

```

// 10. Write a C program to implement Linked list operations
#include <stdio.h>
#include <stdlib.h>
struct node {
    int data;
    struct node *next;
};

struct node *start = NULL;
void insert_at_begin(int);
void insert_at_end(int);
void traverse();
void delete_from_begin();
void delete_from_end();
int count = 0;

int main () {
    int i, data;

    for (;;) {
        printf("1. Insert an element at the beginning of linked list.\n");
        printf("2. Insert an element at the end of linked list.\n");
        printf("3. Traverse linked list.\n");
        printf("4. Delete an element from beginning.\n");
        printf("5. Delete an element from end.\n");
        printf("6. Exit\n");

        scanf("%d", &i);

        if (i == 1) {
            printf("Enter value of element\n");
            scanf("%d", &data);
            insert_at_begin(data);
        }
        else if (i == 2) {
            printf("Enter value of element\n");
            scanf("%d", &data);
            insert_at_end(data);
        }
        else if (i == 3)
            traverse();
        else if (i == 4)
            delete_from_begin();
        else if (i == 5)

```

```

        delete_from_end();
    else if (i == 6)
        break;
    else
        printf("Please enter valid input.\n");
}

return 0;
}

void insert_at_begin(int x) {
    struct node *t;

    t = (struct node*)malloc(sizeof(struct node));
    t->data = x;
    count++;

    if (start == NULL) {
        start = t;
        start->next = NULL;
        return;
    }

    t->next = start;
    start = t;
}

void insert_at_end(int x) {
    struct node *t, *temp;

    t = (struct node*)malloc(sizeof(struct node));
    t->data = x;
    count++;

    if (start == NULL) {
        start = t;
        start->next = NULL;
        return;
    }

    temp = start;

    while (temp->next != NULL)
        temp = temp->next;

```

```

temp->next = t;
t->next = NULL;
}

void traverse() {
    struct node *t;

    t = start;

    if (t == NULL) {
        printf("Linked list is empty.\n");
        return;
    }

    printf("There are %d elements in linked list.\n", count);

    while (t->next != NULL) {
        printf("%d\n", t->data);
        t = t->next;
    }
    printf("%d\n", t->data); // Print last node
}

void delete_from_begin() {
    struct node *t;
    int n;

    if (start == NULL) {
        printf("Linked list is empty.\n");
        return;
    }

    n = start->data;
    t = start->next;
    free(start);
    start = t;
    count--;

    printf("%d deleted from the beginning successfully.\n", n);
}

void delete_from_end() {
    struct node *t, *u;

```

```
int n;
```

```
if (start == NULL) {  
    printf("Linked list is empty.\n");  
    return;  
}
```

```
count--;
```

```
if (start->next == NULL) {  
    n = start->data;  
    free(start);  
    start = NULL;  
    printf("%d deleted from end successfully.\n", n);  
    return;  
}
```

```
t = start;
```

