11 Δομές δεδομένων

Εισαγωγή στην Επιστήμη των Υπολογιστών © Εκδόσεις Κλειδάριθμος

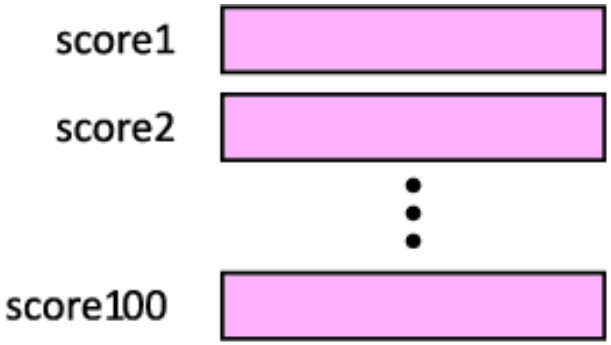
Στόχοι

Μετά την ολοκλήρωση αυτού του κεφαλαίου, ο σπουδαστής θα είναι σε θέση:

- Να ορίζει μια δομή δεδομένων.
- Να ορίζει συστοιχίες ως δομές δεδομένων και να περιγράφει τον τρόπο χρήσης τους για την αποθήκευση λιστών στοιχείων δεδομένων.
- Να ξεχωρίζει το όνομα μιας συστοιχίας από τα ονόματα των στοιχείων της.
- Να περιγράφει τις λειτουργίες που ορίζονται για συστοιχίες.
- Να ορίζει εγγραφές ως δομές δεδομένων και να περιγράφει τον τρόπο χρήσης τους για την αποθήκευση ιδιοτήτων που ανήκουν σε απλά στοιχεία δεδομένων.
- Να ξεχωρίζει το όνομα μιας εγγραφής από τα ονόματα των πεδίων της.
- Να ορίζει μια συνδεδεμένη λίστα ως δομή δεδομένων και να περιγράφει τον τρόπο υλοποίησής της με τη χρήση δεικτών.
- Να κατανοεί τον μηχανισμό μέσω του οποίου προσπελάζονται οι κόμβοι μιας συστοιχίας.
- Να περιγράφει τις λειτουργίες που ορίζονται για τις συνδεδεμένες λίστες.
- Να συγκρίνει και να αντιπαραβάλει συστοιχίες, εγγραφές, και συνδεδεμένες λίστες.
- Να ορίζει τις εφαρμογές συστοιχιών, εγγραφών, και συνδεδεμένων λιστών.

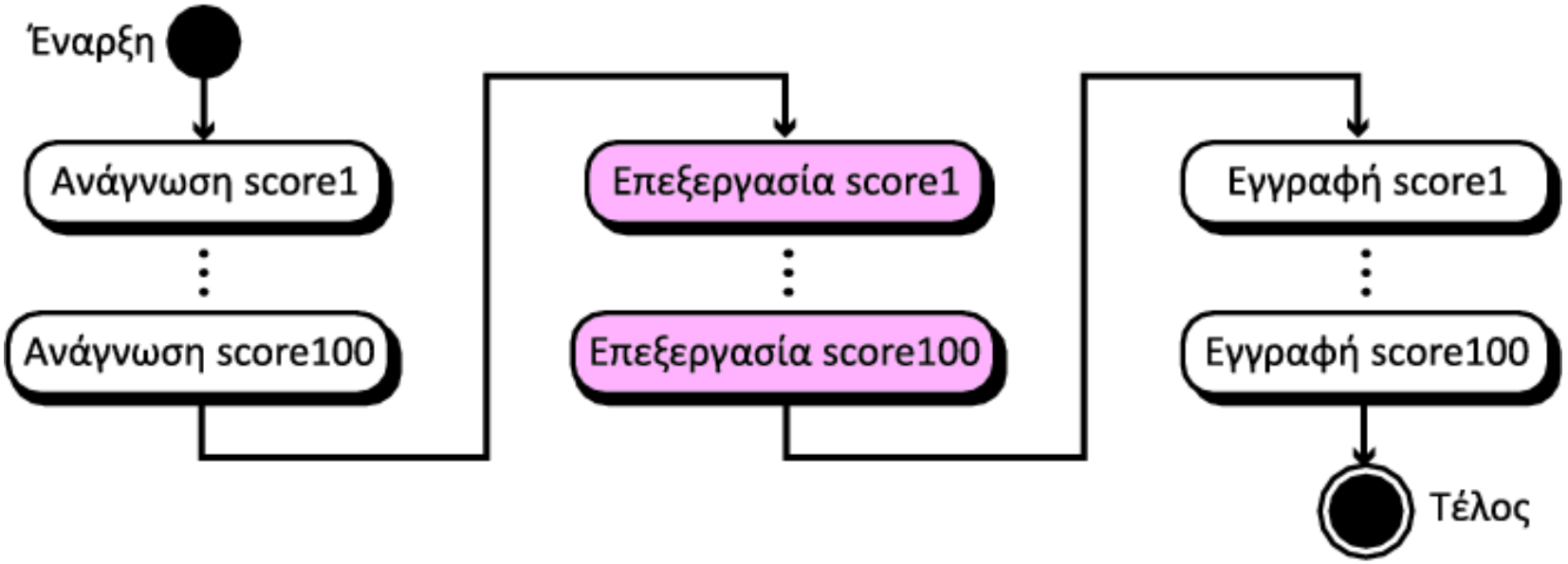
11-1 ΣΥΣΤΟΙΧΙΕΣ

Φανταστείτε ότι έχουμε 100 αριθμούς τους οποίους πρέπει να διαβάσουμε, να επεξεργαστούμε, και να τυπώσουμε. Κατά τη διάρκεια του προγράμματος οι 100 αριθμοί πρέπει να βρίσκονται στη μνήμη. Μία λύση είναι να οριστούν εκατό μεταβλητές, κάθε μία με διαφορετικό όνομα, όπως στην Εικόνα 11.1.



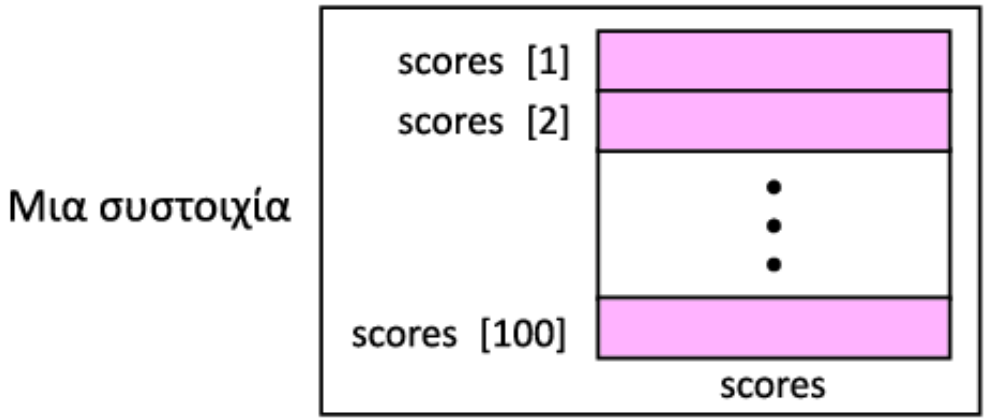
Εικόνα 11.1 **Εκατό ξεχωριστές μεταβλητές**

Όμως η ύπαρξη 100 διαφορετικών ονομάτων δημιουργεί ένα άλλο πρόβλημα. Θα χρειαστούμε 100 αναφορές για την ανάγνωση των μεταβλητών, 100 αναφορές για την επεξεργασία τους, και 100 αναφορές για την εκτύπωσή τους. Το πρόβλημα αυτό παρουσιάζεται σχηματικά στην Εικόνα 11.2.



Εικόνα 11.2 **Επεξεργασία ξεχωριστών μεταβλητών**

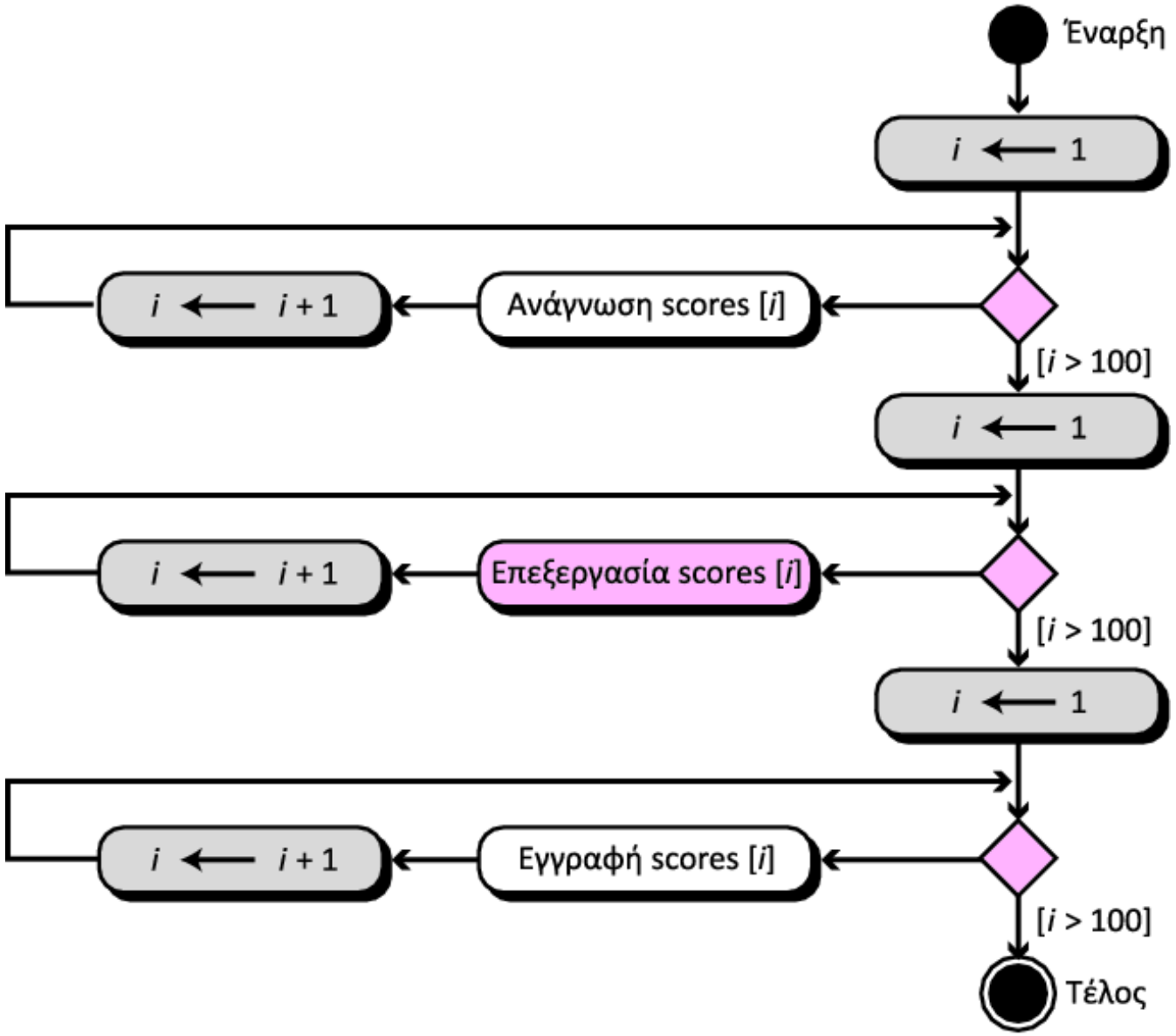
Συστοιχία (array) είναι μια σειριακή δομή στοιχείων, συνήθως ίδιου τύπου δεδομένων, αν και σε ορισμένες γλώσσες προγραμματισμού είναι αποδεκτές και συστοιχίες των οποίων τα στοιχεία είναι διαφορετικού τύπου. Μπορούμε να αναφερόμαστε στα στοιχεία μιας συστοιχίας ως το πρώτο στοιχείο, το δεύτερο στοιχείο, κ.ο.κ. μέχρι να φτάσουμε στο τελευταίο στοιχείο.



