GeeksforGeeks

A computer science portal for geeks

Login

Home	Algorithms	DS	GATE	Intervi	ew Corner	Q&A	С	C++	Java	Books	Contribute	Ask a Q	About
Array	Bit Magic	C/C+	+ Arti	cles	GFacts	Linked L	ist	MCQ	Misc	Output	t String	Tree	Graph

How to write C functions that modify head pointer of a Linked List?

Consider simple representation (without any dummy node) of Linked List. Functions that operate on such Linked lists can be divided in two categories:

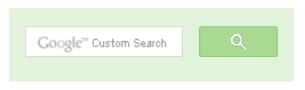
1) Functions that do not modify the head pointer: Examples of such functions include, printing a linked list, updating data members of nodes like adding given a value to all nodes, or some other operation which access/update data of nodes

It is generally easy to decide prototype of functions of this category. We can always pass head pointer as an argument and traverse/update the list. For example, the following function that adds x to data members of all nodes.

```
void addXtoList(struct node *node, int x)
    while (node != NULL)
        node->data = node->data + x;
        node = node->next;
```

2) Functions that modify the head pointer: Examples include, inserting a node at the beginning (head pointer is always modified in this function), inserting a node at the end (head pointer is modified only when the first node is being inserted), deleting a given node (head pointer is modified when the deleted node is first node). There may be different ways to update the head pointer in these functions. Let us discuss these ways using following simple problem:

"Given a linked list, write a function deleteFirst() that deletes the first node of a given linked list. For example, if the list is 1->2->3->4, then it should be modified to 2->3->4"





53,527 people like GeeksforGeeks.











ntory/ov/	Lynariance
	Experiences
	_, , , , , , , , , , , , ,

Advanced Data Structures

Dynamic Programming

Greedy Algorithms

Backtracking

Pattern Searching

Divide & Conquer

Mathematical Algorithms

Recursion

Geometric Algorithms

Algorithm to solve the problem is a simple 3 step process: (a) Store the head pointer (b) change the head pointer to point to next node (c) delete the previous head node.

Following are different ways to update head pointer in deleteFirst() so that the list is updated everywhere.

2.1) Make head pointer global: We can make the head pointer global so that it can be accessed and updated in our function. Following is C code that uses global head pointer.

```
// global head pointer
struct node *head = NULL;
// function to delete first node: uses approach 2.1
// See http://ideone.com/ClfQB for complete program and output
void deleteFirst()
    if (head != NULL)
       // store the old value of head pointer
       struct node *temp = head;
       // Change head pointer to point to next node
       head = head->next;
       // delete memory allocated for the previous head node
       free (temp);
```

See this for complete running program that uses above function.

This is not a recommended way as it has many problems like following:

- a) head is globally accessible, so it can be modified anywhere in your project and may lead to unpredictable results.
- b) If there are multiple linked lists, then multiple global head pointers with different names are needed.

See this to know all reasons why should we avoid global variables in our projects.

2.2) Return head pointer: We can write deleteFirst() in such a way that it returns the modified



Popular Posts

All permutations of a given string

Memory Layout of C Programs

Understanding "extern" keyword in C

Median of two sorted arrays

Tree traversal without recursion and without stack!

Structure Member Alignment, Padding and

Data Packing

Intersection point of two Linked Lists

Lowest Common Ancestor in a BST.

Check if a binary tree is BST or not

Sorted Linked List to Balanced BST

nead pointer. Whoever is using this function, have to use the returned value to update the nead node.

```
// function to delete first node: uses approach 2.2
// See http://ideone.com/P5oLe for complete program and output
struct node *deleteFirst(struct node *head)
    if (head != NULL)
       // store the old value of head pointer
       struct node *temp = head;
       // Change head pointer to point to next node
       head = head->next:
       // delete memory allocated for the previous head node
       free (temp);
    return head;
```

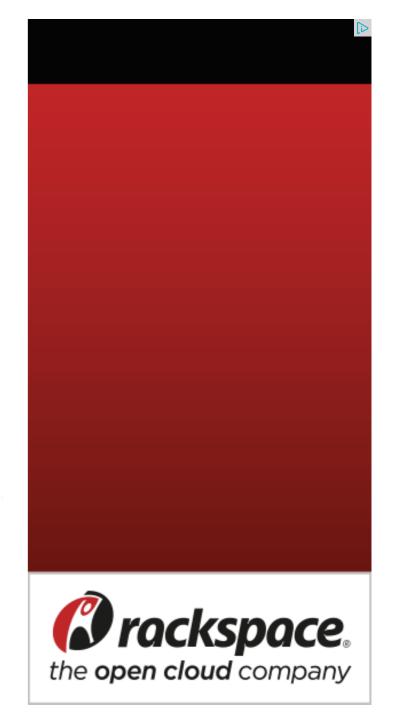
See this for complete program and output.

