4.5 Υλοποίηση ΑΤΔ Συνδεδεμένη Λίστα 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 της C και όχι ένας δείκτης πίνακα. Μια τέτοια υλοποίηση συνδεδεμένης λίστας φαίνεται παρακάτω:

```
typedef int ListElementType;      /*ο τύπος των στοιχείων της λίστας ενδεικτικά
                                  τύπου int*/
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.

Το πλεονέκτημα αυτής της υλοποίησης είναι ότι δεν χρειαζόμαστε μια δεξαμενή διαθέσιμων κόμβων, αφού μπορούμε να χρησιμοποιήσουμε τις διαδικασίες 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:

EmptyList = (List==NULL);

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* /*αρχικοποίηση του *CurrPtr*, ώστε να δείχνει στον πρώτο κόμβο της συνδεδεμένης λίστας *List**/

Όσο *CurrPtr* != **NULL επανάλαβε**

/*όσο ο δείκτης *CurrPtr* έχει τιμή διάφορη του NULL, δηλαδή όσο δείχνει σε κάποιο κόμβο της συνδεδεμένης λίστας και δεν την έχουμε διασχίσει όλη*/

/*Εδώ παρεμβάλλονται οι απαραίτητες εντολές για την επεξεργασία των δεδομένων του τρέχοντος κόμβου*/

CurrPtr ← Next(*CurrPtr*) /*θέσε το δείκτη *CurrPtr* ώστε να δείχνει στον επόμενο κόμβο της λίστας*/

Τέλος_επανάληψης

```
void LinkedTraverse(ListPointer List)
/*Δέχεται:   Μια συνδεδεμένη λίστα με τον List να δείχνει στον πρώτο κόμβο.
 Λειτουργία: Διασχίζει τη συνδεδεμένη λίστα και επεξεργάζεται κάθε δεδομένο ακριβώς μια φορά.
 Επιστρέφει: Εξαρτάται από το είδος της επεξεργασίας.*/
{
   ListPointer CurrPtr;
   if (EmptyList(List))
       printf("EMPTY LIST\n");
   else
   {
    CurrPtr = List;
         while ( CurrPtr!=NULL )
        {
           printf("%d\n", CurrPtr->Data);
             CurrPtr = CurrPtr->Next;
        }
   }
}
```

Στην ενότητα 4.6 περιγράφονται και οι υπόλοιπες λειτουργίες των συνδεδεμένων λιστών με χρήση δεικτών.