Εικόνα 11.3 **Συστοιχίες με αριθμοδείκτες**

Για την ανάγνωση και την εγγραφή στοιχείων σε μια συστοιχία μπορούμε να χρησιμοποιούμε βρόχους. Βρόχους μπορούμε επίσης να χρησιμοποιούμε και για την επεξεργασία των στοιχείων. Τώρα δεν έχει σημασία αν τα στοιχεία προς επεξεργασία είναι 100, 1000, ή 10.000, αφού οι βρόχοι απλοποιούν τον χειρισμό οποιουδήποτε πλήθους. Έτσι, έχουμε τη δυνατότητα να χρησιμοποιήσουμε μια ακέραια μεταβλητή για να ελέγχουμε τον βρόχο και να παραμένουμε σε αυτόν για όσο διάστημα η τιμή της μεταβλητής είναι μικρότερη από το συνολικό πλήθος των στοιχείων της συστοιχίας (Εικόνα 11.4).

Παρόλο που έχουμε χρησιμοποιήσει αριθμοδείκτες οι οποίοι ξεκινούν από το 1, σε ορισμένες σύγχρονες γλώσσες προγραμματισμού, όπως η C, η C++, και η Java, οι αριθμοδείκτες ξεκινούν από το 0.



Εικόνα 11.4 **Επεξεργασία συστοιχίας**

Παράδειγμα 11.1

Συγκρίνετε το πλήθος των εντολών που απαιτούνται για τον χειρισμό των 100 ξεχωριστών στοιχείων στην Εικόνα 11.2 και της συστοιχίας με 100 στοιχεία στην Εικόνα 11.4. Υποθέτουμε ότι για την επεξεργασία κάθε αριθμού απαιτείται μόνο μία εντολή.

Λύση

- Στην πρώτη περίπτωση χρειαζόμαστε 100 εντολές για την ανάγνωση, 100 εντολές για την εγγραφή, και 100 εντολές για την επεξεργασία των στοιχείων. Επομένως χρειαζόμαστε συνολικά 300 εντολές.
- Στη δεύτερη περίπτωση μπορούμε να χρησιμοποιήσουμε τρεις βρόχους. Σε κάθε βρόχο χρησιμοποιούμε δύο εντολές, οπότε χρειαζόμαστε έξι εντολές. Ωστόσο, χρειαζόμαστε επίσης και τρεις εντολές για την απόδοση των αρχικών τιμών του αριθμοδείκτη και τρεις εντολές για τον έλεγχο της τιμής του αριθμοδείκτη. Επομένως, συνολικά χρειαζόμαστε δώδεκα εντολές.

Παράδειγμα 11.2

Το πλήθος των κύκλων (οι φάσεις ανάκλησης, αποκωδικοποίησης, και εκτέλεσης) που πρέπει να εκτελέσει ο υπολογιστής δεν μειώνεται αν χρησιμοποιήσουμε συστοιχία. Αντίθετα, το πλήθος των κύκλων στην πραγματικότητα αυξάνεται επειδή τώρα επιβαρυνόμαστε με την απόδοση αρχικών τιμών, και την αύξηση και τον έλεγχο της τιμής του αριθμοδείκτη. Το πρόβλημά μας, όμως, δεν είναι το πλήθος των κύκλων αλλά το πλήθος των γραμμών κώδικα που χρειαζόμαστε για να γράψουμε το πρόγραμμα.

Παράδειγμα 11.3

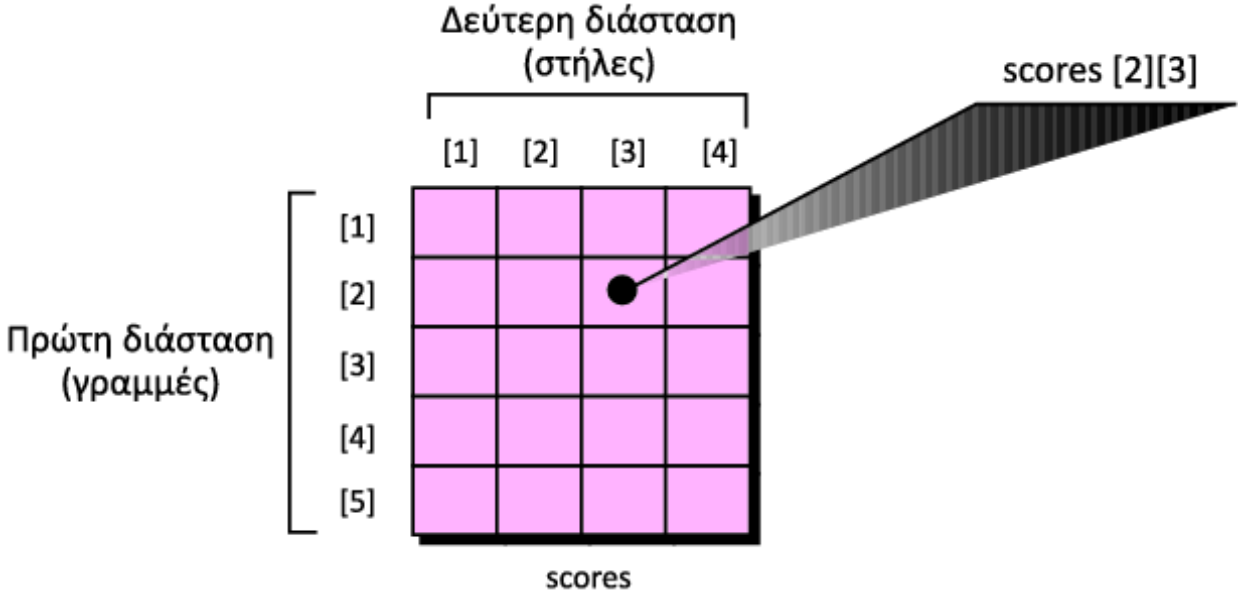
Στην επιστήμη των υπολογιστών, ένα από τα σημαντικότερα θέματα είναι η επαναχρησιμοποίηση των προγραμμάτων — για παράδειγμα, η έκταση των αλλαγών που απαιτούνται όταν αλλάζει το πλήθος των στοιχείων δεδομένων. Ας υποθέσουμε ότι έχουμε γράψει δύο προγράμματα για την επεξεργασία των στοιχείων που παρουσιάζονται στην Εικόνα 11.2 και την Εικόνα 11.4. Αν το πλήθος των στοιχείων αλλάξει από 100 σε 1000, πόσες αλλαγές χρειάζεται να κάνουμε σε κάθε πρόγραμμα; Στο πρώτο πρόγραμμα πρέπει να προσθέσουμε $3 \times 900 = 2700$ εντολές. Στο δεύτερο πρόγραμμα, όμως, χρειάζεται μόνο να αλλάξουμε τρεις παραστάσεις ($I > 100$ έως $I > 1000$). Μάλιστα, μπορούμε να τροποποιήσουμε το διάγραμμα στην Εικόνα 11.4 ώστε να μειώσουμε το πλήθος των αλλαγών μόνο σε μία.

Όνομα συστοιχίας και όνομα στοιχείου

Σε μια συστοιχία υπάρχουν δύο τύποι αναγνωριστικών: το όνομα της συστοιχίας και το όνομα κάθε μεμονωμένου στοιχείου. Το όνομα της συστοιχίας είναι το όνομα ολόκληρης της δομής, ενώ το όνομα ενός στοιχείου μάς επιτρέπει να αναφερόμαστε σε αυτό το στοιχείο. Στην Εικόνα 11.3, το όνομα της συστοιχίας είναι ***scores*** και το όνομα κάθε στοιχείου είναι το όνομα της συστοιχίας ακολουθούμενο από τον αριθμοδείκτη — για παράδειγμα, ***scores*[1]**, ***scores*[2]**, και ούτω καθεξής. Στα παραδείγματα αυτού του κεφαλαίου χρειαζόμαστε κυρίως τα ονόματα των στοιχείων, όμως σε μερικές γλώσσες, όπως η C, πρέπει να χρησιμοποιούμε και το όνομα της συστοιχίας.

Πολυδιάστατες συστοιχίες

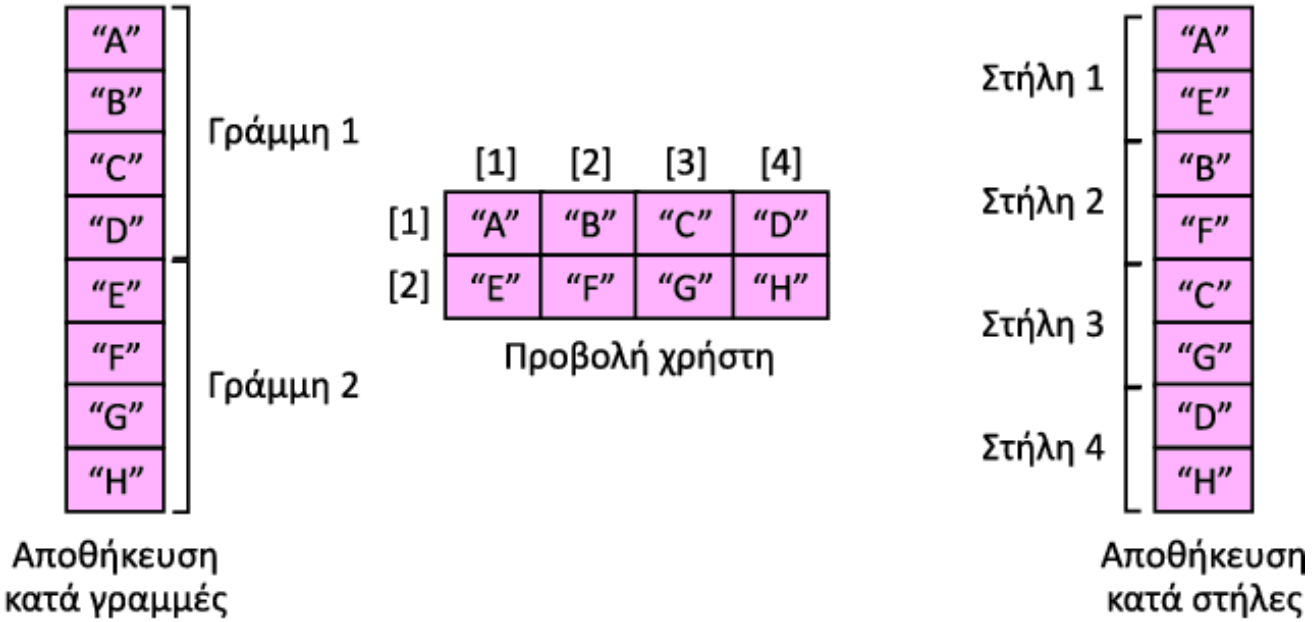
Οι συστοιχίες που περιγράφηκαν μέχρι τώρα είναι γνωστές ως **μονοδιάστατες συστοιχίες**, επειδή τα δεδομένα είναι οργανωμένα με γραμμικό τρόπο σε μία μόνο κατεύθυνση. Ωστόσο, σε πολλές εφαρμογές τα δεδομένα πρέπει να αποθηκεύονται σε περισσότερες από μία διαστάσεις. Στην Εικόνα 11.5 παρουσιάζεται μια συστοιχία, η οποία συνήθως ονομάζεται **διδιάστατη συστοιχία**.



