GeeksforGeeks

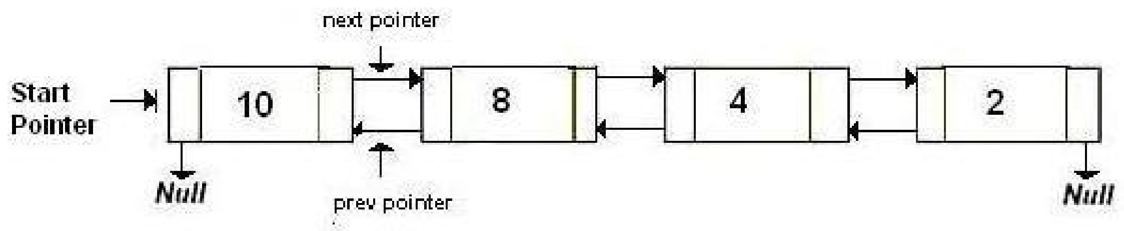
A computer science portal for geeks

Practice IDE Q&A GeeksQuiz

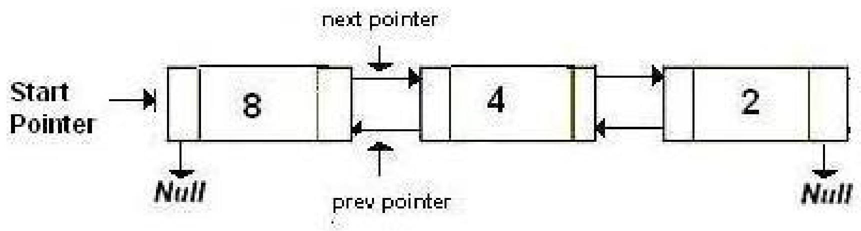
# Delete a node in a Doubly Linked List

Write a function to delete a given node in a doubly linked list.

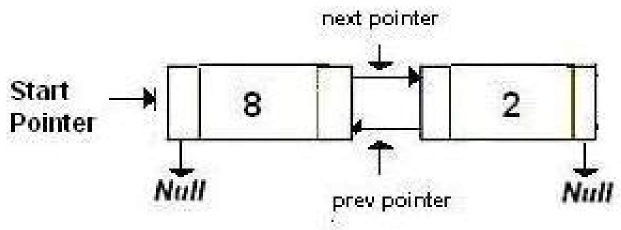
```
(a) Original Doubly Linked List
```



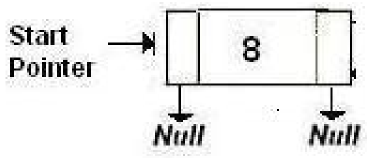
```
(a) After deletion of head node
```



```
(a) After deletion of middle node
```



```
(a) After deletion of last node
```



Algorithm

Let the node to be deleted is *del*.

1) If node to be deleted is head node, then change the head pointer to next current head.

2) Set *next* of previous to *del*, if previous to *del* exixts.

3) Set *prev* of next to *del*, if next to *del* exixts.

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

/* a node of the doubly linked list */
struct node
{
  int data;
  struct node *next;
  struct node *prev;
};

/* Function to delete a node in a Doubly Linked List.
   head_ref --> pointer to head node pointer.
   del  -->  pointer to node to be deleted. */
void deleteNode(struct node **head_ref, struct node *del)
{
  /* base case */
  if(*head_ref == NULL || del == NULL)
    return;

  /* If node to be deleted is head node */
  if(*head_ref == del)
    *head_ref = del->next;

  /* Change next only if node to be deleted is NOT the last node */
  if(del->next != NULL)
    del->next->prev = del->prev;

  /* Change prev only if node to be deleted is NOT the first node */
  if(del->prev != NULL)
    del->prev->next = del->next;

  /* Finally, free the memory occupied by del*/
  free(del);
  return;
}

/* UTILITY FUNCTIONS */
/* Function to insert a node at the beginning of the Doubly Linked List */
void push(struct node** head_ref, int new_data)
{
  /* allocate node */
  struct node* new_node =
       (struct node*) malloc(sizeof(struct node));

