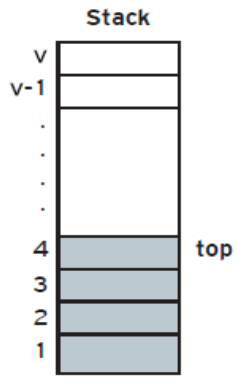
Στοίβα (stack), ονομάζεται μια δομή δεδομένων το σύνολο των στοιχείων της οποίας είναι διατεταγμένο με τέτοιο τρόπο, ώστε τα στοιχεία που βρίσκονται στην κορυφή της στοίβας λαμβάνονται πρώτα, ενώ αυτά που βρίσκονται στο βάθος της στοίβας λαμβάνονται τελευταία.

Η παραπάνω μέθοδος ονομάζεται **Τελευταίο Μέσα, Πρώτο Έξω ή LIFO** (=Last In First Out).

Οι **κύριες λειτουργίες** σε μια στοίβα είναι δύο:

1. Η **ώθηση** (push) στοιχείου στην κορυφή της στοίβας.
   Στη διαδικασία της ώθησης ελέγχουμε αν η στοίβα είναι γεμάτη.
   Στην περίπτωση που προσπαθήσουμε να «προσθέσουμε» ένα στοιχείο σε μια ήδη γεμάτη στοίβα, έχουμε **υπερχείλιση** (overflow) της στοίβας.
2. Η **απώθηση** (pop) στοιχείου από τη στοίβα.
   Στη διαδικασία της απώθησης ελέγχουμε αν υπάρχει ένα τουλάχιστον στοιχείο στη στοίβα.
   Στην περίπτωση που προσπαθήσουμε να «αφαιρέσουμε» ένα στοιχείο από μία κενή στοίβα, έχουμε **υποχείλιση** (underflow) της στοίβας.



Εικόνα 1. 1. Υλοποίηση Στοίβας

- Η **Υλοποίηση στοίβας στη ΓΛΩΣΣΑ** γίνεται με χρήση **μονοδιάστατου πίνακα** και **μια** βοηθητική **μεταβλητή (top),** που δείχνει το στοιχείο που τοποθετήθηκε τελευταίο στην κορυφή της στοίβας.
- Ως αρχική τιμή της μεταβλητής **top** θεωρούμε το μηδέν.

Να αναπτύξετε τμήμα προγράμματος σε ΓΛΩΣΣΑ, που πραγματοποιεί ώθηση στοιχείου στην κορυφή της στοίβας με χρήση μονοδιάστατου πίνακα Α, 10 θέσεων.

```
ΓΡΑΨΕ 'Δώσε στοιχείο για να εισαχθεί στη στοίβα Α:'
ΔΙΑΒΑΣΕ στοιχείο
```

ΣΥΝΘΗΚΗ ΕΛΕΓΧΟΥ

ΓΕΜΑΤΗΣ ΣΤΟΙΒΑΣ

Top=N

```
ΑΝ Top=10 ΤΟΤΕ
        ΓΡΑΨΕ 'ΓΕΜΑΤΗ ΣΤΟΙΒΑ'
ΑΛΛΙΩΣ
        Top←Top+1
        Α[Top]←στοιχείο
ΤΕΛΟΣ_ΑΝ
```

Να αναπτύξετε τμήμα προγράμματος σε ΓΛΩΣΣΑ που πραγματοποιεί την απώθηση στοιχείου από στοίβα με χρήση ενός μονοδιάστατου πίνακα Α, 10 θέσεων

```
ΑΝ Top=0 ΤΟΤΕ
        ΓΡΑΨΕ 'ΑΔΕΙΑ ΣΤΟΙΒΑ'
ΑΛΛΙΩΣ
        ΓΡΑΨΕ Α[Top]
        Top←Top-1
ΤΕΛΟΣ_ΑΝ
```

ΣΥΝΘΗΚΗ ΕΛΕΓΧΟΥ

ΑΔΕΙΑΣ ΣΤΟΙΒΑΣ

Top=0

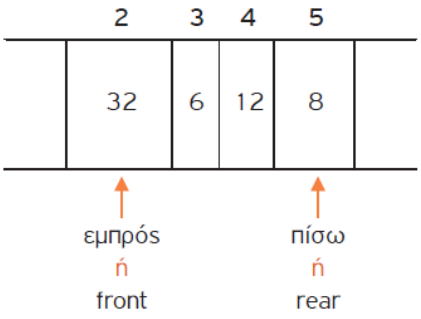
Ουρά (Queue), ονομάζεται μια δομή δεδομένων το σύνολο των στοιχείων της οποίας είναι διατεταγμένο με τέτοιο τρόπο, ώστε τα στοιχεία που τοποθετήθηκαν πρώτα στην ουρά να λαμβάνονται επίσης πρώτα.

Η παραπάνω μέθοδος ονομάζεται **Πρώτο Μέσα, Πρώτο Έξω ή FIFO** (=First In First Out).

Οι **κύριες λειτουργίες** που εκτελούνται σε μια ουρά είναι δύο:

1. Η **εισαγωγή** (enqueue) στοιχείου στο πίσω άκρο της ουράς.
2. Η **εξαγωγή** (dequeue) στοιχείου από το εμπρός άκρο της ουράς.

