Lab Manual for Data Structures

Lab 6
Linked List Implementation

Table of Contents

Lab 6: Linked List Implementation

1. Introduction

This lab will introduce concept of dynamic memory based data structures such as linked lists. In case of some real world applications we don't know about the storage requirements of program before program execution, in that scenario dynamic data storage systems are useful. We will learn about linear linked list structure and how linked lists are helpful to store data during program execution.

1.1 Linear Linked Lists

A linked list is a collection of elements, called nodes. Every node (except the last node) stores the address of the next node. Therefore, every node in a linked list has two components: one to store the data and other to store the address, called the link, of the next node in the list. The address of the first node in the list is stored in a separate location, called the head or first.

Figure 1 is a representation of a node.

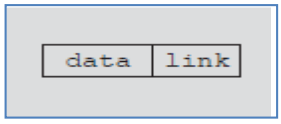

Figure 1: Structure of a node.

Linked list: A list of elements, called nodes, in which the order of the nodes is determined by the address, called the link, stored in each node.

The list in Figure 2 is an example of a linear linked list.

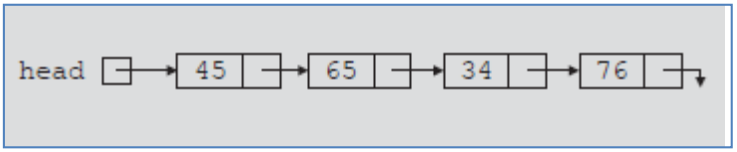

Figure 2: Linear Linked List of nodes

We may perform different operations on linked lists such as inserting a node, deleting a node, traversing a linked list etc.

1.2Relevant Lecture Readings

a) Revise Lecture No. 21 and 22 available at \\fs\lectures$ in instructor's folder.
b) From books: C++ Data Structures by Nell Dale (Page334 - 358) and Data structures using C++ by D. S. Malik (Page 266 – 280, 326 – 338, 310 - 320).

2. Activity Time boxing

Table 1: Activity Time Boxing

Task No.	Activity Name	Activity time	Total Time
5.1	Design Evaluation	20mins	20mins
6.2	Walk through tasks	20mins	20mins
7	Practice tasks	20mins for task 1 , 30 mins for task 2 and 30 mins for task 3	80mins
9	Evaluation Task	60mins for all assigned tasks	50mins

3. Objectives of the experiment

- To understand and implement linear linked lists with their operations in C++.

4. Concept Map

This concept map will help students to understand the main concepts of topic covered in lab.

4.1 Linear Linked Lists in C++

Node of a linked list is a self-referential structure which can be represented by following code.

```
Struct nodeType
{
     int info;
     nodeType *link;
};
```

The basic operations of a linked list are as follows: Search the list to find out whether aparticular item is in the list, insert an item in the list, and delete an item from the list.These operations require the list to be traversed. That is, given a pointer to the first nodeof the list, we must step through the nodes of the list. Following code segment helps in traversal.

```
current = head;
while (current != NULL)
{
//Process current
current = current->link;
}
```

This section discusses how to insert an item into, and delete an item from, a linked list.Consider the following definition of a node. (For simplicity, we assume that the infotype is int.)

```
struct nodeType
{
int info;
nodeType *link;
```

```
};
```

We will use the following variable declaration:

```
nodeType *head, *p, *q, *newNode;
```

Suppose that p points to the node with info 65, and a new node with info 50 is to becreated and inserted after p. Consider the following statements:

```
newNode = new nodeType; //create newNode
newNode->info = 50; //store 50 in the new node
newNode->link = p->link;
p->link = newNode;
```

To delete a node from linked list we can use following code segment.

```
q = p->link;
p->link = q->link;
delete q;
```

5. Homework before Lab

This homework will help students to study the concepts of topic before start of lab.

5.1 Problem Solution Modeling

After studying the introduction and concept map sections you should be ready to provide the solution of following problems. Design the solutions of the problems in C++ and bring your code to lab so that lab instructor should assess and grade it.

5.2Problem description:

Write a program of linear linked list of objects of “person” class. Attributes of “person” class (privately defined) are per_id (int), per_name (string) and per_age (int). “person” class contains member functions (publicly defined): constructor, input and output functions. You are required to define a class for linear linked list. This class should contain the member functions to insert, delete nodes from linear linked list. This class should also contain the member functions to display values of all nodes of linear linked list.

5.3Practices from home

5.3.1 Task-1

Compare linear linked list with circular linked list and provide three differences between the two structures.

5.3.2 Task-2

List three real world examples in which linked list structures can be used.

6. Procedure& Tools

This section provides information about tools and programming procedures used for the lab.

6.1 Tools

Microsoft Visual Studio 2017 with Visual C++ compiler configured.

6.2 Walk through Tasks [Expected time = 20mins]

Following screens in figure 5 to 13represent source code implementation of a linear linked list class in Microsoft Visual Studio 2017. You are required to type, debug and execute this program in Microsoft Visual Studio 2017.

