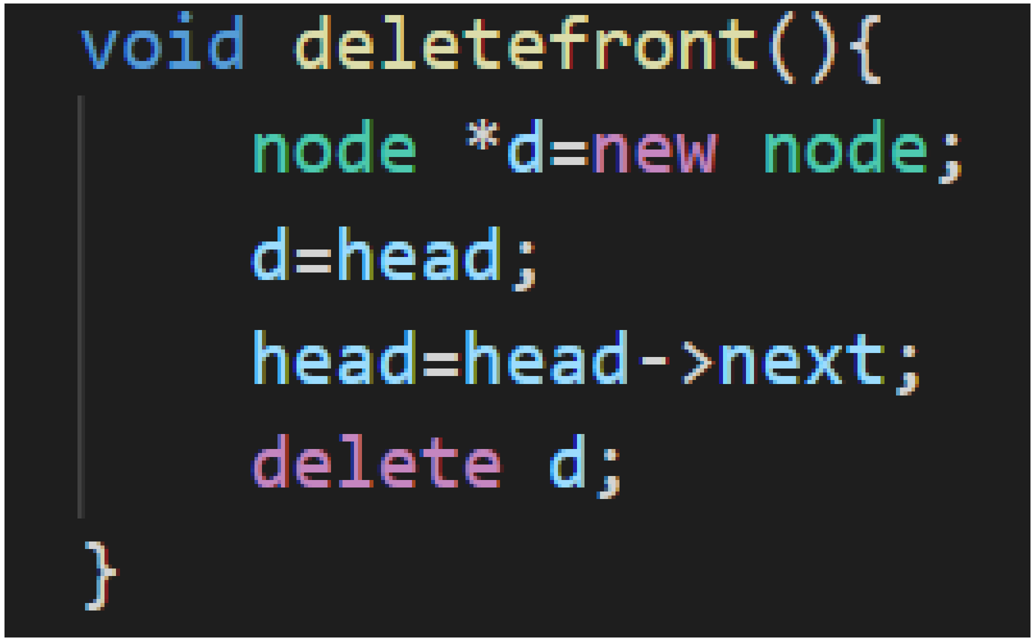
```
bool search(int element){
    node *t=new node;
    t=head;
    while(t!= NULL){
        if(t->data==element)
           { return true;}
        else{
        t=t->next;}
        }
    return false;
}
```

Display Linked List



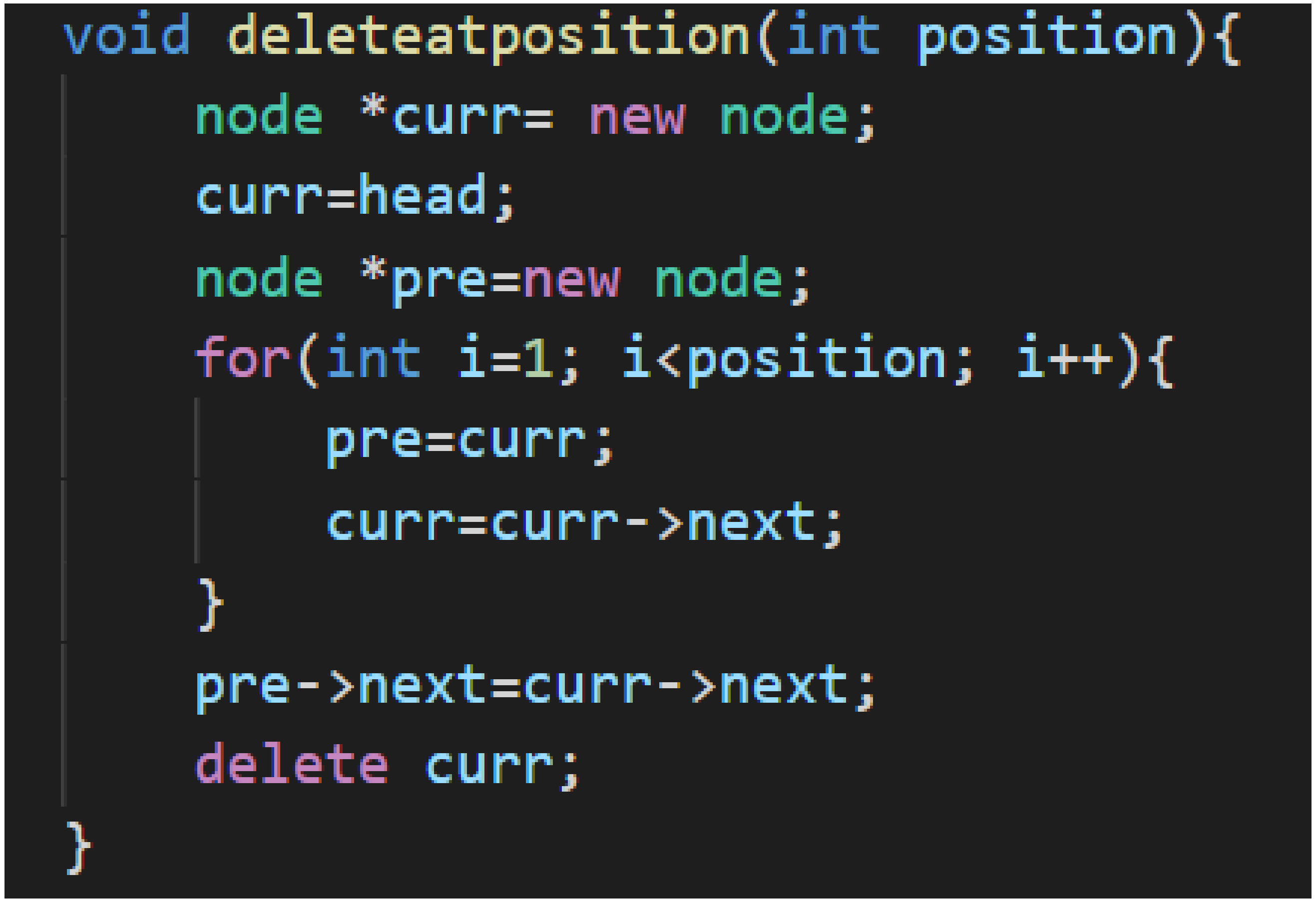
```
  void display()
{
    node *tmp;
    tmp = head;
    while (tmp != NULL)
    {
        cout << tmp->data << endl;
        tmp = tmp->next;
    }
}
```

