



THE UNIVERSITY
OF LAHORE
**ISLAMABAD
CAMPUS**

Data Structures & Algorithms (CS09203)

Lab Report

Name: Muhammad Umer
Registration #: CSU-F16-104
Lab Report #: 04
Dated: 23-04-2018
Submitted To: Mr. Usman Ahmed

The University of Lahore, Islamabad Campus
Department of Computer Science & Information Technology

Experiment # 1

Link List-Basic Insertion and Traversal

Objective

To understand and implement the Link List with basic Insertion and Traversal.

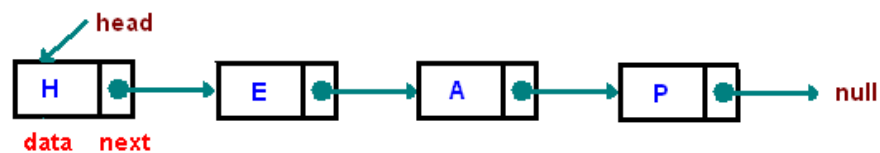
Software Tool

1. Sublime Text Editor
2. Dev C++
3. Window 7 (32 Bit)

1 Theory

One disadvantage of using arrays to store data is that arrays are static structures and therefore cannot be easily extended or reduced to fit the data set. Arrays are also expensive to maintain new insertions and deletions. In this chapter we consider another data structure called Linked Lists that addresses some of the limitations of arrays.

A linked list is a linear data structure where each element is a separate object.



Each element (we will call it a node) of a list is comprising of two items - the data and a reference to the next node. The last node has a reference to null. The entry point into a linked list is called the head of the list. It should be noted that head is not a separate node, but the reference to the first node. If the list is empty then the head is a null reference.

A linked list is a dynamic data structure. The number of nodes in a list is not fixed and can grow and shrink on demand. Any application which has to deal with an unknown number of objects will need to use a linked list.

One disadvantage of a linked list against an array is that it does not allow direct access to the individual elements. If you want to access a particular item then you have to start at the head and follow the references until you get to that item.

2 Task

2.1 Procedure: Task 1 Insertion

In this Linked List user can insert integer type of the data and the data will always be inserted in the start of the list.

```
void insert(int item){
    node *NewNode = (node*) malloc(sizeof(node));
    NewNode -> data = item;
    NewNode -> next = head;
    head = NewNode;
}
```

2.2 Procedure: Task 2 Display

```
void display(){
    if(head == NULL){
        cout<<"\n\nError: Empty List!";
        cout<<"\n\nPress any key to continue...";
        getch();
        return;
    }
    node *NewNode = (node*) malloc(sizeof(node));
    NewNode = head;
```

```

C:\Users\Student\Documents\program.exe
Enter number of vertices:8
Enter adjacency matrix of the graph:0 1 1 1 1 0 0 0
1 0 0 0 0 1 0 0
1 0 0 0 0 1 0 0
1 0 0 0 0 0 1 0
1 0 0 0 0 0 1 0
0 1 1 0 0 0 0 1
0 0 0 1 1 0 0 1
0 0 0 0 0 1 1 0

0
1
5
2
7
6
3
4
Process returned 8 (0x8)   execution time : 64.785 s
Press any key to continue.

```

Figure 1: Time Independent Feature Set

```

cout<<"\n\nData_in_the_List:\n\n";
while(NewNode != NULL){
    cout<<NewNode->data<<" ";
    NewNode = NewNode->next;
}
}

```

3 Program

```

#include<iostream>
#include<stdlib.h>
#include<conio.h>
using namespace std;

struct node{
    int data;
    node* next;
};

node* head = NULL;

void insert(int item){