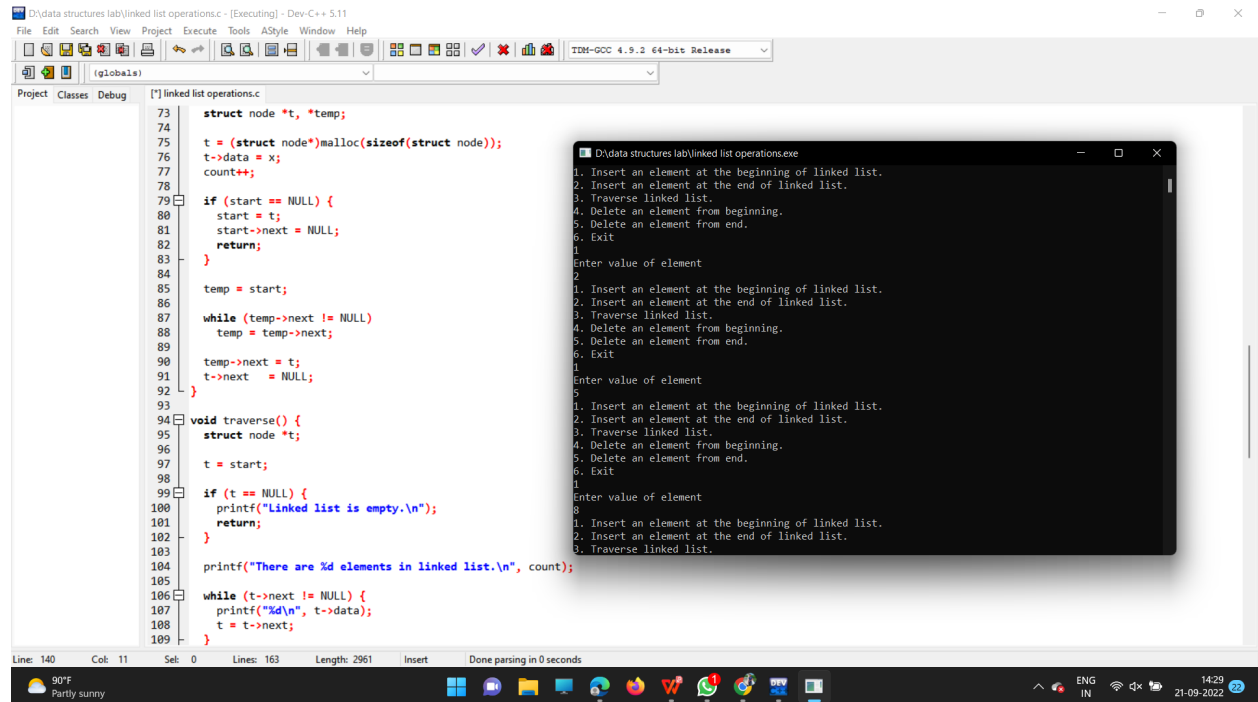
```
while (t->next != NULL) {  
    u = t;  
    t = t->next;  
}
```

```
n = t->data;  
u->next = NULL;  
free(t);
```

```
printf("%d deleted from end successfully.\n", n);  
}
```

```
D:\data structures lab\linked list operations.c - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
(globals)
[*] linked list operations.c
1 // 10. Write a C program to implement Linked List operations
2 #include <stdio.h>
3 #include <stdlib.h>
4 struct node {
5     int data;
6     struct node *next;
7 };
8
9 struct node *start = NULL;
10 void insert_at_begin(int);
11 void insert_at_end(int);
12 void traverse();
13 void delete_from_begin();
14 void delete_from_end();
15 int count = 0;
16
17 int main () {
18     int i, data;
19
20     for (;;) {
21         printf("1. Insert an element at the beginning of linked list.\n");
22         printf("2. Insert an element at the end of linked list.\n");
23         printf("3. Traverse linked list.\n");
24         printf("4. Delete an element from beginning.\n");
25         printf("5. Delete an element from end.\n");
26         printf("6. Exit\n");
27
28         scanf("%d", &i);
29
30         if (i == 1) {
31             printf("Enter value of element\n");
32             scanf("%d", &data);
33             insert_at_begin(data);
34         }
35         else if (i == 2) {
36             printf("Enter value of element\n");
37             scanf("%d", &data);
38         }
39     }
40
41     return 0;
42 }
43
44 void insert_at_begin(int x) {
45     struct node *t;
46     t = (struct node*)malloc(sizeof(struct node));
47     t->data = x;
48     count++;
49     if (start == NULL) {
50         start = t;
51         start->next = NULL;
52         return;
53     }
54     t->next = start;
55     start = t;
56 }
57
58 void insert_at_end(int x) {
59     struct node *t, *temp;
60     t = (struct node*)malloc(sizeof(struct node));
61     t->data = x;
62     count++;
63     if (start == NULL) {
64         start = t;
65         start->next = NULL;
66         return;
67     }
68     temp = start;
69     while (temp->next != NULL) {
70         temp = temp->next;
71     }
72     temp->next = t;
73 }
```

```
D:\data structures lab\linked list operations.c - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
(globals)
[*] linked list operations.c
37 scanf("%d", &data);
38 insert_at_end(data);
39 }
40 else if (i == 3)
41     traverse();
42 else if (i == 4)
43     delete_from_begin();
44 else if (i == 5)
45     delete_from_end();
46 else if (i == 6)
47     break;
48 else
49     printf("Please enter valid input.\n");
50 }
51
52 return 0;
53 }
54
55 void insert_at_begin(int x) {
56     struct node *t;
57     t = (struct node*)malloc(sizeof(struct node));
58     t->data = x;
59     count++;
60     if (start == NULL) {
61         start = t;
62         start->next = NULL;
63         return;
64     }
65     t->next = start;
66     start = t;
67 }
68
69 void insert_at_end(int x) {
70     struct node *t, *temp;
71     t = (struct node*)malloc(sizeof(struct node));
72     t->data = x;
73     count++;
74     if (start == NULL) {
75         start = t;
76         start->next = NULL;
77         return;
78     }
79     temp = start;
80     while (temp->next != NULL) {
81         temp = temp->next;
82     }
83     temp->next = t;
84 }
```