Delete at front



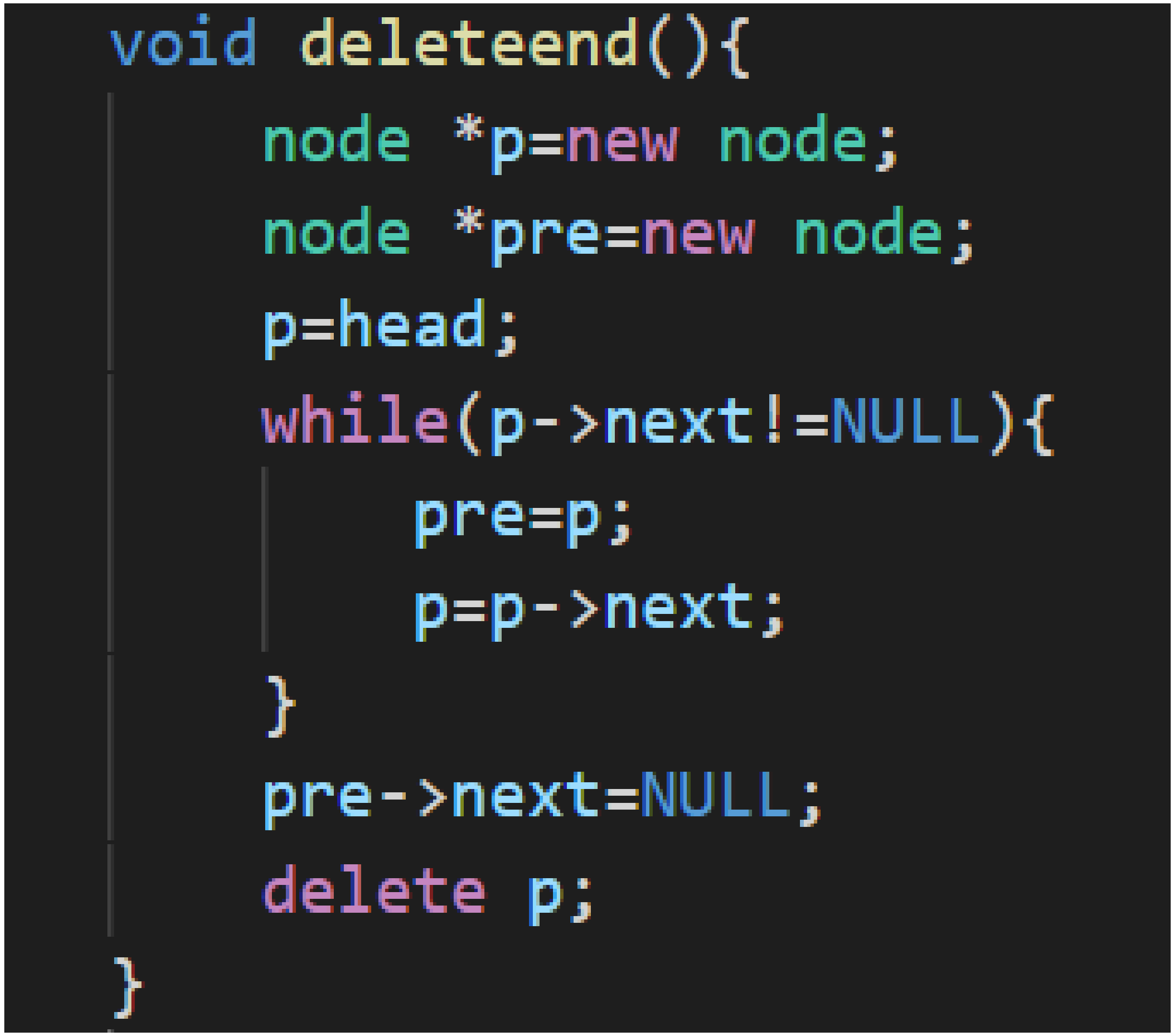
```
void deletefront(){
    node *d=new node;
    d=head;
    head=head->next;
    delete d;
}
```

Delete at any position



```
void deleteatposition(int position){
    node *curr= new node;
    curr=head;
    node *pre=new node;
    for(int i=1; i<position; i++){
        pre=curr;
        curr=curr->next;
    }
    pre->next=curr->next;
    delete curr;
}
```

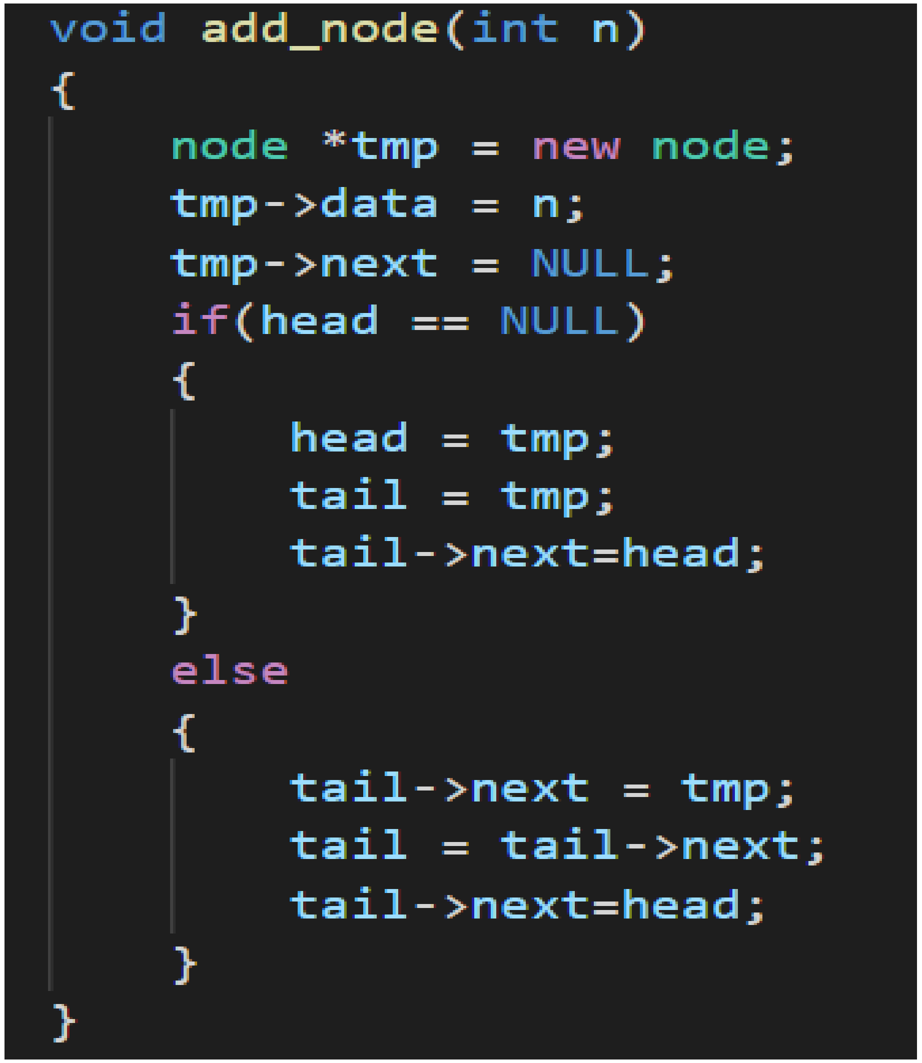
Deletion at the end



```
void deleteend(){
    node *p=new node;
    node *pre=new node;
    p=head;
    while(p->next!=NULL){
        pre=p;
        p=p->next;
    }
    pre->next=NULL;
    delete p;
}
```

