Department of
Computer Science and Engineering

Title: Implementation of Doubly Linked List

Data Structure Lab
CSE 106

Green University of Bangladesh

1 Objective(s)

- To attain knowledge on **doubly linked list**.
- To implement **doubly linked list** using C.

2 Problem Analysis

A linked List is a linear collection of data elements whose order is not given by their physical placement in memory. The elements in a linked list are linked using pointers. It is a data structure consisting of a collection of nodes which together represent a sequence. This structure allows for efficient insertion or removal of elements from any position in the sequence during iteration.

Doubly Linked List - It is also known as two way linked list. A two-way linked list is a more complex type of linked list which contains a pointer to the next as well as the previous node in sequence, Therefore, it contains three parts are data, a pointer to the next node, and a pointer to the previous node. This would enable us to traverse the list in the backward direction as well.
The nodes are connected to each other in back and forth where the value of the next variable of the last node is NULL i.e. next = NULL, which indicates the end of the doubly linked list and value of the previous variable of the first node is NULL i.e. previous = NULL, which indicates the beginning of the doubly linked list

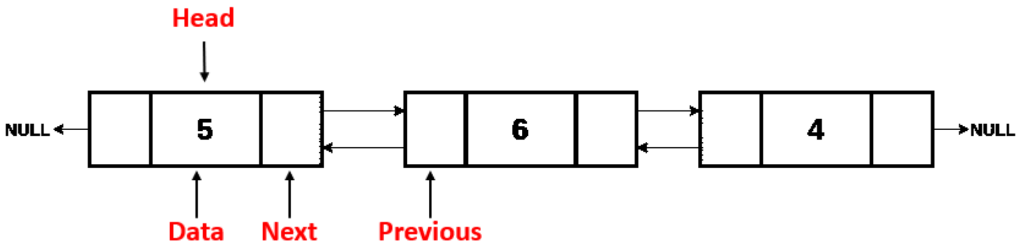

Figure 1: Doubly Linked List

2.1 Basic Operations of a Doubly Linked List

- **Insert**
  - **At the Beginning Position**
  - **At the Last Position**
  - **At any Specific Position**
- **Delete**
  - **From the Beginning Position**
  - **From the Last Position**
  - **From any Specific Position**
- **Traverse**
- **Display**
- **Search**

3 Algorithm

Algorithm 1: Setting up an empty list

```
/* Algorithm for Setting up an empty list */
Include all the header files which are used in the program.
Declare all the user defined functions.
Define a Node structure with three members data, next and previous.
Define a Node pointer 'head' and set it to NULL.
Implement the main method by displaying operations menu and make suitable function calls in the
  main method to perform user selected operation.
```

Algorithm 2: Inserting at beginning of the doubly linked list

```
Input: Element
/* Algorithm for inserting at beginning of the doubly linked list */
Create a newNode
newNode -> data = value
newNode→previous = NULL
if head == NULL then
    newNode→next = NULL
    head = newNode
end
else
    newNode→next = head
    head = newNode
end
print "Insertion success!!!"
```

Algorithm 3: inserting at last position of the doubly linked list

```
Input: Element
/* Algorithm for inserting at last position of the doubly linked list */
Create a newNode
newNode -> data = value
newNode -> next = NULL
if head == NULL then
    newNode -> previous = NULL
    head = newNode;
end
else
    define a node pointer temp
    temp = head
    while temp->next != NULL do
        temp = temp->next
    end
    temp -> next = newNode
    newNode -> previous = temp
end
print "Insertion success!!!"
```

Algorithm 4: Deleting from the beginning of the doubly linked list

```
Input: Element
/* Algorithm for deleting from the beginning of the doubly linked list */
if head == NULL then
    print "List is Empty!!! Deletion not possible!!!"
end
else
    define a node pointer temp
    temp = head
    if temp -> previous == temp -> next  head = NULL; free(temp);  else head = temp -> next; head
      -> previous = NULL; free(temp);  print "Deletion success!!!"
end
```

