



4.2 Ο ΑΤΔ Συνδεδεμένη Λίστα

Μια λίστα είναι μια ακολουθία στοιχείων δεδομένων, πράγμα το οποίο σημαίνει ότι υπάρχει μία διάταξη όσον αφορά τα στοιχεία της λίστας: κάποιο στοιχείο είναι πρώτο, κάποιο δεύτερο, κ.ο.κ. Γι' αυτό, οποιαδήποτε υλοποίηση αυτής της δομής δεδομένων πρέπει να έχει ενσωματωμένη μια μέθοδο που να καθορίζει αυτήν την διάταξη. Στην υλοποίηση με σειριακή αποθήκευση, η διάταξη των στοιχείων της λίστας δινόταν σιωπηρά από την φυσική διάταξη των στοιχείων του πίνακα, αφού το πρώτο στοιχείο ήταν αποθηκευμένο στην πρώτη θέση του πίνακα, το δεύτερο στοιχείο στην δεύτερη θέση, κ.ο.κ. Αυτός ο σιωπηρός προσδιορισμός της διάταξης των στοιχείων της λίστας ήταν που έκανε απαραίτητη την μετατόπισή τους στον πίνακα σε κάθε εισαγωγή ή διαγραφή στοιχείου. Στην ενότητα αυτή, θεωρούμε μια εναλλακτική υλοποίηση λίστας, στην οποία η διάταξη των στοιχείων δίνεται ρητά.

Σε οποιαδήποτε δομή, που χρησιμοποιείται για την αποθήκευση στοιχείων λίστας, πρέπει να υπάρχει η δυνατότητα εκτέλεσης τουλάχιστον των παρακάτω λειτουργιών, αν πρέπει να διατηρηθεί η διάταξη των στοιχείων της λίστας:

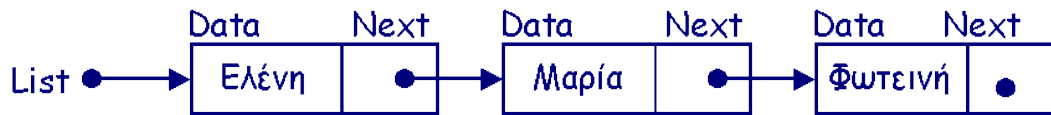
- Εντοπισμός του πρώτου στοιχείου
- Δοσμένης της θέσης οποιουδήποτε στοιχείου, προσδιορισμός της θέσης του στοιχείου που ακολουθεί

Όπως είπαμε, στην υλοποίηση με σειριακή αποθήκευση η διάταξη δίνεται σιωπηρά από τους δείκτες του πίνακα, δηλαδή το πρώτο στοιχείο της λίστας αποθηκεύεται στην θέση 1 του πίνακα, το δεύτερο στοιχείο στην θέση 2, κλπ, και το επόμενο του στοιχείου της θέσης i βρίσκεται στην θέση $i+1$ του πίνακα.

Μια δομή αποθήκευσης στοιχείων λίστας στην οποία η διάταξη δίνεται ρητά ονομάζεται **συνδεδεμένη λίστα (linked lists)**. Μια συνδεδεμένη λίστα είναι μια συλλογή στοιχείων, που ονομάζονται **κόμβοι (nodes)**, και σε κάθε κόμβο αποθηκεύονται δύο είδη πληροφορίας:

- Ένα στοιχείο της λίστας
- Ένας **δεσμός (link)** ή **δείκτης (pointer)** που δείχνει ρητά τη θέση του κόμβου που περιέχει το επόμενο στοιχείο της λίστας

Επίσης, πρέπει να διατηρείται πρόσβαση στον κόμβο όπου είναι αποθηκευμένο το πρώτο στοιχείο της λίστας. Έστω, για παράδειγμα, ότι έχουμε μια συνδεδεμένη λίστα όπου αποθηκεύουμε τα ονόματα *Ελένη*, *Μαρία* και *Φωτεινή*, όπως φαίνεται στο παρακάτω σχήμα.



