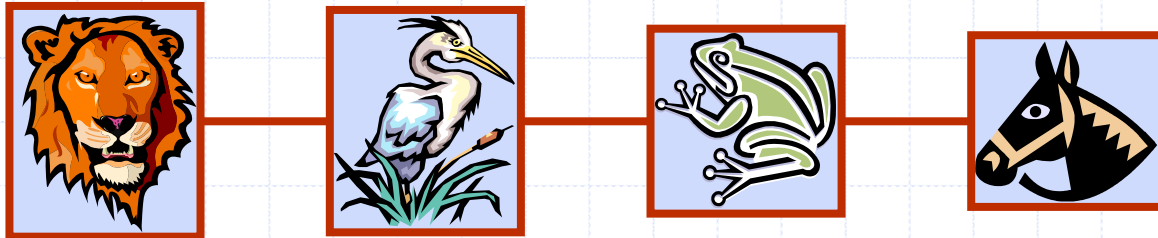


# Λίστες και Ακολουθίες

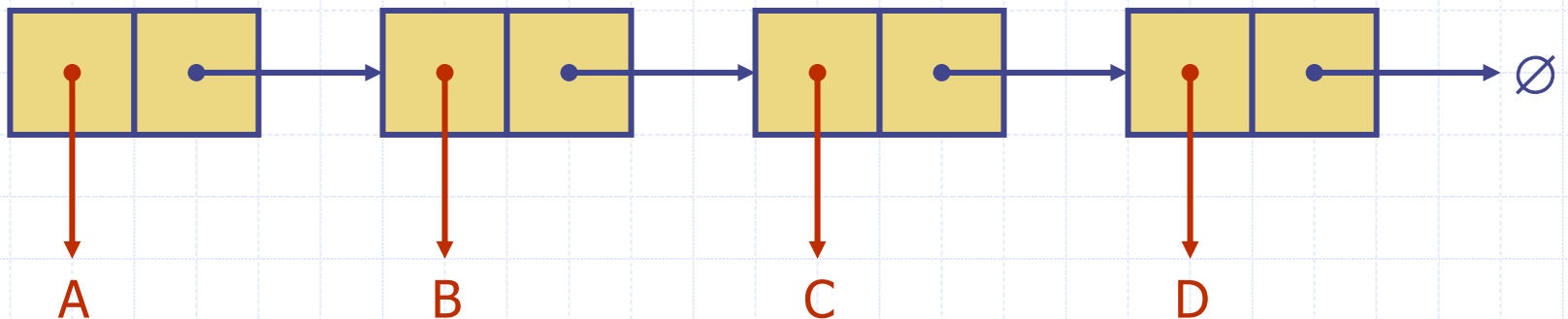
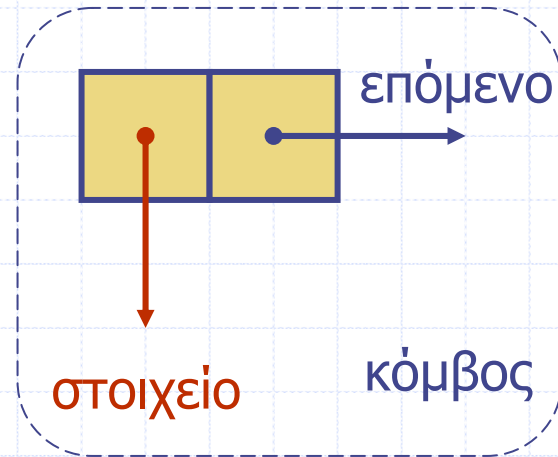


# Κύρια σημεία για μελέτη

- ◆ Απλά συνδεδεμένη λίστα
- ◆ Ο ΑΤΔ της Θέσης και της Λίστας (§2.2.2)
- ◆ Διπλά συνδεδεμένη λίστα (§ 2.2.2)
- ◆ ΑΤΔ της Ακολουθίας (§ 2.2.3)
- ◆ Υλοποίηση του ΑΤΔ της Ακολουθίας (§ 2.2.3)
- ◆ Iterators (2.2.3)