Algorithm 5: Displaying the doubly linked list

```
/* Algorithm for displaying the doubly linked list */
if head == NULL then
    Print “List is Empty!!!”
end
else
    define a node pointer temp
    temp = head
    Print “List elements are: "
    print "NULL <— "
    while temp->next != NULL do
        print "%d <===> ",temp -> data
        temp = temp->next
    end
    print "%d —> NULL", temp -> data
end
```

4 Flowchart

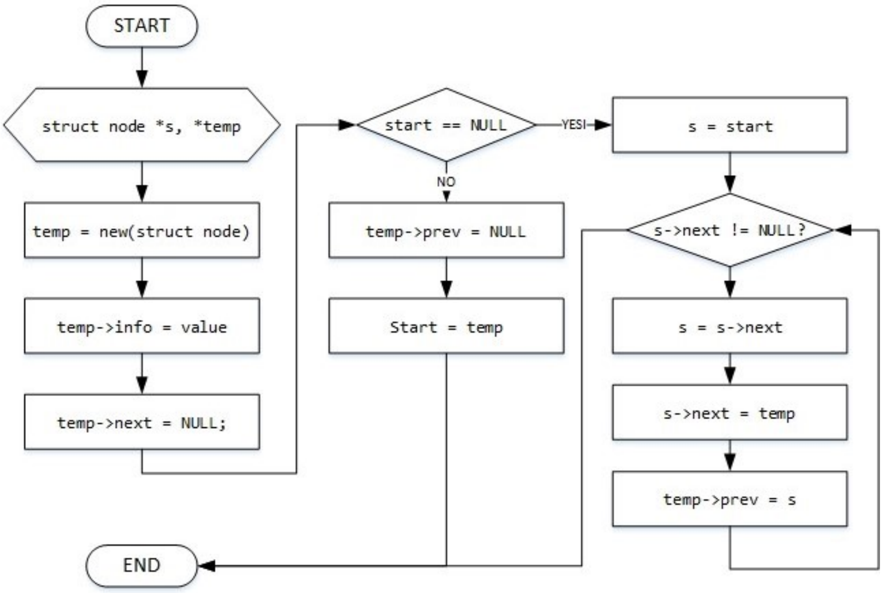

Figure 2: Create Doubly Linked List

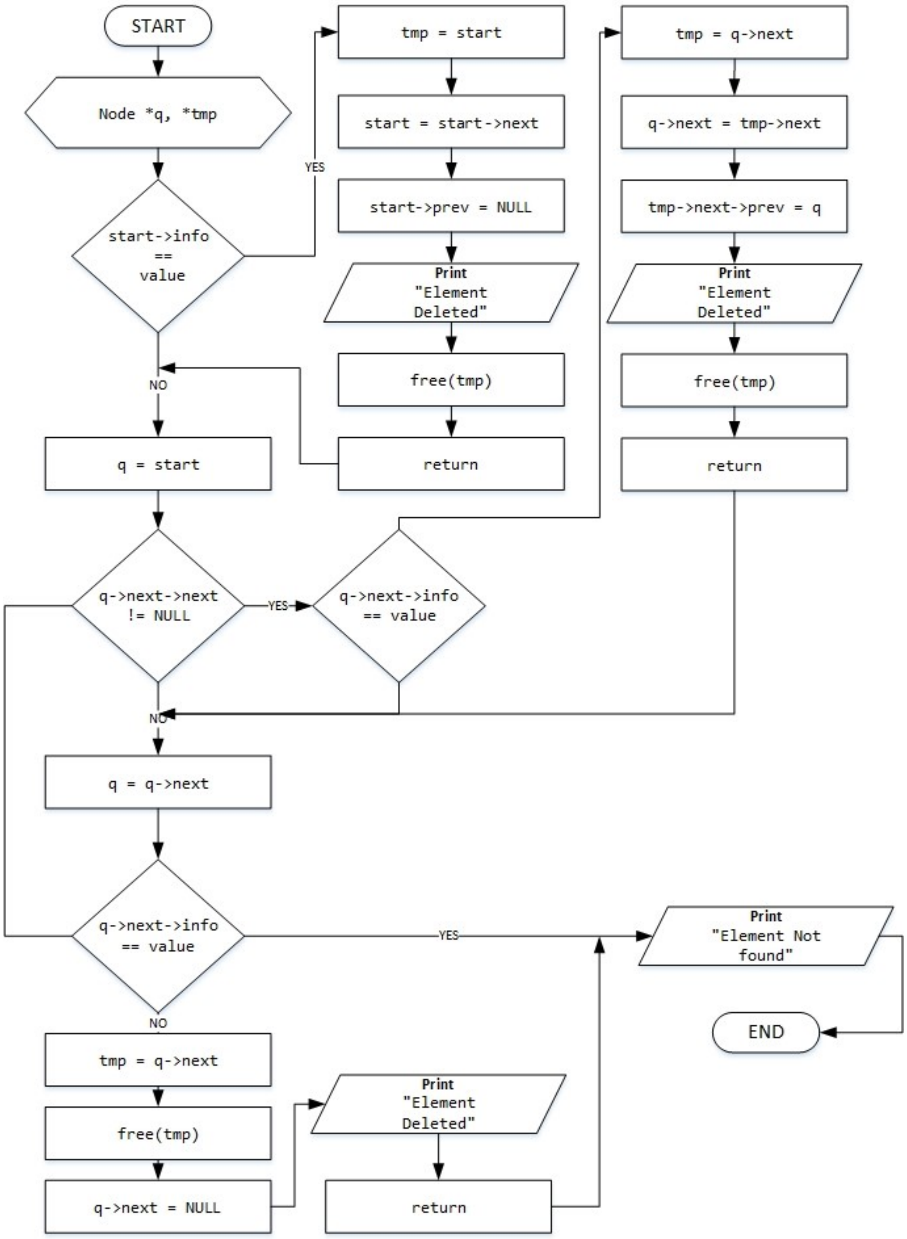

Figure 3: Delete element from Doubly Linked List

5 Implementation in C

```c
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>

struct Node
{
    int data;
    struct Node *previous, *next;
}*head = NULL;

void insertAtBeginning(int value)
{
    struct Node *newNode;
    newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode -> data = value;
    newNode -> previous = NULL;
    newNode -> next = NULL;
    if(head == NULL)
    {
        newNode -> next = NULL;
        head = newNode;
    }
    else
    {
        newNode -> next = head;
        head = newNode;
    }
    printf("\nInsertion success!!!");
}

void insertAtEnd(int value)
{
    struct Node *newNode;
    newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode -> data = value;
    newNode -> next = NULL;
    if(head == NULL)
    {
        newNode -> previous = NULL;
        head = newNode;
    }
    else
    {
        struct Node *temp = head;
        while(temp -> next != NULL)
            temp = temp -> next;
        temp -> next = newNode;
        newNode -> previous = temp;
    }
    printf("\nInsertion success!!!");
}

void deleteBeginning()
{
    if(head == NULL)
        printf("List is Empty!!! Deletion not possible!!!");
    else
    {
        struct Node *temp = head;
        if(temp -> previous == temp -> next)
        {
            head = NULL;
            free(temp);
        }
        else{
            head = temp -> next;
            head -> previous = NULL;
            free(temp);
        }
        printf("\nDeletion success!!!");
    }
}

void display()
{
    if(head == NULL)
        printf("\nList is Empty!!!");
    else
    {
        struct Node *temp = head;
        printf("\nList elements are: \n");
        printf("NULL <--- ");
        while(temp -> next != NULL)
        {