Σχήμα 4.2.1

Στο σχήμα αυτό, τα βέλη παριστάνουν τους δεσμούς και ο δείκτης List δείχνει στον πρώτο κόμβο της λίστας. Στο τμήμα δεδομένων (Data) του κάθε κόμβου αποθηκεύεται ένα από τα ονόματα της λίστας και η κουκίδα στον τελευταίο κόμβο που δεν έχει βέλος να ξεκινάει από αυτήν αντιπροσωπεύει έναν **μηδενικό δείκτη (nil pointer)** και δηλώνει ότι δεν υπάρχει επόμενο γι' αυτό το στοιχείο. Αν με p συμβολίσουμε το δείκτη προς κάποιο κόμβο αυτής της λίστας, τότε θα δηλώνουμε το τμήμα δεδομένου του κόμβου ως $Data(p)$ και το τμήμα δεσμού ως $Next(p)$. Επίσης, θα θεωρούμε ότι NilValue είναι μια ειδική τιμή που μπορεί να δοθεί στο p για να δείξει ότι είναι μηδενικός δείκτης, δηλαδή δε δείχνει σε κανένα κόμβο.

Στη συνέχεια θα δούμε πώς μπορούν να υλοποιηθούν οι πέντε βασικές λειτουργίες της λίστας.

■ **Δημιουργία κενής λίστας - CreateList:** Για να δημιουργήσουμε μια κενή λίστα αρκεί να θέσουμε την τιμή NilValue στον δείκτη List για να δείξουμε ότι δεν δείχνει σε κανένα κόμβο, δηλαδή:

List •

■ **Έλεγχος κενής λίστας - EmptyList:** Για να καθορίσουμε αν η λίστα είναι κενή μπορούμε απλά να εξετάσουμε αν ο δείκτης List έχει την τιμή NilValue.

■ **Διάσχιση λίστας - Traverse:** Έστω ότι θέλουμε να διασχίσουμε μια λίστα όπως η παραπάνω λίστα ονομάτων. Χρησιμοποιούμε έναν βοηθητικό δείκτη CurrP, ο οποίος δείχνει αρχικά στον πρώτο κόμβο, και επεξεργαζόμαστε το στοιχείο *Ελένη* που είναι αποθηκευμένο στον κόμβο αυτό. Για να μετακινηθούμε στον επόμενο κόμβο και να επεξεργαστούμε το στοιχείο *Μαρία*, ακολουθούμε τον δεσμό του τρέχοντος κόμβου ($Next(CurrP)$) θέτοντας $CurrP = Next(CurrP)$. Ομοίως, για να επεξεργαστούμε το επόμενο στοιχείο, *Φωτεινή*, θέτουμε πάλι $CurrP = Next(CurrP)$, οπότε ο CurrP δείχνει τώρα στον τρίτο κόμβο της λίστας. Το στοιχείο *Φωτεινή* είναι το τελευταίο της λίστας, οπότε αν θέσουμε πάλι $CurrP = Next(CurrP)$, ο CurrP γίνεται μηδενικός και καταλαβαίνουμε ότι φτάσαμε στο τέλος της λίστας. Ο παρακάτω αλγόριθμος περιγράφει τη λειτουργία της διάσχισης:

Διάσχιση λίστας - TRAVERSE

*/*Αλγόριθμος διάσχισης μιας συνδεδεμένης λίστας.*

Δέχεται: Μια συνδεδεμένη λίστα με τον δείκτη *List* να δείχνει στον πρώτο κόμβο.

Λειτουργία: Διασχίζει αυτή την συνδεδεμένη λίστα με τη βοήθεια του δείκτη *CurrP* που δείχνει τον τρέχοντα κάθε φορά κόμβο της συνδεδεμένης λίστας και επεξεργάζεται κάθε στοιχείο της συνδεδεμένης λίστας ακριβώς μια φορά.