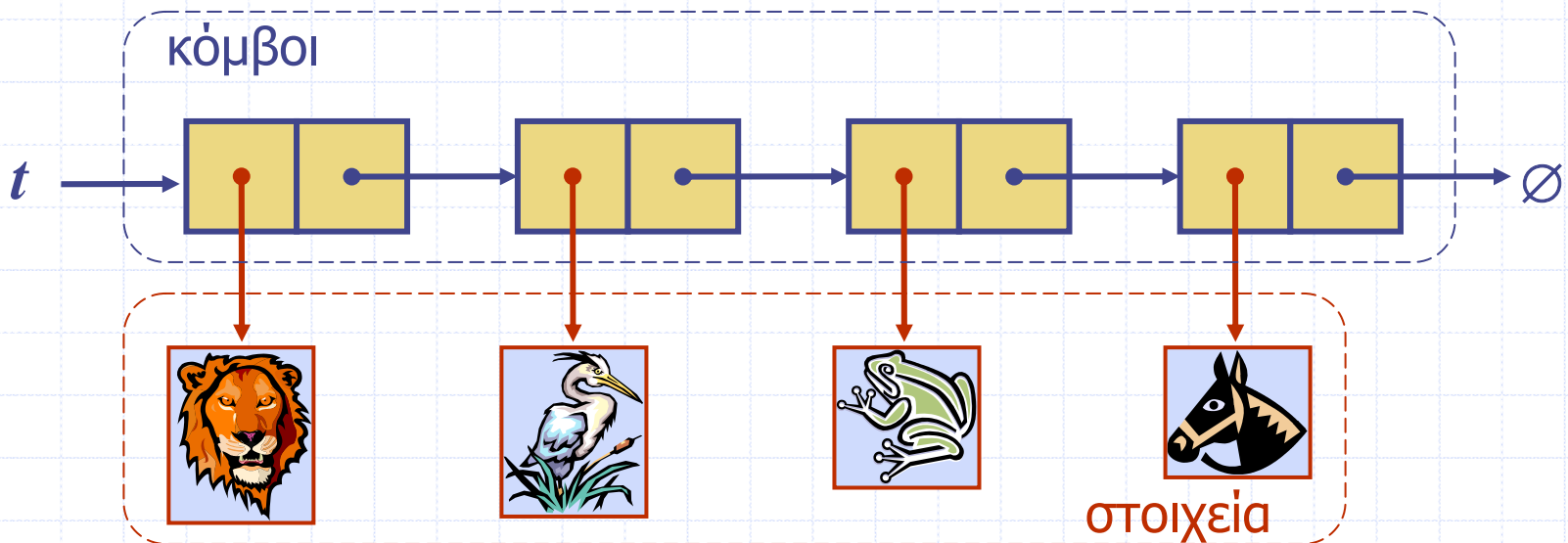
# Απλά Συνδεδεμένη Λίστα

- ◆ Μία απλά συνδεδεμένη λίστα είναι μία δομή δεδομένων αποτελούμενη από μία ακολουθία κόμβων
- ◆ Σε κάθε κόμβο αποθηκεύεται
  - Ένα στοιχείο
  - Ένας σύνδεσμος στον επόμενο κόμβο



