Deletion in Linked List | Linked List Operations

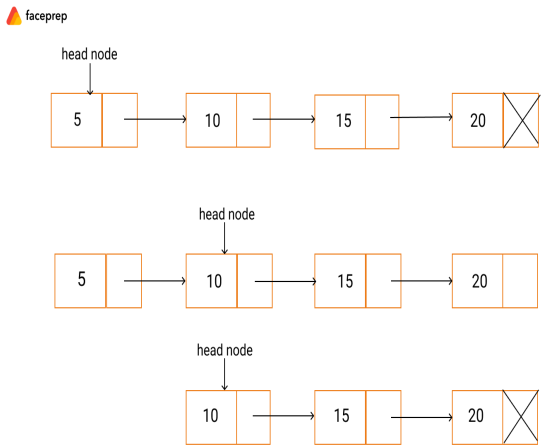
Published on 07 Mar 2020

Deletion in a linked list (../linked-list-introduction/) can happen at various places in a list. A few cases are:

- At the beginning of the linked list.
- At the end of the linked list.
- At a given position in the linked list.

Now, let us look at a program for each of these cases.

Deletion at the beginning of the linked list



- To delete the first node (head node), copy the head node in a temporary node.
- Make the second node as the head node.
- Now, delete the temporary node.

Link copied to clipboard. Share away!

Program to delete a node at the beginning of the Linked List

Login (https://www.faceprep.in/login/?rurl=/data-structures/linked-list-deleting-a-node/)

Dismiss

FACE Prep (https://www.faceprep.in/)

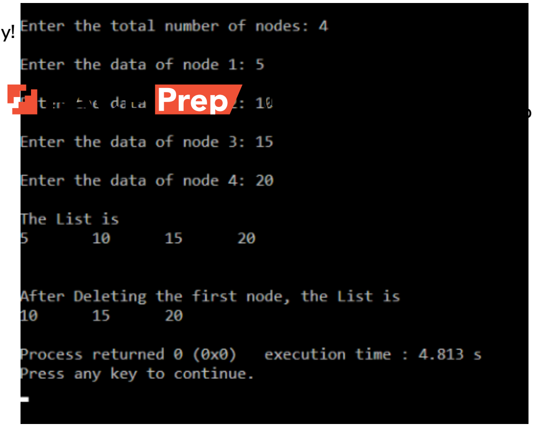
Explore

Wall of Love (https://www.faceprep.in/reviews/)

Sign Up (https://www.faceprep.in/signup/?rurl=/data-structures/linked-list-deleting-a-node/)

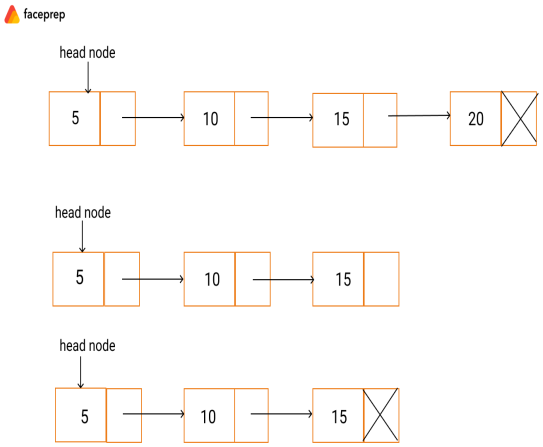
```
#include <stdio.h>
#include <stdlib.h>

// Creating node with data and a pointer
struct node {
 int data;
 struct node *next;
}*head;
void createList(int n);
void deletion_beginning();
void displayList();

int main()
{
 int n, data, pos;
 printf("\nEnter the total number of nodes: ");
 scanf("%d", &n);
 if(n == 0)
 {
 printf("Empty List\n");
 exit(0);
 }
 else
 {
 createList(n);
 }
printf("\nThe List is \n");
 displayList();
 deletion_beginning();
 printf("\n\nAfter Deleting the first node, the List is\n");
 displayList();
 return 0;
}

void createList(int n)
{
 struct node *newNode, *temp;
 int data, i;
head = (struct node *)malloc(sizeof(struct node));
 // When the list is empty
 if(head == NULL)
 {
 printf("Unable to allocate memory.");
 }
 else
 {
 printf("\nEnter the data of node 1: ");
 scanf("%d", &data);
 head->data = data;
 head->next = NULL;
 temp = head;
 for(i=2; i<=n; i++)
 {
 newNode = (struct node *)malloc(sizeof(struct node));
 if(newNode == NULL)
 {
 printf("Unable to allocate memory.");
 break;
 }
 else
 {
 printf("\nEnter the data of node %d: ", i);
 scanf("%d", &data);
 newNode->data = data;
 newNode->next = NULL;
 temp->next = newNode;
 temp = temp->next;
 }
 }
 }
}

/* Function to delete the first node */
void deletion_beginning()
{
 // Empty List
 if(head == NULL)
 printf("\n The list is Empty\n");
 struct node *temp;
 temp = head; //Make temp as head node
 head = head -> next; // Shift the head node
 free(temp); // Delete the temporary node
}
void displayList()
{
 struct node *temp;
 if(head == NULL)
 {
 printf("List is empty.");
 }
 else
 {
 temp = head;
// Print the list
 while(temp != NULL)
 {
 printf("%d\t", temp->data);
 temp = temp->next;
 }
 printf("\n");
 }
}
```

