3η Ενότητα

Σχεδίαση αλγορίθμων

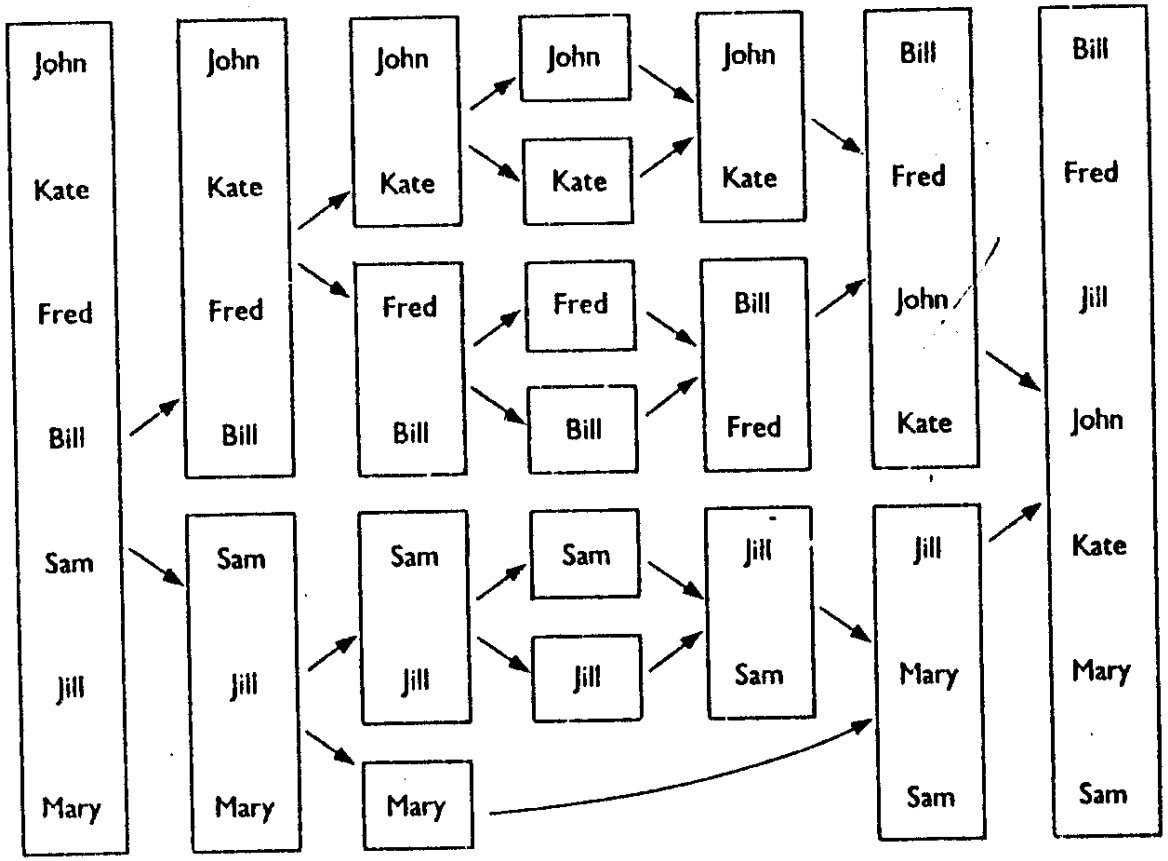
Αλγόριθμος δυαδικής αναζήτησης

```
function binarySearch(a, value, left, right)
while left ≤ right
        mid := floor((left+right)/2)
        if value > a[mid] then left := mid+1
        else
                if value < a[mid] then right := mid-1
        else return mid
    return not found
```

Αλγόριθμος δυαδικής αναζήτησης

- **function** binarySearch(a, value, left, right)
  if right < left
  return *not found*
  mid := floor((left+right)/2)
  if value > a[mid]
  return binarySearch(a, value, mid+1, right)
  else if value < a[mid]
  return binarySearch(a, value, left, mid-1)
  else return mid

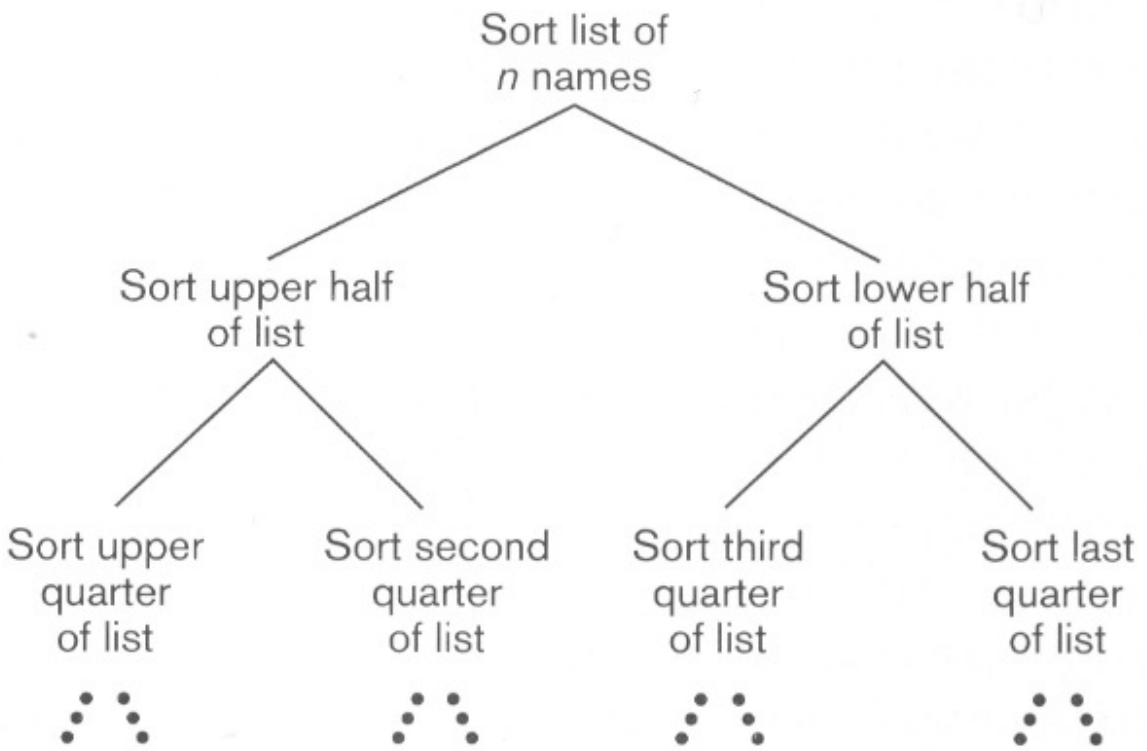
Διαίρει και βασίλευε: Παράδειγμα merge - sort



Παράδειγμα merge - sort

```
procedure MergeSort (List)
if (List has more than one entry)
 then(Apply the procedure MergeSort to sort the first half of List;
      Apply the procedure MergeSort to sort the second half of List;
      Apply the procedure MergeLists to merge the first and second
         halves of List to produce a sorted version of List
      )
```

Παράδειγμα merge - sort



Παράδειγμα merge - sort

```
procedure MergeLists (InputListA, InputListB, OutputList)
if (both input lists are empty) then (Stop, with OutputList empty)
if (InputListA is empty)
   then (Declare it to be exhausted)
   else (Declare its first entry to be its current entry)
if (InputListB is empty)
   then (Declare it to be exhausted)
   else (Declare its first entry to be its current entry)
while (neither input list is exhausted) do
      (Put the "smaller" current entry in OutputList;
      if (that current entry is the last entry in its corresponding input list)
         then (Declare that input list to be exhausted)
      (  else (Declare the next entry in that input list to be the list's current entry
      )
Starting with the current entry in the input list that is not exhausted,
      copy the remaining entries to OutputList.
```

Συγχώνευση δύο λιστών με μήκος α και β δεν περιέχει περισσότερες από α+β συγκρίσεις

Παράδειγμα merge - sort

- Το αρχικό πρόβλημα μεγέθους n μετατρέπεται σε 2 άλλα μεγέθους περίπου n/2
- Τα δύο προβλήματα μετατρέπονται σε 4 προβλήματα μεγέθους περίπου n/4
- Αριθμός συγκρίσεων σε κάθε επίπεδο όχι μεγαλύτερο από το συνολικό άθροισμα των στοιχείων των υπο-λιστών
- Ο αριθμός των επιπέδων στο δέντρο είναι της τάξης του logn

Διαίρει και βασίλευε (divide and conquer)

- Merge – Sort

  $T(n) = 2T(n/2) + cn$

  $T(1) = k$

Αποδεικνύεται ότι $T(n) = cn\log n + kn$

(ασυμπτωτικός χρόνος εκτέλεσης ανάλογος του nlogn)

2η γνωστική ενότητα : σχεδίαση αλγορίθμων

1. Παραλληλία
2. Δομές Δεδομένων

Η έννοια της παραλληλίας

- σειριακότητα : η εκτέλεση, σε ένα διάστημα χρόνου, μόνο 1 βήματος ενός αλγορίθμου, από 1 επεξεργαστή

- παραλληλία βαθμού Ν : η εκτέλεση, στο ίδιο διάστημα χρόνου, Ν βημάτων ενός αλγορίθμου, από Ν επεξεργαστές οι οποίοι λειτουργούν ταυτόχρονα
- συνώνυμο : ταυτοχρονισμός (concurrency) βαθμού Ν

Το χρονικό όφελος της παραλληλίας

- έστω $t_{exec}(S)$ ο χρόνος εκτέλεσης ενός βήματος S

- ο συνολικός χρόνος σειριακής εκτέλεσης N >= 1 βημάτων $S_1, ..., S_N$ θα είναι
  $t_{serial}(S_1, ..., S_N) = t_{exec}(S_1) + ... + t_{exec}(S_N)$
- ο συνολικός χρόνος παράλληλης εκτέλεσης των $S_1, ..., S_N$ θα είναι
  $t_{parallel}(S_1, ..., S_N) = \max\{t_{exec}(S_1), ..., t_{exec}(S_N)\}$

- προφανώς, $t_{parallel}(S_1, ..., S_N) < t_{serial}(S_1, ..., S_N)$
- η διαφορά $t_{serial}(S_1, ..., S_N) - t_{parallel}(S_1, ..., S_N)$ αντιπροσωπεύει την απώλεια χρόνου λόγω της σειριακής εκτέλεσης των $S_1, ..., S_N$

Η δυνατότητα παραλληλίας

- δύο βήματα S, T βρίσκονται σε λογική ακολουθία ανν το T μπορεί να αρχίσει να εκτελείται μόνο αφού ολοκληρωθεί η εκτέλεση του S, διότι αλλιώς θα δώσει λανθασμένα αποτελέσματα
- παράδειγμα : το T χρησιμοποιεί μεταβλητές στις οποίες πρέπει προηγουμένως να έχει δώσει τιμή το S

- δύο βήματα S, T τα οποία βρίσκονται σε λογική ακολουθία θα πρέπει να εκτελεστούν και σε χρονική ακολουθία, δηλαδή σειριακά
- δύο βήματα S, T τα οποία δεν βρίσκονται σε λογική ακολουθία, μπορούν να εκτελεστούν παράλληλα

Παραλληλοποίηση ενός αλγορίθμου

- παραλληλοποίηση ενός αλγορίθμου : ο εντοπισμός των βημάτων του τα οποία μπορούν να εκτελεστούν παράλληλα και ο μετασχηματισμός του κατά τρόπο ώστε η παράλληλη εκτέλεση να διευκολύνεται
- βαθμός παραλληλίας : ο βαθμός στον οποίο η παραλληλοποίηση ενός αλγορίθμου μπορεί να αναδείξει παραλλήλως εκτελέσιμα βήματα
- το όφελος σε μείωση του χρόνου εκτέλεσης εξαρτάται από
  α) το βαθμό παραλληλοποίησης του αλγορίθμου, και
  β) το πλήθος των διαθέσιμων επεξεργαστών

Δομή ελέγχου παραλληλίας

- η δήλωση παράλληλης εκτέλεσης κάποιων βημάτων απαιτεί μια ειδική δομή ελέγχου

παράλληλα για στιγμιότυπο = 1 έως N
    start
        σώμα παραλληλίας (στιγμιότυπο)
    end

- παράλληλες γλώσσες προγραμματισμού : οι γλώσσες προγραμματισμού που προσφέρουν δομές ελέγχου για παράλληλη εκτέλεση

Σημασιολογία εκτέλεσης

- η δομή παραλληλίας εκτελεί ένα σύνολο βημάτων ή καλεί ένα τμήμα (σώμα παραλληλίας) Ν φορές (σε Ν στιγμιότυπα παραλληλίας), μία φορά για κάθε τιμή του στιγμιότυπου

