**USN:- 1NT21IS039**

## Aim:- IMPLEMENTATION OF DOUBLY LINKED LIST

## [GitHub-Link](https://github.com/logan-14/Implementation-of-LinkedList)

### **Algorithm:-**

Assume that START is the first element in the linked list and TAIL is the last element of linked list.

i. Insert At Beginning

1. Start
2. Input the DATA to be inserted
3. Create a new node.
4. NewNode → Data = DATA NewNode →Lpoint =NULL
5. IF START IS NULL NewNode→ Rpoint = NULL
6. Else NewNode → Rpoint = START START→Lpoint = NewNode
7. START =NewNode
8. Stop

**ii. Insertion at location:**

1. Start
2. Input the DATA and POS
3. Initialize TEMP = START; i = 0
4. Repeat the step 4 if (i less than POS) and (TEMP is not equal to NULL)
5. TEMP = TEMP → RPoint; i = i +1
6. If (TEMP not equal to NULL) and (i equal to POS)

(a) Create a New Node

(b) NewNode → DATA = DATA

(c) NewNode → RPoint = TEMP → RPoint

(d) NewNode → LPoint = TEMP

(e) (TEMP → RPoint) → LPoint = NewNode

1. (f ) TEMP → RPoint = New Node
2. Else

(a) Display “Position NOT found”

1. Stop

**iii. Insert at End**

1. Start
2. Input DATA to be inserted
3. Create a NewNode
4. NewNode → DATA = DATA
5. NewNode → RPoint = NULL
6. If (SATRT equal to NULL)

a. START = NewNode

b. NewNode → LPoint=NULL

1. Else

a. TEMP = START

b. While (TEMP → Next not equal to NULL)

i. TEMP = TEMP → Next

c. TEMP → RPoint = NewNode

d. NewNode → LPoint = TEMP

1. Stop

iv. Forward Traversal

1. Start
2. If (START is equal to NULL)

a) Display “The list is Empty”

b) Stop

1. Initialize TEMP = START
2. Repeat the step 5 and 6 until (TEMP == NULL )
3. Display “TEMP → DATA”
4. TEMP = TEMP → Next
5. Stop

**Code:-**

#include <stdio.h>

#include <stdlib.h>

struct node

{

    struct node \*prev;

    struct node \*next;

    int data;

};

struct node \*head;

void insertion\_beginning();

void insertion\_last();

void insertion\_specified();

void deletion\_beginning();

void deletion\_last();

void deletion\_specified();

void display();

void search();

void main()

{

    int choice = 0;

    while (choice != 9)

    {

        printf("\n\*\*\*\*\*\*\*\*\*Main Menu\*\*\*\*\*\*\*\*\*\n");

        printf("\nChoose one option from the following list ...\n");

        printf("\n===============================================\n");

        printf("\n1.Insert in begining\n2.Insert at last\n3.Insert at any random location\n4.Delete from Beginning\n 5.Delete from last \n 6.Delete the node after the given data\n7.Search\n8.Show\n9.Exit\n");

        printf("\nEnter your choice?\n");

        scanf("\n%d", &choice);

        switch (choice)

        {

        case 1:

            insertion\_beginning();

            break;

        case 2:

            insertion\_last();

            break;

        case 3:

            insertion\_specified();

            break;

        case 4:

            deletion\_beginning();

            break;

        case 5:

            deletion\_last();

            break;

        case 6:

            deletion\_specified();

            break;

        case 7:

            search();

            break;

        case 8:

            display();

            break;

        case 9:

            exit(0);

            break;

        default:

            printf("Please enter valid choice..");

        }

    }

}

void insertion\_beginning()

{

    struct node \*ptr;

    int item;

    ptr = (struct node \*)malloc(sizeof(struct node));

    if (ptr == NULL)

    {

        printf("\nOVERFLOW");

    }

    else

    {

        printf("\nEnter Item value");

        scanf("%d", &item);

        if (head == NULL)

        {

            ptr->next = NULL;

            ptr->prev = NULL;

            ptr->data = item;

            head = ptr;

        }

        else

        {

            ptr->data = item;

            ptr->prev = NULL;

            ptr->next = head;

            head->prev = ptr;

            head = ptr;

        }

        printf("\nNode inserted\n");

    }

}

void insertion\_last()

{

    struct node \*ptr, \*temp;

    int item;

    ptr = (struct node \*)malloc(sizeof(struct node));

    if (ptr == NULL)

    {

        printf("\nOVERFLOW");

    }

    else

    {

        printf("\nEnter value");

        scanf("%d", &item);

        ptr->data = item;

        if (head == NULL)

        {

            ptr->next = NULL;

            ptr->prev = NULL;

            head = ptr;

        }

        else

        {

            temp = head;

            while (temp->next != NULL)

            {

                temp = temp->next;

            }

            temp->next = ptr;

            ptr->prev = temp;

            ptr->next = NULL;

        }

    }

    printf("\nnode inserted\n");

}

void insertion\_specified()

{

    struct node \*ptr, \*temp;

    int item, loc, i;

    ptr = (struct node \*)malloc(sizeof(struct node));

    if (ptr == NULL)

    {

        printf("\n OVERFLOW");

    }

    else

    {

        temp = head;

        printf("Enter the location");

        scanf("%d", &loc);

        for (i = 0; i < loc; i++)

        {

            temp = temp->next;

            if (temp == NULL)

            {

                printf("\n There are less than %d elements", loc);

                return;

            }

        }

        printf("Enter value");

        scanf("%d", &item);

        ptr->data = item;

        ptr->next = temp->next;

        ptr->prev = temp;

        temp->next = ptr;

        temp->next->prev = ptr;

        printf("\nnode inserted\n");

