

1. Write a program in c to implement insertion in 1-D arrays.

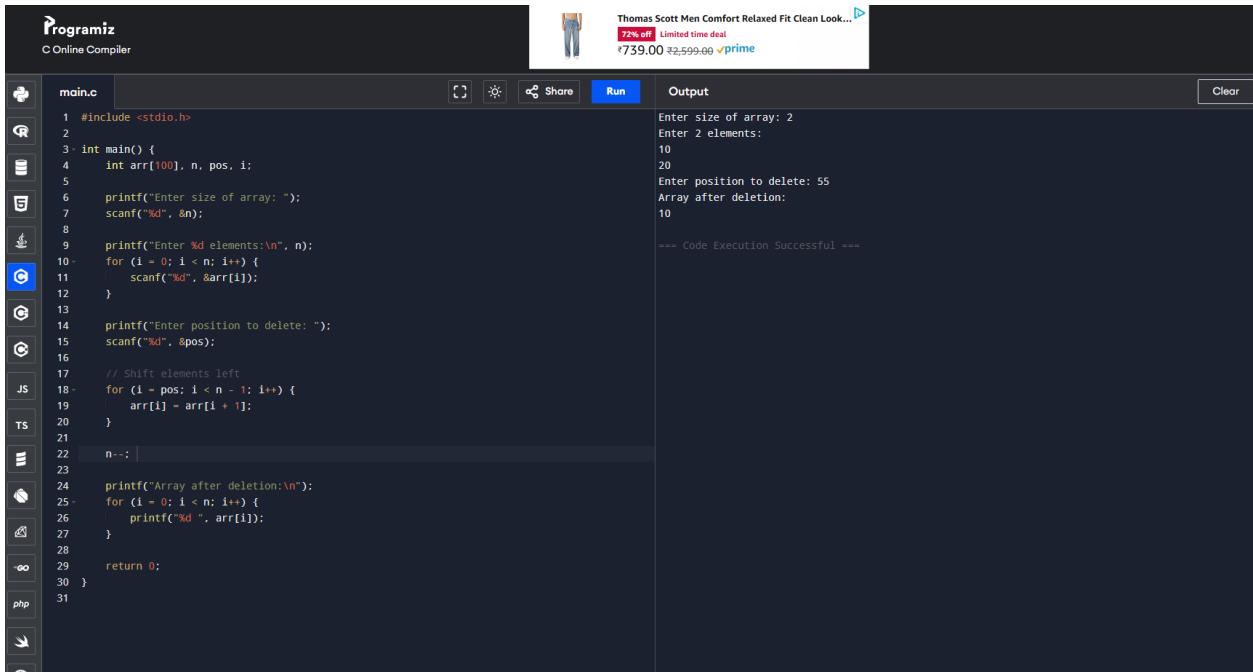
The screenshot shows a C Online Compiler interface on Programiz. The code in the editor is:

```
main.c
1 #include <stdio.h>
2
3 int main() {
4     int arr[100], n, pos, val, i;
5
6     printf("Enter size of array: ");
7     scanf("%d", &n);
8
9     printf("Enter %d elements:\n", n);
10    for (i = 0; i < n; i++) {
11        scanf("%d", &arr[i]);
12    }
13
14    printf("Enter position to insert: ");
15    scanf("%d", &pos);
16
17    printf("Enter value to insert: ");
18    scanf("%d", &val);
19
20    // Shift elements right
21    for (i = n; i > pos; i--) {
22        arr[i] = arr[i - 1];
23    }
24
25    arr[pos] = val;
26    n++;
27
28    printf("Array after insertion:\n");
29    for (i = 0; i < n; i++) {
30        printf("%d ", arr[i]);
31    }
32
33    return 0;
34 }
```

The output window shows the execution results:

```
Enter size of array: 2
Enter 2 elements:
10
20
Enter position to insert: 5
Enter value to insert: 25
Array after insertion:
10 20 0
== Code Execution Successful ==
```

2. Write a program in c to implement deletion in 1-D arrays.



The screenshot shows the Programiz C Online Compiler interface. On the left, the code editor contains a C program named main.c. The code prompts the user for the size of an array, initializes it with 2 elements (10 and 20), and then asks for a position to delete (55). It then prints the array after deletion. The output window on the right shows the execution results: "Enter size of array: 2", "Enter 2 elements: 10 20", "Enter position to delete: 55", "Array after deletion: 10", and "==== Code Execution Successful ===".

```
main.c
1 #include <stdio.h>
2
3 int main() {
4     int arr[100], n, pos, i;
5
6     printf("Enter size of array: ");
7     scanf("%d", &n);
8
9     printf("Enter %d elements:\n", n);
10    for (i = 0; i < n; i++) {
11        scanf("%d", &arr[i]);
12    }
13
14    printf("Enter position to delete: ");
15    scanf("%d", &pos);
16
17    // Shift elements left
18    for (i = pos; i < n - 1; i++) {
19        arr[i] = arr[i + 1];
20    }
21
22    n--;
23
24    printf("Array after deletion:\n");
25    for (i = 0; i < n; i++) {
26        printf("%d ", arr[i]);
27    }
28
29    return 0;
30 }
31
```

3. Write a program in c to implement linear and binary searching in 1-d arrays.

The screenshot shows the Programiz Online Compiler interface. The code editor contains a C program named main.c. The program includes functions for linearSearch and binarySearch, and a main function that prompts the user for array size, elements, and a search key, then chooses between linear or binary search based on user input. The output window shows the execution of the program with an array of size 4 containing 10, 20, 30, 40, and a search for element 78 which is not found.

```
1 #include <stdio.h>
2
3
4 int linearSearch(int arr[], int n, int key) {
5     for (int i = 0; i < n; i++) {
6         if (arr[i] == key)
7             return i;
8     }
9     return -1;
10 }
11
12
13 int binarySearch(int arr[], int n, int key) {
14     int low = 0, high = n - 1, mid;
15
16     while (low <= high) {
17         mid = (low + high) / 2;
18
19         if (arr[mid] == key)
20             return mid;
21         else if (arr[mid] < key)
22             low = mid + 1;
23         else
24             high = mid - 1;
25     }
26
27     return -1;
28 }
29
30 int main() {
31     int arr[100], n, key, choice, result;
32
33     printf("Enter size of array: ");
34     scanf("%d", &n);
35
36     printf("Enter %d elements (for binary search, enter sorted elements):\n", n);
37     for (int i = 0; i < n; i++) {
38         scanf("%d", &arr[i]);
39     }
40
41     printf("Enter element to search: ");
42     scanf("%d", &key);
43
44     printf("Choose search method:\n1. Linear Search\n2. Binary Search\nEnter choice: ");
45     scanf("%d", &choice);
46
47     if (choice == 1)
48         result = linearSearch(arr, n, key);
49     else
50         result = binarySearch(arr, n, key);
51
52     if (result != -1)
53         printf("Element found at position %d\n", result);
54     else
55         printf("Element not found\n");
56
57     return 0;
58 }
```

Output:

```
Enter size of array: 4
Enter 4 elements (for binary search, enter sorted elements):
10
20
30
40
Enter element to search: 78
Choose search method:
1. Linear Search
2. Binary Search
Enter choice: 2
Element not found

== Code Execution Successful ==
```

This screenshot shows a modified version of the C program from the previous one. The main difference is that the array elements are now read from the user before the search begins. The rest of the logic for choosing search methods and performing the search remains the same. The output shows the same execution results as the first screenshot.

```
25     }
26
27     return -1;
28 }
29
30 int main() {
31     int arr[100], n, key, choice, result;
32
33     printf("Enter size of array: ");
34     scanf("%d", &n);
35
36     printf("Enter %d elements (for binary search, enter sorted elements):\n", n);
37     for (int i = 0; i < n; i++) {
38         scanf("%d", &arr[i]);
39     }
40
41     printf("Enter element to search: ");
42     scanf("%d", &key);
43
44     printf("Choose search method:\n1. Linear Search\n2. Binary Search\nEnter choice: ");
45     scanf("%d", &choice);
46
47     if (choice == 1)
48         result = linearSearch(arr, n, key);
49     else
50         result = binarySearch(arr, n, key);
51
52     if (result != -1)
53         printf("Element found at position %d\n", result);
54     else
55         printf("Element not found\n");
56
57     return 0;
58 }
```

Output:

```
Enter size of array: 4
Enter 4 elements (for binary search, enter sorted elements):
10
20
30
40
Enter element to search: 78
Choose search method:
1. Linear Search
2. Binary Search
Enter choice: 2
Element not found

== Code Execution Successful ==
```

4. Write a program in c to implement sorting in 1-D arrays.

The screenshot shows the Programiz C Online Compiler interface. On the left, there's a sidebar with icons for various programming languages: C, C++, Python, Java, JavaScript, TypeScript, Go, PHP, and others. The main area has a dark background with white text. On the left, the code editor contains a C program named 'main.c' that implements bubble sort. On the right, the 'Output' panel shows the execution results: it asks for array size (2), elements (40 and 20), sorts them, and prints the sorted array (20 40). A banner at the top right promotes 'Premium Coding Courses by Programiz'.

```
1 #include <stdio.h>
2
3 int main() {
4     int arr[100], n, i, j, temp;
5
6     printf("Enter size of array: ");
7     scanf("%d", &n);
8
9     printf("Enter %d elements:\n", n);
10    for (i = 0; i < n; i++) {
11        scanf("%d", &arr[i]);
12    }
13
14    // Bubble Sort
15    for (i = 0; i < n - 1; i++) {
16        for (j = 0; j < n - i - 1; j++) {
17            if (arr[j] > arr[j + 1]) {
18                // Swap
19                temp = arr[j];
20                arr[j] = arr[j + 1];
21                arr[j + 1] = temp;
22            }
23        }
24    }
25
26    printf("Sorted array:\n");
27    for (i = 0; i < n; i++) {
28        printf("%d ", arr[i]);
29    }
30
31    return 0;
32 }
33
```

Output

```
Enter size of array: 2
Enter 2 elements:
20
40
Sorted array:
20 40
--- Code Execution Successful ---
```

5. Write a program in c to concatenate two arrays.

The screenshot shows the Programiz Online Compiler interface. On the left, there is a sidebar with icons for various programming languages: C, C++, Java, Python, JavaScript, TypeScript, Go, PHP, and Swift. The main area shows a code editor with the following C code:

```
main.c
1 #include <stdio.h>
2
3 int main() {
4     int a[50], b[50], c[100];
5     int n1, n2, i;
6
7     printf("Enter size of first array: ");
8     scanf("%d", &n1);
9     printf("Enter elements:\n");
10    for (i = 0; i < n1; i++) {
11        scanf("%d", &a[i]);
12        c[i] = a[i];
13    }
14
15
16    printf("Enter size of second array: ");
17    scanf("%d", &n2);
18    printf("Enter elements:\n");
19    for (i = 0; i < n2; i++) {
20        scanf("%d", &b[i]);
21        c[n1 + i] = b[i];
22    }
23
24
25    printf("Concatenated array:\n");
26    for (i = 0; i < n1 + n2; i++) {
27        printf("%d ", c[i]);
28    }
29
30
31    return 0;
32 }
33
```

At the top right, there are buttons for Run, Share, and Output. The Output panel shows the following interaction:

```
Enter size of first array: 3
Enter elements:
1 2 3
Enter size of second array: 2
Enter elements:
4 5
Concatenated array:
1 2 3 4 5
--- Code Execution Successful ---
```

6. Write a program in C to implement the following Operations on 2-D Array (addition; subtraction; multiplication; transpose)

