Κωνσταντίνος Σκορδούλης
ΑΜ:1115 2016 00155

Στο ερώτημα αυτό, ζητήθηκε να υλοποιηθούν δύο συναρτήσεις: **Delete , Insert**

Χρειάστηκε και μια βοηθητική συνάρτηση : **Length** , η οποία υπολογίζει το μήκος της συνδεδεμένης λίστας.

Length

Παίρνει σαν όρισμα δείκτη *L που δείχνει στο πρώτο κόμβο μίας linked list,

και εκτελεί ορισμένα βήματα.

1) Ελέγχει αν η λίστα είναι άδεια, επιστρέφοντας **length=0**, **αλλιώς** αρχικοποιεί ένα ακέραιο μετρητή (counter) **count=1** (αφου ο δείκτης *L δεν είναι NULL τότε υπάρχει τουλάχιστον ένας κόμβος).
2) Στη συνέχεια προχωράει ο *L μέχρι να φτάσει και στον τελευταίο κόμβο, αυξάνοντας το μετρητή κατά 1(**count+=1)**.
3) Τέλος, επιστρέφει το **count**.

Delete

Παίρνει σαν όρισμα δείκτη *L και ένα ακέραιο **i** , ο οποίος καθορίζει ποιος κόμβος

θα αφαιρεθεί(απαίτηση **1<=i<=Length(L)**). Αναλυτικότερα:

1) Ορίζουμε 3 δείκτες σε λίστα(NodeType ***Q,*N,*H)**
2) Αρχικοποιούμε **N=L; H=L;** (**H** για **HEAD**).
3) Ελέγχει, καταρχάς, αν το **i** είναι λάθος(**i<1 ή i>Length(L)**).
4) Ελέγχει αν η λίστα είναι άδεια, τερματίζοντας τη **Delete** άμα ισχύει, και διακρίνει **2 περιπτώσεις:**
   - ✓ Η λίστα έχει **1 κόμβο.**
     - ➢ Ελευθερώνουμε τον κόμβο => Λίστα άδεια
   - ✓ Η λίστα έχει **παραπάνω από 1 κόμβους.** Έχουμε **3 υποπεριπτώσεις:**
     - ➢ **i=1** Ελευθερώνουμε τον πρώτο κόμβο, και ο δείκτης *L δείχνει τον επόμενο κόμβο ως **HEAD**

- **i=2** Ελευθερώνουμε τον δεύτερο κόμβο, ενώ παράλληλα ενώνουμε το πρώτο κόμβο με τον τρίτο, διατηρώντας την ιδιότητα της συνδεδεμένης λίστας.
- **i=>3** Με μία **loop for**, μεταβαίνουμε στο (**i-1)node**, δηλαδή στον προηγούμενο από αυτόν που θέλουμε να σβήσουμε. Σβήνουμε τον **i-node**, **ε**νώ παράλληλα ενώνουμε το (**i-1)node** με το (**i+1)node**.Αν θέλουμε να σβήσουμε το τελευταίο κόμβο, απλά θέτουμε το **link** του προηγούμενου σε NULL , και σβήνουμε το τελευταίο.

Insert

Παίρνει σαν όρισμα, δείκτη *L σε λίστα , ένα δείκτη σε **char *X** και ένα ακέραιο **i** , ο οποίος καθορίζει σε ποιον κόμβο θα εισαχθεί το ***X** ή **για i=Length(L)+1** τη δημιουργία ενός νέου κόμβου στο τέλος της λίστας (απαίτηση **1<=i<=Length(L)+1**).

1) Ορίζουμε δύο δείκτες σε λίστα ***N** και ***Q** (Nodetype).
2) Καταρχάς, ελέγχει αν η δοσμένη λίστα είναι κενή ή όχι, τερματίζοντας αν ισχύει.
3) Ελέγχει αν το εισαγόμενο **i** είναι σωστό, (**1<=i<=Length(L)+1**).
4) Στη συνέχεια, διακρίνουμε δύο περιπτώσεις:
   - ✓ Η λίστα έχει **ένα κόμβο:**
     - Αλλάζουμε την τιμή του κόμβου με το ***X**.
     - Δεσμεύουμε χώρο στη μνήμη για τη δημιουργία ενός δεύτερου κόμβου(χρησιμοποιώντας **malloc για την *Q**), συνδέουμε τον πρώτο κόμβο με τον καινούργιο και εισάγουμε ***X** στη **Q->Airport**.
   - ✓ Η λίστα έχει **παραπάνω από ένα κόμβο**, και διακρίνουμε **3** υποπεριπτώσεις:
     - **i=1** Απλά αλλάζουμε το περιεχόμενο του κόμβου με ***X**.
     - **i=2** Προχωράμε στον επόμενο κόμβο, και αλλάζουμε το **data** του.
     - **i=>3** Με ένα **loop for** φτάνουμε στο **(i-1)node** και τσεκάρουμε αν είναι ο **(i-1)node** είναι ο τελευταίος κόμβος.
       - ❖ Αν ναι, τότε δεσμεύουμε χώρο στη μνήμη για τη δημιουργία ενός καινούργιου κόμβου (χρησιμοποιώντας **malloc για την *Q**), συνδέουμε τον «τελευταίο» κόμβο με τον καινούργιο και εισάγουμε ***X** στη **Q->Airport**.
       - ❖ Αν όχι, τότε προχωράμε στον επόμενο κόμβο και αλλάζουμε την τιμή του **Q->Airport** με ***X**.

Υ.Γ Βελτίωσα και το **information hiding**, έτσι ώστε το **Interface.h** να περιέχει μόνο τα **typedefs** και τα **πρότυπα συναρτήσεων.**