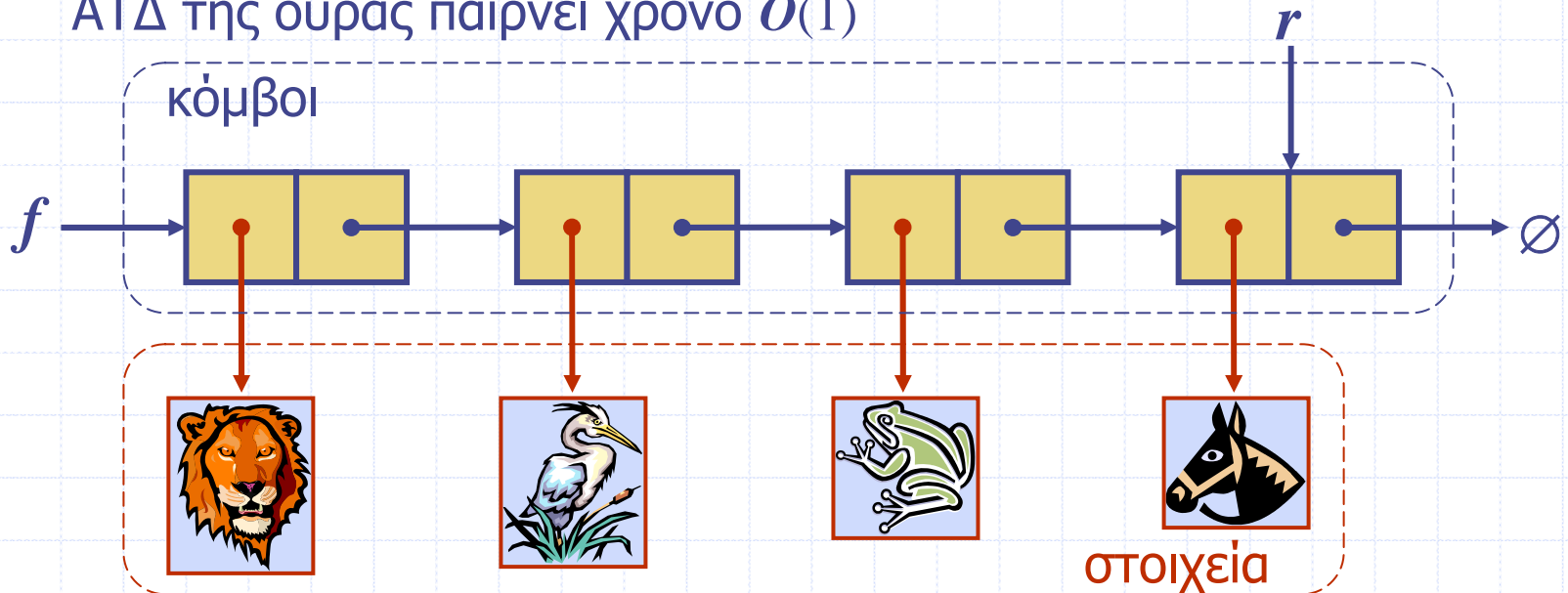
# Στοιίβα με μία Απλά Συνδεδεμένη Λίστα

- ◆ Μπορούμε να υλοποιήσουμε μια στοιίβα με μια απλά συνδεδεμένη λίστα
- ◆ Το στοιχείο κορυφή αποθηκεύεται στον πρώτο κόμβο της λίστας
- ◆ Ο χώρος που χρησιμοποιείται είναι  $O(n)$  και κάθε λειτουργία του ΑΤΔ της στοιίβας παίρνει χρόνο  $O(1)$



# Ουρά με Απλά Συνδεδεμένη Λίστα

- ◆ Μπορούμε να υλοποιήσουμε μια ουρά με μια απλά συνδεδεμένη λίστα
  - Το πρώτο στοιχείο αποθηκεύεται στον πρώτο κόμβο
  - Το τελευταίο στοιχείο αποθηκεύεται στον τελευταίο κόμβο
- ◆ Ο χώρος που χρησιμοποιείται είναι  $O(n)$  και κάθε λειτουργία του ΑΤΔ της ουράς παίρνει χρόνο  $O(1)$



# Ο ΑΤΔ της Θέσης

- ◆ Ο ΑΤΔ της **Θέσης** μοντελοποιεί την ιδέα της θέσης μέσα σε μία δομή δεδομένων όπου είναι αποθηκευμένο ένα απλό αντικείμενο
- ◆ Παρέχει μια ενοποιημένη οπτική διαφορετικών τρόπων αποθήκευσης δεδομένων, όπως
  - Ένα κελί ενός πίνακα
  - Ένα κόμβο από μια συνδεδεμένη λίστα
- ◆ Μόνο μία μέθοδος χρησιμοποιείται:
  - Το αντικείμενο **element()**: επιστρέφει το στοιχείο που είναι αποθηκευμένο στη θέση

