

1.Delection at Beginning

Programiz C Online Compiler

Programiz PRO >

main.c

To exit full screen, press and hold **Esc**

Clear

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 struct Node{
5     int data;
6     struct Node *next;
7 };
8
9 void deleteAtStart(struct Node **head){
10     if (*head == NULL){
11         printf("list is empty");
12         return;
13     }
14
15     struct Node *temp = *head;
16     *head = (*head)->next;
17     free(temp);
18 }
19
20 void display(struct Node *head){
21     struct Node *temp = head;
22     while(temp != NULL){
23         printf("%d -> ", temp->data);
24         temp = temp->next;
25     }
26     printf("NULL\n");
27 }
28
29 int main(){
30     struct Node *head, *first, *second, *third;
31
32     head = (struct Node*)malloc(sizeof(struct Node));
33     first = (struct Node*)malloc(sizeof(struct Node));
34     second = (struct Node*)malloc(sizeof(struct Node));
```

original list:
10 -> 20 -> 30 -> 40 -> NULL
after deleting first node:
20 -> 30 -> 40 -> NULL

=== Code Execution Successful ===

Programiz C Online Compiler

Programiz PRO >

main.c

Run

Output

Clear

```
25     }
26     printf("NULL\n");
27 }
28
29 int main(){
30     struct Node *head, *first, *second, *third;
31
32     head = (struct Node*)malloc(sizeof(struct Node));
33     first = (struct Node*)malloc(sizeof(struct Node));
34     second = (struct Node*)malloc(sizeof(struct Node));
35     third = (struct Node*)malloc(sizeof(struct Node));
36
37     head->data=10;
38     head->next=first;
39
40     first->data=20;
41     first->next=second;
42
43     second->data=30;
44     second->next=third;
45
46     third->data=40;
47     third->next=NULL;
48
49     printf("original list:\n");
50     display(head);
51
52     deleteAtStart(&head);
53
54     printf("after deleting first node:\n");
55     display(head);
56
57     return 0;
58 }
```

original list:
10 -> 20 -> 30 -> 40 -> NULL
after deleting first node:
20 -> 30 -> 40 -> NULL

=== Code Execution Successful ===

2.Delection at End

main.c

Run

Clear

```

1 #include <stdio.h>
2 #include <stdlib.h>
3
4 struct Node{
5     int data;
6     struct Node *next;
7 };
8
9 void deleteAtEnd(struct Node **head){
10     if (*head == NULL){
11         printf("list is empty");
12         return;
13     }
14     if ((*head) -> next == NULL){
15         free(head);
16         head = NULL;
17     }
18     struct Node *temp = *head;
19     while(temp->next->next != NULL){
20         temp = temp->next;
21     }
22     free(temp->next);
23     temp->next = NULL;
24 }
25
26 void display(struct Node *head){
27     struct Node *temp = head;
28     while(temp != NULL){
29         printf("%d -> ", temp->data);
30         temp = temp->next;
31     }
32     printf("NULL\n");
33 }
34

```

Output

```

original list:
10 -> 20 -> 30 -> 40 -> NULL
after deleting first node:
10 -> 20 -> 30 -> NULL

=== Code Execution Successful ===

```

Programiz PRO

Premium
Courses by
Programiz

[Learn More](#)

Programiz
C Online Compiler

Programiz PRO >

main.c

Run

Clear

```

30     temp = temp -> next;
31 }
32 }
33 printf("NULL\n");
34 }
35
36 int main(){
37     struct Node *head, *first, *second, *third;
38
39     head = (struct Node*)malloc(sizeof(struct Node));
40     first = (struct Node*)malloc(sizeof(struct Node));
41     second = (struct Node*)malloc(sizeof(struct Node));
42     third = (struct Node*)malloc(sizeof(struct Node));
43
44     head->data=10;
45     head->next=first;
46
47     first->data=20;
48     first->next=second;
49
50     second->data=30;
51     second->next=third;
52
53     third->data=40;
54     third->next=NULL;
55
56     printf("original list:\n");
57     display(head);
58
59     deleteAtEnd(&head);
60
61     printf("after deleting first node:\n");
62     display(head);

```

Output

```

original list:
10 -> 20 -> 30 -> 40 -> NULL
after deleting first node:
10 -> 20 -> 30 -> NULL