- εφόσον υπάρχουν διαθέσιμοι Ν επεξεργαστές, κάθε στιγμιότυπο παραλληλίας εκτελείται από έναν από αυτούς και όλα τα στιγμιότυπα παραλληλίας εκτελούνται ταυτόχρονα

- η εκτέλεση της δομής παραλληλίας ολοκληρώνεται όταν ολοκληρωθούν οι εκτελέσεις όλων των στιγμιότυπων παραλληλίας
- ο χρόνος εκτέλεσης της δομής παραλληλίας είναι ίσος με το χρόνο εκτέλεσης του βραδύτερου στιγμιότυπου

Παράδειγμα Ι – λίστα τιμών

- έστω μια λίστα Ν >= 1 προϊόντων η οποία αναφέρει κατά προϊόν τις τιμές αγοράς και πώλησης
- το ζητούμενο είναι να σαρώσουμε όλη τη λίστα υπολογίζοντας και συμπληρώνοντας, για κάθε προϊόν, το περιθώριο κέρδους

```
{σειριακή υλοποίηση υπολογισμού κέρδους}
    επανάληψη για μετρητής = 1 έως Ν
        πάρε το επόμενο στοιχείο της λίστας
        περιθώριο κέρδους := τιμή πώλησης – τιμή αγοράς
        συμπλήρωσε το περιθώριο κέρδους στο στοιχείο αυτό
    end
```

Παράδειγμα Ι – λίστα τιμών

- συνολικά εκτελούνται Ν όμοια βήματα κατά σειριακό τρόπο
- εάν ο χρόνος εκτέλεσης ενός βήματος είναι t, τότε
  ο συνολικός χρόνος εκτέλεσης είναι T = N*t
  και αυξάνει γραμμικά ως προς το μήκος Ν της λίστας

- τα επί μέρους βήματα παρουσιάζουν λογική ανεξαρτησία,
  επομένως δυνατότητα παραλληλοποίησης

- όλα τα Ν βήματα μπορούν να εκτελεστούν παράλληλα,
  εφόσον έχουμε διαθέσιμους ισάριθμους (Ν) επεξεργαστές
  οι οποίοι θα λειτουργήσουν ταυτόχρονα

Παράδειγμα Ι – λίστα τιμών

```
{παράλληλη υλοποίηση υπολογισμού κέρδους}
start
    παράλληλα για θέση λίστας = 1 έως Ν
    start
        παράλληλο κέρδος (θέση λίστας)
    end
end

module παράλληλο κέρδος (Θ)
start
    περιθώριο κέρδους [Θ] := τιμή πώλησης [Θ] – τιμή αγοράς [Θ]
end
```

Παράδειγμα Ι – λίστα τιμών

- συνολικά εκτελούνται Ν όμοια βήματα κατά παράλληλο τρόπο
- εάν ο χρόνος εκτέλεσης ενός βήματος είναι t, τότε
  ο συνολικός χρόνος εκτέλεσης είναι T = t
  και ανεξάρτητος από το μήκος Ν της λίστας

- έχουμε μέγιστο όφελος μείωσης του χρόνου εκτέλεσης, διότι
  α) ο αλγόριθμος παρουσιάζει μέγιστο βαθμό παραλληλίας, και
  β) θεωρούμε διαθέσιμο ένα πλήθος Ν επεξεργαστών

Παράδειγμα ΙΙ – άθροισμα λίστας

- έστω μια λίστα Ν >= 1 αριθμών τους οποίους θέλουμε να αθροίσουμε

```
{σειριακή υλοποίηση αθροίσματος λίστας}
    άθροισμα := 0
    for  θέση λίστας = 1 to N
         άθροισμα := άθροισμα + στοιχείο [θέση λίστας]
    end
```

Παράδειγμα ΙΙ – άθροισμα λίστας

- συνολικά εκτελούνται Ν όμοιες αθροίσεις κατά σειριακό τρόπο
- εάν ο χρόνος εκτέλεσης μιας άθροισης είναι t, τότε
  ο συνολικός χρόνος εκτέλεσης είναι T = N*t
  και αυξάνει γραμμικά ως προς το μήκος Ν της λίστας

- τα επί μέρους βήματα δεν παρουσιάζουν λογική ανεξαρτησία, διότι όλα χρησιμοποιούν και ενημερώνουν την ίδια μεταβλητή

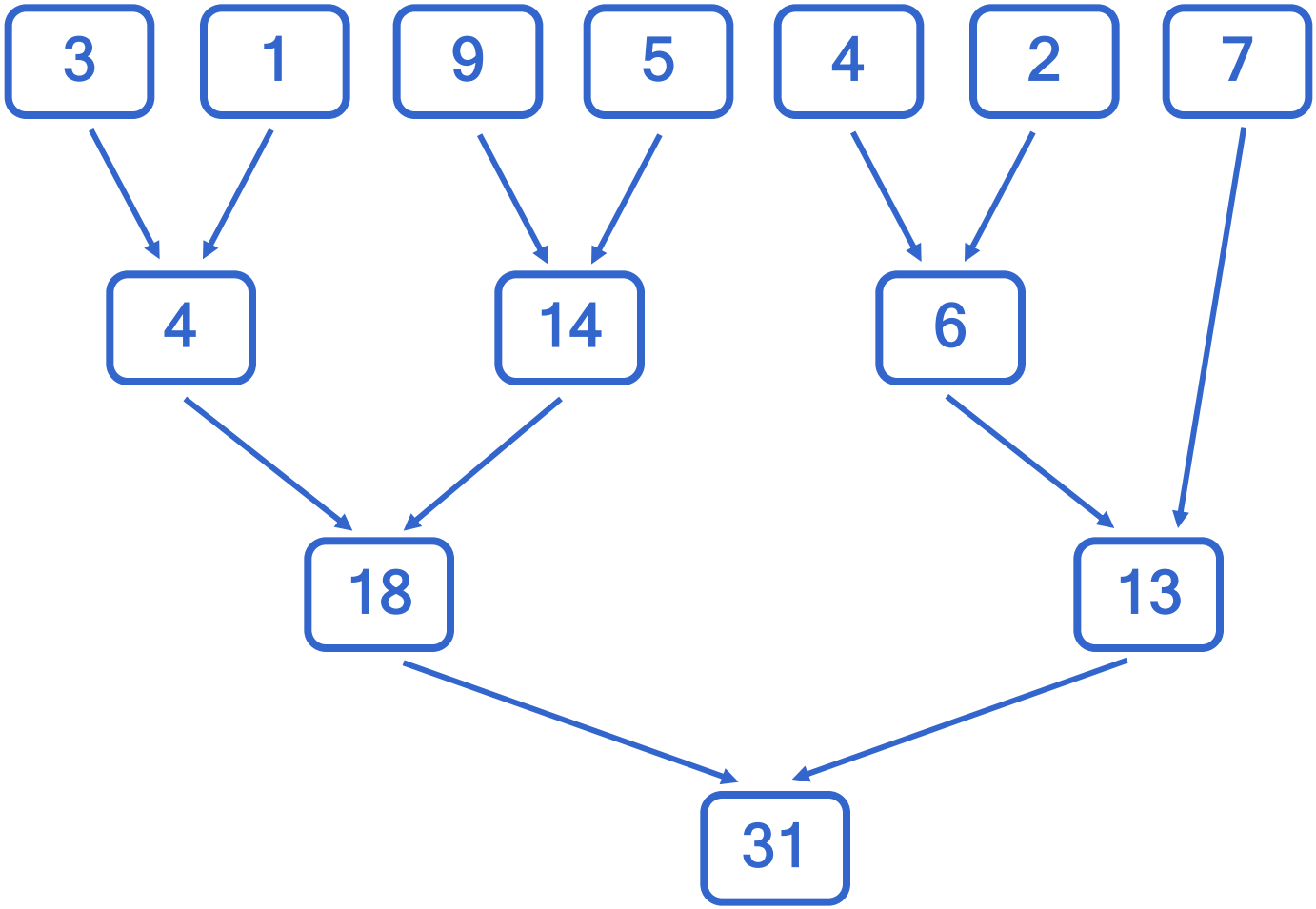
Παράδειγμα ΙΙ – άθροισμα λίστας

- μπορούμε να δημιουργήσουμε λογικώς ανεξάρτητα βήματα εάν, αντί να αθροίζουμε διαδοχικά, αθροίσουμε πρώτα όλα τα στοιχεία κατά ζεύγη, μετά τα ενδιάμεσα πάλι κατά ζεύγη, κ.ο.κ.

- εάν η λίστα έχει μήκος 4 θα χρειαστούν 2 βήματα :
  1ο βήμα : 2 αθροίσεις (λογικά ανεξάρτητες)
  2ο βήμα : 1 άθροιση (τελικό αποτέλεσμα)

- γενικά, εάν η λίστα έχει μήκος N θα χρειαστούν $\log_2 N$ βήματα (για την ακρίβεια $\lceil \log_2 N \rceil$ βήματα) στα οποία θα εκτελεστούν N/2, N/4, …, 1 άθροιση

Παράδειγμα ΙΙ – άθροισμα λίστας



Παράδειγμα ΙΙ – άθροισμα λίστας

- οι αθροίσεις κάθε βήματος είναι λογικώς ανεξάρτητες και επομένως μπορούν να εκτελεστούν παράλληλα
- τα βήματα μεταξύ τους είναι λογικώς εξαρτημένα και επομένως πρέπει να εκτελεστούν σειριακά
- θα εκτελεστούν $\log_2 N$ σειριακά βήματα εκ των οποίων το κάθε ένα θα περιλαμβάνει το πολύ N/2 αθροίσεις
- οι αθροίσεις κάθε βήματος μπορούν να εκτελεστούν παράλληλα εφόσον έχουμε διαθέσιμους N/2 επεξεργαστές

Παράδειγμα ΙΙ – άθροισμα λίστας

```
{παράλληλη υλοποίηση αθροίσματος λίστας}
start
    παράλληλα για θέση λίστας = 1 έως N/2
    start
        παράλληλη άθροιση (θέση λίστας)
    end
end

module παράλληλη άθροιση (Θ)
start
    for βήμα = 1 to log2N
        στοιχείο [Θ] := στοιχείο [(2*Θ)-1] + στοιχείο [2*Θ]
end
```

Παράδειγμα ΙΙ – άθροισμα λίστας

εκτέλεση για τη λίστα [3, 1, 9, 5, 4, 2, 7]

αρχική σειρά	3	1	9	5	4	2	7
μετά το 1ο βήμα	4	14	6	7	4	2	7
μετά το 2ο βήμα	18	13	6	7	4	2	7
μετά το 3ο βήμα	31	13	6	7	4	2	7

το στοιχείο λίστα[1] έχει μετά το 3ο βήμα την τιμή του συνολικού αθροίσματος

Παράδειγμα ΙΙ – άθροισμα λίστας

- συνολικά εκτελούνται $\log_2 N$ όμοια βήματα κατά σειριακό τρόπο, και σε κάθε βήμα N/2 όμοιες αθροίσεις κατά παράλληλο τρόπο
- εάν ο χρόνος εκτέλεσης μιας άθροισης είναι t, τότε
  ο χρόνος εκτέλεσης κάθε βήματος είναι t, ενώ
  ο συνολικός χρόνος εκτέλεσης είναι $T = \log_2 N * t$
  και αυξάνει λογαριθμικά ως προς το μήκος N της λίστας

- η λογαριθμική αύξηση είναι πολύ βραδύτερη της γραμμικής

- έχουμε όφελος μείωσης του χρόνου εκτέλεσης, διότι
  α) ο αλγόριθμος παρουσιάζει (μη προφανή) παραλληλία, και
  β) θεωρούμε διαθέσιμο ένα πλήθος N/2 επεξεργαστών

Η ανάγκη του συγχρονισμού

- στην παράλληλη άθροιση λίστας, τα επί μέρους βήματα είναι λογικά εξαρτημένα (το επόμενο αθροίζει τα ενδιάμεσα αποτελέσματα που παρήγαγε το προηγούμενο), επομένως πρέπει να εκτελεστούν σειριακά

- όλοι οι επεξεργαστές E_1, …, $E_{N/2}$ πρέπει να ολοκληρώνουν ταυτόχρονα ένα βήμα της σειριακής επανάληψης και να αρχίζουν ταυτόχρονα το επόμενο
- κάθε επεξεργαστής E_i πρέπει να αρχίσει το επόμενο βήμα αφού
  α) ολοκληρώσει το τρέχον βήμα, και
  β) ειδοποιηθεί ότι και όλοι οι υπόλοιποι επεξεργαστές το ολοκλήρωσαν

