

**Manav Rachna International Institute of Research and Studies Bachelor's in
computer applications**

Data Structures using C



Submitted by: Urvashi Pahuja

Department: School Of Computer Applications

Course: Bcahelors in Computers 1pplications

Roll no. : 24/SCA/BCA/087

Semester: 2nd

Subject: Data Structures Using C

DS FILE

AIM1: ADDING AN ELEMENT AT THE BEGINNING OF THE LINKED LIST.

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

// Define node structure
struct node {
    int data;
    struct node *next;
};

// Global pointers
struct node *head = NULL;

// Function to insert at the beginning
void insertatbegin(int data) {
    struct node *lk = (struct node*) malloc(sizeof(struct node));
    if (lk == NULL) {
        printf("Memory allocation failed\n");
        return;
    }
    lk->data = data;
```

```
lk->next = head;
head = lk;
}
```

// Function to print the list

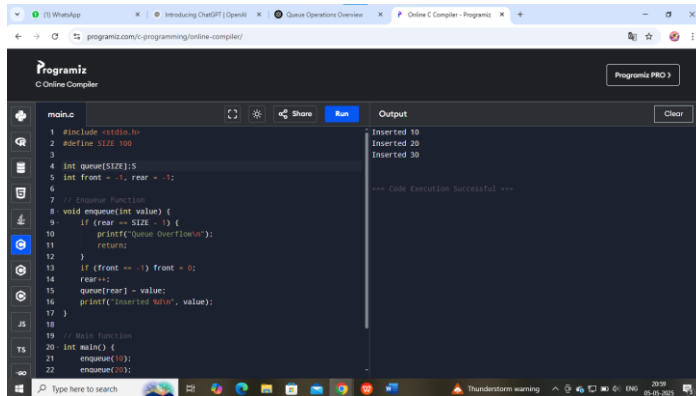
```
void printList() {
    struct node *p = head;
    printf("\n[");
    while (p != NULL) {
        printf(" %d ", p->data);
        p = p->next;
    }
    printf("]\n");
}
```

```
int main() {
    insertatbegin(12);
    insertatbegin(22);
    insertatbegin(30);
    insertatbegin(44);
    insertatbegin(50);

    printf("Linked List: ");
    printList();

    return 0;
}
```

OUTPUT:



```
1 #include <stdio.h>
2 #define SIZE 100
3
4 int queue[SIZE];
5 int front = -1, rear = -1;
6
7 // Queue Function
8 void enqueue(int value) {
9     if (rear == SIZE - 1) {
10         printf("Queue Overflow\n");
11         return;
12     }
13     if (front == -1) front = 0;
14     rear++;
15     queue[rear] = value;
16     printf("Inserted %d\n", value);
17 }
18
19 // Main Function
20 int main() {
21     enqueue(10);
22     enqueue(20);
23 }
```

Inserted 10
Inserted 20
Inserted 30

Code Execution Successful

AIM 2: ADDING AN ELEMENT AT THE ENDING OF THE LINKED LIST.

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

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

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

```
struct node *head = NULL;
```

```
// Display the list
```

```
void printList() {  
    struct node *p = head;  
    printf("\n[");  
    while (p != NULL) {  
        printf(" %d ", p->data);  
        p = p->next;  
    }  
    printf("]\n");  
}
```

```
// Insert at the beginning
```

```
void insertatbegin(int data) {  
    struct node *lk = (struct node*) malloc(sizeof(struct  
node));  
    if (lk == NULL) {  
        printf("Memory allocation failed\n");  
        return;  
    }  
    lk->data = data;  
    lk->next = head;  
    head = lk;  
}
```

```

// Insert at the end
void insertatend(int data) {
    struct node *lk = (struct node*) malloc(sizeof(struct
node));
    if (lk == NULL) {
        printf("Memory allocation failed\n");
        return;
    }
    lk->data = data;
    lk->next = NULL;

    if (head == NULL) {
        // If the list is empty
        head = lk;
    } else {
        struct node *linkedlist = head;
        while (linkedlist->next != NULL)
            linkedlist = linkedlist->next;
        linkedlist->next = lk;
    }
}

```

```

// Main function
int main() {
    insertatbegin(12); // List: 12

```

```

insertatend(22);    // List: 12 -> 22
insertatend(30);    // List: 12 -> 22 -> 30
insertatend(44);    // ...
insertatend(50);

printf("Linked List: ");
printList();

return 0;
}

```

OUTPUT:

The screenshot displays the Programiz Online C Compiler interface. The code editor on the left contains the following C code:

```

main.c
47- } else {
48-     struct node *linkedList = head;
49-     while (linkedList->next != NULL)
50-         linkedList = linkedList->next;
51-     linkedList->next = lk;
52- }
53- }
54-
55- // Main function
56- int main() {
57-     insertatbegin(12); // List: 12
58-     insertatend(22);   // List: 12 -> 22
59-     insertatend(30);   // List: 12 -> 22 -> 30
60-     insertatend(44);   // ...
61-     insertatend(50);
62-
63-     printf("Linked List: ");
64-     printList();
65-
66-     return 0;
67- }
68-

```

The output window on the right shows the result of the program execution:

```

Linked List:
[ 12 22 30 44 50 ]

=== Code Execution Successful ===

```

The browser's taskbar at the bottom indicates the system temperature is 27°C, it is hazy, and the date is 05-05-2025.

AIM 3: ADDING AN ELEMENT AT ANY POSITION WITHIN THE LIST.

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

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

```
// Node structure
```

```
struct node {
```

```
    int data;
```

```
    struct node *next;
```

```
};
```

```
// Global head pointer
```

```
struct node *head = NULL;
```

```
// Display the list
```

```
void printList() {
```

```
    struct node *p = head;
```

```
    printf("\n[");
```

```
    while (p != NULL) {
```

```
        printf(" %d ", p->data);
```

```
        p = p->next;
```



```
    }  
    printf("]\n");  
}
```

// Insert at beginning

```
void insertatbegin(int data) {  
    struct node *lk = (struct node *)malloc(sizeof(struct node));  
    if (lk == NULL) {  
        printf("Memory allocation failed\n");  
        return;  
    }  
    lk->data = data;  
    lk->next = head;  
    head = lk;  
}
```

// Insert after a given node

```
void insertafternode(struct node *list, int data) {  
    if (list == NULL) {  
        printf("The given previous node is NULL.\n");  
        return;  
    }
```

```
struct node *lk = (struct node *)malloc(sizeof(struct node));
if (lk == NULL) {
    printf("Memory allocation failed\n");
    return;
}
lk->data = data;
lk->next = list->next;
list->next = lk;
}
```

// Main function

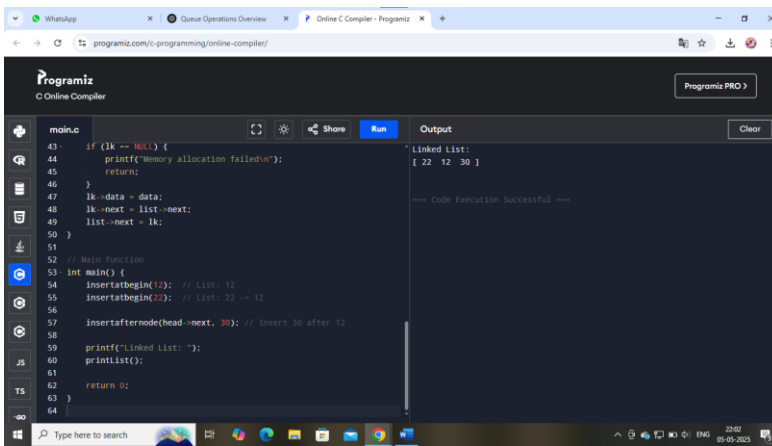
```
int main() {
    insertatbegin(12); // List: 12
    insertatbegin(22); // List: 22 -> 12

    insertafternode(head->next, 30); // Insert 30 after 12

    printf("Linked List: ");
    printList();

    return 0;
}
```

OUTPUT:



The screenshot shows the Programiz Online C Compiler interface. The code in the editor is as follows:

```
43- if (lk == NULL) {
44-     printf("Memory allocation failed\n");
45-     return;
46- }
47- lk->data = data;
48- lk->next = list->next;
49- list->next = lk;
50- }
51-
52- // Main Function
53- int main() {
54-     insertatbegin(12); // List: 12
55-     insertatbegin(22); // List: 22 -> 12
56-
57-     insertafternode(head->next, 30); // Insert 30 after 12
58-
59-     printf("Linked List: ");
60-     printList();
61-
62-     return 0;
63- }
```

The output on the right shows:

```
Linked List:
[ 22 12 30 ]

--- Code Execution Successful ---
```

AIM 4: DELETION AT THE BEGINNING OF THE LINKED LIST.

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

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