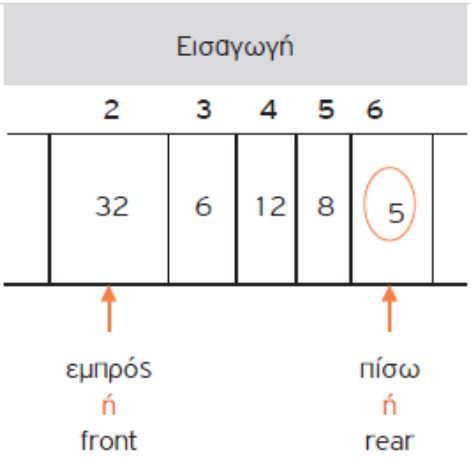
- Η **Υλοποίηση ουράς στη ΓΛΩΣΣΑ** γίνεται με **χρήση μονοδιάστατου πίνακα** και **δύο** μεταβλητές, την **front** (ή εμπρός) που δείχνει τη θέση του **1ου** στοιχείου της ουράς και την **rear** (ή πίσω) που δείχνει τη θέση του **τελευταίου** στοιχείου.
- Ως αρχικές τιμές των μεταβλητών rear και front θεωρούμε το μηδέν.



- Η **εισαγωγή** ενός νέου στοιχείου γίνεται από το πίσω άκρο της ουράς και η τιμή της μεταβλητής rear αλλάζει ως εξής:

rear← rear+1

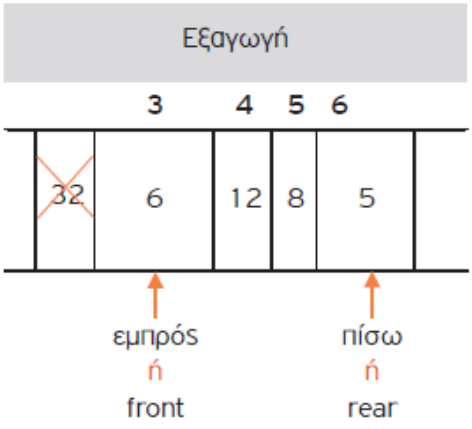
Κατά την εισαγωγή, πρώτα αυξάνουμε τον δείκτη rear κατά ένα και μετά εισάγουμε το στοιχείο στον πίνακα.



- Η **εξαγωγή** ενός στοιχείου γίνεται από το εμπρός άκρο της ουράς και η τιμή της μεταβλητής front αλλάζει ως εξής:

front ← front +1

Κατά την εξαγωγή ενός στοιχείου, αυξάνεται ο δείκτης front κατά ένα (δείχνει στην επόμενη θέση του πίνακα) χωρίς στην πραγματικότητα να γίνεται καμία παρέμβαση στα περιεχόμενα του πίνακα (χωρίς να διαγράφεται κάποιο στοιχείο).



Παράδειγμα 2 – Εισαγωγή στοιχείου σε ουρά με χρήση μονοδιάστατου πίνακα

Να αναπτύξετε τμήμα προγράμματος σε ΓΛΩΣΣΑ που υλοποιεί την εισαγωγή στοιχείου σε ουρά, με χρήση μονοδιάστατου πίνακα Α, 10 θέσεων.

Απάντηση

Κώδικας σε ΓΛΩΣΣΑ [1.4]

```
ΓΡΑΨΕ 'Δώσε στοιχείο για εισαγωγή στην ουρά Α:'
ΔΙΑΒΑΣΕ στοιχείο
ΑΝ rear = 10 ΤΟΤΕ
   ΓΡΑΨΕ 'Γεμάτη ουρά'
ΑΛΛΙΩΣ_ΑΝ (front = 0 ΚΑΙ rear = 0) ΤΟΤΕ
   front<- 1
   rear<-1
   A[rear]<-στοιχείο
ΑΛΛΙΩΣ
      rear <- rear + 1
      A[rear]<-στοιχείο
ΤΕΛΟΣ_ΑΝ
```

ΣΥΝΘΗΚΗ ΕΛΕΓΧΟΥ

ΓΕΜΑΤΗΣ ΟΥΡΑΣ

Rear = N

Παράδειγμα 3 – Εξαγωγή στοιχείου από ουρά με χρήση μονοδιάστατου πίνακα

Να αναπτύξετε τμήμα προγράμματος σε ΓΛΩΣΣΑ που υλοποιεί την εξαγωγή στοιχείου από ουρά, με χρήση μονοδιάστατου πίνακα Α, 10 θέσεων.