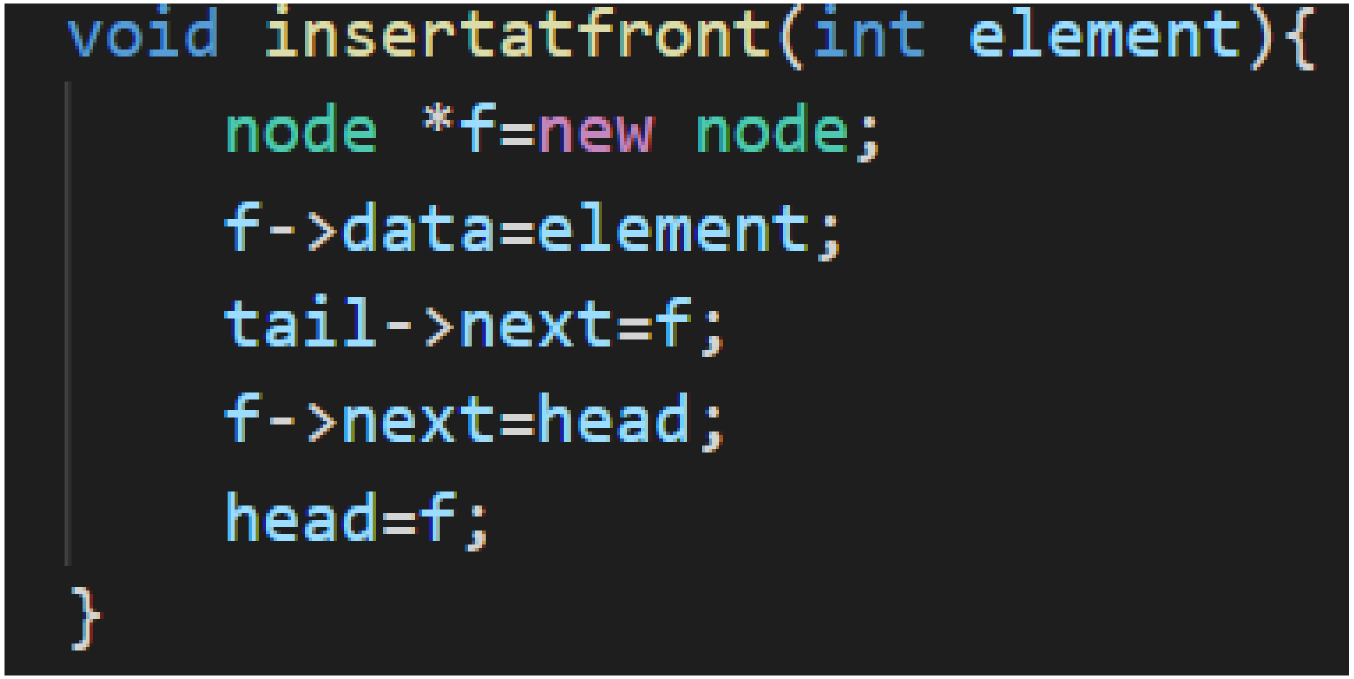
Circular Linked List

Insert in an empty CLL and at end



```
void add_node(int n)
{
    node *tmp = new node;
    tmp->data = n;
    tmp->next = NULL;
    if(head == NULL)
    {
        head = tmp;
        tail = tmp;
        tail->next=head;
    }
    else
    {
        tail->next = tmp;
        tail = tail->next;
        tail->next=head;
    }
}
```

Insert at Front

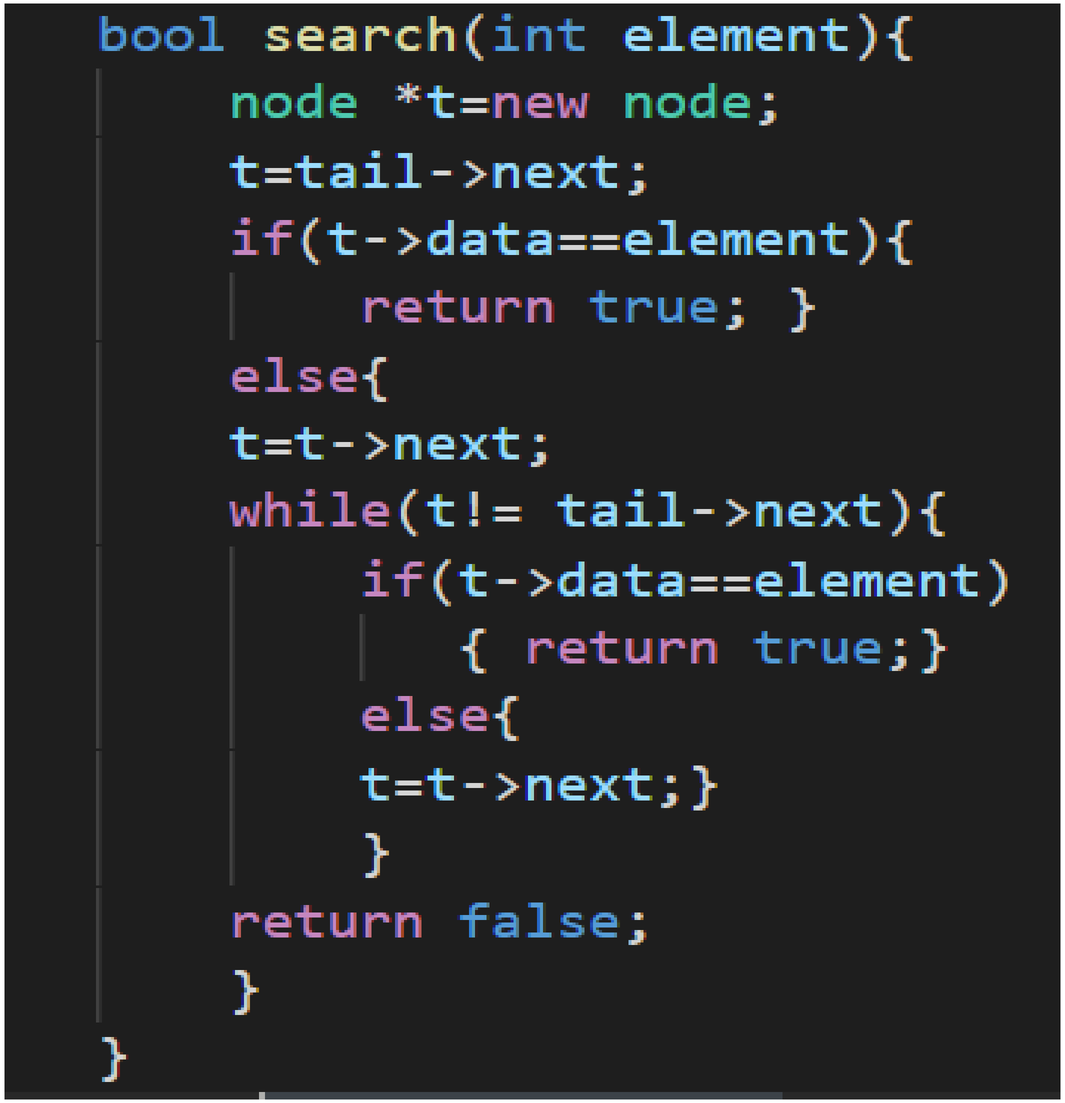


```
void insertatfront(int element){
    node *f=new node;
    f->data=element;
    tail->next=f;
    f->next=head;
    head=f;
}
```