    }

}

void deletion\_beginning()

{

    struct node \*ptr;

    if (head == NULL)

    {

        printf("\n UNDERFLOW");

    }

    else if (head->next == NULL)

    {

        head = NULL;

        free(head);

        printf("\nnode deleted\n");

    }

    else

    {

        ptr = head;

        head = head->next;

        head->prev = NULL;

        free(ptr);

        printf("\nnode deleted\n");

    }

}

void deletion\_last()

{

    struct node \*ptr;

    if (head == NULL)

    {

        printf("\n UNDERFLOW");

    }

    else if (head->next == NULL)

    {

        head = NULL;

        free(head);

        printf("\nnode deleted\n");

    }

    else

    {

        ptr = head;

        if (ptr->next != NULL)

        {

            ptr = ptr->next;

        }

        ptr->prev->next = NULL;

        free(ptr);

        printf("\nnode deleted\n");

    }

}

void deletion\_specified()

{

    struct node \*ptr, \*temp;

    int val;

    printf("\n Enter the data after which the node is to be deleted : ");

    scanf("%d", &val);

    ptr = head;

    while (ptr->data != val)

        ptr = ptr->next;

    if (ptr->next == NULL)

    {

        printf("\nCan't delete\n");

    }

    else if (ptr->next->next == NULL)

    {

        ptr->next = NULL;

    }

    else

    {

        temp = ptr->next;

        ptr->next = temp->next;

        temp->next->prev = ptr;

        free(temp);

        printf("\nnode deleted\n");

    }

}

void display()

{

    struct node \*ptr;

    printf("\n printing values...\n");

    ptr = head;

    while (ptr != NULL)

    {

        printf("%d\n", ptr->data);

        ptr = ptr->next;

    }

}

void search()

{

    struct node \*ptr;

    int item, i = 0, flag;

    ptr = head;

    if (ptr == NULL)

    {

        printf("\nEmpty List\n");

    }

    else

    {

        printf("\nEnter item which you want to search?\n");

        scanf("%d", &item);

        while (ptr != NULL)

        {

            if (ptr->data == item)

            {

                printf("\nitem found at location %d ", i + 1);

                flag = 0;

                break;

            }

            else

            {

                flag = 1;

            }

            i++;

            ptr = ptr->next;

        }

        if (flag == 1)

        {

            printf("\nItem not found\n");

        }

    }

}

**Output:-**

\*\*\*\*\*\*\*\*\*Main Menu\*\*\*\*\*\*\*\*\*

Choose one option from the following list ...

===============================================

1.Insert in begining

2.Insert at last

3.Insert at any random location

4.Delete from Beginning

 5.Delete from last

 6.Delete the node after the given data

7.Search

8.Show

9.Exit

Enter your choice?

1

Enter Item value10

Node inserted

\*\*\*\*\*\*\*\*\*Main Menu\*\*\*\*\*\*\*\*\*

Choose one option from the following list ...

===============================================

1.Insert in begining

2.Insert at last

3.Insert at any random location

4.Delete from Beginning

 5.Delete from last

 6.Delete the node after the given data

7.Search

8.Show

9.Exit

Enter your choice?

1

Enter Item value20

Node inserted

\*\*\*\*\*\*\*\*\*Main Menu\*\*\*\*\*\*\*\*\*

Choose one option from the following list ...

===============================================

1.Insert in begining

2.Insert at last

3.Insert at any random location

4.Delete from Beginning

 5.Delete from last

 6.Delete the node after the given data

7.Search

8.Show

9.Exit

Enter your choice?

2

Enter value30

node inserted

\*\*\*\*\*\*\*\*\*Main Menu\*\*\*\*\*\*\*\*\*

Choose one option from the following list ...

===============================================

1.Insert in begining

2.Insert at last

3.Insert at any random location

4.Delete from Beginning

 5.Delete from last

 6.Delete the node after the given data

7.Search

8.Show

9.Exit

Enter your choice?

2

Enter value40

node inserted

\*\*\*\*\*\*\*\*\*Main Menu\*\*\*\*\*\*\*\*\*

Choose one option from the following list ...

===============================================

1.Insert in begining

2.Insert at last

3.Insert at any random location

4.Delete from Beginning

 5.Delete from last

 6.Delete the node after the given data

7.Search

8.Show

9.Exit

Enter your choice?

8

 printing values...

20

10

30

40

\*\*\*\*\*\*\*\*\*Main Menu\*\*\*\*\*\*\*\*\*

Choose one option from the following list ...

===============================================

1.Insert in begining

2.Insert at last

3.Insert at any random location

4.Delete from Beginning

 5.Delete from last

 6.Delete the node after the given data

7.Search

8.Show

9.Exit

Enter your choice?

4

node deleted

\*\*\*\*\*\*\*\*\*Main Menu\*\*\*\*\*\*\*\*\*

Choose one option from the following list ...

===============================================

1.Insert in begining

2.Insert at last

3.Insert at any random location

4.Delete from Beginning

 5.Delete from last

 6.Delete the node after the given data

7.Search

8.Show

9.Exit

Enter your choice?

5

node deleted

\*\*\*\*\*\*\*\*\*Main Menu\*\*\*\*\*\*\*\*\*

Choose one option from the following list ...

===============================================

1.Insert in begining

2.Insert at last

3.Insert at any random location

4.Delete from Beginning

 5.Delete from last

 6.Delete the node after the given data

7.Search

8.Show

9.Exit

Enter your choice?

9