Απάντηση

Κώδικας σε ΓΛΩΣΣΑ [1.5]

```
ΑΝ (front = 0 ΚΑΙ rear = 0) ΤΟΤΕ
    ΓΡΑΨΕ 'Άδεια ουρά'
ΑΛΛΙΩΣ ΑΝ (front = rear ) ΤΟΤΕ
    ΓΡΑΨΕ 'Εξάγεται το στοιχείο:',A[front]
    front <-  0
    rear  <-  0
ΑΛΛΙΩΣ
    ΓΡΑΨΕ 'Εξάγεται το στοιχείο:',A[front]
    front <- front + 1
ΤΕΛΟΣ_ΑΝ
```

ΣΥΝΘΗΚΗ ΕΛΕΓΧΟΥ

ΑΔΕΙΑΣ ΟΥΡΑΣ

Front=0 ΚΑΙ Rear = 0

ΣΥΝΘΗΚΗ ΕΛΕΓΧΟΥ

ΕΝΑΣ ΠΕΛΑΤΗΣ ΣΤΗΝ ΟΥΡΑ

Front=Rear

ΟΛΙΣΘΗΣΗ ΟΥΡΑΣ ΠΡΟΣ ΤΟ ΕΜΠΡΟΣ ΑΚΡΟ

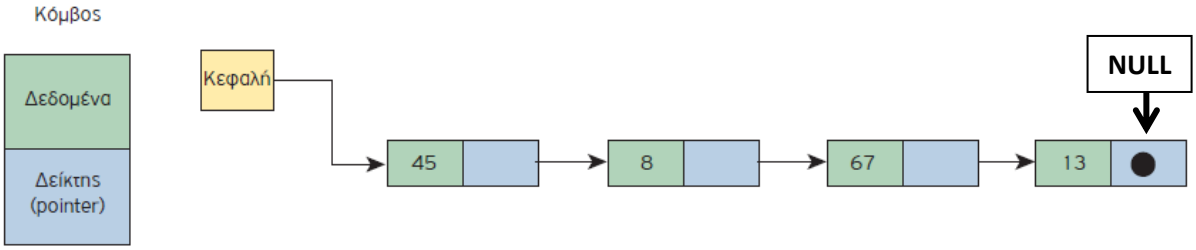
Κ←0
ΓΙΑ i ΑΠΟ **FRONT** ΜΕΧΡΙ **REAR**
    Κ←Κ+1
    ΟΥΡΑ[Κ]←ΟΥΡΑ[i]
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
FRONT←1
REAR←Κ

ΠΟΣΟΙ ΠΕΛΑΤΕΣ ΕΙΝΑΙ ΣΤΗΝ ΟΥΡΑ

REAR–FRONT +1

Μία (απλά) **συνδεδεμένη λίστα (linked list)** είναι ένα σύνολο κόμβων διατεταγμένων γραμμικά (ο ένας μετά τον άλλο). Κάθε κόμβος περιέχει εκτός από τα **δεδομένα** του και έναν **δείκτη** που δείχνει προς τον επόμενο κόμβο.
Ο δείκτης του τελευταίου κόμβου δε δείχνει σε κάποιον κόμβο (δείκτης στο κενό). Για να το δηλώσουμε αυτό λέμε ότι το πεδίο δείκτη του τελευταίου κόμβου έχει την τιμή **NULL**.

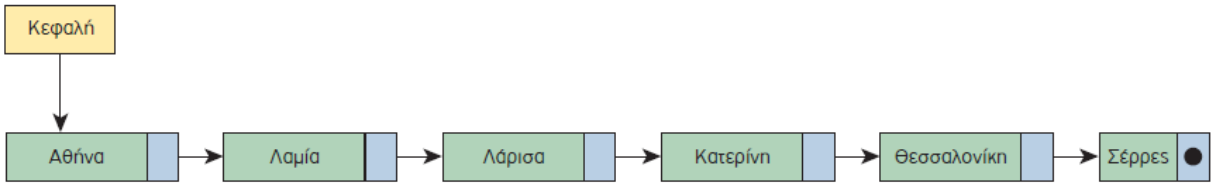
Για να προσπελάσουμε τους κόμβους της λίστας χρειάζεται να γνωρίζουμε τη διεύθυνση (θέση στη μνήμη) του πρώτου κόμβου της λίστας. Η διεύθυνση αυτή αποθηκεύεται σε μία ειδική μεταβλητή που την ονομάζουμε συνήθως **Κεφαλή (Head)**.



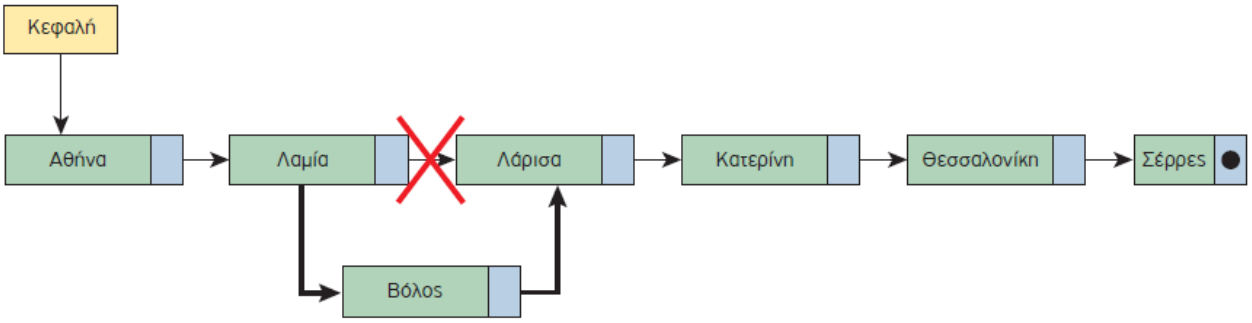
Εικόνα 1.3.2. α: Αναπαράσταση κόμβου β: Αναπαράσταση απλά συνδεδεμένης λίστας με 4 κόμβους