```
// Node structure
```

```
struct node {
```

```
    int data;
```

```
    struct node *next;
```

```
};
```

```
// Global head pointer
```

```
struct node *head = NULL;
```

```
// Display the list
```

```
void printList() {  
    struct node *p = head;  
    printf("\n[");  
    while (p != NULL) {  
        printf(" %d ", p->data);  
        p = p->next;  
    }  
    printf("]\n");  
}
```

// Insertion at the beginning

```
void insertatbegin(int data) {  
    struct node *lk = (struct node *)malloc(sizeof(struct node));  
    if (lk == NULL) {  
        printf("Memory allocation failed\n");  
        return;  
    }  
    lk->data = data;  
    lk->next = head;  
    head = lk;  
}
```

```
// Delete at beginning
void deleteatbegin() {
    if (head == NULL) {
        printf("List is already empty.\n");
        return;
    }
    struct node *temp = head;
    head = head->next;
    free(temp); // Free memory to avoid leak
}
```

```
int main() {
    insertatbegin(12);
    insertatbegin(22);
    insertatbegin(30);
    insertatbegin(40);
    insertatbegin(55);

    printf("Linked List: ");
    printList();

    deleteatbegin();
}
```

```

printf("\nLinked List after deletion: ");

printList();

return 0;
}

```

OUTPUT:

The screenshot shows the Programiz Online C Compiler interface. The code editor on the left contains the following C code:

```

main.c
43 head = head->next;
44 free(temp); // Free memory to avoid leak
45 }
46
47 int main() {
48     insertatbegin(12);
49     insertatbegin(22);
50     insertatbegin(30);
51     insertatbegin(40);
52     insertatbegin(55);
53
54     printf("Linked List: ");
55     printList();
56
57     deleteatbegin();
58
59     printf("\nLinked List after deletion: ");
60     printList();
61
62     return 0;
63 }
64

```

The output panel on the right shows the following output:

```

Linked List:
[ 55 40 30 22 12 ]

Linked List after deletion:
[ 40 30 22 12 ]

--- Code Execution Successful ---

```

The browser's address bar shows the URL: programiz.com/c-programming/online-compiler/. The browser tabs include WhatsApp, Queue Operations Overview, and Online C Compiler - Programiz.

AIM 5: DELETING AN ELEMENT AT THE ENDING OF THE LINKED LIST.

```

#include <stdio.h>

#include <stdlib.h>

```

```
// Node structure
```

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

```
// Global head pointer
```

```
struct node *head = NULL;
```

```
// Display the list
```

```
void printList() {  
    struct node *p = head;  
    printf("\n[");  
    while (p != NULL) {  
        printf(" %d ", p->data);  
        p = p->next;  
    }  
    printf("]");  
}
```

```
// Insert at the beginning
```

```
void insertatbegin(int data) {  
    struct node *lk = (struct node *)malloc(sizeof(struct  
node));
```

```
if (lk == NULL) {  
    printf("Memory allocation failed\n");  
    return;  
}  
lk->data = data;  
lk->next = head;  
head = lk;  
}
```

// Delete the last node

```
void deleteatend() {  
    if (head == NULL) {  
        printf("\nList is empty.");  
        return;  
    }
```

// Only one node

```
if (head->next == NULL) {  
    free(head);  
    head = NULL;  
    return;  
}
```

```
struct node *linkedlist = head;
```



```
// Traverse to second last node
while (linkedlist->next->next != NULL) {
    linkedlist = linkedlist->next;
}

// Delete last node
free(linkedlist->next);
linkedlist->next = NULL;
}

int main() {
    insertatbegin(12);
    insertatbegin(22);
    insertatbegin(30);
    insertatbegin(40);
    insertatbegin(55);

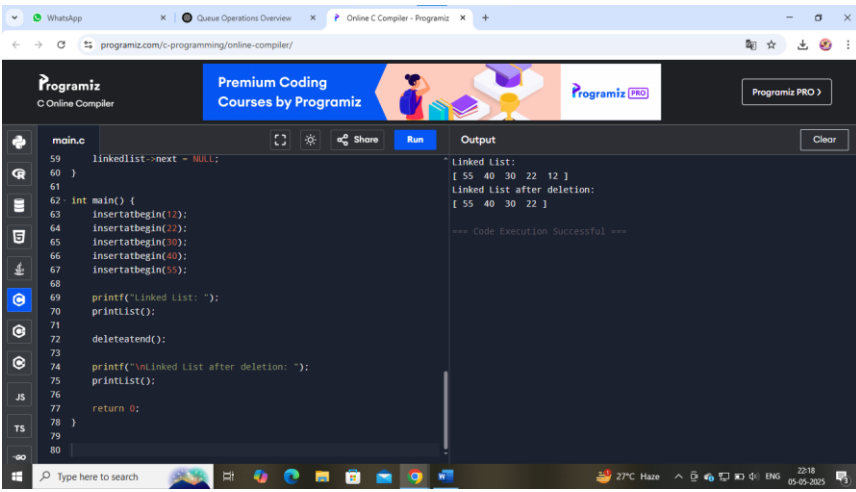
    printf("Linked List: ");
    printList();

