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;
    }
}
```

Run on IDE

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"

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);
    }
}
```

Run on IDE

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 head pointer. Whoever is using this function, have to use the returned value to update the head 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;
}
```

Run on IDE

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 compiler
```

Run on IDE

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);
    }
}
```

Run on IDE

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.

60 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)

1.8 Average Difficulty : **1.8/5.0**
Based on **9** vote(s)

Add to TODO List
Mark as DONE

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

rahul singh • 11 days ago

how does the third method of double pointers work??can anyone explain please

1 • Reply • Share ›

Gaurav Kishore • 14 days ago

Another Way is to modify the head pointer is "Passing by Reference". This helps in avoiding the handling of double pointers.
Sample:

```
void DeleteFirst(node* &head)
{
node* current = head;
if(current==NULL){
cout<<"The LinkList is empty.Nothing to delete\n";
}else{
head=head->next;
free(current);
}
}
```

And Calling the function as
DeleteFirst(head);

• Reply • Share ›

Nitin Pandey • 6 months ago

http://codepad.org/CJMY8wMc

This code is updating the values and reflecting the changes made by function into the main method. Can anyone explain?

• Reply • Share ›

Siddesh ➚ Nitin Pandey • 6 months ago

Yes. Understand that you have done changes in data part of node and not node itself (so your Functions do not modify any pointer) .Both pointer are still pointer ie one in main andaddXtoList function to same head node. So changes will be reflect . You can visualize both ptr as nothing but reference in c++ or alias

• Reply • Share ›

Gautham Kumaran • 7 months ago

Is there a java equivalent for the double pointer (method 3) ?? I always return the head in java.

• Reply • Share ›

GLOBAL • 7 months ago

4
3
2
1
4 are the no.of nodesafter deleting
0
3
2
1

but 0 is still showing why

• Reply • Share ›

Ankit Singh • a year ago

ideone links are not running.

4 • Reply • Share ›

Shubhendu Kumar • a year ago

In Sec. 2.2, where we are returning the head pointer , we have used struct instead of a function deleteFirst. Please have a look at it.

1 • Reply • Share ›

devakar verma ➚ Shubhendu Kumar • 10 months ago

it's correct , as the return type of function deleteFirst is pointer to a node and the function is also returning head which is pointer to first node of list.

• Reply • Share ›

Abhishek Sharma • a year ago

Why do we need these two lines:

struct node *temp = *head_ref;
free(temp);

I tried without these two and the program worked.

Also in the line between these two, we do the meet of the program and we dont even use the temp variable in that.

2 • Reply • Share ›

bala ➚ Abhishek Sharma • a year ago

mem leakage

2 • Reply • Share ›

Ayushi Gupta ➚ Abhishek Sharma • a year ago

The program will work but by removing these lines, you are not releasing the memory of the deleted node which was assigned by malloc() . The memory is still occupied, but is not in use, thus, program becomes memory inefficient.

3 ^ | v • Reply • Share ›

Abhishek Sharma ➚ Ayushi Gupta • a year ago

apologies for not being clear enough. but the main question was that why do we even need to create the temp variable in the first place when we are not even using it at any later point.

2 ^ | v • Reply • Share ›

DS+Algo ➚ Abhishek Sharma • 7 months ago

who said we don't use that temp variable.. see we used it in:
free(temp);
if we didn't save the old head pointer in temp variable, then there would be no way to access the old head to free it because head pointer gets changed.

2 ^ | v • Reply • Share ›

fadoo ➚ DS+Algo • 7 months ago

gandu.....hai tu

^ | v • Reply • Share ›

Swapnil Tiwari ➚ Abhishek Sharma • a year ago

If you don't use those two lines, the program will work just fine but the memory allocated to the first node won't be deallocated which is a bad programming practice.

3 ^ | v • Reply • Share ›

Palash • a year ago

Thanks Geeks for this post. Was really confused in these methods.

1 ^ | v • Reply • Share ›

Aditya Nalluri • a year ago

Hey, those ideone.com links arent opening. Please look into the issue.

Thanks.

3 ^ | v • Reply • Share ›

Aditya • a year ago

Hey I am unable to

^ | v • Reply • Share ›

coderr • a year ago

why is 3rd a better option than 2nd..? i mean wat exactly is the difference bw 2nd n 3rd..as in what change occured wen we created a pointer wich points to head?

1 • Reply • Share ›

Adauta Garcia Ariel coderr • a year ago

The to understand why 2nd is not a good approach, let's assume the next. Guess head's content has 0x22 in the main method, so the next is not actually C code.

```
/*head receive the memory address*/

 struct node* deleteFirst(headCopy = 0x22){

   //create tmp

   headCopy = headCopy->next; headCopy's content 0x26

  //free tmp

}
```

In case you miss catch the headCopy content in main(). how would know that the new head is 0x26.

In 3rd options is better, because you don't worry about it, you know that when deleteFirts ends, your head pointer will be updated.

3 • Reply • Share ›

Rehan Adauta Garcia Ariel • a year ago

How is head updated automatically in 3rd case?

Actually I am not able to understand simple C rule: if you want to modify local variable of one function inside another function, pass pointer to that variable.

Since program link is not opening tell what will we pass in function in 3rd case.

1 • Reply • Share ›

Adauta Garcia Ariel Rehan • a year ago

Hello Rehan, please read me and follow me.

One step a time.
First you know that in main you have a pointer to node called head, something like:

```
struct node* head = NULL;
/***********************
In memory concetually is a box with content NULL
----------
| NULL |
----------
 0X20
***********************/
```

But like any other variable head also has a memory address. Recall the concept of pointer and is just a **variable** that store the address

see more

3 • Reply • Share ›

Nelson Fernando ➚ Adauta Garcia Ariel • 3 months ago
awesome explanation...
• Reply • Share ›

sagar allamdas ➚ coderr • a year ago
Because you do not need to return value of head in the program since it is using pass by reference in which value at that specific address will be modified.whereas as in the 2 nd method value is not modified so we have to return it
1 • Reply • Share ›

Neil • a year ago
Thanks a lot for this post! I was really confused when to use which approach. A note may be very important :
1. WHEN YOUR FUNCTION USES RECURSION, MOST OF THE TIME IT IS CONVENIENT TO USE THE SECOND APPROACH.
2. OTHERWISE USE THE THIRD.
5 • Reply • Share ›

DS+Algo • a year ago
And here I got the point to use double pointer but the only thing I am still confused of is " C/C++ compilers allows to call a function without assigning the returned value.". Does this mean Java and any other languages do not allow to call a function without assigning the

returned value?'?'?

1 • Reply • Share ›

Kim Jong-il ➔ DS+Algo • a year ago

Java does not allow, Because it follows "Strict type checking."
Hope you get this.

2 • Reply • Share ›

Yamini yadav • 2 years ago

I use

void deleteFirst(struct node * &head_ref)
This simplifies the code and works like a charm

3 • Reply • Share ›

Ishan Gupta ➔ Yamini yadav • 7 months ago

No, you just nullified the use of pointers in your code.

• Reply • Share ›

Kim Jong-il ➔ Yamini yadav • a year ago

I am not getting whats going in this.? Because *and & are inverse of each other they will cancel each other and that become "struct node head_ref".

is not it??

5 • Reply • Share ›

Sanyam Garg ➔ Yamini yadav • a year ago

Can you explain in detail please

1 • Reply • Share ›

karthik ➔ Sanyam Garg • a year ago

&i=p; (address of variable i is stored in p)

i=*p; (value in i can be accessed through p)
so when & and i are used together ,they cancel out i.e (*&i = i)

1 • Reply • Share ›

Pankaj Kushwaha ➔ karthik • 6 months ago

its not like that karthik , he is using concept of refrence , which is only supported in c++ , not in c , usually use of refrence make code more readable and elegant , you can follow following links

;

http://stackoverflow.com/quest

http://stackoverflow.com/quest...

http://markgodwin.blogspot.in/...

http://www.functionx.com/cppcl...

• Reply • Share ›

Ankit Singh • 2 years ago

what the statement struct node *deleteFirst(struct node *head) means?
anyone?

• Reply • Share ›

VeridisQuo ➚ Ankit Singh • 2 years ago

It defines a deleteFirst() function which takes struct node* head as input and returns struct node* (pointer to a struct node)

3 • Reply • Share ›

Ashish Patil • 2 years ago

ideone links are not working, plz fix it,

4 • Reply • Share ›

Anil • 2 years ago

ideone link is broken.. kindly FIX it!!!

5 • Reply • Share ›

VeridisQuo • 2 years ago

ideone links are broken...please fix!!

5 • Reply • Share ›

srikanth • 2 years 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;

// delete memory allocated for the previous head node
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!!!!

1 • Reply • Share ›

Kim Jong-il ➚ srikanth • a year ago

you need to read Memory leak probelem.

• Reply • Share ›

Guest00 ➚ Kim Jong-il • a year ago

He actually means, that the author is not even using the "temp" variable.

```
struct node *temp = *head_ref;
// Change head pointer to point to next node
*head_ref = (*head_ref)->next;
free(temp);
```

temp uses head_ref
but head_ref (which is what we are really working with) is not related with "temp".

So it's unnecessary.

• Reply • Share ›

Ashish Miglani ➚ srikanth • 2 years ago

Free(temp) will remove the node from the memory also otherwise it would needlessly consume memory.

• Reply • Share ›

Ankit Singh ➚ srikanth • 2 years ago

it just passes the head pointer to the next node.
it does not deletes the first node.

• Reply • Share ›

anon ➚ srikanth • 2 years ago

Here it doesn't matter as it is a concern of few byte, but in the long run you will

have to pay for this in terms of memory leaks, so freeing unused memory is a good habit that must be developed.

• Reply • Share ›

Srikanth Marepalli srikanth • 2 years ago

yeah!!! as this is for deleting the head node only, we can modify as above, huh!!

• Reply • Share ›

yash • 2 years ago

Function to add a node at the end doesn't need to take the super pointer to the head as input. We can just pass the head pointer.

```
struct node{

int value;

struct node* next;

}node;

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

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

struct node * currentNodePointer=headPointer;

while (currentNodePointer->next!=NULL) {

currentNodePointer=currentNodePointer->next;
```

see more

• Reply • Share ›

Anupam Gupta • 3 years ago

how to create image.

• Reply • Share ›

Aditya Ambashtha • 3 years ago

first node can be deleted inside a function WITHOUT passing it the superpointer(node **)

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

1 • Reply • Share ›

pefullarton ➚ Aditya Ambashtha • 3 years ago

Moving the data b/w different nodes in not a good approach. It should always be avoided until necessary.

4 ^ | v • Reply • Share ›

Load more comments