Πρόσβαση στους κόμβους μιας συνδεδεμένης λίστας

Οι κόμβοι μιας (απλά) συνδεδεμένης λίστας είναι διατεταγμένοι σε μια συγκεκριμένη σειρά, χωρίς αυτό να σημαίνει ότι αποθηκεύονται σε συνεχόμενες θέσεις στη μνήμη. Αντίθετα, είναι διασκορπισμένοι σε όλη τη μνήμη και η σύνδεση μεταξύ τους γίνεται μέσω των δεικτών. Έχουμε άμεση πρόσβαση μόνο στον πρώτο κόμβο της λίστας. Επομένως, για να εντοπίσουμε κάποιον από τους ενδιάμεσους κόμβους, πρέπει να ξεκινήσουμε από τον πρώτο κόμβο της λίστας και να ακολουθήσουμε τους δείκτες με τη σειρά, μέχρι να φτάσουμε στον επιθυμητό κόμβο.

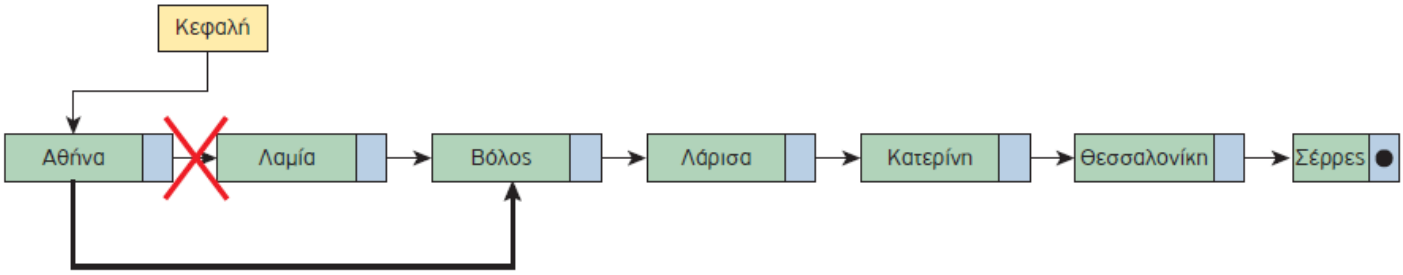


Εικόνα 1.3.4. Μια απλά συνδεδεμένη λίστα



Εικόνα 1.3.5. Εισαγωγή νέου κόμβου στη λίστα

Οι απαιτούμενες ενέργειες για την εισαγωγή (παρεμβολή) του νέου κόμβου ΒΟΛΟΣ είναι ο δείκτης του προηγούμενου κόμβου να δείχνει τον νέο κόμβο και ο δείκτης του νέου κόμβου να δείχνει τον επόμενο κόμβο.

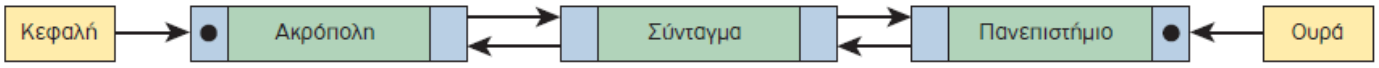


Εικόνα 1.3.6. Διαγραφή κόμβου από τη λίστα

Για τη διαγραφή ενός κόμβου ΛΑΜΙΑ αρκεί ν' αλλάξει τιμή ο δείκτης του προηγούμενου κόμβου και να δείχνει πλέον στον επόμενο αυτού που διαγράφεται.

Μια λίστα μπορεί να είναι **απλά συνδεδεμένη**, δηλαδή να μπορούμε να κινηθούμε προς μία μόνο κατεύθυνση, ξεκινώντας από τον πρώτο κόμβο και μετακινούμενοι προς τον τελευταίο,

ή να είναι **διπλά συνδεδεμένη (doubly linked list)**, δηλαδή, να μπορούμε να τη διατρέξουμε και προς τις δύο κατευθύνσεις. Η χρήση του δεύτερου δείκτη προσφέρει τη δυνατότητα ξεκινώντας από οποιοδήποτε κόμβο της λίστας να μπορούμε να διαβάσουμε τη λίστα και προς τις δυο κατευθύνσεις.



Εικόνα 1.3.8. Οι σταθμοί του μετρό - Μια διπλά συνδεδεμένη λίστα

Βασικές πράξεις των συνδεδεμένων λιστών

Οι βασικές πράξεις των συνδεδεμένων λιστών είναι οι παρακάτω:

- Εισαγωγή κόμβου στη λίστα (εισαγωγή κόμβου στην αρχή, στο τέλος της λίστας ή ενδιάμεσα).
- Διαγραφή κόμβου από τη λίστα (διαγραφή από την αρχή, το τέλος της λίστας ή ενδιάμεσα).
- Έλεγχος για το αν η λίστα είναι κενή.
- Αναζήτηση κόμβου για την εύρεση συγκεκριμένου στοιχείου.
- Διάσχιση της λίστας και προσπέλαση των στοιχείων της (π.χ. εκτύπωση των δεδομένων που περιέχονται σε όλους τους κόμβους της λίστας).

