Δομές Δεδομένων & Ανάλυση Αλγορίθμων
3ο Εξάμηνο

Συνδεδεμένες Λίστες
(Linked List)

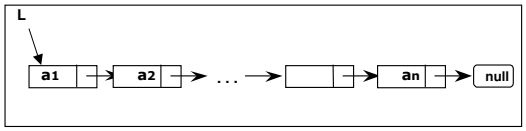
Δημοσθένης Σταμάτης
http:/www.iee.ihu.gr/~demos
Τμήμα Μηχανικών Πληροφορικής & Ηλεκτρονικών Συστημάτων

Συνδεδεμένη Λίστα (Linked List)

☞ Λίστα (List) είναι ένα διατεταγμένο σύνολο από 0 ή περισσότερα στοιχεία τα οποία, κατά κανόνα, είναι όλα του ίδιου τύπου:

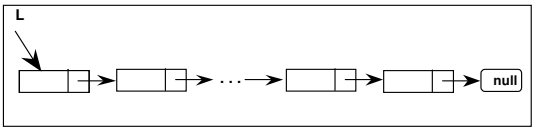
$a_1, a_2, \ldots, a_n$

☞ Σε μία Συνδεδεμένη Λίστα (Linked List) τα παραπάνω στοιχεία είναι τοποθετημένα σε κόμβους οι οποίοι είναι συνδεδεμένοι μεταξύ τους όπως φαίνεται στο παρακάτω σχήμα:

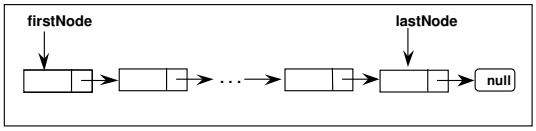


☞ Η συνδεδεμένη λίστα είναι δυναμική δομή δεδομένων. Ο αριθμός των κόμβων της μεταβάλλεται στη διάρκεια της εκτέλεσης του προγράμματος, καθώς κόμβοι μπορούν να εισάγονται και να διαγράφονται

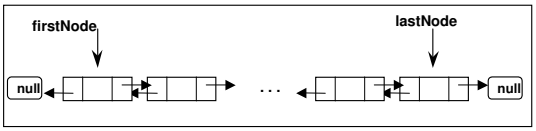
Τρόποι Αναπαράστασης Συνδεδεμένης Λίστας



Απλά Συνδεδεμένη λίστα με πρόσβαση στην αρχή



Απλά Συνδεδεμένη λίστα με πρόσβαση στην αρχή και στο τέλος



Διπλά Συνδεδεμένη λίστα με πρόσβαση στην αρχή και στο τέλος

Βασικές λειτουργίες Συνδεδεμένης Λίστας

Οι πιο βασικές λειτουργίες που απαιτούνται για το χειρισμό μιας συνδεδεμένης λίστας ανήκουν σε μία από τις παρακάτω τρεις κατηγορίες:

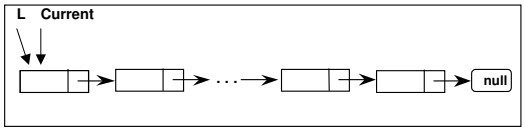
- Επίσκεψη κάποιου κόμβου της συνδεδεμένης λίστας
- Προσθήκη ενός κόμβου στη συνδεδεμένη λίστα
- Διαγραφή ενός κόμβου από τη συνδεδεμένη λίστα

Οποιαδήποτε άλλη λειτουργία μπορεί να οριστεί σαν συνδυασμός των παραπάνω τριών:

Επίσκεψη Κόμβου σε Συνδεδεμένη Λίστα

☞ ΒΗΜΑ 1: Τοποθετούμε ένα βοηθητικό δείκτη ο οποίος δείχνει στην αρχή της λίστας:

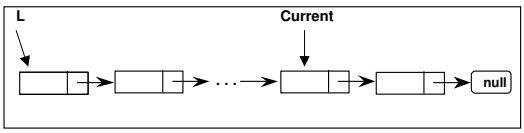
Current = L;



☞ ΒΗΜΑ 2: Στη συνέχεια μετακινούμε τον δείκτη Current στον επόμενο κόμβο, όσες φορές απαιτείται για να τον τοποθετήσουμε στην επιθυμητή θέση:

Current = Current.next; ή ισοδύναμα

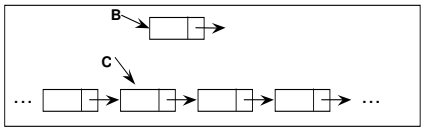
Current = Current.getNext( );



Εισαγωγή Κόμβου σε Συνδεδεμένη Λίστα

☞ ΒΗΜΑ 1: Δημιουργούμε τον καινούργιο κόμβο:

B = new Node(T, null)

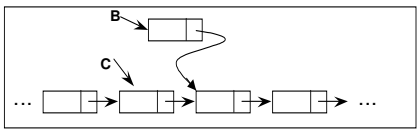


☞ ΒΗΜΑ 2: Τον συνδέουμε με τον επόμενο C:

B.next = C. next;

ή ισοδύναμα

B.setNext(C.getNext( ));

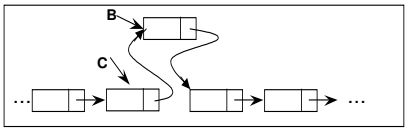


☞ ΒΗΜΑ 3: Τέλος κάνουμε επόμενο του κόμβου C τον κόμβο B

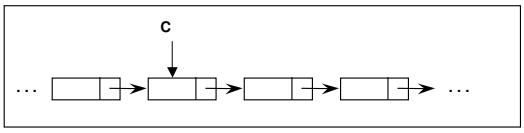
C.next = B;