Εικόνα 11.5 Διδιάστατη συστοιχία

Διάταξη μνήμης

Οι αριθμοδείκτες σε μια μονοδιάστατη συστοιχία ορίζουν άμεσα τη σχετική θέση των στοιχείων στην πραγματική μνήμη. Στην Εικόνα 11.6 παρουσιάζεται μια διδιάστατη συστοιχία και ο τρόπος με τον οποίο αποθηκεύεται στη μνήμη με τη μέθοδο αποθήκευσης κατά γραμμές και τη μέθοδο αποθήκευσης κατά στήλες. Η πιο συνηθισμένη, ωστόσο, είναι η αποθήκευση κατά γραμμές.



Εικόνα 11.6 Διάταξη μνήμης με συστοιχίες

Παράδειγμα 11.4

Έχουμε αποθηκεύσει στη μνήμη τη διδιάστατη συστοιχία students. Το μέγεθος της συστοιχίας είναι 100 × 4 (100 γραμμές και 4 στήλες). Δείξτε τη διεύθυνση του στοιχείου students[5][3], υποθέτοντας ότι το στοιχείο student[1][1] είναι αποθηκευμένο στη θέση μνήμης με διεύθυνση 1000 και ότι κάθε στοιχείο καταλαμβάνει μόνο μία θέση μνήμης. Ο υπολογιστής χρησιμοποιεί αποθήκευση κατά γραμμές.

Λύση

Για να βρούμε τη θέση ενός στοιχείου μπορούμε να χρησιμοποιήσουμε τον ακόλουθο τύπο, με την προϋπόθεση ότι κάθε στοιχείο καταλαμβάνει μία θέση μνήμης.

$$y = x + \text{Στήλες} \times (i - 1) + (j - 1)$$

Έτσι, αν το πρώτο στοιχείο καταλαμβάνει τη θέση 1000, τότε το στοιχείο που θέλουμε καταλαμβάνει τη θέση 1018.

Λειτουργίες σε συστοιχίες

Παρόλο που μπορούμε να εφαρμόζουμε συμβατικές λειτουργίες για κάθε στοιχείο μιας συστοιχίας (δείτε το Κεφάλαιο 4), υπάρχουν ορισμένες τις οποίες μπορούμε να ορίσουμε σε μια συστοιχία ως δομή δεδομένων. Ορισμένες συνηθισμένες λειτουργίες που εκτελούνται σε συστοιχίες ως δομές είναι η **αναζήτηση**, η **εισαγωγή**, η **διαγραφή**, η **ανάκτηση**, και η **διάσχιση**.

Παρόλο που η αναζήτηση, η ανάκτηση, και η διάσχιση μιας συστοιχίας γίνονται εύκολα, η εισαγωγή και η διαγραφή είναι χρονοβόρες. Αυτό συμβαίνει επειδή τα στοιχεία πρέπει να μετατοπίζονται προς τα κάτω πριν από μια εισαγωγή και προς τα πάνω μετά από μια διαγραφή.

Στον Αλγόριθμο 11.1 παρέχεται ένα παράδειγμα εύρεσης του μέσου όρου των στοιχείων μιας συστοιχίας τα οποία είναι πραγματικοί αριθμοί.

Αλγόριθμος 11.1 **Υπολογισμός του μέσου όρου των στοιχείων μιας συστοιχίας**

```
Αλγόριθμος: ΜέσοςΌροςΣυστοιχίας (Συστοιχία, n)
Σκοπός: Υπολογισμός της μέσης τιμής
Προ-συνθήκη: Δίνεται η Συστοιχία και το πλήθος των στοιχείων, n.
Μετα-συνθήκη: Τίποτα
Επιστρέφεται: Η μέση τιμή
{
        άθροισμα ← 0,0
        i ← 1

        όσο (i ≤ n)


        {

                άθροισμα ← άθροισμα + Συστοιχία[i]
                i ← i + 1

        }


        μέσος_όρος ← άθροισμα / n


        Επιστροφή (μέσος_όρος)

}
```

Εφαρμογή

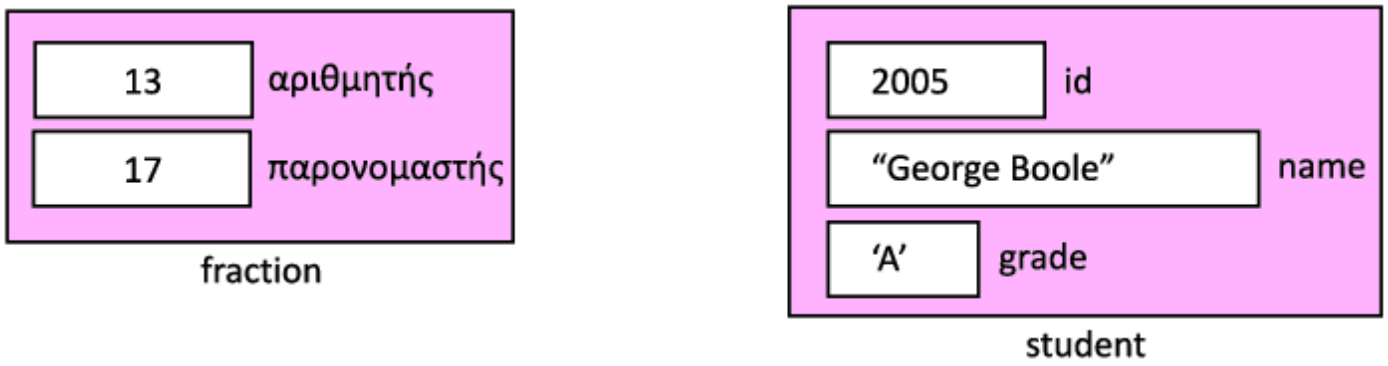
Αν μελετήσουμε τις λειτουργίες που περιγράψαμε στην προηγούμενη ενότητα, μπορούμε να πάρουμε μια ιδέα για την εφαρμογή των συστοιχιών. Αν έχουμε μια λίστα στην οποία αναμένονται πολλές προσθήκες και διαγραφές μετά από τη δημιουργία της, δεν πρέπει να χρησιμοποιήσουμε συστοιχία. Οι συστοιχίες είναι καταλληλότερες όταν το πλήθος των διαγραφών και των προσθηκών είναι μικρό και αναμένονται πολλές ενέργειες αναζήτησης και ανάκτησης.

Μια συστοιχία αποτελεί κατάλληλη δομή όταν απαιτείται μικρό πλήθος προσθηκών και διαγραφών, αλλά χρειάζονται πολλές αναζητήσεις και ανακτήσεις.

11-2 ΕΓΓΡΑΦΕΣ

Μια εγγραφή είναι μια συλλογή από σχετικά μεταξύ τους στοιχεία, πιθανώς διαφορετικών τύπων, η οποία έχει ένα μοναδικό όνομα. Κάθε στοιχείο μιας εγγραφής ονομάζεται **πεδίο**. Το πεδίο είναι το μικρότερο στοιχείο ενός επώνυμου συνόλου δεδομένων με κάποια σημασία. Τα πεδία έχουν τύπο και καταλαμβάνουν θέσεις στη μνήμη. Στα πεδία αντιστοιχίζονται τιμές, οι οποίες με τη σειρά τους μπορούν να προσπελαστούν για επιλογή ή χειρισμό. Τα πεδία διαφέρουν από τις μεταβλητές κυρίως στο ότι αποτελούν τμήματα εγγραφών.

Στην Εικόνα 11.7 μπορείτε να δείτε δύο παραδείγματα εγγραφών. Στο πρώτο παράδειγμα η εγγραφή fraction έχει δύο πεδία, τα οποία είναι και τα δύο ακέραιοι. Στο δεύτερο παράδειγμα η εγγραφή student έχει τρία πεδία τα οποία ανήκουν σε δύο διαφορετικούς τύπους.



Εικόνα 11.7 **Εγγραφές**

Όνομα εγγραφής *και* όνομα πεδίου

Όπως και σε μια συστοιχία, υπάρχουν δύο τύποι αναγνωριστικών σε μια εγγραφή: το όνομα της εγγραφής και το όνομα κάθε μεμονωμένου πεδίου μέσα σε αυτήν. Το όνομα της εγγραφής είναι το όνομα ολόκληρης της δομής, ενώ το όνομα κάθε πεδίου μάς επιτρέπει να αναφερόμαστε στο συγκεκριμένο πεδίο. Για παράδειγμα, στην εγγραφή student της Εικόνας 11.7 το όνομα της εγγραφής είναι student, ενώ τα ονόματα των πεδίων είναι **student.id**, **student.name**, και **student.grade**. Στις περισσότερες γλώσσες προγραμματισμού χρησιμοποιείται η ***τελεία*** (.) για τον διαχωρισμό του ονόματος της δομής (δηλαδή της εγγραφής) από τα ονόματα των στοιχείων της (πεδία). Αυτή είναι και η σύμβαση που χρησιμοποιείται σε αυτό το βιβλίο.

Παράδειγμα 11.5

Στη συνέχεια μπορείτε να δείτε πώς αποθηκεύονται οι τιμές των πεδίων της Εικόνας 11.7.