This approach is much better than the previous 1. There is only one issue with this, if user misses to assign the returned value to head, then things become messy. C/C++ compilers allows to call a function without assigning the returned value.

```
head = deleteFirst(head); // proper use of deleteFirst()
deleteFirst(head); // improper use of deleteFirst(), allowed by compi
```

2.3) Use Double Pointer: This approach follows the simple C rule: if you want to modify local variable of one function inside another function, pass pointer to that variable. So we can pass pointer to the head pointer to modify the head pointer in our deleteFirst() function.

```
// function to delete first node: uses approach 2.3
// See http://ideone.com/9GwTb for complete program and output
void deleteFirst(struct node **head ref)
    if(*head ref != NULL)
       // store the old value of pointer to head pointer
       struct node *temp = *head ref;
       // Change head pointer to point to next node
```



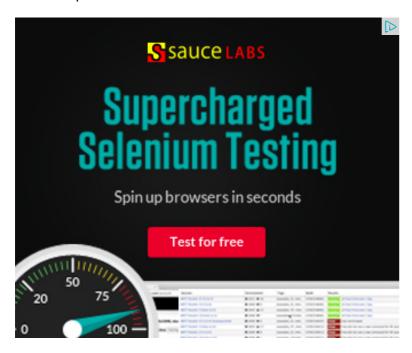


```
*head ref = (*head ref)->next;
// delete memory allocated for the previous head node
free (temp);
```

See this for complete program and output.

This approach seems to be the best among all three as there are less chances of having problems.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.



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





Recent Comments

Abhi You live US or India?

Google (Mountain View) interview · 43 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

AdChoices D

- ► Linked List
- ▶ Java Source Code
- ► Linked Data

AdChoices D

► Pointers Pointer

- 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.

16 Comments

GeeksforGeeks

Sort by Newest ▼



Join the discussion...



Anil · 8 days ago ideone link is broken.. kindly FIX it!!!



VeridisQuo • a month ago ideone links are broken...please fix!! 2 A Reply · Share >



srikanth • 2 months ago void deleteFirst(struct node **head ref) if(*head ref!= NULL) // store the old value of pointer to head pointer struct node *temp = *head ref; // Change head pointer to point to next node *head ref = (*head ref)->next;

▶ Void Pointer

AdChoices [>

- ▶ One Pointer
- ► Pointer Free
- ► Pointer X

```
// delete memory allocated for the previous nead hode
free(temp);
}}
In this 3rd approach of deleting first node, the above code can be modified as:
void deleteFirst(struct node **head_ref)
if(*head ref!= NULL)
// Change head pointer to point to next node
*head ref = (*head ref)->next;
why to allocate temp and free it again!!!!
Srikanth Marepalli → srikanth • 2 months ago
       yeah!!! as this is for deleting the head node only, we can modify as about
       yash ⋅ 2 months ago
Function to add a node at the end doesn't need to take the super pointer to the
head pointer.
struct node{
int value;
struct node* next;
}node;
```

void append(struct node *neadPointer,int newData){

struct node * nodeToadd=(struct node*) malloc(sizeof(struct node));

struct node * currentNodePointer=headPointer;

while (currentNodePointer->next!=NULL) {

currentNodePointer=currentNodePointer->next;

see more



Anupam Gupta • a year ago how to create image.



Aditya Ambashtha • a year ago

first node can be deleted inside a function WITHOUT passing it the superpoint

Algo

- 1)Copy value in node 2 to node 1..
- 2)Link node 1 to node 3
- 3)Delete node 2



pefullarton → Aditya Ambashtha • a year ago

Moving the data b/w different nodes in not a good approach. It should a



sk007 · 2 years ago

What do you do in Java when you want to use the double pointer approach? In passed by value. So what can be done to achieve the same effect as that of a

 $/^{\star}$ Paste your code here (You may delete these lines if not writing code



kartik → sk007 · 2 years ago

In Java, create a class linkedListNode and another class linkedList. The to head (of linkedListNode type).

When you pass a linked list, you pass reference of object of type linked modify head reference.



sk007 → kartik · 2 years ago

Do you mean that the head should be made as a static membe code would really help.

Thanks.



kartik → sk007 · 2 years ago

@sk007: I didn't mean static. I think following post and c

http://www.geeksforgeeks.org/a...

http://www.geeksforgeeks.org/a...

Let me know if this clarifies



inalapavan • 2 years ago super



jogi · 2 years ago

You can always use dummy nodes to avoid double pointer and if conditions th



suhan • 2 years ago

Awesome! how about doubly linked list?



kartik → suhan · 2 years ago

Same concepts works for doubly linked lists also.



Subscribe



Add Disgus to your site

@geeksforgeeks, Some rights reserved

Contact Us!

Powered by WordPress & MooTools, customized by geeksforgeeks team