

ΑΛΓΟΡΙΘΜΟΙ

Δρ. Χάρης Κουζινόπουλος

Εθνικό Κέντρο Έρευνας και Τεχνολογικής Ανάπτυξης

Πανεπιστήμιο Μακεδονίας

Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης



CMOR
Computational Methodologies
& Operations Research

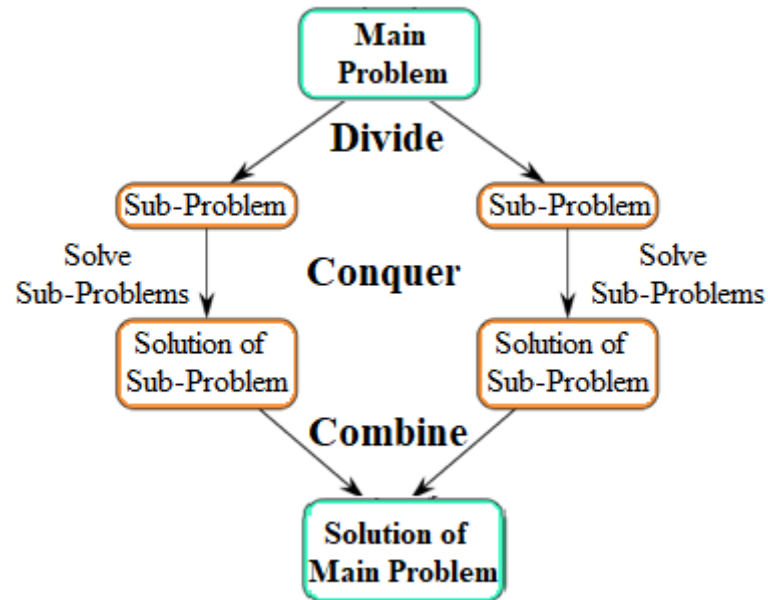
Quiz – [12]

Έστω T ένας μονοδιάστατος πίνακας με θετικούς ακέραιους αριθμούς, π.χ. $T = [1, \dots, 1000]$.

Διαλέγουμε τυχαία έναν αριθμό $X \in T = [1, \dots, 1000]$.

Μπορεί να εντοπιστεί ο αριθμός X κάνοντας το πολύ 10 ερωτήσεις?

Διαίρει-και-Βασίλευε



Δυαδική Αναζήτηση – [1]

Το πρόβλημα. Είναι αναζήτηση αριθμού a σε ταξινομημένο T .

Μέθοδος. Συγκρίνουμε τον αριθμό a με τον $T(k)$ (k επιλέγεται ώστε το $T(i:j)$ να διαιρείται σε δυο περίπου ισοδιάστατα υποδιανύσματα: $k \leftarrow \lfloor (i + j) / 2 \rfloor$).

α) $T(k) = a$. STOP.

β) $a < T(k)$. Ερευνούμε $T(1:k-1)$

γ) $T(k) < a$. Ερευνούμε $T(k+1:n)$

Ψευδοκώδικας `binsearch`: i και j προσδιορίζουν το διάστημα έρευνας ($T(i:j)$)

Δυναδική Αναζήτηση – [2]

| | |
|-------------|--|
| Αλγόριθμος: | Binsearch |
| Είσοδος: | T, n, a |
| Έξοδος: | found, k |
| 1 | $i \leftarrow 1$ |
| 2 | $j \leftarrow n$ |
| 3 | $found \leftarrow 0$ |
| 4 | while $i \leq j$ and $found = 0$ |
| 5 | $k \leftarrow \lfloor (i+j) / 2 \rfloor$ |
| 6 | if $T(k) = a$ |
| 7 | $found \leftarrow 1$ |
| 8 | elseif $a < T(k)$ |
| 9 | $j \leftarrow k - 1$ |
| 10 | else |
| 11 | $i \leftarrow k+1$ |

Δυναδική Αναζήτηση – [3]

a=45

i

k

j

| | | | | | | | | | | | | | | | |
|---------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| | ↓ | | | | | | | ↓ | | | | | | | ↓ |
| Δείκτες | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| T= | 20 | 25 | 30 | 35 | 40 | 45 | 50 | 55 | 60 | 65 | 70 | 75 | 80 | 85 | 90 |

i

k

j

| | | | | | | | |
|---------|----|----|----|----|----|----|----|
| | ↓ | | | ↓ | | | ↓ |
| Δείκτες | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| T= | 20 | 25 | 30 | 35 | 40 | 45 | 50 |

i

k

j

| | | | |
|---------|----|----|----|
| | ↓ | ↓ | ↓ |
| Δείκτες | 5 | 6 | 7 |
| T= | 40 | 45 | 50 |

Δυναδική Αναζήτηση – [4]

| | | | | | | | | | | | | | | | |
|---------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| a=44 | | | | | | | | | | | | | | | |
| | i | | | | | | | k | | | | | | | j |
| | ↓ | | | | | | | ↓ | | | | | | | ↓ |
| Δείκτες | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| T= | 20 | 25 | 30 | 35 | 40 | 45 | 50 | 55 | 60 | 65 | 70 | 75 | 80 | 85 | 90 |

| | | | | | | | |
|---------|----|----|----|----|----|----|----|
| | i | | | k | | | j |
| | ↓ | | | ↓ | | | ↓ |
| Δείκτες | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| T= | 20 | 25 | 30 | 35 | 40 | 45 | 50 |

Αν $i > j$ τότε ο αριθμός που ψάχνουμε δεν υπάρχει.

| | | | |
|---------|----|----|----|
| | i | k | j |
| | ↓ | ↓ | ↓ |
| Δείκτες | 5 | 6 | 7 |
| T= | 40 | 45 | 50 |

$i = j = 5, k=5$

$i = 6 > j = 5$

Δυαδική Αναζήτηση – [5]

Έστω $T=[12 \ 18 \ 24 \ 36 \ 48 \ 60 \ 67 \ 90]$ και $a=67$.

Το $n=8$.

Αρχικά τίθεται $i=1, j=8, found=0$

| Επανάληψη | Συνθήκη όσο ($i \leq j$ και $found=0$) | k | Συνθήκη αν ...αλλιώς αν | found | i | j |
|----------------|---|---|------------------------------|-------|---|---|
| 1 ^η | αληθής | 4 | $T(4)=36 < 67$ ($i=k+1=5$) | 0 | 5 | 8 |
| 2 ^η | αληθής | 6 | $T(6)=60 < 67$ ($i=k+1=7$) | 0 | 7 | 8 |
| 3 ^η | αληθής | 7 | $T(7)=67$ ($found=1$) | 1 | 7 | 8 |
| 4 ^η | ψευδής | | | | | |

- Ο αλγόριθμος επιστρέφει $k=7, found=1$.

Demo: [Binary Search](#)

Μεταβλητές Εισόδου/Εξόδου

ΟΝΟΜΑ ΑΡΧΕΙΟΥ: 07-binsearch.c

ΜΕΤΑΒΛΗΤΕΣ ΕΙΣΟΔΟΥ:

T – Πίνακας (τύπου double) στον οποίο θα πραγματοποιηθεί η αναζήτηση.

N – Το πλήθος των στοιχείων του T (τύπου int)

a – Ο αριθμός προς αναζήτηση (τύπου double).

ΜΕΤΑΒΛΗΤΕΣ ΕΞΟΔΟΥ:

Found – (τύπου int) αν found=0 ο αριθμός δεν βρέθηκε. Αν found=1 ο αριθμός βρέθηκε.


Index – Επιστρέφει τη θέση στην οποία βρέθηκε ο αριθμός a (τύπου int).