D:\data structures lab\linked list operations.c - [Executing] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Release

(globals)

Project Classes Debug

[*] linked list operations.c

```
100 printf("Linked list is empty.\n");
101 return;
102 }
103
104 printf("There are %d elements in linked list.\n", count);
105
106 while (t->next != NULL) {
107     printf("%d\n", t->data);
108     t = t->next;
109 }
110 printf("%d\n", t->data); // Print Last node
111 }
112
113 void delete_from_begin() {
114     struct node *t;
115     int n;
116
117     if (start == NULL) {
118         printf("Linked list is empty.\n");
119         return;
120     }
121
122     n = start->data;
123     t = start->next;
124     free(start);
125     start = t;
126     count--;
127
128     printf("%d deleted from the beginning successfully.\n", n);
129 }
130
131 void delete_from_end() {
132     struct node *t, *u;
133     int n;
134
135     if (start == NULL) {
136         printf("Linked list is empty.\n");
137         return;
138     }
139 }
```

Line: 140 Col: 11 Sel: 0 Lines: 163 Length: 2961 Insert Done parsing in 0 seconds

90°F Partly sunny

D:\data structures lab\linked list operations.exe

```
2. Insert an element at the end of linked list.
3. Traverse linked list.
4. Delete an element from beginning.
5. Delete an element from end.
6. Exit
2
Enter value of element
2
1. Insert an element at the beginning of linked list.
2. Insert an element at the end of linked list.
3. Traverse linked list.
4. Delete an element from beginning.
5. Delete an element from end.
6. Exit
3
There are 4 elements in linked list.
8
2
2
1. Insert an element at the beginning of linked list.
2. Insert an element at the end of linked list.
3. Traverse linked list.
4. Delete an element from beginning.
5. Delete an element from end.
6. Exit
4
8 deleted from the beginning successfully.
1. Insert an element at the beginning of linked list.
2. Insert an element at the end of linked list.
3. Traverse linked list.
4. Delete an element from beginning.
5. Delete an element from end.
6. Exit
2
2 deleted from end successfully.
1. Insert an element at the beginning of linked list.
2. Insert an element at the end of linked list.
3. Traverse linked list.
4. Delete an element from beginning.
5. Delete an element from end.
6. Exit
6
```

D:\data structures lab\linked list operations.c - [Executing] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Release

(globals)

Project Classes Debug

[*] linked list operations.c

```
128 printf("%d deleted from the beginning successfully.\n", n);
129 }
130
131 void delete_from_end() {
132     struct node *t, *u;
133     int n;
134
135     if (start == NULL) {
136         printf("Linked list is empty.\n");
137         return;
138     }
139     count--;
140
141     if (start->next == NULL) {
142         n = start->data;
143         free(start);
144         start = NULL;
145         printf("%d deleted from end successfully.\n", n);
146         return;
147     }
148     t = start;
149
150     while (t->next != NULL) {
151         u = t;
152         t = t->next;
153     }
154
155     n = t->data;
156     u->next = NULL;
157     free(t);
158
159     printf("%d deleted from end successfully.\n", n);
160 }
161
162 }
163 }
```

Line: 140 Col: 11 Sel: 0 Lines: 163 Length: 2961 Insert Done parsing in 0 seconds

90°F Partly sunny

D:\data structures lab\linked list operations.exe

```
1. Insert an element at the beginning of linked list.
2. Insert an element at the end of linked list.
3. Traverse linked list.
4. Delete an element from beginning.
5. Delete an element from end.
6. Exit
3
There are 4 elements in linked list.
8
2
2
1. Insert an element at the beginning of linked list.
2. Insert an element at the end of linked list.
3. Traverse linked list.
4. Delete an element from beginning.
5. Delete an element from end.
6. Exit
4
8 deleted from the beginning successfully.
1. Insert an element at the beginning of linked list.
2. Insert an element at the end of linked list.
3. Traverse linked list.
4. Delete an element from beginning.
5. Delete an element from end.
6. Exit
2
2 deleted from end successfully.
1. Insert an element at the beginning of linked list.
2. Insert an element at the end of linked list.
3. Traverse linked list.
4. Delete an element from beginning.
5. Delete an element from end.
6. Exit
6
Process exited after 108.7 seconds with return value 0
Press any key to continue . . .
```