            printf("%d <===> ",temp -> data);
            temp = temp->next;
        }
        printf("%d ---> NULL", temp -> data);
    }
}

int main()
{
    int choice1, choice2, value;
    while(1)
    {
        Start:
        printf("\n*********** MENU *************\n");
        printf("1. Insert\n2. Delete\n3. Display\n4. Exit\nEnter your choice: ");
        scanf("%d",&choice1);
        switch(choice1)
        {
            case 1: printf("Enter the value to be inserted: ");
                scanf("%d",&value);
                while(1)
                {
                    printf("\nSelect from the following Inserting options\n");
                  printf("1. At Beginning\n2. At End\n3. Cancel\nEnter your choice: ");
                    scanf("%d",&choice2);
                    switch(choice2)
                    {
                        case 1:
                            insertAtBeginning(value);
                            break;
                        case 2:
                            insertAtEnd(value);
                          break;
                        case 3:
                            goto EndSwitch;
                        default:
                            printf("\nPlease select correct Inserting option!!!\n");
                    }
                    goto Start;
                }
                break;
            case 2:
                while(1)
                {
                    printf("\nSelect from the following Deleting options\n");
                  printf("1. At Beginning\n2. Cancel\nEnter your choice: ");
                    scanf("%d",&choice2);
                    switch(choice2)
                    {
                        case 1:
                            deleteBeginning();
                          break;
                        case 2:
                            goto EndSwitch;
                        default:
                            printf("\nPlease select correct Deleting option!!!\n");
                    }
                    goto Start;
                }
                break;
                EndSwitch:
                    break;
            case 3:
                display();
                break;
            case 4:
                exit(0);
                break;
            default:
                printf("\nPlease select correct option!!!");
        }
    }
    return 0;
}
```