Filename: [07-binsearch.c](#)

Πολυπλοκότητα Δυναμικής Αναζήτησης – [1]

Αναζήτηση του στοιχείου 45

| | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 11 | 13 | 21 | 26 | 29 | 36 | 40 | 41 | 45 | 51 | 54 | 56 | 65 | 72 | 77 | 83 |



Πόσα στοιχεία εξετάζουμε στη **χειρότερη περίπτωση**;


Ισοδύναμο ερώτημα:

Πόσες φορές μπορούμε να διαιρούμε στη μέση έναν πίνακα με n στοιχεία έτσι ώστε ο τελικός πίνακας να έχει ένα μόνο κελί;

Πολυπλοκότητα Δυαδικής Αναζήτησης – [1]

Αναζήτηση του στοιχείου 45

| | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 11 | 13 | 21 | 26 | 29 | 36 | 40 | 41 | 45 | 51 | 54 | 56 | 65 | 72 | 77 | 83 |



$$d = \log_2(n)$$

Ισοδύναμο με $2^x = n$. Παράδειγμα: αν $n = 8$, $\log_2(8) = 3$ καθώς $2^3 = 8$

Πολυπλοκότητα Δυναμικής Αναζήτησης – [2]

Πολυπλοκότητα Χειρότερης Περίπτωσης: $\log_2 n$

| n | $\log_2 n$ |
|---------------|------------|
| 1.000 | 9,96 < 10 |
| 10.000 | 13,29 < 14 |
| 100.000 | 16,61 < 17 |
| 1.000.000 | 19,93 < 20 |
| 10.000.000 | 23,25 < 24 |
| 100.000.000 | 26,57 < 27 |
| 1.000.000.000 | 29,89 < 30 |

Σειριακή vs Δυαδική Αναζήτηση

- Η σειριακή αναζήτηση είναι καλύτερη αν το n είναι σχετικά μικρό.
- Η σειριακή αναζήτηση είναι προτιμότερη αν η αναζήτηση πραγματοποιείται σπάνια.
- Η δυαδική αναζήτηση είναι προτιμότερη αν η αναζήτηση πραγματοποιείται συχνά.
- Η δυαδική αναζήτηση δεν μπορεί να εφαρμοστεί σε
 - (a). Μη ταξινομημένους πίνακες
 - (b). Συνδεδεμένες λίστες
 - (c). Πίνακες κατακερματισμού
 - (d). Πολύ-διάστατους πίνακες

Παρόμοια Προβλήματα

- Ποιοι είναι οι πιο κοντινοί γείτονες του a , αν ο αριθμός a δεν υπάρχει στον πίνακα T ?
- Που θα ήταν το a (σε ποια θέση) αν ήταν στον πίνακα T ?

Παραλλαγές Αλγορίθμων Αναζήτησης

- Ακριβές ταίριασμα (Exact match)
- Μερικό ταίριασμα (Partial match)
- Ερώτηση περιοχής (Range query)

Πρόβλημα για το σπίτι – [11]

[A]. Ο αλγόριθμος δυαδικής αναζήτησης στηρίζεται στη διχοτόμηση του πίνακα ή του υπό εξέταση τμήματός του. Με βάση τον αλγόριθμο δυαδικής αναζήτησης σχεδιάστε ένα αλγόριθμο «τριαδικής» αναζήτησης, δηλαδή ένα αλγόριθμο που να τριχοτομεί τον πίνακα ή το υπό εξέταση τμήμα του. Στην περίπτωση αυτή επιλέγουμε ένα από τα τρία τμήματα και προχωρούμε τριχοτομώντας το έως ότου είτε βρούμε το ζητούμενο στοιχείο είτε αποτύχουμε.

[B]. Πόσα στοιχεία εξετάζουμε στην χειρότερη περίπτωση;

Quiz – [13]

Ένας ταχυδακτυλουργός ζητά από ένα άτομο να επιλέξει ένα χαρτί τράπουλας από μια στοίβα 27 χαρτιών και στη συνέχεια ζητά από το άτομο να το επιστρέψει στη στοίβα χωρίς να δείξει το χαρτί σε κανέναν.

Μετά ο ταχυδακτυλουργός ανακατεύει την τράπουλα και τοποθετεί τα χαρτιά σε τρεις στοίβες και ζητά από το άτομο να του πει σε ποια στοίβα από τις τρεις υπάρχει το χαρτί που επέλεξε. Η ίδια διαδικασία επαναλαμβάνεται άλλες δυο φορές. Στο τέλος ο ταχυδακτυλουργός λέει ποιο χαρτί επέλεξε το άτομο.

Εξηγείστε το κόλπο.

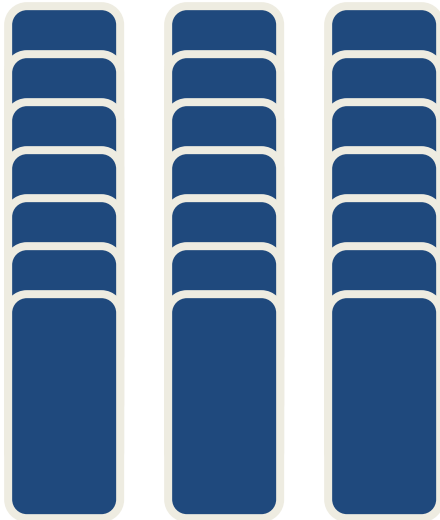
Quiz – [13]

| File 1 | File 2 | File 3 | File 1 | File 2 | File 3 | File 1 | File 2 | File 3 |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| a1 | b1 | c1 | b1 | b2 | b3 | b1 | b4 | b7 |
| a2 | b2 | c2 | b4 | b5 | b6 | a1 | a4 | a7 |
| a3 | b3 | c3 | b7 | b8 | b9 | c1 | c4 | c7 |
| a4 | b4 | c4 | a1 | a2 | a3 | b3 | b6 | b9 |
| a5 | b5 | c5 | a4 | a5 | a6 | a3 | a6 | a9 |
| a6 | b6 | c6 | a7 | a8 | a9 | c3 | c6 | c9 |
| a7 | b7 | c7 | c1 | c2 | c3 | b2 | b5 | b8 |
| a8 | b8 | c8 | c4 | c5 | c6 | a2 | a5 | a8 |
| a9 | b9 | c9 | c7 | c8 | c9 | c2 | c5 | c8 |
| (a) | | | (b) | | | (c) | | |

Quiz – [13]



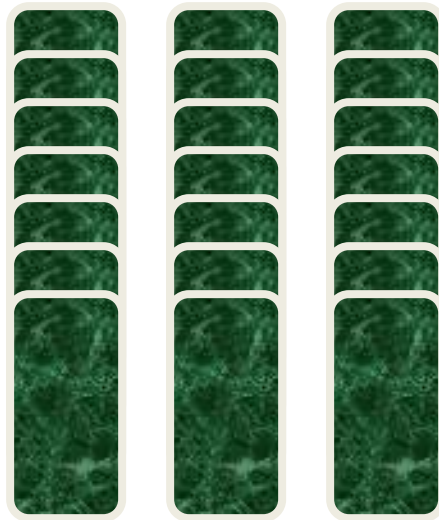
OK



Επέλεξε ένα
χαρτί



Quiz – [13]



Ποια στήλη;

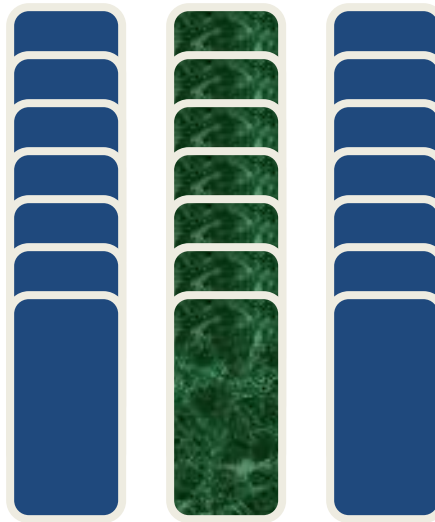


Η αριστερή

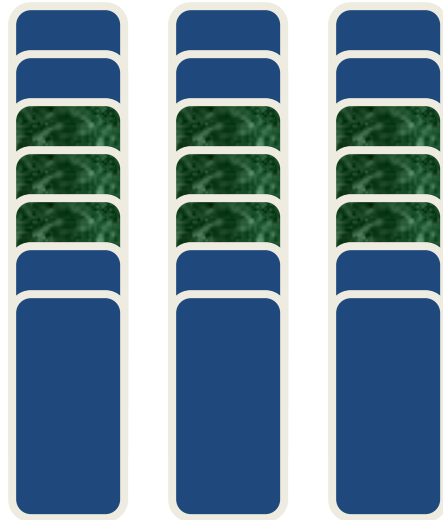


Quiz – [13]

Η επιλεγμένη στήλη
τοποθετείται στη μέση.



