

```
singlyLinkedList.c - CodeBlocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools- Plugins DocyBlocks Settings Help
Start here X singlyLinkedList.c X
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 struct node {
5     int data;
6     struct node *next;
7 }
8
9 struct node *head = NULL;
10
11
12
13 void createList(int n) {
14     struct node *newNode, *temp = NULL;
15     int data, i;
16
17     if (n <= 0) {
18         printf("Number of nodes should be greater than 0\n");
19         return;
20     }
21
22     if (head != NULL) {
23         printf("Warning: Overwriting existing list.\n");
24         head = NULL;
25     }
26
27     for (i = 1; i <= n; i++) {
28         newNode = (struct node *) malloc(sizeof(struct node));
29         if (newNode == NULL) {
30             printf("Memory allocation failed\n");
31             return;
32         }
33         printf("Enter data for node %d: ", i);
34         scanf("%d", &data);
35         newNode->data = data;
36         newNode->next = NULL;
37
38         if (head == NULL)
39             head = newNode;
40         else
41             temp->next = newNode;
42         temp = newNode;
43     }
44
45     printf("\nLinked list created successfully\n");
46 }
47
48
49 void insertAtBeginning(int data) {
50     struct node *newNode = (struct node *) malloc(sizeof(struct node));
51     if (newNode == NULL) {
52         printf("Memory allocation failed\n");
53         return;
54     }
55     newNode->data = data;
56     newNode->next = head;
57     head = newNode;
58     printf("Node inserted at the beginning\n");
59 }
60
61 void insertAtEnd(int data) {
62     struct node *newNode = (struct node *) malloc(sizeof(struct node));
63     if (newNode == NULL) {
64         printf("Memory allocation failed\n");
65         return;
66     }
67     newNode->data = data;
68     newNode->next = NULL;
69
70     if (head == NULL) {
71         head = newNode;
72     }
73     else {
74         struct node *temp = head;
75         while (temp->next != NULL)
76             temp = temp->next;
77         temp->next = newNode;
78     }
79     printf("Node inserted at the end\n");
80 }
81
82 void insertAtPosition(int data, int pos) {
83     int i;
84     struct node *newNode, *temp = head;
85
86     if (pos < 1) {
87         printf("Invalid position. Position must be 1 or greater.\n");
88         return;
89     }
90     if (pos == 1) {
91         insertAtBeginning(data);
92         return;
93     }
94
95     for (i = 1; i < pos - 1 && temp != NULL; i++)
96         temp = temp->next;
97
98     newNode = (struct node *) malloc(sizeof(struct node));
99     if (newNode == NULL) {
100         printf("Memory allocation failed\n");
101         return;
102     }
103     newNode->data = data;
104     newNode->next = temp->next;
105     temp->next = newNode;
106 }
```

```
singlylinkedlist.c - CodeBlocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools- Plugins DocyBlocks Settings Help
Start here X singlylinkedlist.c X
103     temp = temp->next;
104
105
106     if (temp == NULL) {
107         printf("Position out of range: List is not long enough to reach position %d.\n", pos);
108         return;
109     }
110
111     struct node* newNode = malloc(sizeof(struct node));
112     if (newNode == NULL) {
113         printf("Memory allocation failed.\n");
114         return;
115     }
116     newNode->data = data;
117
118     newNode->next = temp->next;
119     temp->next = newNode;
120     temp = temp->next;
121     printf("Node inserted at position %d\n", pos);
122 }
123
124 void displayList() {
125     struct node *temp = head;
126
127     if (head == NULL) {
128         printf("List is empty\n");
129         return;
130     }
131     printf("\nLinked list: ");
132     while (temp != NULL) {
133         printf("%d -> ", temp->data);
134         temp = temp->next;
135     }
136     printf("NULL\n");
137 }
138
139 int main() {
140     int choice, n, data, pos;
141
142     while (1) {
143         printf("\n---- Singly Linked List Operations ----\n");
144         printf("1. Create linked list\n");
145         printf("2. Insert at Beginning\n");
146         printf("3. Insert at any Position\n");
147         printf("4. Insert at End\n");
148         printf("5. Display list\n");
149         printf("6. Exit\n");
150         printf("Enter your choice: ");
151         if (scanf("%d", &choice) != 1) {
152             while (getchar() != '\n');