The screenshot shows the Programiz C Online Compiler interface. The code in the editor is for matrix operations. The output window shows the execution of the concatenation operation:

```

main.c
=====
1 #include <stdio.h>
2
3 int main() {
4     int a[10][10], b[10][10], res[10][10];
5     int i, j, r, c, choice;
6
7     printf("Enter number of rows and columns: ");
8     scanf("%d %d", &r, &c);
9
10    printf("Enter elements of first matrix:\n");
11    for (i = 0; i < r; i++) {
12        for (j = 0; j < c; j++) {
13            scanf("%d", &a[i][j]);
14        }
15    }
16    printf("Enter elements of second matrix:\n");
17    for (i = 0; i < r; i++) {
18        for (j = 0; j < c; j++) {
19            scanf("%d", &b[i][j]);
20        }
21    }
22    printf("\n1. Addition\n2. Subtraction\n3. Multiplication\n4. Transpose of First
Matrix\n");
23    printf("Enter your choice: ");
24    scanf("%d", &choice);
25    switch (choice) {
26        case 1: {
27            for (i = 0; i < r; i++) {
28                for (j = 0; j < c; j++) {
29                    res[i][j] = a[i][j] + b[i][j];
30                }
31            }
32        case 2: {
33            for (i = 0; i < r; i++) {
34                for (j = 0; j < c; j++)
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
3 Enter size of first array: 3
Enter elements:
1 2 3
Enter size of second array: 2
Enter elements:
4 5
Concatenated array:
1 2 3 4 5
==== Code Execution Successful ====

```

The screenshot shows the Programiz C Online Compiler interface. The code in the editor is for matrix operations. The output window shows the execution of the multiplication operation:

```

main.c
=====
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
case 3: {
    for (i = 0; i < r; i++) {
        for (j = 0; j < c; j++) {
            res[i][j] = 0;
            for (int k = 0; k < c; k++) {
                res[i][j] += a[i][k] * b[k][j];
            }
        }
    }
    printf("Result (Multiplication):\n");
    break;
}
case 4: {
    printf("Transpose of first matrix:\n");
    for (i = 0; i < c; i++) {
        for (j = 0; j < r; j++) {
            printf("%d ", a[j][i]);
        }
        printf("\n");
    }
    return 0;
}
default: {
    printf("Invalid choice!\n");
    return 1;
}

for (i = 0; i < r; i++) {
    for (j = 0; j < c; j++) {
        printf("%d ", res[i][j]);
    }
    printf("\n");
}
return 0;
}
==== Code Execution Successful ====

```

7. Write a program in C to implement operations on Stack using arrays.

Programiz
C Online Compiler

Premium Coding Courses by Programiz

Output

```
main.c
1 #include <stdio.h>
2 #define SIZE 100
3
4 int stack[SIZE];
5 int top = -1;
6
7 // Push operation
8 void push(int val) {
9     if (top == SIZE - 1) {
10         printf("Stack Overflow\n");
11     } else {
12         top++;
13         stack[top] = val;
14         printf("Pushed %d\n", val);
15     }
16 }
17
18 // Pop operation
19 void pop() {
20     if (top == -1) {
21         printf("Stack Underflow\n");
22     } else {
23         printf("Popped %d\n", stack[top]);
24         top--;
25     }
26 }
27
28 // Display operation
29 void display() {
30     if (top == -1) {
31         printf("Stack is empty\n");
32     } else {
33         printf("Stack elements:\n");
34         for (int i = top; i >= 0; i--) {
35             printf("%d\n", stack[i]);
36         }
37     }
38 }
```

1. Push
2. Pop
3. Display
4. Exit
Enter choice: 3
Stack is empty

1. Push
2. Pop
3. Display
4. Exit
Enter choice: 2
Stack Underflow

1. Push
2. Pop
3. Display
4. Exit
Enter choice: 4

==== Code Execution Successful ===

Programiz
C Online Compiler

Premium Coding Courses by Programiz

Output

```
main.c
31     printf("Stack is empty\n");
32 } else {
33     printf("Stack elements:\n");
34     for (int i = top; i >= 0; i--) {
35         printf("%d\n", stack[i]);
36     }
37 }
38
39 int main() {
40     int choice, value;
41
42     while (1) {
43         printf("\n1. Push\n2. Pop\n3. Display\n4. Exit\nEnter choice: ");
44         scanf("%d", &choice);
45
46         switch (choice) {
47             case 1:
48                 printf("Enter value to push: ");
49                 scanf("%d", &value);
50                 push(value);
51                 break;
52             case 2:
53                 pop();
54                 break;
55             case 3:
56                 display();
57                 break;
58             case 4:
59                 return 0;
60             default:
61                 printf("Invalid choice\n");
62         }
63     }
64 }
65
66 return 0;
67 }
68 }
```

1. Push
2. Pop
3. Display
4. Exit
Enter choice: 3
Stack is empty

1. Push
2. Pop
3. Display
4. Exit
Enter choice: 2
Stack Underflow

1. Push
2. Pop
3. Display
4. Exit
Enter choice: 4

==== Code Execution Successful ===

8. Write a program in c to implement operations on stack using linked list.

The screenshot shows the Programiz C Online Compiler interface. The code editor contains a C program named main.c. The code defines a stack of size 100, performs push, pop, and display operations, and handles stack overflow and underflow cases. The output window shows the execution of the program, starting with a stack display, followed by pushing the value 3, which results in a stack overflow error. Then, popping the stack leads to a stack underflow error. Finally, the program exits. The output concludes with a message indicating successful code execution.

```
1 #include <stdio.h>
2 #define SIZE 100
3
4 int stack[SIZE];
5 int top = -1;
6
7 // Push operation
8 void push(int val) {
9     if (top == SIZE - 1) {
10         printf("Stack Overflow\n");
11     } else {
12         top++;
13         stack[top] = val;
14         printf("Pushed %d\n", val);
15     }
16 }
17
18 // Pop operation
19 void pop() {
20     if (top == -1) {
21         printf("Stack Underflow\n");
22     } else {
23         printf("Popped %d\n", stack[top]);
24         top--;
25     }
26 }
27
28 // Display operation
29 void display() {
30     if (top == -1) {
31         printf("Stack is empty\n");
32     } else {
33         printf("Stack elements:\n");
34         for (int i = top; i >= 0; i--) {
35             printf("%d\n", stack[i]);
36         }
37     }
38 }
```

Output:

```
1. Push
2. Pop
3. Display
4. Exit
Enter choice: 3
Stack is empty

1. Push
2. Pop
3. Display
4. Exit
Enter choice: 2
Stack Underflow

1. Push
2. Pop
3. Display
4. Exit
Enter choice: 4

== Code Execution Successful ==
```

The screenshot shows the Programiz C Online Compiler interface. The code editor contains a modified C program named main.c. This version uses a linked list structure for the stack instead of an array. It includes a main loop that repeatedly prompts the user for a choice (Push, Pop, Display, or Exit). The display operation shows all elements of the stack. The push operation adds a new node at the top. The pop operation removes the top node. The program handles stack underflow by returning 0 from the pop operation. The output window shows the execution of the program, starting with a stack display, followed by pushing the values 1, 2, and 3, then popping them back out. The program then exits. The output concludes with a message indicating successful code execution.

```
31     printf("Stack is empty\n");
32 } else {
33     printf("Stack elements:\n");
34     for (int i = top; i >= 0; i--) {
35         printf("%d\n", stack[i]);
36     }
37 }
38 }

40 int main() {
41     int choice, value;
42
43     while (1) {
44         printf("\n1. Push\n2. Pop\n3. Display\n4. Exit\nEnter choice: ");
45         scanf("%d", &choice);
46
47         switch (choice) {
48             case 1:
49                 printf("Enter value to push: ");
50                 scanf("%d", &value);
51                 push(value);
52                 break;
53             case 2:
54                 pop();
55                 break;
56             case 3:
57                 display();
58                 break;
59             case 4:
60                 return 0;
61             default:
62                 printf("Invalid choice\n");
63         }
64     }
65
66     return 0;
67 }
```

Output:

```
1. Push
2. Pop
3. Display
4. Exit
Enter choice: 3
Stack is empty

1. Push
2. Pop
3. Display
4. Exit
Enter choice: 2
Stack Underflow

1. Push
2. Pop
3. Display
4. Exit
Enter choice: 4

== Code Execution Successful ==
```

9. Write a program in c to implement applications of stack.

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

```
main.c
1 #include <stdio.h>
2 #include <stdlib.h>
3
4
5 struct Node {
6     char data;
7     struct Node* next;
8 };
9
10 struct Node* top = NULL;
11
12
13 void push(char value) {
14     struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
15     if (newNode == NULL) {
16         printf("Stack Overflow\n");
17         return;
18     }
19     newNode->data = value;
20     newNode->next = top;
21     top = newNode;
22 }
23
24
25 char pop() {
26     if (top == NULL) {
27         return -1;
28     }
29     struct Node* temp = top;
30     top = top->next;
31     char popped = temp->data;
32     free(temp);
33     return popped;
34 }
35
36
37 int isBalanced(char* expr) {
38     for (int i = 0; expr[i] != '\0'; i++) {
```

The output window shows the result of running the program with the input "((a+b)*(c/d))". The output is:

```
Enter an expression with parentheses: ((a+b)*(c/d))
The parentheses are balanced.

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

Below the main code editor, there is another instance of the same code, likely a continuation or a different run. The output for this instance is identical.

```
main.c
31     char popped = temp->data;
32     free(temp);
33     return popped;
34 }
35
36
37 int isBalanced(char* expr) {
38     for (int i = 0; expr[i] != '\0'; i++) {
39         if (expr[i] == '(' || expr[i] == '[' || expr[i] == '{') {
40             push(expr[i]);
41         } else if (expr[i] == ')' || expr[i] == ']' || expr[i] == '}') {
42             char topChar = pop();
43             if ((expr[i] == ')') && topChar != '(') ||
44                 (expr[i] == ']' && topChar != '[') ||
45                 (expr[i] == '}' && topChar != '{')) {
46                 return 0; // Not balanced
47             }
48         }
49     }
50     return top == NULL;
51 }
52
53
54 int main() {
55     char expr[100];
56
57     printf("Enter an expression with parentheses: ");
58     scanf("%s", expr);
59
60     if (isBalanced(expr)) {
61         printf("The parentheses are balanced.\n");
62     } else {
63         printf("The parentheses are not balanced.\n");
64     }
65
66     return 0;
67 }
68
```

10. Write a program in c to implement operation on queue using stack.

Programiz
C Online Compiler

Premium Coding Courses by Programiz

Run Output Clear

```
main.c
1 #include <stdio.h>
2 #include <stdlib.h>
3
4
5 struct Node {
6     int data;
7     struct Node* next;
8 };
9
10 struct Node* stack1 = NULL;
11 struct Node* stack2 = NULL;
12
13
14 void push(struct Node** top, int value) {
15     struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
16     if (newNode == NULL) {
17         printf("Memory error\n");
18         return;
19     }
20     newNode->data = value;
21     newNode->next = *top;
22     *top = newNode;
23 }
24
25
26 int pop(struct Node** top) {
27     if (*top == NULL) {
28         return -1;
29     }
30     struct Node* temp = *top;
31     *top = (*top)->next;
32     int popped = temp->data;
33     free(temp);
34     return popped;
35 }
36
37 // Enqueue operation (push to stack1)
38 void enqueue(int value) {
```

1. Enqueue
2. Dequeue
3. Display Queue
4. Exit
Enter choice: 1
Enter value to enqueue: 5
Enqueued 5

1. Enqueue
2. Dequeue
3. Display Queue
4. Exit
Enter choice: 4

*** Code Execution Successful ***

Programiz
C Online Compiler

Premium Coding Courses by Programiz

Run Output Clear