Quiz – [13]



Ποια στήλη;

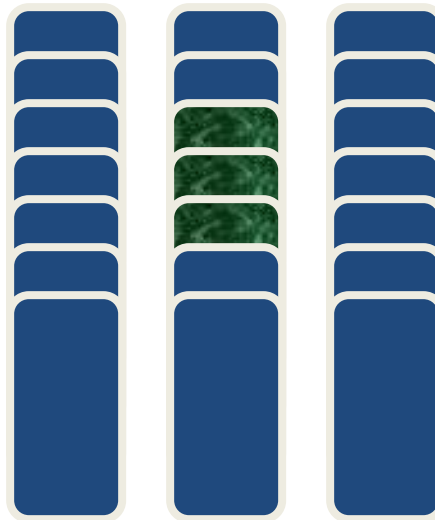


Η δεξιά

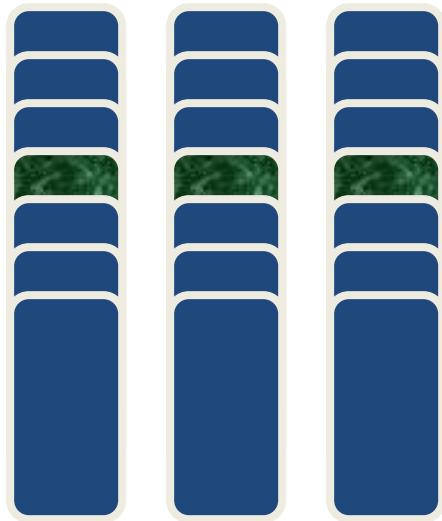


Quiz – [13]

Η επιλεγμένη στήλη
τοποθετείται στη μέση.



Quiz – [13]



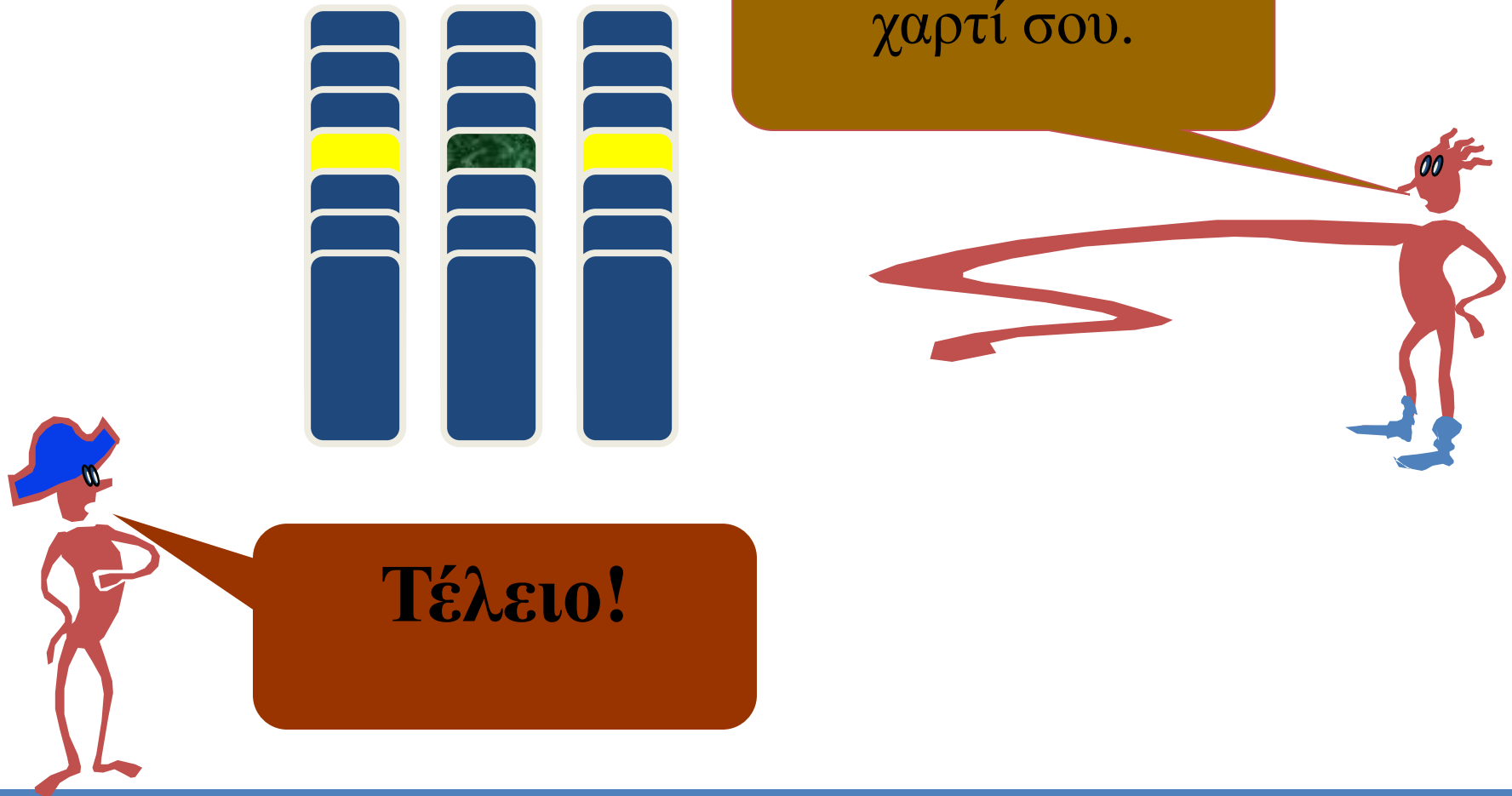
Ποια στήλη;



Η αριστερή



Quiz – [13]



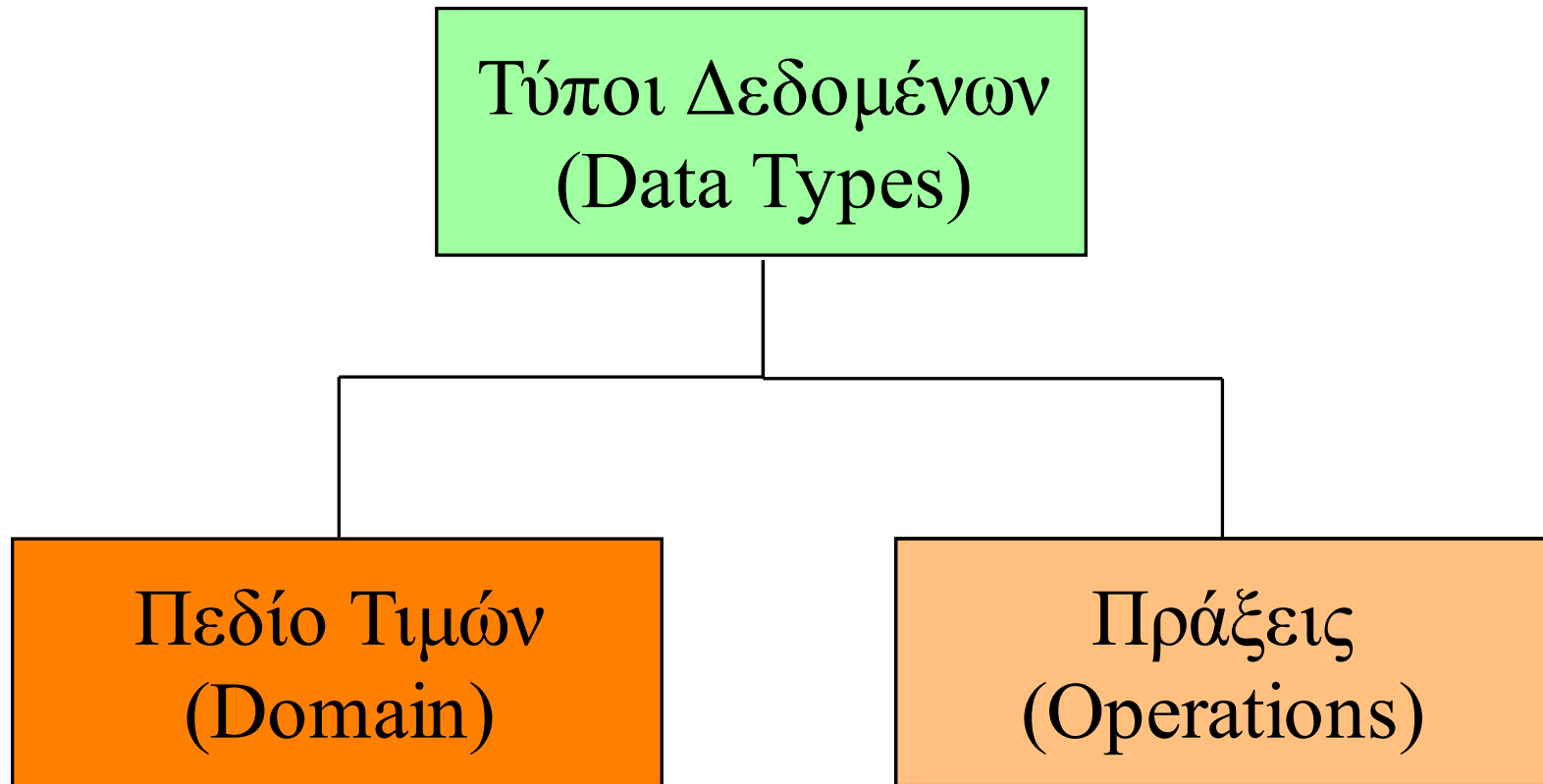
Δομές Δεδομένων – [1]

ΣΤΟΧΟΣ: Να μπορείτε να χρησιμοποιείτε τις δομές δεδομένων στην ανάπτυξη «αποδοτικών» προγραμμάτων.