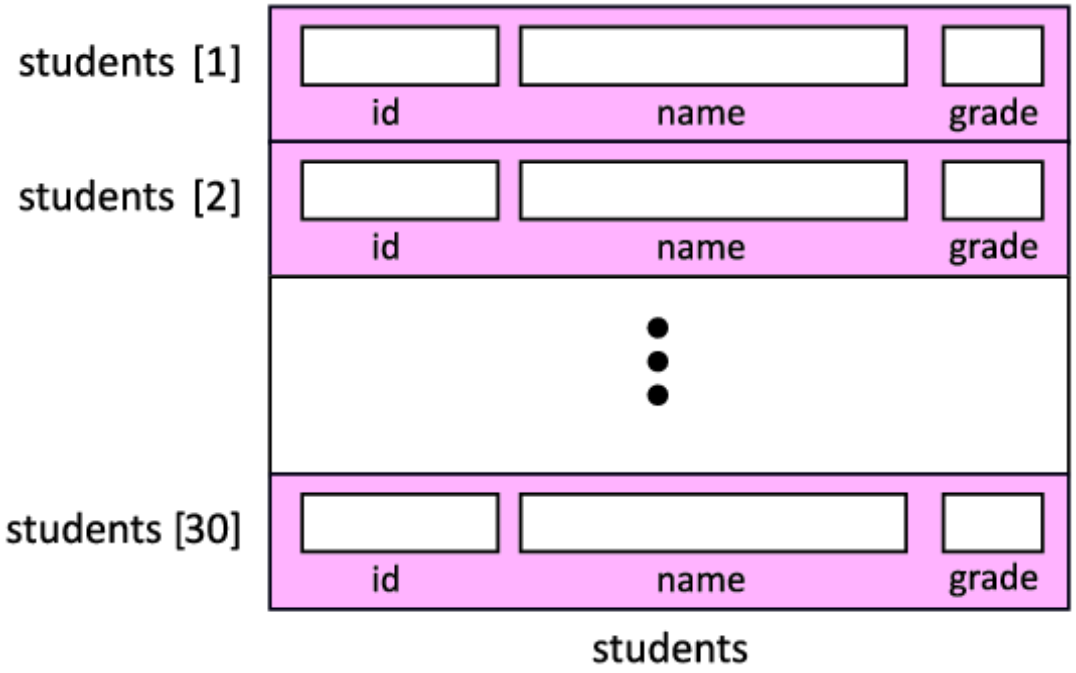
```
student.id ← 2005        student.name ← "G. Boole"      student.grade ← 'A'
```

Σύγκριση εγγραφών και συστοιχιών

Μπορούμε να συγκρίνουμε εννοιολογικά μια συστοιχία με μια εγγραφή. Αυτό θα μας βοηθήσει να κατανοήσουμε πότε πρέπει να χρησιμοποιούμε συστοιχίες και πότε εγγραφές. Μια συστοιχία ορίζει έναν συνδυασμό στοιχείων, ενώ μια εγγραφή ορίζει τα αναγνωρίσιμα μέρη ενός στοιχείου. Για παράδειγμα, μια συστοιχία μπορεί να ορίζει μια τάξη σπουδαστών (40 σπουδαστές), αλλά μια εγγραφή ορίζει διαφορετικές ιδιότητες ενός σπουδαστή, όπως το αναγνωριστικό (id), το όνομα, ή ο βαθμός.

Συστοιχία εγγραφών

Αν χρειάζεται να ορίσουμε έναν συνδυασμό στοιχείων και ταυτόχρονα κάποιες από τις ιδιότητες κάθε στοιχείου, μπορούμε να χρησιμοποιήσουμε μια συστοιχία εγγραφών. Για παράδειγμα, για μια τάξη με 30 σπουδαστές, μπορούμε να χρησιμοποιήσουμε μια συστοιχία με 30 εγγραφές, με κάθε εγγραφή να αντιστοιχεί σε έναν σπουδαστή.



Εικόνα 11.8 **Συστοιχία εγγραφών**

Παράδειγμα 11.6

Στη συνέχεια βλέπετε πώς μπορούμε να προσπελάσουμε τα πεδία κάθε εγγραφής στη συστοιχία students για να αποθηκεύσουμε τιμές σε αυτά.

```
(students[1]).id ← 1001        (students[1]).name ← "J. Aron"      (students[1]).grade ← 'A'
(students[2]).id ← 2007        (students[2]).name ← "F. Bush"      (students[2]).grade ← 'F'
        …                              …                                   …

(students[30]).id ← 3012       (students[30]).name ← "M. Blair"    (students[1]).grade ← 'B'
```

Παράδειγμα 11.7

Ωστόσο, συνήθως θα πρέπει να χρησιμοποιούμε έναν βρόχο για την ανάγνωση των δεδομένων σε μια συστοιχία εγγραφών. Στον Αλγόριθμο 11.2 παρουσιάζεται ένα τμήμα του ψευδοκώδικα που χρησιμοποιείται για αυτήν τη διαδικασία.

Αλγόριθμος 11.2 **Τμήμα ψευδοκώδικα που χρησιμοποιείται για την ανάγνωση των εγγραφών των σπουδαστών**

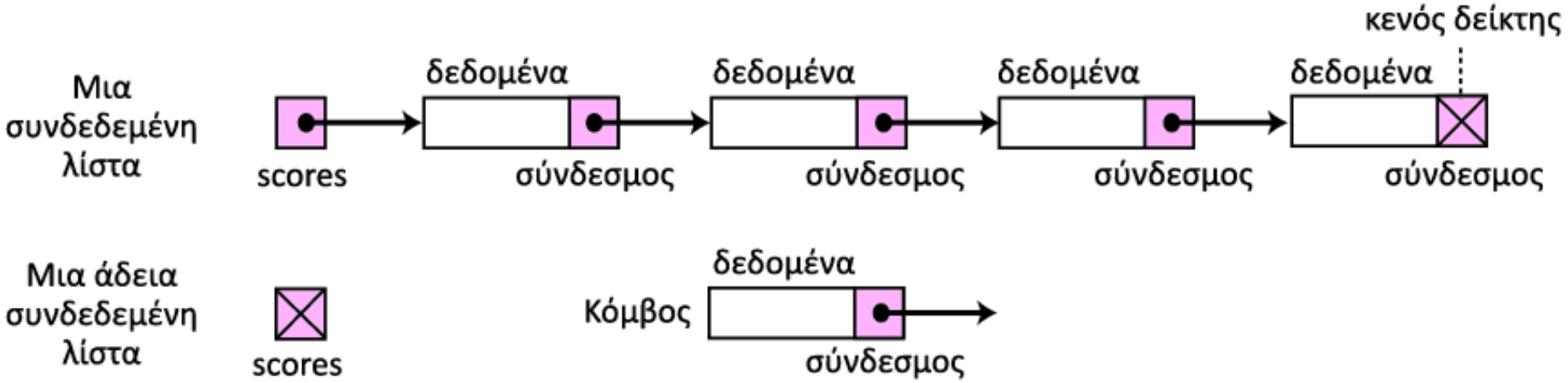
```
i ← 1

while (I < 31)

{
        read(students[i]).id
        read(students[i]).name
        read(students[i]).grade
        i ← i + 1
}
```

Συστοιχίες και συστοιχίες εγγραφών

Τόσο μια συστοιχία όσο και μια συστοιχία εγγραφών αντιπροσωπεύει μια λίστα στοιχείων. Μια συστοιχία μπορεί να θεωρηθεί ως μια ειδική περίπτωση συστοιχίας εγγραφών, στην οποία κάθε στοιχείο είναι μια εγγραφή με ένα μόνο πεδίο.

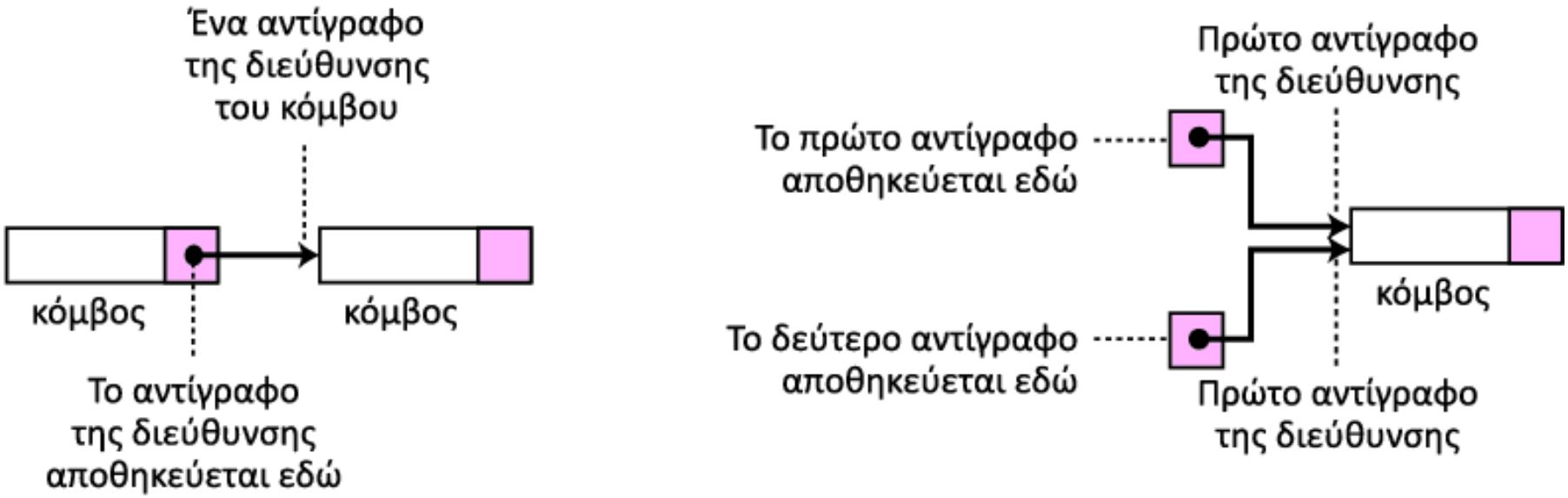
11-3 ΣΥΝΔΕΔΕΜΕΝΕΣ ΛΙΣΤΕΣ

Μια συνδεδεμένη λίστα είναι μια συλλογή δεδομένων στην οποία κάθε στοιχείο περιέχει τη θέση του επόμενου στοιχείου - δηλαδή, κάθε στοιχείο χωρίζεται σε δύο τμήματα: τα **δεδομένα** και τον **σύνδεσμο**. Το όνομα της λίστας είναι ίδιο με το όνομα αυτής της μεταβλητής δείκτη. Στην Εικόνα 11.9 παρουσιάζεται μια συνδεδεμένη λίστα με όνομα *scores*, η οποία περιέχει τέσσερα στοιχεία. Μια κενή συνδεδεμένη λίστα ορίζεται με τη μορφή ενός κενού δείκτη. Στην Εικόνα 11.9 φαίνεται επίσης ένα παράδειγμα κενής συνδεδεμένης λίστας.



