DHAKA UNIVERSITY OF ENGINEERING & TECHNOLOGY, GAZIPUR

Department of Computer Science and Engineering

Course No.: **CSE-2112**

Course Title: **Object Oriented Programming Language Sessional**

Lab No: **06**
Lab Name: Linked List using pointer.

Experiment Date: **27-04-2021**

Submission Date: **03-05-2021**

Submitted To:

Mr.Md. Omor Farque

Associative Professor, Department of CSE

Mss. Sabah Binte Noor

Associative Professor, Department of CSE

Submitted By-

Name: **Mehedi Hasan Shuvo**

Student Id.: 194016

Year: 2nd

Semester: 1st

Session: 2019 - 2020

Problems No: 01

Problem Title:

Linked List:

A linked list is a linear data structure and also a collection of elements, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers as shown in the below image:

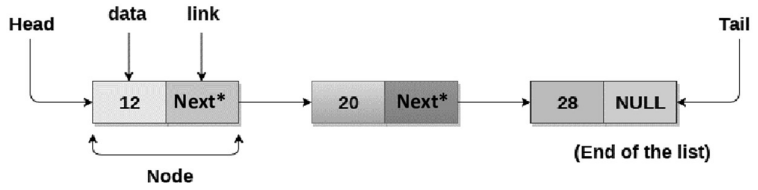

Fig.: Singly linked list or One way

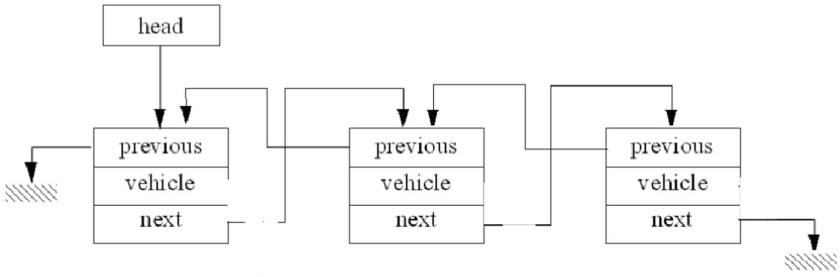

Fig.: Doubly linked list or Two way

Problem: In this problem you have to develop a menu driven program to implement a Doubly linked list of vehicles (ID, Type, Model, Price) in OOP concept. The linked have to support following operation:

[A]: Add a vehicle at beginning

[B]: Add a vehicle at end

[D] Delete a vehicle with ID

[S]: Show all vehicles from end

[N]: Number of vehicles

[Q]: Quit

The user should be prompted to give one of the above characters, upper or lower case, and the program should proceed accordingly. If any other character is given the program should insist on getting one of the specified characters.