  /* put in the data  */
  new_node->data  = new_data;

  /* since we are adding at the begining,
    prev is always NULL */
  new_node->prev = NULL;

  /* link the old list off the new node */
  new_node->next = (*head_ref);

  /* change prev of head node to new node */
  if((*head_ref) !=  NULL)
   (*head_ref)->prev = new_node ;

  /* move the head to point to the new node */
   (*head_ref)    = new_node;
}

/* Function to print nodes in a given doubly linked list
   This function is same as printList() of singly linked lsit */
void printList(struct node *node)
{
  while(node!=NULL)
  {
   printf("%d ", node->data);
   node = node->next;
  }
}

/* Drier program to test above functions*/
int main()
{
  /* Start with the empty list */
  struct node* head = NULL;

  /* Let us create the doubly linked list 10<->8<->4<->2 */
  push(&head, 2);
  push(&head, 4);
  push(&head, 8);
  push(&head, 10);

  printf("\n Original Linked list ");
  printList(head);

  /* delete nodes from the doubly linked list */
  deleteNode(&head, head);  /*delete first node*/
  deleteNode(&head, head->next);  /*delete middle node*/
  deleteNode(&head, head->next);  /*delete last node*/

  /* Modified linked list will be NULL<-8->NULL */
  printf("\n Modified Linked list ");
  printList(head);

  getchar();
}
```

Time Complexity: O(1)

Time Complexity: O(1)

Please write comments if you find any of the above codes/algorithms incorrect, or find better ways to solve the same problem.

24 Comments Category: Linked Lists

## Related Posts:

- Merge two sorted linked lists such that merged list is in reverse order
- Compare two strings represented as linked lists
- Rearrange a given linked list in-place.
- Sort a linked list that is sorted alternating ascending and descending orders?

- Select a Random Node from a Singly Linked List
- Merge Sort for Doubly Linked List
- Point to next higher value node in a linked list with an arbitrary pointer
- Swap nodes in a linked list without swapping data

(Login to Rate and Mark)

1 Average Difficulty : **1/5.0**
Based on **1** vote(s)

Add to TODO List

Mark as DONE

Like Share 4 people like this.

Writing code in comment? Please use code.geeksforgeeks.org, generate link and share the link here.

**24 Comments** **GeeksforGeeks** 1 Login

Recommend 3 Share Sort by Newest

Join the discussion…

**Amanshu Kataria** • 4 months ago

The code given for deletion of node requires a pointer to the node to be deleted, but what if there are 100 nodes in the list.
I think the following code may work in that case:

http://ideone.com/VRdB83

• Reply • Share ›

**Hitesh Saini** • 6 months ago

we can also perform deletion in singly linked list in O(1).
then what is advantage of DLL.

• Reply • Share ›

**Prashant Pacific** Hitesh Saini • 6 months ago

and if we take the advantage part then it would be useful where we need to traverse the list in both the direction , like binary search , quick sort , insertion sort etc

• Reply • Share ›

**Hitesh Saini** Prashant Pacific • 6 months ago

ohk! thank you for the valuable reply..:)

• Reply • Share ›

**Prashant Pacific** Hitesh Saini • 6 months ago

deletion would be simpler to code if we need to search the element to be deleted ...

• Reply • Share ›

**erginbilgin** • 7 months ago

You have this in code:
/* If node to be deleted is head node */
if(*head_ref == del)
*head_ref = del->next;

But you need to have also this one after that:

/* If node to be deleted is head node */
if(*tail_ref == del)
*tail_ref = del->prev;

Otherwise it will work well until you try to iterate reverse.

• Reply • Share ›

**Preethi** • a year ago

Ques:- DataStructure with Insert O(1), Deletion O(1) Search O(1) and ReturnAnyElement O(1).
----> Are the doubly lineked list is used for the above purpose.
----> If so... Can you explain How?????

• Reply • Share ›

**Holden** Preethi • 6 months ago

It will be hash table

1 • Reply • Share ›

**coder** Preethi • 9 months ago

How can you search in a doubly linked list in O(1)???
So it is not that data structure which u r looking for.

1 • Reply • Share ›

**Mahesh** • a year ago

if(del->next != NULL)
del->next->prev = del->prev;

/* Change prev only if node to be deleted is NOT the first node */
if(del->prev != NULL)
del->prev->next = del->next;

This code was not executing

Giving an error like undefined symbol prev, next....

Is it possible to write
del->next->prev = del->prev;
del->prev->next = del->next;

• Reply • Share ›

**Alok Patel** • 2 years ago

Don't we need to update previous node pointer in case of deleting node is pointing at head ? I've used search and delete by value.

http://ideone.com/WDRwoJ

1 • Reply • Share ›

**ANA** Alok Patel • 2 years ago

yes we need to update the previous of node pointer in case of deleting head

3 • Reply • Share ›

**ronny** • 3 years ago

@geeksforgeeks @kartik @venki @sandeep @kartik

In case of deleting the last node, shouldn't the next of second last element be changed to NULL. Since it is still pointing the last element which is being freed. So isn't this a case of DANGLING POINTER.

Similarly for case of deleting the first node, shouldn't the prev of second element be made NULL for the above said reasons.

This can be done by removing the if conditions in the method.
Correct me if I am wrong.

• Reply • Share ›

**ronny** ronny • 3 years ago

@geeksforgeeks
Sorry for the previoius comment.
I misread the function.
So i apologize for any inconvenience hereby caused.
I shall be more careful before posting any comment in future.

6 • Reply • Share ›

**Holden** ronny • 6 months ago

you can simply delete your comment! :)

• Reply • Share ›

**ashatm** • 3 years ago

when we are making the linked list ourselves and then we need to delete a node, how do we obtain the pointer to the node that is to be deleted, i.e., how do we get *del?

• Reply • Share ›

**DS+Algo** ashatm • a year ago

this will depend on why u want to delete a node. When u implement it practically, u will get it somehow.

e.g: If u want to remove duplicate nodes, u compare the nodes while traversing and u find the duplicate node, i.e u get the pointer of the node to be deleted automatically

1 • Reply • Share ›

**Bala Mastanaiah Yadav** • 3 years ago

theory ple

• Reply • Share ›

**himank** • 3 years ago

i neeed algo of this .............

• Reply • Share ›

**sagar patni** • 4 years ago