Εικόνα 11.9 **Συνδεδεμένες λίστες**

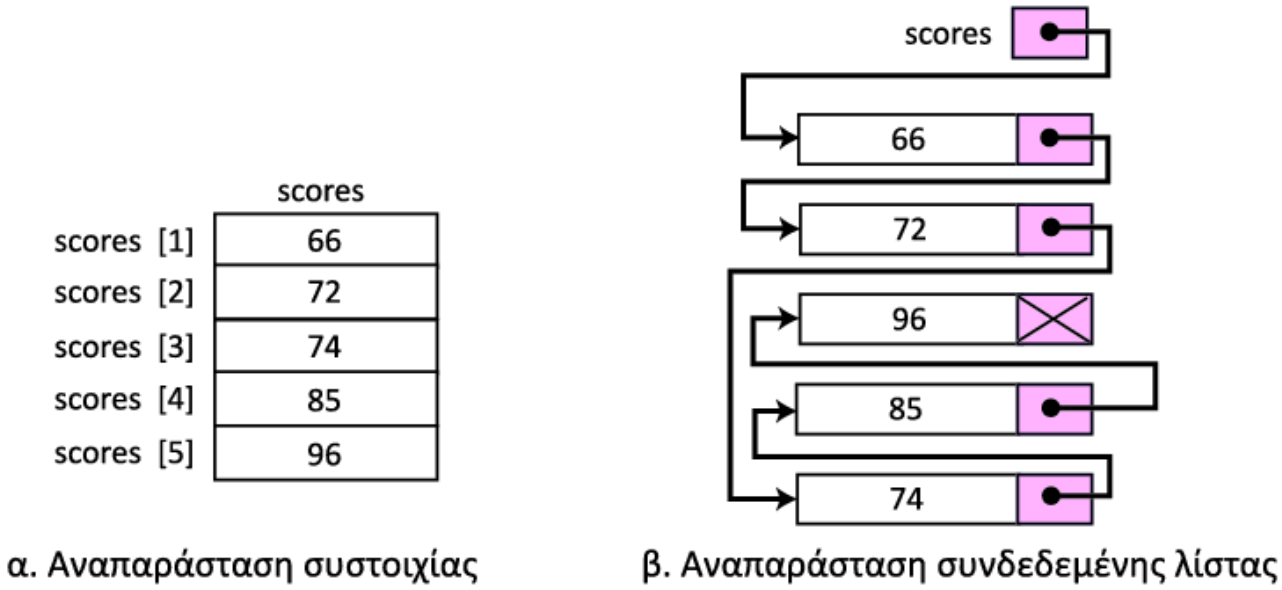
Πριν περιγράψουμε με περισσότερες λεπτομέρειες τις συνδεδεμένες λίστες, πρέπει να εξηγήσουμε τον συμβολισμό που χρησιμοποιούμε στις εικόνες. Η σύνδεση μεταξύ δύο κόμβων υποδεικνύεται με μια γραμμή. Στο ένα άκρο της γραμμής υπάρχει ένα βέλος και στο άλλο άκρο ένας συμπαγής κύκλος.



Εικόνα 11.10 Η έννοια της αντιγραφής και της αποθήκευσης δεικτών

Συστοιχίες και συνδεδεμένες λίστες

Τόσο μια συστοιχία όσο και μια συνδεδεμένη λίστα αποτελούν αναπαραστάσεις μιας λίστας στοιχείων στη μνήμη. Η μοναδική διαφορά τους βρίσκεται στον τρόπο με τον οποίο τα στοιχεία συνδέονται μεταξύ τους. Στην Εικόνα 11.11 παρουσιάζεται μια σύγκριση των δύο αναπαραστάσεων για μια λίστα με πέντε ακεραίους.

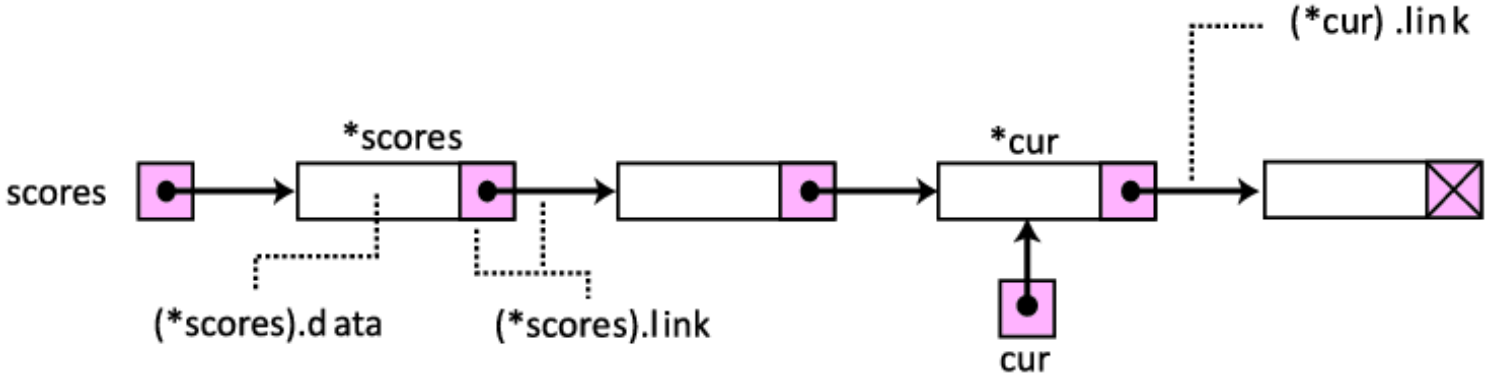


α. Αναπαράσταση συστοιχίας β. Αναπαράσταση συνδεδεμένης λίστας

Εικόνα 11.11 Συστοιχία και συνδεδεμένη λίστα

Ονόματα συνδεδεμένων λιστών και ονόματα κόμβων

Όπως και με τις συστοιχίες και τις εγγραφές, θα πρέπει να γίνεται διάκριση ανάμεσα στο όνομα της συνδεδεμένης λίστας και τα ονόματα των κόμβων, δηλαδή τα στοιχεία της συνδεδεμένης λίστας. Μια συνδεδεμένη λίστα πρέπει να έχει ένα όνομα. Το όνομα μιας συνδεδεμένης λίστας είναι το όνομα του δείκτη αρχής που δείχνει στον πρώτο κόμβο της λίστας. Οι κόμβοι, από την άλλη πλευρά, δεν έχουν σαφή ονόματα σε μια συνδεδεμένη λίστα, αλλά υπονοούμενα.



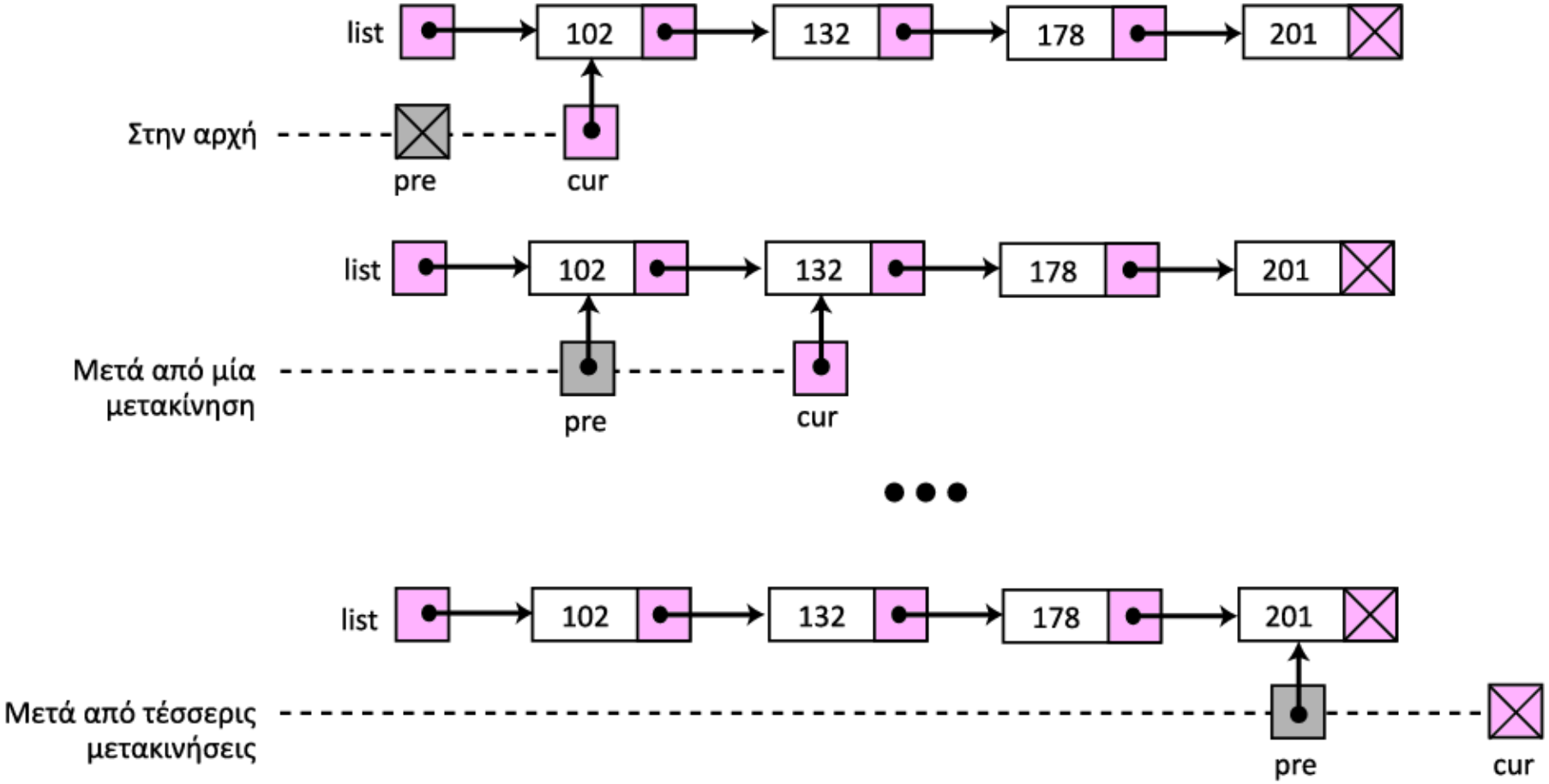
Εικόνα 11.12 **Το όνομα μιας συνδεδεμένης λίστας και τα ονόματα των κόμβων**

Λειτουργίες σε συνδεδεμένες λίστες

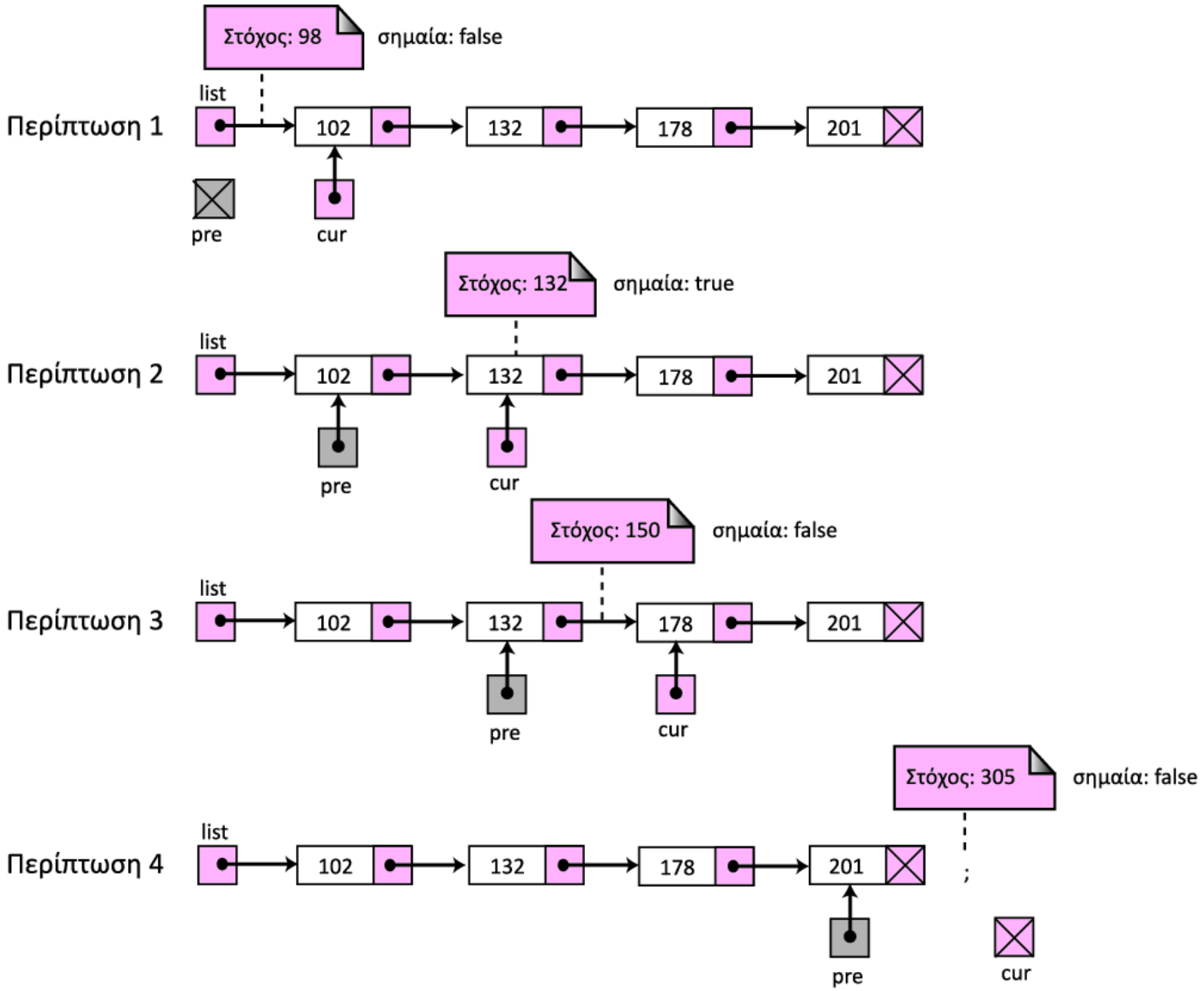
Οι ίδιες λειτουργίες που ορίζονται σε μια συστοιχία μπορούν να εφαρμοστούν και σε μια συνδεδεμένη λίστα.