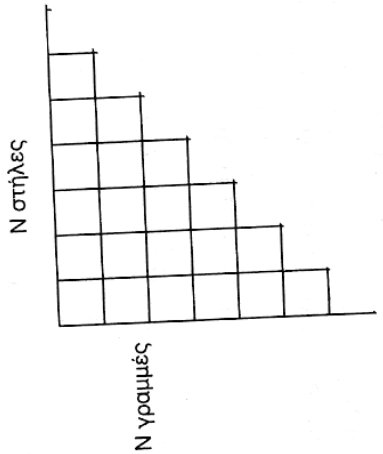
Η έννοια του συγχρονισμού

- συγχρονισμός δύο αλγοριθμικών βημάτων S, T :
  η παράλληλη εκτέλεση των βημάτων αυτών κατά τέτοιο τρόπο ώστε να μην ξεκινά το επόμενο σειριακό βήμα εάν δεν ελεγχθεί ότι ολοκληρώθηκαν και τα δύο
- ο συγχρονισμός N >= 2 παράλληλων βημάτων S1, ..., SN απαιτεί ένα επιπλέον έλεγχο για την ολοκλήρωση όλων των βημάτων πριν την έναρξη του σειριακά επόμενου

- ο αναγκαίος συγχρονισμός πρέπει να δηλώνεται κατά τη σχεδίαση του αλγορίθμου
- οι παράλληλες γλώσσες προγραμματισμού προσφέρουν δομές ελέγχου συγχρονισμένης παράλληλης εκτέλεσης

Παράλληλη σύγκριση

	Ιωάννης	Κατερίνα	Στέλιος	Γιώργος	Βασίλης	Ιουλία	Μαρία	Θέση στη διατεταγμένη
Ιωάννης	1	0	0	1	1	1	0	4
Κατερίνα	1	1	0	1	1	1	0	5
Στέλιος	1	1	1	1	1	1	1	7
Γιώργος	0	0	0	1	1	0	0	2
Βασίλης	0	0	0	0	1	0	0	1
Ιουλία	0	0	0	1	1	1	0	3
Μαρία	1	1	0	1	1	1	1	6



module *παράλληλη-σύγκριση (i,j)*
{κάθε επεξεργαστής το εκτελεί με διαφορετικές τιμές i και j}
if *όνομα[i] προηγείται αλφαβητικά του όνομα[j]*
  then *βάλε 0 στην i-οστή γραμμή και j-οστή στήλη του πλέγματος*
  else *βάλε 1 στην i-οστή γραμμή και j-οστή στήλη του πλέγματος*

Σύγκριση χρόνων εκτέλεσης αλγορίθμων

Μέγεθος N των δεδομένων εισόδου	$\log_2 N$ μικροδευτερόλεπτα	$N \log_2 N$ μικροδευτερόλεπτα	N^2 μικροδευτερόλεπτα
10	0.000003 δευτερόλεπτα	0.00003 δευτερόλεπτα	0.0001 δευτερόλεπτα
100	0.000007 δευτερόλεπτα	0.0007 δευτερόλεπτα	0.01 δευτερόλεπτα
1000	0.00001 δευτερόλεπτα	0.01 δευτερόλεπτα	1 δευτερόλεπτα
10000	0.000013 δευτερόλεπτα	0.13 δευτερόλεπτα	1.7 λεπτά
100000	0.000σζ017 δευτερόλεπτα	1.7 δευτερόλεπτα	2.8 ώρες

Τύποι και δομές δεδομένων

- ατομικά δεδομένα
- εγγραφές
- απλοί και σύνθετοι τύποι δεδομένων
- δηλώσεις ορισμού τύπων δεδομένων
- δομές δεδομένων
- χαρακτηριστικά μιας δομής δεδομένων
- επεξεργασία μιας δομής δεδομένων

Εγγραφές

- εγγραφή : μια συλλογή πληροφοριών για μια οντότητα του πραγματικού κόσμου ή μια σχέση μεταξύ τέτοιων οντοτήτων
- παράδειγμα :
  α) τα στοιχεία ενός φοιτητή (ονοματεπώνυμο, ΑΜ κ.λπ.)
  β) τα στοιχεία ενός μαθήματος (τίτλος, εξάμηνο κ.λπ.)
  γ) τα στοιχεία βαθμολογίας (φοιτητής, μάθημα, βαθμός)

- πεδία μιας εγγραφής : οι επί μέρους ατομικές πληροφορίες που την αποτελούν
- κλειδί μιας εγγραφής : ένα πεδίο ή ένας συνδυασμός πεδίων που χαρακτηρίζει την εγγραφή αυτή μοναδικά

Οριζόμενοι από το Χρήστη Τύποι Δεδομένων

- Πρότυπο για μια ετερογενή δομή

- Παράδειγμα:

```
όρισε τύπο ΤύποςΥπαλλήλου ως
{char   Όνομα[25];
int     Ηλικία;
real    ΒαθμόςΙκανότητας;
}
```

Αφηρημένοι Τύποι Δεδομένων

- Ένας οριζόμενος από το χρήστη τύπος δεδομένων με διαδικασίες για προσπέλαση και χειρισμό
- Παράδειγμα:

```
όρισε τύπο ΤύποςΣτοίβας ως
{int ΚαταχωρίσειςΣτοίβας[20];
 int ΔείκτηςΣτοίβας = 0;
 διαδικασία push(τιμή)
    {ΚαταχωρίσειςΣτοίβας[ΔείκτηςΣτοίβας] ← τιμή;
     ΔείκτηςΣτοίβας ← ΔείκτηςΣτοίβας + 1;
    }
 διαδικασία pop . . .
}
```

Απλοί και σύνθετοι τύποι δεδομένων

- τύπος δεδομένων : ένα είδος στο οποίο μπορούν να ανήκουν ορισμένα απλά ή σύνθετα δεδομένα

- απλοί τύποι δεδομένων : είδη απλών δεδομένων
  α) δυαδικά ψηφία, ψηφιολέξεις
  β) χαρακτήρες, συμβολοσειρές
  γ) ακέραιοι, πραγματικοί αριθμοί
  δ) λογικές τιμές
  ε) κείμενα, ήχος, εικόνα, βίντεο
  στ) οριζόμενοι τύποι (π.χ. οι ημέρες της εβδομάδας)

- σύνθετοι τύποι δεδομένων : είδη εγγραφών

Δηλώσεις ορισμού τύπων δεδομένων

- συνηθισμένοι απλοί τύποι : χωρίς δήλωση ορισμού

- δήλωση ορισμού σύνθετου τύπου δεδομένων

```
τύπος εγγραφής όνομα τύπου εγγραφής
        όνομα πεδίου 1 τύπος όνομα τύπου πεδίου 1
        όνομα πεδίου 2 τύπος όνομα τύπου πεδίου 2
        ...
τέλος
```

Δομές δεδομένων

- δομή δεδομένων : ένας κανόνας για τη λογική οργάνωση μιας συλλογής δεδομένων του ίδιου τύπου κατά τρόπο ώστε να διευκολύνεται
  α) η αξιοποίηση της συλλογής αυτής
  (αναζήτηση συγκεκριμένων δεδομένων μέσα στη συλλογή)
  β) η συντήρηση της συλλογής αυτής
  (εισαγωγή νέων δεδομένων στη συλλογή,
  διαγραφή υφιστάμενων δεδομένων από τη συλλογή)

Χαρακτηριστικά μιας δομής δεδομένων

- διατακτικότητα των δεδομένων :
  α) έννοια «προηγούμενου» στοιχείου
  β) έννοια «επόμενου» στοιχείου
  γ) πλήθος των προηγούμενων ενός στοιχείου
  δ) πλήθος των επόμενων ενός στοιχείου

- προσπελασιμότητα των δεδομένων
  α) άμεση προσπέλαση με απευθείας αναφορά στη θέση
  β) έμμεση προσπέλαση μέσω των γειτονικών στοιχείων

- χωρητικότητα της δομής δεδομένων
  α) προκαθορισμένη από τον ορισμό της δομής
  β) μη προκαθορισμένη από τον ορισμό της δομής

Επεξεργασία μιας δομής δεδομένων

- σάρωση της δομής : ανάκτηση όλων των στοιχείων της δομής σύμφωνα με τη φυσική τους διάταξη
- ταξινόμηση της δομής : αναδιάταξη των στοιχείων της δομής ώστε η φυσική τους διάταξη να συμπέσει με την διάταξη των τιμών τους

Επεξεργασία μιας δομής δεδομένων (συνέχ.)

- αναζήτηση ενός στοιχείου : εύρεση της θέσης του στοιχείου μέσα στη δομή ή διαπίστωση ότι το στοιχείο δεν υπάρχει (εάν υπάρχει ταξινόμηση, αξιοποίησή της)

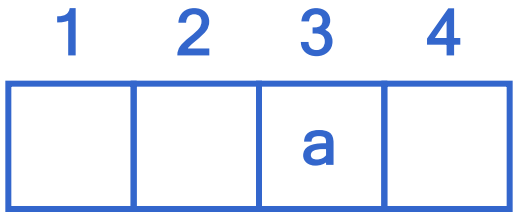
- εισαγωγή ενός στοιχείου : προσθήκη ενός νέου στοιχείου μέσα στη δομή
(εάν υπάρχει ταξινόμηση, αποκατάστασή της)

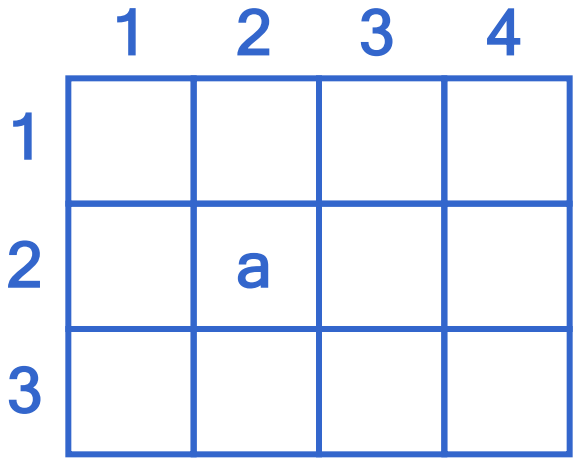
- διαγραφή ενός στοιχείου : αφαίρεση ενός υφιστάμενου στοιχείου από τη δομή
(εάν υπάρχει ταξινόμηση, αποκατάστασή της)

Η δομή δεδομένων πίνακα

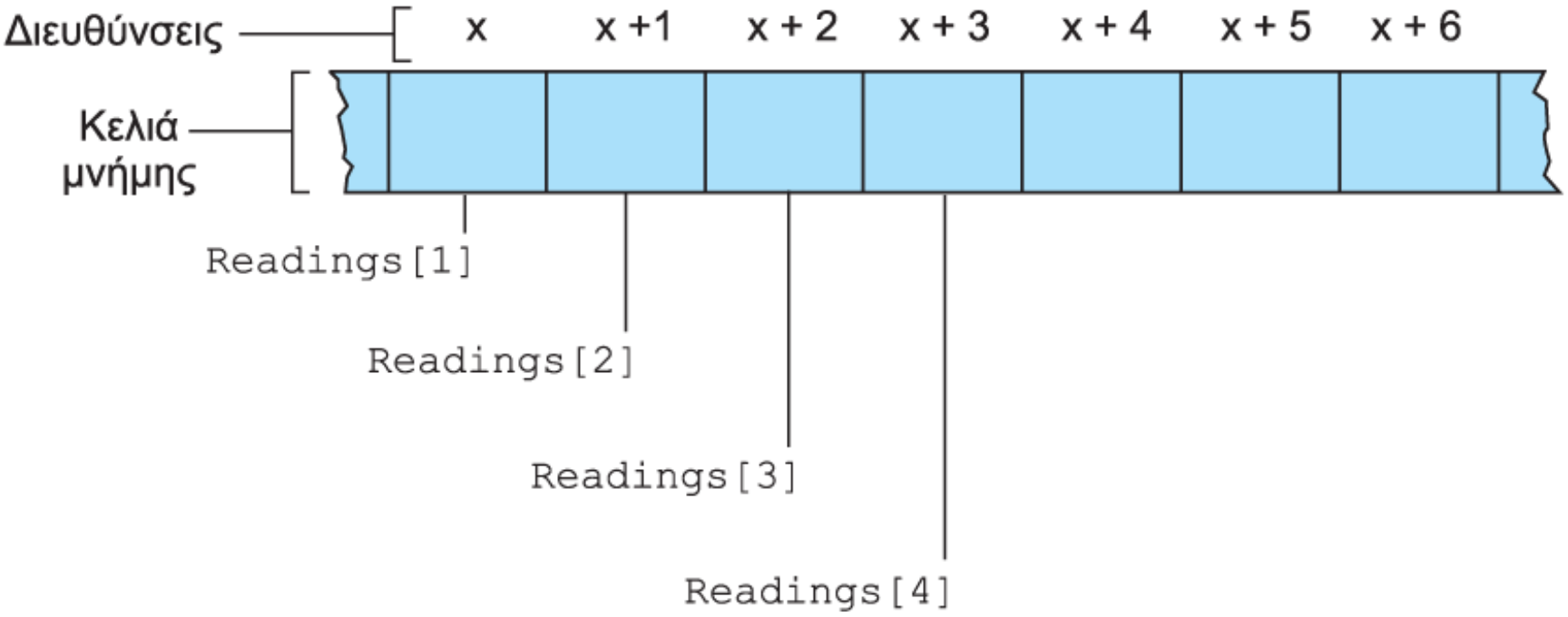
- δομή δεδομένων πίνακα : μια λογική οργάνωση δεδομένων στα κελιά ενός πίνακα
- κάθε κελί του πίνακα προσδιορίζεται μοναδικά από ένα συνδυασμό N >= 1 δεικτών
- διάσταση του πίνακα : το πλήθος δεικτών που προσδιορίζουν μοναδικά ένα συγκεκριμένο κελί του πίνακα