153             printf("Invalid input. Please enter a number.\n");
154             continue;
155         }
156         switch (choice) {
157             case 1:
158                 printf("Enter number of nodes: ");
159                 scanf("%d", &n);
160                 createList(n);
161                 break;
162             case 2:
163                 printf("Enter data to insert: ");
164                 scanf("%d", &data);
165                 insertAtBeginning(data);
166                 break;
167             case 3:
168                 printf("Enter data: ");
169                 scanf("%d", &data);
170                 printf("Enter position: ");
171                 scanf("%d", &pos);
172                 insertAtPosition(data, pos);
173                 break;
174             case 4:
175                 printf("Enter data to insert: ");
176                 scanf("%d", &data);
177                 insertAtEnd(data);
178                 break;
179             case 5:
180                 displayList();
181                 break;
182             case 6:
183                 printf("Exiting...\n");
184                 exit(0);
185             default:
186                 printf("Invalid choice. Try again.\n");
187         }
188     }
189     return 0;
190 }
```

```
singlylinkedlist.c - CodeBlocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools- Plugins DocyBlocks Settings Help
Start here X singlylinkedlist.c X
148     while (1) {
149         printf("\n---- Singly Linked List Operations ----\n");
150         printf("1. Create linked list\n");
151         printf("2. Insert at Beginning\n");
152         printf("3. Insert at any Position\n");
153         printf("4. Insert at End\n");
154         printf("5. Display list\n");
155         printf("6. Exit\n");
156         printf("Enter your choice: ");
157         if (scanf("%d", &choice) != 1) {
158             while (getchar() != '\n');
159             printf("Invalid input. Please enter a number.\n");
160             continue;
161         }
162         switch (choice) {
163             case 1:
164                 printf("Enter number of nodes: ");
165                 scanf("%d", &n);
166                 createList(n);
167                 break;
168             case 2:
169                 printf("Enter data to insert: ");
170                 scanf("%d", &data);
171                 insertAtBeginning(data);
172                 break;
173             case 3:
174                 printf("Enter data: ");
175                 scanf("%d", &data);
176                 printf("Enter position: ");
177                 scanf("%d", &pos);
178                 insertAtPosition(data, pos);
179                 break;
180             case 4:
181                 printf("Enter data to insert: ");
182                 scanf("%d", &data);
183                 insertAtEnd(data);
184                 break;
185             case 5:
186                 displayList();
187                 break;
188             case 6:
189                 printf("Exiting...\n");
190                 exit(0);
191             default:
192                 printf("Invalid choice. Try again.\n");
193         }
194     }
195     return 0;
196 }
```

```
D:\chehanDP\undlylinkedlist.exe

---- Singly Linked List Operations ----
1. Create linked list
2. Insert at Beginning
3. Insert at any Position
4. Insert at End
5. Display list
6. Exit
Enter your choice: 1
Enter number of nodes: 2
Enter data for node 1: 3
Enter data for node 2: 4
Linked list created successfully

---- Singly Linked List Operations ----
1. Create linked list
2. Insert at Beginning
3. Insert at any Position
4. Insert at End
5. Display list
6. Exit
Enter your choice: 5
Linked list: 3 -> 4 -> NULL

---- Singly Linked List Operations ----
1. Create linked list
2. Insert at Beginning
3. Insert at any Position
4. Insert at End
5. Display list
6. Exit
Enter your choice: 2
Enter data to insert: 344
Node inserted at the beginning

---- Singly Linked List Operations ----
1. Create linked list
2. Insert at Beginning
3. Insert at any Position
4. Insert at End
5. Display list
6. Exit
Enter your choice: 5
Linked list: 344 -> 3 -> 4 -> NULL

---- Singly Linked List Operations ----
1. Create linked list
2. Insert at Beginning
3. Insert at any Position
4. Insert at End
5. Display list
6. Exit
Enter your choice: 4
Enter data to insert: 678
Node inserted at the end

---- Singly Linked List Operations ----
1. Create linked list
2. Insert at Beginning
3. Insert at any Position
```