=== Code Execution Successful ===

```

Programiz PRO

Premium
Courses by
Programiz

[Learn More](#)

Programiz

C Online Compiler

Programiz PRO

main.c

<>

↺

↻

Share

Run

33

printf("NULL\n");

34

}

35

36

int main(){

37

struct Node *head, *first, *second, *third;

38

39

head = (struct Node*)malloc(sizeof(struct Node));

40

first = (struct Node*)malloc(sizeof(struct Node));

41

second = (struct Node*)malloc(sizeof(struct Node));

42

third = (struct Node*)malloc(sizeof(struct Node));

43

44

head->data=10;

45

head->next=first;

46

47

first->data=20;

48

first->next=second;

49

50

second->data=30;

51

second->next=third;

52

53

third->data=40;

54

third->next=NULL;

55

56

printf("original list:\n");

57

display(head);

58

59

deleteAtEnd(&head);

60

61

printf("after deleting first node:\n");

62

display(head);

63

64

return 0;

65

}

Output

Clear

original list:
10 -> 20 -> 30 -> 40 -> NULL
after deleting first node:
10 -> 20 -> 30 -> NULL

=== Code Execution Successful ===

Programiz PRO

Premium Courses by Programiz

Learn More

3.Delection at position

```
main.c
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 struct Node{
5     int data;
6     struct Node *next;
7 };
8
9 void deleteAtPosition(struct Node **head, int position){
10     if (*head == NULL || position < 1){
11         printf("Invalid operation\n");
12         return;
13     }
14
15     struct Node *temp = *head;
16
17
18     if(position == 1){
19         *head = temp->next;
20         free(temp);
21         return;
22     }
23
24     for(int i = 1; i < position - 1 && temp->next != NULL; i++){
25         temp = temp->next;
26     }
27
28     if(temp->next == NULL){
29         printf("Position out of range\n");
30         return;
31     }
32
33     struct Node *delNode = temp->next;
34     temp->next = delNode->next;
```

Original list:
10 -> 20 -> 30 -> 40 -> NULL
After deleting node at position 3:
10 -> 20 -> 40 -> NULL

=== Code Execution Successful ===

```
main.c
31 }
32
33 struct Node *delNode = temp->next;
34 temp->next = delNode->next;
35 free(delNode);
36 }
37
38 void display(struct Node *head){
39     struct Node *temp = head;
40     while(temp != NULL){
41         printf("%d -> ", temp->data);
42         temp = temp->next;
43     }
44     printf("NULL\n");
45 }
46
47 int main(){
48     struct Node *head, *first, *second, *third;
49
50     head = (struct Node*)malloc(sizeof(struct Node));
51     first = (struct Node*)malloc(sizeof(struct Node));
52     second = (struct Node*)malloc(sizeof(struct Node));
53     third = (struct Node*)malloc(sizeof(struct Node));
54
55     head->data = 10;
56     head->next = first;
57
58     first->data = 20;
59     first->next = second;
60
61     second->data = 30;
62     second->next = third;
63
64     third->data = 40;
```

Original list:
10 -> 20 -> 30 -> 40 -> NULL
After deleting node at position 3:
10 -> 20 -> 40 -> NULL

=== Code Execution Successful ===

```
main.c
44 printf("NULL\n");
45 }
46
47 int main()
48 {
49     struct Node *head, *first, *second, *third;
50
51     head = (struct Node*)malloc(sizeof(struct Node));
52     first = (struct Node*)malloc(sizeof(struct Node));
53     second = (struct Node*)malloc(sizeof(struct Node));
54     third = (struct Node*)malloc(sizeof(struct Node));
55
56     head->data = 10;
57     head->next = first;
58
59     first->data = 20;
60     first->next = second;
61
62     second->data = 30;
63     second->next = third;
64
65     third->data = 40;
66     third->next = NULL;
67
68     printf("Original list:\n");
69     display(head);
70
71     deleteAtPosition(&head, 3);
72
73     printf("After deleting node at position 3:\n");
74     display(head);
75
76     return 0;
77 }
```

Output

Original list:
10 -> 20 -> 30 -> 40 -> NULL
After deleting node at position 3:
10 -> 20 -> 40 -> NULL

=== Code Execution Successful ===