This linked list program provides different functions like addition of node at start of list, deletion of node from list, displaying elements of list, addition of node at end of list, add a node after some specific node in the list.

```
#include<iostream>
using namespace std;
class node
{
public:
     int data;
     node *next;
     node()
     {
          data=0;
          next=NULL;
     }
     node(int x)
     {
          data=x;
          next=NULL;
     }
     int getdata()
     {
          return data;
     }
     void setdata(int x)
     {
          data=x;
     }
```

Figure 5: Implementation of linear linked list class in Visual Studio 2017

```
    node* getnext()
    {
        return next;
    }

    void setnext(node *n)
    {
        next=n;
    }
};

class linked_list
{
    node *head;
public:
    int addNodeAtFront(node *n);
    int isEmpty();
    node* getLastNode();
    int addNodeAtEnd(node *n);
    node* search(int k);
    node* deleteNode(node*);
    void display();
    linked_list()
    {
        head=NULL;
    }
};
```

Figure 6: Implementation of linear linked list class in Visual Studio 2017

```
int linked_list::addNodeAtFront(node *n)
{
    int i=0;
    n->next=head;
    head=n;
    i=1;
    return i;
}
int linked_list::isEmpty()
{
    if(head==NULL)
        return 1;
    else
        return 0;
}
node* linked_list::getLastNode()
{
    node *ptr=head;
    while(ptr->next!=NULL)
    {
        ptr=ptr->next;
    }
    return ptr;
}
```

Figure 7: Implementation of linear linked list class in Visual Studio 2017

```
node* linked_list::search(int k)
    {
        node *ptr=head;int i=0;
        if(head==NULL)
        {
            cout<<"List is empty"<<endl;
            return NULL;
        }
        else
        {
        while(ptr->next!=NULL && ptr->getdata()!=k)
        {
            ptr=ptr->getnext();

            if(ptr->getdata()==k)
            {
                i=1;
                break;
            }
        }
        if(i==0)
        {
            return NULL;
        }
        else
        return ptr;
        }
     }
```

Figure 8: Implementation of linear linked list class in Visual Studio 2017

```
node* linked_list::deleteNode(node* n)
{
    node *ptr=head;
    if(head==NULL)
    {
        cout<<"List is empty"<<endl;
        return NULL;
    }
    else
    {
    if(ptr==n)
    {
        ptr->setnext(n->getnext());
        return n;
    }
    else
    {
        while(ptr->getnext()!=n)
        {
            ptr=ptr->getnext();
        }
        ptr->setnext(n->getnext());
    }
    return n;
    }
}
```

Figure 9: Implementation of linear linked list class in Visual Studio 2017

```
int linked_list::addNodeAtEnd(node *n)
{
    if(head==NULL)
    { head=n; }
    else
    {
        node *n1=getLastNode();
        n1->next=n;
    }
    return 1;
}
void linked_list::display()
{
    node *ptr=head;
    if(head==NULL)
    {
        cout<<"List is empty"<<endl;
    }
    else
    {
        while(ptr!=NULL)
        {
            cout<<"Data="<<ptr->data<<endl;
            ptr=ptr->next;
        }
    }
}
```

Figure 10: Implementation of linear linked list class in Visual Studio 2017

```
void main()
{
    int x,q,ch=0,num,s;
    linked_list l;
    node *n3=new node();
    node *n1=new node();
        while(ch!=7)
        {
            cout<<"Plz enter your choice:\n1. Add node in front\n2.Find last node\n";
            cout<<"3.Add node at end\n4.Search\n5.Delete\n6.Display\n"<<endl;
            cin>>ch;
            switch(ch)
            {
            case 1:
                cout<<"Enter number of elements you want to enter"<<endl;
                cin>>num;
                for(int i=0;i<num;i++)
                {
                    cout<<i+1<<"Enter value"<<endl;
                    cin>>x;
                    node *n4=new node(x);
                    q=l.addNodeAtFront(n4);
                }
                if(q==1)
                {
                    cout<<"Inserted"<<endl;
                    l.display();
                }
                break;
```

Figure 11: Implementation of linear linked list class in Visual Studio 2017

```
        case 2:
            n3=l.getLastNode();
            cout<<"Last node="<<n3->getdata()<<endl;
            break;

        case 3:
            cout<<"Enter number of value you want to add"<<endl;
            cin>>num;
            for(int i=0;i<num;i++)
            {
                cout<<i+1<<"Enter value"<<endl;
                cin>>x;
                node *n=new node(x);
                q=l.addNodeAtEnd(n);
            }
            if(q==1)
            {
                cout<<"Inserted"<<endl;
                l.display();
            }
            break;
```

Figure 12: Implementation of linear linked list class in Visual Studio 2017

```
            case 4:
                cout<<"Enter number  you want to search"<<endl;
                cin>>s;
                n1=l.search(s);
                if(n1==NULL)
                {   cout<<"Match not found"<<endl;  }
                else
                {
                    cout<<"Match found"<<n1->getdata()<<endl;
                }
                break;
            case 5:
                cout<<"Enter number you want to del"<<endl;
                cin>>s;
                n3=l.search(s);
                if(n3==NULL)
                {
                    cout<<"Match not found"<<endl;
                }
                else
                {
                    cout<<"Match found"<<endl;
                    l.deleteNode(n3);
                }
                break;
            case 6:
                l.display();
                break;
            }
        }
    }
```

Figure 13: Implementation of linear linked list class in Visual Studio 2017

7. Practice Tasks

This section will provide information about the practice tasks which are required to be performed in lab session. Design solutions of problems discussed in each task and placesolution code in a folder specified by your lab instructor.

