

**DOPE** (/blog/)

Deletion of a given node from a linked list in C

🕒 May 25, 2017 📌 C (/blog/tag/c/?tag=c) LINKED LIST (/blog/tag/linked-list/?tag=linked-list) DATA STRUCTURE (/blog/tag/data-structure/?tag=data-structure) 👁 19474



Become an Author

(/blog/submit-article/)

Download Our App.



(<https://play.google.com/store/apps/details?id=com.blogsdope&pcampaignid=MKT-Other-global-all-co-prtnr-py-PartBadge-Mar2515-1>)

Previous:

1. Linked lists in C (Singly linked list)

(<https://www.codesdope.com/blog/article/linked-lists-in-c-singly-linked-list/>)

2. Linked list traversal using while loop and recursion

(<https://www.codesdope.com/blog/article/linked-list-traversal-using-while-loop-and-recursi/>)

3. Concatenating two linked lists in C

(<https://www.codesdope.com/blog/article/concatenating-two-linked-lists-in-c/>)

4. Inserting a new node in a linked list in C

(<https://www.codesdope.com/blog/article/inserting-a-new-node-in-a-linked-list-in-c/>)

Make sure that you are familiar with the concepts explained in the article(s) mentioned above before proceeding further.

We will proceed further by taking the linked list we made in the previous article.

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

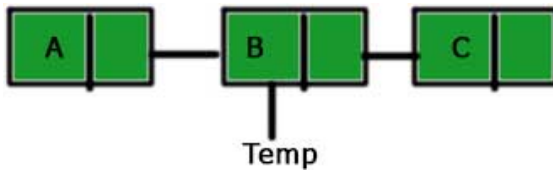
struct node
{
    int data;
    struct node *next;
};

int main()
{
    struct node *prev, *head, *p;
    int n, i;
    printf ("number of elements:");
    scanf ("%d", &n);
    head=NULL;
    for(i=0; i<n; i++)
    {
        p=malloc(sizeof(struct node));
        scanf ("%d", &p->data);
        p->next=NULL;
        if(head==NULL)
            head=p;
        else
            prev->next=p;
        prev=p;
    }
    return 0;
}
```

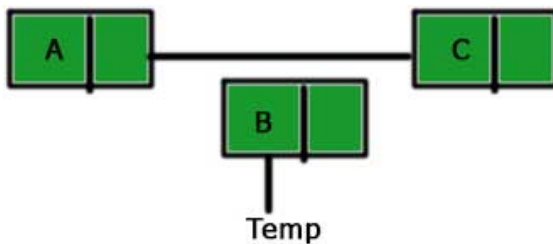
We delete any node of a linked list by connecting the predecessor node of the node to be deleted by the successor node of the same node. For example, if we have a linked list $a \rightarrow b \rightarrow c$, then to delete the node 'b', we will connect 'a' to 'c' i.e., $a \rightarrow c$. But this will make the node 'b' inaccessible and this type of inaccessible nodes are called garbage and we need to clean this garbage. We do this cleaning by the use of 'free' function. If you are not familiar with the 'free'

function then you can visit the 'Dynamic memory' chapter of the C course (<https://www.codesdope.com/c-dynamic-memory/>). So, the steps to be followed for deletion of the node 'B' from the linked list $A \rightarrow B \rightarrow C$ are as follows:

1. Create a temporary pointer to the node 'B'.



2. Connect node 'A' to 'B'.



3. Free the node 'B'.



The code representing the above steps is:

```
del (struct node *before_del)
{
    struct node *temp;
    temp = before_del->next;
    before_del->next = temp->next;
    free(temp);
}
```

Here, 'before_node' is the predecessor of the node to be deleted.

`temp = before_del->next` – We are making a temporary pointer to the node to be deleted.

`before_del->next = temp->next` – Connecting the predecessor of the node to be deleted with the successor of the node to be deleted.

`free(temp)` – Making the 'temp' free.

And the overall code is:

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

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

display(struct node *head)
{
    if(head == NULL)
    {
        printf("NULL\n");
    }
    else
    {
        printf("%d\n", head -> data);
        display(head->next);
    }
}


del (struct node *before_del)
{
    struct node *temp;
    temp = before_del->next;
    before_del->next = temp->next;
    free(temp);
}


int main()
{
    struct node *prev,*head, *p;
    int n,i;
    printf ("number of elements:");
    scanf ("%d",&n);
    head=NULL;
    for(i=0;i<n;i++)
    {
        p=malloc(sizeof(struct node));
        scanf ("%d",&p->data);
        p->next=NULL;
        if(head==NULL)
            head=p;
        else
            prev->next=p;
        prev=p;
    }
    /*node to be deleted is head->next->next*/
    del(head->next);
    display(head);
    return 0;
}
```


Next:

1. Array vs Linked list in C (<https://www.codesdope.com/blog/article/array-vs-linked-list-in-c/>)

Liked the post?

 (<https://www.facebook.com/sharer/sharer.php?u=https://www.codesdope.com/blog/article/deletion-of-a-give-node-from-a-linked-list-in-c/>)

 ([https://twitter.com/intent/tweet?url=https://www.codesdope.com/blog/article/deletion-of-a-give-node-from-a-linked-list-in-c/&text=Deletion of a given node from a linked list in C &via=codesdope](https://twitter.com/intent/tweet?url=https://www.codesdope.com/blog/article/deletion-of-a-give-node-from-a-linked-list-in-c/&text=Deletion%20of%20a%20given%20node%20from%20a%20linked%20list%20in%20C%20&via=codesdope))

 (<https://plus.google.com/share?url=https://www.codesdope.com/blog/article/deletion-of-a-give-node-from-a-linked-list-in-c/>)

 ([https://www.linkedin.com/shareArticle?url=https://www.codesdope.com/blog/article/deletion-of-a-give-node-from-a-linked-list-in-c/&title=Deletion of a given node from a linked list in C](https://www.linkedin.com/shareArticle?url=https://www.codesdope.com/blog/article/deletion-of-a-give-node-from-a-linked-list-in-c/&title=Deletion%20of%20a%20given%20node%20from%20a%20linked%20list%20in%20C))

 ([https://pinterest.com/pin/create/bookmarklet/?media=https://www.codesdope.com/media/blog_images/1/2017/6/1/icon-1728548_640.jpg&url=https://www.codesdope.com/blog/article/deletion-of-a-give-node-from-a-linked-list-in-c/&description=Deletion of a given node from a linked list in C](https://pinterest.com/pin/create/bookmarklet/?media=https://www.codesdope.com/media/blog_images/1/2017/6/1/icon-1728548_640.jpg&url=https://www.codesdope.com/blog/article/deletion-of-a-give-node-from-a-linked-list-in-c/&description=Deletion%20of%20a%20given%20node%20from%20a%20linked%20list%20in%20C))

Amit Kumar (/blog/author/54322/?author=54322)

Developer and founder of CodesDope.

 (<https://www.facebook.com/codesdope>)  (<https://www.twitter.com/codesdope>)  (<https://www.linkedin.com/in/amit-kumar-66903395>)