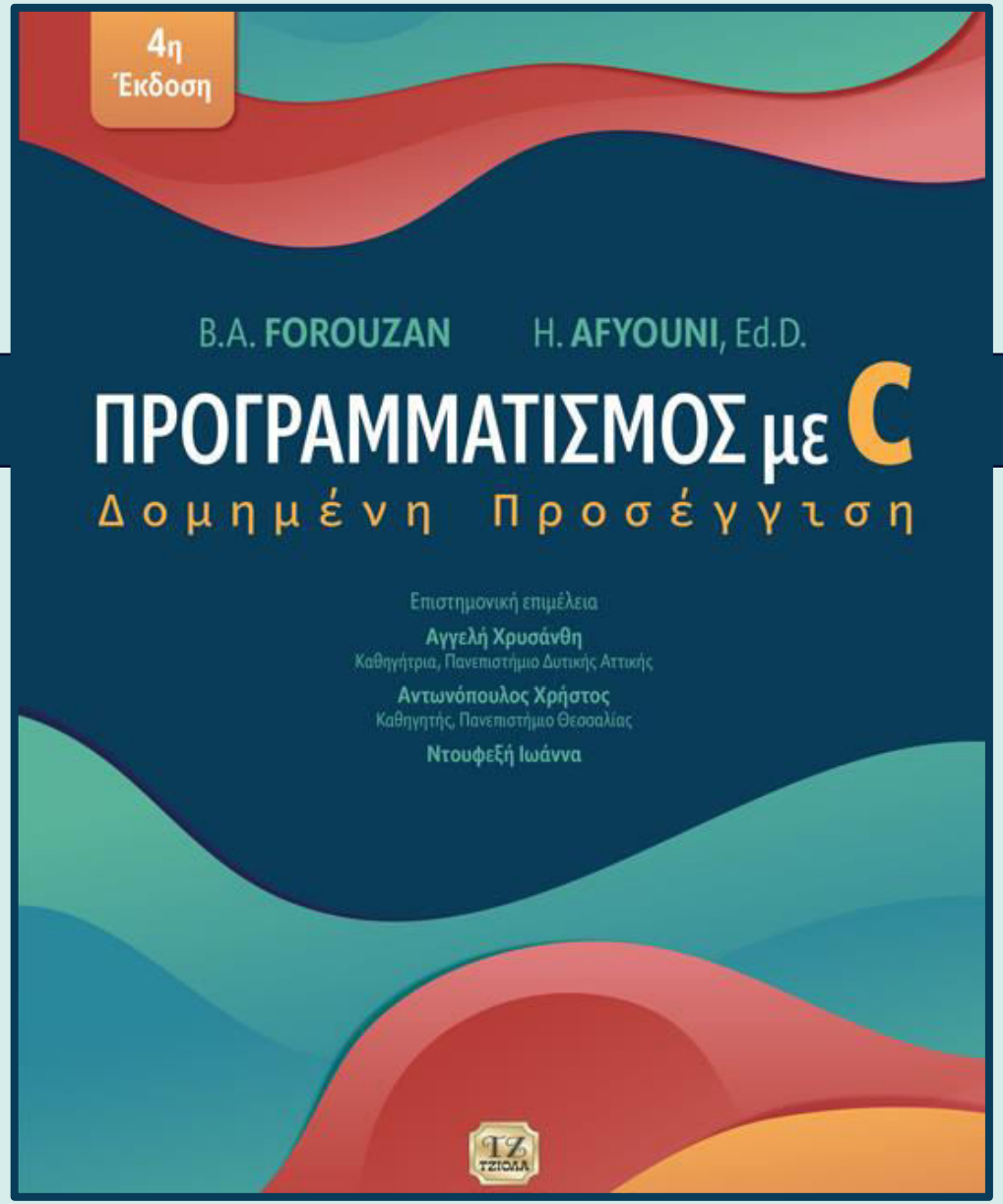


Επιστημονικές Εκδόσεις
ΤΖΙΟΛΑ

ΚΕΦΑΛΑΙΟ 15

Λίστες Δεδομένων

ΘΕΜΑΤΑ ΚΕΦΑΛΑΙΟΥ

- Υλοποίηση διασυνδεδεμένων λιστών στη C
- Βασικές λειτουργίες χειρισμού γραμμικών λιστών
- Υλοποίηση δομών στοίβας στη C
- Υλοποίηση δομών ουράς στη C
- Βασικές έννοιες των δομών δένδρων
- Βασικές έννοιες των δομών γράφων

15.1
Υλοποιήσεις λιστών δεδομένων

Ορισμός

Ο όρος λίστα (list) αναφέρεται σε μία συλλογή σχετιζόμενων δεδομένων

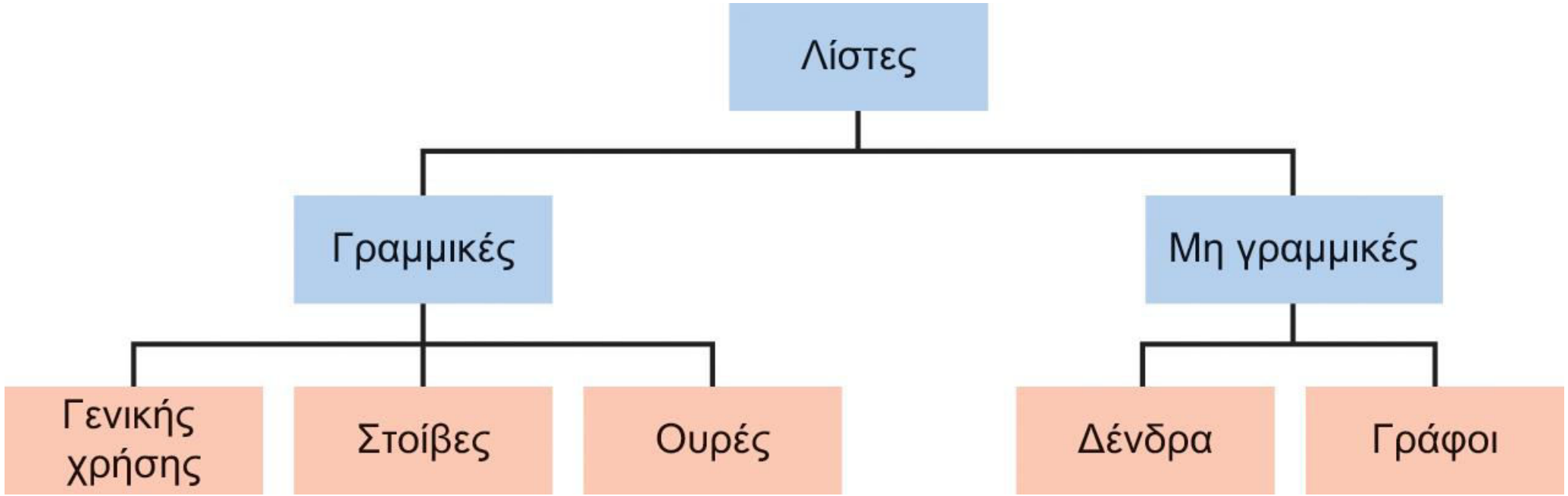
Υλοποιήσεις λιστών

- Σε μία γραμμική λίστα (linear list), κάθε στοιχείο μπορεί να έχει μηδέν ή ένα διάδοχο (επόμενο) στοιχείο
  - Οι γραμμικές λίστες διακρίνονται σε
    - λίστες «γενικής χρήσης»
    - στοίβες
    - ουρές

Υλοποιήσεις λιστών

- Σε μία μη γραμμική λίστα (nonlinear list), κάθε στοιχείο μπορεί να έχει μηδέν, ένα ή περισσότερα επόμενα στοιχεία
  - Οι μη γραμμικές λίστες διακρίνονται σε
    - δένδρα
    - γράφους

Υλοποιήσεις λιστών



ΣΧΗΜΑ 15-1 Τύποι λιστών.

Υλοποίηση δομών λίστας με χρήση πινάκων

- Σε μία υλοποίηση με χρήση πίνακα, ο ακολουθιακός χαρακτήρας της λίστας εξασφαλίζεται από τη διαδοχικότητα των στοιχείων του πίνακα
- Οι πίνακες χρησιμοποιούνται σπάνια για την υλοποίηση λιστών, ιδιαίτερα δε όταν μία λίστα πρόκειται να αλλάζει συχνά, κυρίως διότι
  - Η προσθήκη και η διαγραφή στοιχείων σε έναν πίνακα είναι πολύπλοκες και αναποτελεσματικές διαδικασίες
  - Οι υλοποιήσεις λιστών με πίνακες τείνουν να γίνονται ογκώδεις ως προς το μέγεθος, πολύπλοκες ως προς τη δομή και δαπανηρές ως προς τη μνήμη

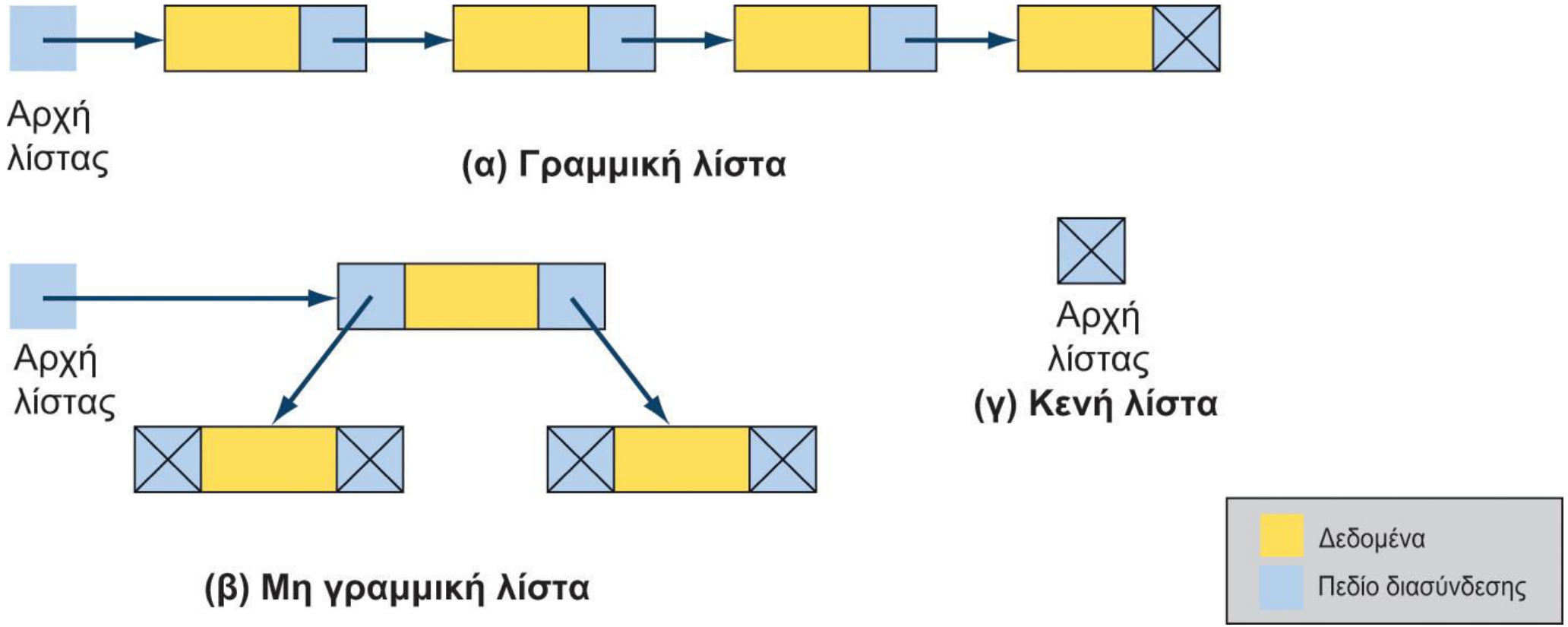
Υλοποίηση ως διασυνδεδεμένη λίστα (1/6)

- Μία διασυνδεδεμένη λίστα (linked list) είναι μία οργανωμένη συλλογή στοιχείων δεδομένων, όπου κάθε στοιχείο-μέλος της λίστας περιέχει, επιπρόσθετα με τα δεδομένα, πληροφορία για τη θέση μνήμης του επόμενου ή των επόμενων στοιχείων
  - Κάθε στοιχείο συγκροτείται από δύο τμήματα: το «τμήμα δεδομένων» και το «τμήμα διασύνδεσης»
    - Το «τμήμα δεδομένων» περιέχει τα καθαυτό δεδομένα προς επεξεργασία από την εφαρμογή
    - Το «τμήμα διασύνδεσης» περιέχει ένα ή περισσότερα πεδία, τα οποία που χρησιμοποιούνται για τη λογική διασύνδεση των δεδομένων

Υλοποίηση ως διασυνδεδεμένη λίστα (2/6)