```

```

        node *NewNode = (node*) malloc(sizeof(node));
        NewNode -> data = item;
        NewNode -> next = head;
        head = NewNode;
        cout<<"\n\nData Inserted Successfully!";
        cout<<"\n\nPress any key to continue ...";
        getch();
    }

    void display(){
        if(head == NULL){
            cout<<"\n\nError: Empty List!";
            cout<<"\n\nPress any key to continue ...";
            getch();
            return;
        }
        node *NewNode = (node*) malloc(sizeof(node));
        NewNode = head;
        cout<<"\n\nData in the List:\n\n";
        while(NewNode != NULL){
            cout<<NewNode -> data<<" ";
            NewNode = NewNode -> next;
        }
        cout<<"\n\nPress any key to continue ...";
        getch();
    }

    int main(){
        int choice, item;
        up:
        system("cls");
        cout<<"\n\n\tCHOOSE FROM THE FOLLOWING:";
        cout<<"\n\n\t1. Insert Data in the List";
        cout<<"\n\n\t2. Display Data of the List";
        cout<<"\n\n\t3. Exit\n\n\t";
        cin>>choice;
        if(choice == 1){
            cout<<"\n\nEnter a value: ";
            cin>>item;
            insert(item);

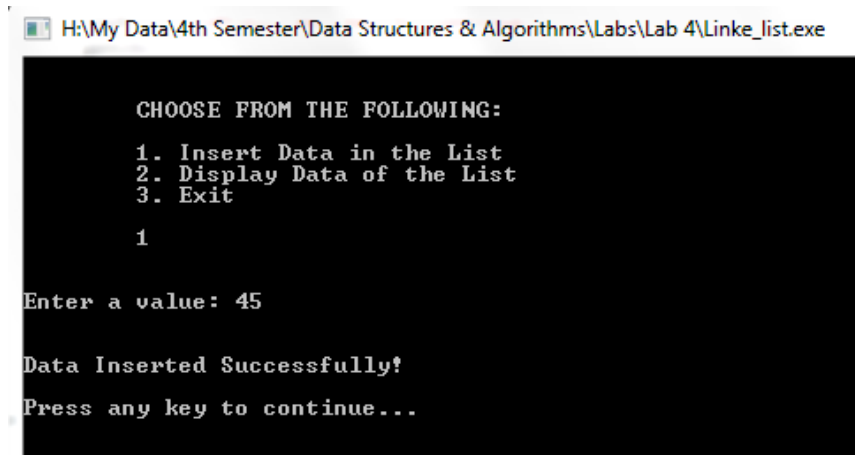
```

```

        goto up;
    }
    else if(choice == 2){
        display();
        goto up;
    }
    else if(choice == 3){
        exit(0);
    }
    else{
        cout<<"\n\nWRONG_CHOICE!";
        cout<<"\n\nPress any key to choose again ...";
        getch();
        goto up;
    }
    return 0;
}

```

4 Output



```

H:\My Data\4th Semester\Data Structures & Algorithms\Labs\Lab 4\Linke_list.exe

CHOOSE FROM THE FOLLOWING:
1. Insert Data in the List
2. Display Data of the List
3. Exit
1

Enter a value: 45

Data Inserted Successfully!
Press any key to continue...

```

Figure 2: Main Menu and Insertion

```
H:\My Data\4th Semester\Data Structures & Algorithms\Labs\Lab 4\Linke_list.exe

CHOOSE FROM THE FOLLOWING:
1. Insert Data in the List
2. Display Data of the List
3. Exit

2

Data in the List:
-56 34 67 45

Press any key to continue...
```

Figure 3: Displaying List

```
H:\My Data\4th Semester\Data Structures & Algorithms\Labs\Lab 4\Linke_list.exe

CHOOSE FROM THE FOLLOWING:
1. Insert Data in the List
2. Display Data of the List
3. Exit

3

-----
Process exited after 84.34 seconds with return value 0
Press any key to continue . . .
```

Figure 4: Exit

5 Conclusion

A linked list is a linear data structure where each element is a separate object. Each element is called as a node, that contains two item - the data and a reference to the next node. The last node has a reference to null. The entry point into a linked list is called the head of the list. A linked list is a dynamic data structure. The number of nodes in a list is not fixed and can

grow and shrink on demand.

(Concerned Teacher/Lab Engineer)