Insert at any position

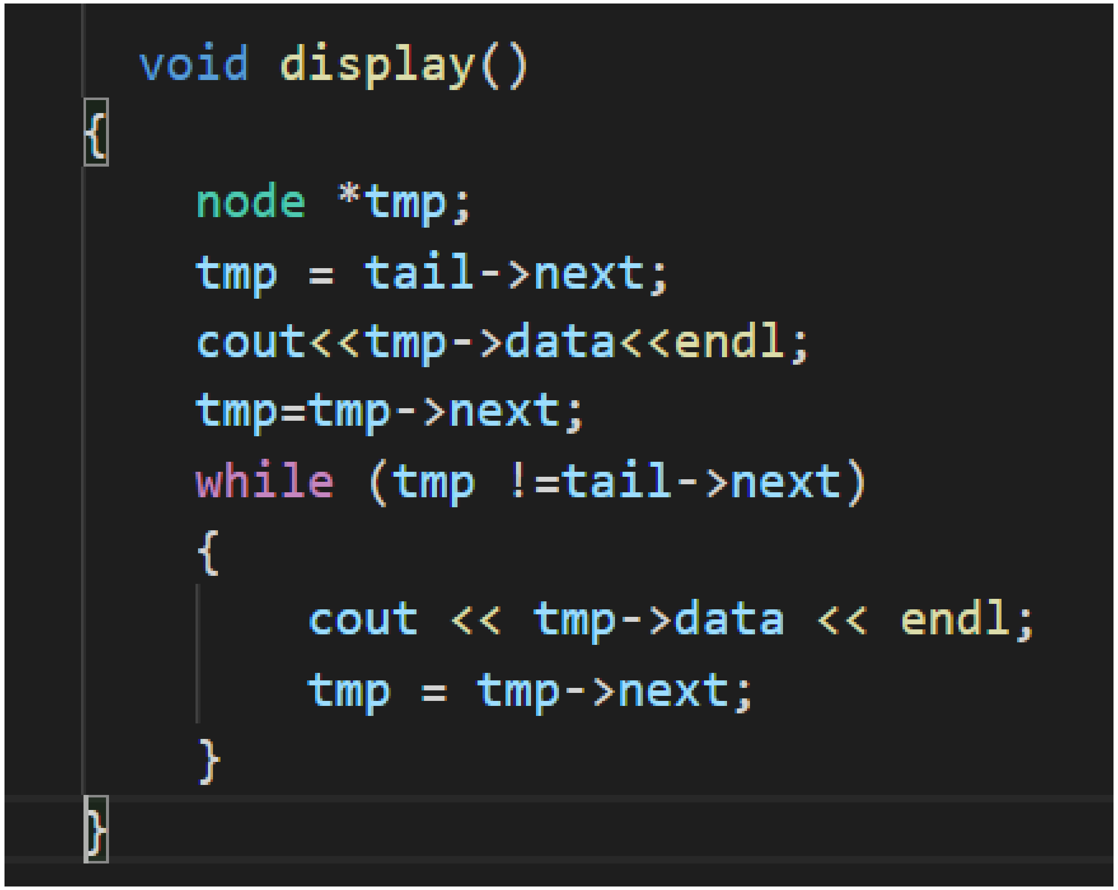
- Same as SLL

Searching



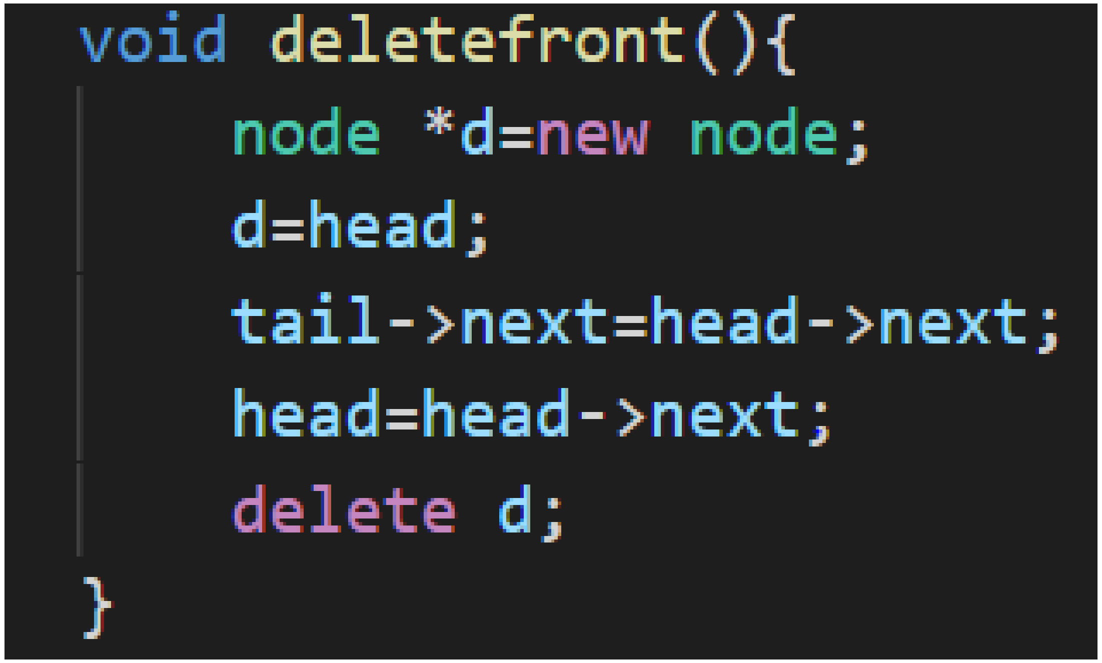
```
bool search(int element){
    node *t=new node;
    t=tail->next;
    if(t->data==element){
        return true; }
    else{
    t=t->next;
    while(t!= tail->next){
        if(t->data==element)
            { return true;}
        else{
        t=t->next;}
        }
    return false;
    }
}
```

Display Linked List



```
  void display()
{
    node *tmp;
    tmp = tail->next;
    cout<<tmp->data<<endl;
    tmp=tmp->next;
    while (tmp !=tail->next)
    {
        cout << tmp->data << endl;
        tmp = tmp->next;
    }
}
```