Επιστρέφει: Εξαρτάται από το είδος της επεξεργασίας.**/*


1. *CurrP* ← *List* */*Αρχικοποίηση του CurrP, ώστε να δείχνει στον πρώτο κόμβο της συνδεδεμένης λίστας*/*

2. **Όσο** *CurrP* <> *NilValue* **επανάλαβε** */*Όσο δεν διέσχισες όλη τη λίστα, δηλαδή όσο ο δείκτης CurrP δείχνει σε κάποιο κόμβο της συνδεδεμένης λίστας*/*

α. Πάρε το τρέχον στοιχείο *Data(CurrP)* για επεξεργασία

β. *CurrP* ← *Next(CurrP)* */*Θέσε το δείκτη CurrP ώστε να δείχνει στον επόμενο κόμβο της λίστας*/*

Τέλος_επανάληψης

 **Εισαγωγή στοιχείου - Insert:** Για να εισαγάγουμε ένα νέο στοιχείο στη λίστα, χρειάζεται πρώτα να αποκτήσουμε ένα νέο κόμβο, ώστε να αποθηκεύσουμε την τιμή του στοιχείου στο τμήμα δεδομένου του κόμβου αυτού. Υποθέτουμε ότι υπάρχει μια *δεξαμενή* (*storage pool*) διαθέσιμων κόμβων καθώς και ένας μηχανισμός απόκτησης τέτοιων κόμβων από τη δεξαμενή. Υποθέτουμε, δηλαδή, ότι υπάρχει μια διαδικασία *GetNode* η οποία, αν κληθεί με μια πρόταση της μορφής *GetNode(TempP)*, θα επιστρέψει έναν προσωρινό δείκτη *TempP* προς έναν διαθέσιμο κόμβο. Το επόμενο βήμα είναι να συνδεθεί αυτός ο κόμβος με την υπάρχουσα λίστα. Διακρίνουμε δύο περιπτώσεις:

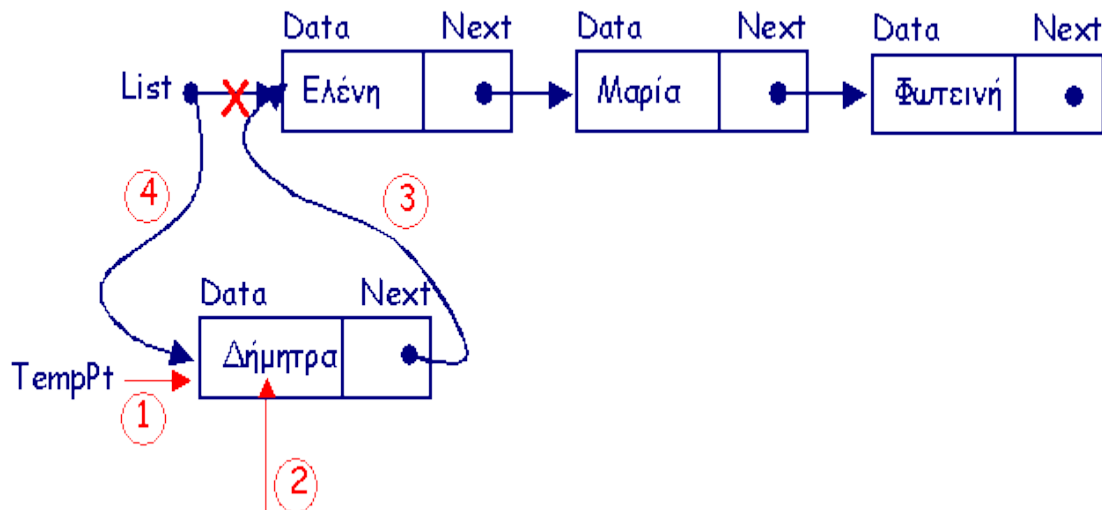
α) η εισαγωγή να γίνει στην αρχή της λίστας: έστω ότι θέλουμε να εισαγάγουμε το όνομα *Δήμητρα* στην αρχή της λίστας. Κατ' αρχήν (1) παίρνουμε έναν νέο κόμβο με την εντολή *GetNode(TempP)*, στον οποίο δείχνει προσωρινά ο δείκτης *TempP*, και (2) αποθηκεύουμε την τιμή *Δήμητρα* στο τμήμα δεδομένου του θέτοντας *Data(TempP)='Δήμητρα'*. Στη συνέχεια, (3) εισάγουμε τον κόμβο αυτό στην αρχή της λίστας κάνοντας το τμήμα δεσμού του να δείχνει στον πρώτο κόμβο της λίστας, δηλαδή θέτουμε *Next(TempP)=List*, και, τέλος, (4) θέτουμε *List=TempP* ώστε ο *List* να δείχνει στον νέο κόμβο. Συνολικά, δηλαδή, εκτελούμε τις ακόλουθες εντολές:

(1) *GetNode(TempP)*

(2) $\text{Data}(\text{TempP}) = \text{'Δήμητρα'}$

(3) $\text{Next}(\text{TempP}) = \text{List}$

(4) $\text{List} = \text{TempP}$



Σχήμα 4.2.2. Εισαγωγή στοιχείου στην αρχή της λίστας.