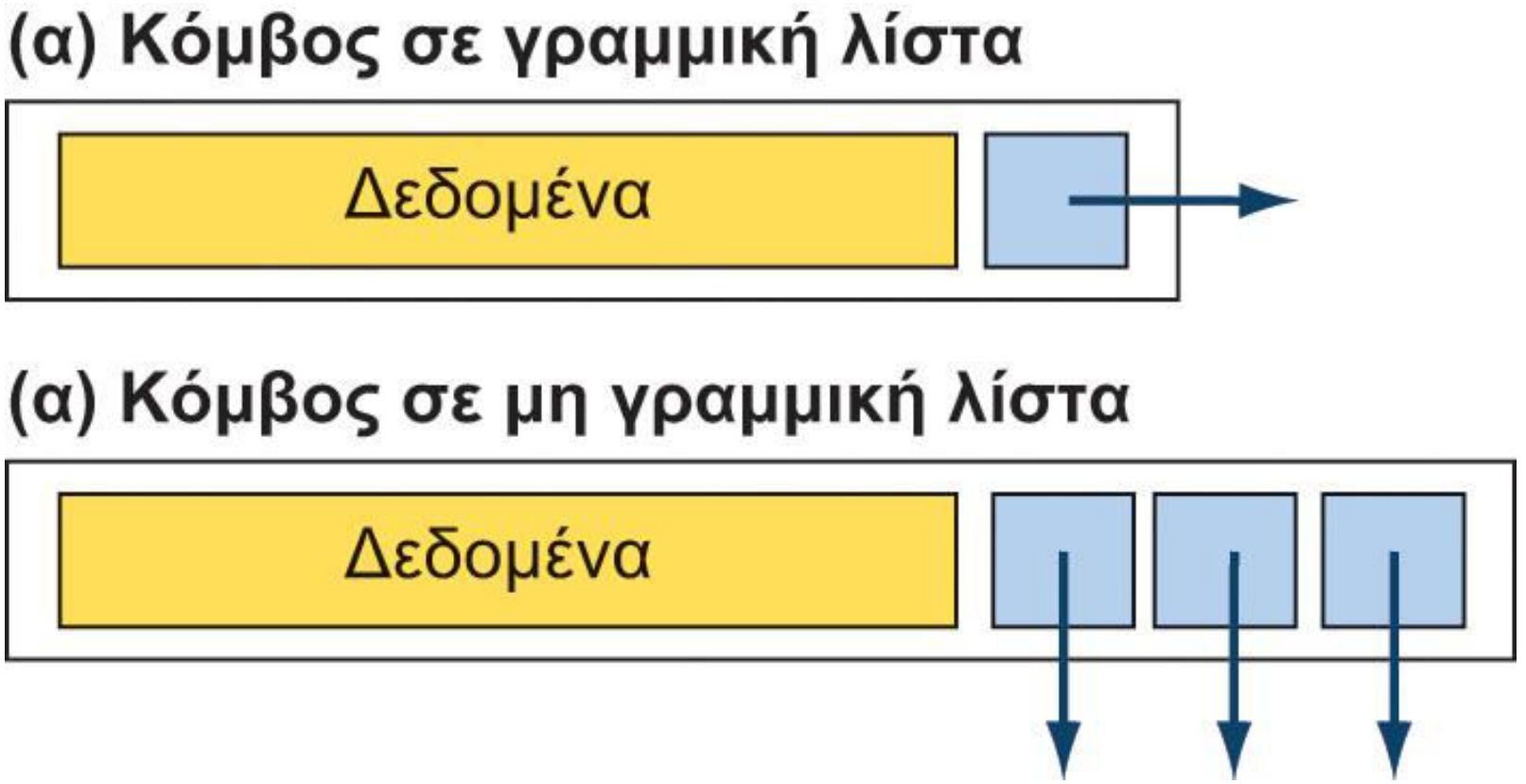
- Στην ορολογία των δομών δεδομένων, τα στοιχεία μιας λίστας αποκαλούνται κόμβοι
  - Ένας κόμβος (node) είναι και ο ίδιος μία δομή, με δύο μέρη: τα καθαυτό δεδομένα και ένα ή περισσότερα πεδία διασύνδεσης

Υλοποίηση ως διασυνδεδεμένη λίστα (3/6)



ΣΧΗΜΑ 15-2 Υλοποιήσεις δομών λίστας με διασυνδεδεμένες λίστες.

Υλοποίηση ως διασυνδεδεμένη λίστα (4/6)

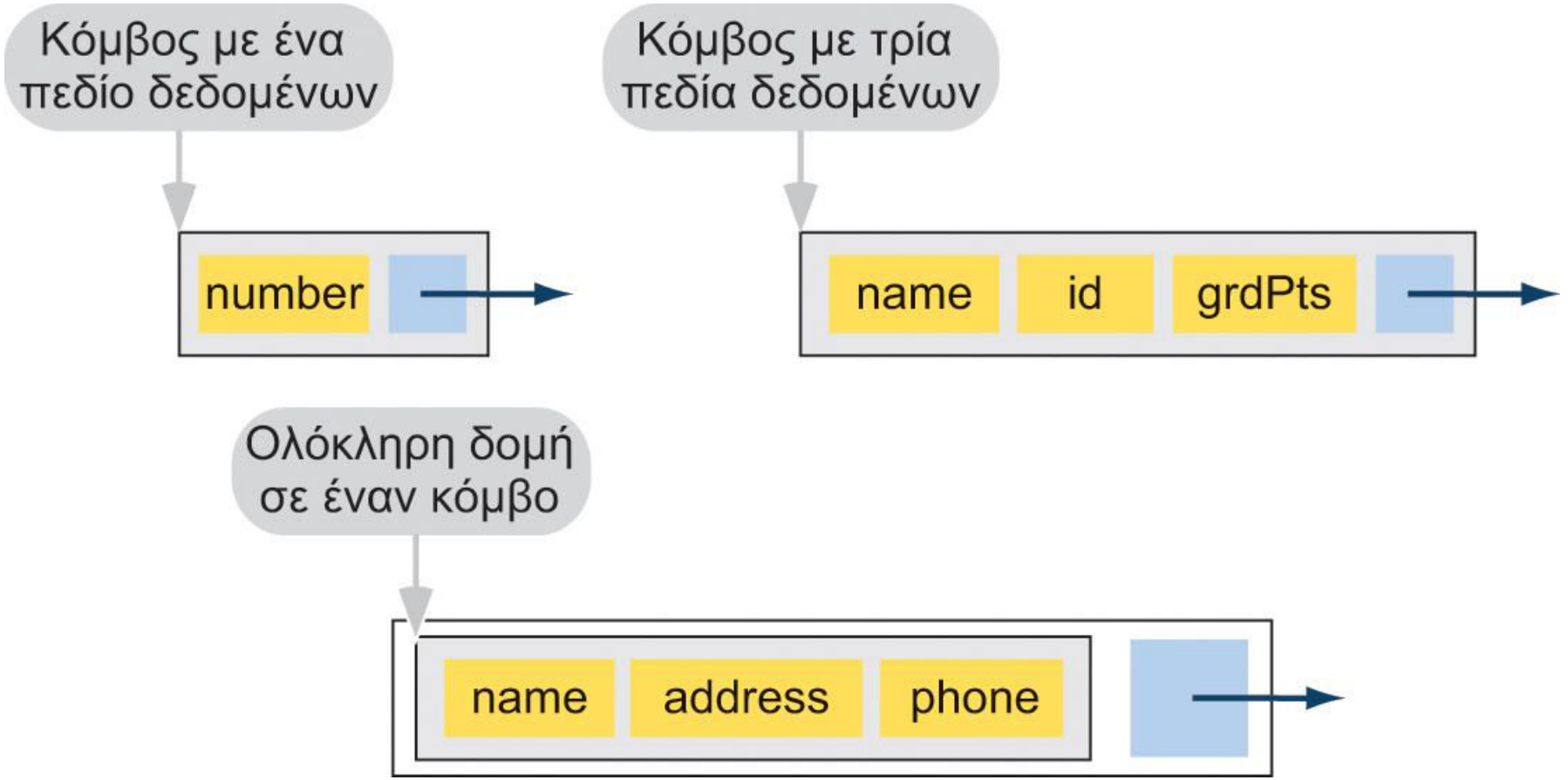


ΣΧΗΜΑ 15-3 Κόμβοι για διαφορετικούς τύπους λιστών.

Υλοποίηση ως διασυνδεδεμένη λίστα (5/6)

- Οι κόμβοι μιας διασυνδεδεμένης λίστας είναι αυτοαναφορικές δομές (self-referential structures)
  - αυτοαναφορική: μία δομή της οποίας κάθε στιγμιότυπο (instance) περιλαμβάνει έναν ή περισσότερους δείκτες προς άλλα στιγμιότυπα του ίδιου τύπου δομής
- Κάθε διασυνδεδεμένη λίστα πρέπει να έχει έναν δείκτη κεφαλής, ο οποίος δείχνει στην αρχή της λίστας
  - Ανάλογα με τον τρόπο χρήσης της, μία λίστα μπορεί να χρησιμοποιεί διάφορους άλλους δείκτες

Υλοποίηση ως διασυνδεδεμένη λίστα (6/6)



ΣΧΗΜΑ 15-4 Πιθανές δομές κόμβων μιας διασυνδεδεμένης λίστας.

15.2
Γραμμικές λίστες γενικής χρήσης

Γραμμικές λίστες γενικής χρήσης

- Ο χαρακτηρισμός «γενικής χρήσης» αναφέρεται σε μία γραμμική λίστα για την οποία όλες οι λειτουργίες επεξεργασίας μπορούν να πραγματοποιούνται σε οποιαδήποτε θέση—στην αρχή, στο μέσον, ή στο τέλος της λίστας
- Για να δουλέψουμε με μία γραμμική λίστα, χρειαζόμαστε κάποιες βασικές λειτουργίες για την επεξεργασία των δεδομένων της, δηλαδή για τον χειρισμό των κόμβων

Εισαγωγή κόμβου (1/6)

- Για να εισάγουμε έναν κόμβο σε μία γραμμική λίστα:

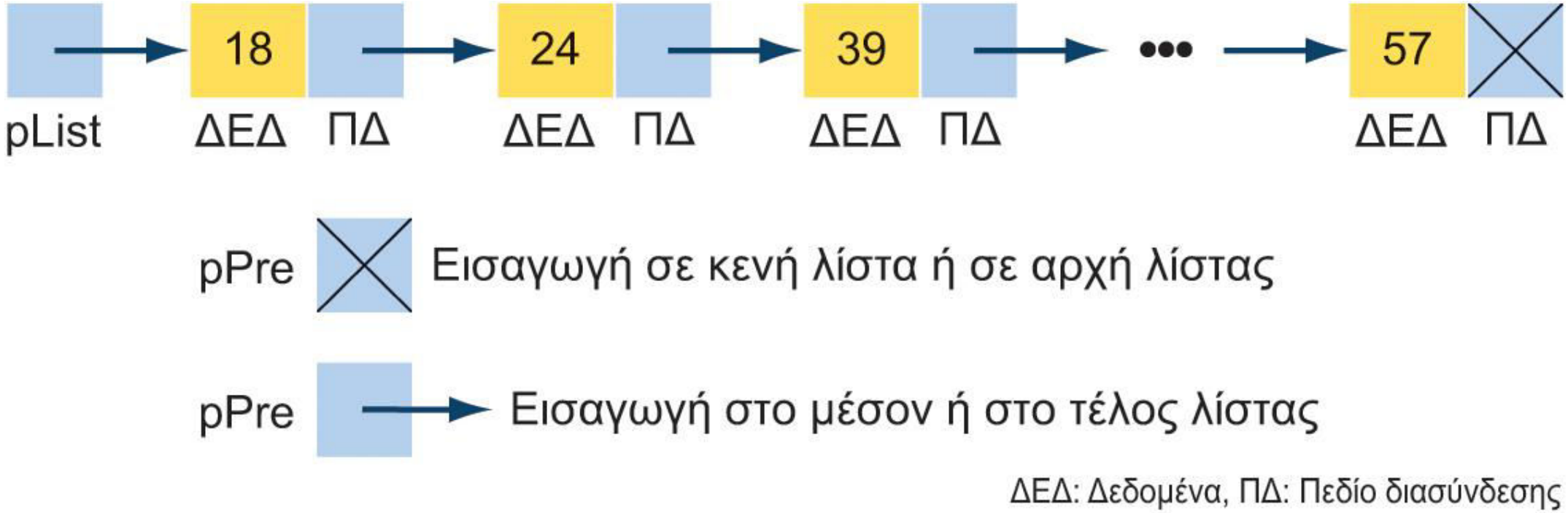
  1. Δεσμεύουμε μνήμη για τον νέο κόμβο

  2. Καθορίζουμε το σημείο εισαγωγής—τη θέση εντός της λίστας όπου θα τοποθετηθούν τα νέα δεδομένα

  3. Θέτουμε το πεδίο διασύνδεσης του νέου κόμβου ώστε να δείχνει στον λογικά επόμενο κόμβο της λίστας

  4. Θέτουμε το πεδίο διασύνδεσης του προηγούμενου κόμβου ώστε να δείχνει στον νέο κόμβο

Εισαγωγή κόμβου (2/6)



ΣΧΗΜΑ 15-5 Εισαγωγή κόμβου σε λίστα: καταστάσεις δεικτών.