Delete at front

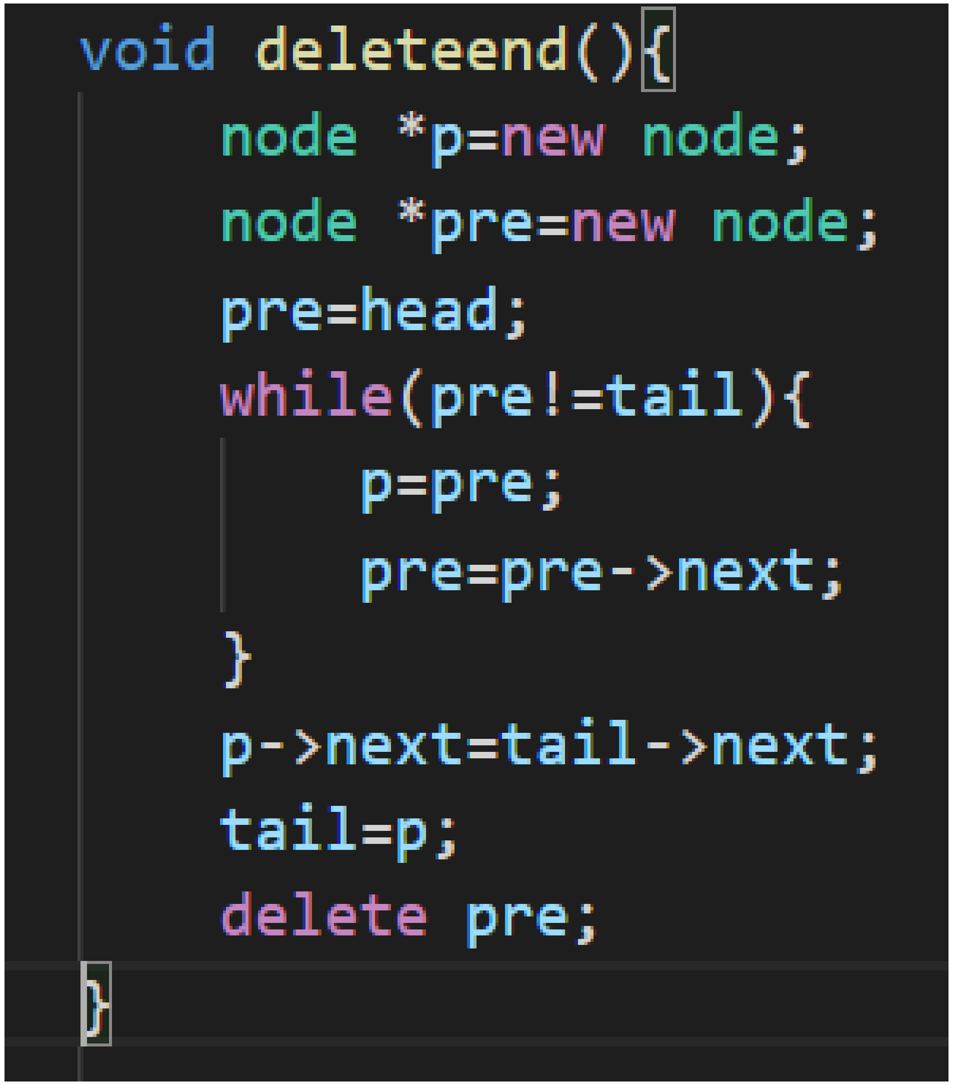


```
void deletefront(){
    node *d=new node;
    d=head;
    tail->next=head->next;
    head=head->next;
    delete d;
}
```

Delete at any position

- Same as SLL

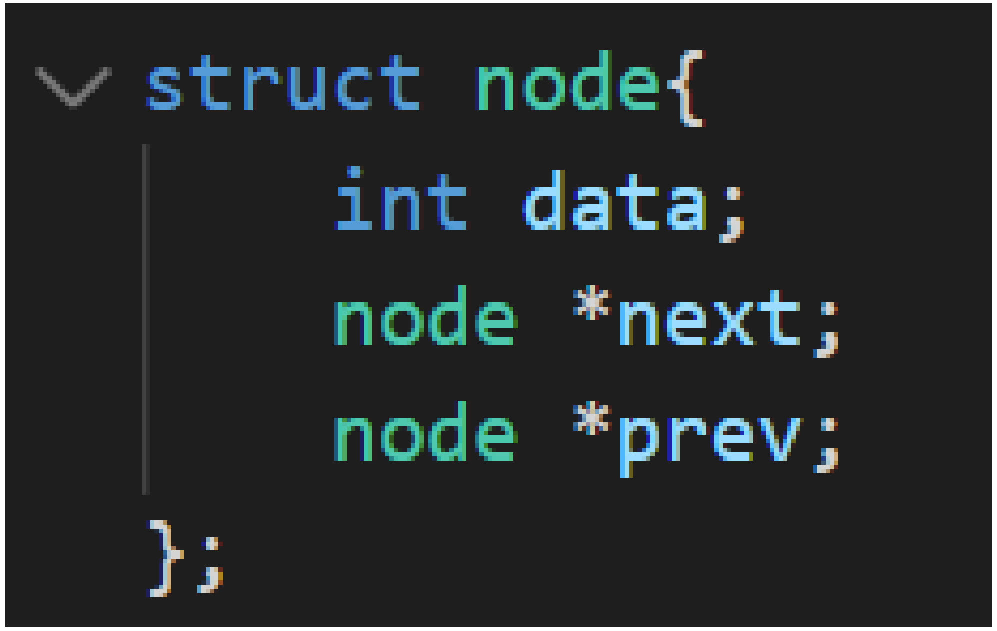
Delete at end



```
void deleteend(){
    node *p=new node;
    node *pre=new node;
    pre=head;
    while(pre!=tail){
        p=pre;
        pre=pre->next;
    }
    p->next=tail->next;
    tail=p;
    delete pre;
}
```

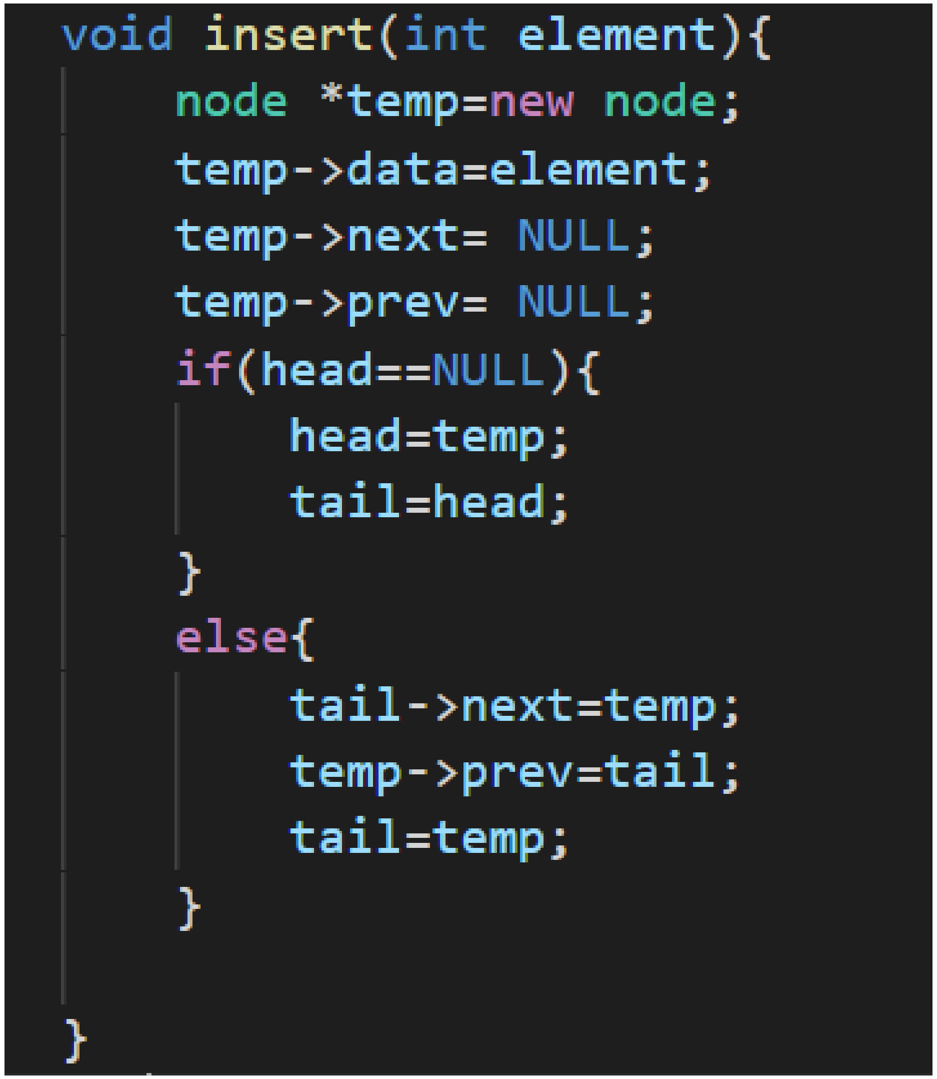
Doubly Linked List (DLL)