```
#include"dll.h"
#include
#include
void remove1(node *q,dll *l);
void init(dll *l)
{
l->head=l->tail=NULL;
}

void append(dll *l ,int x)
{
node *p,*q;
p=(node*)malloc(sizeof(node));
p->val=x;
p->next=p->pre=NULL;
if(l->head==NULL)
{
l->head=l->tail=p;
```

see more

• Reply • Share ›

**sagar patni** • 4 years ago

```
 /* Paste your code here (You may delete these lines if not writing code) */#include"d
#include<stdio.h>
#include<stdlib.h>
void remove1(node *q,dll *l);
void init(dll *l)
{
        l->head=l->tail=NULL;
}

void append(dll *l ,int x)
{
        node *p,*q;
        p=(node*)malloc(sizeof(node));
        p->val=x;
        p->next=p->pre=NULL;
        if(l->head==NULL)
        {
                l->head=l->tail=p;
```

see more

1 • Reply • Share ›

**Mohamed** • 4 years ago

Great examples, nicely implemented thank you!

• Reply • Share ›

**Dreamer** • 6 years ago

You have not covered the case.. if node is to be deleted is the last node..

Also u could have bit more simplified in terms of cases.. if node to be deleted is
1 Head node
2 Last node
3 Else..

1 • Reply • Share ›

**Sandeep** ➦ Dreamer • 6 years ago

@Dreamer: I believe that case of last node is covered and demonstrated with an example.

Could you provide the simplified code that handles all cases?

• Reply • Share ›