Lab instructors are guided to help students learn how ACM problems work and provide students with certain practice/home tasks on the pattern of ACM Problems.

7.1 Practice Task 1 [Expected time = 20mins]

Implement the following functions for linked list:

1) Insert front
2) Insert End
3) Insert after number
4) Delete front
5) Delete End
6) Delete any number from list
7) Display all element
8) Limit user so that he/she can only add 10 numbers in the list. After that give error that list is full.

7.2 Practice Task 2 [Expected time = 20mins]

Use Practice task 1 and add two functions Remove_Duplicate() and Count() in the class. Remove_Duplicate() function should remove all the number that are occurring more than one time. Count() function should count the number of nodes present at any time in the list.

7.3Practice Task 3 [Expected time = 30mins]

Manage data of Books (Title, Authors, Edition, Price) using linked list. Add Books in the list. Now delete all Books having price more than 2000 from the list.

7.5 Out comes

After completing this lab, student will be able to understand and develop programs related to linear linked lists, circular linked lists and doubly linked listsin C++ using Microsoft Visual Studio 2017 environment.

7.6Testing

Test Cases for Practice Task-1

Sample Input	Sample Output
Enter the values of elements of linked list:	
2	
3	
4	
5	
7	
8	
6	
9	
11	

1	
Enter the value after which you want to insert: 11	New value inserted after 11
Enter the value to delete from linked list: 5	Key value 5 is deleted from linked list.

Test Cases for Practice Task-2

Sample Input	Sample Output
Enter the values of elements of linked list:	
12	
33	
43	
12	
52	
12	
52	
	After Removing duplicates:
	33
	43
	Total nodes present in the list are: 2

Test Cases for Practice Task-3

Sample input/output:

Press 1 to add Book
Press 2 to delete Books having price more than 2000
Press 3 to display list
Enter choice:1
Enter Title: The Art of Computer Programming
Enter Authors: Donald Knuth
Enter Edition: 3
Enter Price: 1500
Press Y/y to continue: y

Press 1 to add Book
Press 2 to delete Books having price more than 2000
Press 3 to display list
Enter choice:1
Enter Title: Introduction to Algorithms
Enter Authors: Thomas H. Cormen, Charles E. Leiserson, Ronald Rivest, Clifford Stein
Enter Edition: 3
Enter Price: 1000
Press Y/y to continue: y

Press 1 to add Book
Press 2 to delete Books having price more than 2000
Press 3 to display list
Enter choice:1

Enter Title: Programming Pearls
Enter Authors: Jon Bentley
Enter Edition: 2
Enter Price: 2000
Press Y/y to continue: y

Press 1 to add Book
Press 2 to delete Books having price more than 2000
Press 3 to display list
Enter choice:1
Enter Title: Algorithms + Data Structures = Programs
Enter Authors: Niklaus Wirth
Enter Edition: 2
Enter Price: 2500
Press Y/y to continue: y

Press 1 to add Book
Press 2 to delete Books having price more than 2000
Press 3 to display list
Enter choice:2
Algorithms + Data Structures = Programs has been deleted
Press Y/y to continue: y
n

8.Evaluation Task (Unseen) **[Expected time = 60mins for two tasks]**

The lab instructor will give you unseen task depending upon the progress of the class.

9. Evaluation criteria

The evaluation criteria for this lab will be based on the completion of the following tasks. Each task is assigned the marks percentage which will be evaluated by the instructor in the lab whether the student has finished the complete/partial task(s).

Table 2: Evaluation of the Lab

Sr. No.	Task No	Description	Marks
1	4	Problem Modeling	20
2	6	Procedures and Tools	10
3	7,8	Practice tasks and Testing	35
4	8.1	Evaluation Tasks (Unseen)	20
5		Comments	5
6		Good Programming Practices	10

10. Further Readings

10.1 Web sites related to C++ tutorials about linked lists

1. http://www.cprogramming.com/tutorial/lesson15.html
2. http://www.element14.com/community/community/code_exchange/blog/2013/03/26/c-tutorial--linked-list
3. http://www.dreamincode.net/forums/topic/31357-c-linked-lists-custom-linked-lists-part-1/

10.2 Web sites containing supporting material

2. https://www.cpp.edu/~ftang/courses/CS240/lectures/slist.htm
3. http://www.baumann.info/public/cpp_lernaufgabe_linked_list.pdf