```
main.c
1 // Enqueue operation (push to stack1)
2 void enqueue(int value) {
3     push(&stack1, value);
4     printf("Enqueued %d\n", value);
5 }
6
7 // Dequeue operation (pop from stack2)
8 int dequeue() {
9     if (stack1 == NULL && stack2 == NULL) {
10         printf("Queue is empty\n");
11         return -1;
12     }
13
14     // If stack2 is empty, move elements from stack1 to stack2
15     if (stack2 == NULL) {
16         while (stack1 != NULL) {
17             int value = pop(&stack1);
18             push(&stack2, value);
19         }
20     }
21
22     return pop(&stack2); // Pop from stack2 (FIFO order)
23 }
24
25 // Display operation (for debugging)
26 void displayQueue() {
27     if (stack1 == NULL && stack2 == NULL) {
28         printf("Queue is empty\n");
29         return;
30     }
31
32     struct Node* temp = stack2;
33     printf("Queue elements: ");
34     while (temp != NULL) {
35         printf("%d ", temp->data);
36         temp = temp->next;
37     }
38 }
```

1. Enqueue
2. Dequeue
3. Display Queue
4. Exit
Enter choice: 1
Enter value to enqueue: 5
Enqueued 5

1. Enqueue
2. Dequeue
3. Display Queue
4. Exit
Enter choice: 4

*** Code Execution Successful ***

Programiz

C Online Compiler

Premium Coding Courses by Programiz

Share Run Output Clear

```
main.c
    temp = temp->next;
79 }
80
81     printf("\n");
82 }
83
84- int main() {
85     int choice, value;
86
87-     while () {
88         printf("\n1. Enqueue\n2. Dequeue\n3. Display Queue\n4. Exit\nEnter choice: ");
89         scanf("%d", &choice);
90
91-         switch (choice) {
92             case 1:
93                 printf("Enter value to enqueue: ");
94                 scanf("%d", &value);
95                 enqueue(value);
96                 break;
97
98             case 2:
99                 value = dequeue();
100                if (value != -1) {
101                    printf("Dequeued %d\n", value);
102                }
103                break;
104            case 3:
105                displayQueue();
106                break;
107            case 4:
108                return 0;
109            default:
110                printf("Invalid choice\n");
111        }
112
113    return 0;
114 }
115 }
```

Output

1. Enqueue
2. Dequeue
3. Display Queue
4. Exit
Enter choice: 1
Enter value to enqueue: 5
Enqueued 5
1. Enqueue
2. Dequeue
3. Display Queue
4. Exit
Enter choice: 4

== Code Execution Successful ==

11. Write a program in C to implement operations on queue using linked list simple code

Programiz
C Online Compiler

Edit PDFs
effortlessly.

Start free trial

Adobe Acrobat

main.c

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4
5 struct Node {
6     int data;
7     struct Node* next;
8 };
9
10 struct Node* front = NULL;
11 struct Node* rear = NULL;
12
13
14 void enqueue(int value) {
15     struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
16     if (newNode == NULL) {
17         printf("Memory error\n");
18         return;
19     }
20     newNode->data = value;
21     newNode->next = NULL;
22
23     if (rear == NULL) {
24         front = rear = newNode;
25     } else {
26         rear->next = newNode;
27         rear = newNode;
28     }
29     printf("Enqueued %d\n", value);
30 }
31
32
33 int dequeue() {
34     if (front == NULL) {
35         printf("Queue is empty\n");
36         return -1;
37     }
38     struct Node* temp = front;
```

Output

```
1. Enqueue
2. Dequeue
3. Display Queue
4. Exit
Enter choice: 1
Enter value to enqueue: 10
Enqueued 10

Enter choice: 1
Enter value to enqueue: 20
Enqueued 20

Enter choice: 3
Queue elements: 10 20

Enter choice: 2
Dequeued 10

Enter choice: 3
Queue elements: 20

Enter choice: 2
Dequeued 20

Enter choice: 3
Queue is empty

Enter choice: 4
```

Clear

Programiz
C Online Compiler

Premium Coding
Courses by Programiz

Programiz PRO

main.c

```
38     struct Node* temp = front;
39     int value = temp->data;
40     front = front->next;
41     if (front == NULL) {
42         rear = NULL;
43     }
44     free(temp);
45     return value;
46 }
47
48
49 void display() {
50     if (front == NULL) {
51         printf("Queue is empty\n");
52         return;
53     }
54     struct Node* temp = front;
55     printf("Queue elements: ");
56     while (temp != NULL) {
57         printf("%d ", temp->data);
58         temp = temp->next;
59     }
60     printf("\n");
61 }
62
63 int main() {
64     int choice, value;
65
66     while () {
67         printf("\n1. Enqueue\n2. Dequeue\n3. Display Queue\n4. Exit\nEnter choice: ");
68         scanf("%d", &choice);
69
70         switch (choice) {
71             case 1:
72                 printf("Enter value to enqueue: ");
73                 scanf("%d", &value);
74                 enqueue(value);
75             case 2:
```

Output

```
1. Enqueue
2. Dequeue
3. Display Queue
4. Exit
Enter choice: 1
Enter value to enqueue: 10
Enqueued 10

Enter choice: 1
Enter value to enqueue: 20
Enqueued 20

Enter choice: 3
Queue elements: 10 20

Enter choice: 2
Dequeued 10

Enter choice: 3
Queue elements: 20

Enter choice: 2
Dequeued 20

Enter choice: 3
Queue is empty

Enter choice: 4
```

Clear

Programiz
C Online Compiler

Premium Coding Courses by Programiz

Share Run Output Clear

```
main.c
57     printf("%d", temp->data);
58     temp = temp->next;
59 }
60 printf("\n");
61 }
62
63 int main() {
64     int choice, value;
65
66     while (1) {
67         printf("\n1. Enqueue\n2. Dequeue\n3. Display Queue\n4. Exit\nEnter choice: ");
68         scanf("%d", &choice);
69
70         switch (choice) {
71             case 1:
72                 printf("Enter value to enqueue: ");
73                 scanf("%d", &value);
74                 enqueue(value);
75                 break;
76             case 2:
77                 value = dequeue();
78                 if (value != -1) {
79                     printf("Dequeued %d\n", value);
80                 }
81                 break;
82             case 3:
83                 display();
84                 break;
85             case 4:
86                 return 0;
87             default:
88                 printf("Invalid choice\n");
89         }
90     }
91
92     return 0;
93 }
```

1. Enqueue
2. Dequeue
3. Display Queue
4. Exit
Enter choice: 1
Enter value to enqueue: 10
Enqueued 10

Enter choice: 1
Enter value to enqueue: 20
Enqueued 20

Enter choice: 2
Dequeued 10

Enter choice: 3
Queue elements: 10 20

Enter choice: 2
Dequeued 20

Enter choice: 3
Queue is empty

Enter choice: 4

12. Write a program in C to implement operations on circular queue using array

Programiz
C Online Compiler

main.c

```
1 #include <stdio.h>
2 #define SIZE 5
3
4 int queue[SIZE];
5 int front = -1, rear = -1;
6
7
8 void enqueue(int value) {
9     if ((front == 0 && rear == SIZE - 1) || (rear + 1 == front)) {
10        printf("Queue is full\n");
11        return;
12    }
13
14    if (front == -1) {
15        front = rear = 0;
16    } else if (rear == SIZE - 1 && front != 0) {
17        rear = 0;
18    } else {
19        rear++;
20    }
21
22    queue[rear] = value;
23    printf("Enqueued %d\n", value);
24 }
25
26
27 void dequeue() {
28    if (front == -1) {
29        printf("Queue is empty\n");
30        return;
31    }
32
33    printf("Dequeued %d\n", queue[front]);
34
35    if (front == rear) {
36        front = rear = -1;
37    } else if (front == SIZE - 1) {
38        front = 0;
39    } else {
40        front++;
41    }
42 }
43
44
45 void display() {
46    if (front == -1) {
47        printf("Queue is empty\n");
48        return;
49    }
50
51    printf("Queue elements: ");
52    int i = front;
53    while () {
54        printf("%d ", queue[i]);
55        if (i == rear)
56            break;
57        i = (i + 1) % SIZE;
58    }
59    printf("\n");
60 }
61
62 int main() {
63    int choice, value;
64
65    while () {
66        printf("\n1. Enqueue\n2. Dequeue\n3. Display\n4. Exit\nEnter choice: ");
67        scanf("%d", &choice);
68
69        switch (choice) {
70            case 1:
71                printf("Enter value to enqueue: ");
72                scanf("%d", &value);
73                enqueue(value);
74                break;
75            case 2:
76                dequeue();
77        }
78    }
79 }
```

Run Output

1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter choice: 1
Enter value to enqueue: 10
Enqueued 10

Enter choice: 1
Enter value to enqueue: 20
Enqueued 20

Enter choice: 3
Queue elements: 10 20

Enter choice: 2
Dequeued 10

Enter choice: 3
Queue elements: 20

Enter choice: 4

Clear

Programiz
C Online Compiler

Premium Coding Courses by Programiz

main.c

```
39    } else {
40        front++;
41    }
42 }
43
44
45 void display() {
46    if (front == -1) {
47        printf("Queue is empty\n");
48        return;
49    }
50
51    printf("Queue elements: ");
52    int i = front;
53    while () {
54        printf("%d ", queue[i]);
55        if (i == rear)
56            break;
57        i = (i + 1) % SIZE;
58    }
59    printf("\n");
60 }
61
62 int main() {
63    int choice, value;
64
65    while () {
66        printf("\n1. Enqueue\n2. Dequeue\n3. Display\n4. Exit\nEnter choice: ");
67        scanf("%d", &choice);
68
69        switch (choice) {
70            case 1:
71                printf("Enter value to enqueue: ");
72                scanf("%d", &value);
73                enqueue(value);
74                break;
75            case 2:
76                dequeue();
77        }
78    }
79 }
```

Run Output

1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter choice: 1
Enter value to enqueue: 10
Enqueued 10

Enter choice: 1
Enter value to enqueue: 20
Enqueued 20

Enter choice: 3
Queue elements: 10 20

Enter choice: 2
Dequeued 10

Enter choice: 3
Queue elements: 20

Enter choice: 4

Clear

Programiz
C Online Compiler

Premium Coding
Courses by Programiz

Share Run Clear

```
main.c
53-     while (1) {
54-         printf("%d ", queue[i]);
55-         if (i == rear)
56-             break;
57-         i = (i + 1) % SIZE;
58-     }
59-     printf("\n");
60 }

62- int main() {
63-     int choice, value;
64

65-     while () {
66-         printf("\n1. Enqueue\n2. Dequeue\n3. Display\n4. Exit\nEnter choice: ");
67-         scanf("%d", &choice);
68

69-         switch (choice) {
70-             case 1:
71-                 printf("Enter value to enqueue: ");
72-                 scanf("%d", &value);
73-                 enqueue(value);
74-                 break;
75-             case 2:
76-                 dequeue();
77-                 break;
78-             case 3:
79-                 display();
80-                 break;
81-             case 4:
82-                 return 0;
83-             default:
84-                 printf("Invalid choice\n");
85-         }
86     }
87
88     return 0;
89 }

90 }
```

Output

```
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter choice: 1
Enter value to enqueue: 10
Enqueued 10

Enter choice: 1
Enter value to enqueue: 20
Enqueued 20

Enter choice: 3
Queue elements: 10 20

Enter choice: 2
Dequeued 10

Enter choice: 3
Queue elements: 20

Enter choice: 4
```