Μορφή μιας δομής πίνακα



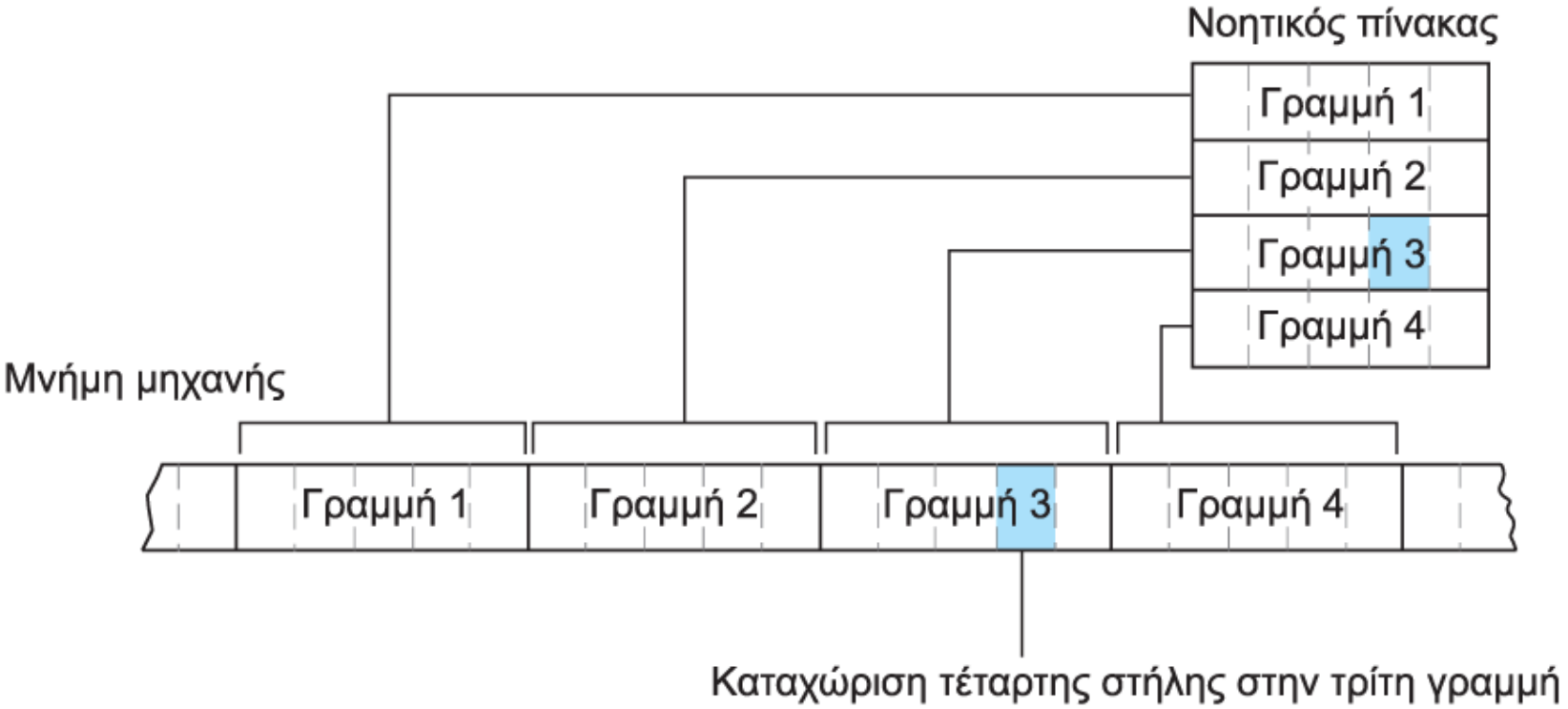


Αποθήκευση του πίνακα ενδείξεων θερμοκρασίας στη μνήμη με αρχή τη διεύθυνση x



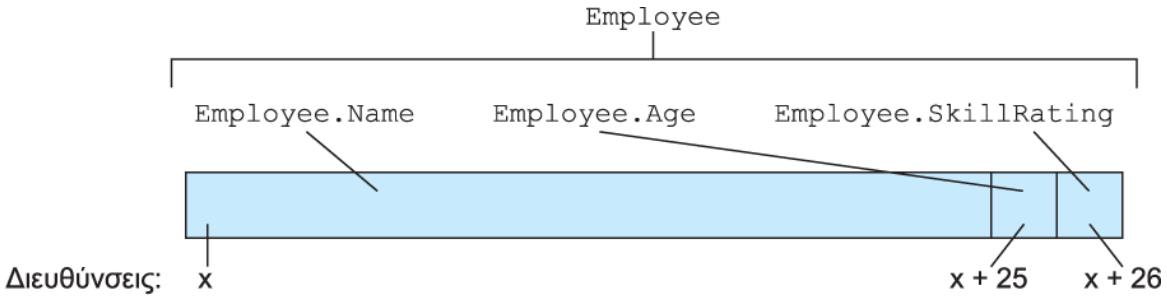
Η νιοστή ένδειξη είναι στη θέση μνήμης χ+(ν-1)

Ένας δισδιάστατος πίνακας με τέσσερις γραμμές και πέντε στήλες, αποθηκευμένος με διάταξη κατά γραμμές

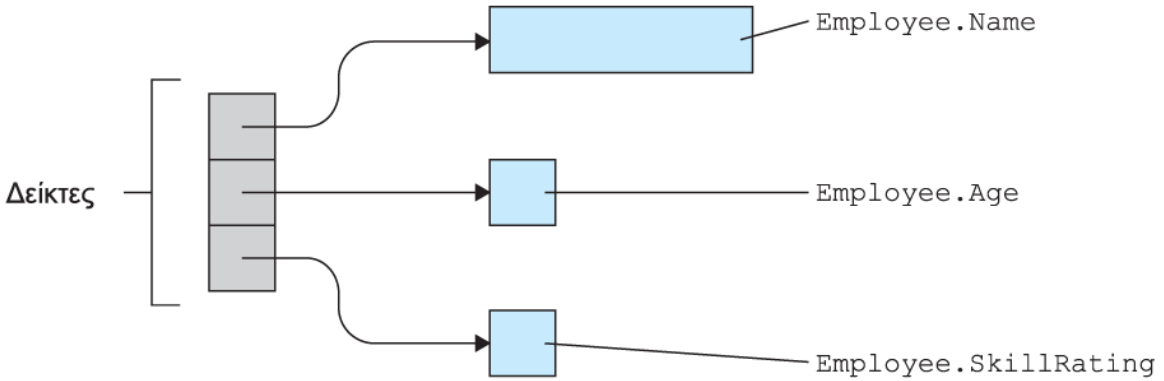


x + (c * (i -1)) + (j – 1) : χ διεύθυνση κελιού που περιέχει την καταχώρηση της πρώτης γραμμή, ι -1 γραμμές με c καταχωρήσεις η καθεμία και j-1 κελιά στη στήλη π.χ., το 4ο στοιχείο της 3ης γραμμής (5 στοιχείων) : χ + (5* (3-1)) + (4-1) =χ +13

Αποθήκευση του ετερογενούς πίνακα Employee



α. Πίνακας αποθηκευμένος σε συνεχόμενο τμήμα



β. Στοιχεία πίνακα αποθηκευμένα σε διαφορετικές θέσεις

Σάρωση μιας δομής πίνακα

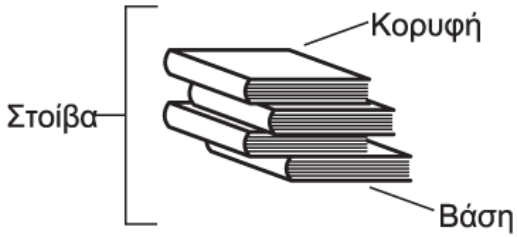
τμήμα σάρωση πίνακα (P, N, D_1, …, D_N)
{σάρωση ενός πίνακα p με N διαστάσεις και D_i θέσεις κατά τη διάσταση i}
start
    for a = 1 to D_1
        for b = 1 to D_2
            …
                επεξεργασία στοιχείου(P [a, b, …])
end

Η δομή δεδομένων λίστας

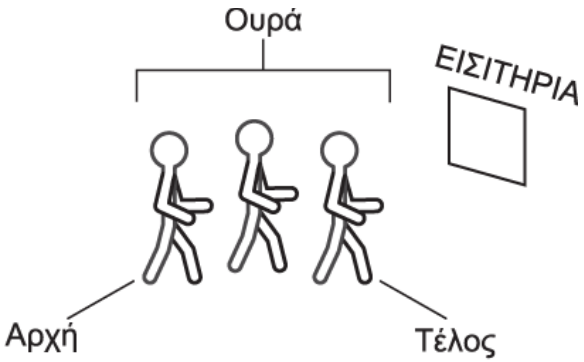
- δομή δεδομένων λίστας : μια λογική οργάνωση δεδομένων στις διαδοχικές θέσεις μιας λίστας
- κάθε στοιχείο μιας λίστας έχει 1 προηγούμενο και 1 επόμενο στοιχείο
- εξαίρεση : το πρώτο στοιχείο της λίστας δεν έχει προηγούμενο, το τελευταίο δεν έχει επόμενο
- έμμεση προσπέλαση ενός στοιχείου μέσω των προηγούμενων ή των επόμενων στοιχείων
- το μέγεθος μιας δομής λίστας δεν είναι προκαθορισμένο

Λίστες, στοίβες, και ουρές

α. Λίστα ονομάτων



β. Στοίβα βιβλίων



γ. Ουρά ατόμων

Ορολογία για Λίστες

- **Λίστα:** Μια συλλογή δεδομένων των οποίων οι καταχωρίσεις είναι διατεταγμένες σειριακά
- **Κεφαλή:** Η αρχή της λίστας
- **Ουρά:** Το τέλος της λίστας

Είδη δομών λίστας

- συνδεδεμένες λίστες : λίστες των οποίων κάθε στοιχείο έχει δείκτες προς το επόμενο ή και το προηγούμενό του
- απλά συνδεδεμένες λίστες
- διπλά συνδεδεμένες λίστες
- κυκλικά συνδεδεμένες λίστες

- ουρές
- στοίβες

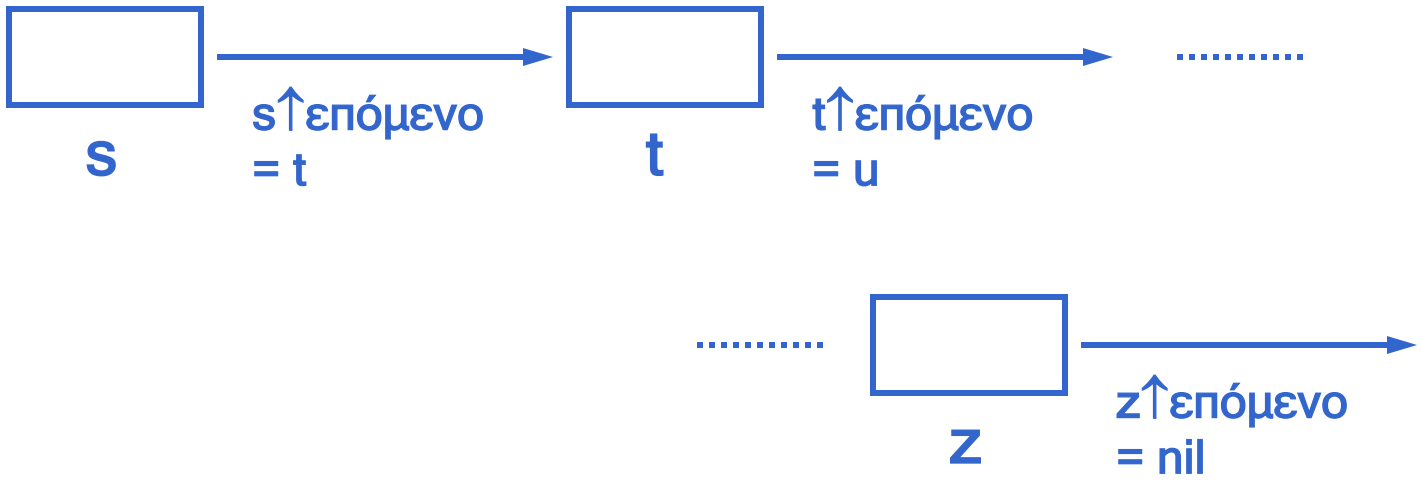
Δομή απλά συνδεδεμένης λίστας

- δομή δεδομένων απλά συνδεδεμένης λίστας : μια δομή λίστας στην οποία κάθε στοιχείο s έχει ένα δείκτη s↑επόμενο προς το επόμενο στοιχείο της λίστας