# Ο ΑΤΔ της Λίστας

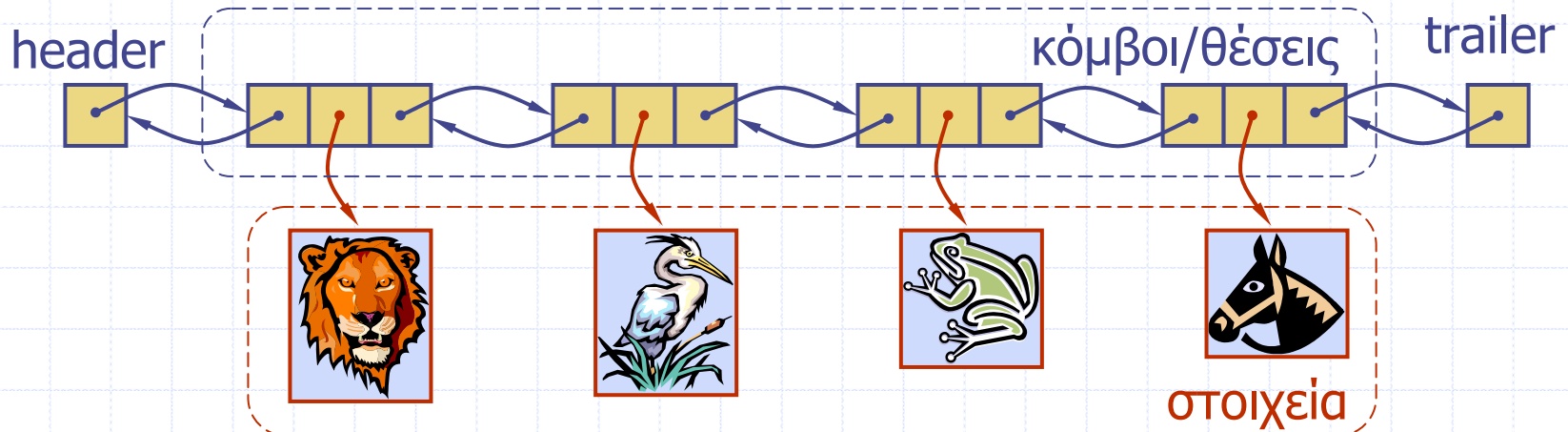
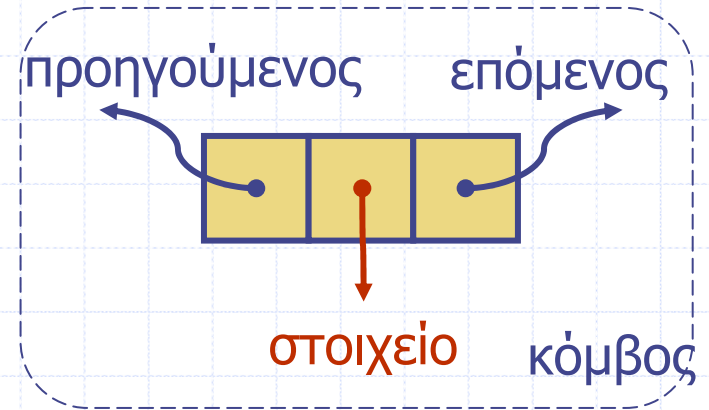
- ◆ Ο ΑΤΔ της **Λίστας** μοντελοποιεί μια ακολουθία από θέσεις που έχουν αποθηκευθεί αυθαίρετα δεδομένα
- ◆ Καθιστά μία σχέση before/after ανάμεσα στις θέσεις των αντικειμένων
- ◆ Γενικές μέθοδοι(Generic) :
  - **size()**, **isEmpty()**
- ◆ Μέθοδοι ερωτήσεων(Query) :
  - **isFirst(p)**, **isLast(p)**

Μέθοδοι πρόσπελασης(Accessor) :

- **first()**, **last()**
- **before(p)**, **after(p)**
- ◆ Update methods:
  - **replaceElement(p, o)**, **swapElements(p, q)**
  - **insertBefore(p, o)**, **insertAfter(p, o)**,
  - **insertFirst(o)**, **insertLast(o)**
  - **remove(p)**