<u>Διαφορές Λίστας σε σχέση με τον Πίνακα – Πλεονεκτήματα – Μειονεκτήματα</u>

<u>3 σημαντικές διαφορές:</u>

• Ο πίνακας θεωρείται μια δομή τυχαίας προσπέλασης, σε αντίθεση με μια λίστα που είναι στην ουσία μια δομή ακολουθιακής ή σειριακής προσπέλασης.
• Ο πίνακας έχει σταθερό μέγεθος, το οποίο δηλώνεται εξαρχής κατά την υλοποίηση, σε αντίθεση με τη λίστα που το μέγεθός της μπορεί να μεταβάλλεται καθώς εισέρχονται νέοι κόμβοι στη λίστα ή διαγράφονται κάποιοι άλλοι.
• Οι κόμβοι της λίστας αποθηκεύονται σε μη συνεχόμενες θέσεις μνήμης σε αντιδιαστολή με τους πίνακες, όπου τα στοιχεία αποθηκεύονται σε συνεχόμενες θέσεις μνήμης.

<u>3 πλεονεκτήματα των λιστών (έναντι των πινάκων)</u> :

• Το δυναμικό τους μέγεθος,
• η ευκολία εισαγωγής και διαγραφής από οποιοδήποτε μέρος της λίστας, καθώς και
• η μη αναγκαιότητα δήλωσης του μεγέθους τους.

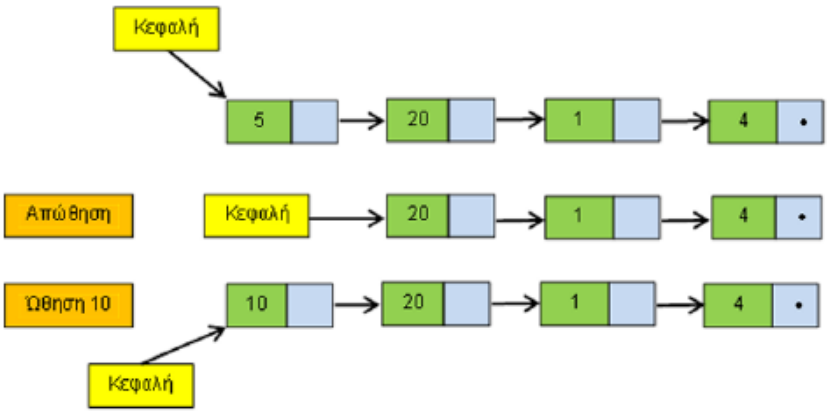
2 μειονεκτήματα των λιστών (έναντι των πινάκων) :

• Η τυχαία πρόσβαση στη λίστα δεν επιτρέπεται. Επομένως, δεν μπορούμε να πραγματοποιήσουμε με αποτελεσματικό τρόπο δυαδική αναζήτηση σε συνδεδεμένες λίστες.
• Οι συνδεδεμένες λίστες έχουν πολύ μεγαλύτερη επιβάρυνση στη χρήση της μνήμης από τους πίνακες, αφού κάθε κόμβος στη λίστα πρέπει, επιπλέον, να αποθηκεύσει έναν πρόσθετο δείκτη που θα δείχνει στον επόμενο κόμβο. Στην περίπτωση των διπλά συνδεδεμένων λιστών χρειαζόμαστε επιπλέον έναν δεύτερο δείκτη που θα δείχνει στον προηγούμενο κόμβο.

Οι συνδεδεμένες λίστες αξιοποιούνται για την υλοποίηση της στοίβας και της ουράς, λόγω της δυνατότητάς αυξομείωσης του μεγέθους τους. Μπορείτε να εξηγήσετε γιατί;

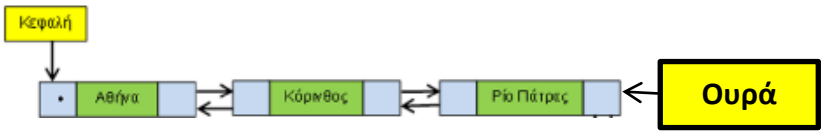
Οι συνδεδεμένες λίστες μπορούν να αξιοποιηθούν για την υλοποίηση της στοίβας και της ουράς.

Η στοίβα μπορεί να υλοποιηθεί με **μία απλά συνδεδεμένη λίστα** αφού μπορεί να γίνει **ΩΘΗΣΗ** στοιχείου **στην κορυφή της στοίβας** με **εισαγωγή** κόμβου (στοιχείου) **στην αρχή της λίστας** και **ΑΠΩΘΗΣΗ** στοιχείου **από την κορυφή της στοίβας** με **διαγραφή** κόμβου (στοιχείου) **από την αρχή της λίστας**.