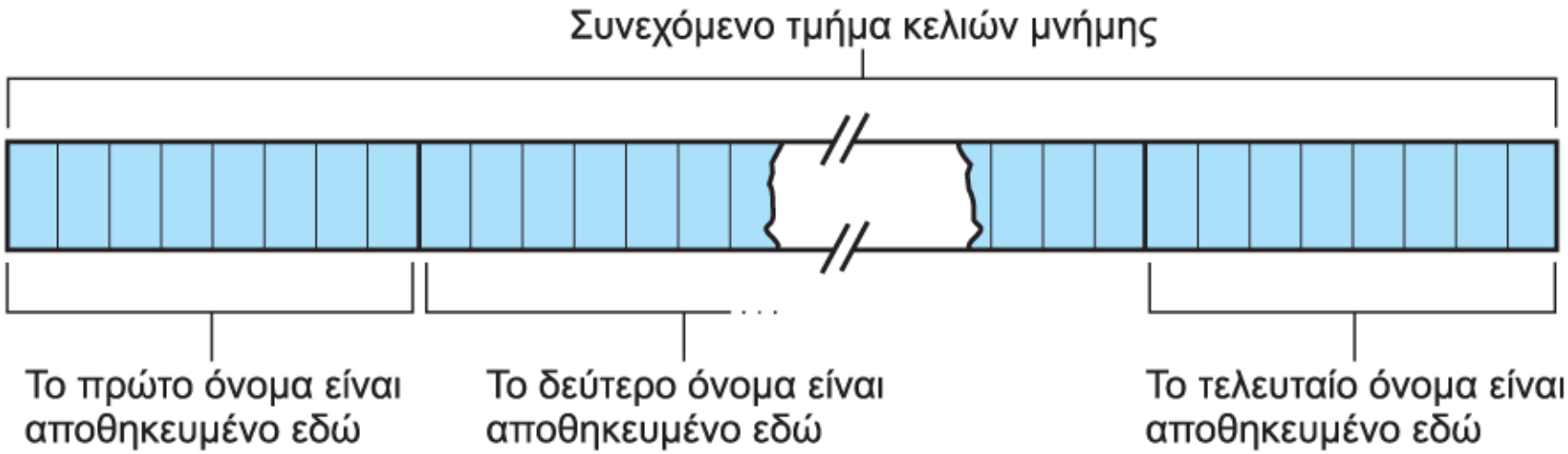
- η ειδική τιμή “nil” ισοδυναμεί με την κενό
- για το τελευταίο στοιχείο last ισχύει last↑επόμενο = nil

- εισαγωγή στοιχείου μπορεί να γίνει σε οποιαδήποτε θέση της λίστας, με αποκατάσταση των δεικτών
- διαγραφή στοιχείου μπορεί να γίνει από οποιαδήποτε θέση της λίστας, με αποκατάσταση των δεικτών

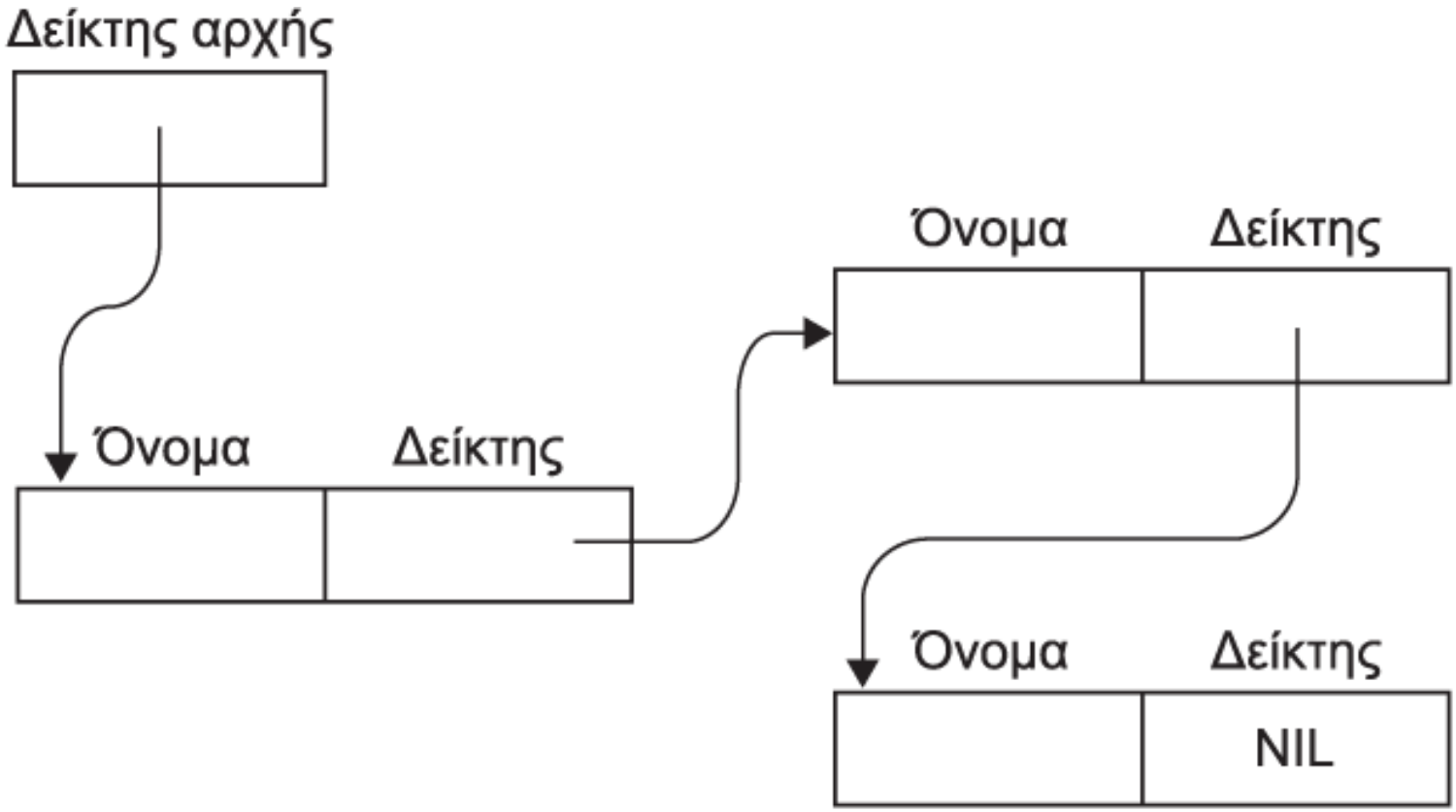
Μορφή μιας απλά συνδεδεμένης λίστας



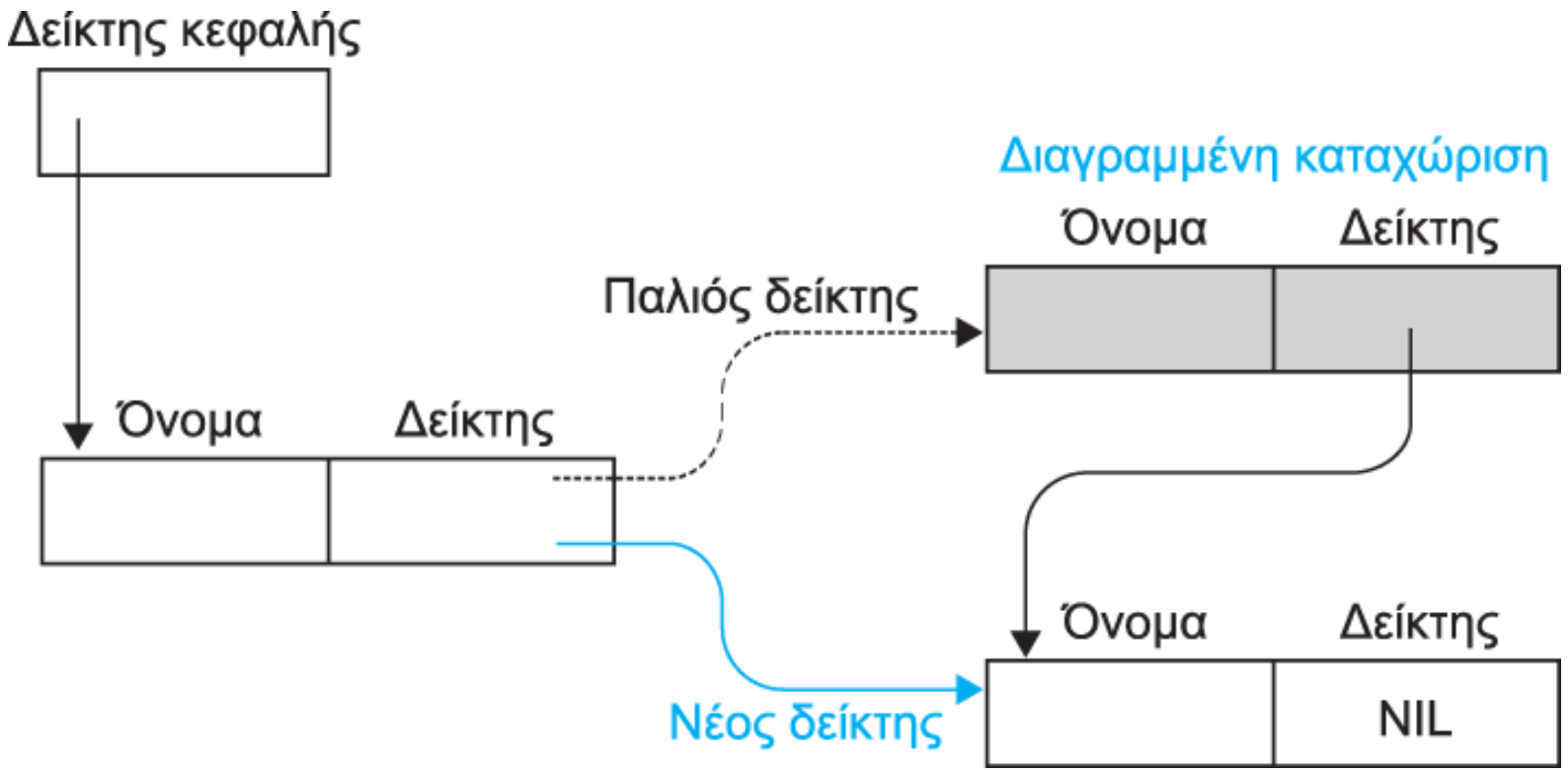
Ονόματα αποθηκευμένα στη μνήμη με τη μορφή συνεχόμενης λίστας