Τα μεγαλύτερα πλεονεκτήματα της ορθής χρήσης δομών δεδομένων είναι:

1. Αποτελεσματική διαχείριση της μνήμης του Η/Υ
2. Βελτίωση της υπολογιστικής πολυπλοκότητας ενός αλγορίθμου
3. Βελτίωση της θεωρητικής πολυπλοκότητας ενός αλγορίθμου
4. Συγγραφή ευανάγνωστου πηγαίου κώδικα
5. Ελάττωση του παραγόμενου πηγαίου κώδικα

Δομές Δεδομένων – [2]



Δομές Δεδομένων – [3]

Οι ΔΔ είναι τρισυπόστατα αντικείμενα αποτελούμενα:

- 1) Από τα ίδια τα δεδομένα τα οποία ονομάζονται **κλειδιά (keys)**
- 2) Από το σύνολο λειτουργιών – αλγορίθμων που ονομάζονται **πράξεις (operations)**
- 3) Από μεταβλητές, διανύσματα, μήτρες στα οποία αποθηκεύονται τα δεδομένα καθώς και χρήσιμες πληροφορίες για την εκτέλεση των πράξεων.

Δομές Δεδομένων – [4]

Κυριότερες Πράξεις στις Δομές Δεδομένων

- (α) *διαπέραση (traversal)*: προσπέλαση και επεξεργασία κάθε στοιχείου ή κόμβου.
- (β) *αναζήτηση (search)*: εύρεση στοιχείου ή κόμβου με κάποια δεδομένη τιμή.
- (γ) *εισαγωγή (insertion)*: πρόσθεση ενός νέου στοιχείου ή κόμβου.
- (δ) *διαγραφή (deletion)*: αφαίρεση ενός υπάρχοντος στοιχείου ή κόμβου.
- (ε) *διάταξη (sorting)*: τακτοποίηση των στοιχείων ή κόμβων σε κάποια σειρά.

Πίνακες

πίνακας μίας διάστασης με ακεραίους

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 2 | 5 | 2 | 3 | 6 | 9 | 8 |
|---|---|---|---|---|---|---|---|

πίνακας μίας διάστασης με εγγραφές

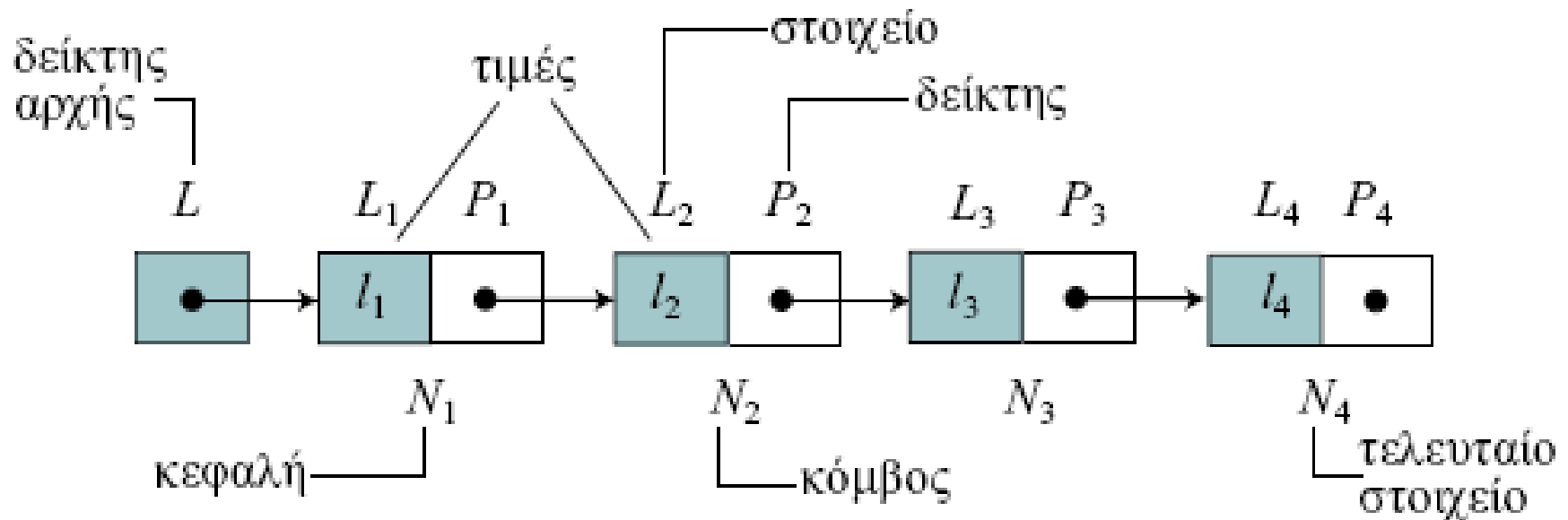
| | | | | | |
|-----------------|-----------------|-----------------|-----------------|-----------------|------------------|
| 1 Mary 25 | 2 John 21 | 3 Kate 30 | 4 Bill 44 | 5 Jack 31 | 6 Jason 50 |
|-----------------|-----------------|-----------------|-----------------|-----------------|------------------|

πίνακας δύο διαστάσεων (3 x 5) με ακεραίους

| | | | | |
|---|---|---|---|---|
| 5 | 4 | 6 | 7 | 5 |
| 4 | 4 | 2 | 2 | 1 |
| 5 | 6 | 1 | 1 | 2 |

Δομή Λίστας

Δομή Λίστας



Στοιίβα (stack) – [1]

Δομή δεδομένων που προσομοιάζει τη λειτουργία στοιβών πιάτων σε εστιατόρια. Αντί για πιάτα χρησιμοποιούνται αριθμοί.

Προσομοιάζει τη λειτουργία T.M.Π.Ε. (Τελευταίος Μέσα Πρώτος Έξω), ή LIFO (Last In First Out).

Περιγραφή: σύνολο αριθμών αποθηκευμένων στις πρώτες (ή τελευταίες) θέσεις διανύσματος όπου αριθμοί εισάγονται (push) και εξάγονται (pop) από το ένα άκρο.