Εισαγωγή κόμβου (3/6)

- Όταν ο δείκτης κεφαλής της λίστας (pList) είναι κενός, τότε η λίστα είναι κενή

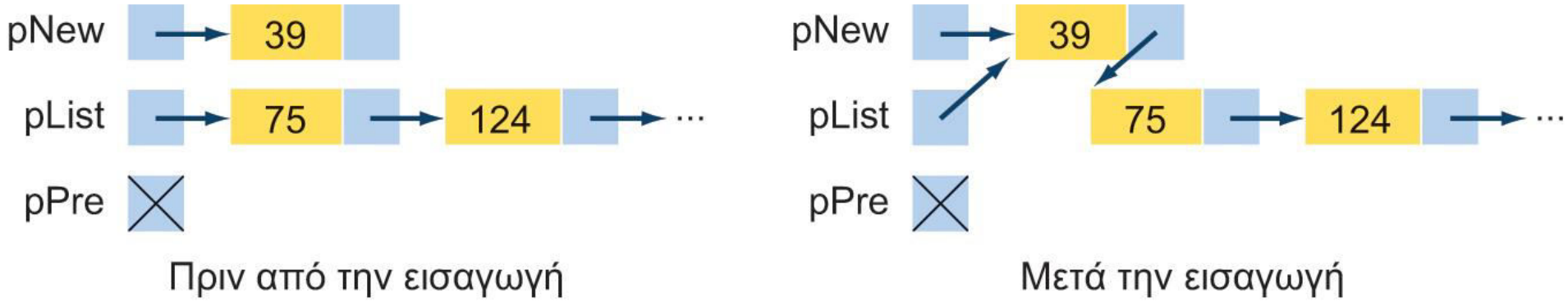
```
pNew->link = pList;
pList      = pNew;
```

ΣΧΗΜΑ 15-6 Εισαγωγή κόμβου σε κενή λίστα.

Εισαγωγή κόμβου (4/6)

- Εισαγωγή στην αρχή της λίστας σημαίνει ότι ο νέος κόμβος πρέπει να τοποθετηθεί πριν από τον υπάρχοντα πρώτο κόμβο της λίστας

```
pNew->link = pList;
pList      = pNew;
```

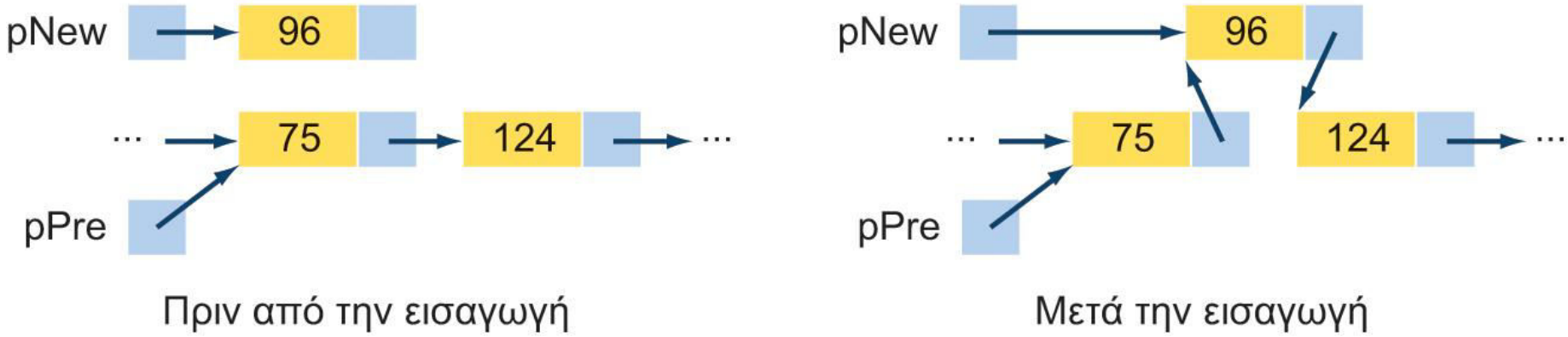


ΣΧΗΜΑ 15-7 Εισαγωγή κόμβου στην αρχή λίστας.

Εισαγωγή κόμβου (5/6)

- Όταν εισάγουμε έναν νέο κόμβο οπουδήποτε στο εσωτερικό μιας λίστας, ο δείκτης στο πεδίο διασύνδεσης του προηγούμενου κόμβου (pPre) δεν είναι null

```
pNew->link = pPre->link;
pPre->link = pNew;
```

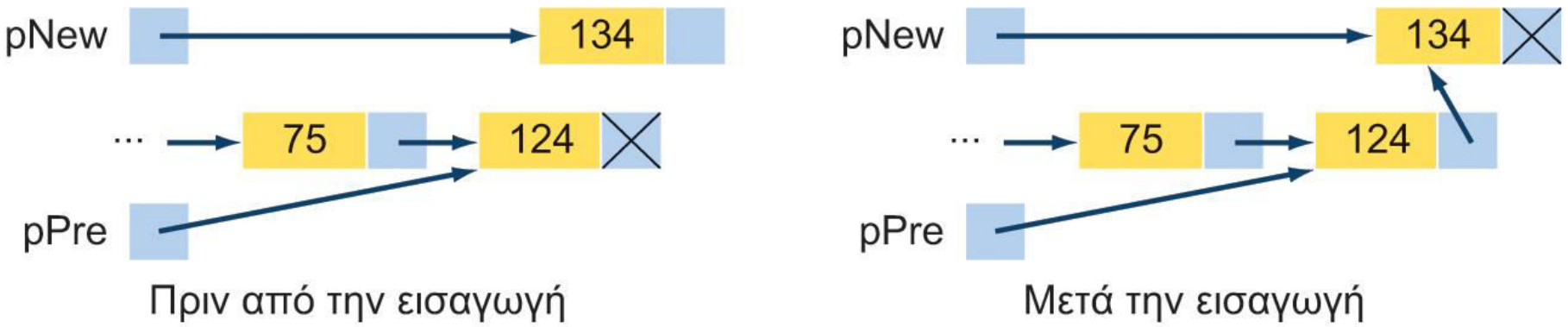


ΣΧΗΜΑ 15-8 Εισαγωγή κόμβου στο μέσον λίστας.

Εισαγωγή κόμβου (6/6)

- Για να εισάγουμε έναν νέο κόμβο στο τέλος της λίστας, χρειάζεται μόνο να προσαρμόσουμε τον δείκτη του προηγούμενου τελικού κόμβου ώστε να δείχνει στον νέο κόμβο

```
pNew->link = pPre->link;
pPre->link = pNew;
```

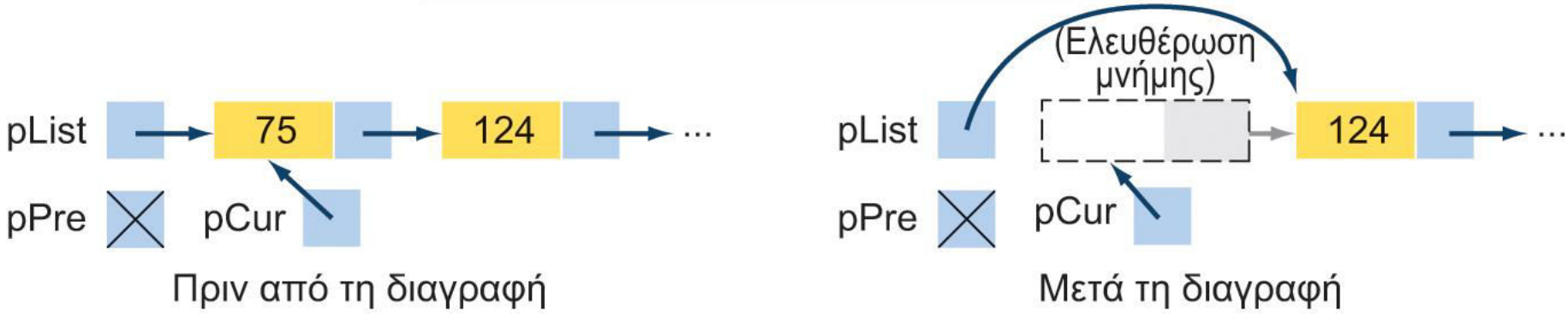


ΣΧΗΜΑ 15-9 Εισαγωγή κόμβου στο τέλος λίστας.

Διαγραφή κόμβου (1/2)

- Απομακρύνουμε «λογικά» τον κόμβο από τη δομή της λίστας προσαρμόζοντας τους δείκτες των κατάλληλων πεδίων διασύνδεσης
- Απομακρύνουμε «φυσικά» τον κόμβο από τη μνήμη του προγράμματος
  - Οι περιπτώσεις διαγραφής είναι ανάλογες με αυτές της εισαγωγής: μπορούμε να διαγράψουμε τον πρώτο, τον τελευταίο, ή οποιονδήποτε εσωτερικό κόμβο μιας λίστας

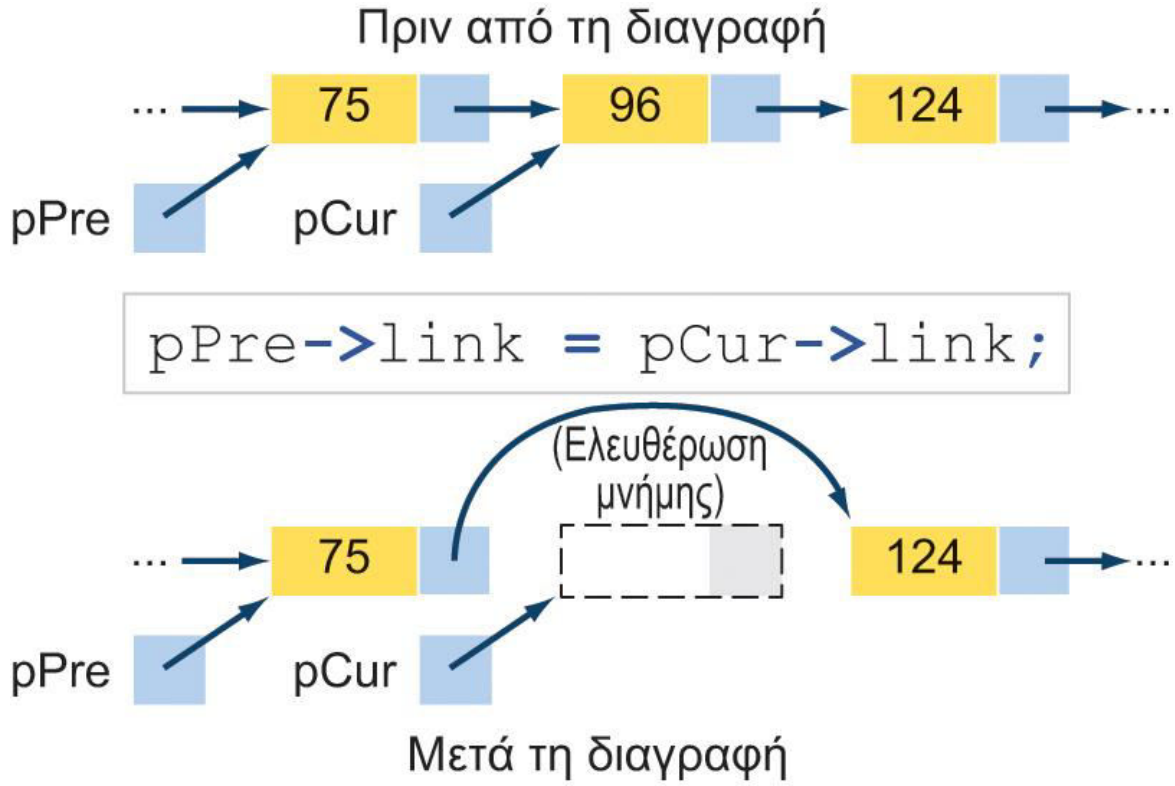
```
pList = pCur->link;
```



ΣΧΗΜΑ 15-10
Διαγραφή πρώτου κόμβου.

Διαγραφή κόμβου (2/2)

Η διαγραφή οποιουδήποτε κόμβου μιας λίστας εκτός του πρώτου εντάσσεται στη γενική περίπτωση διαγραφής, διότι η ίδια λογική χειρίζεται και τις διαγραφές κόμβων στο εσωτερικό και στο τέλος της λίστας



ΣΧΗΜΑ 15-11 Διαγραφή κόμβου από λίστα—γενική περίπτωση.