13. Write a program in C to implement insertion in a linked list(beg; mid; end)

Programiz
C Online Compiler

Ad removed [Details](#)

main.c

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4
5 struct Node {
6     int data;
7     struct Node* next;
8 };
9
10 struct Node* head = NULL;
11
12
13 void insertAtBeginning(int value) {
14     struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
15     newNode->data = value;
16     newNode->next = head;
17     head = newNode;
18     printf("Inserted %d at beginning\n", value);
19 }
20
21
22 void insertAtEnd(int value) {
23     struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
24     newNode->data = value;
25     newNode->next = NULL;
26
27     if (head == NULL) {
28         head = newNode;
29     } else {
30         struct Node* temp = head;
31         while (temp->next != NULL)
32             temp = temp->next;
33         temp->next = newNode;
34     }
35     printf("Inserted %d at end\n", value);
36 }
37
38
```

Output

```
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Display
5. Exit
Enter your choice: 1
Enter value: 10
Inserted 10 at beginning

Enter your choice: 2
Enter value: 30
Inserted 30 at end

Enter your choice: 3
Enter value and position: 20 1
Inserted 20 after position 1

Enter your choice: 4
Linked List: 10 -> 20 -> 30 -> NULL
```

Clear

Programiz
C Online Compiler

SHARPEN YOUR SKILLS WITH A
MASTER OF DATA SCIENCE  [Discover More](#)

main.c

```
39 void insertAtPosition(int value, int position) {
40     struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
41     newNode->data = value;
42
43     struct Node* temp = head;
44     for (int i = 1; i < position && temp != NULL; i++) {
45         temp = temp->next;
46     }
47
48     if (temp == NULL) {
49         printf("Position not found.\n");
50         return;
51     }
52
53     newNode->next = temp->next;
54     temp->next = newNode;
55     printf("Inserted %d after position %d\n", value, position);
56 }
57
58
59 void display() {
60     struct Node* temp = head;
61     printf("Linked List: ");
62     while (temp != NULL) {
63         printf("%d -> ", temp->data);
64         temp = temp->next;
65     }
66     printf("NULL\n");
67 }
68
69 int main() {
70     int choice, value, position;
71
72     while (1) {
73         printf("\n1. Insert at Beginning\n2. Insert at End\n3. Insert at Position\n4. Display\n5. Exit\nEnter your choice: ");
74         scanf("%d", &choice);
75     }
76 }
```

Output

```
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Display
5. Exit
Enter your choice: 1
Enter value: 10
Inserted 10 at beginning

Enter your choice: 2
Enter value: 30
Inserted 30 at end

Enter your choice: 3
Enter value and position: 20 1
Inserted 20 after position 1

Enter your choice: 4
Linked List: 10 -> 20 -> 30 -> NULL

Enter your choice: 5
Linked List: 10 -> 20 -> 30 -> NULL
```

Clear

Programiz
C Online Compiler

Premium Coding Courses by Programiz

1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Display
5. Exit

Enter your choice: 1
Enter value: 10
Inserted 10 at beginning

Enter your choice: 2
Enter value: 30
Inserted 30 at end

Enter your choice: 3
Enter value and position: 20 1
Inserted 20 after position 1

Enter your choice: 4
Linked List: 10 -> 20 -> 30 -> NULL

```
main.c
68
69 int main() {
70     int choice, value, position;
71
72     while (1) {
73         printf("1. Insert at Beginning\n2. Insert at End\n3. Insert at Position\n4. Display\n5. Exit\n");
74         scanf("%d", &choice);
75
76         switch (choice) {
77             case 1:
78                 printf("Enter value: ");
79                 scanf("%d", &value);
80                 insertAtBeginning(value);
81                 break;
82             case 2:
83                 printf("Enter value: ");
84                 scanf("%d", &value);
85                 insertAtEnd(value);
86                 break;
87             case 3:
88                 printf("Enter value and position: ");
89                 scanf("%d %d", &value, &position);
90                 insertAtPosition(value, position);
91                 break;
92             case 4:
93                 display();
94                 break;
95             case 5:
96                 return 0;
97             default:
98                 printf("Invalid choice\n");
99         }
100    }
101
102    return 0;
103 }
```

14. Write a program in C to implement deletion from a linked list(beg; mid; end).

Programiz
C Online Compiler

Premium Coding Courses by Programiz

Share Run Output Clear

```
main.c
1 #include <stdio.h>
2 #include <stdlib.h>
3
4
5 struct Node {
6     int data;
7     struct Node* next;
8 };
9
10 struct Node* head = NULL;
11
12
13 void insertAtEnd(int value) {
14     struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
15     newNode->data = value;
16     newNode->next = NULL;
17
18     if (head == NULL)
19         head = newNode;
20     else {
21         struct Node* temp = head;
22         while (temp->next != NULL)
23             temp = temp->next;
24         temp->next = newNode;
25     }
26 }
27
28
29 void deleteFromBeginning() {
30     if (head == NULL) {
31         printf("List is empty\n");
32         return;
33     }
34     struct Node* temp = head;
35     head = head->next;
36     printf("Deleted %d from beginning\n", temp->data);
37     free(temp);
38 }
```

Output

- 1. Delete from Beginning
- 2. Delete from End
- 3. Delete from Position
- 4. Display
- 5. Exit

Enter your choice: 4
Linked List: 10 -> 20 -> 30 -> 40 -> NULL

Enter your choice: 1
Deleted 10 from beginning

Enter your choice: 2
Deleted 40 from end

Enter your choice: 3
Enter position to delete: 2
Deleted 30 from position 2

Enter your choice: 4
Linked List: 20 -> NULL

Programiz
C Online Compiler

Premium Coding Courses by Programiz

Share Run Output Clear

```
main.c
38 }
39
40
41 void deleteFromEnd() {
42     if (head == NULL) {
43         printf("List is empty\n");
44         return;
45     }
46     struct Node* temp = head;
47     struct Node* prev = NULL;
48
49     while (temp->next != NULL) {
50         prev = temp;
51         temp = temp->next;
52     }
53
54     if (prev == NULL)
55         head = NULL;
56     else
57         prev->next = NULL;
58
59     printf("Deleted %d from end\n", temp->data);
60     free(temp);
61 }
62
63
64 void deleteFromPosition(int position) {
65     if (head == NULL) {
66         printf("List is empty\n");
67         return;
68     }
69
70     if (position == 1) {
71         deleteFromBeginning();
72         return;
73     }
74 }
```

Output

- 1. Delete from Beginning
- 2. Delete from End
- 3. Delete from Position
- 4. Display
- 5. Exit

Enter your choice: 4
Linked List: 10 -> 20 -> 30 -> 40 -> NULL

Enter your choice: 1
Deleted 10 from beginning

Enter your choice: 2
Deleted 40 from end

Enter your choice: 3
Enter position to delete: 2
Deleted 30 from position 2

Enter your choice: 4
Linked List: 20 -> NULL

Programiz
C Online Compiler

Premium Coding
Courses by Programiz

Share Run Output Clear

```
main.c
1. Delete from Beginning
2. Delete from End
3. Delete from Position
4. Display
5. Exit
Enter your choice: 4
Linked List: 10 -> 20 -> 30 -> 40 -> NULL

Enter your choice: 1
Deleted 10 from beginning

Enter your choice: 2
Deleted 40 from end

Enter your choice: 3
Enter position to delete: 2
Deleted 30 from position 2

Enter your choice: 4
Linked List: 20 -> NULL
```

JS TS GO PHP

```
int main() {
    int choice, value, position;
    insertAtEnd(10);
    insertAtEnd(20);
    insertAtEnd(30);
    insertAtEnd(40);

    while (1) {
        printf("1. Delete from Beginning\n2. Delete from End\n3. Delete from Position\n4. Display\n5. Exit\nEnter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                deleteFromBeginning();
                break;
            case 2:
                deleteFromEnd();
                break;
            case 3:
                printf("Enter position to delete: ");
                scanf("%d", &position);
                deleteFromPosition(position);
                break;
            case 4:
                display();
                break;
            case 5:
                return 0;
            default:
                printf("Invalid choice\n");
        }
    }
    return 0;
}
```

Programiz
C Online Compiler

SHARPEN YOUR SKILLS

Mobile University

Discover More

Run Output Clear

```
main.c
1. Delete from Beginning
2. Delete from End
3. Delete from Position
4. Display
5. Exit
Enter your choice: 4
Linked List: 10 -> 20 -> 30 -> 40 -> NULL

Enter your choice: 1
Deleted 10 from beginning

Enter your choice: 2
Deleted 40 from end

Enter your choice: 3
Enter position to delete: 2
Deleted 30 from position 2

Enter your choice: 4
Linked List: 20 -> NULL
```

JS TS GO PHP

```
int main() {
    int choice, value, position;
    insertAtEnd(10);
    insertAtEnd(20);
    insertAtEnd(30);
    insertAtEnd(40);

    while (1) {
        printf("1. Delete from Beginning\n2. Delete from End\n3. Delete from Position\n4. Display\n5. Exit\nEnter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                deleteFromBeginning();
                break;
            case 2:
                deleteFromEnd();
                break;
            case 3:
                printf("Enter position to delete: ");
                scanf("%d", &position);
                deleteFromPosition(position);
                break;
            case 4:
                display();
                break;
            case 5:
                return 0;
            default:
                printf("Invalid choice\n");
        }
    }
    return 0;
}
```

15. Write a program in C to implement insertion in a circular linked list(beg; mid; end)

Programiz
C Online Compiler

Premium Coding Courses by Programiz

Run

Output

```
main.c
1 #include <stdio.h>
2 #include <stdlib.h>
3
4- struct Node {
5     int data;
6     struct Node* next;
7 };
8
9 struct Node* last = NULL;
10
11
12- void display() {
13-     if (last == NULL) {
14-         printf("List is empty\n");
15-         return;
16     }
17
18     struct Node* temp = last->next;
19     printf("Circular Linked List: ");
20-    do {
21-        printf("%d -> ", temp->data);
22-        temp = temp->next;
23-    } while (temp != last->next);
24     printf("(head)\n");
25 }
26
27
28- void insertAtBeginning(int value) {
29-     struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
30     newNode->data = value;
31
32-     if (last == NULL) {
33-         newNode->next = newNode;
34-         last = newNode;
35-     } else {
36-         newNode->next = last->next;
37-         last->next = newNode;
38     }

```

1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Display
5. Exit
Enter your choice: 1
Enter value: 10
Inserted 10 at beginning

Enter your choice: 2
Enter value: 30
Inserted 30 at end

Enter your choice: 3
Enter value and position: 20 1
Inserted 20 after position 1

Enter your choice: 4
Circular Linked List: 10 -> 20 -> 30 -> (head)

Enter your choice: 5

Clear

Programiz
C Online Compiler

Premium Coding Courses by Programiz

Run

Output

```
main.c
1 #include <stdio.h>
2 #include <stdlib.h>
3
4- struct Node {
5     int data;
6     struct Node* next;
7 };
8
9 struct Node* last = NULL;
10
11
12- void display() {
13-     if (last == NULL) {
14-         printf("List is empty\n");
15-         return;
16     }
17
18     struct Node* temp = last->next;
19     printf("Circular Linked List: ");
20-    do {
21-        printf("%d -> ", temp->data);
22-        temp = temp->next;
23-    } while (temp != last->next);
24     printf("(head)\n");
25 }
26
27
28- void insertAtBeginning(int value) {
29-     struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
30     newNode->data = value;
31
32-     if (last == NULL) {
33-         newNode->next = newNode;
34-         last = newNode;
35-     } else {
36-         newNode->next = last->next;
37-         last->next = newNode;
38     }

```

1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Display
5. Exit
Enter your choice: 1
Enter value: 10
Inserted 10 at beginning

Enter your choice: 2
Enter value: 30
Inserted 30 at end

Enter your choice: 3
Enter value and position: 20 1
Inserted 20 after position 1

Enter your choice: 4
Circular Linked List: 10 -> 20 -> 30 -> (head)

Enter your choice: 5

Clear

Programiz
C Online Compiler

Premium Coding
Courses by Programiz

Programiz PRO

main.c