# Διπλά Συνδεδεμένες Λίστες

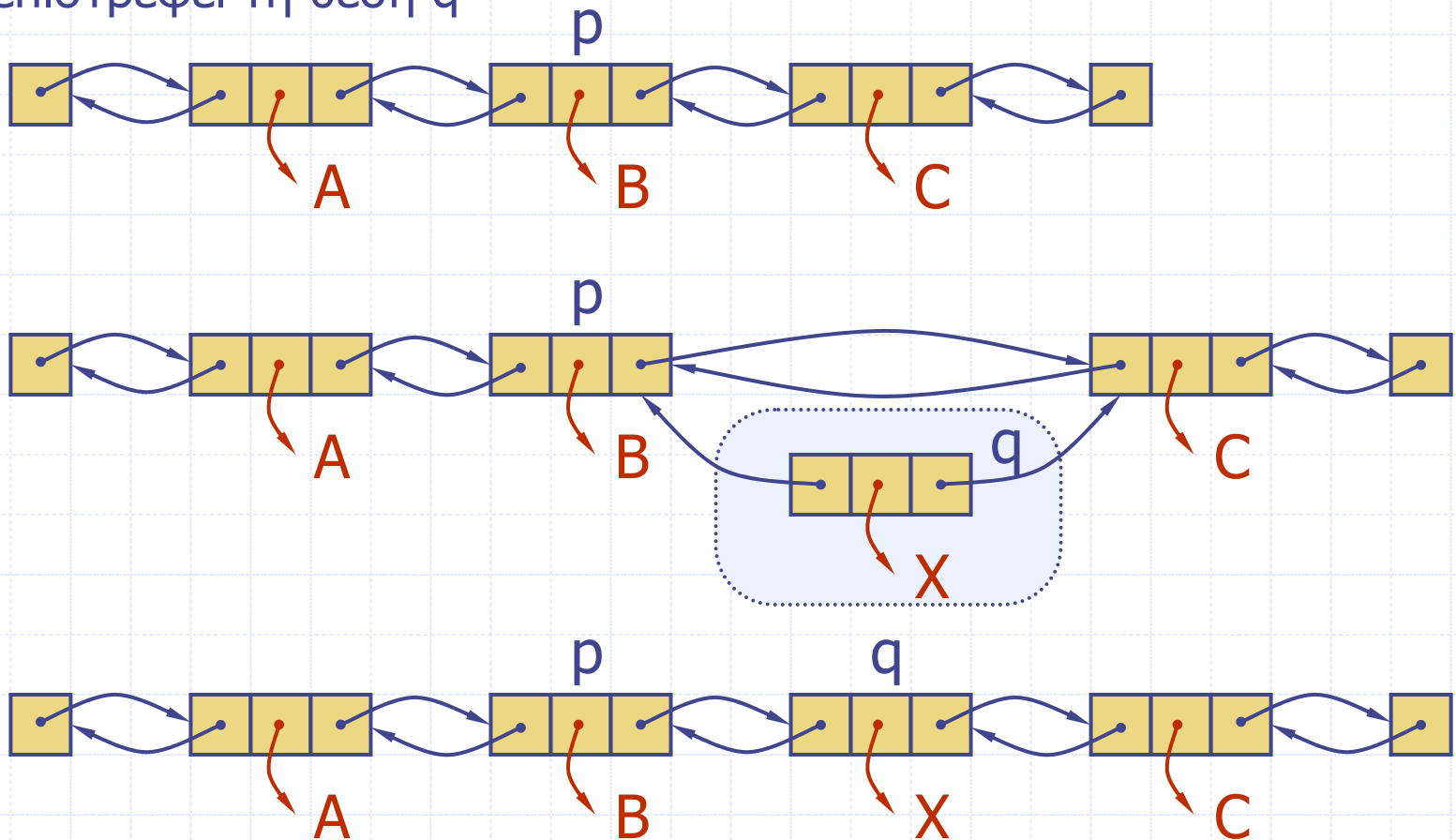
- ◆ Μια Διπλά Συνδεδεμένη Λίστα ορίζει μια φυσική υλοποίηση του ΑΤΔ της λίστας
- ◆ Οι κόμβοι υλοποιούν τη θέση και αποθηκεύουν:
  - Ένα στοιχείο
  - Ένα σύνδεσμο στον προηγούμενο κόμβο
  - Ένα σύνδεσμο στον επόμενο κόμβο
- ◆ Ειδικοί κόμβοι trailer και header