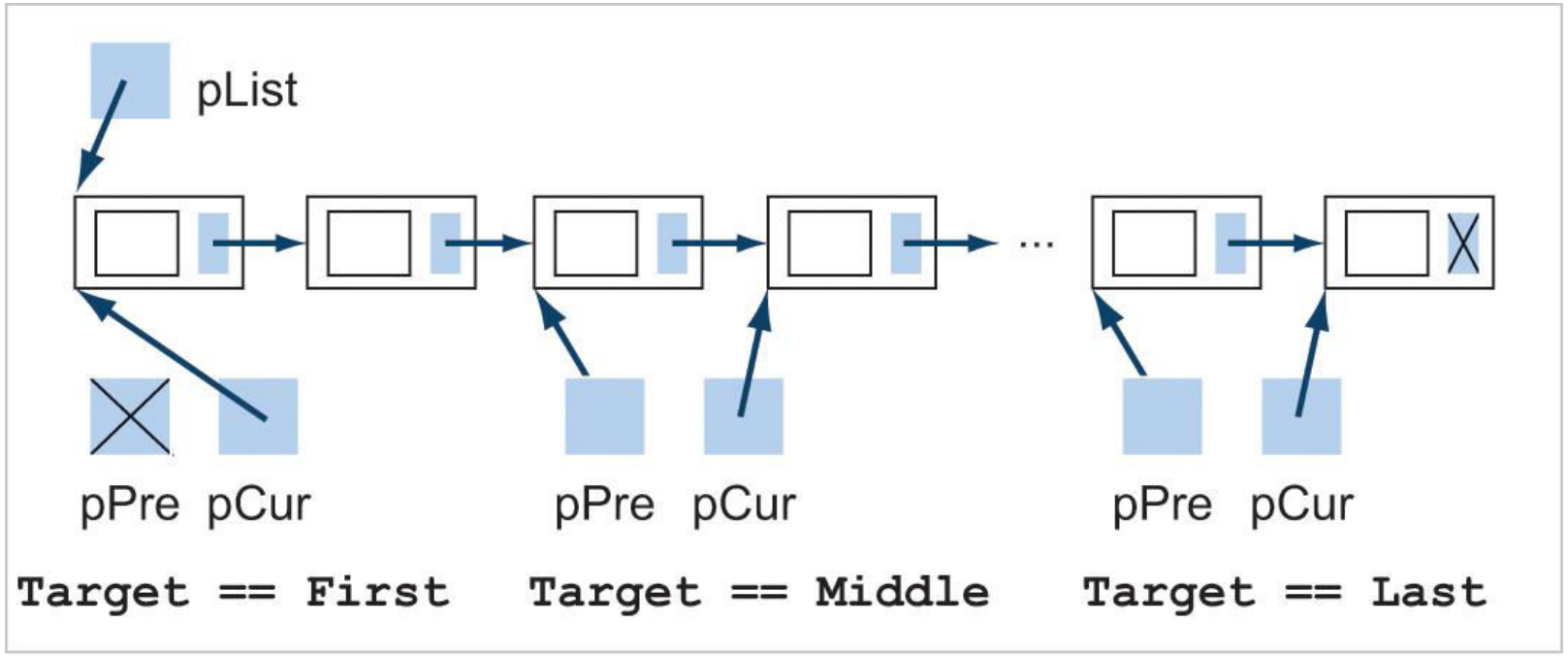
Εντοπισμός δεδομένων σε γραμμικές λίστες

- Οι ενέργειες εισαγωγής και διαγραφής δεδομένων σε μία γραμμική λίστα προϋποθέτουν δυνατότητα διεξαγωγής αναζητήσεων σε αυτήν
  - Για να εισάγουμε ένα νέο κόμβο, πρέπει να εντοπίσουμε τον λογικό προηγούμενό του στη λίστα
  - Για να διαγράψουμε έναν κόμβο, πρέπει να γνωρίζουμε τη θέση του κόμβου που πρόκειται να διαγραφεί και τον λογικό προηγούμενό του

Εντοπισμός δεδομένων σε γραμμικές λίστες

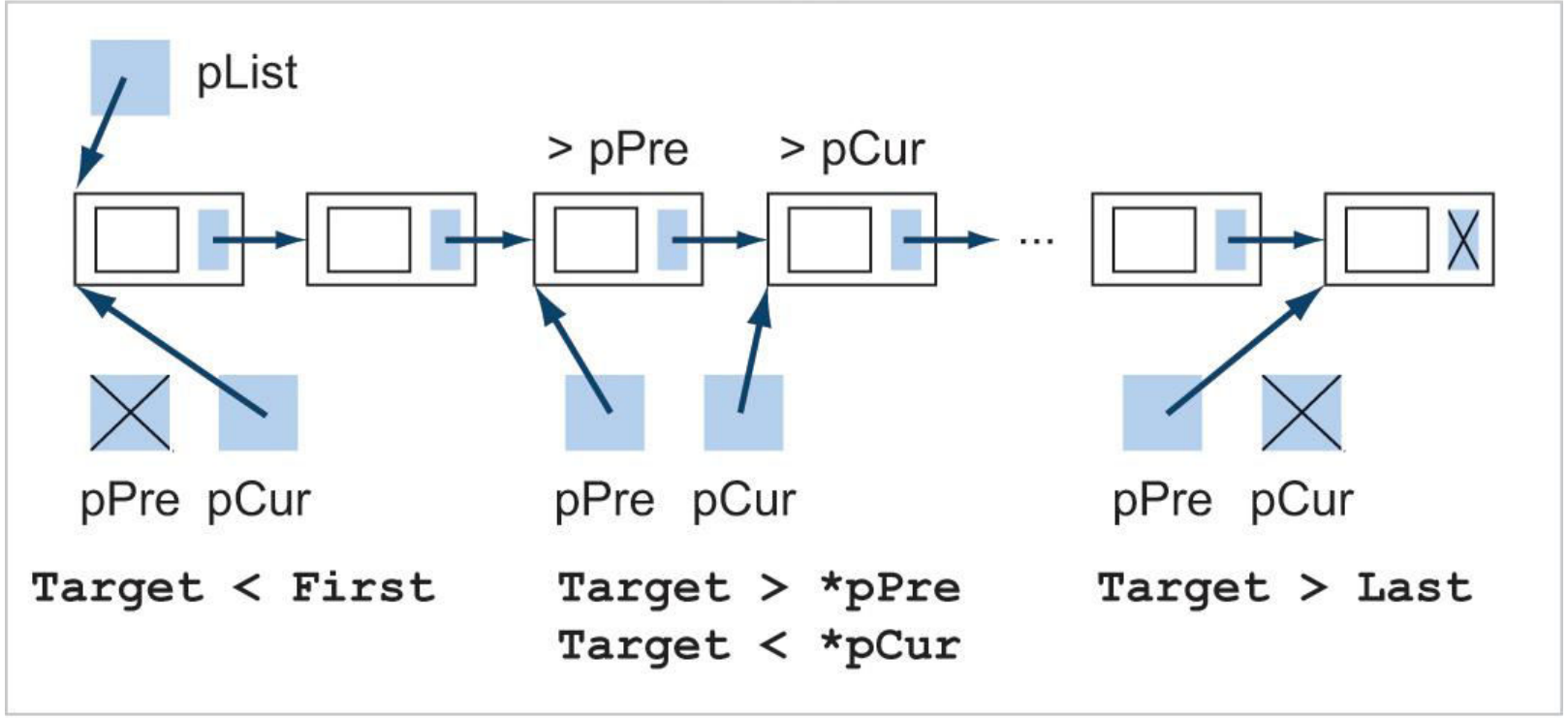
- Όταν χρειάζεται να επεξεργαστούμε έναν κόμβο, π.χ., για υπολογισμό ενός συνόλου ή για την εκτύπωση των περιεχομένων του, πρέπει να γνωρίζουμε τη θέση του
  - Γενικά, τα δεδομένα των λιστών διατηρούνται οργανωμένα με βάση κάποιο κλειδί
  - Ο όρος κλειδί (key) αναφέρεται σε ένα πεδίο της δομής δεδομένων (λίστας), το οποίο μπορεί να χρησιμοποιείται για την ταυτοποίηση των δεδομένων

Εντοπισμός δεδομένων σε γραμμικές λίστες



Επιτυχείς αναζητήσεις (επιστρέφουν `true`)

Εντοπισμός δεδομένων σε γραμμικές λίστες



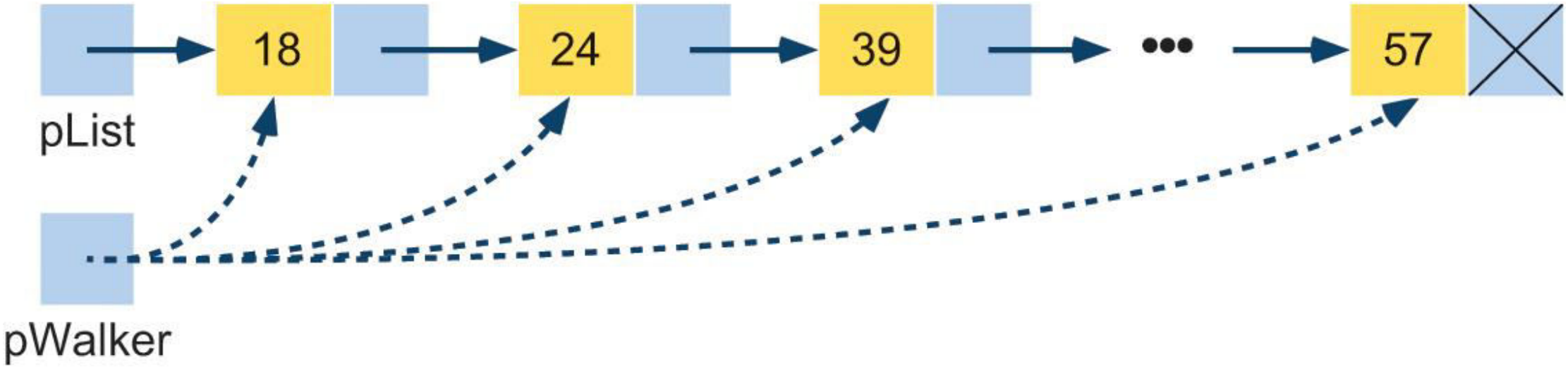
Ανεπιτυχείς αναζητήσεις (επιστρέφουν `false`)

ΠΙΝΑΚΑΣ 15-1 Αποτελέσματα αναζήτησης σε γραμμική λίστα

ΣΥΝΘΗΚΗ	pPre	pCur	ΕΠΙΣΤΡΕΦΕΙ
target < πρώτος κόμβος	NULL	πρώτος κόμβος	0
target == πρώτος κόμβος	NULL	πρώτος κόμβος	1
πρώτος < target < τελευταίος	ο κόμβος με το μεγαλύτερο κλειδί που είναι μικρότερο του target	ο κόμβος με το αμέσως μεγαλύτερο κλειδί από την τιμή του target	0
target == εσωτερικός κόμβος	προηγούμενος κόμβου	ανευρεθείς κόμβος	1
target == τελευταίος κόμβος	προηγούμενος τελευταίου	τελευταίος κόμβος	1
target > τελευταίος κόμβος	τελευταίος κόμβος	NULL	1

Διάσχιση γραμμικών λιστών

- Οι αλγόριθμοι που διατρέχουν μία λίστα ξεκινούν από τον πρώτο κόμβο και εξετάζουν κάθε κόμβο διαδοχικά, μέχρι τον τελευταίο
  - Οποιαδήποτε εφαρμογή απαιτεί την επεξεργασία όλων των στοιχείων μιας λίστας χρησιμοποιεί λογική διάσχισης



ΣΧΗΜΑ 15-13 Διάσχιση γραμμικής λίστας.