```
82
83 int main() {
84     int choice, value, position;
85
86     while (1) {
87         printf("\n1. Insert at Beginning\n2. Insert at End\n3. Insert at Position\n4. Display\n5. Exit\n");
88         scanf("%d", &choice);
89
90         switch (choice) {
91             case 1:
92                 printf("Enter value: ");
93                 scanf("%d", &value);
94                 insertAtBeginning(value);
95                 break;
96             case 2:
97                 printf("Enter value: ");
98                 scanf("%d", &value);
99                 insertAtEnd(value);
100                break;
101            case 3:
102                printf("Enter value and position: ");
103                scanf("%d %d", &value, &position);
104                insertAtPosition(value, position);
105                break;
106            case 4:
107                display();
108                break;
109            case 5:
110                return 0;
111            default:
112                printf("Invalid choice\n");
113        }
114    }
115
116    return 0;
117 }
```

Run

Output

1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Display
5. Exit

Enter your choice: 1
Enter value: 10
Inserted 10 at beginning

Enter your choice: 2
Enter value: 30
Inserted 30 at end

Enter your choice: 3
Enter value and position: 20 1
Inserted 20 after position 1

Enter your choice: 4
Circular Linked List: 10 -> 20 -> 30 -> (head)

Enter your choice: 5

16. Write a program in C to implement deletion from a circular linked list(beg; mid; end) queue.

Programiz
C Online Compiler

Premium Coding
Courses by Programiz

Programiz PRO

main.c

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 struct Node {
5     int data;
6     struct Node* next;
7 };
8
9 struct Node* last = NULL;
10
11
12 void insertAtEnd(int value) {
13     struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
14     newNode->data = value;
15     if (last == NULL) {
16         newNode->next = newNode;
17         last = newNode;
18     } else {
19         newNode->next = last->next;
20         last->next = newNode;
21         last = newNode;
22     }
23 }
24
25
26 void display() {
27     if (last == NULL) {
28         printf("List is empty\n");
29         return;
30     }
31     struct Node* temp = last->next;
32     printf("Circular Linked List: ");
33     do {
34         printf("%d -> ", temp->data);
35         temp = temp->next;
36     } while (temp != last->next);
37     printf("(head)\n");
38 }
```

Run

Output

```
1. Delete From Beginning
2. Delete From End
3. Delete From Position
4. Display
5. Exit
Enter your choice: 4
Circular Linked List: 10 -> 20 -> 30 -> 40 -> (head)

Enter your choice: 1
Deleted 10 from beginning

Enter your choice: 2
Deleted 40 from end

Enter your choice: 3
Enter position to delete: 2
Deleted 30 from position 2

Enter your choice: 4
Circular Linked List: 20 -> (head)
```

Clear

Programiz
C Online Compiler

Premium Coding
Courses by Programiz

Programiz PRO

main.c

```
38 }
39
40
41 void deleteFromBeginning() {
42     if (last == NULL) {
43         printf("List is empty\n");
44         return;
45     }
46
47     struct Node* temp = last->next;
48
49     if (last == last->next) { // Only one node
50         printf("Deleted %d from beginning\n", temp->data);
51         free(temp);
52         last = NULL;
53     } else {
54         last->next = temp->next;
55         printf("Deleted %d from beginning\n", temp->data);
56         free(temp);
57     }
58 }
59
60
61 void deleteFromEnd() {
62     if (last == NULL) {
63         printf("List is empty\n");
64         return;
65     }
66
67     struct Node* temp = last->next;
68     if (last == last->next) { // Only one node
69         printf("Deleted %d from end\n", last->data);
70         free(last);
71         last = NULL;
72     } else {
73         struct Node* prev = NULL;
74         while (temp->next != last->next) {
```

Run

Output

```
1. Delete From Beginning
2. Delete from End
3. Delete from Position
4. Display
5. Exit
Enter your choice: 4
Circular Linked List: 10 -> 20 -> 30 -> 40 -> (head)

Enter your choice: 1
Deleted 10 from beginning

Enter your choice: 2
Deleted 40 from end

Enter your choice: 3
Enter position to delete: 2
Deleted 30 from position 2

Enter your choice: 4
Circular Linked List: 20 -> (head)
```

Clear

Programiz
C Online Compiler

Premium Coding Courses by Programiz

Programiz PRO

```
main.c
13     struct Node* prev = NULL;
14     while (temp->next != last->next) {
15         prev = temp;
16         temp = temp->next;
17     }
18     prev->next = last->next;
19     printf("Deleted %d from end\n", last->data);
20     free(last);
21     last = prev;
22 }
23
24 void deleteFromPosition(int position) {
25     if (last == NULL) {
26         printf("List is empty\n");
27         return;
28     }
29     if (position == 1) {
30         deleteFromBeginning();
31         return;
32     }
33     struct Node* temp = last->next;
34     struct Node* prev = NULL;
35     for (int i = 1; i < position && temp != last; i++) {
36         prev = temp;
37         temp = temp->next;
38     }
39     if (temp == last && position != 1) {
40         deleteFromEnd();
41     } else if (temp != last->next) {
42         prev->next = temp->next;
43         printf("Deleted %d from position %d\n", temp->data, position);
44         free(temp);
45     } else {
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83 }
```

Output

```
1. Delete from Beginning
2. Delete from End
3. Delete from Position
4. Display
5. Exit
Enter your choice: 4
Circular Linked List: 10 -> 20 -> 30 -> 40 -> (head)

Enter your choice: 1
Deleted 10 from beginning

Enter your choice: 2
Deleted 40 from end

Enter your choice: 3
Enter position to delete: 2
Deleted 30 from position 2

Enter your choice: 4
Circular Linked List: 20 -> (head)
```

Programiz
C Online Compiler

Premium Coding Courses by Programiz

Programiz PRO

```
main.c
116 int main() {
117     int choice, position;
118
119     insertAtEnd(10);
120     insertAtEnd(20);
121     insertAtEnd(30);
122     insertAtEnd(40);
123
124     while () {
125         printf("\n1. Delete from Beginning\n2. Delete from End\n3. Delete from Position\n4.
126             Enter choice: ");
127         scanf("%d", &choice);
128
129         switch (choice) {
130             case 1:
131                 deleteFromBeginning();
132                 break;
133             case 2:
134                 deleteFromEnd();
135                 break;
136             case 3:
137                 printf("Enter position to delete: ");
138                 scanf("%d", &position);
139                 deleteFromPosition(position);
140                 break;
141             case 4:
142                 display();
143                 break;
144             case 5:
145                 return 0;
146             default:
147                 printf("Invalid choice\n");
148         }
149     }
150     return 0;
151 }
```

Output

```
1. Delete from Beginning
2. Delete from End
3. Delete from Position
4. Display
5. Exit
Enter your choice: 4
Circular Linked List: 10 -> 20 -> 30 -> 40 -> (head)

Enter your choice: 1
Deleted 10 from beginning

Enter your choice: 2
Deleted 40 from end

Enter your choice: 3
Enter position to delete: 2
Deleted 30 from position 2

Enter your choice: 4
Circular Linked List: 20 -> (head)
```

17. Write a program in C to implement insertion in a doubly linked list(beg; mid; end)

Programiz
C Online Compiler

Premium Coding Courses by Programiz

Run

Output

```
main.c
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 struct Node {
5     int data;
6     struct Node* prev;
7     struct Node* next;
8 };
9
10 struct Node* head = NULL;
11
12 void display() {
13     struct Node* temp = head;
14     printf("Doubly Linked List: ");
15     while (temp != NULL) {
16         printf("%d <-> ", temp->data);
17         temp = temp->next;
18     }
19     printf("NULL\n");
20 }
21
22
23 void insertAtBeginning(int value) {
24     struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
25     newNode->data = value;
26     newNode->prev = NULL;
27     newNode->next = head;
28
29     if (head != NULL)
30         head->prev = newNode;
31
32     head = newNode;
33
34     printf("Inserted %d at beginning\n", value);
35 }
36
37
38 void insertAtEnd(int value) {
39     struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
40     newNode->data = value;
41     newNode->next = NULL;
42
43     if (head == NULL) {
44         newNode->prev = NULL;
45         head = newNode;
46         printf("Inserted %d at end\n", value);
47         return;
48     }
49
50     struct Node* temp = head;
51     while (temp->next != NULL)
52         temp = temp->next;
53
54     temp->next = newNode;
55     newNode->prev = temp;
56
57     printf("Inserted %d at end\n", value);
58 }
59
60
61 void insertAtPosition(int value, int position) {
62     if (head == NULL || position == 0) {
63         insertAtBeginning(value);
64         return;
65     }
66
67     struct Node* temp = head;
68     for (int i = 1; i < position && temp->next != NULL; i++) {
69         temp = temp->next;
70     }
71
72     struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
73     newNode->data = value;
74     newNode->prev = temp;
75     newNode->next = temp->next;
76
77     temp->next->prev = newNode;
78     temp->next = newNode;
79
80     printf("Inserted %d at position %d\n", value, position);
81 }
82
83
84 void display() {
85     struct Node* temp = head;
86     printf("Doubly Linked List: ");
87     while (temp != NULL) {
88         printf("%d <-> ", temp->data);
89         temp = temp->next;
90     }
91     printf("NULL\n");
92 }
```

1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Display
5. Exit
Enter your choice: 1
Enter value: 10
Inserted 10 at beginning

Enter your choice: 2
Enter value: 30
Inserted 30 at end

Enter your choice: 3
Enter value and position: 20 1
Inserted 20 after position 1

Enter your choice: 4
Doubly Linked List: 10 <-> 20 <-> 30 <-> NULL

https://googleads.g.doubleclick.net/pcs/click?xai=AKAOjssjVF01EZuFeFXSDhKVVeubEc2w_KoyHlo-V7PpF7RYu4yA1w6zR-3ZS_Z_KWBgTpMzYBj0Gkd6U3M84uXlIeEXQfQjp5InsPrYzqGztd9QCycPbc7ZvIW0dBO47h5d7_-7PAKKHcMVzBvSfkHeab1FOvGeFuCNZcuT6o

Programiz
C Online Compiler

Premium Coding Courses by Programiz

Run

Output

```
main.c
37
38
39 void insertAtEnd(int value) {
40     struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
41     newNode->data = value;
42     newNode->next = NULL;
43
44     if (head == NULL) {
45         newNode->prev = NULL;
46         head = newNode;
47         printf("Inserted %d at end\n", value);
48         return;
49     }
50
51     struct Node* temp = head;
52     while (temp->next != NULL)
53         temp = temp->next;
54
55     temp->next = newNode;
56     newNode->prev = temp;
57
58     printf("Inserted %d at end\n", value);
59 }
60
61
62 void insertAtPosition(int value, int position) {
63     if (head == NULL || position == 0) {
64         insertAtBeginning(value);
65         return;
66     }
67
68     struct Node* temp = head;
69     for (int i = 1; i < position && temp->next != NULL; i++) {
70         temp = temp->next;
71     }
72
73     struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
74     newNode->data = value;
75     newNode->prev = temp;
76     newNode->next = temp->next;
77
78     temp->next->prev = newNode;
79     temp->next = newNode;
80
81     printf("Inserted %d at position %d\n", value, position);
82 }
83
84
85 void display() {
86     struct Node* temp = head;
87     printf("Doubly Linked List: ");
88     while (temp != NULL) {
89         printf("%d <-> ", temp->data);
90         temp = temp->next;
91     }
92     printf("NULL\n");
93 }
```

1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Display
5. Exit
Enter your choice: 1
Enter value: 10
Inserted 10 at beginning