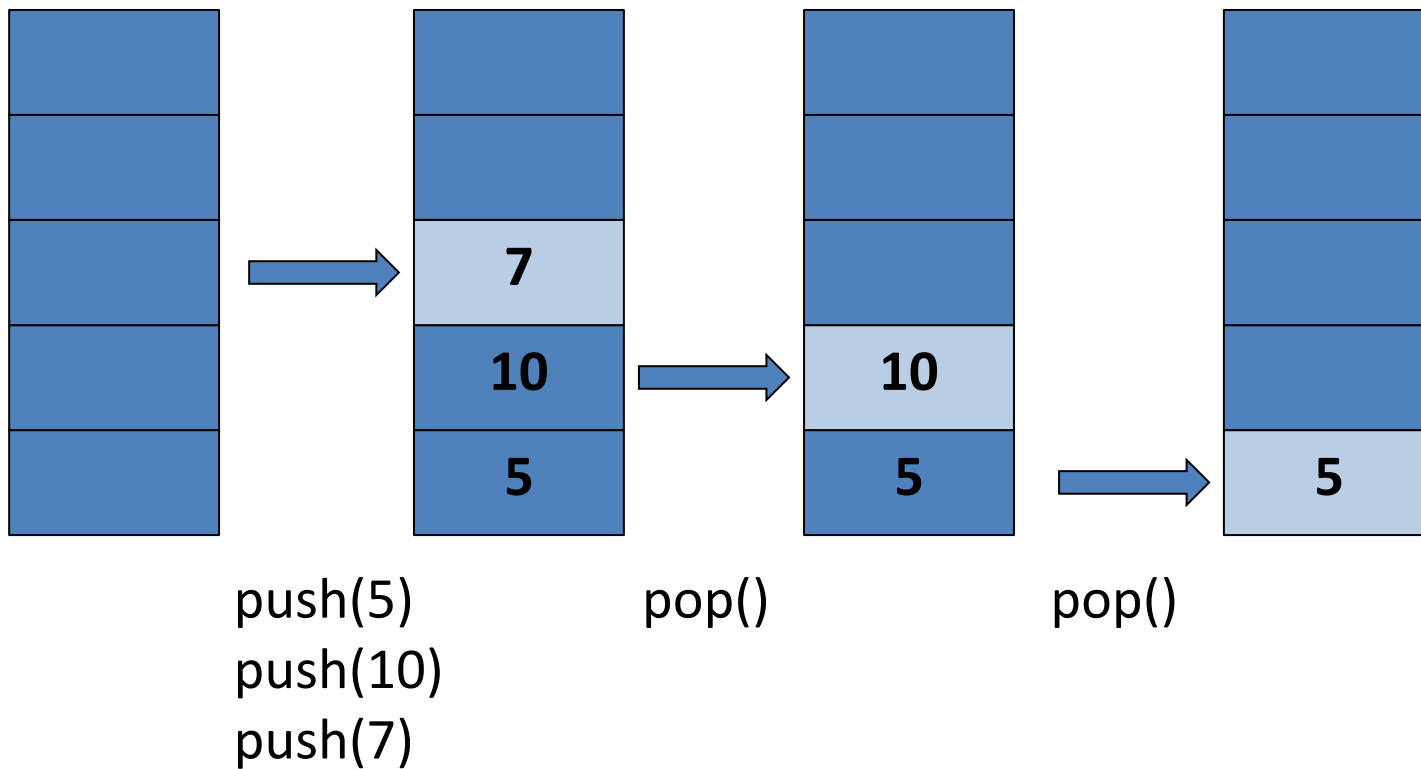
Πολλές χρήσεις: μνήμη Η/Υ, αναδρομικοί μεταγλωττιστές, συναρτήσεις στις γλώσσες προγραμματισμού.

Υποστηρίζονται δύο βασικές λειτουργίες:

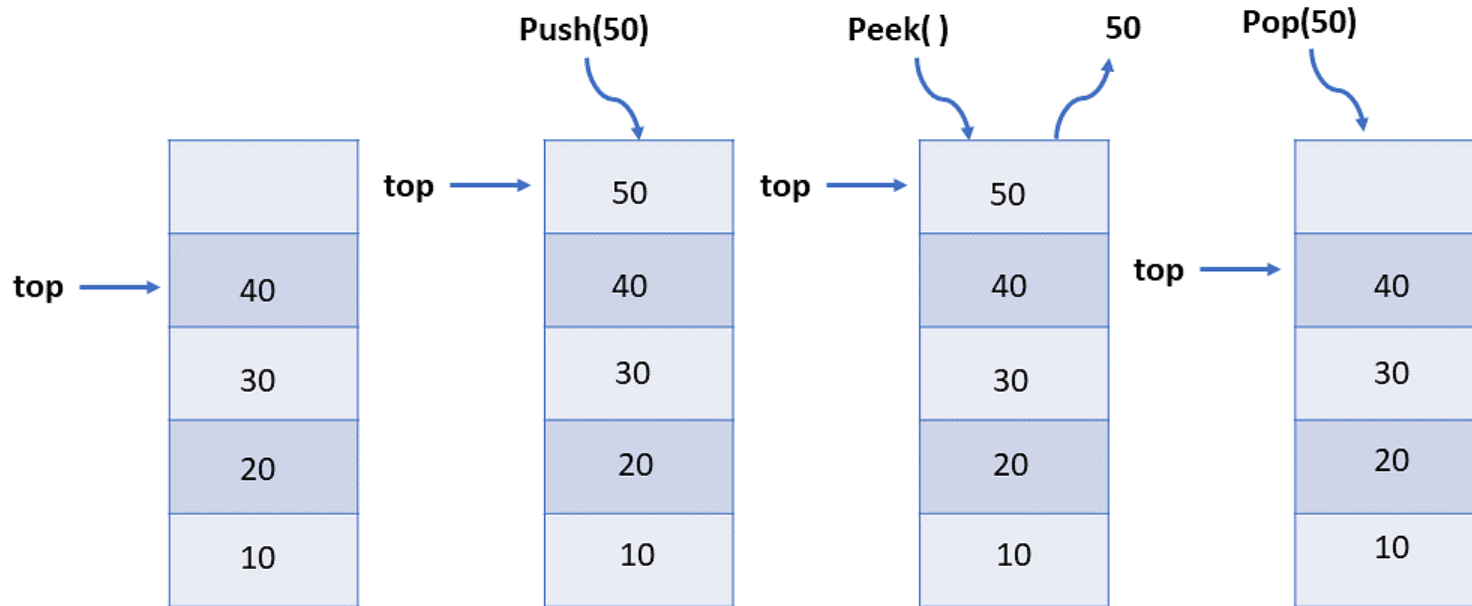
- ❑ εισαγωγή (**push**)
- ❑ εξαγωγή (**pop**).

Στοιίβες – [2]

Παράδειγμα



Στοίβες – [3]



Demo: Stack

Στοιίβες – [4]

Ψευδοκώδικας πράξης push : Εισάγεται ο a στην στοίβα S

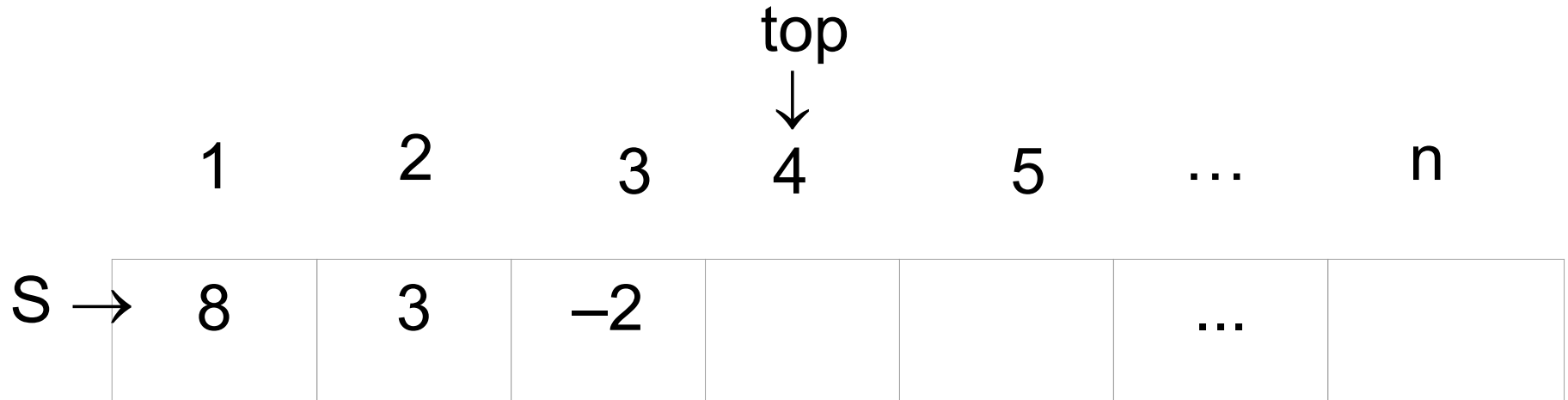
| | |
|-------------------------|--------------------------|
| Αλγόριθμος: push | |
| Είσοδος: S, n, top, a | |
| Έξοδος: $S, top, done$ | |
| 1 | $done \leftarrow 0$ |
| 2 | if $top \leq n$ |
| 3 | $S(top) \leftarrow a$ |
| 4 | $top \leftarrow top + 1$ |
| 5 | $done \leftarrow 1$ |

Ο αριθμός δεν έχει εισαχθεί ακόμη

Ο αριθμός έχει εισαχθεί

Στοίβες – [5]

Παράδειγμα `push(5)`.



