Write a function to delete a Linked List

Algorithm: Iterate through the linked list and delete all the nodes one by one. Main point here is not to access next of the current pointer if current pointer is deleted.

Implementation:

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

/* Link list node */
struct node
{
    int data;
    struct node* next;
};

/* Function to delete the entire linked list */
void deleteList(struct node** head_ref)
{
   /* deref head_ref to get the real head */
   struct node* current = *head_ref;
   struct node* next;

   while (current != NULL)
   {
       next = current->next;
       free(current);
       current = next;
   }

   /* deref head_ref to affect the real head back
      in the caller. */
   *head_ref = NULL;
```


```
}

/* Given a reference (pointer to pointer) to the head
  of a list and an int, push a new node on the front
  of the 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;

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

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

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

    /* Use push() to construct below list
     1->12->1->4->1  */
    push(&head, 1);
    push(&head, 4);
    push(&head, 1);
    push(&head, 12);
    push(&head, 1);

    printf("\n Deleting linked list");
    deleteList(&head);

    printf("\n Linked list deleted");
    getchar();
}
```

Time Complexity: O(n)
Space Complexity: O(1)

Popular Posts

Related Tpoics:

- Given a linked list, reverse alternate nodes and append at the end
- Pairwise swap elements of a given linked list by changing links
- Self Organizing List | Set 1 (Introduction)
- Merge a linked list into another linked list at alternate positions
- QuickSort on Singly Linked List
- Delete N nodes after M nodes of a linked list
- Design a stack with operations on middle element
- Swap Kth node from beginning with Kth node from end in a Linked List

Writing code in comment? Please use **ideone.com** and share the link here.

18 Comments GeeksforGeeks

Sort by Newest

Join the discussion…

Himanshu Dagar • 3 months ago

http://ideone.com/7RkLNM

can refer to this

• Reply • Share ›

mahesh • 5 months ago

Recursive solution for the same problem.

```
void deleteList(struct node **head)
{
    if(*head)
       {
            deleteList(&(*head)->next);
            free(*head);
            *head = NULL;
       }
}
```

• Reply • Share ›

jayasurya j ➚ mahesh • 3 months ago

i have a doubt! can u explain me *head = NULL ... does this mean evel

• Reply • Share ›

anon • 7 months ago

I haven't understood the last line in the deleteList function.When I run the funct and call a function to find the length of the list it gives the correct length.

705

Subscribe

Recent Comments

Abhi You live US or India?

Google (Mountain View) interview · 47 minutes ago

Aman Hi, Why arent we checking for conditions...

Write a C program to Delete a Tree. · 1 hour ago

kzs please provide solution for the problem...

Backtracking | Set 2 (Rat in a Maze) · 1 hour ago

Sanjay Agarwal bool tree::Root_to_leaf_path_given_sum(tree...

Root to leaf path sum equal to a given number · 1 hour ago

GOPI GOPINATH @admin Highlight this sentence "We can easily...

Count trailing zeroes in factorial of a number · 1 hour ago

newCoder3006 If the array contains negative numbers also. We...

Find subarray with given sum · 2 hours ago

So my question is what happens if we do not write the last line in deleteList an

asunel • 7 months ago

@GeeksforGeeks: Is it possible to delete a linked list having a loop without ren

nitin • a year ago

```
 #include<stdio.h>
#include<malloc.h>
struct node
{
    int data;
    struct node * link;
};
void insert1(struct node **p,int data)
{
    struct node *temp,*t;
    temp=(struct node *)malloc(sizeof(struct node));
    temp->data=data;
    temp->link=NULL;
    if((*p)==NULL)
    {
        *p=temp;
    }
    else
```

see more

Ankit Malhotra • a year ago

A simple tail recursive C++ function instead of head recursion examples seen

passes to next.