    deleteatend();

    printf("\nLinked List after deletion: ");
    printList();
}
```

```
return 0;
}
```

OUTPUT:



The screenshot shows the Programiz Online C Compiler interface. The code editor on the left contains the following C code:

```
59 linkedlist->next = NULL;
60 }
61
62 int main() {
63     insertatbegin(12);
64     insertatbegin(22);
65     insertatbegin(30);
66     insertatbegin(40);
67     insertatbegin(55);
68
69     printf("Linked List: ");
70     printlist();
71
72     deleteatend();
73
74     printf("\nLinked List after deletion: ");
75     printlist();
76
77     return 0;
78 }
79
80
```

The output panel on the right shows the following output:

```
Linked List:
[ 55 40 30 22 12 ]
Linked List after deletion:
[ 55 40 30 22 ]
=== Code Execution Successful ===
```

AIM 6: DELETING AN ELEMENT AT ANY POSITION WITHIN THE LIST.

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

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

```
// Node structure
```

```
struct node {
```

```
    int data;
```

```
    struct node *next;
```

```
};
```

```
// Head pointer
```

```
struct node *head = NULL;
```

```
// Function to print the list
```

```
void printList() {
```

```
    struct node *p = head;
```

```
    printf("\n[");
```

```
    while (p != NULL) {
```

```
        printf(" %d ", p->data);
```

```
        p = p->next;
```

```
    }
```

```
    printf("]\n");
```

```
}
```

```
// Insert at beginning
```

```
void insertatbegin(int data) {
```

```
    struct node *lk = (struct node *)malloc(sizeof(struct node));
```

```
    if (lk == NULL) {
```

```
        printf("Memory allocation failed\n");
```

```
        return;
```

```
    }
```

```
    lk->data = data;
```

```
    lk->next = head;
```

```
    head = lk;
```

```
}
```

```
// Delete a node by key
```

```
void deletenode(int key) {
```

```
    struct node *temp = head, *prev = NULL;
```

```
    // If head node holds the key
```

```
    if (temp != NULL && temp->data == key) {
```

```
        head = temp->next;
```

```
        free(temp);
```

```
        return;
```

```
    }
```

```
// Search for the key
```

```
while (temp != NULL && temp->data != key) {
```

```
    prev = temp;
```

```
    temp = temp->next;
```

```
}
```

```
// If key not found
```

```
if (temp == NULL) return;
```

```
// Unlink and delete the node
```

```
prev->next = temp->next;
```

```
    free(temp);  
}  
  
int main() {  
    insertatbegin(12);  
    insertatbegin(22);  
    insertatbegin(30);  
    insertatbegin(40);  
    insertatbegin(55); // List: 55 40 30 22 12  
  
    printf("Linked List: ");  
    printList();  
  
    deletenode(30);  
  
    printf("Linked List after deletion: ");  
    printList();  
  
    return 0;  
}
```

OUTPUT:

The screenshot displays the Programiz Online C Compiler interface. The browser's address bar shows the URL `programiz.com/c-programming/online-compiler/`. The page header includes the Programiz logo, a banner for 'Premium Coding Courses by Programiz', and a 'Programiz PRO' button. On the left, a sidebar contains icons for various programming languages, with C selected. The main editor area, titled 'main.c', contains the following C code:

```
57 prev->next = temp->next;
58 free(temp);
59 }
60
61 int main() {
62     insertatbegin(12);
63     insertatbegin(22);
64     insertatbegin(30);
65     insertatbegin(40);
66     insertatbegin(55); // List: 55 40 30 22 12
67
68     printf("Linked List: ");
69     printList();
70
71     deletenode(30);
72
73     printf("Linked List after deletion: ");
74     printList();
75
76     return 0;
77 }
78
```

The 'Output' panel on the right shows the execution results:

```
Linked List:
[ 55 40 30 22 12 ]
Linked List after deletion:
[ 55 40 22 12 ]

=== Code Execution Successful ===
```

The bottom of the image shows a Windows taskbar with the search bar, taskbar icons, and system tray showing the time as 22:26 on 05-05-2025.