**DSA Assignment: 5**

**Exp 5:** [Implementation of Singly Linked List](https://classroom.google.com/c/NDg4NTg4NTM0NDIz/a/NTM4MDQwNzg0MzIx/details" \t "_self)

Shashwat Tripathi

D10A Roll No: 60

**AIM:** In this experiment, we will implement Singly Linked List.

**CODE:**

// Exp 05 Implementation of Singly Linked List.

#include <stdio.h>

#include <stdlib.h>

struct node

{

    int data;

    struct node \*next;

};

struct node \*start = NULL;

struct node \*create(struct node \*);

struct node \*display(struct node \*);

struct node \*insertbeginning(struct node \*);

struct node \*insertend(struct node \*);

struct node \*insertmiddle(struct node \*);

struct node \*deletebeginning(struct node \*);

struct node \*deleteend(struct node \*);

struct node \*deletemiddle(struct node \*);

int main(int argc, char \*argv[])

{

    int choice;

    printf("D10A\_60\_Shashwat Tripathi");

    printf("\n#############################################\n");

    printf("Your choices are: ");

        printf("\n 1: Create list");

        printf("\n 2: Display list");

        printf("\n 3: Add a node at the start");

        printf("\n 4: Add a node at the end");

        printf("\n 5: Add a node in the middle");

        printf("\n 6: Delete a node from the beginning");

        printf("\n 7: Delete a node from the end");

        printf("\n 8: Delete a node after a given node");

        printf("\n 9: EXIT");

    printf("\n#############################################\n");

    do

    {

        printf("\n Enter your choice : ");

        scanf("%d", &choice);

        switch (choice)

        {

        case 1:

            start = create(start);

            printf("\n LINKED LIST CREATED");

            break;

        case 2:

            start = display(start);

            break;

        case 3:

            start = insertbeginning(start);

            break;

        case 4:

            start = insertend(start);

            break;

        case 5:

            start = insertmiddle(start);

            break;

        case 6:

            start = deletebeginning(start);

            break;

        case 7:

            start = deleteend(start);

            break;

        case 8:

            start = deletemiddle(start);

            break;

        }

    } while (choice != 9);

    return 0;

}

struct node \*create(struct node \*start)

{

    struct node \*new\_node, \*ptr;

    int num;

    printf("\n Enter -1 to end");

    printf("\n Enter the data : ");

    scanf("%d", &num);

    while (num != -1)

    {

        new\_node = (struct node \*)malloc(sizeof(struct node));

        new\_node->data = num;

        if (start == NULL)

        {

            new\_node->next = NULL;

            start = new\_node;

        }

        else

        {

            ptr = start;

            while (ptr->next != NULL)

                ptr = ptr->next;

            ptr->next = new\_node;

            new\_node->next = NULL;

        }

        printf("\n Enter the data : ");

        scanf("%d", &num);

    }

    return start;

}

struct node \*display(struct node \*start)

{

    struct node \*ptr;

    ptr = start;

    while (ptr != NULL)

    {

        printf("\t %d", ptr->data);

        ptr = ptr->next;

    }

    return start;

}

struct node \*insertbeginning(struct node \*start)

{

    struct node \*new\_node;

    int num;

    printf("\n Enter the data : ");

    scanf("%d", &num);

    new\_node = (struct node \*)malloc(sizeof(struct node));

    new\_node->data = num;

    new\_node->next = start;

    start = new\_node;

    return start;

}

struct node \*insertend(struct node \*start)

{

    struct node \*ptr, \*new\_node;

    int num;

    printf("\n Enter the data : ");

    scanf("%d", &num);

    new\_node = (struct node \*)malloc(sizeof(struct node));

    new\_node->data = num;

    new\_node->next = NULL;

    ptr = start;

    while (ptr->next != NULL)

        ptr = ptr->next;

    ptr->next = new\_node;

    return start;

}

struct node \*insertmiddle(struct node \*start)

{

    struct node \*new\_node, \*ptr, \*preptr;

    int num, val;

    printf("\n Enter the data : ");

    scanf("%d", &num);

    printf("\n Enter the value after which the data has to be inserted : ");

    scanf("%d", &val);

    new\_node = (struct node \*)malloc(sizeof(struct node));

    new\_node->data = num;

    ptr = start;

    preptr = ptr;

    while (preptr->data != val)

    {

        preptr = ptr;

        ptr = ptr->next;

    }

    preptr->next = new\_node;

    new\_node->next = ptr;

    return start;

}

struct node \*deletebeginning(struct node \*start)

{

    struct node \*ptr;

    ptr = start;

    start = start->next;

    free(ptr);

    return start;

}

struct node \*deleteend(struct node \*start)

{

    struct node \*ptr, \*preptr;

    ptr = start;

    while (ptr->next != NULL)

    {

        preptr = ptr;

        ptr = ptr->next;

    }

    preptr->next = NULL;

    free(ptr);

    return start;

}

struct node \*deletemiddle(struct node \*start)

{

    struct node \*ptr, \*preptr;

    int val;

    printf("\n Enter the value of the node which has to be deleted : ");

    scanf("%d", &val);

    ptr = start;

    if (ptr->data == val)

    {

        start = deletebeginning(start);

        return start;

    }

    else

    {

        while (ptr->data != val)

        {

            preptr = ptr;

            ptr = ptr->next;

        }

        preptr->next = ptr->next;

        free(ptr);

        return start;

    }

}

**OUTPUT:**