Enter your choice: 2
Enter value: 30
Inserted 30 at end

Enter your choice: 3
Enter value and position: 20 1
Inserted 20 after position 1

Enter your choice: 4
Doubly Linked List: 10 <-> 20 <-> 30 <-> NULL

*** Session Ended. Please Run the code again ***

Programiz

C Online Compiler

Premium Coding Courses by Programiz

Programiz PRO

```
main.c
72
73     struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
74     newNode->data = value;
75
76     newNode->next = temp->next;
77     newNode->prev = temp;
78
79     if (temp->next != NULL)
80         temp->next->prev = newNode;
81
82     temp->next = newNode;
83
84     printf("Inserted %d after position %d\n", value, position);
85 }
86
87
88 int main() {
89     int choice, value, position;
90
91     while (1) {
92         printf("\n1. Insert at Beginning\n2. Insert at End\n3. Insert at Position\n4. Display\n5.
93             ExitnEnter your choice: ");
94         scanf("%d", &choice);
95
96         switch (choice) {
97             case 1:
98                 printf("Enter value: ");
99                 scanf("%d", &value);
100                insertAtBeginning(value);
101                break;
102            case 2:
103                 printf("Enter value: ");
104                 scanf("%d", &value);
105                insertAtEnd(value);
106                break;
107            case 3:
108                 printf("Enter value and position: ");
109                 scanf("%d %d", &value, &position);
110                insertAtPosition(value, position);
111                break;
112            case 4:
113                 display();
114                break;
115            case 5:
116                 return 0;
117            default:
118                 printf("Invalid choice\n");
119         }
120     }
121 }
```

Output

```
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Display
5. Exit
Enter your choice: 1
Enter value: 10
Inserted 10 at beginning

Enter your choice: 2
Enter value: 30
Inserted 30 at end

Enter your choice: 3
Enter value and position: 20 1
Inserted 20 after position 1

Enter your choice: 4
Doubly Linked List: 10 <-> 20 <-> 30 <-> NULL
*** Session Ended. Please Run the code again ***
```

Programiz

C Online Compiler

Premium Coding Courses by Programiz

Programiz PRO

```
main.c
85 }
86
87
88 int main() {
89     int choice, value, position;
90
91     while (1) {
92         printf("\n1. Insert at Beginning\n2. Insert at End\n3. Insert at Position\n4. Display\n5.
93             ExitnEnter your choice: ");
94         scanf("%d", &choice);
95
96         switch (choice) {
97             case 1:
98                 printf("Enter value: ");
99                 scanf("%d", &value);
100                insertAtBeginning(value);
101                break;
102            case 2:
103                 printf("Enter value: ");
104                 scanf("%d", &value);
105                insertAtEnd(value);
106                break;
107            case 3:
108                 printf("Enter value and position: ");
109                 scanf("%d %d", &value, &position);
110                insertAtPosition(value, position);
111                break;
112            case 4:
113                 display();
114                break;
115            case 5:
116                 return 0;
117            default:
118                 printf("Invalid choice\n");
119         }
120     }
121 }
```

Output

```
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Display
5. Exit
Enter your choice: 1
Enter value: 10
Inserted 10 at beginning

Enter your choice: 2
Enter value: 30
Inserted 30 at end

Enter your choice: 3
Enter value and position: 20 1
Inserted 20 after position 1

Enter your choice: 4
Doubly Linked List: 10 <-> 20 <-> 30 <-> NULL
*** Session Ended. Please Run the code again ***
```

18. Write a program in C to implement deletion from a doubly linked list(beg; mid; end)

Programiz
C Online Compiler

Premium Coding Courses by Programiz

Programiz PRO

main.c

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 struct Node {
5     int data;
6     struct Node *prev, *next;
7 };
8
9 struct Node *head = NULL;
10
11 // Insert at end for testing
12 void insert(int value) {
13     struct Node *newNode = (struct Node*)malloc(sizeof(struct Node));
14     newNode->data = value;
15     newNode->next = NULL;
16
17     if (head == NULL) {
18         newNode->prev = NULL;
19         head = newNode;
20     } else {
21         struct Node *temp = head;
22         while (temp->next) temp = temp->next;
23         temp->next = newNode;
24         newNode->prev = temp;
25     }
26 }
27
28 // Display list
29 void display() {
30     struct Node *temp = head;
31     printf("List: ");
32     while (temp) {
33         printf("%d <-> ", temp->data);
34         temp = temp->next;
35     }
36     printf("NULL\n");
37 }
38
```

Output

```
1. Delete Beginning
2. Delete End
3. Delete at Position
4. Display
5. Exit
Enter choice: 4
List: 10 <-> 20 <-> 30 <-> 40 <-> NULL

Enter choice: 1
Deleted 10 from beginning

Enter choice: 3
Enter position: 2
Deleted 30 from position 2

Enter choice: 2
Deleted 40 from end

Enter choice: 4
List: 20 <-> NULL
```

Clear

Programiz
C Online Compiler

Premium Coding Courses by Programiz

Programiz PRO

main.c

```
39 // Delete from beginning
40 void deleteBeginning() {
41     if (!head) {
42         printf("List is empty\n");
43         return;
44     }
45     struct Node *temp = head;
46     head = head->next;
47     if (head) head->prev = NULL;
48     printf("Deleted %d from beginning\n", temp->data);
49     free(temp);
50 }
51
52 // Delete from end
53 void deleteEnd() {
54     if (!head) {
55         printf("List is empty\n");
56         return;
57     }
58     struct Node *temp = head;
59     if (!temp->next) {
60         printf("Deleted %d from end\n", temp->data);
61         free(temp);
62         head = NULL;
63         return;
64     }
65     while (temp->next) temp = temp->next;
66     printf("Deleted %d from end\n", temp->data);
67     temp->next = NULL;
68     free(temp);
69 }
70
71 // Delete from position
72 void deletePosition(int pos) {
73     if (!head || pos < 0) {
74         printf("Invalid position\n");
75         return;
76     }
```

Output

```
1. Delete Beginning
2. Delete End
3. Delete at Position
4. Display
5. Exit
Enter choice: 4
List: 10 <-> 20 <-> 30 <-> 40 <-> NULL

Enter choice: 1
Deleted 10 from beginning

Enter choice: 3
Enter position: 2
Deleted 30 from position 2

Enter choice: 2
Deleted 40 from end

Enter choice: 4
List: 20 <-> NULL

*** Session Ended. Please Run the code again ***
```

Clear

Programiz
C Online Compiler

Premium Coding Courses by Programiz

Programiz PRO

```
main.c
74     printf("Invalid position\n");
75     return;
76 }
77 if (pos == 1) {
78     deleteBeginning();
79     return;
80 }
81 struct Node *temp = head;
82 for (int i = 1; temp && i < pos; i++)
83     temp = temp->next;
84
85 if (!temp) {
86     printf("Position out of range\n");
87     return;
88 }
89
90 if (temp->next)
91     temp->next->prev = temp->prev;
92 if (temp->prev)
93     temp->prev->next = temp->next;
94
95 printf("Deleted %d from position %d\n", temp->data, pos);
96 free(temp);
97 }
98 // Main
99 int main() {
100     insert(10);
101     insert(20);
102     insert(30);
103     insert(40);
104     insert(50);
105
106     int choice, pos;
107     while (!) {
108         printf("\n1. Delete Beginning\n2. Delete End\n3. Delete at Position\n4. Display\n5.
109         Enter choice: ");
110         scanf("%d", &choice);
111         switch (choice) {
112             case 1: deleteBeginning(); break;
113             case 2: deleteEnd(); break;
114             case 3:
115                 printf("Enter position: ");
116                 scanf("%d", &pos);
117                 deletePosition(pos);
118             case 4: display(); break;
119             case 5: return 0;
120             default: printf("Invalid choice\n");
121         }
122     }
123 }
```

Output

```
1. Delete Beginning
2. Delete End
3. Delete at Position
4. Display
5. Exit
Enter choice: 4
List: 10 <-> 20 <-> 30 <-> 40 <-> NULL

Enter choice: 1
Deleted 10 from beginning

Enter choice: 3
Enter position: 2
Deleted 30 from position 2

Enter choice: 2
Deleted 40 from end

Enter choice: 4
List: 20 <-> NULL

*** Session Ended. Please Run the code again ***
```

Programiz
C Online Compiler

Premium Coding Courses by Programiz

Programiz PRO

```
main.c
89
90 if (temp->next)
91     temp->next->prev = temp->prev;
92 if (temp->prev)
93     temp->prev->next = temp->next;
94
95 printf("Deleted %d from position %d\n", temp->data, pos);
96 free(temp);
97 }
98 // Main
99 int main() {
100     insert(10);
101     insert(20);
102     insert(30);
103     insert(40);
104     insert(50);
105
106     int choice, pos;
107     while (!) {
108         printf("\n1. Delete Beginning\n2. Delete End\n3. Delete at Position\n4. Display\n5.
109         Enter choice: ");
110         scanf("%d", &choice);
111         switch (choice) {
112             case 1: deleteBeginning(); break;
113             case 2: deleteEnd(); break;
114             case 3:
115                 printf("Enter position: ");
116                 scanf("%d", &pos);
117                 deletePosition(pos);
118             case 4: display(); break;
119             case 5: return 0;
120             default: printf("Invalid choice\n");
121         }
122     }
123 }
```

Output

```
1. Delete Beginning
2. Delete End
3. Delete at Position
4. Display
5. Exit
Enter choice: 4
List: 10 <-> 20 <-> 30 <-> 40 <-> NULL

Enter choice: 1
Deleted 10 from beginning

Enter choice: 3
Enter position: 2
Deleted 30 from position 2

Enter choice: 2
Deleted 40 from end

Enter choice: 4
List: 20 <-> NULL

*** Session Ended. Please Run the code again ***
```

<https://www.programiz.com>

19. Write a program in C to implement insertion in Binary tree 20. Write a program in C to implement deletion from Binary tree simple tree

The screenshot shows the Programiz C Online Compiler interface. The code in the editor is for inserting into a binary tree. The output window shows the result of an in-order traversal: "In-order traversal of the Binary Tree: 4 2 5 1 3" followed by "Code Execution Successful".

```
main.c
1 #include <stdio.h>
2 #include <stdlib.h>
3
4
5 struct Node {
6     int data;
7     struct Node* left;
8     struct Node* right;
9 };
10
11
12 struct Node* createNode(int value) {
13     struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
14     newNode->data = value;
15     newNode->left = newNode->right = NULL;
16     return newNode;
17 }
18
19
20 void insert(struct Node** root, int value) {
21     struct Node* newNode = createNode(value);
22     if (*root == NULL) {
23         *root = newNode;
24         return;
25     }
26
27     struct Node* queue[100];
28     int front = 0, rear = 0;
29
30     queue[rear++] = *root;
31
32     while (front != rear) {
33         struct Node* temp = queue[front++];
34
35         if (temp->left == NULL) {
36             temp->left = newNode;
37             return;
38         } else {
39             queue[rear++] = temp->left;
40         }
41
42         if (temp->right == NULL) {
43             temp->right = newNode;
44             return;
45         } else {
46             queue[rear++] = temp->right;
47         }
48     }
49 }
50
51
52 void inorder(struct Node* root) {
53     if (root != NULL) {
54         inorder(root->left);
55         printf("%d ", root->data);
56         inorder(root->right);
57     }
58 }
59
60
61 int main() {
62     struct Node* root = NULL;
63
64     insert(&root, 1);
65     insert(&root, 2);
66     insert(&root, 3);
67     insert(&root, 4);
68     insert(&root, 5);
69
70     printf("In-order traversal of the Binary Tree: ");
71     inorder(root); // Output the tree structure in in-order
72     printf("\n");
73
74     return 0;
75 }
```

The screenshot shows the same Programiz C Online Compiler interface, but the code now includes deletion logic. The output window shows the same in-order traversal result: "In-order traversal of the Binary Tree: 4 2 5 1 3" followed by "Code Execution Successful".