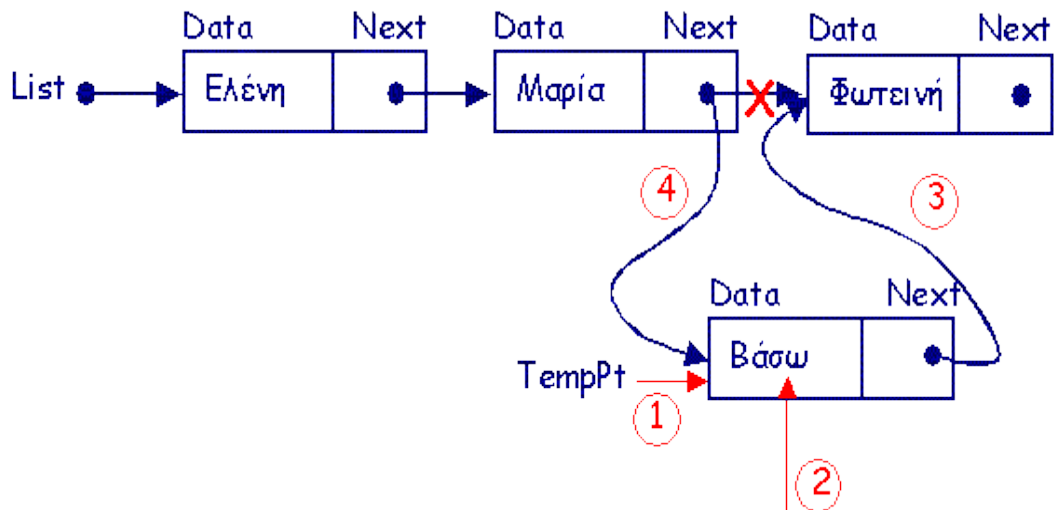
β) η εισαγωγή να γίνει μετά από κάποιο στοιχείο της λίστας: έστω ότι θέλουμε να εισαγάγουμε το όνομα *Βάσω* μετά από τον κόμβο που περιέχει το όνομα *Μαρία* και ότι ο δείκτης *PredP* δείχνει σ' αυτόν τον κόμβο. (1) Παίρνουμε πάλι έναν νέο κόμβο με την εντολή $\text{GetNode}(\text{TempP})$, στον οποίο δείχνει προσωρινά ο δείκτης *TempP*, και (2) αποθηκεύουμε την τιμή *Στέλλα* στο τμήμα δεδομένου του θέτοντας $\text{Data}(\text{TempP}) = \text{'Βάσω'}$. Εισάγουμε τον κόμβο αυτό στη λίστα και (3) θέτουμε το τμήμα δεσμού του ίσο με $\text{Next}(\text{PredP})$ ώστε να δείχνει στον επόμενο του κόμβου που περιέχει το όνομα *Μαρία* με την εντολή $\text{Next}(\text{TempP}) = \text{Next}(\text{PredP})$ και (4) αλλάζουμε το τμήμα δεσμού του προηγούμενου κόμβου ώστε να δείχνει στον νέο κόμβο, δηλαδή θέτουμε $\text{Next}(\text{PredP}) = \text{TempP}$. Συνολικά, δηλαδή, εκτελούμε τις ακόλουθες εντολές:

(1) $\text{GetNode}(\text{TempP})$

(2) $\text{Data}(\text{TempP}) = \text{'Βάσω'}$

(3) $\text{Next}(\text{TempP}) = \text{Next}(\text{PredP})$

(4) $\text{Next}(\text{PredP}) = \text{TempP}$



Σχήμα 4.2.3. Εισαγωγή στοιχείου μέσα στη λίστα.