Κατασκευή γραμμικής λίστας

Προϋποθέτει τις εξής στοιχειώδεις ενέργειες:

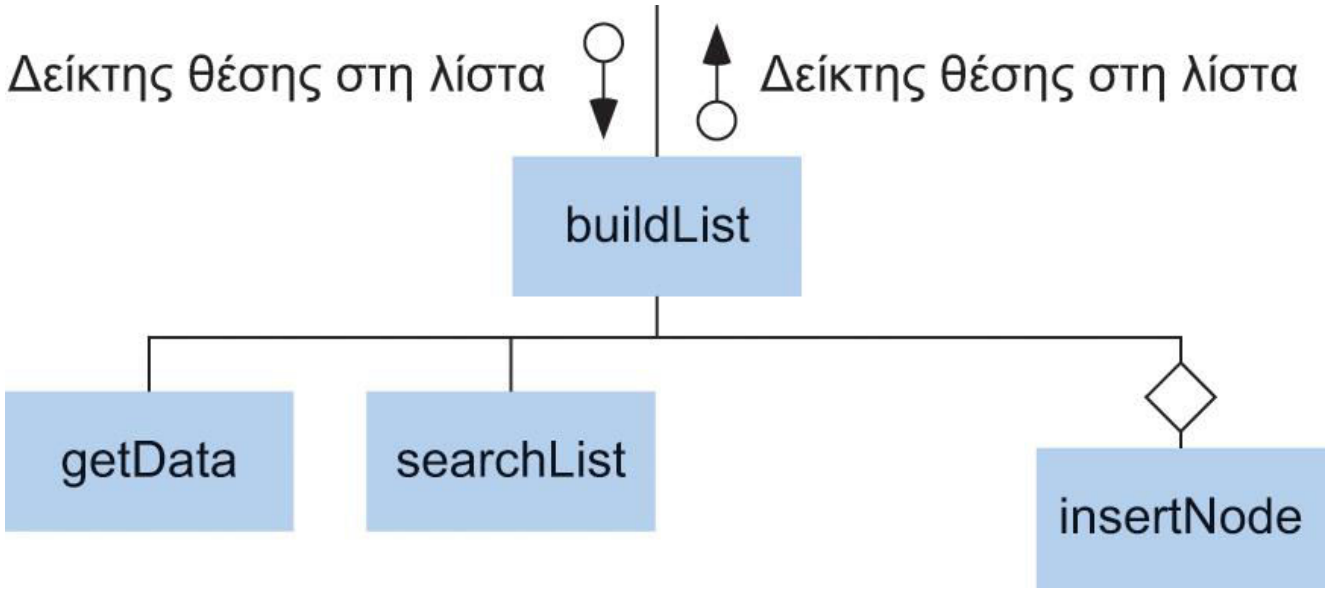
λήψη δεδομένων

δημιουργία κόμβου

προσδιορισμός θέσης εισαγωγής

εισαγωγή κόμβου στη λίστα

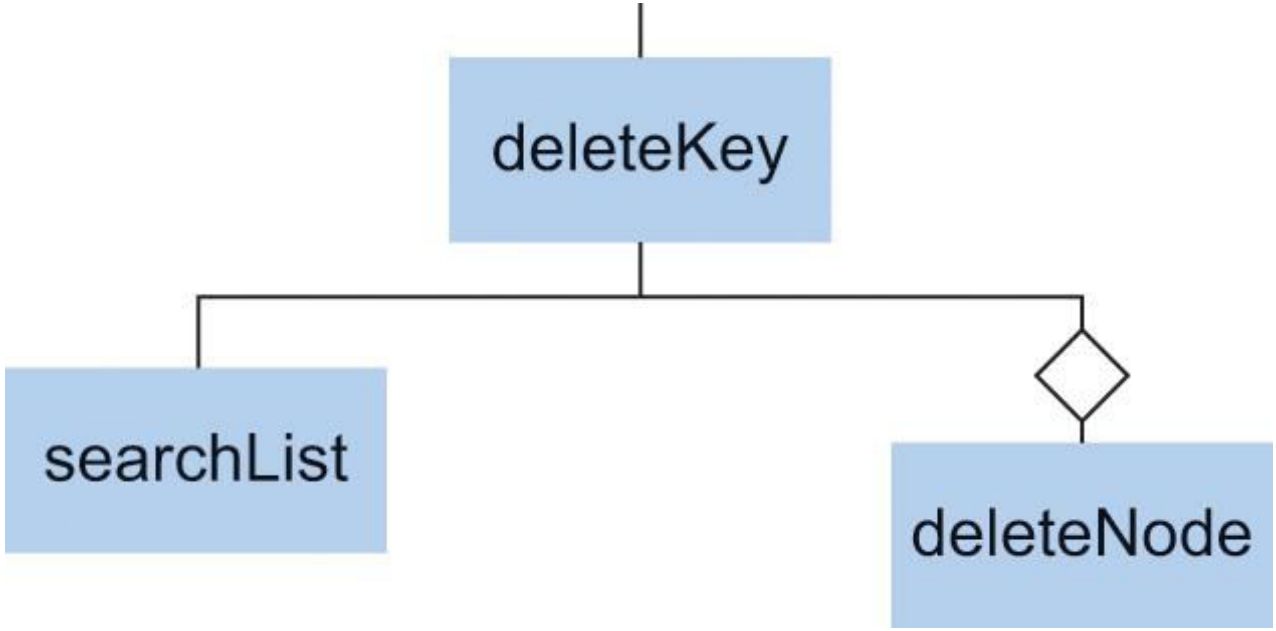
χρησιμοποιείται αναζήτηση για να προσδιοριστεί η θέση εισαγωγής κόμβων σε μία διατεταγμένη βάσει κλειδιού λίστα



ΣΧΗΜΑ 15-14 Βασικά βήματα κατασκευής λίστας.

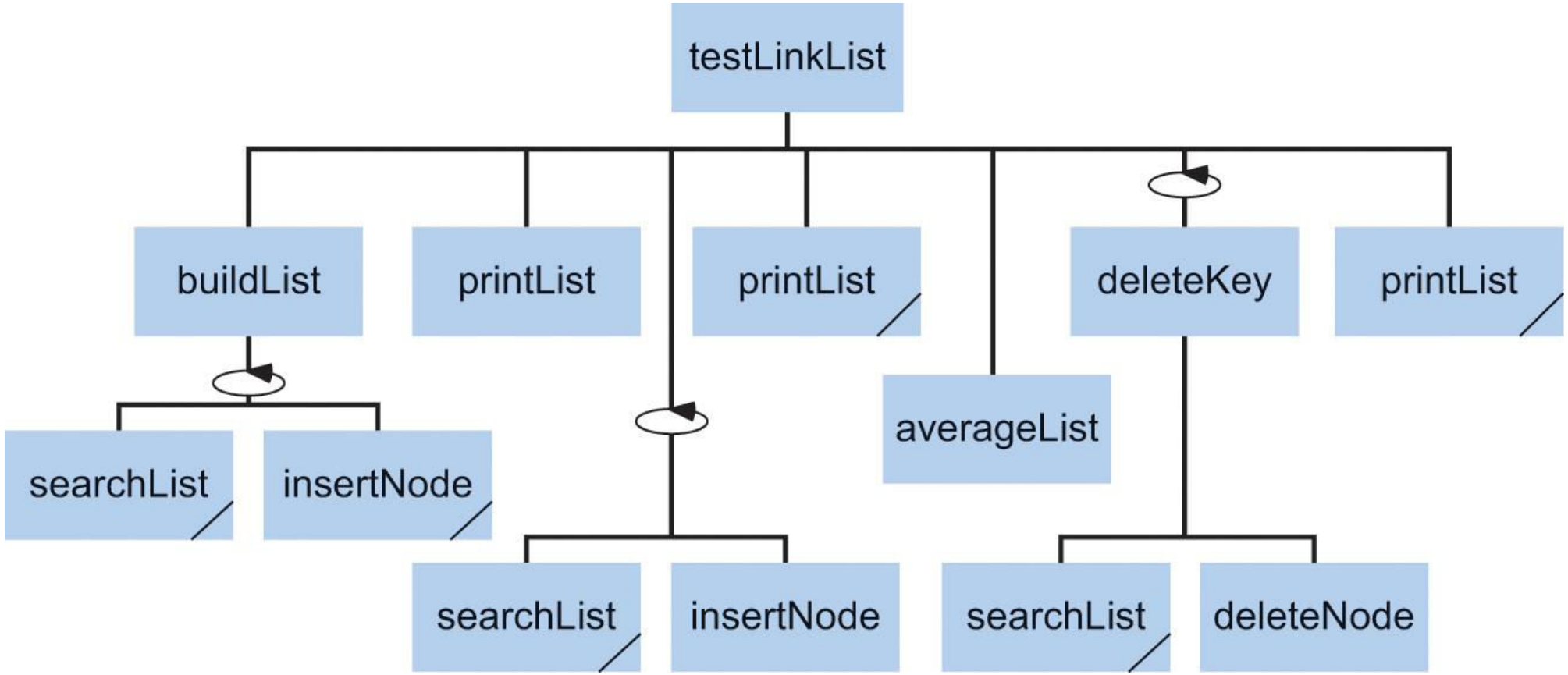
Διαγραφή κόμβου από γραμμική λίστα

Αφού λάβουμε το κλειδί του προς διαγραφή κόμβου, εντοπίζουμε τη θέση του και τον λογικό προηγούμενό του και στη συνέχεια καλούμε τη συνάρτηση deleteNode για τη φυσική απομάκρυνση του κόμβου



ΣΧΗΜΑ 15-15 Σχεδίαση για τη διαγραφή κόμβου.