```
main.c
38     } else {
39         queue[rear++] = temp->left;
40     }
41
42     if (temp->right == NULL) {
43         temp->right = newNode;
44         return;
45     } else {
46         queue[rear++] = temp->right;
47     }
48 }
49 }
50
51
52 void inorder(struct Node* root) {
53     if (root != NULL) {
54         inorder(root->left);
55         printf("%d ", root->data);
56         inorder(root->right);
57     }
58 }
59
60
61 int main() {
62     struct Node* root = NULL;
63
64     insert(&root, 1);
65     insert(&root, 2);
66     insert(&root, 3);
67     insert(&root, 4);
68     insert(&root, 5);
69
70     printf("In-order traversal of the Binary Tree: ");
71     inorder(root); // Output the tree structure in in-order
72     printf("\n");
73
74     return 0;
75 }
```

20. Write a program in C to implement deletion from Binary tree

The screenshot shows the Programiz Online Compiler interface. The code in the editor is for inserting values into a binary tree. The output window shows the original tree's in-order traversal (20 30 40 50 60 70 80), the deletion of node 50, and the resulting tree's in-order traversal (20 30 40 60 70 80). The status message indicates successful execution.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 struct Node {
5     int data;
6     struct Node* left;
7     struct Node* right;
8 };
9
10
11 struct Node* createNode(int value) {
12     struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
13     newNode->data = value;
14     newNode->left = newNode->right = NULL;
15     return newNode;
16 }
17
18
19
20 struct Node* insert(struct Node* root, int value) {
21     if (root == NULL)
22         return createNode(value);
23
24     if (value < root->data)
25         root->left = insert(root->left, value);
26     else if (value > root->data)
27         root->right = insert(root->right, value);
28
29     return root;
30 }
31
32
33 struct Node* minValueNode(struct Node* node) {
34     struct Node* current = node;
35
36     while (current && current->left != NULL)
37         current = current->left;
38 }
```

The screenshot shows the Programiz Online Compiler interface. The code in the editor is for deleting a node from a binary tree. The output window shows the original tree's in-order traversal (20 30 40 50 60 70 80), the deletion of node 50, and the resulting tree's in-order traversal (20 30 40 60 70 80). The status message indicates successful execution.

```
1 struct Node* minValueNode(struct Node* node) {
2     struct Node* current = node;
3
4     while (current && current->left != NULL)
5         current = current->left;
6
7     return current;
8 }
9
10
11 struct Node* deleteNode(struct Node* root, int key) {
12     if (root == NULL) return root;
13
14     if (key < root->data)
15         root->left = deleteNode(root->left, key);
16     else if (key > root->data)
17         root->right = deleteNode(root->right, key);
18     else {
19
20         if (root->left == NULL) {
21             struct Node* temp = root->right;
22             free(root);
23             return temp;
24         }
25
26         else if (root->right == NULL) {
27             struct Node* temp = root->left;
28             free(root);
29             return temp;
30         }
31
32         struct Node* temp = minValueNode(root->right);
33
34         root->data = temp->data;
35
36         root->right = deleteNode(root->right, temp->data);
37     }
38
39     return root;
40 }
```

Programiz
C Online Compiler

Premium Coding Courses by Programiz

Inorder traversal of the original tree:
20 30 40 50 60 70 80

Deleting 50
Inorder traversal after deletion:
20 30 40 60 70 80

== Code Execution Successful ==

```
main.c
72 }
73
74     return root;
75 }
76
77
78 void inorder(struct Node* root) {
79     if (root != NULL) {
80         inorder(root->left);
81         printf("%d ", root->data);
82         inorder(root->right);
83     }
84 }
85
86
87 int main() {
88     struct Node* root = NULL;
89
90
91     root = insert(root, 50);
92     root = insert(root, 30);
93     root = insert(root, 70);
94     root = insert(root, 20);
95     root = insert(root, 40);
96     root = insert(root, 60);
97     root = insert(root, 80);
98
99     printf("Inorder traversal of the original tree:\n");
100    inorder(root);
101
102    printf("\n\nDeleting 50\n");
103    root = deleteNode(root, 50);
104
105    printf("Inorder traversal after deletion:\n");
106    inorder(root);
107
108    return 0;
109 }
```

Programiz
C Online Compiler

Premium Coding Courses by Programiz

Inorder traversal of the original tree:
20 30 40 50 60 70 80

Deleting 50
Inorder traversal after deletion:
20 30 40 60 70 80

== Code Execution Successful ==

```
main.c
74     return root;
75 }
76
77
78 void inorder(struct Node* root) {
79     if (root != NULL) {
80         inorder(root->left);
81         printf("%d ", root->data);
82         inorder(root->right);
83     }
84 }
85
86
87 int main() {
88     struct Node* root = NULL;
89
90
91     root = insert(root, 50);
92     root = insert(root, 30);
93     root = insert(root, 70);
94     root = insert(root, 20);
95     root = insert(root, 40);
96     root = insert(root, 60);
97     root = insert(root, 80);
98
99     printf("Inorder traversal of the original tree:\n");
100    inorder(root);
101
102    printf("\n\nDeleting 50\n");
103    root = deleteNode(root, 50);
104
105    printf("Inorder traversal after deletion:\n");
106    inorder(root);
107
108    return 0;
109 }
110 }
```

21. Write a program in C to implement recursive tree traversals (Inorder; Preorder; Postorder)

Programiz
C Online Compiler

Premium Coding Courses by Programiz

Run Output Clear

```
main.c
1 #include <stdio.h>
2 #include <stdlib.h>
3
4
5 struct Node {
6     int data;
7     struct Node* left;
8     struct Node* right;
9 };
10
11
12 struct Node* createNode(int value) {
13     struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
14     newNode->data = value;
15     newNode->left = newNode->right = NULL;
16     return newNode;
17 }
18
19
20 void inorder(struct Node* root) {
21     if (root != NULL) {
22         inorder(root->left);
23         printf("%d ", root->data);
24         inorder(root->right);
25     }
26 }
27
28 void preorder(struct Node* root) {
29     if (root != NULL) {
30         printf("%d ", root->data);
31         preorder(root->left);
32         preorder(root->right);
33     }
34 }
35
36
37 void postorder(struct Node* root) {
38     if (root != NULL) {
```

Inorder Traversal: 4 2 5 1 6 3 7
Preorder Traversal: 1 2 4 5 3 6 7
Postorder Traversal: 4 5 2 6 7 3 1

*** Code Execution Successful ***

Programiz
C Online Compiler

Premium Coding Courses by Programiz

Run Output Clear

```
main.c
39 }
40
41
42 }
43 }
44
45
46 int main() {
47
48     struct Node* root = createNode(1);
49     root->left = createNode(2);
50     root->right = createNode(3);
51     root->left->left = createNode(4);
52     root->left->right = createNode(5);
53     root->right->left = createNode(6);
54     root->right->right = createNode(7);
55
56     printf("Inorder Traversal: ");
57     inorder(root);
58     printf("\n");
59
60     printf("Preorder Traversal: ");
61     preorder(root);
62     printf("\n");
63
64     printf("Postorder Traversal: ");
65     postorder(root);
66     printf("\n");
67
68     return 0;
69 }
70
```

Inorder Traversal: 4 2 5 1 6 3 7
Preorder Traversal: 1 2 4 5 3 6 7
Postorder Traversal: 4 5 2 6 7 3 1

*** Code Execution Successful ***

22. Write a program in C to Sort a list using Bubble Sort.

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

```
main.c
1 #include <stdio.h>
2
3 void bubbleSort(int arr[], int n) {
4     for (int i = 0; i < n-1; i++) {
5
6         for (int j = 0; j < n-i-1; j++) {
7
8             if (arr[j] > arr[j+1]) {
9                 int temp = arr[j];
10                arr[j] = arr[j+1];
11                arr[j+1] = temp;
12            }
13        }
14    }
15
16
17 void displayArray(int arr[], int n) {
18     for (int i = 0; i < n; i++) {
19         printf("%d ", arr[i]);
20     }
21     printf("\n");
22 }
23
24
25
26 int main() {
27     int arr[] = {64, 34, 25, 12, 22, 11, 90}; // Example array
28     int n = sizeof(arr) / sizeof(arr[0]);
29
30     printf("Original Array: ");
31     displayArray(arr, n);
32
33     bubbleSort(arr, n);
34
35     printf("Sorted Array: ");
36     displayArray(arr, n);
37
38     return 0;
39 }
```

The output window shows the results of the execution:

```
Original Array: 64 34 25 12 22 11 90
Sorted Array: 11 12 22 25 34 64 90
--- Code Execution Successful ---
```

The screenshot shows the Programiz C Online Compiler interface. The code in the editor is identical to the one in the previous screenshot:

```
main.c
1 #include <stdio.h>
2
3 void bubbleSort(int arr[], int n) {
4     for (int i = 0; i < n-1; i++) {
5
6         for (int j = 0; j < n-i-1; j++) {
7
8             if (arr[j] > arr[j+1]) {
9                 int temp = arr[j];
10                arr[j] = arr[j+1];
11                arr[j+1] = temp;
12            }
13        }
14    }
15
16
17 void displayArray(int arr[], int n) {
18     for (int i = 0; i < n; i++) {
19         printf("%d ", arr[i]);
20     }
21     printf("\n");
22 }
23
24
25
26 int main() {
27     int arr[] = {64, 34, 25, 12, 22, 11, 90}; // Example array
28     int n = sizeof(arr) / sizeof(arr[0]);
29
30     printf("Original Array: ");
31     displayArray(arr, n);
32
33     bubbleSort(arr, n);
34
35     printf("Sorted Array: ");
36     displayArray(arr, n);
37
38     return 0;
39 }
```

The output window shows the results of the execution:

```
Original Array: 64 34 25 12 22 11 90
Sorted Array: 11 12 22 25 34 64 90
--- Code Execution Successful ---
```

23. Write a program in C to Sort a list using Selection Sort.

The screenshot shows the Programiz C Online Compiler interface. The code editor contains a C program named main.c. The code implements the Selection Sort algorithm. The output window shows the original array [64, 34, 25, 12, 22, 11, 90] and the sorted array [11, 12, 22, 25, 34, 64, 90]. A message "Code Execution Successful" is displayed at the bottom of the output.

```
1 #include <stdio.h>
2
3 void selectionSort(int arr[], int n) {
4     for (int i = 0; i < n-1; i++) {
5         int minIndex = i;
6         for (int j = i+1; j < n; j++) {
7             if (arr[j] < arr[minIndex]) {
8                 minIndex = j;
9             }
10        }
11
12        if (minIndex != i) {
13            int temp = arr[i];
14            arr[i] = arr[minIndex];
15            arr[minIndex] = temp;
16        }
17    }
18 }
19
20
21 void displayArray(int arr[], int n) {
22     for (int i = 0; i < n; i++) {
23         printf("%d ", arr[i]);
24     }
25     printf("\n");
26 }
27
28
29
30 int main() {
31     int arr[] = {64, 34, 25, 12, 22, 11, 90};
32     int n = sizeof(arr) / sizeof(arr[0]);
33
34     printf("Original Array: ");
35     displayArray(arr, n);
36
37     selectionSort(arr, n);
38
39     printf("Sorted Array: ");
40     displayArray(arr, n);
41
42     return 0;
43 }
```