Εικόνα 1. Υλοποίηση Στοίβας με απλά συνδεδεμένη λίστα

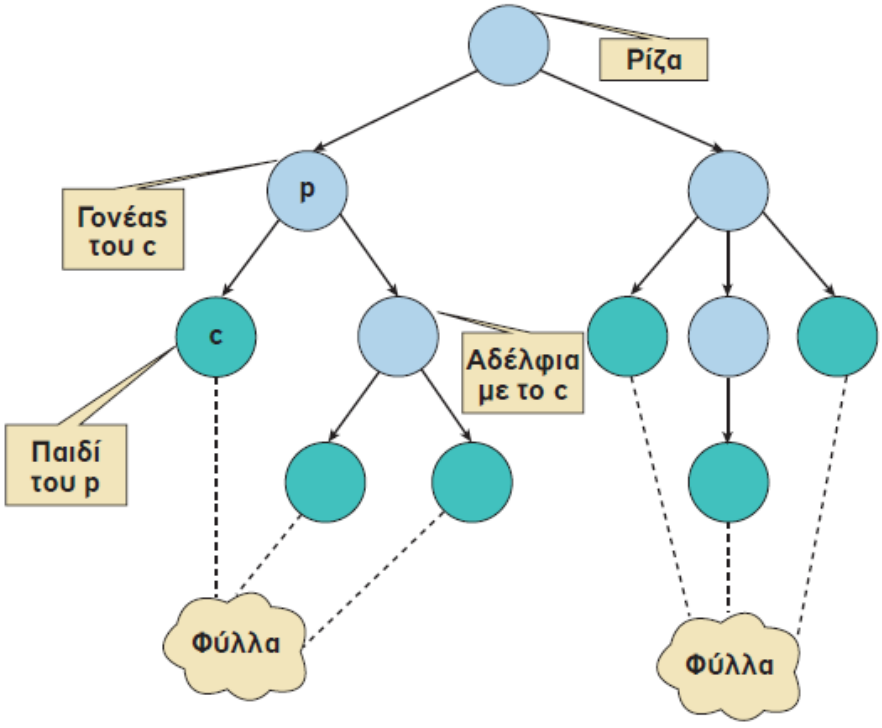
Η ουρά μπορεί να υλοποιηθεί με **μία διπλά συνδεδεμένη λίστα**, αφού μπορούμε να κάνουμε **εισαγωγή** κόμβου **στο τέλος της διπλά συνδεδεμένης λίστας** (**εισαγωγή** στοιχείου **στο πίσω άκρο της ουράς**). Επίσης μπορούμε να κάνουμε **διαγραφή** κόμβου **στην αρχή της διπλά συνδεδεμένης λίστας** (**εξαγωγή** στοιχείου **από το εμπρός άκρο της ουράς**).



<u>Διαφορές στην προσπέλαση μεταξύ δομών δεδομένων . Εντοπίστε τις διαφορές στην προσπέλαση όσον αφορά τη λίστα, τη στοίβα και την ουρά.</u>

Στη στοίβα μπορεί να εισέλθει (ώθηση) στοιχείο μόνο στην κορυφή. Στην ουρά ένα στοιχείο μπορεί να εισέλθει μόνο στο πίσω άκρο. Στη λίστα ένα στοιχείο (κόμβος) μπορεί να εισέλθει σε οποιαδήποτε θέση (αρχή, τέλος ή ενδιάμεσα).
Μια άλλη διαφορά εντοπίζεται κατά την εξαγωγή στοιχείου. Συγκεκριμένα, στη στοίβα ένα στοιχείο μπορεί να εξέλθει μόνο από την κορυφή (απώθηση), ενώ στην ουρά ένα στοιχείο μπορεί να εξέλθει μόνο από την αρχή της. Στη λίστα ένα στοιχείο (κόμβος) μπορεί να διαγραφεί από οποιοδήποτε σημείο της (αρχή, τέλος ή ενδιάμεσα).

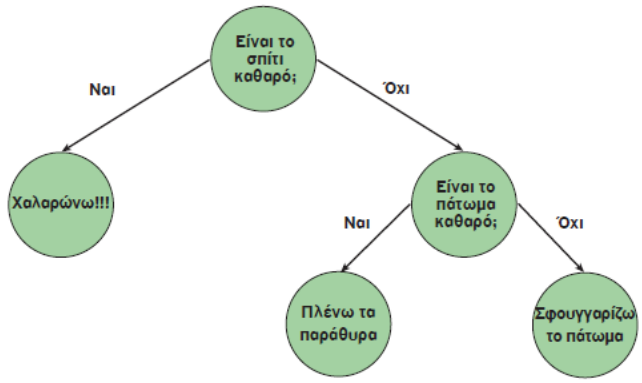
<u>Ένα δένδρο (tree)</u> είναι μία δομή δεδομένων που αποτελείται από ένα σύνολο κόμβων και ένα σύνολο ακμών μεταξύ των κόμβων με βάση τους εξής 3 κανόνες:
• Υπάρχει ένας ξεχωριστός κόμβος που ονομάζεται ρίζα. Αυτός είναι ένας κόμβος χωρίς γονέα.
• Για κάθε κόμβο c, εκτός από τη ρίζα, υπάρχει μόνο μια ακμή που καταλήγει στον κόμβο αυτόν ξεκινώντας από κάποιον άλλον κόμβο p. Ο κόμβος p ονομάζεται γονέας του c και ο κόμβος c παιδί του p.
• Για κάθε κόμβο υπάρχει μία μοναδική διαδρομή, δηλαδή, μια ακολουθία διαδοχικών ακμών, που ξεκινάει από τη ρίζα και τερματίζει σε αυτόν τον κόμβο.
Δένδρο θεωρούμε και το **κενό δένδρο**, δηλαδή το δένδρο που δεν έχει ούτε κόμβους, ούτε ακμές. Το κενό δένδρο είναι το μόνο δένδρο χωρίς ρίζα.