Πρόγραμμα δοκιμής για τις συναρτήσεις χειρισμού γραμμικής λίστας

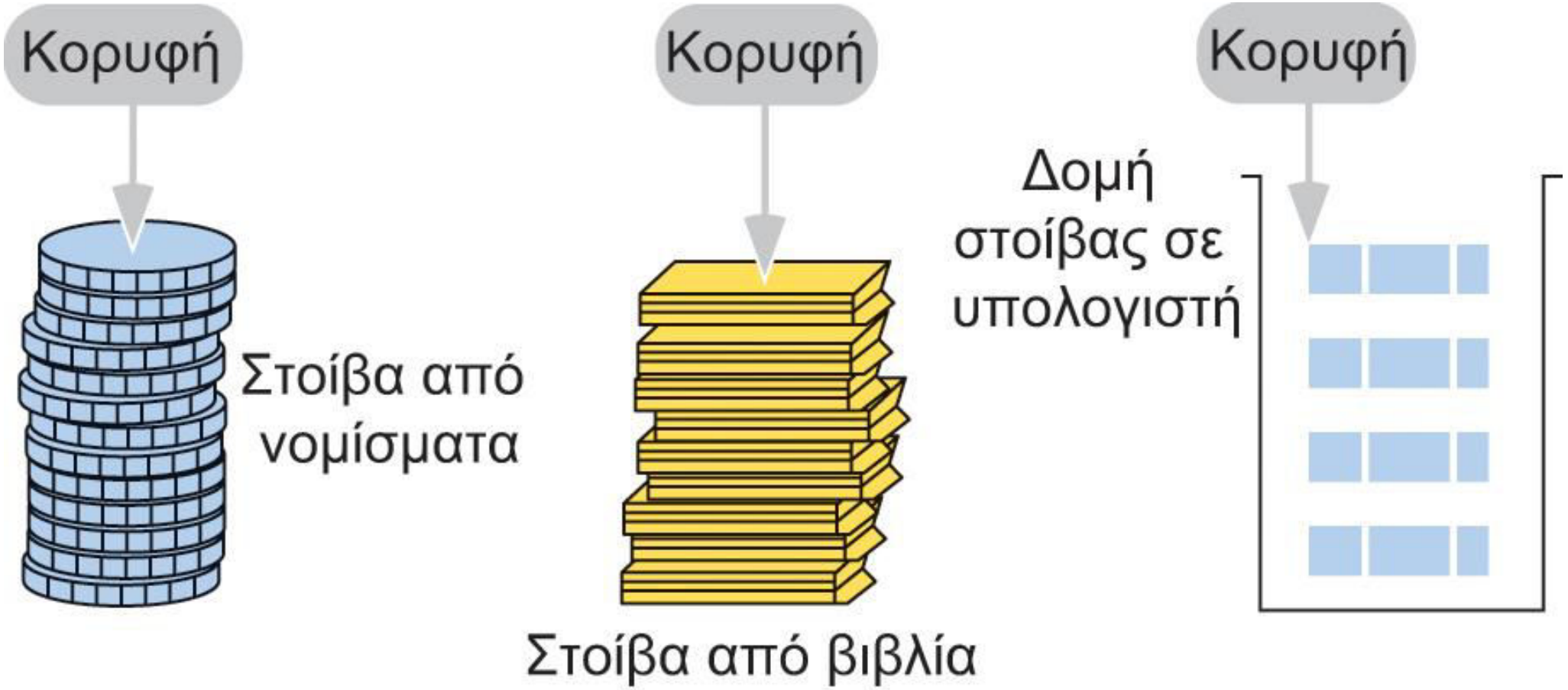


15.3
Στοίβες

Στοίβες

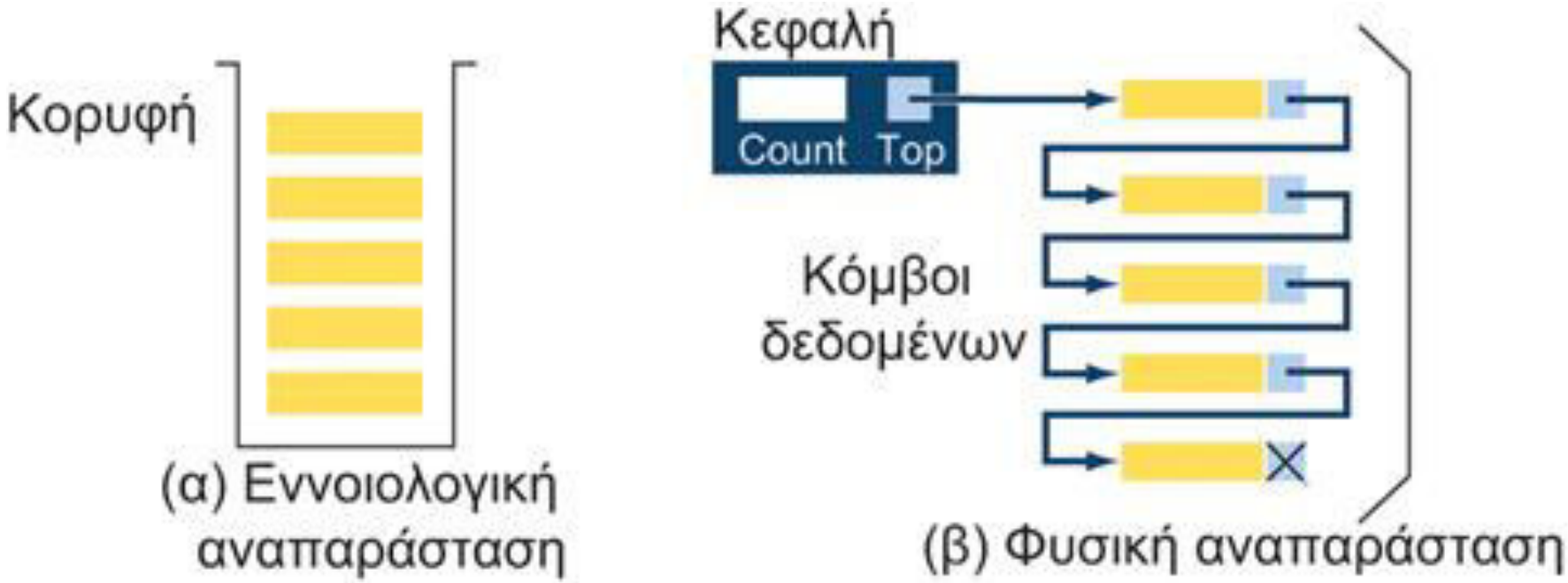
- Στοίβα (stack) είναι μία δομή γραμμικής λίστας, για την οποία ισχύει ο εξής περιορισμός: όλες οι εισαγωγές και διαγραφές στοιχείων μπορούν να γίνονται μόνο από το ένα άκρο της λίστας, το οποίο αναφέρεται ως κορυφή της στοίβας
- Οι στοίβες είναι δομές δεδομένων LIFO
  - last in-first out: τελευταίο μπαίνει, πρώτο βγαίνει
  - μία ακολουθία δεδομένων που εισάγεται σε μία στοίβα εξάγεται από αυτήν με αντίστροφη σειρά

Στοίβες



ΣΧΗΜΑ 15-17 Παραδείγματα δομών στοίβας.

Στοίβες



ΣΧΗΜΑ 15-18 Εννοιολογική και φυσική υλοποίηση στοίβας.

Δομές στοίβας

- Για υλοποίηση μιας στοίβας υπό μορφή διασυνδεδεμένης λίστας, χρειαζόμαστε δύο διαφορετικές δομές:
  - μία για την «κεφαλή» της στοίβας (κόμβος κεφαλής)
  - μία για τα καθαυτό δεδομένα (κόμβος δεδομένων)

Δομές στοίβας

ΣΧΗΜΑ 15-19 Δομές δεδομένων για την υλοποίηση της στοίβας.

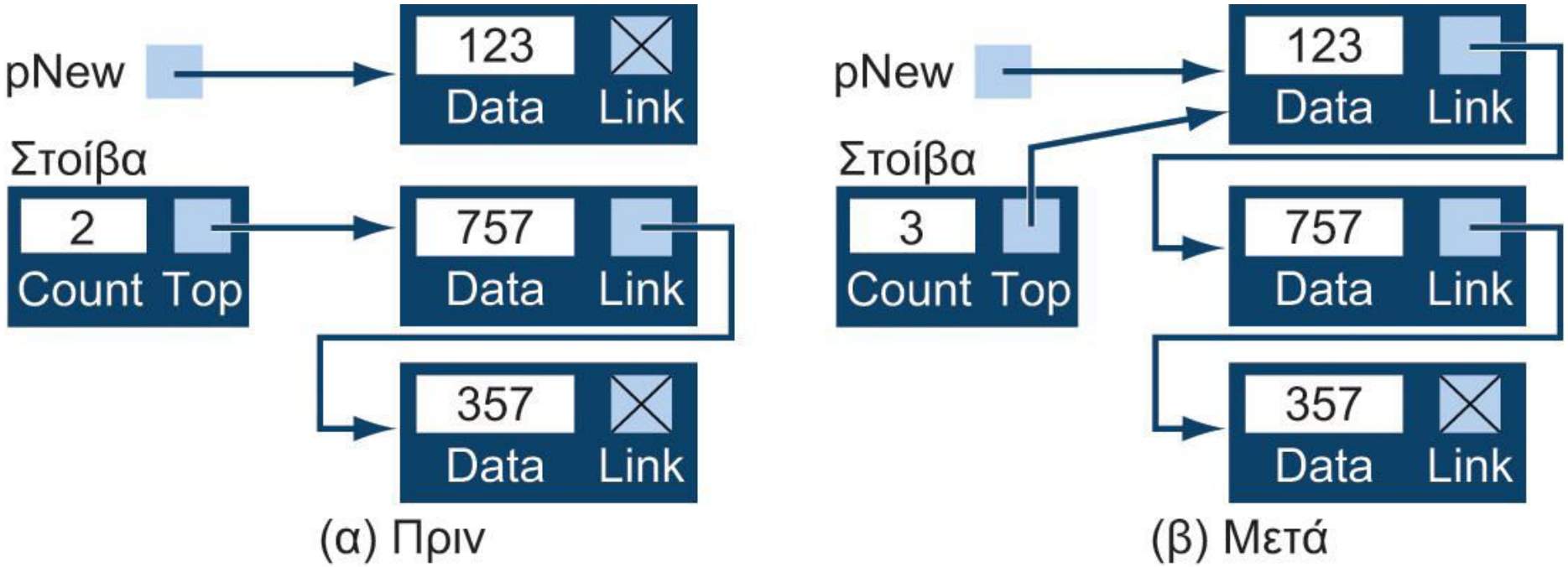
Δομή κεφαλής στοίβας

Δομή κόμβου στοίβας

```
typedef struct
{
    int count;
    struct node* top;
} STACK;

typedef struct node
{
    int data;
    struct node* link;
} STACK_NODE;
```

Αλγόριθμοι χειρισμού στοίβας

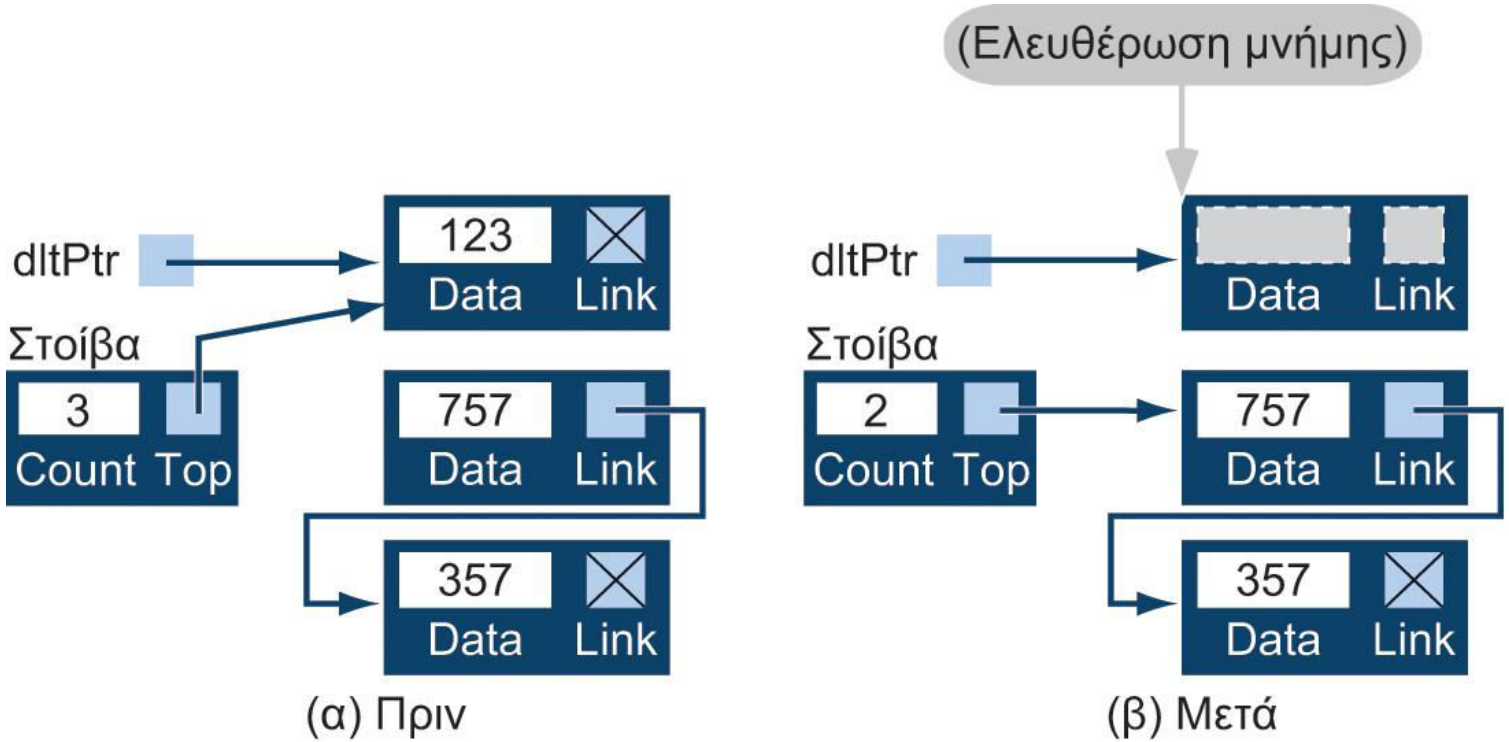
- push (ώθηση): εισαγωγή ενός κόμβου δεδομένων σε στοίβα



ΣΧΗΜΑ 15-20 Εισαγωγή σε στοίβα.

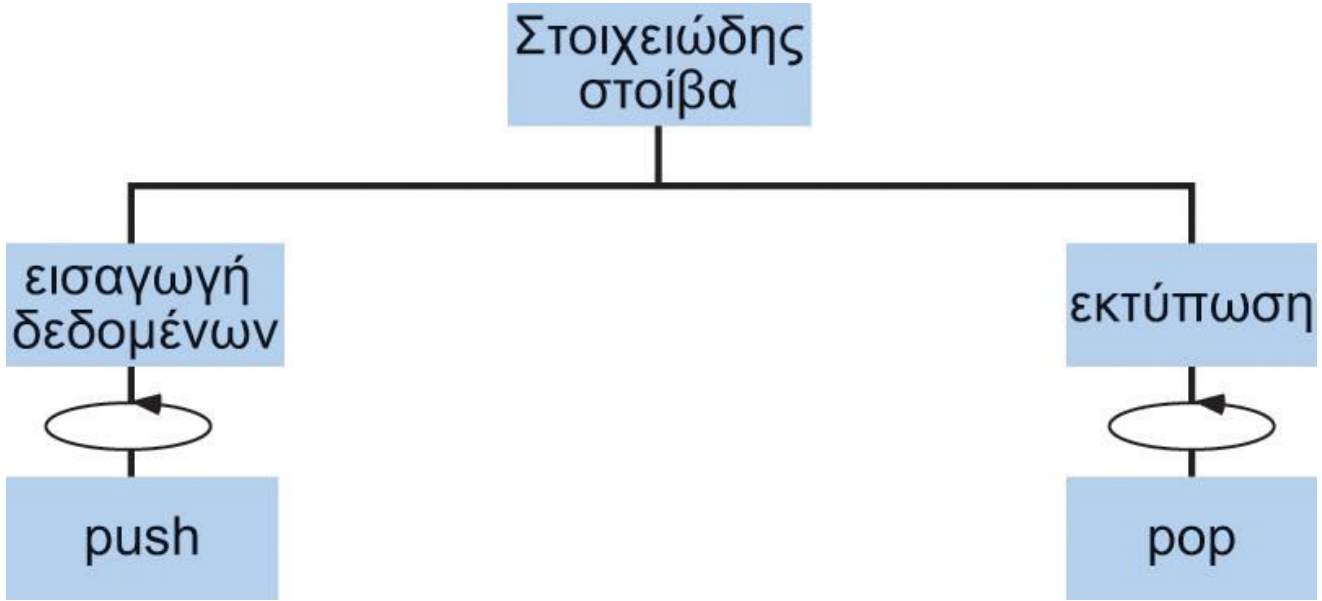
Αλγόριθμοι χειρισμού στοίβας

- pop: εξάγει τον τελευταίο κόμβο που έχει εισαχθεί σε μία στοίβα και επιστρέφει τα δεδομένα του στον κώδικα που την καλεί



ΣΧΗΜΑ 15-21 Εξαγωγή κόμβου από στοίβα.