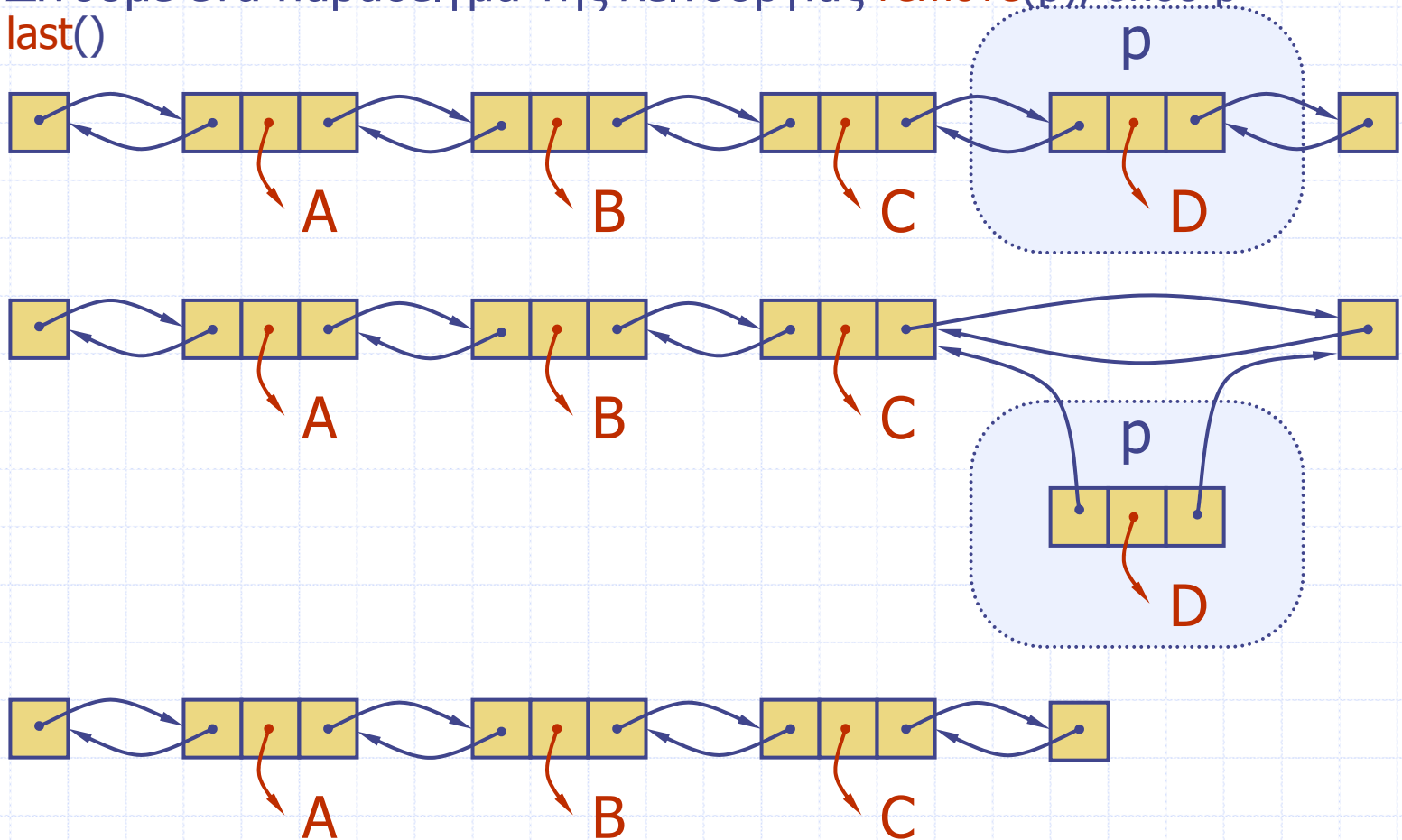
# Εισαγωγή

- ◆ Δίνουμε ένα παράδειγμα της λειτουργίας `insertAfter(p, X)`, η οποία επιστρέφει τη θέση `q`



# Διαγραφή

- ◆ Δίνουμε ένα παράδειγμα της λειτουργίας `remove(p)`, όπου  $p = \text{last}()$