Μετά την εισαγωγή του 5:



Στοίβες – [6]

Ψευδοκώδικας πράξης pop : Εξάγεται το πρώτο στοιχείο της στοίβας S

Αλγόριθμος: pop
Είσοδος: S, n, top
Έξοδος: S, top, x, done

| | |
|---|--------------------------|
| 1 | done \leftarrow 0 |
| 2 | if top \geq 2 |
| 3 | top \leftarrow top - 1 |
| 4 | x \leftarrow S(top) |
| 5 | done \leftarrow 1 |

Στοίβες – [7]

Παράδειγμα pop.

Η στοίβα πριν την εξαγωγή του 5:

1 2 3 4 5 ... n

top
↓



Η στοίβα μετά την εξαγωγή του 5:

1 2 3 4 5 ... n

top
↓



Μεταβλητές Εισόδου/Εξόδου

ΟΝΟΜΑ ΑΡΧΕΙΟΥ: 08-push-pop.c

ΜΕΤΑΒΛΗΤΕΣ ΕΙΣΟΔΟΥ (push) :

S – Πίνακας (τύπου double) στον οποίο θα αποθηκευτούν τα στοιχεία της στοίβας.

N – Το πλήθος των στοιχείων της S (τύπου int)

x – Ο αριθμός προς εισαγωγή (τύπου double).

ΜΕΤΑΒΛΗΤΕΣ ΕΞΟΔΟΥ (push) :

S – Πίνακας (τύπου double) στον οποίο θα αποθηκευτούν τα στοιχεία της νέας στοίβας.

done – Επιστρέφει 1 αν έγινε με επιτυχία η εισαγωγή του x και 0 διαφορετικά (τύπου int).

Filename: [08-push-pop.c](#)

Μεταβλητές Εισόδου/Εξόδου

ΟΝΟΜΑ ΑΡΧΕΙΟΥ: 08-push-pop.c

ΜΕΤΑΒΛΗΤΕΣ ΕΙΣΟΔΟΥ (pop):

S – Πίνακας (τύπου double) στον οποίο θα αποθηκευτούν τα στοιχεία της στοίβας.

N – Το πλήθος των στοιχείων της S (τύπου int)

ΜΕΤΑΒΛΗΤΕΣ ΕΞΟΔΟΥ (pop):

S – Πίνακας (τύπου double) στον οποίο θα αποθηκευτούν τα στοιχεία της νέας στοίβας.

done – Επιστρέφει 1 αν έγινε με επιτυχία η εξαγωγή του x και 0 διαφορετικά (τύπου int).

x – Ο αριθμός προς εξαγωγή (τύπου double).

Filename: [08-push-pop.c](#)

Ουρές – [1]

Υλοποιούν τις κοινές ουρές που συναντούμε στην καθημερινή ζωή μας.

Προτεραιότητα Π.Μ.Π.Ε. (Πρώτος Μέσα Πρώτος Έξω) ή FIFO (First In First Out).

Χρησιμοποιείται ένα διάνυσμα Q , στο οποίο οι αριθμοί είναι αποθηκευμένοι στις θέσεις

$front + 1, front + 2, \dots, rear - 1, rear$

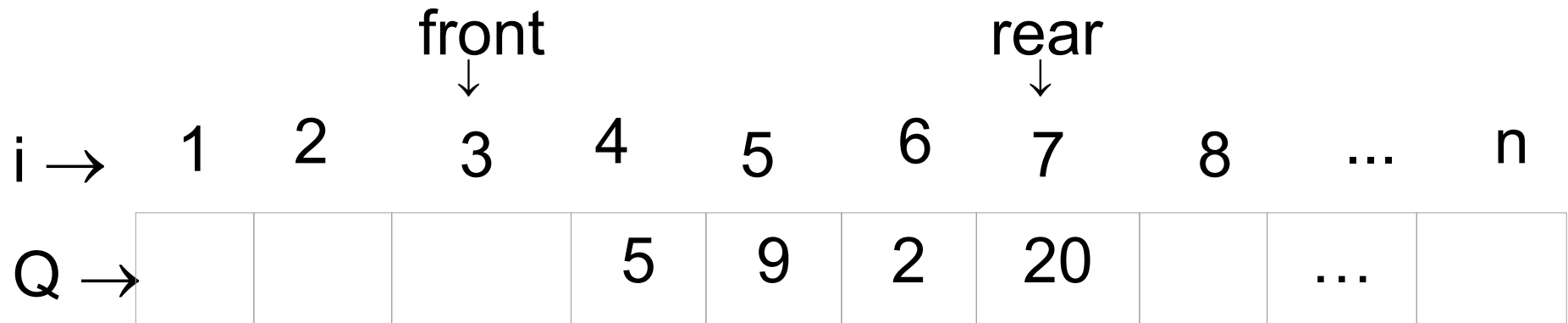
front = θέση ακριβώς πριν τον πρώτο αριθμό

rear = θέση τελευταίου αριθμού της ουράς.

1 2 $\overset{\text{front}}{\downarrow}$ 3 4 5 6 $\overset{\text{rear}}{\downarrow}$ 7 8 ... n

| | | | | | | | | | |
|--|--|--|---|---|---|----|--|-----|--|
| | | | 5 | 9 | 2 | 20 | | ... | |
|--|--|--|---|---|---|----|--|-----|--|

Ουρές – [2]



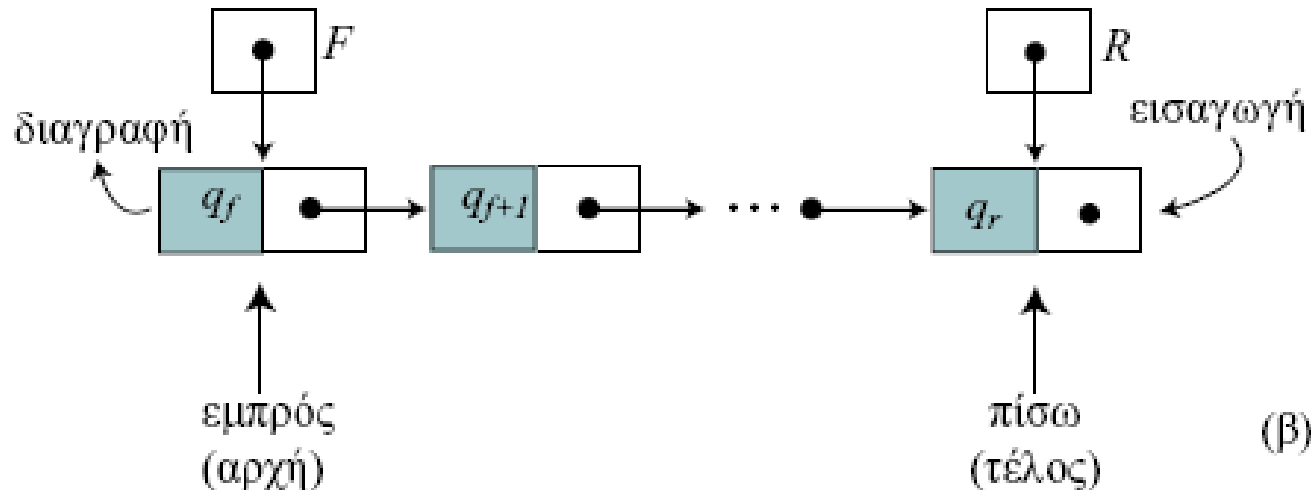
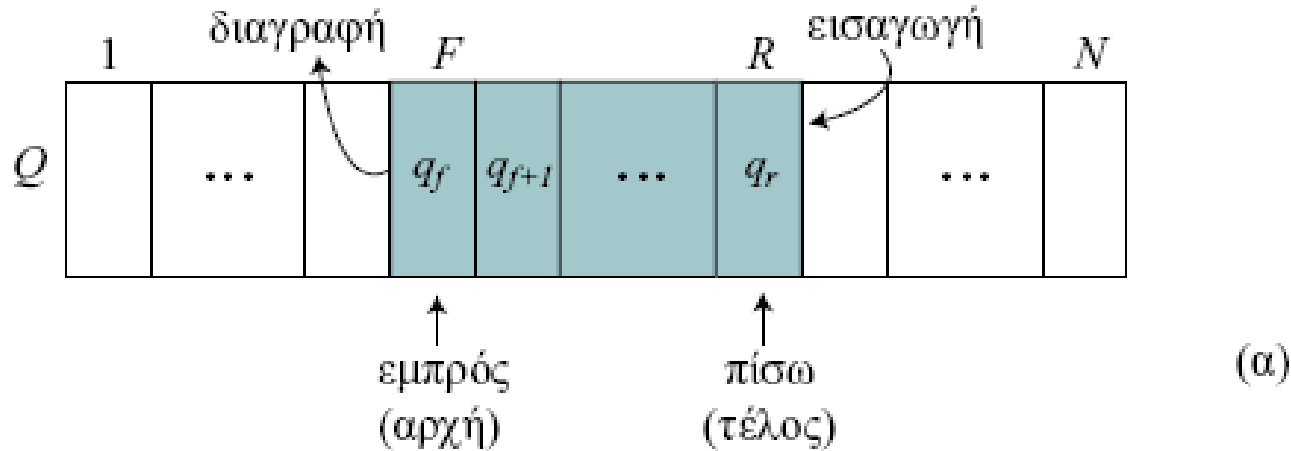
2 πράξεις επιτρέπονται:

enqueue: εισάγει αριθμό στο πίσω μέρος της ουράς

dequeue : εξάγει τον αριθμό στο μπρός μέρος της ουράς

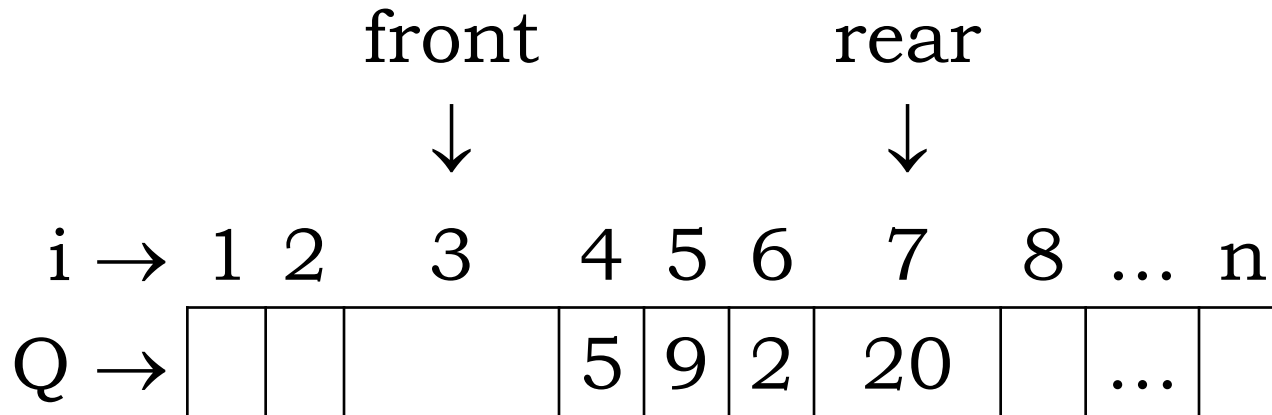
Οι πράξεις γίνονται με αλλαγές των τιμών των front, rear

Ουρές – [3]



Demo: Queue

Ουρές – [4]



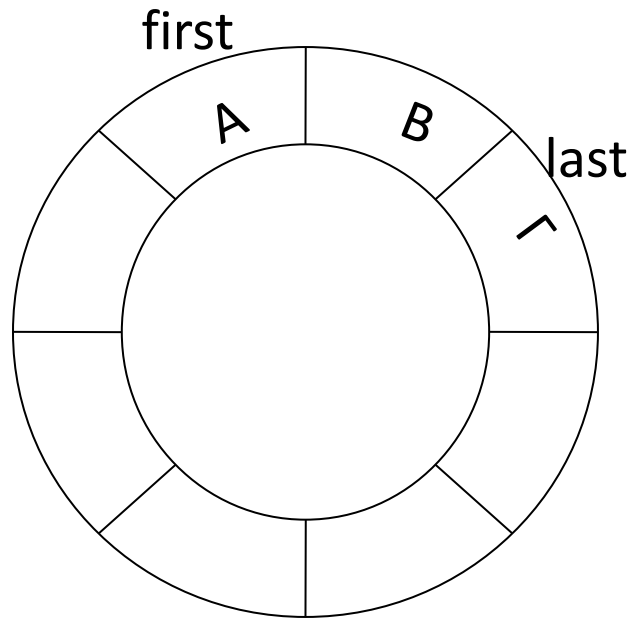
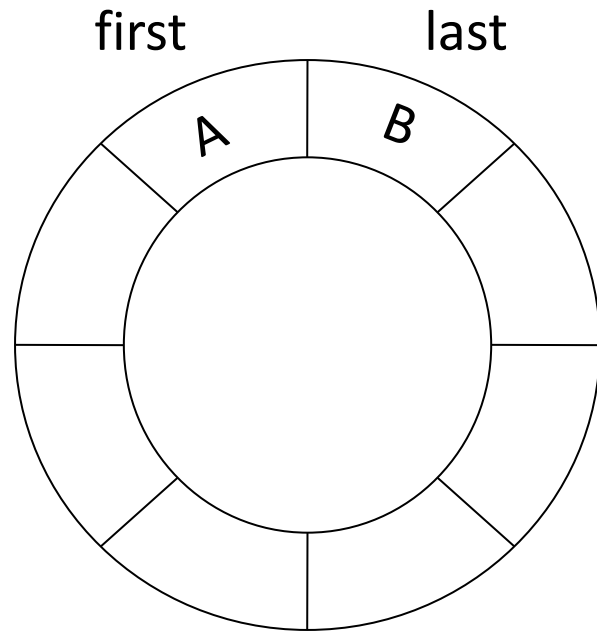
Εξαγωγή αριθμού: $Q(\text{front} + 1) = Q(4) = 5$

Εισαγωγή αριθμού: αν πρόκειται να εισαχθεί ο αριθμός 10, θα τοποθετηθεί στη θέση 8 θέτοντας $Q(\text{rear} + 1) = Q(8) = 10$

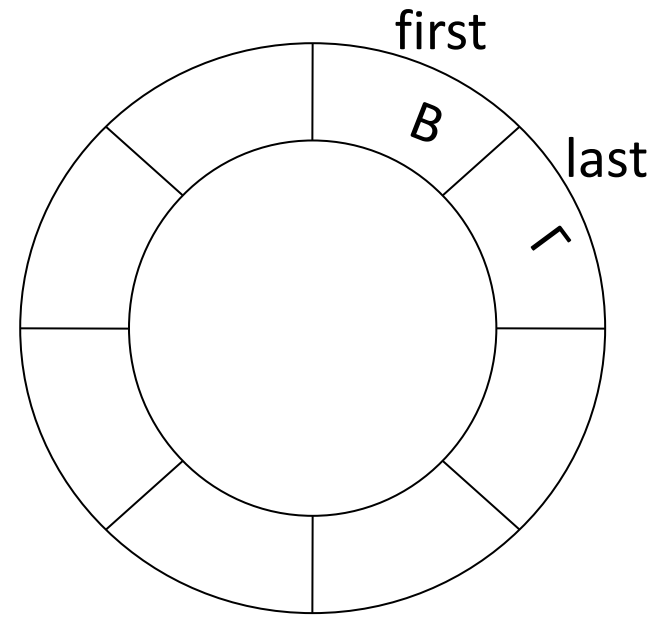
Μετά από αρκετές εισαγωγές και εξαγωγές μπορούν να υπάρχουν αριθμοί στις τελευταίες θέσεις του διανύσματος Q, μέχρι και την τελευταία θέση, ενώ ταυτόχρονα να υπάρχουν κενές θέσεις στην αρχή.

Ουρές – [5]

Με χρήση «κυκλικού» πίνακα



insert(Γ)



delete(A)

Ερώτηση: πώς ελέγχουμε εάν η ουρά είναι γεμάτη;

Ουρές – [6]

Κυκλική ουρά: Χρήση mod, $\text{rear} = \text{mod}(\text{rear} + 1, n)$
 $\text{front} = \text{mod}(\text{front} + 1, n).$

Προσοχή !