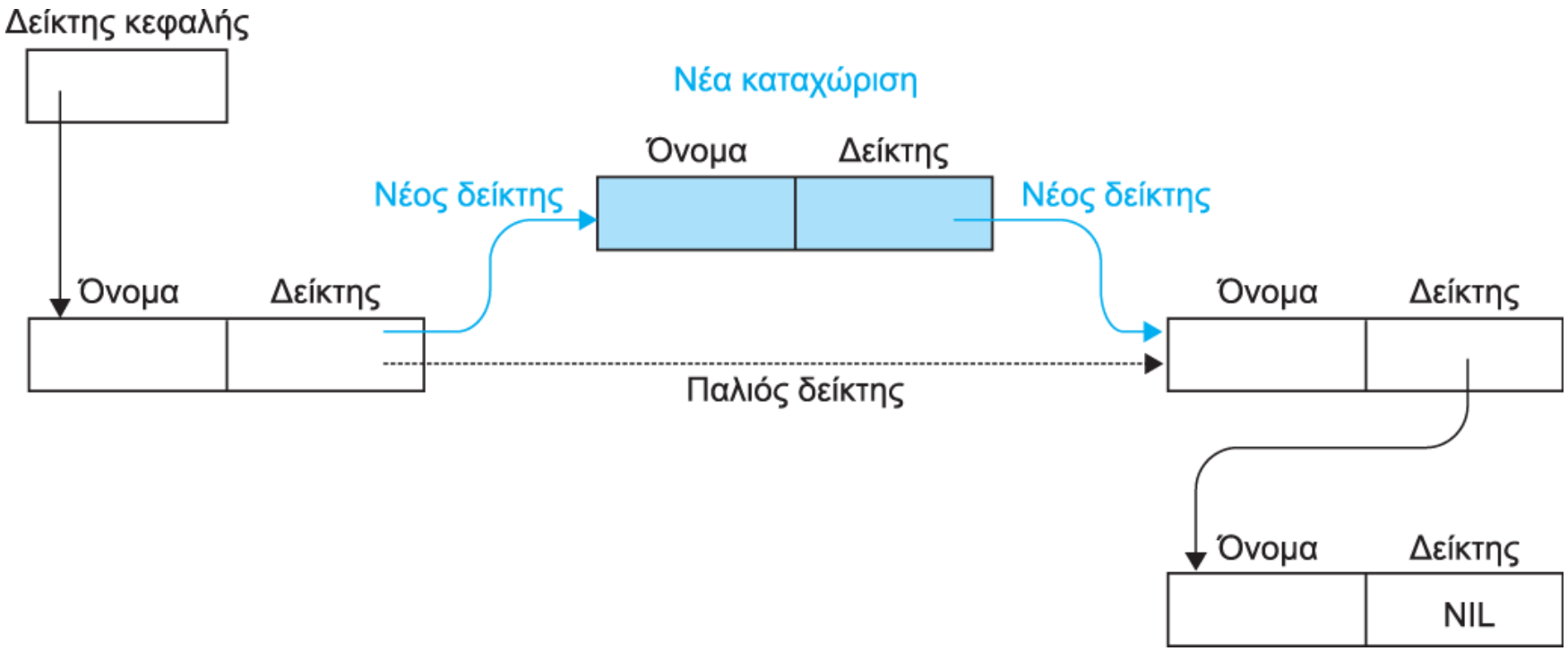
Η δομή μιας συνδεδεμένης λίστας



Διαγραφή καταχώρισης από συνδεδεμένη λίστα



Προσθήκη καταχώρισης σε συνδεδεμένη λίστα



Σάρωση μιας απλά συνδεδεμένης λίστας

```
module σάρωση απλά συνδεδεμένης λίστας (L)
start
    τρέχον στοιχείο := L
    while τρέχον στοιχείο <> nil
        επεξεργασία στοιχείου (τρέχον στοιχείο)
        τρέχον στοιχείο := τρέχον στοιχείο↑επόμενο
end
```

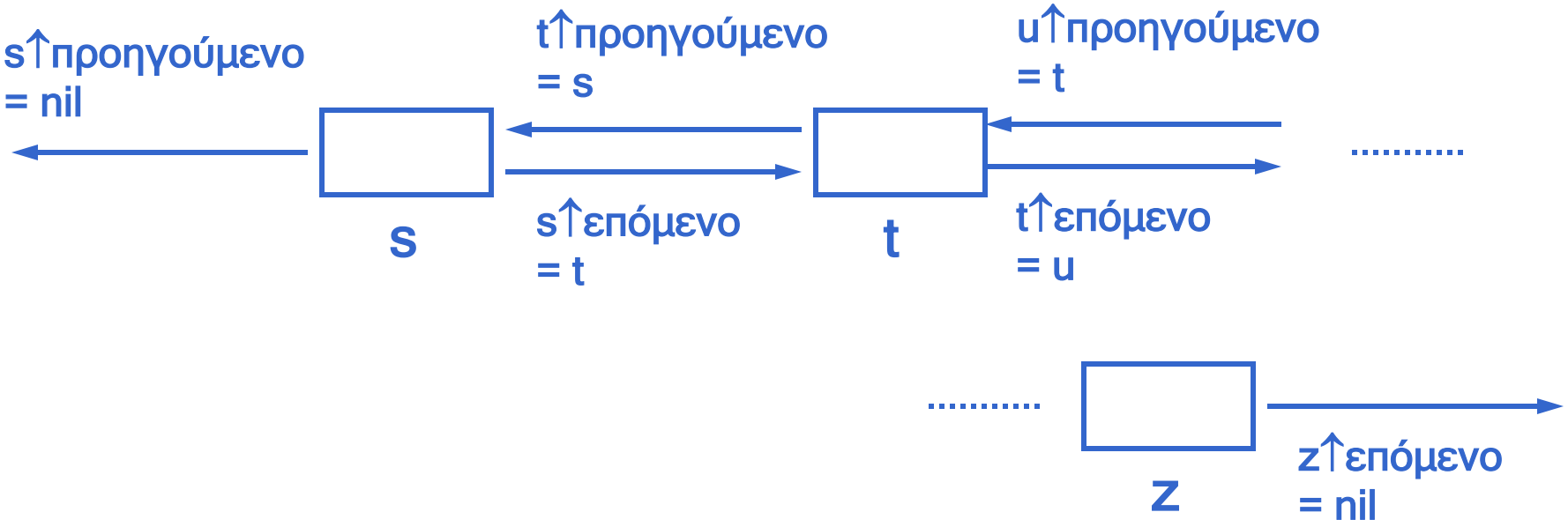
Δομή διπλά συνδεδεμένης λίστας

- δομή δεδομένων διπλά συνδεδεμένης λίστας : μια δομή λίστας στην οποία κάθε στοιχείο s έχει ένα δείκτη s↑επόμενο προς το επόμενο στοιχείο και ένα δείκτη s↑προηγούμενο προς το προηγούμενο στοιχείο της λίστας

- για το τελευταίο στοιχείο last ισχύει last↑επόμενο = nil
- για το πρώτο στοιχείο first ισχύει first↑προηγούμενο = nil

- εισαγωγή στοιχείου μπορεί να γίνει σε οποιαδήποτε θέση της λίστας, με αποκατάσταση των δεικτών
- διαγραφή στοιχείου μπορεί να γίνει από οποιαδήποτε θέση της λίστας , με αποκατάσταση των δεικτών

Μορφή μιας διπλά συνδεδεμένης λίστας



Σάρωση μιας διπλά συνδεδεμένης λίστας

```
τμήμα σάρωση διπλά συνδεδεμένης λίστας (L)
start
    τρέχον στοιχείο := L
    while τρέχον στοιχείο <> nil
        επεξεργασία στοιχείου (τρέχον στοιχείο)
        τρέχον στοιχείο := τρέχον στοιχείο↑επόμενο
end
```

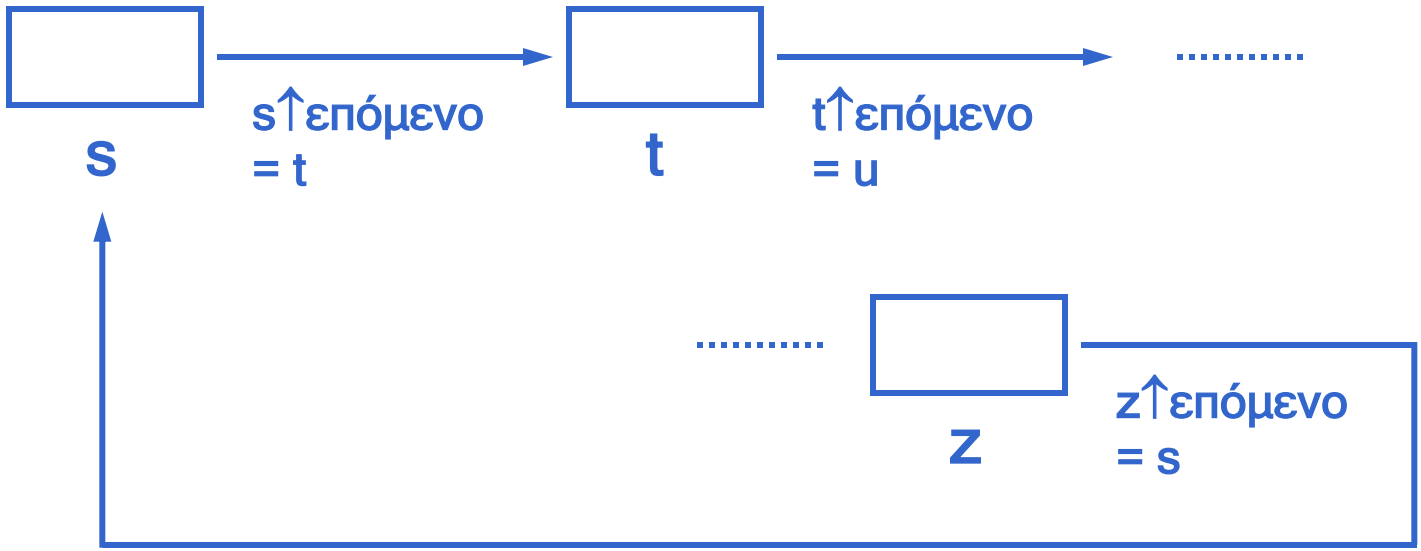
Δομή κυκλικά συνδεδεμένης λίστας

- δομή δεδομένων κυκλικά συνδεδεμένης λίστας : μια δομή απλά ή διπλά συνδεδεμένης λίστας στην οποία το τελευταίο στοιχείο last συνδέεται με το πρώτο στοιχείο first

- για το τελευταίο στοιχείο last ισχύει last↑επόμενο = first
- για το πρώτο στοιχείο first ισχύει first↑προηγούμενο = last

- εισαγωγή στοιχείου μπορεί να γίνει σε οποιαδήποτε θέση της λίστας, με αποκατάσταση των δεικτών
- διαγραφή στοιχείου μπορεί να γίνει από οποιαδήποτε θέση της λίστας , με αποκατάσταση των δεικτών

Μορφή μιας κυκλικά συνδεδεμένης λίστας



Σάρωση μιας κυκλικά συνδεδεμένης λίστας

```
start σάρωση κυκλικά συνδεδεμένης λίστας (L)
start
    if L <> nil then
    start
        τρέχον στοιχείο := L
        repeat
            επεξεργασία στοιχείου(τρέχον στοιχείο)
            τρέχον στοιχείο := τρέχον στοιχείο↑επόμενο
        until (τρέχον στοιχείο = L)
    end
end
```

Δομή ουράς

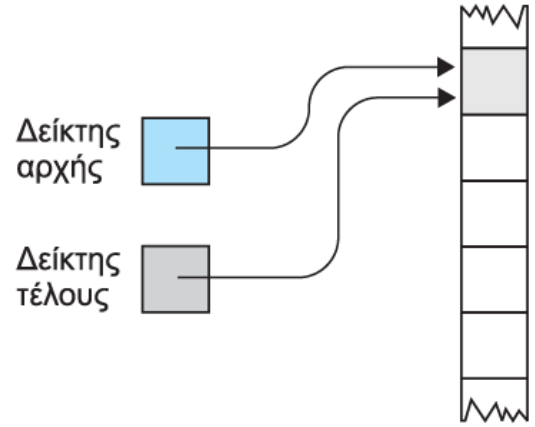
- δομή δεδομένων ουράς : μια δομή
  στην οποία
  α) εισαγωγή στοιχείου μπορεί να γίνει μόνο στην αρχή
  της λίστας
  β) διαγραφή στοιχείου μπορεί να γίνει μόνο από το τέλος
  της λίστας

- ειδική λειτουργία εισαγωγής στοιχείου : σπρώξε (Q, σ)
  {προσθέτει το στοιχείο σ στην αρχή της ουράς Q}

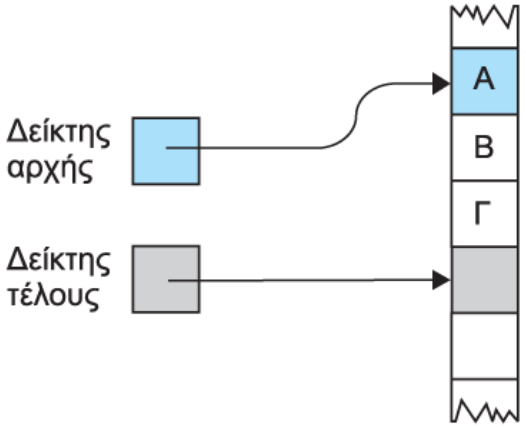
- ειδική λειτουργία διαγραφής στοιχείου : τράβηξε (Q, σ)
  {αφαιρεί το στοιχείο σ από το τέλος της ουράς Q}

Μορφή μιας δομής ουράς

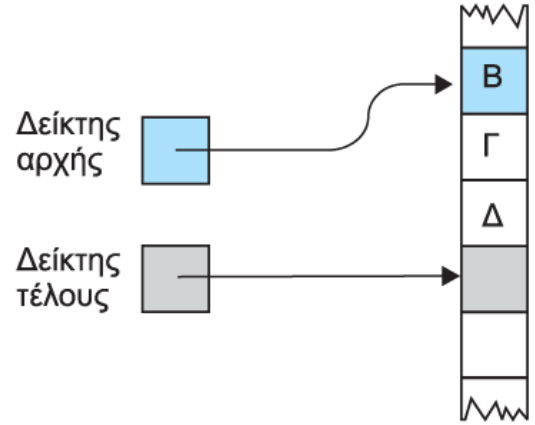
Μια υλοποίηση ουράς με δείκτες αρχής και τέλους