# Απόδοση

- ◆ Στην υλοποίηση του ΑΤΔ της λίστας μέσω μιας διπλά συνδεδεμένης λίστας
  - Ο χώρος που χρησιμοποιείται από μία λίστα με  $n$  στοιχεία είναι  $O(n)$
  - Ο χώρος που χρησιμοποιείται από κάθε θέση της λίστας είναι  $O(1)$
  - Όλες οι λειτουργίες του ΑΤΔ της λίστας τρέχουν σε χρόνο  $O(1)$
  - Η λειτουργία `element()` του ΑΤΔ της λίστας τρέχει σε χρόνο  $O(1)$

# ΑΤΔ της Ακολουθίας

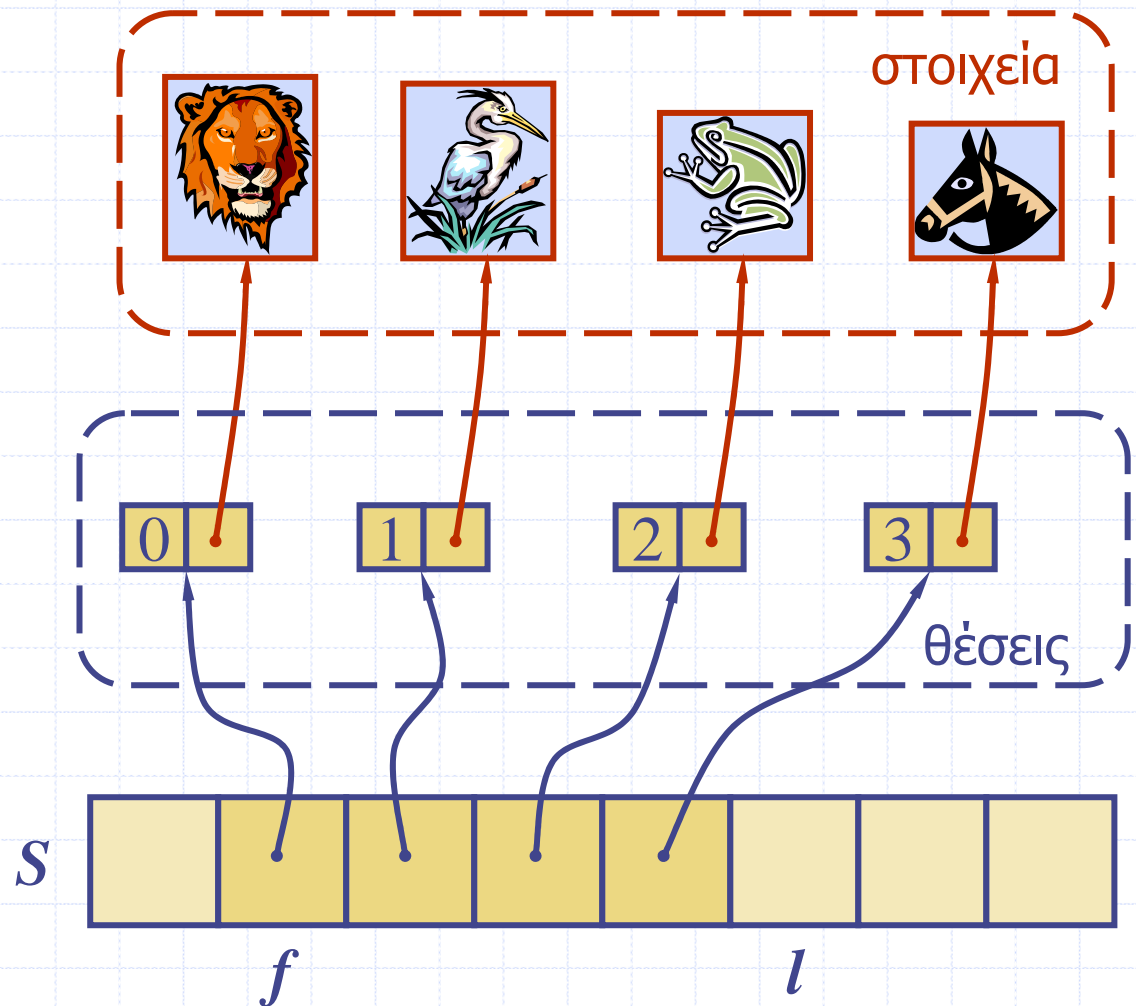
- ◆ Ο ΑΤΔ της Ακολουθίας είναι η ένωση των ΑΤΔ του Διανύσματος και της Λίστας
- ◆ Τα στοιχεία προσπελάζονται από
  - την τάξη (rank), ή
  - τη θέση(position)
- ◆ Γενικές μέθοδοι(Generic):
  - `size()`, `isEmpty()`
- ◆ Μέθοδοι βασισμένοι σε διάνυσμα(Vector-based):
  - `elemAtRank(r)`,  
`replaceAtRank(r, o)`,  
`insertAtRank(r, o)`,  
`removeAtRank(r)`
- ◆ Μέθοδοι βασισμένοι σε λίστα(List-based):
  - `first()`, `last()`,  
`before(p)`, `after(p)`,  
`replaceElement(p, o)`,  
`swapElements(p, q)`,  
`insertBefore(p, o)`,  
`insertAfter(p, o)`,  
`insertFirst(o)`,  
`insertLast(o)`,  
`remove(p)`
- ◆ Μέθοδοι της Γέφυρας (Bridge):
  - `atRank(r)`, `rankOf(p)`