Structure of a Node



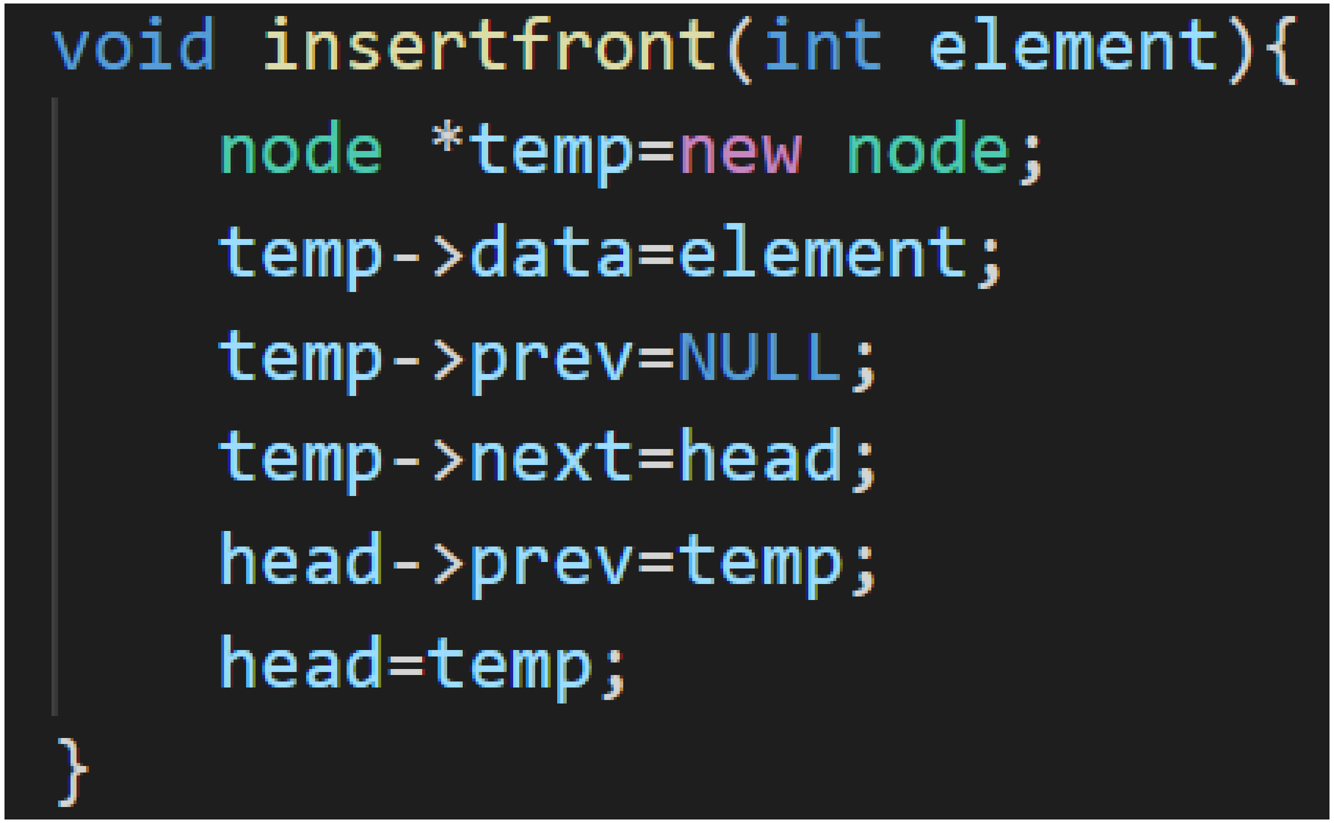
```
struct node{
    int data;
    node *next;
    node *prev;
};
```