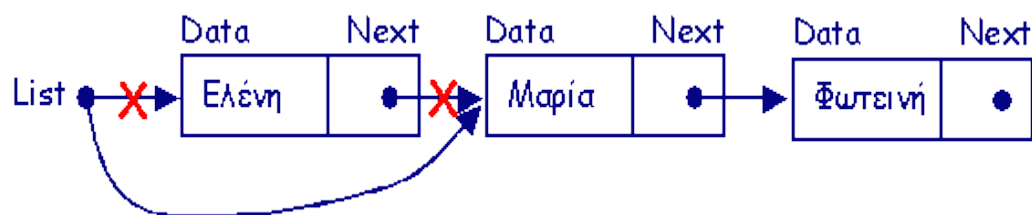
➤ **Διαγραφή στοιχείου - Delete:** Στην διαγραφή ενός στοιχείου από μια λίστα έχουμε πάλι δύο περιπτώσεις:

α) διαγραφή του πρώτου στοιχείου της λίστας: έστω ότι θέλουμε να διαγράψουμε το στοιχείο *Ελένη* που βρίσκεται στην αρχή της λίστας του Σχήματος 4.2.1. Αυτό είναι πολύ εύκολο να γίνει αν αλλάξουμε τον δείκτη *List* ώστε να δείχνει στον δεύτερο κόμβο της λίστας, θέτοντας $TempP = List$ και μετά $List = Next(List)$, και να επιστρέψουμε τον διαγραμμένο κόμβο στην δεξαμενή με τους διαθέσιμους κόμβους καλώντας μια διαδικασία *ReleaseNode*, δηλαδή $ReleaseNode(TempP)$. Συνολικά, δηλαδή, εκτελούμε τις ακόλουθες εντολές:

$TempP = List$

$List = Next(List)$

$ReleaseNode(TempP)$



Σχήμα 4.2.4. Διαγραφή του 1ου στοιχείου της λίστας.

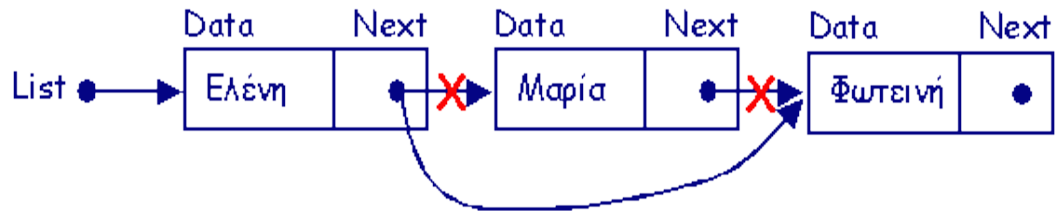
β) διαγραφή ενός στοιχείου από κόμβο για τον οποίο υπάρχει κάποιος προηγούμενος κόμβος: έστω για παράδειγμα ότι θέλουμε να διαγράψουμε τον κόμβο με το όνομα *Μαρία* της λίστας του Σχήματος 4.2.1. Το μόνο που χρειάζεται να κάνουμε είναι να δείχνει ο προηγούμενος κόμβος αυτού που θέλουμε να διαγραφεί στον επόμενο του, δηλαδή $TempP = Next(PredP)$ και μετά $Next(PredP) = Next(TempP)$, και, τέλος, να καλέσουμε την

ReleaseNode για να επιστρέψουμε τον διαγραμμένο κόμβο στη δεξαμενή των διαθέσιμων κόμβων, ReleaseNode(TempP). Συνολικά, δηλαδή, εκτελούμε τις ακόλουθες εντολές:

TempP=Next(PredP)

Next(PredP)=Next(TempP)

ReleaseNode(TempP)



Σχήμα 4.2.5. Διαγραφή στοιχείου που βρίσκεται μέσα στη λίστα.

Όπως φαίνεται από τα παραπάνω, το πλεονέκτημα των συνδεδεμένων λιστών είναι ότι οι εισαγωγές και διαγραφές στοιχείων μπορούν να γίνουν σε οποιαδήποτε θέση της λίστας ανεξάρτητα από το πλήθος των στοιχείων της και χωρίς μετατοπίσεις στοιχείων.