OUTPUT:



```
Enter the total number of nodes: 4

Enter the data of node 1: 5

Enter the data of node 2: 10

Enter the data of node 3: 15

Enter the data of node 4: 20

The List is
5       10      15      20


After Deleting the first node, the List is
10      15      20

Process returned 0 (0x0)   execution time : 4.813 s
Press any key to continue.
```

Deletion at the end of the linked list

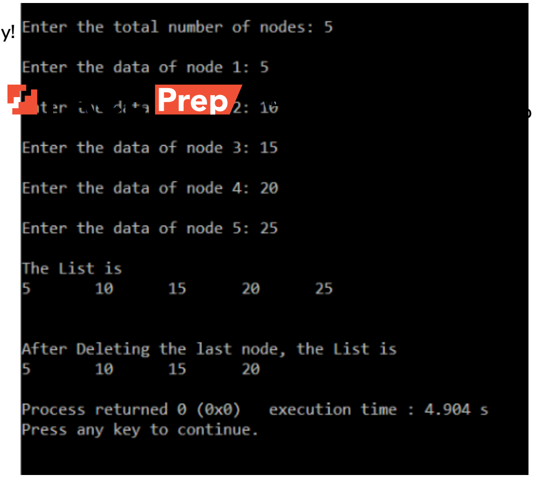


- To delete the last node, start traversing the list from the head node and continue traversing until the address part of the node is **NULL**.
- Keep track of the **second last node** in some temporary variable say **prev_node**.
- Once the address part of the node is NULL, set the **address part** of the **prev_node** as **NULL** and then delete the last node.

Program to delete a node at the end of the Linked List

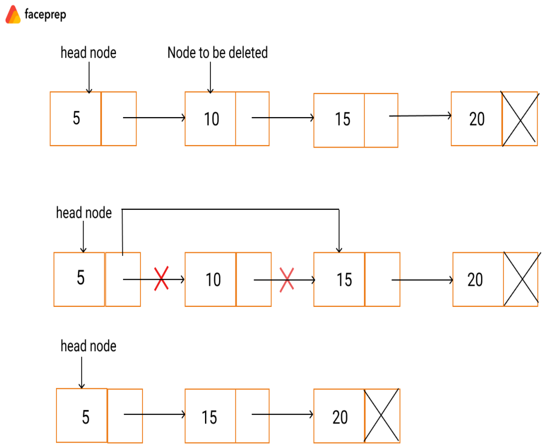
```
#include <stdio.h>
#include <stdlib.h>

// Creating node with data and a pointer
struct node {
int data;
struct node *next;
}*head;
void createList(int n);
void deletion_end();
void displayList();

int main()
{
int n, data, pos;
printf("\nEnter the total number of nodes: ");
scanf("%d", &n);
createList(n);
printf("\nThe List is \n");
displayList();
deletion_end();
printf("\n\nAfter Deleting the last node, the List is\n");
displayList();
return 0;
}

void createList(int n)
{
struct node *newNode, *temp;
int data, i;
head = (struct node *)malloc(sizeof(struct node));
// When the list is empty
if(head == NULL)
{
printf("Unable to allocate memory.");
}
else
{
printf("\nEnter the data of node 1: ");
scanf("%d", &data);
head->data = data;
head->next = NULL;
temp = head;
for(i=2; i<=n; i++)
{
newNode = (struct node *)malloc(sizeof(struct node));
if(newNode == NULL)
{
printf("Unable to allocate memory.");
break;
}
else
{
printf("\nEnter the data of node %d: ", i);
scanf("%d", &data);
newNode->data = data;
newNode->next = NULL;
temp->next = newNode;
temp = temp->next;
}
}
}
}

/* Function to delete the last node */
void deletion_end()
{
// Empty List
if(head -> next == NULL)
{
free(head); //Delete head
head = NULL; // make the reference to NULL
}
struct node *temp = head,*prev_node;
while(temp -> next != NULL)
{
prev_node = temp;
temp = temp -> next; //Traverse to to the last node
}
free(temp); //Delete the last node
prev_node -> next = NULL;
}

void displayList()
{
struct node *temp;
if(head == NULL)
{
printf("List is empty.");
}
else
{
temp = head;
// Print the list
while(temp != NULL)
{
printf("%d\t", temp->data);
temp = temp->next;
}
printf("\n");
}
}
```