Insert in an empty DLL and at end



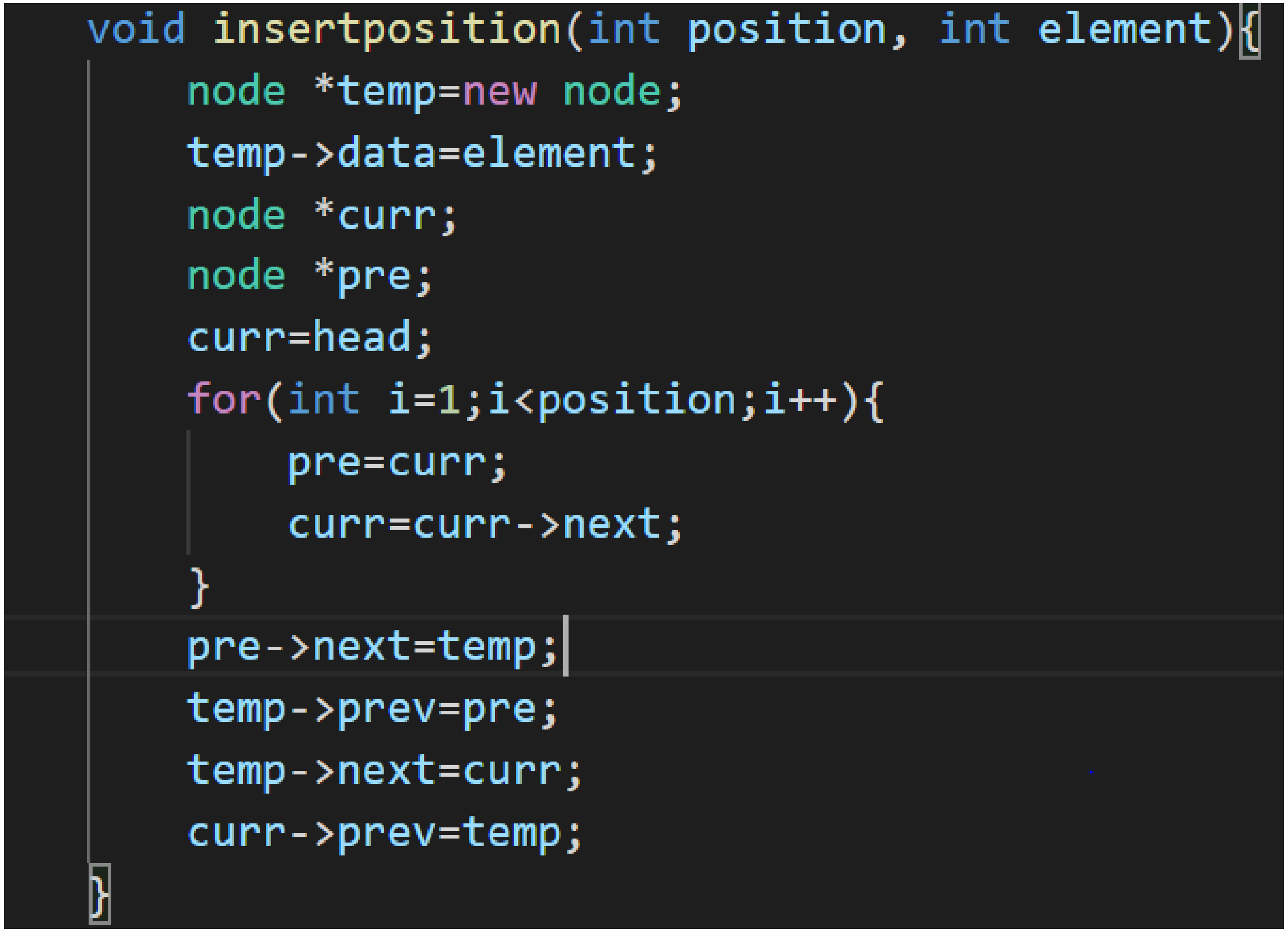
```
void insert(int element){
    node *temp=new node;
    temp->data=element;
    temp->next= NULL;
    temp->prev= NULL;
    if(head==NULL){
        head=temp;
        tail=head;
    }
    else{
        tail->next=temp;
        temp->prev=tail;
        tail=temp;
    }

}
```

Insert at front



```
void insertfront(int element){
    node *temp=new node;
    temp->data=element;
    temp->prev=NULL;
    temp->next=head;
    head->prev=temp;
    head=temp;
}
```

Insert at any position



```
void insertposition(int position, int element){
    node *temp=new node;
    temp->data=element;
    node *curr;
    node *pre;
    curr=head;
    for(int i=1;i<position;i++){
        pre=curr;
        curr=curr->next;
    }
    pre->next=temp;
    temp->prev=pre;
    temp->next=curr;
    curr->prev=temp;
}
```