The screenshot shows the Programiz C Online Compiler interface. The code editor contains a C program named main.c. The code implements the Selection Sort algorithm. The output window shows the original array [64, 34, 25, 12, 22, 11, 90] and the sorted array [11, 12, 22, 25, 34, 64, 90]. A message "Code Execution Successful" is displayed at the bottom of the output.

```
1 #include <stdio.h>
2
3 void selectionSort(int arr[], int n) {
4     for (int i = 0; i < n-1; i++) {
5         int minIndex = i;
6         for (int j = i+1; j < n; j++) {
7             if (arr[j] < arr[minIndex]) {
8                 minIndex = j;
8             }
9         }
10        if (minIndex != i) {
11            int temp = arr[i];
12            arr[i] = arr[minIndex];
13            arr[minIndex] = temp;
14        }
15    }
16 }
17
18
19
20
21 void displayArray(int arr[], int n) {
22     for (int i = 0; i < n; i++) {
23         printf("%d ", arr[i]);
24     }
25     printf("\n");
26 }
27
28
29
30 int main() {
31     int arr[] = {64, 34, 25, 12, 22, 11, 90};
32     int n = sizeof(arr) / sizeof(arr[0]);
33
34     printf("Original Array: ");
35     displayArray(arr, n);
36
37     selectionSort(arr, n);
38
39     printf("Sorted Array: ");
40     displayArray(arr, n);
41
42     return 0;
43 }
```

24. Write a program in C to sort a list using Quick Sort

Programiz
C Online Compiler

Premium Coding Courses by Programiz

Run

Output

```
main.c
1 #include <stdio.h>
2
3
4- void swap(int* a, int* b) {
5     int temp = *a;
6     *a = *b;
7     *b = temp;
8 }
9
10
11- int partition(int arr[], int low, int high) {
12     int pivot = arr[high];
13     int i = (low - 1);
14
15-     for (int j = low; j <= high - 1; j++) {
16         if (arr[j] < pivot) {
17             i++;
18             swap(&arr[i], &arr[j]);
19         }
20     }
21
22
23     swap(&arr[i + 1], &arr[high]);
24     return (i + 1);
25 }
26
27
28- void quickSort(int arr[], int low, int high) {
29     if (low < high) {
30
31         int pi = partition(arr, low, high);
32
33
34         quickSort(arr, low, pi - 1);
35         quickSort(arr, pi + 1, high);
36     }
37 }
38
```

Original Array: 64 34 25 12 22 11 90
Sorted Array: 11 12 22 25 34 64 90
*** Code Execution Successful ***

Programiz
C Online Compiler

Premium Coding Courses by Programiz

Run

Output

```
main.c
24     return (i + 1);
25 }
26
27
28- void quickSort(int arr[], int low, int high) {
29     if (low < high) {
30
31         int pi = partition(arr, low, high);
32
33
34         quickSort(arr, low, pi - 1);
35         quickSort(arr, pi + 1, high);
36     }
37 }
38
39 /
40- void displayArray(int arr[], int n) {
41     for (int i = 0; i < n; i++) {
42         printf("%d ", arr[i]);
43     }
44     printf("\n");
45 }
46
47 int main() {
48     int arr[] = {64, 34, 25, 12, 22, 11, 90};
49     int n = sizeof(arr) / sizeof(arr[0]);
50
51     printf("Original Array: ");
52     displayArray(arr, n);
53
54     quickSort(arr, 0, n - 1);
55
56     printf("Sorted Array: ");
57     displayArray(arr, n);
58
59     return 0;
60 }
61
```

Original Array: 64 34 25 12 22 11 90
Sorted Array: 11 12 22 25 34 64 90
*** Code Execution Successful ***

25. Write a program in C to sort a list using Merge Sort

Programiz
C Online Compiler

Premium Coding Courses by Programiz

Programiz PRO

```
main.c
1 #include <stdio.h>
2 |
3 void merge(int arr[], int left, int mid, int right) {
4     int i, j, k;
5     int n1 = mid - left + 1;
6     int n2 = right - mid;
7
8     int L[n1], R[n2];
9
10    for (i = 0; i < n1; i++)
11        L[i] = arr[left + i];
12    for (j = 0; j < n2; j++)
13        R[j] = arr[mid + 1 + j];
14
15    i = 0, j = 0, k = left;
16    while (i < n1 && j < n2) {
17        if (L[i] <= R[j])
18            arr[k++] = L[i++];
19        else
20            arr[k++] = R[j++];
21
22    }
23
24
25
26    while (i < n1)
27        arr[k++] = L[i++];
28
29
30    while (j < n2)
31        arr[k++] = R[j++];
32
33 }
34
35
36 void mergeSort(int arr[], int left, int right) {
37     if (left < right) {
38         int mid = (left + right) / 2;
39
40         mergeSort(arr, left, mid);
41         mergeSort(arr, mid + 1, right);
42
43
44         merge(arr, left, mid, right);
45     }
46 }
47
48
49 void displayArray(int arr[], int size) {
50     for (int i = 0; i < size; i++)
51         printf("%d ", arr[i]);
52     printf("\n");
53 }
54
55
56 int main() {
57     int arr[] = {64, 34, 25, 12, 22, 11, 90};
58     int size = sizeof(arr) / sizeof(arr[0]);
59
60     printf("Original Array: ");
61     displayArray(arr, size);
62
63     mergeSort(arr, 0, size - 1);
64
65     printf("Sorted Array: ");
66     displayArray(arr, size);
67
68     return 0;
69 }
```

Output

Original Array: 64 34 25 12 22 11 90
Sorted Array: 11 12 22 25 34 64 90
*** Code Execution Successful ***

Programiz
C Online Compiler

Premium Coding Courses by Programiz

Programiz PRO

```
main.c
33
34
35
36 void mergeSort(int arr[], int left, int right) {
37     if (left < right) {
38         int mid = (left + right) / 2;
39
40         mergeSort(arr, left, mid);
41         mergeSort(arr, mid + 1, right);
42
43
44         merge(arr, left, mid, right);
45     }
46 }
47
48
49 void displayArray(int arr[], int size) {
50     for (int i = 0; i < size; i++)
51         printf("%d ", arr[i]);
52     printf("\n");
53 }
54
55
56 int main() {
57     int arr[] = {64, 34, 25, 12, 22, 11, 90};
58     int size = sizeof(arr) / sizeof(arr[0]);
59
60     printf("Original Array: ");
61     displayArray(arr, size);
62
63     mergeSort(arr, 0, size - 1);
64
65     printf("Sorted Array: ");
66     displayArray(arr, size);
67
68     return 0;
69 }
```

Output

Original Array: 64 34 25 12 22 11 90
Sorted Array: 11 12 22 25 34 64 90
*** Code Execution Successful ***

26. Write a program in C to sort a list using Insertion Sort

Programiz
C Online Compiler

Premium Coding Courses by Programiz

Original Array: 29 10 14 37 13
Sorted Array: 10 13 14 29 37

==== Code Execution Successful ===

```
main.c
1 #include <stdio.h>
2
3
4 void insertionSort(int arr[], int n) {
5     for (int i = 1; i < n; i++) {
6         int key = arr[i];
7         int j = i - 1;
8
9         while (j >= 0 && arr[j] > key) {
10             arr[j + 1] = arr[j];
11             j--;
12         }
13
14         arr[j + 1] = key;
15     }
16 }
17
18 void displayArray(int arr[], int n) {
19     for (int i = 0; i < n; i++)
20         printf("%d ", arr[i]);
21     printf("\n");
22 }
23
24
25
26 int main() {
27     int arr[] = {29, 10, 14, 37, 13};
28     int n = sizeof(arr) / sizeof(arr[0]);
29
30     printf("Original Array: ");
31     displayArray(arr, n);
32
33     insertionSort(arr, n);
34
35     printf("Sorted Array: ");
36     displayArray(arr, n);
37
38     return 0;
39 }
```

Programiz
C Online Compiler

Premium Coding Courses by Programiz

Original Array: 29 10 14 37 13
Sorted Array: 10 13 14 29 37

==== Code Execution Successful ===

```
main.c
1 #include <stdio.h>
2
3
4 void insertionSort(int arr[], int n) {
5     for (int i = 1; i < n; i++) {
6         int key = arr[i];
7         int j = i - 1;
8
9         while (j >= 0 && arr[j] > key) {
10             arr[j + 1] = arr[j];
11             j--;
12         }
13
14         arr[j + 1] = key;
15     }
16 }
17
18 void displayArray(int arr[], int n) {
19     for (int i = 0; i < n; i++)
20         printf("%d ", arr[i]);
21     printf("\n");
22 }
23
24
25
26 int main() {
27     int arr[] = {29, 10, 14, 37, 13};
28     int n = sizeof(arr) / sizeof(arr[0]);
29
30     printf("Original Array: ");
31     displayArray(arr, n);
32
33     insertionSort(arr, n);
34
35     printf("Sorted Array: ");
36     displayArray(arr, n);
37
38     return 0;
39 }
```

27. Write a program in C to sort a list using Heap Sort

Programiz
C Online Compiler

Premium Coding Courses by Programiz

Original array: 12 11 13 5 6 7
Sorted array: 5 6 7 11 12 13

==== Code Execution Successful ===

```
1 #include <stdio.h>
2
3
4 void heapify(int arr[], int n, int i) {
5     int largest = i;
6     int left = 2 * i + 1;
7     int right = 2 * i + 2;
8
9     if (left < n && arr[left] > arr[largest])
10        largest = left;
11
12
13     if (right < n && arr[right] > arr[largest])
14        largest = right;
15
16
17     if (largest != i) {
18         int temp = arr[i];
19         arr[i] = arr[largest];
20         arr[largest] = temp;
21
22         heapify(arr, n, largest);
23     }
24
25 }
26
27
28
29 void heapSort(int arr[], int n) {
30
31     for (int i = n / 2 - 1; i >= 0; i--)
32         heapify(arr, n, i);
33
34
35     for (int i = n - 1; i >= 0; i--) {
36
37         int temp = arr[0];
38         arr[0] = arr[i];
39         arr[i] = temp;
40
41         heapify(arr, i, 0);
42     }
43
44 }
45
46
47 void displayArray(int arr[], int n) {
48     for (int i = 0; i < n; i++)
49         printf("%d ", arr[i]);
50     printf("\n");
51 }
52
53 // Driver code
54 int main() {
55     int arr[] = {12, 11, 13, 5, 6, 7};
56     int n = sizeof(arr) / sizeof(arr[0]);
57
58     printf("Original array: ");
59     displayArray(arr, n);
60
61     heapSort(arr, n);
62
63     printf("Sorted array: ");
64     displayArray(arr, n);
65
66     return 0;
67 }
```

Programiz
C Online Compiler

Premium Coding Courses by Programiz

Original array: 12 11 13 5 6 7
Sorted array: 5 6 7 11 12 13

==== Code Execution Successful ===

```
1 #include <stdio.h>
2
3
4 void heapify(int arr[], int n, int i) {
5     int largest = i;
6     int left = 2 * i + 1;
7     int right = 2 * i + 2;
8
9     if (left < n && arr[left] > arr[largest])
10        largest = left;
11
12
13     if (right < n && arr[right] > arr[largest])
14        largest = right;
15
16     if (largest != i) {
17         int temp = arr[i];
18         arr[i] = arr[largest];
19         arr[largest] = temp;
20
21         heapify(arr, n, largest);
22     }
23
24 }
25
26
27
28
29 void displayArray(int arr[], int n) {
30     for (int i = 0; i < n; i++)
31         printf("%d ", arr[i]);
32     printf("\n");
33 }
34
35 // Driver code
36 int main() {
37     int arr[] = {12, 11, 13, 5, 6, 7};
38     int n = sizeof(arr) / sizeof(arr[0]);
39
40     printf("Original array: ");
41     displayArray(arr, n);
42
43     heapSort(arr, n);
44
45     printf("Sorted array: ");
46     displayArray(arr, n);
47
48     return 0;
49 }
```