Επίδειξη λειτουργιών στοίβας



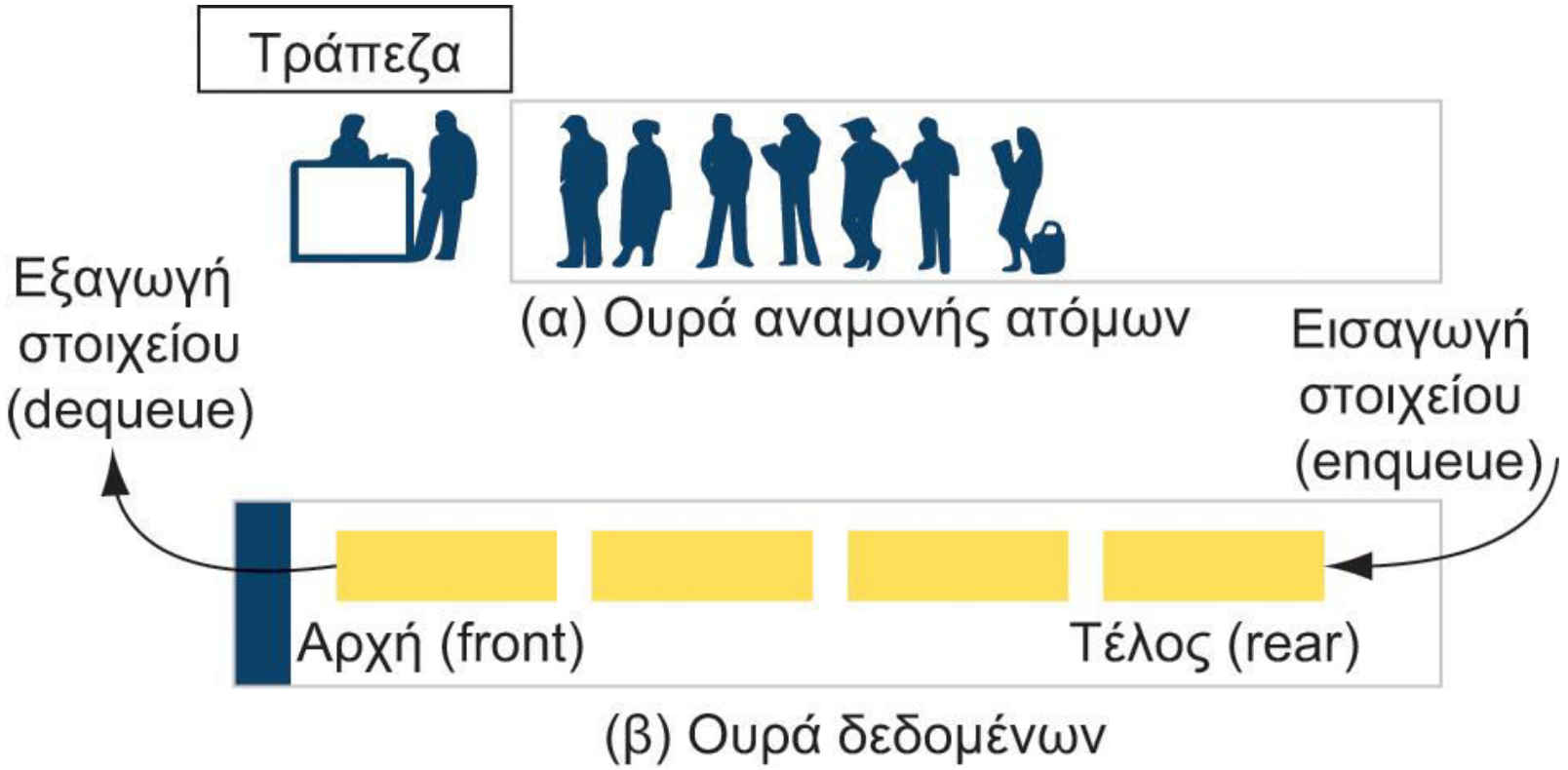
ΣΧΗΜΑ 15-22 Σχεδίαση για το πρόγραμμα επίδειξης βασικών λειτουργιών σε στοίβες.

15.4
Ουρές

Ουρές

- Ουρά (queue) είναι μία γραμμική δομή δεδομένων για την οποία ισχύουν οι εξής περιορισμοί:
  - η εισαγωγή δεδομένων γίνεται μόνο από το ένα άκρο (τέλος ουράς)
  - η εξαγωγή δεδομένων γίνεται μόνο από το άλλο άκρο (αρχή ουράς)

Ουρές

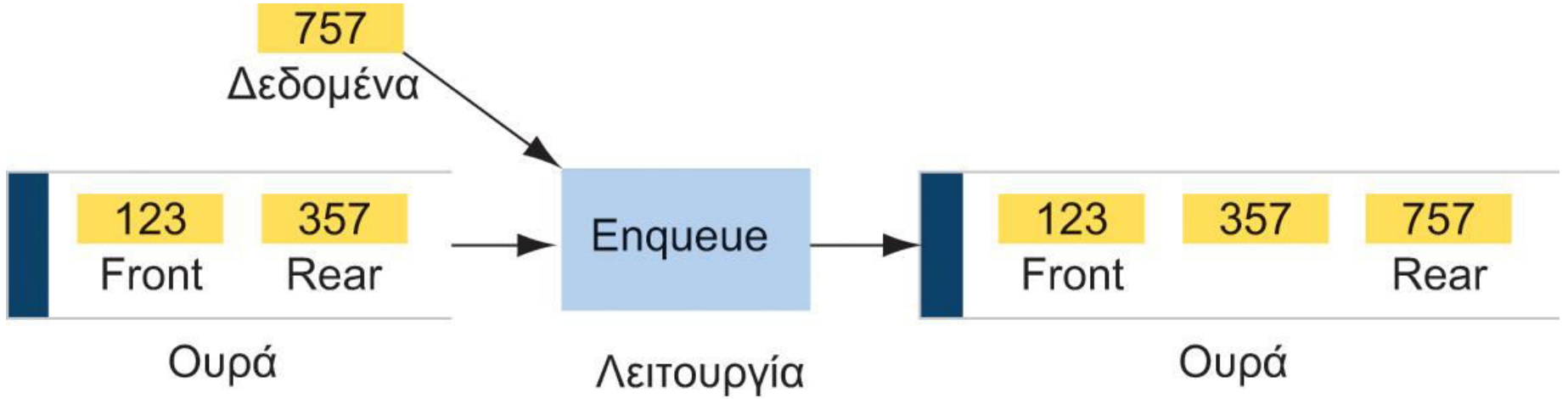


ΣΧΗΜΑ 15-23 Σχηματικές αναπαραστάσεις δύο τύπων ουράς.

Βασικές λειτουργίες σε ουρές

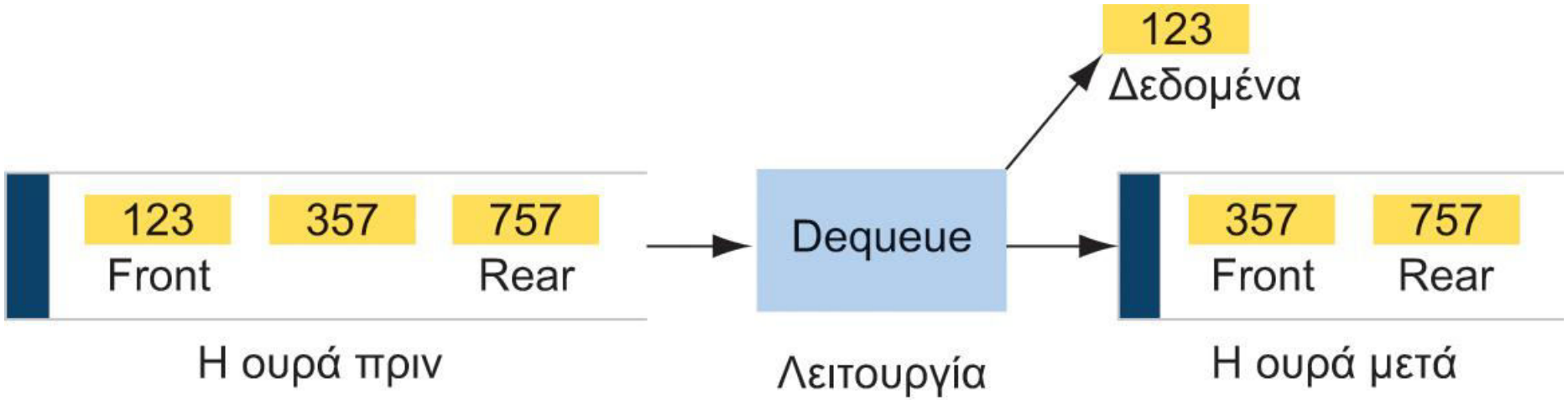
- Οι δύο βασικές λειτουργίες για μία ουρά αφορούν την εισαγωγή και την εξαγωγή (διαγραφή) δεδομένων·
  - Η λειτουργία enqueue χρησιμοποιείται για την εισαγωγή δεδομένων σε μία ουρά
  - Η λειτουργία dequeue χρησιμοποιείται για την εξαγωγή δεδομένων από μία ουρά
  - Τα δεδομένα μπορούν να εισάγονται μόνο στο τέλος και να εξάγονται μόνο από την αρχή της ουράς

Βασικές λειτουργίες σε ουρές



ΣΧΗΜΑ 15-24 Εισαγωγή δεδομένων σε ουρά.

Βασικές λειτουργίες σε ουρές

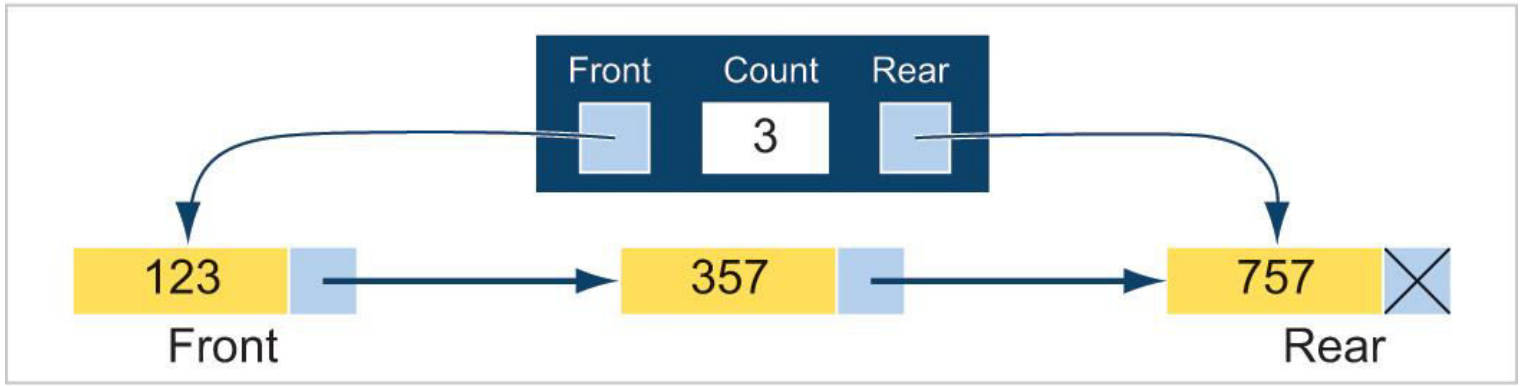


ΣΧΗΜΑ 15-25 Εξαγωγή δεδομένων από ουρά.

