Linked Lists

Introduction

Linked Lists are a fundamental data structure in computer science that differ from arrays in key ways:

Arrays: Each item occupies a particular position and can be directly accessed using an index number.

Linked Lists: Need to follow along the chain of elements to find a particular element. A data item cannot be accessed directly.

Linked lists are probably the second most commonly used general purpose storage structures after arrays.

Core Concepts

- In a linked list, each data item is embedded in a link
- There are many similar links
- Each link object contains a reference to the next link in the list
- In a typical application, there would be many more data items in a link

Real World Applications

Image Viewer Previous and next images are linked, hence can be accessed by next and previous buttons.

Web Browser Navigation We can access previous and next URLs searched in web browsers by pressing back and next buttons since they are linked as a linked list.

Music Player Songs in music players are linked to previous and next songs. You can play songs either from the start or end of the list.

Computer Science Applications

- Implementation of stacks and queues
- Implementation of graphs: Adjacency list representation of graphs is most popular which uses linked lists to store adjacent vertices
- Dynamic memory allocation: We use linked lists of free blocks
- Maintaining directory of names

Main Operations

Find Find a link with a specified key value.

Insert Insert links anywhere in the list.

Delete Delete a link with the specified value.

Implementation

Link Class

```
class Link {
    public int iData;    // data item
    public Link next;    // reference to the next link

    public Link(int id) { // constructor
        iData = id;
        next = null;
    }

    public void displayLink() {  // display data item
        System.out.println(iData);
    }
}
```

The Link class is the fundamental building block of a linked list. Each Link object contains:

- A data item (iData)
- A reference to the next link in the list
- Methods for creating and displaying the link

LinkList Class

The LinkList class contains only one data item, a reference to the first link on the list called 'first'.

```
class LinkList {
    private Link first;

    public LinkList() {      //constructor
        first = null;
    }

    public boolean isEmpty() {  // true if list is empty
        return (first == null);
    }

    public void displayList() {
        Link current = first;
        while (current != null) {
            current.displayLink();
            current = current.next;
        }
        System.out.println(" ");
    }
}
```

The LinkList class provides:

- A constructor that initializes an empty list
- A method to check if the list is empty
- A method to display all elements in the list by traversing from the first element to the last

Important Methods

Insert at the Beginning

```
public void insertFirst(int id) {
    Link newLink = new Link(id);
    newLink.next = first;
    first = newLink;
}
```

This method inserts a new link at the beginning of the list by:

1. Creating a new link with the given data
2. Setting its next reference to the current first element
3. Making the new link the first element

Delete from the Beginning

```
public Link deleteFirst() {
    Link temp = first;
    first = first.next;
    return temp;
}
```

This method removes the first link from the list by:

1. Storing a reference to the first link
2. Setting the first reference to the second link
3. Returning the removed link

Example Program

Program to create a new linked list, insert four links, display the list, and remove items one by one:

```
class myList {
    public static void main(String[] args) {
        LinkList theList = new LinkList();    // create a new list

        theList.insertFirst(23);    // insert four items
        theList.insertFirst(89);
        theList.insertFirst(12);
        theList.insertFirst(55);

        theList.displayList();           //display the list

        while(!theList.isEmpty()) {   // delete item one by one
            Link aLink = theList.deleteFirst();
            System.out.print("Deleted ");
            aLink.displayLink();
        }
    }
}
```

When executed, this program will:

1. Create an empty linked list
2. Insert 4 values (55, 12, 89, 23) in that order
3. Display all values in the list (from first to last)
4. Delete each item one by one until the list is empty, showing each deleted value

Note that when displaying the list, the values will appear in reverse order of insertion: 55, 12, 89, 23.

Advanced Types of Linked Lists

Double-Ended List

A double-ended list is similar to an ordinary linked list with an additional reference to the last link. This allows operations at both ends of the list to be performed efficiently.

Doubly Linked List

A doubly linked list allows traversal backwards as well as forward through the list. Each link has two references:

- One to the next link (as in a standard linked list)
- One to the previous link

This makes operations like deletion more efficient but increases memory usage and complexity.