Εικόνα 1.3.11. Σχέσεις μεταξύ των κόμβων ενός δένδρου

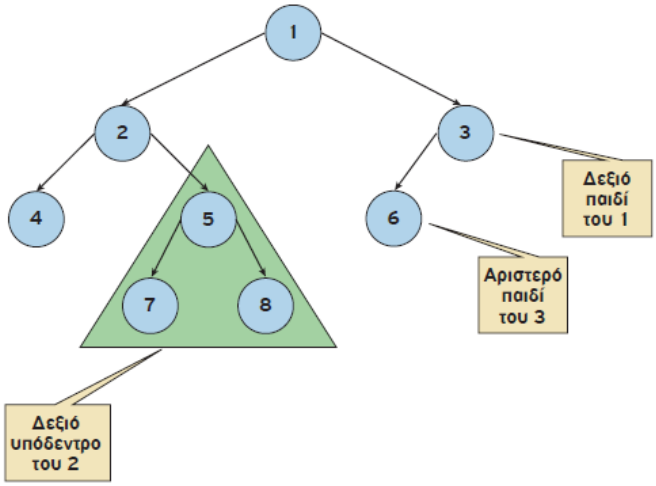
- Όταν δύο κόμβοι συνδέονται μεταξύ τους με μία ακμή, τότε ονομάζουμε «γονέα» τον κόμβο από τον οποίο ξεκινάει η ακμή και «παιδί» τον κόμβο στον οποίο καταλήγει η ακμή.
- Ένας κόμβος μπορεί να έχει κανένα, ένα ή περισσότερα παιδιά.
- Όλοι οι κόμβοι, εκτός από έναν, έχουν ακριβώς έναν γονέα. Ο κόμβος χωρίς γονέα ονομάζεται «ρίζα» (root) και βρίσκεται στην κορυφή του δένδρου.
- Κόμβοι με τον ίδιο γονέα ονομάζονται «αδέλφια». Οι κόμβοι χωρίς παιδιά ονομάζονται «φύλλα».
- Μπορούμε να έχουμε ένα απλό δένδρο, το οποίο να απαρτίζεται από έναν μόνο κόμβο. Αυτός ο κόμβος είναι και ρίζα του απλού αυτού δένδρου, διότι δεν έχει γονέα και φύλλο, και διότι δεν έχει παιδιά.
- **Τα δένδρα είναι μία μη-γραμμική ευέλικτη δομή δεδομένων.**

- Τα **δένδρα απόφασης**, όπως πολύ εύκολα μπορείτε να συμπεράνετε από την Εικόνα 1.3.19, είναι δένδρα στα οποία κάθε κόμβος αντιπροσωπεύει ένα χαρακτηριστικό (ιδιότητα), κάθε ακμή αντιπροσωπεύει μια απόφαση (κανόνα) και κάθε φύλλο αντιπροσωπεύει ένα αποτέλεσμα. Στους αλγορίθμους μηχανικής μάθησης (machine learning) τα δένδρα απόφασης έχουν πρωτεύοντα ρόλο.



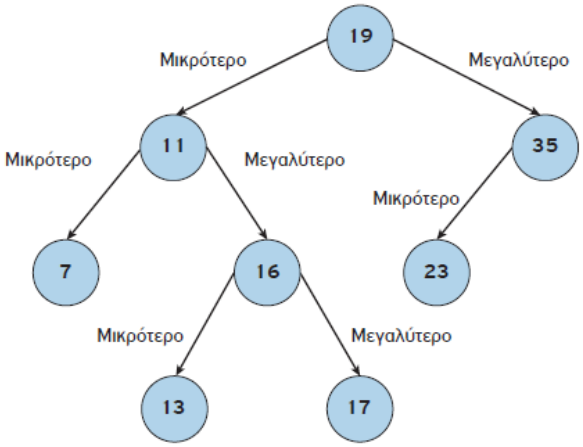
Εικόνα 1.3.19. Δένδρο Απόφασης

- Ένα **δυαδικό δένδρο** (binary tree) είναι ένα διατεταγμένο δένδρο, στο οποίο κάθε κόμβος έχει **το πολύ δύο παιδιά, το αριστερό και το δεξί παιδί**. Μπορούμε, συνεπώς, να μιλάμε για αριστερό και δεξιό υποδένδρο ενός κόμβου.