```
void rdellist (node * &ptr) {
 if (!ptr) return;
 node * temp = ptr->next;
 delete ptr;
 rdellist (ptr = temp);
}
```

Recursion is best avoided with loops where feasible so

```

```

void dellist (node * &ptr) {
node * temp;
while (ptr) {
temp = ptr->next;
delete ptr;
ptr = temp;
}
}

 • Reply • Share ›

Harsh Agarwal • a year ago

A recursive code:

```
void delete_list(node1 *start)
{
    if(start==NULL)
    {
                printf("Empty linked list...\n");
```

```
                return;
    }
    node1 *p=start;
    if(p->link==NULL)
    free(p);
    else
    delete_list(p->link);
    p->link=NULL;
}
```

• Reply • Share ›

Ankit Malhotra Harsh Agarwal • a year ago

Segmentation Fault. After free(p) reference to p->link is invalid.

```
 /* Paste your code here (You may delete these lines if not wri
```

1 • Reply • Share ›

abhishek08aug Ankit Malhotra • a year ago

Corrected version :)

```
 /* Function to delete the entire linked list */
void deleteList(struct node** head_ref)
{
  if(*head_ref==NULL) {
    return;
  } else {
    deleteList(&((* head_ref)->next));
    free(*head_ref);
  }
}
```

1 ^ | v • Reply • Share ›

neelabhsingh ➦ abhishek08aug • a year ago

In recursive time complexity will be O(n) but space com
O(n).
Abhisek there is some improvement in your code. If you
infinite. In if condition is only for to reach last node then
reaching start node you free. But in your function you pa
pointer. But you did not make it NULL. If i am wrong plz

```
NODE * deleteList(NODE **start)
{
if(*start==NULL)
{
printf("Now head is NULL");
return NULL;
}
else
{
deleteList(&((*start)->next));
free(*start);
return NULL;
}
}
```

^ | v • Reply • Share ›

Sunil • a year ago

A recursive algorithm to delete the linked list :)

```
 void delete_list(struct node** node)
{
    if(NULL == *node)
```

```
        return;
    delete_list((*node)->link);
    free(*node);
}
```

• Reply • Share ›

Gupta ➚ Sunil • a year ago

Wrong code.. U just need 2 pass d adress. make it simple..
void deleteNode(struct node* head)
{
if(head==NULL)
return;
else
{
delete(head->next);
free(head);
}
}

```
/* Paste your code here (You may delete these lines if not wri
```

• Reply • Share ›

neelabhsingh ➚ Sunil • a year ago

In recursive time complexity will be O(n) but space complexity will be ir

```
/* Paste your code here (You may delete these lines if not wri
```

• Reply • Share ›

abhishek08aug ➚ Sunil • a year ago

Wrong code Sunil. you should change

delete_list((*node)->link);

to

delete_list(&((*node)->link))

Check my code above.

kpnigalye • 2 years ago

[sourcecode language="C++"]
/* Delete a linked list */
void DeleteList(struct LinkedListnode*& head_ref)
{
struct LinkedListnode* current = head_ref;
if(head_ref == NULL)
return;
while(current!=NULL)
{
struct LinkedListnode* temp = current;
current = current->next;
cout<<"Node deleted is: "<<temp->data<<endl;
delete temp;
}
head_ref = NULL;
}

/* Recursive way to delete a linked list */

see more

amitp49 • 2 years ago

Recursive solution for the same can be...

```
 /* Function to delete the entire linked list */
void deleteList(struct node** head_ref)
{
        if(head_ref==NULL)
                return;
        if((*head_ref)->next)
                deleteList(&(*head_ref)->next);
        free(*head_ref);
        *head_ref = NULL;
}
```

 • Reply • Share ›

Yogendra Singh Vimal amitp49 • 9 months ago

i think @amitp49 that ur code is a Li'L bit wrong...u Should write if(*hea

instead of,

if(head_ref==NULL) ,

as here double pointer " head_ref " receives the address of a single po the address of the single pointer but here our intention is towards the a which can be achieved by writing *head_ref.

 • Reply • Share ›

 Subscribe Add Disqus to your site