Υλοποίηση ουράς ως διασυνδεδεμένη λίστα

- Για την υλοποίηση μιας ουράς χρειαζόμαστε δύο διαφορετικές δομές:
  - Μία για τον κόμβο «κεφαλής» της ουράς
  - Μία για τους κόμβους δεδομένων

(α) Εννοιολογική αναπαράσταση



(β) Φυσική αναπαράσταση

Υλοποίηση ουράς ως διασυνδεδεμένη λίστα

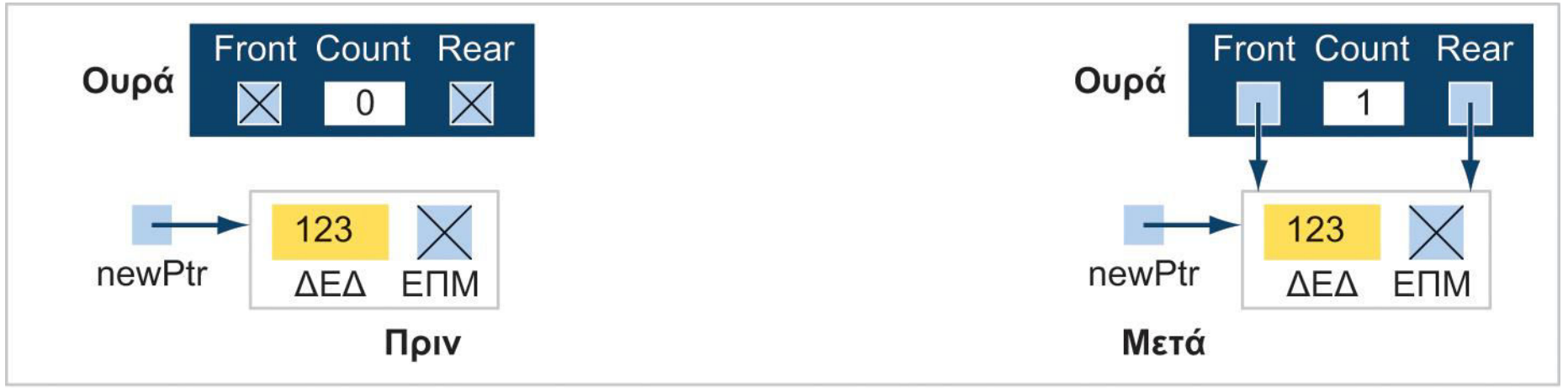
Δομή κόμβου

Δομή κεφαλής

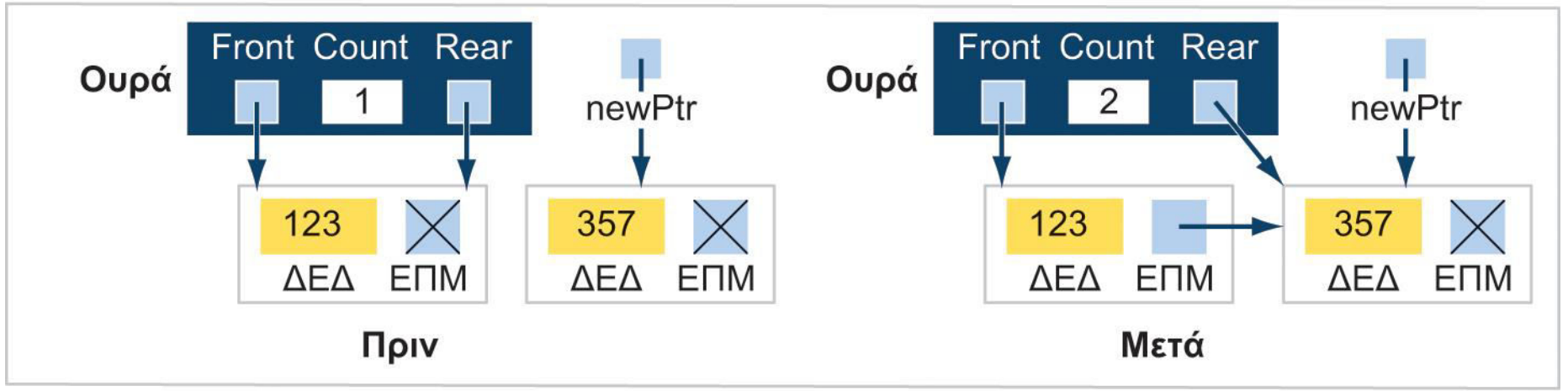
```
typedef struct node
{
    int           data;
    struct node*  next;
} QUEUE_NODE;

typedef struct
{
   QUEUE_NODE*  front;
   int          count;
   QUEUE_NODE*  rear;
} QUEUE;
```

ΣΧΗΜΑ 15-27 Δομή για τον κόμβο κεφαλής της ουράς.

Εισαγωγή δεδομένων σε ουρά



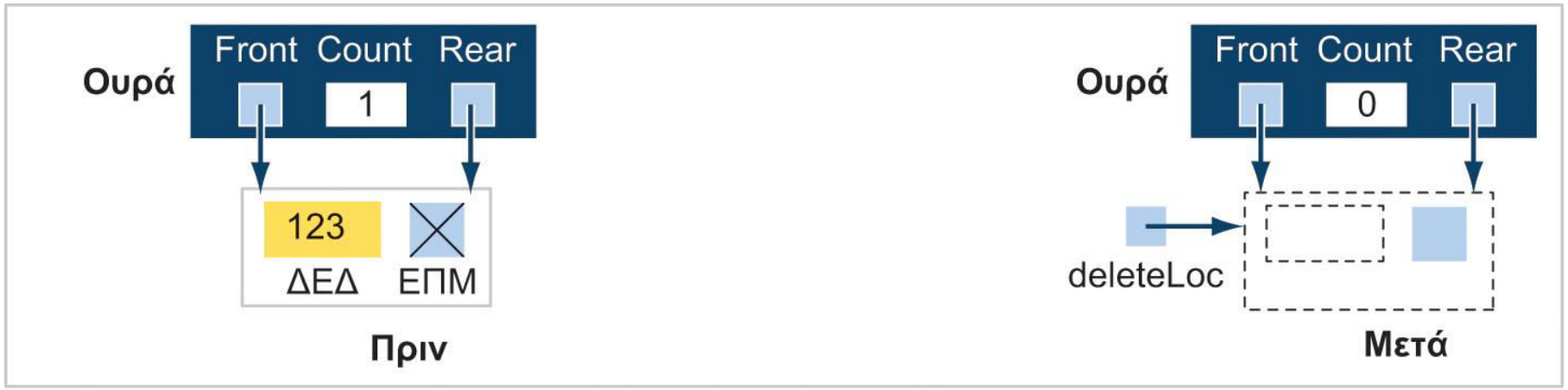
(α) Περίπτωση 1: Εισαγωγή σε κενή ουρά



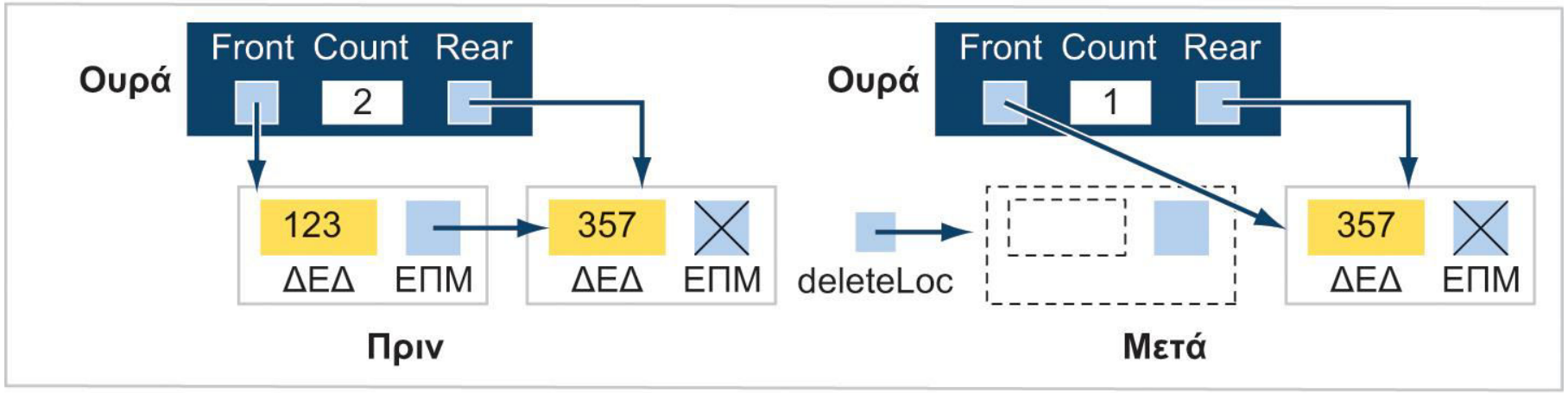
(β) Περίπτωση 2: Εισαγωγή σε ουρά με δεδομένα

(ΔΕΔ: δεδομένα, ΕΠΜ: δείκτης επόμενου)

Εξαγωγή δεδομένων από ουρά



(α) Περίπτωση 1: Διαγραφή μοναδικού στοιχείου ουράς



(β) Περίπτωση 2: Διαγραφή στοιχείου στην αρχή ουράς

(ΔΕΔ: δεδομένα, ΕΠΜ: δείκτης επόμενου)

Σύνοψη

- Οι δομές λίστας διακρίνονται σε γραμμικές και μη γραμμικές λίστες
- Οι γραμμικές λίστες κατηγοριοποιούνται σε λίστες γενικής χρήσης, στοίβες και ουρές
- Οι μη γραμμικές λίστες κατηγοριοποιούνται σε δένδρα και γράφους
- Σε μία γραμμική λίστα γενικής χρήσης, επιτρέπεται εισαγωγή δεδομένων σε οποιαδήποτε θέση και δεν υπάρχουν περιορισμοί ως προς τις λειτουργίες που μπορούν να χρησιμοποιηθούν για την επεξεργασία της λίστας
- Οι τέσσερις βασικές λειτουργίες που χρησιμοποιούνται συνήθως με γραμμικές λίστες γενικής χρήσης: εισαγωγή, διαγραφή, ανάκτηση και διάσχιση
- Διάσχιση μιας γραμμικής λίστας γενικής χρήσης σημαίνει ότι διατρέχουμε τη λίστα, στοιχείο προς στοιχείο, π.χ., για σκοπούς επεξεργασίας

Σύνοψη

- Για εισαγωγή δεδομένων σε μία γραμμική λίστα γενικής χρήσης, πρέπει να εξετάσουμε τέσσερις περιπτώσεις:
  - Εισαγωγή σε κενή λίστα
  - Εισαγωγή στην αρχή της λίστας
  - Εισαγωγή στο μέσον της λίστας
  - Εισαγωγή στο τέλος της λίστας
- Για διαγραφή δεδομένων από μία γραμμική λίστα γενικής χρήσης, πρέπει να εξετάσουμε δύο περιπτώσεις:
  - Διαγραφή του πρώτου κόμβου
  - Διαγραφή οποιουδήποτε άλλου κόμβου

Σύνοψη

- Στοίβα είναι μία γραμμική λίστα στην οποία ισχύει ο περιορισμός ότι όλες οι εισαγωγές δεδομένων γίνονται από το ένα άκρο, το οποίο αποκαλείται κορυφή
  - –push: εισάγει ένα στοιχείο στην κορυφή της στοίβας. Το νέο στοιχείο γίνεται το κορυφαίο της στοίβας
  - –pop: εξάγει το στοιχείο που βρίσκεται στην κορυφή της στοίβας. Το αμέσως επόμενο στοιχείο, εάν υπάρχει,γίνεται το κορυφαίο της στοίβας
- Ουρά είναι μία γραμμική λίστα στην οποία τα δεδομένα εισάγονται από το ένα άκρο (τέλος ουράς) και διαγράφονται από το άλλο άκρο (αρχή ουράς)
  - –enqueue: εισάγει ένα στοιχείο στο τέλος της ουράς
  - –dequeue: εξάγει το στοιχείο που βρίσκεται στην αρχή της ουράς

Σύνοψη

- Ένα δένδρο αποτελείται από ένα πεπερασμένο σύνολο στοιχείων που ονομάζονται κόμβοι και ένα πεπερασμένο σύνολο κατευθυντικών γραμμών που ενώνουν τους κόμβους και ονομάζονται κλάδοι

- Ένας κόμβος σε ένα δένδρο μπορεί να είναι γονέας, παιδί ή και τα δύο. Δύο ή περισσότεροι κόμβοι με τους ίδιους γονείς αποκαλούνται αδέρφια

- Ένα δυαδικό δένδρο είναι ένα δένδρο στο οποίο κανένας κόμβος δεν μπορεί να έχει περισσότερα από δύο παιδιά

- Κατά την προδιατεταγμένη διάσχιση ενός δένδρου, επισκεπτόμαστε πρώτα τη ρίζα, μετά το αριστερό υποδένδρο και κατόπιν το δεξιό υποδένδρο

- Κατά την ενδοδιατεταγμένη διάσχιση ενός δένδρου, επισκεπτόμαστε πρώτα το αριστερό υποδένδρο, μετά τη ρίζα και κατόπιν το δεξιό υποδένδρο

Σύνοψη

- Ένα δυαδικό δένδρο αναζήτησης είναι ένα δυαδικό δένδρο με τις ακόλουθες ιδιότητες:
  - Όλα τα στοιχεία στο αριστερό υποδένδρο έχουν κλειδί μικρότερο από το κλειδί της ρίζας
  - Όλα τα στοιχεία στο δεξιό υποδένδρο έχουν κλειδί μεγαλύτερο ή ίσο με το κλειδί της ρίζας
  - Κάθε υποδένδρο είναι το ίδιο ένα δυαδικό δένδρο αναζήτησης

Σύνοψη

- Ένας γράφος είναι μία συλλογή κόμβων (ονομάζονται κορυφές) σε συνδυασμό με μία συλλογή ακμών ή τόξων που ενώνουν ζεύγη κόμβων.
- Οι γράφοι μπορούν να είναι κατευθυνόμενοι ή μη κατευθυνόμενοι.
- Σε έναν κατευθυνόμενο γράφο, όλες οι ακμές που ενώνουν κόμβους έχουν μία κατεύθυνση και αποκαλούνται τόξα.
- Σε ένα μη κατευθυνόμενο γράφο, οι γραμμές που ενώνουν τους κόμβους δεν έχουν χαρακτηριστικό κατεύθυνσης.
- Υπάρχουν δύο βασικοί τρόποι διάσχισης γράφων: πρώτα κατά βάθος, ή πρώτα κατά πλάτος.