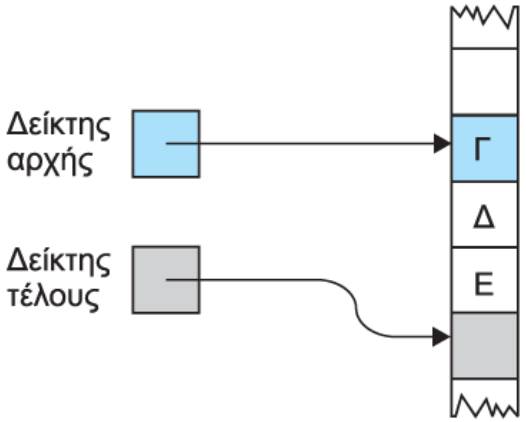
α. Άδεια ουρά



β. Μετά την εισαγωγή των καταχωρίσεων Α, Β, και Γ



γ. Μετά την αφαίρεση του Α και την εισαγωγή του Δ



δ. Μετά την αφαίρεση του Β και την εισαγωγή του Ε

Χρησιμότητα μιας δομής ουράς

- λειτουργία σύμφωνα με πειθαρχία FIFO :
  first-in $\Rightarrow$ first-out

- τήρηση προτεραιότητας άφιξης : αποθήκευση πληροφοριών για να εξυπηρετηθούν κάποιες αιτήσεις σύμφωνα με τη χρονική σειρά με την οποία έφτασαν

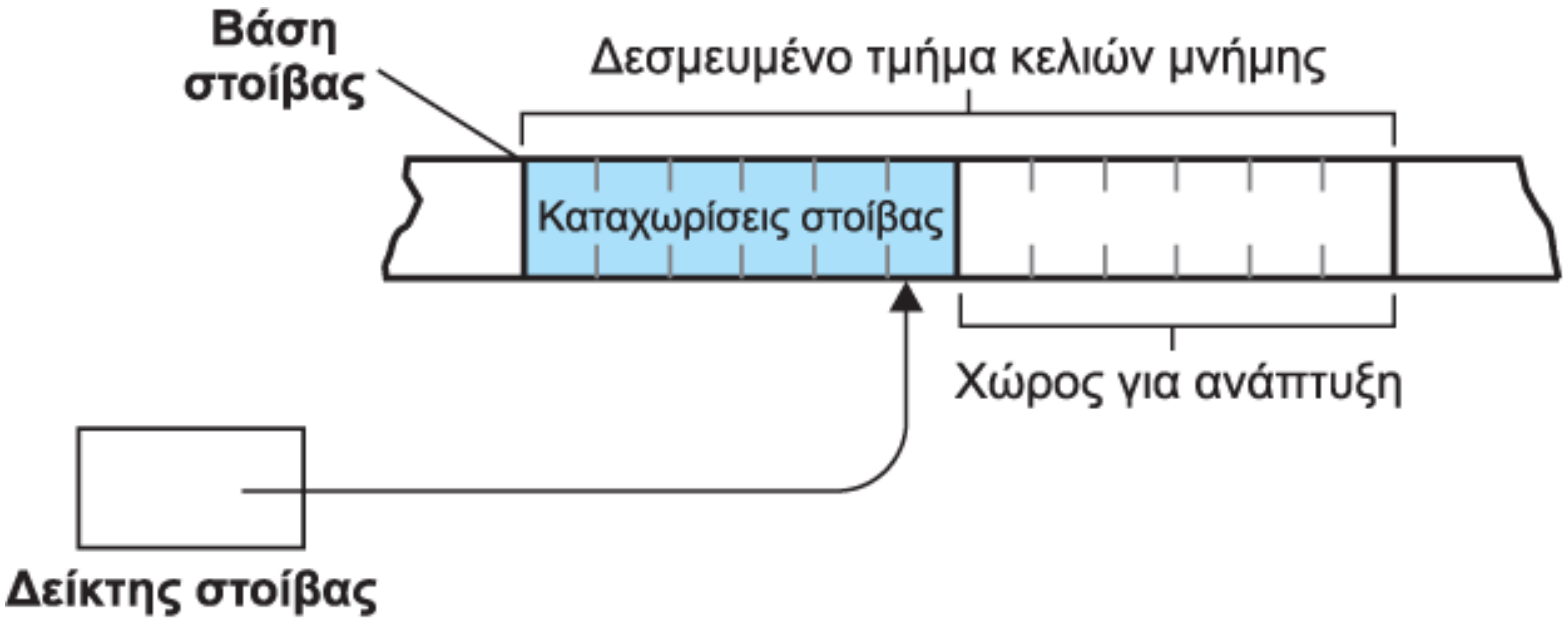
Δομή στοίβας

- δομή δεδομένων στοίβας : μια δομή
  στην οποία
  α) εισαγωγή στοιχείου μπορεί να γίνει μόνο στο τέλος
  της λίστας
  β) διαγραφή στοιχείου μπορεί επίσης να γίνει μόνο από
  το τέλος της λίστας

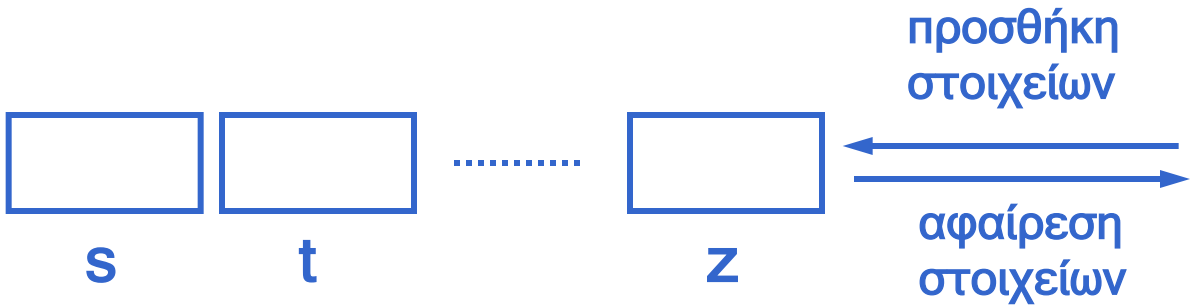
- ειδική λειτουργία εισαγωγής στοιχείου : σπρώξε (S, σ)
  {προσθέτει το στοιχείο σ στο τέλος της στοίβας S}

- ειδική λειτουργία διαγραφής στοιχείου : τράβηξε (S, σ)
  {αφαιρεί το στοιχείο σ από το τέλος της στοίβας S}

Μια στοίβα στη μνήμη



Μορφή μιας δομής στοίβας



Χρησιμότητα μιας δομής στοίβας

- λειτουργία σύμφωνα με πειθαρχία LIFO :
  last-in ⇒ first-out

- τήρηση προτεραιότητας διακοπής : αποθήκευση πληροφοριών για να συνεχίσει η εξυπηρέτηση αιτήσεων μετά από αλλεπάλληλες διακοπές

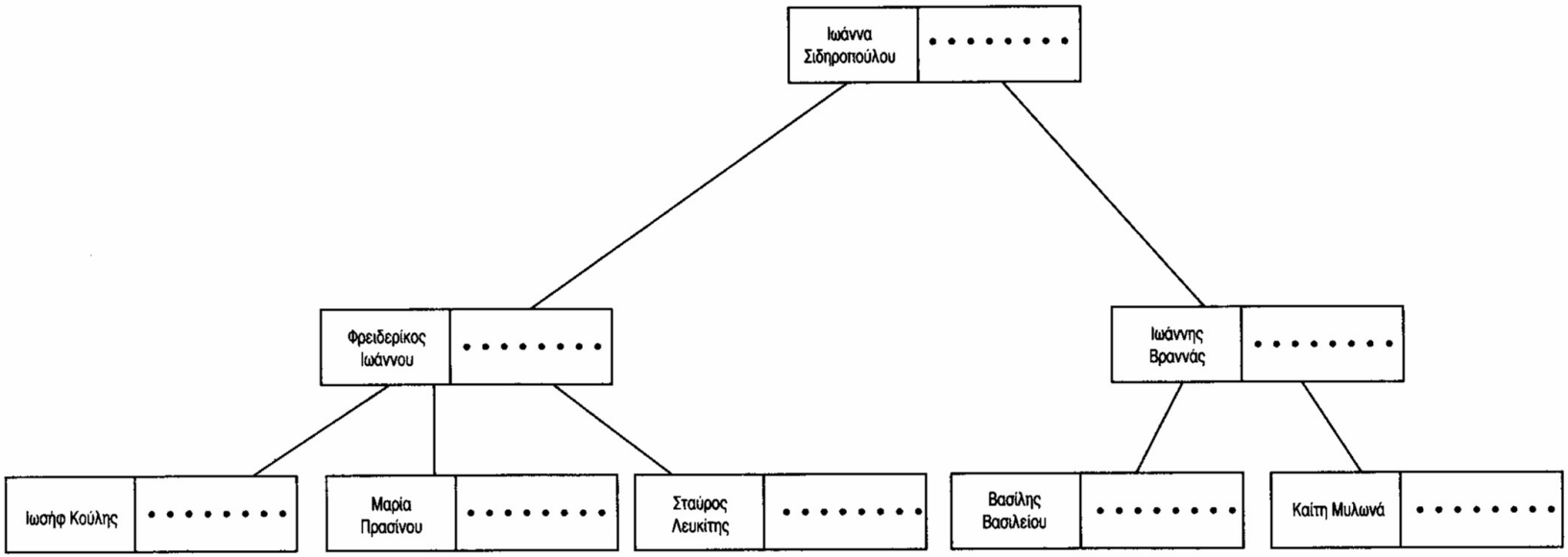
- τήρηση ιστορικού ενεργειών : αποθήκευση πληροφοριών για να ακυρωθούν κάποιες ενέργειες από την πλέον πρόσφατη προς τις προηγούμενες εάν διαπιστωθεί σφάλμα

Η έννοια του δέντρου

- δέντρο : κατευθυνόμενο ακυκλικό γράφημα στο οποίο
  α) υπάρχει ακριβώς μία κορυφή η οποία δεν έχει προηγούμενη
  β) όλες οι άλλες κορυφές έχουν ακριβώς μία προηγούμενη

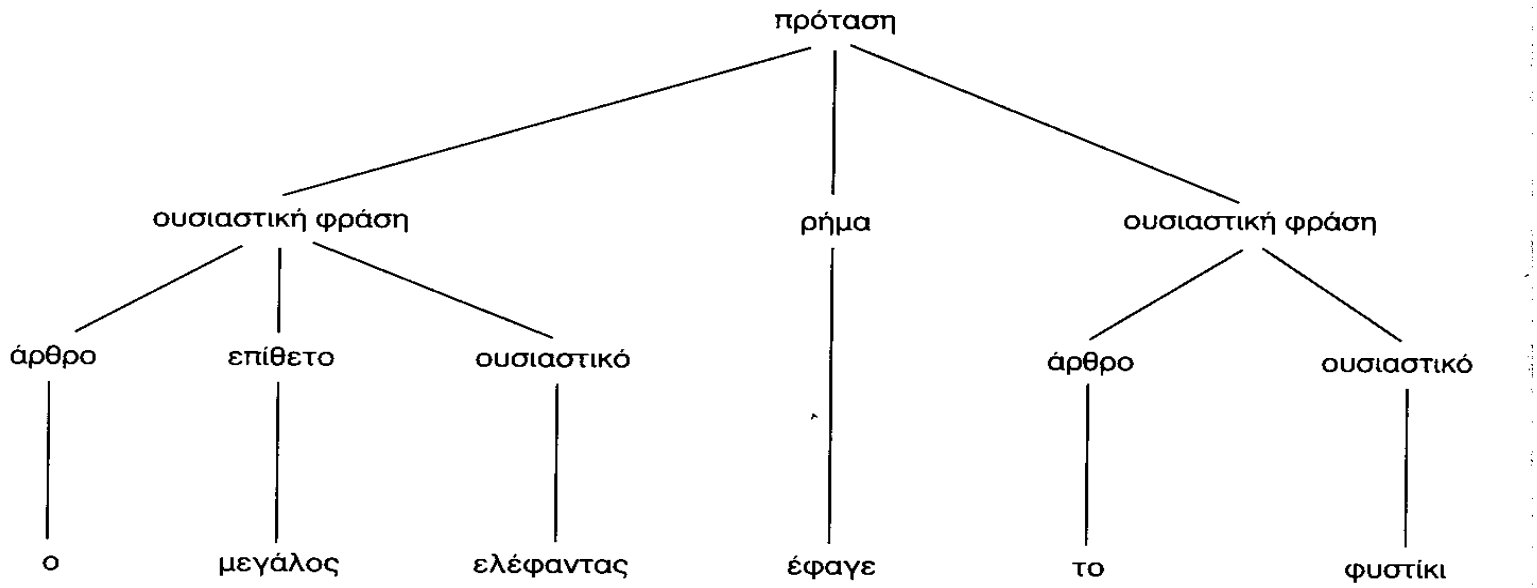
- γονέας μιας κορυφής : η μοναδική προηγούμενη κορυφή της
- παιδιά μιας κορυφής : οι N >= 0 επόμενες κορυφές της
- ρίζα του δέντρου : η μοναδική κορυφή του δέντρου η οποία δεν έχει γονέα
- φύλλα του δέντρου : οι N >= 1 κορυφές οι οποίες δεν έχουν παιδιά
- κλάδος του δέντρου : μια κορυφή και όλες οι άμεσα και έμμεσα επόμενές της, μαζί με τις αντίστοιχες ακμές