ή ισοδύναμα

C.setNext(B);



Διαγραφή Κόμβου από Συνδεδεμένη Λίστα

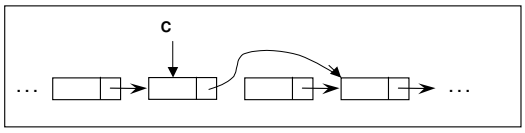


Πριν τη διαγραφή του κόμβου που βρίσκεται μετά τον C

☞ Διαγράφουμε τον επόμενο κόμβο του C παρακάμπτοντάς τον ως εξής:

`C.next = C.next.next;` ή ισοδύναμα

`C.setNext(C.getNext( ).getNext( ))`



Μετά τη διαγραφή του κόμβου


```
class Node
//Ορισμός κόμβου μιας απλά συνδεδεμένης λίστας (linked list)
{       Object item;
        Node next;

        public Node( ) {
                this(null,null); }
        public Node(Object it, Node n) {
                item = it;
                next = n; }
        public void setItem(Object newItem) {
                item = newItem;  }
        public void setNext(Node newNext) {
                next = newNext;  }
        public Object getItem( ) {
                return(item);  }
        public Node getNext( ) {
                return(next);  }
}
```

```
public class LinkedList
 //Υλοποίηση μιάς απλά συνδεδεμένης λίστας (linked list)
{
        private Node firstNode, lastNode;

        public LinkedList( ) {
                firstNode = lastNode = null;
        }

        public Node getFirst( ) {
          return firstNode;
        }

        public Node getLast( ) {
          return lastNode;
        }

        public boolean isEmpty( ) {
                return(firstNode == null);
        }
```



```
//Υλοποίηση μιάς απλά συνδεδεμένης λίστας (linked list) (συνέχεια)

public void insertFirst(Object newItem) {
        if (isEmpty( ))
                firstNode = lastNode = new Node(newItem, null);
        else
                firstNode = new Node(newItem, firstNode);
}

public void insertLast(Object newItem) {
        if (isEmpty( ))
                firstNode = lastNode = new Node(newItem, null);
        else
                lastNode = lastNode.next = new Node(newItem, null);
        }
```

//Υλοποίηση μιάς απλά συνδεδεμένης λίστας (linked list) (συνέχεια)

```
public Object removeFirst( ) throws ListEmptyException
{
        if (isEmpty( ))
                throw new ListEmptyException("Empty List!!!");
        Object removeItem = firstNode.item;
        if (firstNode == lastNode ) /* Αν χρησιμοποιήσουμε equals ??? */
                firstNode = lastNode = null;
        else
                firstNode = firstNode.next;
        return removeItem;
}
```


//Υλοποίηση μιάς απλά συνδεδεμένης λίστας (linked list) (συνέχεια)

```
public Object removeLast( ) throws ListEmptyException
{
        if (isEmpty( ))
                throw new ListEmptyException("Empty List!!!");
        Object removeItem = lastNode.item;
        if (firstNode == lastNode)
                firstNode = lastNode = null;
        else
        {       Node current = firstNode;
                while (current.next != lastNode)
                        current = current.next;
                lastNode = current;
                current.next = null;
                }
        return removeItem;
}
```

```
//Υλοποίηση μιάς απλά συνδεδεμένης λίστας (linked list) (συνέχεια)

public void printList( )
{
        if (isEmpty( ))
                System.out.println("Empty List");
        else
        {       Node current = firstNode;
                while (current != null)
                {       System.out.print(current.item.toString( ) + " ");
                        current = current.next;
                }
                System.out.println("\n");
        }
}
```


ΤΑΞΙΝΟΜΗΣΗ ΣΥΝΔΕΔΕΜΕΝΗΣ ΛΙΣΤΑΣ

```
public /*LinkedList*/ SortList( ) {
   Node trace, current, min;
   trace = getFirst( );
   while (trace!=null)
          {  current = trace;
             min = trace;
             while (current!=null)
               {     if ((current.getItem( )).compareTo(min.getItem( ))<0)
                           min=current;
                     current = current.getNext();
               } //endwhile current
             String temp = trace.getItem();
             trace.setItem(min.getItem());
             min.setItem(temp);
             trace = trace.getNext();
           }//endwhile trace
   /*return this*/;
```

ΤΑΞΙΝΟΜΗΣΗ ΣΥΝΔΕΔΕΜΕΝΗΣ ΛΙΣΤΑΣ

```
public class ListToSort extends LinkedList
{
public ListToSort() { super(); }

public LinkedList SortList() {
        …
        …
        …
return this;
        }
}
```


ΤΑΞΙΝΟΜΗΣΗ ΣΥΝΔΕΔΕΜΕΝΗΣ ΛΙΣΤΑΣ

```
public class ListSort extends LinkedList
{
   public ListSort() {super();}

   public LinkedList SortList()
   {
       Node trace, current, min;

       trace = getFirst();
       while (trace!=null)
          { current = trace;
            min = trace;
            while (current!=null)
             {  if ((current.getItem()).compareTo(min.getItem())<0)
                    min=current;
                current = current.getNext();
             }
          String temp = trace.getItem();
          trace.setItem(min.getItem());
          min.setItem(temp);
          trace = trace.getNext();
       }//endwhile of trace

       return this;
   }

}
```