OUTPUT:



```
Enter the total number of nodes: 5

Enter the data of node 1: 5

Enter the data of node 2: 10

Enter the data of node 3: 15

Enter the data of node 4: 20

Enter the data of node 5: 25

The List is
5       10      15      20      25


After Deleting the last node, the List is
5       10      15      20

Process returned 0 (0x0)   execution time : 4.904 s
Press any key to continue.
```

Deletion at a given position in the linked list



- Now let us assume that the node at position 2 has to be deleted.
- Start traversing the list from the head node and move up to that **position.**
- While traversing, keep track of the previous node to the node to be deleted.
- In this case, since we want to delete the second node, you need to traverse till node 2, storing node 1 in some temporary variable.

- Now, the address part of node 2 is assigned to the address part of node 1 and then node 2 is deleted.

Program to delete a node at a given position in the Linked List

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

// Creating node with data and a pointer
struct node {
int data;
struct node *next;
}*head;
void createList(int n);
void deletion_given_pos(int pos);
void displayList();

int main()
{
int n, data, pos;
printf("\nEnter the total number of nodes: ");
scanf("%d", &n);
createList(n);
printf("\nThe List is \n");
displayList();
printf("\nEnter the position : "); //position of the node to b
scanf("%d",&pos);
deletion_given_pos(pos);
printf("\n\nAfter Deleting the node at given position, the Lis
displayList();
return 0;
}

void createList(int n)
{
struct node *newNode, *temp;
int data, i;
head = (struct node *)malloc(sizeof(struct node));
// When the list is empty
if(head == NULL)
{
printf("Unable to allocate memory.");
}
else
{
printf("\nEnter the data of node 1: ");
scanf("%d", &data);
head->data = data;
head->next = NULL;
temp = head;
for(i=2; i<=n; i++)
{
newNode = (struct node *)malloc(sizeof(struct node));
if(newNode == NULL)
{
printf("Unable to allocate memory.");
break;
}
else
{
printf("\nEnter the data of node %d: ", i);
scanf("%d", &data);
newNode->data = data;
newNode->next = NULL;
```

Explore 'DATA STRUCTURES'

Articles
(https://www.faceprep.in/data-structures#articles)

Mock Tests
(https://www.faceprep.in/data-structures#mock)

Practice Exercises
(https://www.faceprep.in/data-structures#practice)

```
temp->next = newNode;
temp = temp->next;
}
}
}
}


/* Function to delete the node at given position */
void deletion_given_pos(int pos)
{
// Empty List
if(head == NULL)
{
free(head); //Delete head
head = NULL; // make the reference to NULL
}
struct node *temp = head,*prev_node;
int count = 0;
while(temp -> next != NULL && pos != count)
{
prev_node = temp;  // Tracking the (position - 1 )node
temp = temp -> next;
count = count + 1;
}
if(pos == count)
{
prev_node -> next = temp -> next;  // Assigning the address of
free(temp); // Delete the node
}
}

void displayList()
{
struct node *temp;
if(head == NULL)
{
printf("List is empty.");
}
else
{
temp = head;
// Print the list
while(temp != NULL)
{
printf("%d\t", temp->data);
temp = temp->next;
}
printf("\n");
}
}
```

OUTPUT:

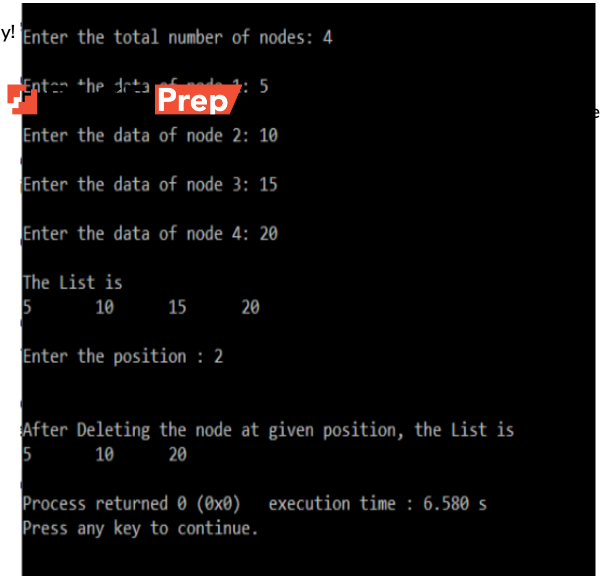
Link copied to clipboard. Share away!

Login (https://www.faceprep.in/login/?rurl=/data-structures/linked-list-deleting-a-node/)

Sign Up (https://www.faceprep.in/signup/?rurl=/data-structures/linked-list-deleting-a-node/)

eviews/)

Dismiss