Solution:

```
#include<bits/stdc++.h>
using namespace std;

class Vehical
{
public:
    int id;
    string model;
    double price;
    string type;
    Vehical* previous;
    Vehical* next;
};

void push(Vehical ** head_ref,int new_id,string new_model,string new_type,double new_price)
{
    Vehical* new_vehical=new Vehical();

    // put in the data
    new_vehical->id=new_id;
    new_vehical->model=new_model;
    new_vehical->price=new_price;
    new_vehical->type=new_type;

    // make next of new node as head and previously as NULL
    new_vehical->next=(*head_ref);
    new_vehical->previous=NULL;

    // Change Prev of head node to new node
    if((*head_ref) !=NULL)
    {
        (*head_ref)->previous=new_vehical;
    }

    // Move the head to point to the new node
    (*head_ref)=new_vehical;
}

void append(Vehical ** head_ref,int new_id,string new_model,string new_type,double new_price)
{
    //1. Allocate node
    Vehical* new_vehical=new Vehical();
    Vehical* last=*head_ref;
```

```
    // put in the data
    new_vehical->id=new_id;
    new_vehical->model=new_model;
    new_vehical->price=new_price;
    new_vehical->type=new_type;

    // this new node is going to be the last node, sp make next of it as NULL
    new_vehical->next=NULL;

    // if the linked list is empty, then make the new node as head
    if(*head_ref==NULL)
    {
        new_vehical->previous=NULL;
        *head_ref=new_vehical;
        return;
    }


    // else traverse till the last node
    while(last->next !=NULL)
        last=last->next;

    // change the next of last node
    last->next=new_vehical;

    // make last node as previous of new node
    new_vehical->previous=last;
    return;
}

void addAVehical(Vehical **vehial,bool isAddBeganning,bool isAddEnd)
{
    int id;
    string model;
    double price;
    string type;
    cout<<"\nEnter ID: ";
    cin>>id;
    cout<<"Enter Vehical Model: ";
    cin>>model;
    cout<<"Enter Vehical Type: ";
    cin>>type;
    cout<<"Enter Vehical Price: ";
    cin>>price;
    cout<<endl;
```

```
    if(isAddBeganning==true)
        push(vehial,id,model,type,price);

    if(isAddEnd==true)
        append(vehial,id,model,type,price);

}

void deleteNode(Vehical** head_ref,Vehical* del)
{
    //base case
    if(*head_ref==NULL || del==NULL) return;

    // if node to be deleted is head node
    if(*head_ref==del) *head_ref=del->next;

    // change next only if node to be deleted is not the last node
    if(del->next !=NULL) del->next->previous=del->previous;

    // change prev only if node to be deleted is not the first node
    if(del->previous!=NULL) del->previous->next=del->next;

    // finally free the memory occupied by delete
    free(del);
}


void deleteAllOccurofX(Vehical** head_ref,int x)
{
    //if list is empty
    if((*head_ref)==NULL) return;

    Vehical* current=*head_ref;
    Vehical* next;

    // traverse the list up to the end
    while (current !=NULL)
    {
        // if node found with the value of X
        if(current->id==x)
        {
            //save currents next node in the pointer next
            next=current->next;
```

```
            //delete the node pointer to by current
            deleteNode(head_ref,current);

            // update current
            current=next;
        }
        else
        {
            current=current->next;
        }
    }

    cout<<x<<" is successfully deleted form vehical list";
}


int countVehicalList(Vehical* vehical)
{
    int count=0;
    Vehical* last;
    while (vehical != NULL)
    {
        last = vehical;
        vehical = vehical->next;
        count++;
    }
    return count;
}


void printVehicalList(Vehical* vehical)
{
    Vehical* last=NULL;
    while (vehical != NULL)
    {
        last = vehical;
        vehical = vehical->next;
    }

    cout<<"\nID-> Type-> Model-> Price\n";
    while (last != NULL)
    {
        cout<<last->id<<" "<<last->type<<" "<<last->model<<" "<<last->price<<" \n";
        last = last->previous;
    }
}
```

```
int main()
{

    Vehical *head_vehial=NULL;
    char choose;
    int query;

    while (choose!='Q' || choose!='q')
    {
        printf("============================================\n");
        printf("DOUBLY LINKED LIST PROGRAM\n");
        printf("============================================\n");
        printf("[A]: Add a vehicle at beginning\n");
        printf("[B]: Add a vehicle at end\n");
        printf("[D]: Delete a vehicle with ID\n");
        printf("[S]: Show all vehicles from end\n");
        printf("[N]: Number of vehicles\n");
        printf("[Q]: Quit\n");
        printf("--------------------------------------------\n");
        printf("Enter your choose : ");

        cin>>choose;

        switch (choose)
        {
        case 'a':
        case 'A':
            addAVehical(&head_vehial,true,false);
            break;

        case 'b':
        case 'B':
            addAVehical(&head_vehial,false,true);
            break;

        case 'd':
        case 'D':
            cout<<"Enter Vehical ID: ";
            cin>>query;
            deleteAllOccurofX(&head_vehial,query);
            break;
```

```
        case 's':
        case 'S':
            printVehicalList(head_vehial);
            break;

        case 'n':
        case 'N':
            cout<<"Length of Vehical list is: "<<countVehicalList(head_vehial);
            break;

        case 'q':
        case 'Q':
            return 0;

        default:
            cout<<"Error! Invalid choice. Please choose between 0-5";

        }

        printf("\n\n");
    }

    return 0;

}
```