Εικόνα 1.3.21. Δυαδικό δένδρο

- Ένα **δυαδικό δένδρο αναζήτησης** (binary search tree) είναι ένα δυαδικό δένδρο, όπου για κάθε κόμβο u, όλοι οι κόμβοι του **αριστερού υποδένδρου** έχουν τιμές **μικρότερες** της τιμής του κόμβου u και όλοι οι κόμβοι του **δεξιού υποδένδρου** έχουν τιμές **μεγαλύτερες (ή ίσες)** της τιμής του κόμβου u.



Εικόνα 1.3.22. Δυαδικό δένδρο αναζήτησης

Το πλεονέκτημα των δυαδικών δένδρων αναζήτησης είναι ότι η αναζήτηση για μια συγκεκριμένη τιμή γίνεται ταχύτερα χάρη στον τρόπο αποθήκευσης των τιμών.

Τα δυαδικά δένδρα αναζήτησης συνδυάζουν τα πλεονεκτήματα των λιστών, όσον αφορά τις πράξεις της εισαγωγής και της διαγραφής, αλλά και τα πλεονεκτήματα των ταξινομημένων πινάκων, όσον αφορά την πράξη της αναζήτησης.

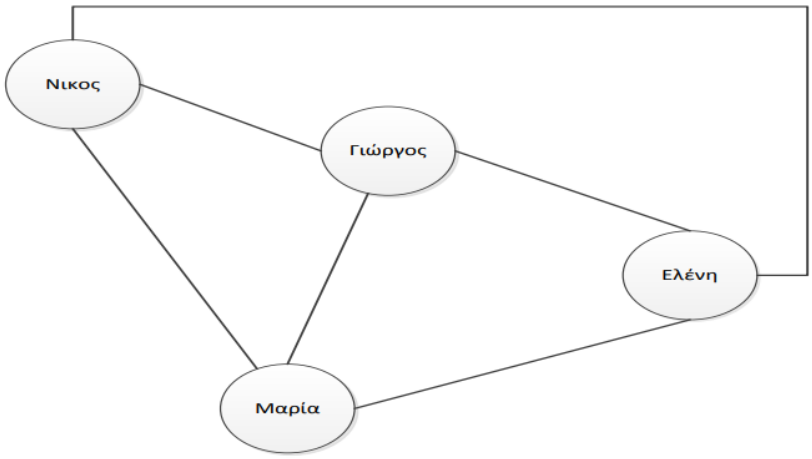
Ένας **γράφος (graph)** είναι μία δομή που αποτελείται από ένα **σύνολο κόμβων** (ή σημείων ή κορυφών) και ένα **σύνολο γραμμών** (ή ακμών ή τόξων) που **ενώνουν μερικούς ή όλους** τους κόμβους.

Τύποι Γράφων

1. Εάν **όλες** οι ακμές σε έναν γράφο **έχουν κατεύθυνση**, ο γράφος ονομάζεται **κατευθυνόμενος γράφος**
2. Εάν **όλες** οι ακμές σε έναν γράφο **δεν έχουν κατεύθυνση**, ο γράφος ονομάζεται **μη κατευθυνόμενος γράφος**

Κοινωνικά Δίκτυα – Facebook

Δεν είναι δυνατόν να είμαι φίλος σου στο δίκτυο χωρίς να είσαι και δικός μου. Η **σχέση** μεταξύ δύο χρηστών (κόμβοι), είναι **αμφίδρομη**, επομένως αναπαρίσταται με **μη κατευθυνόμενες ακμές**.



Εικόνα 4. Κοινωνικά Δίκτυα - Facebook

Κοινωνικά Δίκτυα – Instagram – Twitter

Μπορώ να σε ακολουθήσω, αλλά δεν απαιτείται να με ακολουθήσεις.

Θα μπορούσαμε να αναπαραστήσουμε το Instagram ως **κατευθυνόμενο γράφο**. Κάθε ακμή που δημιουργούμε αντιπροσωπεύει μια **μονόδρομη σχέση.**

Εικόνα 5. Twitter – Η Μαρία ακολουθεί τον Γιώργο

- **Τι θα συμβεί εάν αποφασίσω να σε ακολουθήσω και εγώ; Αλλάζω την κατεύθυνση της ακμής που δημιουργήθηκε όταν με ακολούθησες; Ξαφνικά η ακμή γίνεται αμφίδρομη και επομένως μη κατευθυνόμενη;**

Β) Όχι, διότι θα μπορούσα να σε αγνοήσω σε οποιοδήποτε χρονική στιγμή. Όταν σε ακολουθώ στο Twitter, δημιουργώ μια δεύτερη ακμή, με τον λογαριασμό μου σαν κόμβο προέλευσης και τον δικό σου σαν κόμβο προορισμού (Εικόνα 6).

Εικόνα 6. Twitter - Η Μαρία ακολουθεί τον Γιώργο και ο Γιώργος ακολουθεί τη Μαρία

Οι ακμές του ανωτέρω γράφου αναπαρίστανται ως δύο διαφορετικές κατευθυνόμενες ακμές.

Ο γράφος αποτελεί την πιο γενική ΔΥΝΑΜΙΚΗ δομή δεδομένων, με την έννοια ότι όλες οι προηγούμενες δομές που παρουσιάστηκαν μπορούν να θεωρηθούν **περιπτώσεις γράφων.**