- 11) write a program to implement ~~linked~~ singly linked list with the following
- create a linked list
  - Insertion at a node at first position \* any position \* end of the list
  - Display the contents of linked list

pseudocode

structure node

data

next  $\rightarrow$  node

end structure

$\Rightarrow$  empty linked list = head  $\leftarrow$  null

$\Rightarrow$  New node create

create node (value)

newnode  $\leftarrow$  allocate memory for node

newnode.data  $\leftarrow$  value

newnode.next  $\leftarrow$  null

return node

end function

$\Rightarrow$  Insert at first position

firstposition (value)

newnode  $\leftarrow$  create node (value)

newnode.next  $\leftarrow$  head

head  $\leftarrow$  newnode

end function

$\Rightarrow$  Insert at end

insertatend (value)

newnode  $\rightarrow$  create node (value)

if head = null then

head  $\leftarrow$  newnode

return



Shot on OnePlus

chE\_Reddy

$\leftarrow$  head

DATE: \_\_\_\_\_

```
while temp.next ≠ null  
temp ← temp.next  
end while  
temp.next ← newnode  
end function
```

⇒ Insert at any position:

Insertatanyposition (targetvalue, newvalue)

temp ← head

while temp ≠ null and temp.data ≠ targetvalue  
temp ← temp.next

end while

If temp = null then

print "target not found"

return

end if

newnode ← create node (newvalue)

newnode.next ← temp.next

temp.next ← newnode

end

⇒ Display

Display ()

temp ← head

while temp ≠ null

print temp.data

temp ← temp.next

end while

end function





```

#include <stdio.h>
#include <stdlib.h>
struct node {
    int data;
    struct node* next;
};
struct node* head = NULL;
void createlist(int n) {
    struct node* newnode, * temp = NULL;
    int data, i;
    if (n <= 0) {
        printf("Number of nodes should be greater than 0\n");
        return;
    }
    if (head != NULL) {
        printf("Warning: Overwriting existing list.\n");
        head = NULL;
    }
    for (i = 1; i <= n; i++) {
        newnode = (struct node*) malloc(sizeof(struct node));
        if (newnode == NULL) {
            printf("Memory allocation failed.\n");
            return;
        }
        printf("Enter data for node %d: ", i);
        scanf("%d", &data);
        newnode->data = data;
        newnode->next = NULL;
        if (head == NULL)
            head = newnode;
        else
            temp->next = newnode;
        temp = newnode;
    }
}

```



DATE: \_\_\_\_\_

```

    printf("\n linked list created \n");
}

void insertatBeginning(int data){
    struct node * newnode = (struct node *) malloc(sizeof(struct node));
    if (newnode == NULL){
        printf("Memory allocation failed \n");
        return;
    }
    newnode->data = data;
    newnode->next = head;
    head = newnode;
    printf("node inserted at the beginning \n");
}

```

```

void insertatend(int data)
{
    struct node * newnode = (struct node *) malloc(sizeof(struct node));
    if (newnode == NULL) {
        printf("Memory allocation failed \n");
        return;
    }
    newnode->data = data;
    newnode->next = NULL;
    if (head == NULL) {
        head = newnode;
    } else {
        struct node * temp = head;
        while (temp->next != NULL)
            temp = temp->next;
        temp->next = newnode;
    }
}

```





```

void insertAtPosition(int data, int pos) {
    int i;
    struct node * newnode, * temp = head;
    if (pos <= 1) {
        printf("Invalid position\n");
        return;
    }
    if (pos == 1) {
        insertAtBeginning(data);
        return;
    }
    for (i = 1; i < pos - 1; i++) {
        if (temp != NULL) {
            temp = temp->next;
        }
        if (temp == NULL) {
            printf("position out of range\n", pos);
            return;
        }
        newnode = (struct node *) malloc(sizeof(struct node));
        if (newnode == NULL) {
            printf("Memory allocation failed\n");
            return;
        }
        newnode->data = data;
        newnode->next = temp->next;
        temp->next = newnode;
        printf("Node inserted at position %d", pos);
    }
}

void displayList() {
    struct node * temp = head;
    if (head == NULL) {
        printf("List is empty\n");
        return;
    }
}

```





```

printf("In linked list:");
while (temp != NULL) {
    printf("%d->", temp->data);
    temp = temp->next;
}
printf("NULL\n");
}

int main() {
    int choice, n, data, pos;
    while (1) {
        printf("\n 1. create linked list\n");
        printf("\n 2. Insert at Beginning\n");
        printf("\n 3. Insert at any position\n");
        printf("\n 4. Insert at End\n");
        printf("\n 5. Display list\n");
        printf("\n 6. Exit\n");
        if (scanf("%d", &choice) != 1) {
            while (getchar() != '\n');
            printf("Invalid input. please enter a number\n");
            continue;
        }
        switch (choice) {
            case 1:
                printf("Enter number of nodes:");
                scanf("%d", &n);
                createList(n);
                break;
            case 2:
                printf("Enter data to insert:");
                scanf("%d", &data);
                insertAtBeginning(data);
                break;

```



CASE 3:-

```
printf("Enter data:");  
scanf("%d", &data);  
printf("Enter position:");  
scanf("%d", &pos);  
insertatposition(data, pos);  
break;
```

CASE 4:-

```
printf("Enter data to insert:");  
scanf("%d", &data);  
insertatend(data);  
break;
```

CASE 5:-

```
displaylist();  
break;
```

CASE 6:-

```
printf("Exiting");  
exit(0);
```

}

}

Output:-

singly linked list operation.

1. create linked list
2. Insert at Beginning
3. Insert at any position
4. Display a list
5. Insert at End
6. Exit.

Enter your choice:-

Enter number of nodes:- 3

Enter data for node 1:- 10

Enter data for node 2:- 20



Enter data for node 3: 30

Enter your choice: 5

10 20 30

Enter your choice: 5

Enter your data: 40

Enter position: 3

Node inserted at position: 3

Enter your choice: 4

Enter data to insert: 50

Node inserted at the end

~~12/1/25~~  
Seen