6 Input/Output (Compilation, Debugging & Testing)

Input & Output:

```
*********** MENU *************
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 1
Enter the value to be inserted: 100

Select from the following Inserting options
1. At Beginning
2. At End
3. Cancel
Enter your choice: 1

Insertion success!!!
*********** MENU *************
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 1
Enter the value to be inserted: 200

Select from the following Inserting options
1. At Beginning
2. At End
3. Cancel
Enter your choice: 1

Insertion success!!!
*********** MENU *************
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 1
Enter the value to be inserted: 500

Select from the following Inserting options
1. At Beginning
2. At End
3. Cancel
Enter your choice: 2

Insertion success!!!
*********** MENU *************
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 3

List elements are:
NULL <— 200 <===> 100 <===> 500 —> NULL
```

```
*********** MENU *************
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 2

Select from the following Deleting options
1. At Beginning
2. Cancel
Enter your choice: 1

Deletion success!!!
*********** MENU *************
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 3

List elements are:
NULL <— 100 <===> 500 —> NULL
*********** MENU *************
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 4

Process returned 0 (0x0) execution time : 69.744 s
Press any key to continue.
```

7 Discussion & Conclusion

Based on the focused objective(s) and basic operations of a singly linked list, the additional lab exercise made me more confident to have a clear understanding about singly linked list and ultimately lead me towards the fulfilment of the objectives(s).

8 Lab Task (Please implement yourself and show the output to the instructor)

1. Modify the C program that is able to insert element at any specific position, Delete element from the last and any specific position as well as display the list after any modification.

8.1 Algorithm

Algorithm 6: Inserting At Any Specific Position of the list

```
Input: Element
/* Algorithm for Inserting Element at Any Specific Position */
Create a newNode with given value.
newNode -> data = value
if head == NULL then
    newNode→next = NULL
    newNode→previous = NULL
    head = newNode
end
else
    define a node pointer temp1 & temp2
    temp1 = head
    while temp1 -> data != location do
        if temp1 -> next == NULL then
            Print "Given node is not found in the list!!!"
            goto EndFunction
        end
        else
            temp1 = temp1 -> next
        end
    end
    temp2 = temp1 -> next
    temp1 -> next = newNode
    newNode -> previous = temp1
    newNode -> next = temp2
    temp2 -> previous = newNode
    print "One node inserted!!!"
end
EndFunction:
```

9 Lab Exercise (Submit as a report)

1. Find the specific node of element that is present or not in the singly linked list.
2. Call a function that will generate the size of the singly linked list.
3. Insert element between any specific position of the singly linked list.

10 Policy

Copying from internet, classmate, seniors, or from any other source is strongly prohibited. 100% marks will be *deducted* if any such copying is detected.