# Εφαρμογές των Ακολουθιών

- ◆ Ο ΑΤΔ μιας ακολουθίας είναι μία βασική, γενικού σκοπού, δομή δεδομένων για αποθήκευση μιας διατεταγμένης συλλογής στοιχείων
- ◆ Άμεσες εφαρμογές:
  - Γενική αντικατάσταση για στοίβα, ουρά, διάνυσμα, ή λίστα
  - Μικρή βάση δεδομένων (π.χ, βιβλίο διευθύνσεων)
- ◆ Έμμεσες εφαρμογές:
  - Δημιουργία τμήματος(block) πιο περίπλοκων δομών δεδομένων

# Υλοποίηση βασισμένη σε πίνακα

- ◆ Χρησιμοποιούμε έναν κυκλικό πίνακα που αποθηκεύει τις θέσεις
- ◆ Ένα αντικείμενο θέσης αποθηκεύει :
  - Ένα στοιχείο
  - Μία τάξη
- ◆ Οι δείκτες  $f$  και  $l$  δείχνουν την πρώτη και τελευταία θέση



# Υλοποιήσεις Ακολουθίας

Λειτουργία	Πίνακας	Λίστα
size, isEmpty	1	1
atRank, rankOf, elemAtRank	1	<i>n</i>
first, last, before, after	1	1
replaceElement, swapElements	1	1
replaceAtRank	1	<i>n</i>
insertAtRank, removeAtRank	<i>n</i>	<i>n</i>
insertFirst, insertLast	1	1
insertAfter, insertBefore	<i>n</i>	1
remove	<i>n</i>	1

# Iterators

- ◆ Ένας iterator αποτελεί μια γενίκευση της διαδικασίας σαρώματος μιας συλλογής στοιχείων
- ◆ Μέθοδοι του ΑΤΔ του ObjectIterator :
  - object **object()**
  - boolean **hasNext()**
  - object **nextObject()**
  - **reset()**
- ◆ Αποτελεί μια προέκταση της ιδέας της Θέσης προσθέτοντας μία δυνατότητα διάσχισης
- ◆ Υλοποίηση με πίνακα ή απλά συνδεδεμένη λίστα
- ◆ Ένας iterator σχετίζεται τυπικά με μία άλλη δομή δεδομένων
- ◆ Μπορούμε να ενδυναμώσουμε τον ΑΤΔ της στοίβας, της ουράς, του διανύσματος, της λίστας και της ακολουθίας με τη μέθοδο:
  - ObjectIterator **elements()**
- ◆ Δύο οπτικές του iterator:
  - στιγμιότυπο: παγώνει τα περιεχόμενα της δομής δεδομένων σε μια δεδομένη χρονική στιγμή
  - δυναμικό: ακολουθεί τις αλλαγές στη δομή δεδομένων