ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ

ΛΙΣΤΕΣ
4.5 Υλοποίηση ΑΤΔ Συνδεδεμένη Λίστα με δείκτες

Υλοποίηση συνδεδεμένης λίστας με δείκτες

Στην υλοποίηση των συνδεδεμένων λιστών με πίνακα χρησιμοποιήσαμε εγγραφές ως τη βασική αποθηκευτική δομή για τους κόμβους.

Έτσι και εδώ θα χρησιμοποιήσουμε πάλι **εγγραφές** με πεδία Data και Next.

Ο τύπος δεδομένων του πεδίου Data θα είναι αυτός που είναι κατάλληλος για την αποθήκευση των δεδομένων και στο πεδίο Next θα αποθηκεύεται ένας δεσμός που θα δείχνει στο επόμενο στοιχείο της λίστας.

Τώρα, όμως, ο δεσμός αυτός θα είναι ένας δείκτης και όχι ένας αριθμοδείκτης πίνακα.

Υλοποίηση συνδεδεμένης λίστας με δείκτες

Μια τέτοια υλοποίηση συνδεδεμένης λίστας φαίνεται παρακάτω:

```
typedef int ListElementType;          (*ο τύπος των στοιχείων της λίστας*)
typedef struct ListNode *ListPointer;
typedef struct ListNode
{
        ListElementType Data;
        ListPointer Next;
} ListNode;
```

Αξίζει να σημειωθεί ότι ο ορισμός του αναγνωριστικού ListPointer προηγείται του ορισμού τού τύπου ListNode.

.

Δημιουργία & έλεγχος κενής συνδεδεμένης λίστας

Το **πλεονέκτημα** αυτής της υλοποίησης είναι ότι δεν χρειαζόμαστε μια δεξαμενή διαθέσιμων κόμβων, αφού μπορούμε να χρησιμοποιήσουμε τις συναρτήσεις malloc και free, για απόκτηση και απελευθέρωση μνήμης αντίστοιχα, αντί για τις GetNode και ReleaseNode.

Για τη δημιουργία μιας κενής συνδεδεμένης λίστας, αναθέτουμε απλά την τιμή NULL σε μια μεταβλητή List τύπου ListPointer,

List = NULL;

η οποία διατηρεί την πρόσβαση στον πρώτο κόμβο της συνδεδεμένης λίστας.

Για να ελέγξουμε αν μια συνδεδεμένη λίστα είναι κενή, μπορούμε απλά να εξετάσουμε αν η List έχει τιμή NULL:

List == NULL;

Δημιουργία & έλεγχος κενής συνδεδεμένης λίστας

Για τη διάσχιση μιας συνδεδεμένης λίστας υλοποιημένης με δείκτες, αρχικοποιούμε έναν δείκτη CurrPtr να δείχνει στον πρώτο κόμβο της συνδεδεμένης λίστας και στη συνέχεια παίρνει τις τιμές των δεσμών των κόμβων και διατρέχει τη λίστα, όπως φαίνεται στον παρακάτω αλγόριθμο και στην υλοποίησή του:

TRAVERSE

CurrPtr ← *List*

Όσο *CurrPtr* <> **nil επανάλαβε**

(*Εδώ παρεμβάλλονται οι απαραίτητες εντολές για την επεξεργασία του *CurrPtr^.Data**)

CurrPtr ← Next(*CurrPtr*)

Τέλος_επανάληψης

TRAVERSE

```
CurrPtr = List;

while (CurrPtr != NULL)
{
        (*Εδώ παρεμβάλλονται οι
        απαραίτητες εντολές για την
        επεξεργασία του CurrPtr->Data*)

        CurrPtr = CurrPtr->Next;

}
```