Αναζήτηση σε συνδεδεμένη λίστα

Επειδή οι κόμβοι μιας συνδεδεμένης λίστας δεν έχουν ονόματα, χρησιμοποιούμε δύο δείκτες, τους **pre** (από τη λέξη previous, προηγούμενος) και **cur** (από τη λέξη current, δηλαδή τρέχων). Στην αρχή της αναζήτησης, ο δείκτης pre είναι κενός και ο δείκτης cur δείχνει στον πρώτο κόμβο. Ο αλγόριθμος αναζήτησης μετακινεί ταυτόχρονα τους δύο δείκτες προς το τέλος της λίστας. Στην Εικόνα 11.13 βλέπετε την κίνηση αυτών των δύο δεικτών μέσα στη λίστα σε μια ακραία περίπτωση: όταν η τιμή που ψάχνουμε είναι μεγαλύτερη από οποιαδήποτε άλλη τιμή της λίστας.



Εικόνα 11.13 **Η μετακίνηση των δεικτών *pre* και *cur* κατά την αναζήτηση σε μια συνδεδεμένη λίστα**



Εικόνα 11.14 **Οι τιμές των δεικτών *pre* και *cur* σε διαφορετικές περιπτώσεις**

Αλγόριθμος 11.3 Αναζήτηση σε συνδεδεμένη λίστα

Αλγόριθμος: **ΑναζήτησηΣυνδεδεμένηςΛίστας** (list, target)

Σκοπός: Αναζήτηση στη λίστα με χρήση δύο δεικτών **pre** και **cur**.

Προ-συνθήκη: Δίνεται η συνδεδεμένη λίστα (ο δείκτης αρχής) και η τιμή προορισμού (ο στόχος)

Μετα-συνθήκη: Τίποτα

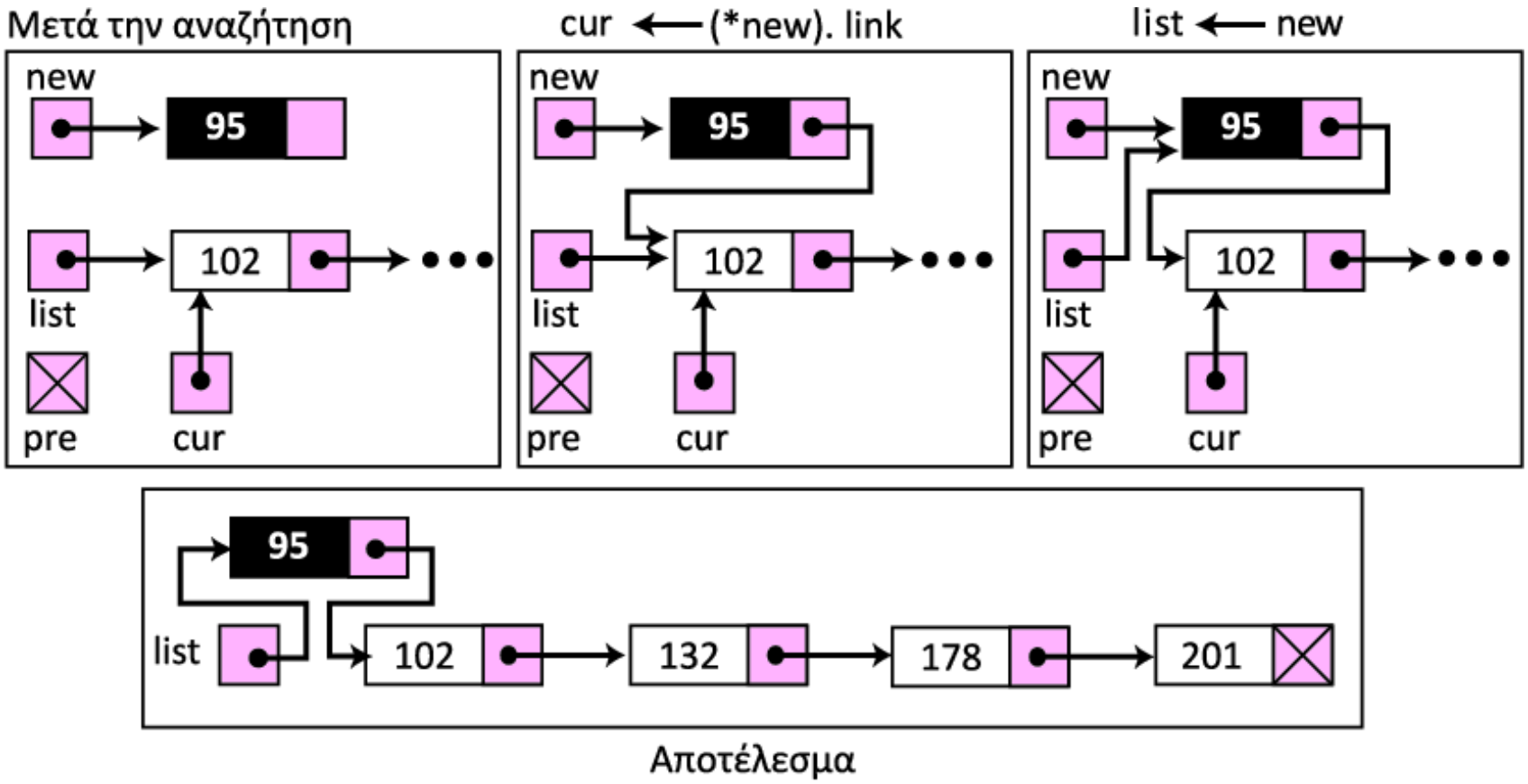
Επιστρέφεται: Η θέση των δεικτών **pre** και **cur** και η τιμή της σημαίας (*true* ή *false*)

```
{
        pre ← null
        cur ← list
        while (target < (*cur).data)
        {
            pre ← cur
            cur ← (*cur).link
        }
        if ((*cur).data = target)    flag ← true
        else   flag ← false
        return (cur, pre, flag)
}
```

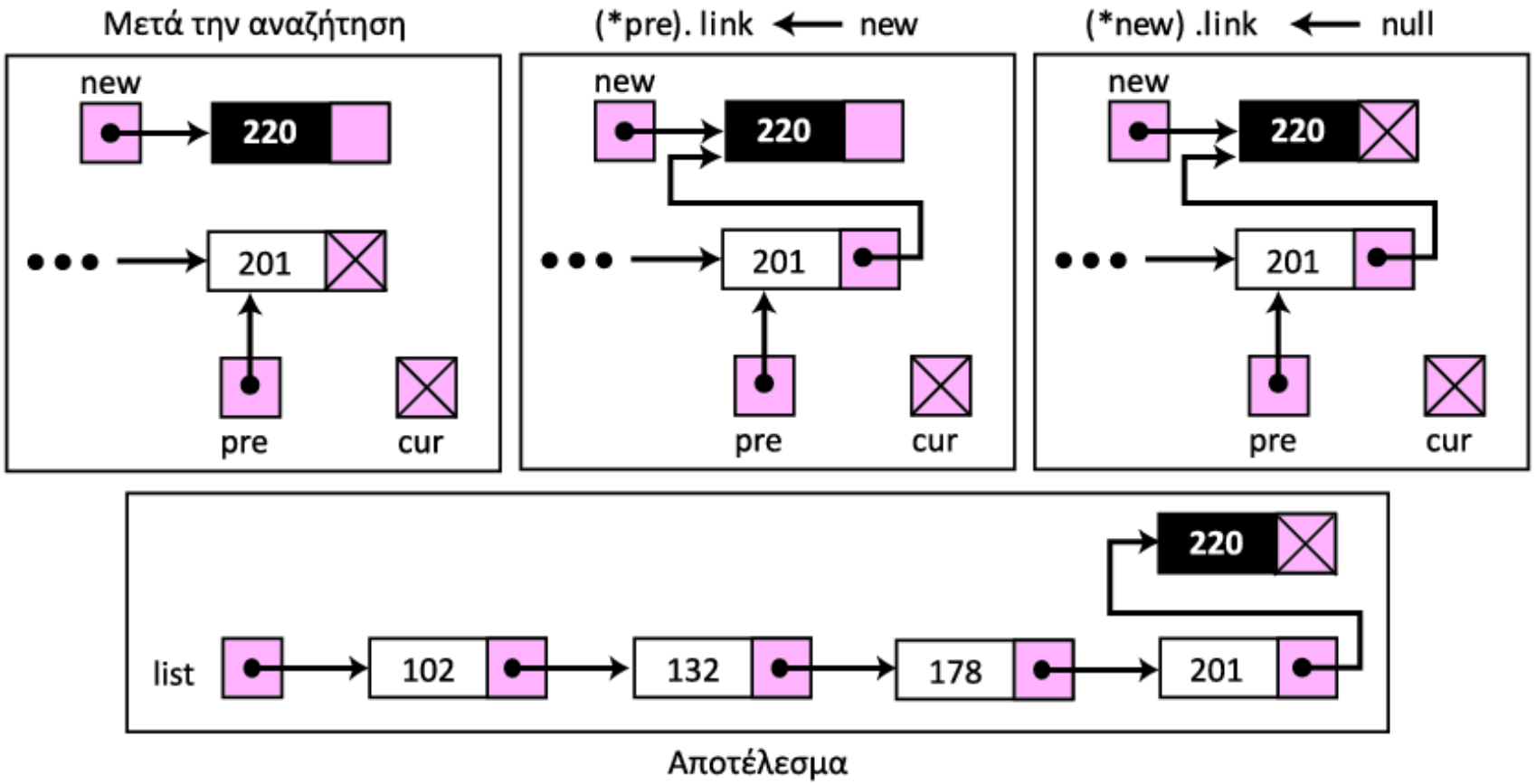
Εισαγωγή κόμβου

Πριν την εισαγωγή ενός κόμβου σε μια συνδεδεμένη λίστα, πρέπει πρώτα να εφαρμόσουμε τον αλγόριθμο αναζήτησης. Αν η σημαία που επιστρέφεται από τον αλγόριθμο αναζήτησης έχει την τιμή false τότε η εισαγωγή επιτρέπεται, διαφορετικά πρέπει να εγκαταλείψουμε τον αλγόριθμο εισαγωγής επειδή δεν επιτρέπονται δεδομένα με ίδιες τιμές. Εδώ μπορούν να παρουσιαστούν τέσσερις περιπτώσεις:

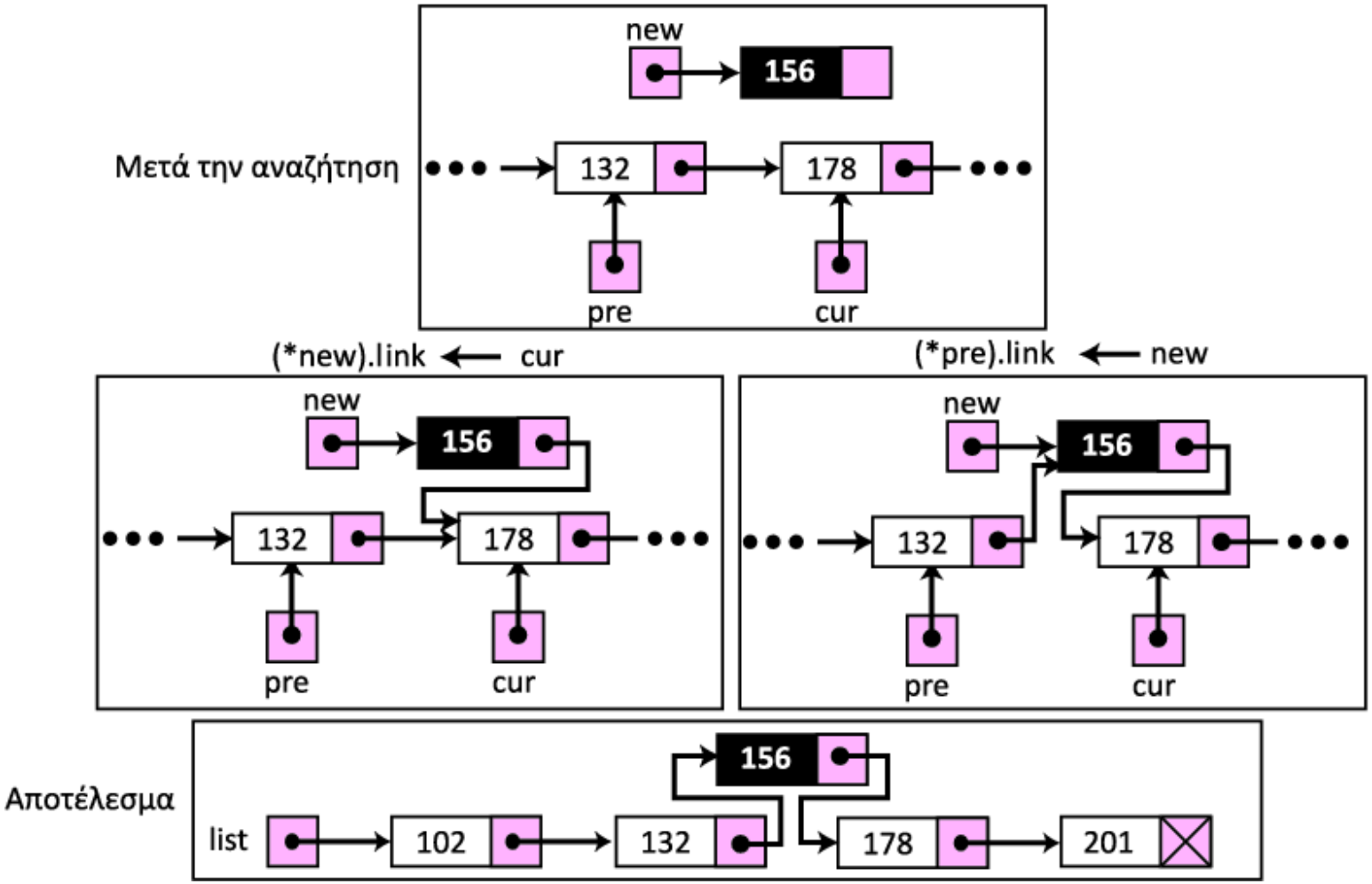
- Εισαγωγή σε κενή λίστα.
- Εισαγωγή στην αρχή της λίστας.
- Εισαγωγή στο τέλος της λίστας.
- Εισαγωγή στο μέσο της λίστας.



Εικόνα 11.15 **Εισαγωγή κόμβου στην αρχή μιας συνδεδεμένης λίστας**



Εικόνα 11.16 **Εισαγωγή κόμβου στο τέλος μιας συνδεδεμένης λίστας**



Εικόνα 11.17 Εισαγωγή κόμβου στο μέσο μιας συνδεδεμένης λίστας

Αλγόριθμος 11.4 Εισαγωγή κόμβου σε συνδεδεμένη λίστα

Αλγόριθμος: **ΕισαγωγήΣεΣυνδεδεμένηΛίστα** (list, target, new)

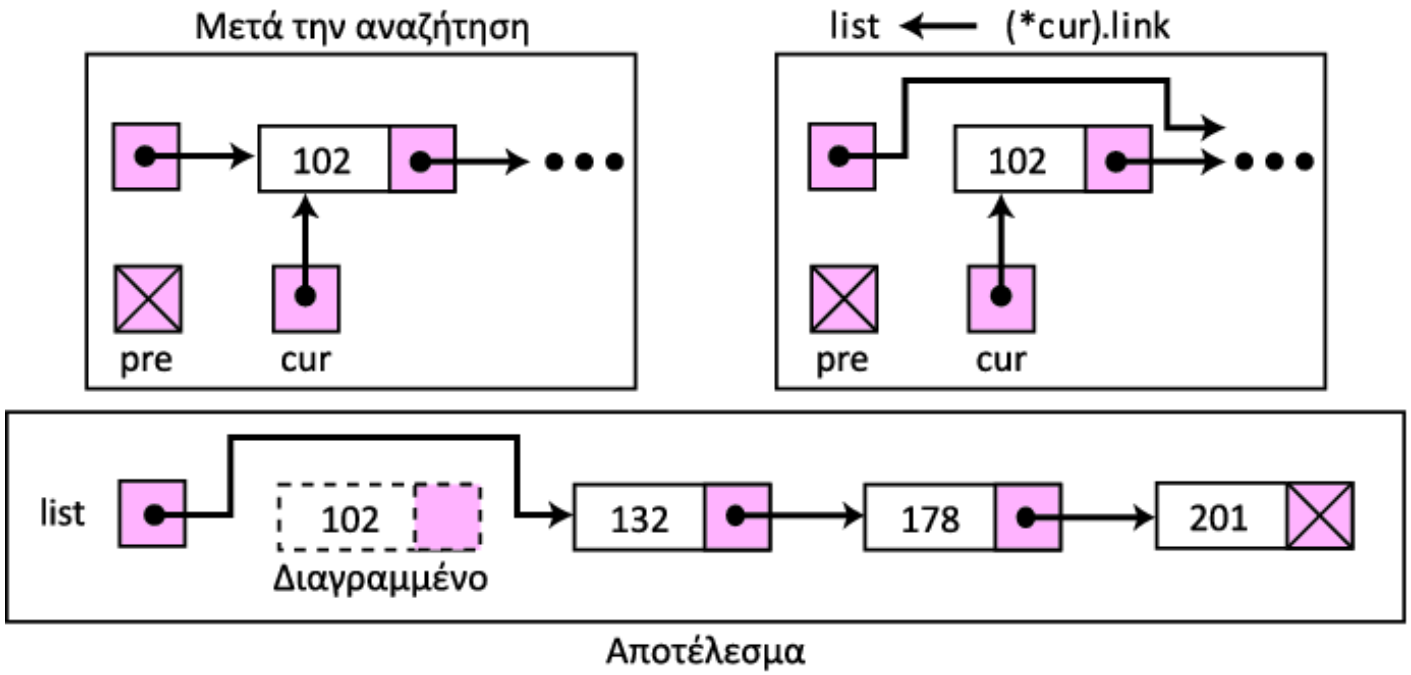
Σκοπός: Εισαγωγή ενός κόμβου στη λίστα συνδέσμων μετά από αναζήτηση στη λίστα για την κατάλληλη θέση.

Προ-συνθήκη: Δίνεται η συνδεδεμένη λίστα και τα δεδομένα προορισμού που πρέπει να εισαχθούν
Μετα-συνθήκη: Τίποτα
Επιστρέφεται: Η νέα συνδεδεμένη λίστα

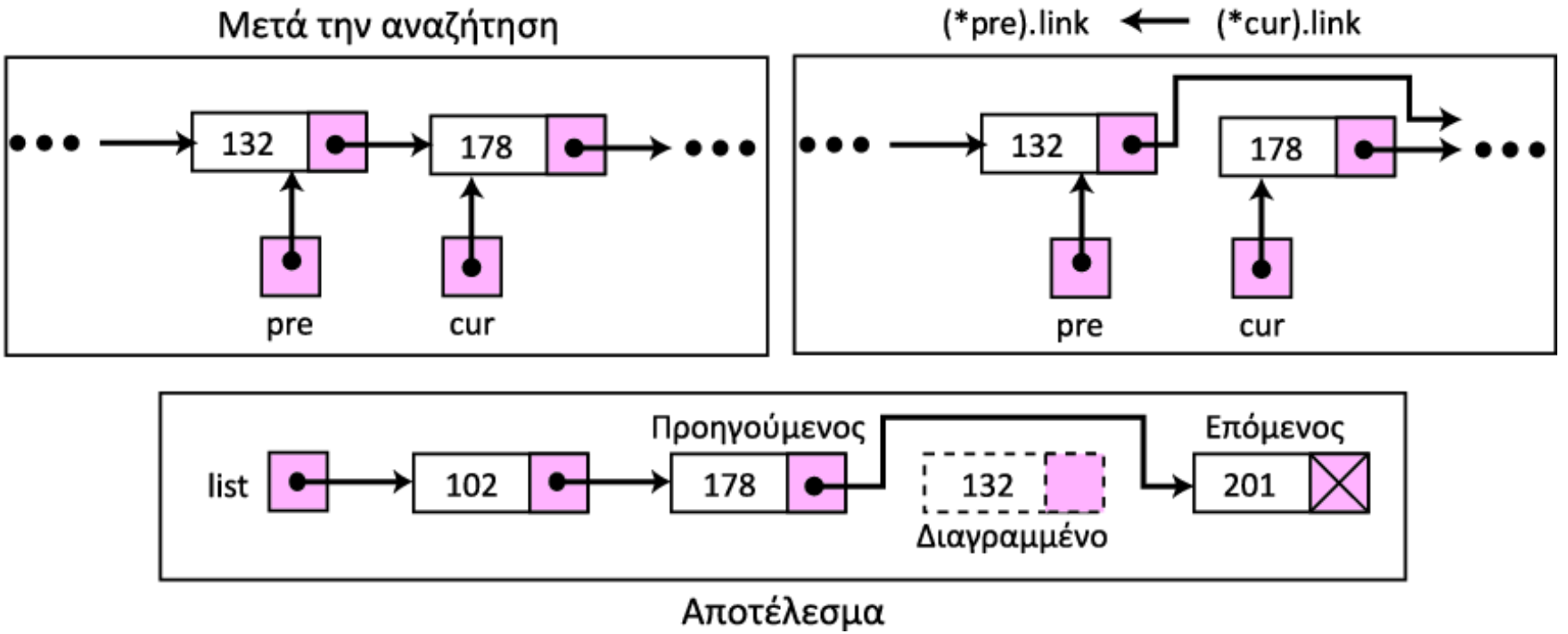
```
{
        searchlinkedlist (list, target, pre, cur, flag)
        // Δίνεται ο στόχος και οι επιστρεφόμενες τιμές pre, cur, και flag

        if (flag = true)  return list        // Δεν επιτρέπονται διπλότυπα
        if (list != null)                    // Εισαγωγή σε κενή λίστα
        {
                list ← new
        }
        if (pre = null)                      // Εισαγωγή στην αρχή
        {
            (*new).link ← cur
            list ← new
            return list
        }

        if (cur = null)                      // Εισαγωγή στο τέλος
        {
                (*pre).link ← new
                (*new).link ← null
                return list
        }

        (*new).link ← cur                    // Εισαγωγή στο μέσο
        (*pre).link ← new
        return list
}
```