Δομή δεδομένων: Δέντρο



Κατάλληλο για την απεικόνιση λογικών σχέσεων μεταξύ εγγραφών που αντανακλούν τις οργανωτικές σχέσεις μεταξύ υπαλλήλων

Δομή δεδομένων: Δέντρο



Αναπαράσταση με δομή δέντρου μιας ελληνικής πρότασης

Χαρακτηριστικά ενός δέντρου

- βαθμός ενός δέντρου : το μέγιστο πλήθος N >= 2 παιδιών που μπορεί να έχει μια κορυφή ενός δέντρου
- ύψος ενός δέντρου : το πλήθος ακμών της μέγιστης διαδρομής μεταξύ της ρίζας του δέντρου και ενός φύλλου

Δυαδικά δέντρα

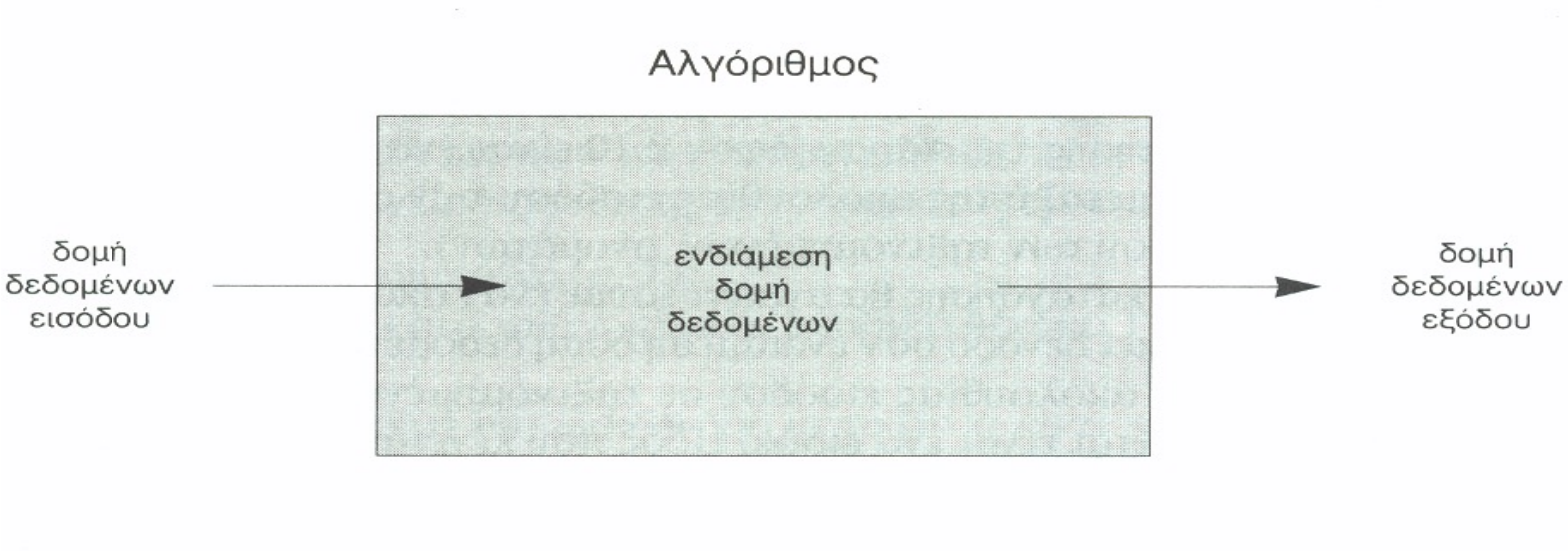
- δυαδικό δέντρο : ένα δέντρο βαθμού 2
- αριστερός και δεξιός υποκλάδος : οι δύο κλάδοι που ξεκινούν από μια κορυφή ενός δυαδικού δέντρου

Η δομή δεδομένων δυαδικού δέντρου

- έμμεση προσπέλαση ενός στοιχείου μέσω των προηγούμενων στοιχείων, ξεκινώντας από τη ρίζα

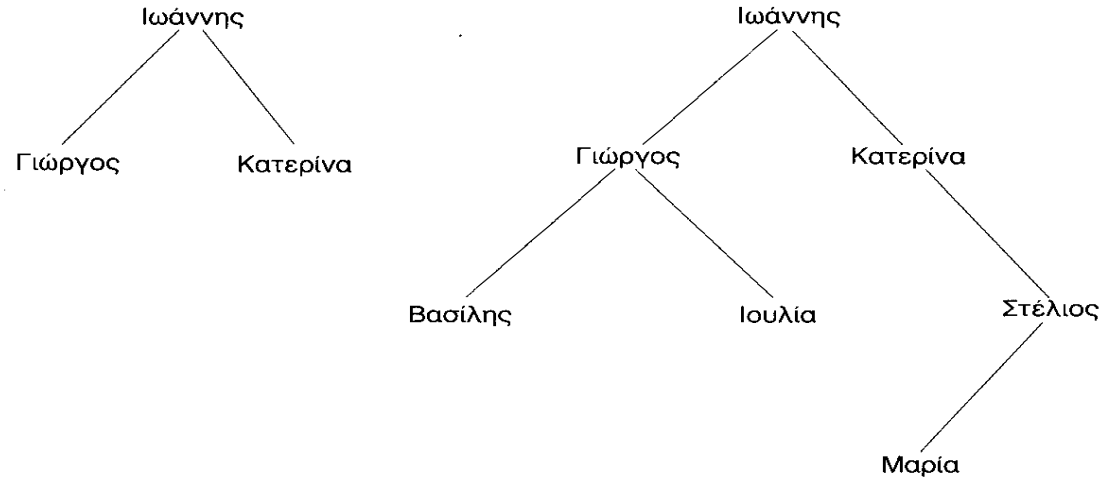
- η ειδική τιμή nil δηλώνει το κενό δυαδικό δέντρο
- ένα στοιχείο s ενός δυαδικού δέντρου έχει
  α) ένα δείκτη s↑αριστερά προς το αριστερό παιδί του s
  β) ένα δείκτη s↑δεξιά προς το δεξιό παιδί του s

Δομή δεδομένων: Δέντρο



Η δομή δεδομένων δυαδικού δέντρου

- Χρήσιμη δομή για ταξινόμηση αφού τα δύο κλαδιά μπορούν να παραστήσουν τις σχέσεις «πριν» και «μετά»
- Μετασχημάτισε την αταξινόμητη ακολουθία εισόδου σε ένα ταξινομημένο δυαδικό δέντρο & μετασχημάτισε το ταξινομημένο δυαδικό δέντρο σε μια ακολουθία εξόδου
- Ένα δυαδικό δέντρο είναι ταξινομημένο αν κάθε κόμβος έπεται από όλα τα δεδομένα του αριστερού κλαδιού, και προηγείται από όλα τα δεδομένα του δεξιού υπο-δέντρου του κόμβου



Η δομή δεδομένων δυαδικού δέντρου

L= Ιωάννης, Κατερίνα, Στέλιος, Γιώργος, Βασίλης, Ιουλία

Module build-tree (L,T)

{κτίζει ένα ταξινομημένο δυαδικό δέντρο Τ από μία λίστα εισόδου L από ονόματα}

Ξεκίνα από την αρχή της λίστας

While δεν εξαντλείται η L do

πρόσθεσε επόμενο όνομα στο δέντρο Τ

Η δομή δεδομένων δυαδικού δέντρου

Module πρόσθεσε όνομα (όνομα,Τ)

{προσθέτει ένα όνομα στο ταξινομημένο δυαδικό δέντρο Τ}

If Τ είναι άδειο

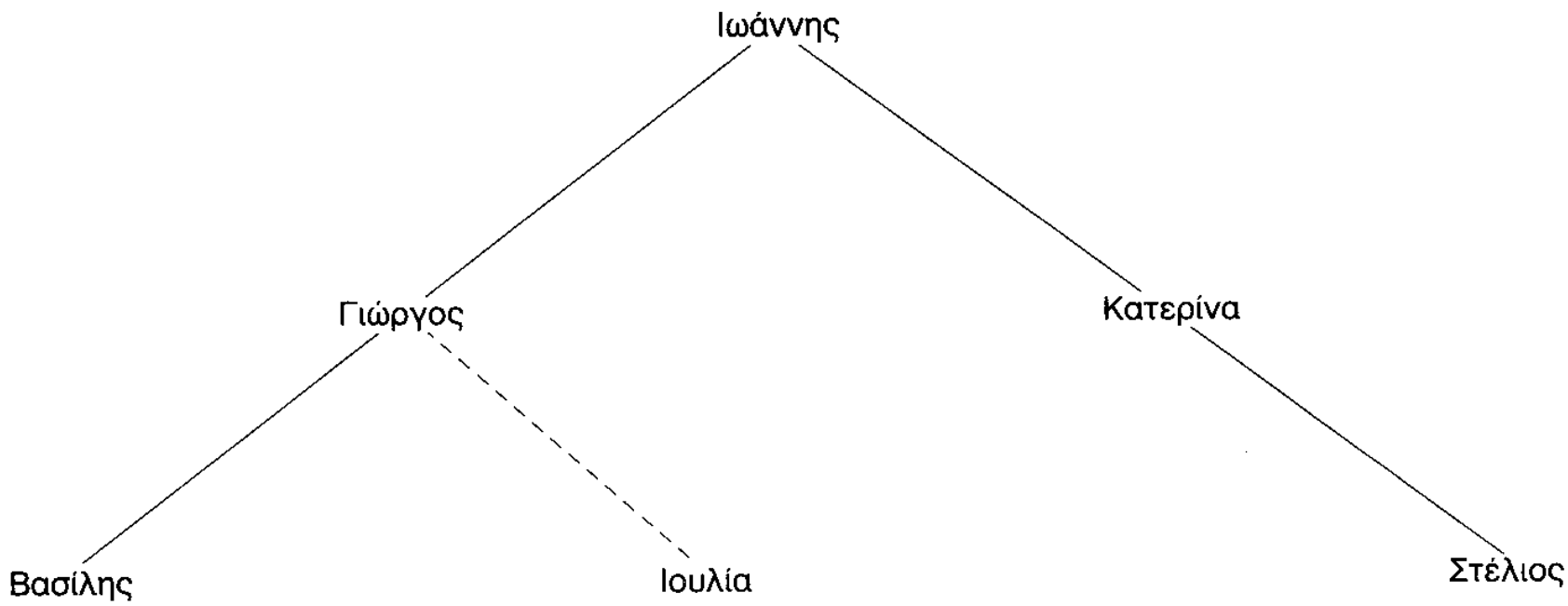
then δημιούργησε ένα νέο υποδέντρο με το όνομα σαν ρίζα

else if το όνομα προηγείται του ονόματος της ρίζας του Τ

then πρόσθεσε όνομα (όνομα, αριστερό υποδέντρο του Τ)

else πρόσθεσε όνομα (όνομα, δεξιό υποδέντρο του Τ)

Η δομή δεδομένων δυαδικού δέντρου



Η δομή δεδομένων δυαδικού δέντρου

Module έξοδος-δέντρου (Τ)

{Παράγει στην έξοδο όλους τους κόμβους του δυαδικού δέντρου Τ με διάταξη από αριστερά προς τα δεξιά}

If Τ όχι άδειο

then έξοδος-δέντρου (αριστερό υποδέντρο του Τ)

γράψε το όνομα της ρίζας του Τ

έξοδος-δέντρου (δεξιό υποδέντρο του Τ)

Η δομή δεδομένων δυαδικού δέντρου

- Ο πλήρης αλγόριθμος ταξινόμησης

Module ταξινόμηση L (T)
{Ταξινόμηση μιας λίστας σε αλφαβητική διάταξη}

Ξεκίνα με ένα άδειο δέντρο T
Κτίσε-δέντρο(L, T)
Έξοδος-δέντρου(T)