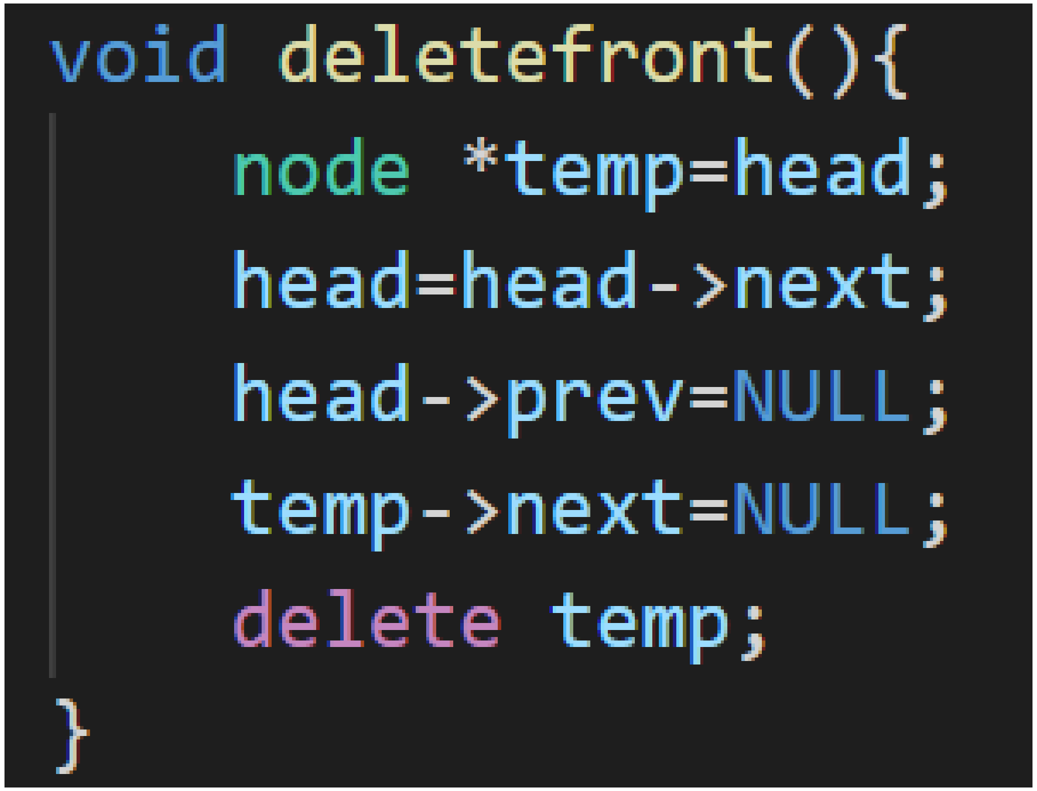
Searching

- Similar to SLL

Display

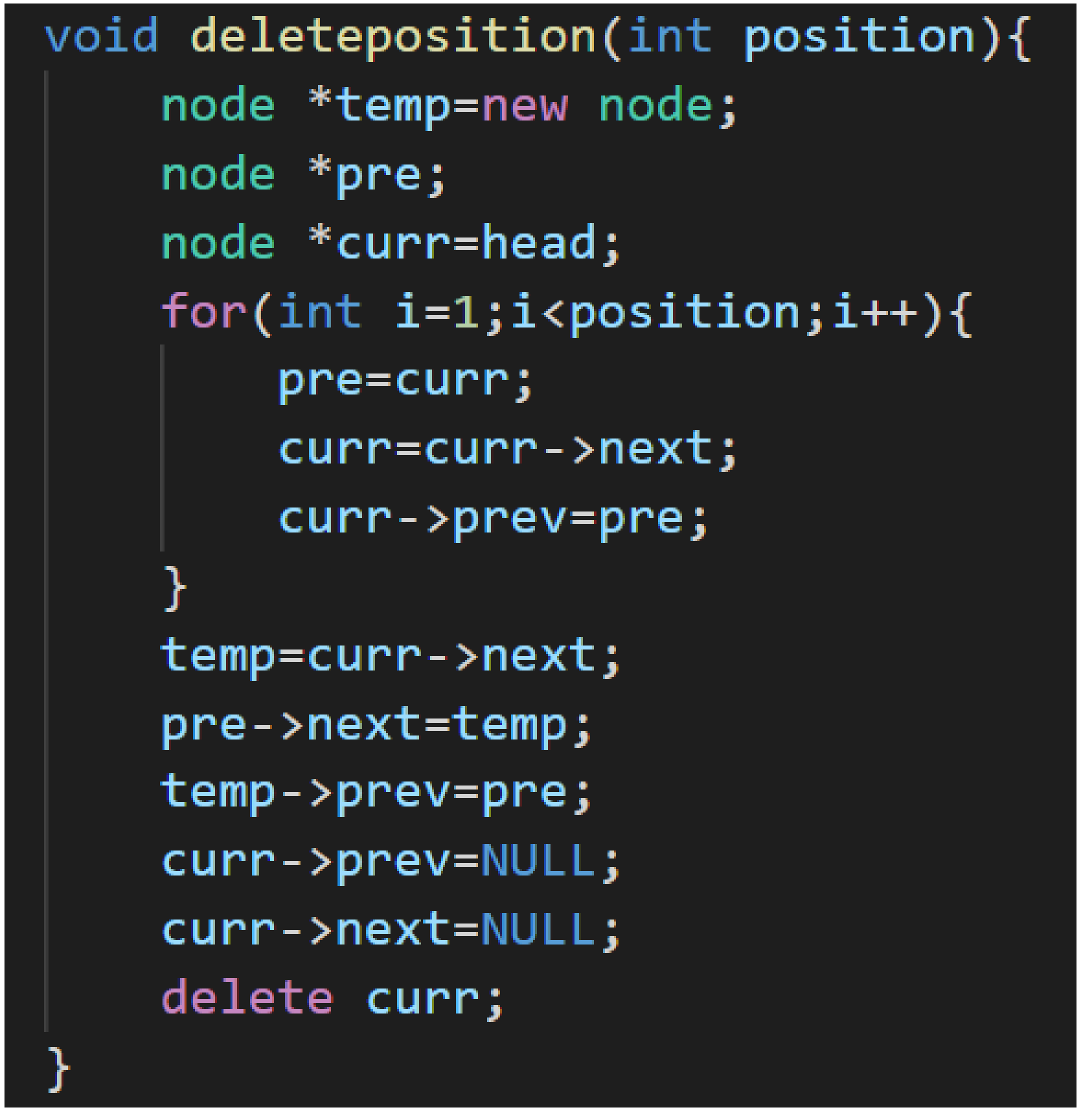
- Similar to SLL

Delete at front



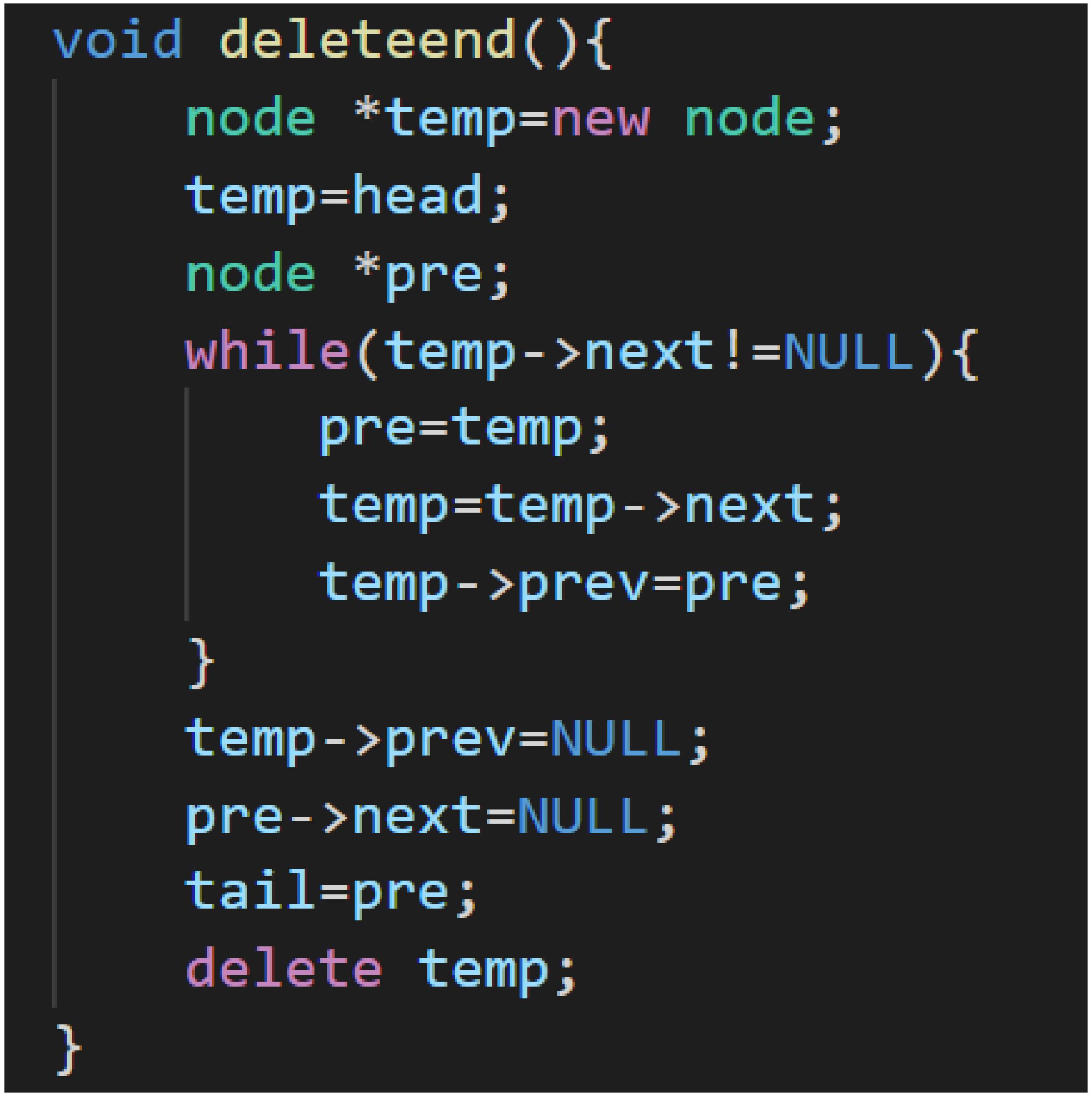
```
void deletefront(){
    node *temp=head;
    head=head->next;
    head->prev=NULL;
    temp->next=NULL;
    delete temp;
}
```

Delete at any position



```
void deleteposition(int position){
    node *temp=new node;
    node *pre;
    node *curr=head;
    for(int i=1;i<position;i++){
        pre=curr;
        curr=curr->next;
        curr->prev=pre;
    }
    temp=curr->next;
    pre->next=temp;
    temp->prev=pre;
    curr->prev=NULL;
    curr->next=NULL;
    delete curr;
}
```

Delete at end



```
void deleteend(){
    node *temp=new node;
    temp=head;
    node *pre;
    while(temp->next!=NULL){
        pre=temp;
        temp=temp->next;
        temp->prev=pre;
    }
    temp->prev=NULL;
    pre->next=NULL;
    tail=pre;
    delete temp;
}
```

Class Activity

Circular Linked List using DLL

Open Discussion on Applications of Linked List