Διαγραφή κόμβου

Πριν τη διαγραφή ενός κόμβου σε μια συνδεδεμένη λίστα, πρέπει να εφαρμόσουμε τον αλγόριθμο αναζήτησης. Αν η σημαία που επιστρέφεται από τον αλγόριθμο αναζήτησης είναι true (αν δηλαδή εντοπιστεί ο κόμβος), μπορούμε να διαγράψουμε τον κόμβο από τη συνδεδεμένη λίστα. Ωστόσο, η διαγραφή είναι απλούστερη από την εισαγωγή αφού υπάρχουν μόνο δύο περιπτώσεις — η διαγραφή του πρώτου κόμβου και η διαγραφή οποιουδήποτε άλλου κόμβου. Με άλλα λόγια, η διαγραφή του τελευταίου και των μεσαίων κόμβων μπορεί να γίνει με την ίδια διαδικασία.



Εικόνα 11.18 **Διαγραφή του πρώτου κόμβου μιας συνδεδεμένης λίστας**



Εικόνα 11.19 **Διαγραφή ενός κόμβου στο μέσο ή το τέλος μιας συνδεδεμένης λίστας**

Αλγόριθμος 11.5 Διαγραφή ενός κόμβου σε μια συνδεδεμένη λίστα

```
Αλγόριθμος: ΔιαγραφήΑπόΣυνδεδεμένηΛίστα (list, target)
Σκοπός: Διαγραφή ενός κόμβου σε μια συνδεδεμένη λίστα μετά από αναζήτηση στη λίστα για τον κατάλληλο κόμβο
Προ-συνθήκη:  Δίνεται η συνδεδεμένη λίστα και τα δεδομένα προορισμού που πρέπει να διαγραφούν
Μετα-συνθήκη:  Τίποτα
Επιστρέφεται: Η νέα συνδεδεμένη λίστα
{
        // Δίνεται ο στόχος και οι επιστρεφόμενες τιμές των pre, cur, και flag
        searchlinkedlist (list, target, pre, cur, flag)

        if (flag = false) return list                    // Δεν εντοπίστηκε ο κόμβος προς διαγραφή

        if (pre = null)                                  // Διαγραφή του πρώτου κόμβου
        {
                list ← (*cur).link
                return list
        }

        (*pre).link ← (*cur).link                        // Διαγραφή άλλων κόμβων

        return list
}
```

Ανάκτηση κόμβου

Ανάκτηση είναι η τυχαία προσπέλαση ενός κόμβου για την εξέταση ή την αντιγραφή των δεδομένων που περιέχονται σε αυτόν. Πριν την ανάκτηση ενός κόμβου, όμως, αυτός πρέπει να εντοπιστεί μέσα στη συνδεδεμένη λίστα. Αν βρεθεί το στοιχείο δεδομένων θα ανακτηθεί, διαφορετικά η διαδικασία εγκαταλείπεται. Στην ανάκτηση χρησιμοποιείται μόνο ο δείκτης cur, ο οποίος δείχνει στον κόμβο που εντοπίζεται από τον αλγόριθμο αναζήτησης. Ο Αλγόριθμος 11.6 δείχνει τον ψευδοκώδικα για την ανάκτηση των δεδομένων ενός κόμβου. Ο αλγόριθμος είναι πολύ πιο απλός από τους αλγορίθμους εισαγωγής και διαγραφής.

Αλγόριθμος 11.6 Ανάκτηση ενός κόμβου σε μια συνδεδεμένη λίστα

Αλγόριθμος: **ΑνάκτησηΑπόΣυνδεδεμένηΛίστα** (list, target)

Σκοπός: Ανάκτηση των δεδομένων σε έναν κόμβο μιας συνδεδεμένης λίστας μετά από αναζήτηση στη λίστα για τον κατάλληλο κόμβο

Προ-συνθήκη: Δίνεται η συνδεδεμένη λίστα (ο δείκτης αρχής) και ο στόχος (τα δεδομένα που πρέπει να ανακτηθούν)

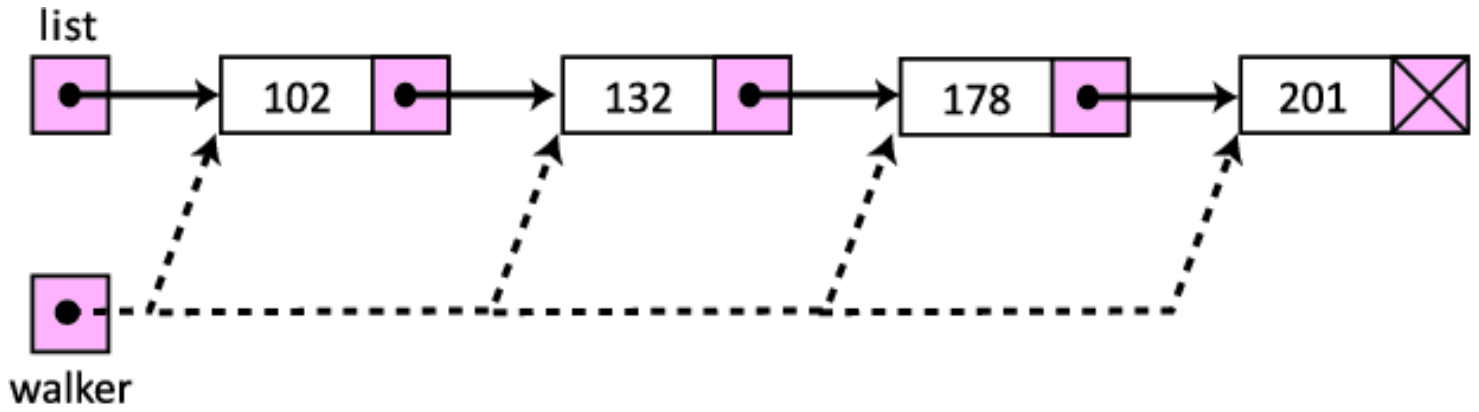
Μετα-συνθήκη: Τίποτα

Επιστρέφονται: Τα ανακτημένα δεδομένα

```
{
        searchlinkedlist (list, target, pre, cur, flag)
        if (flag = false) return error     // Δεν βρέθηκε ο κόμβος
        return (*cur).data
}
```

Διάσχιση συνδεδεμένης λίστας

Για να διασχίσουμε μια λίστα χρειαζόμαστε έναν "μετακινούμενο" δείκτη, δηλαδή έναν δείκτη ο οποίος μετακινείται από κόμβο σε κόμβο κατά την επεξεργασία κάθε στοιχείου. Η διάσχιση ξεκινά με τον ορισμό του μετακινούμενου δείκτη στον πρώτο κόμβο της λίστας. Κατόπιν, με τη χρήση ενός βρόχου συνεχίζουμε μέχρι να ολοκληρωθεί η επεξεργασία όλων των δεδομένων. Με κάθε επανάληψη του βρόχου γίνεται επεξεργασία του τρέχοντος κόμβου, και έπειτα ο μετακινούμενος δείκτης προχωρά στον επόμενο κόμβο. Όταν ολοκληρωθεί η επεξεργασία και του τελευταίου κόμβου, ο μετακινούμενος δείκτης γίνεται κενός (null) και ο βρόχος τερματίζεται (Εικόνα 11.20).



Εικόνα 11.20 **Διάσχιση μιας συνδεδεμένης λίστας**

Αλγόριθμος 11.7 Διάσχιση συνδεδεμένης λίστας

Αλγόριθμος: **ΔιάσχισηΣυνδεδεμένηςΛίστας** (list)

Σκοπός: Διάσχιση μιας συνδεδεμένης λίστας και επεξεργασία κάθε στοιχείου δεδομένων

Προ-συνθήκη: Δίνεται η συνδεδεμένη λίστα (ο δείκτης αρχής)

Μετα-συνθήκη: Τίποτα

Επιστρέφεται: Η λίστα

```
{
        walker ← list

        while (walker ≠ null)

        {
                Process (*walker).data
                walker ←
                (*walker).link
        }

        return list

}
```

Εφαρμογές συνδεδεμένων λιστών

Μια συνδεδεμένη λίστα είναι μια δομή δεδομένων ιδιαίτερα αποδοτική για την αποθήκευση δεδομένων στα οποία πρόκειται να γίνουν πολλές εισαγωγές και διαγραφές. Μια συνδεδεμένη λίστα είναι μια δυναμική δομή δεδομένων η οποία αρχικά μπορεί να μην έχει κανέναν κόμβο και κατόπιν να αρχίσει να αναπτύσσεται με την προσθήκη απαιτούμενων κόμβων. Η διαγραφή ενός κόμβου μπορεί να γίνει εύκολα χωρίς να μετακινηθούν άλλοι κόμβοι, όπως συμβαίνει στην περίπτωση μιας συστοιχίας. Για παράδειγμα, θα μπορούσε να χρησιμοποιηθεί μια συνδεδεμένη λίστα για την αποθήκευση των εγγραφών των σπουδαστών σε ένα σχολείο. Κάθε τρίμηνο ή εξάμηνο εγγράφονται νέοι σπουδαστές στο σχολείο, ενώ άλλοι μπορεί να φεύγουν ή να αποφοιτούν.

Οι συνδεδεμένες λίστες αποτελούν κατάλληλες δομές όταν απαιτείται ένα μεγάλο πλήθος εισαγωγών και διαγραφών, όμως οι αναζητήσεις σε αυτές είναι πιο αργές από ό,τι οι αναζητήσεις σε συστοιχίες.