Output:

```
==========================================
DOUBLY LINKED LIST PROGRAM
==========================================
[A]: Add a vehicle at beginning
[B]: Add a vehicle at end
[D]: Delete a vehicle with ID
[S]: Show all vehicles from end
[N]: Number of vehicles
[Q]: Quit
------------------------------------------
Enter your choose : a

Enter ID: 1001
Enter Vehical Model: model_1
Enter Vehical Type: type_1
Enter Vehical Price: 20



==========================================
DOUBLY LINKED LIST PROGRAM
==========================================
[A]: Add a vehicle at beginning
[B]: Add a vehicle at end
[D]: Delete a vehicle with ID
[S]: Show all vehicles from end
[N]: Number of vehicles
[Q]: Quit
------------------------------------------
Enter your choose : a

Enter ID: 1002
Enter Vehical Model: model_2
Enter Vehical Type: type_2
Enter Vehical Price: 30
```

```
==========================================
DOUBLY LINKED LIST PROGRAM
==========================================
[A]: Add a vehicle at beginning
[B]: Add a vehicle at end
[D]: Delete a vehicle with ID
[S]: Show all vehicles from end
[N]: Number of vehicles
[Q]: Quit
------------------------------------------
Enter your choose : s

ID-> Type-> Model-> Price
1001 type_1 model_1 20
1002 type_2 model_2 30


==========================================
DOUBLY LINKED LIST PROGRAM
==========================================
[A]: Add a vehicle at beginning
[B]: Add a vehicle at end
[D]: Delete a vehicle with ID
[S]: Show all vehicles from end
[N]: Number of vehicles
[Q]: Quit
------------------------------------------
Enter your choose : b

Enter ID: 3003
Enter Vehical Model: model_3
Enter Vehical Type: type_3
Enter Vehical Price: 30
```

```
==========================================
DOUBLY LINKED LIST PROGRAM
==========================================
[A]: Add a vehicle at beginning
[B]: Add a vehicle at end
[D]: Delete a vehicle with ID
[S]: Show all vehicles from end
[N]: Number of vehicles
[Q]: Quit
------------------------------------------
Enter your choose : s

ID-> Type-> Model-> Price
3003 type_3 model_3 30
1001 type_1 model_1 20
1002 type_2 model_2 30


==========================================
DOUBLY LINKED LIST PROGRAM
==========================================
[A]: Add a vehicle at beginning
[B]: Add a vehicle at end
[D]: Delete a vehicle with ID
[S]: Show all vehicles from end
[N]: Number of vehicles
[Q]: Quit
------------------------------------------
Enter your choose : n
Length of Vehical list is: 3

------------------------------------------
```

```
==========================================
DOUBLY LINKED LIST PROGRAM
==========================================
[A]: Add a vehicle at beginning
[B]: Add a vehicle at end
[D]: Delete a vehicle with ID
[S]: Show all vehicles from end
[N]: Number of vehicles
[Q]: Quit
------------------------------------------
Enter your choose : d
Enter Vehical ID: 1001
1001 is successfully deleted form vehical list

==========================================
DOUBLY LINKED LIST PROGRAM
==========================================
[A]: Add a vehicle at beginning
[B]: Add a vehicle at end
[D]: Delete a vehicle with ID
[S]: Show all vehicles from end
[N]: Number of vehicles
[Q]: Quit
------------------------------------------
Enter your choose : s

ID-> Type-> Model-> Price
3003 type_3 model_3 30
1002 type_2 model_2 30
```

```
==========================================
DOUBLY LINKED LIST PROGRAM
==========================================
[A]: Add a vehicle at beginning
[B]: Add a vehicle at end
[D]: Delete a vehicle with ID
[S]: Show all vehicles from end
[N]: Number of vehicles
[Q]: Quit
------------------------------------------
Enter your choose : q

Process returned 0 (0x0)   execution time : 192.969 s
Press any key to continue.
```