```
Enter the total number of nodes: 4

Enter the data of node 1: 5

Enter the data of node 2: 10

Enter the data of node 3: 15

Enter the data of node 4: 20

The List is
5       10      15      20

Enter the position : 2


After Deleting the node at given position, the List is
5       10      20

Process returned 0 (0x0)   execution time : 6.580 s
Press any key to continue.
```

Data Structures (Https://Www.Faceprep.In/Data-Structures)

Algorithms (Https://Www.Faceprep.In/Algorithms)

C Programming (Https://Www.Faceprep.In/C)

C++ Programming (Https://Www.Faceprep.In/C-Plus-Plus)

Java Programming (Https://Www.Faceprep.In/Java)

Python Programming (Https://Www.Faceprep.In/Python)

If you have any feedback about this article and want to improve this, please write to **enquiry@faceprep.in**

Link copied to clipboard. Share away!

FACE Prep (https://www.faceprep.in/)

Login (https://www.faceprep.in/login/?rurl=/data-structures/linked-list-deleting-a-node/)

Dismiss

FACE Prep (https://www.faceprep.in/)

Explore

Wall of Love (https://www.faceprep.in/reviews/)

Sign Up (https://www.faceprep.in/signup/?rurl=/data-structures/linked-list-deleting-a-node/)

COMPANIES

TCS (https://www.faceprep.in/tcs/)

AMCAT (https://www.faceprep.in/amcat-exam/)

WIPRO (https://www.faceprep.in/wipro/)

Infosys (https://www.faceprep.in/infosys/)

TATA ELXSI (https://www.faceprep.in/tata-elxsi/)

Capgemini (https://www.faceprep.in/capgemini/)

Accenture (https://www.faceprep.in/accenture/)

Tech Mahindra (https://www.faceprep.in/tech-mahindra/)

Cognizant (https://www.faceprep.in/cognizant/)

IBM (https://www.faceprep.in/ibm/)

BYJU's (https://www.faceprep.in/byjus/)

Deloitte (https://www.faceprep.in/deloitte/)

eLitmus (https://www.faceprep.in/elitmus-exam/)

TCS Ninja (https://www.faceprep.in/tcs/)

Cocubes (https://www.faceprep.in/cocubes-exam/)

Mindtree (https://www.faceprep.in/mindtree/)

PROGRAD

About ProGrad (https://www.faceprep.in/prograd/)

ProGrad Microdegree (https://prograd.org/microdegree/)

ProGrad Bootcamp (https://prograd.org/bootcamp/)

Our Story (https://www.focusacademy.in/ourstory/)

Contact Us (https://www.faceprep.in/contact/)

Terms & Conditions (https://www.faceprep.in/terms-and-conditions/)

Privacy Policy (https://www.faceprep.in/privacy-policy/)

PLACEMENT PREP

Articles (https://www.faceprep.in/articles/)

Tests (https://www.faceprep.in/tests/)

Videos (https://www.faceprep.in/videos/)

Webinars (https://www.faceprep.in/webinars/)

SOCIAL

Youtube (https://www.youtube.com/channel/UCzl

Facebook (https://www.facebook.com/fa

oDJMA)

WhatsApp (https://www.focusacademy.i

Instagram (https://www.instagram.com/f

Telegram (https://t.me/faceprepoffcl)

Linkedin (https://www.linkedin.com/scho

SUBJECTS

Quantitative Aptitude (https://www.faceprep.in/quantitative-aptitude/)

Java (https://www.faceprep.in/java/)

Data Structures (https://www.faceprep.in/data-structures/)

Verbal Ability (https://www.faceprep.in/verbal-ability/)

Python (https://www.faceprep.in/python/)

C (https://www.faceprep.in/c/)

Algorithms (https://www.faceprep.in/algorithms/)

Logical Reasoning (https://www.faceprep.in/logical-reasoning/)