- 1) Θέση $n \leftrightarrow \text{mod}(\text{rear} + 1, n) = 0$
- 2) Πότε η ουρά είναι κενή και πότε γεμάτη? $\text{front} = \text{rear}$ δεν είναι αρκετό. $\text{nofelem} = 0$ (κενή), $= n$ (γεμάτη).
- 3) Αρχικές συνθήκες : $\text{front} = \text{rear} = 1, \text{nofelem} = 0$

Ουρές – [7]

εισαγόμενος αριθμός



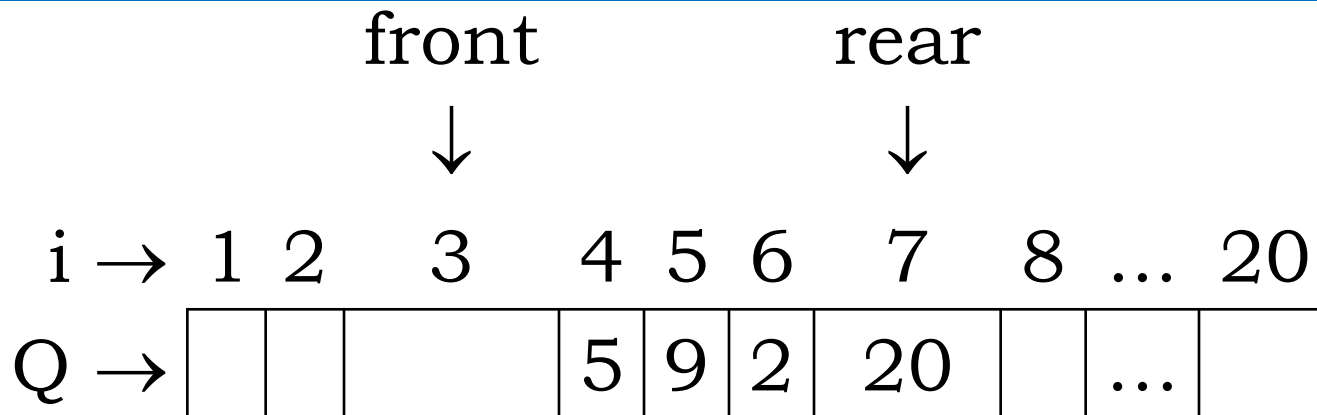
Αλγόριθμος: enqueue

Είσοδος: Q, n, k, rear, nofelem

Έξοδος: Q, rear, nofelem, done

| | |
|---|----------------------------------|
| 1 | done \leftarrow 0 |
| 2 | if nofelem < n |
| 3 | rear \leftarrow mod(rear+1, n) |
| 4 | if rear = 0 |
| 5 | rear \leftarrow n |
| 6 | Q(rear) \leftarrow k |
| 7 | nofelem \leftarrow nofelem + 1 |
| 8 | done \leftarrow 1 |

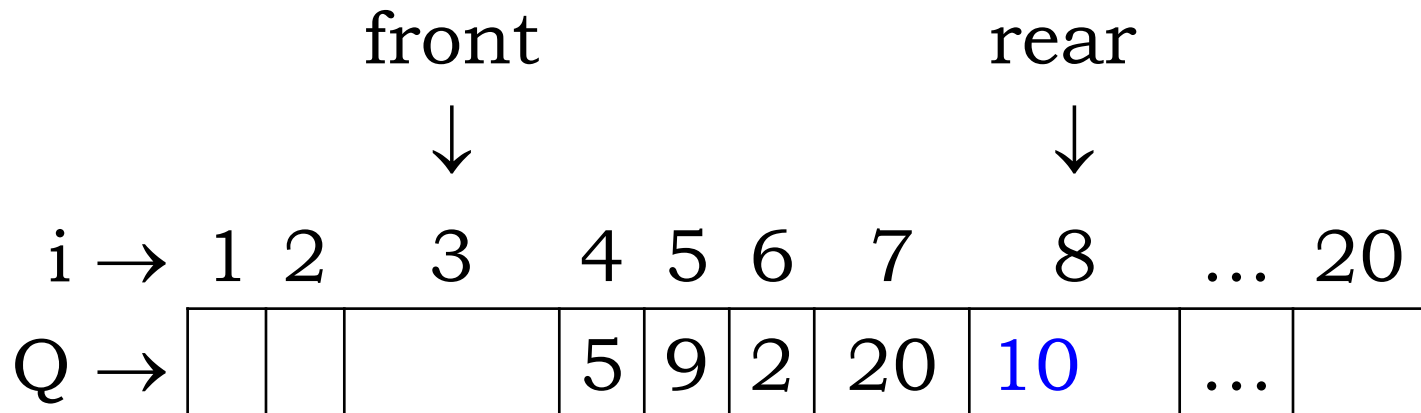
Ουρές – [8]



Εισαγωγή αριθμού: k=10

$\text{rear} = \text{mod}(\text{rear}+1, 20) = \text{mod}(7+1, 20) = 8$

Άρα $Q(\text{rear}) = Q(8) = 10$



Μεταβλητές Εισόδου/Εξόδου

ΟΝΟΜΑ ΑΡΧΕΙΟΥ: 09-enqueue-dequeue.c

ΜΕΤΑΒΛΗΤΕΣ ΕΙΣΟΔΟΥ (enqueue):

Q - Πίνακας (τύπου double) στον οποίο θα αποθηκευτούν τα στοιχεία (N) της ουράς.

nofelem - Το πλήθος των στοιχείων της Q (τύπου int)

k - Ο αριθμός προς εισαγωγή (τύπου double).

rear - Δείκτης στο τέλος της ουράς (τύπου int).

ΜΕΤΑΒΛΗΤΕΣ ΕΞΟΔΟΥ (enqueue):

Q - Πίνακας (τύπου double) στον οποίο θα αποθηκευτούν τα στοιχεία της νέας ουράς.

done - Επιστρέφει 1 αν έγινε με επιτυχία η εισαγωγή του k και 0 διαφορετικά (τύπου int).

Filename: [09-enqueue-dequeue.c](#)

Ουρές – [9]

εξαγόμενος αριθμός

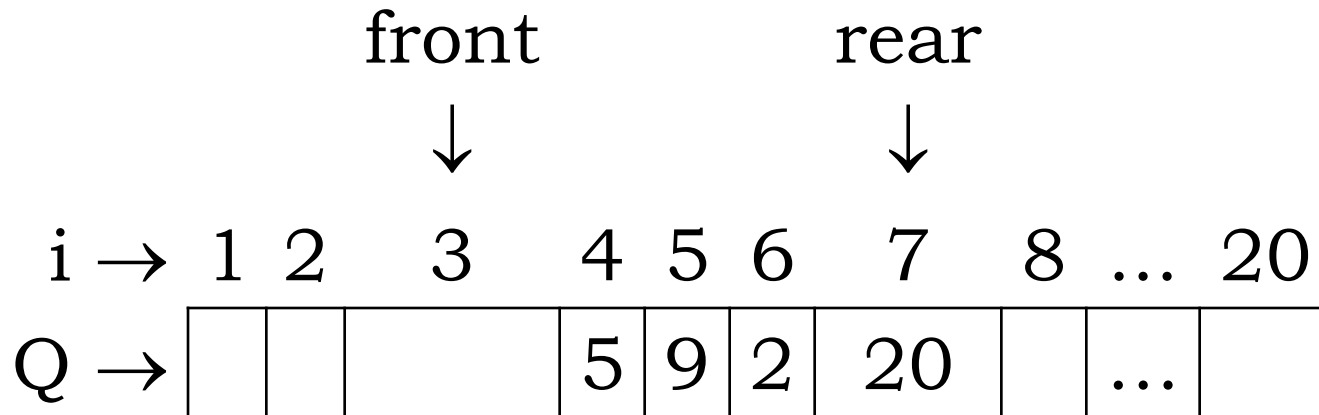
Αλγόριθμος: dequeue

Είσοδος: Q, n, front, nofelem

Έξοδος: Q, k, front, nofelem, done

| | |
|---|------------------------------------|
| 1 | done \leftarrow 0 |
| 2 | if nofelem > 0 |
| 3 | front \leftarrow mod(front+1, n) |
| 4 | if front = 0 |
| 5 | front \leftarrow n |
| 6 | k \leftarrow Q(front) |
| 7 | nofelem \leftarrow nofelem – 1 |
| 8 | done \leftarrow 1 |

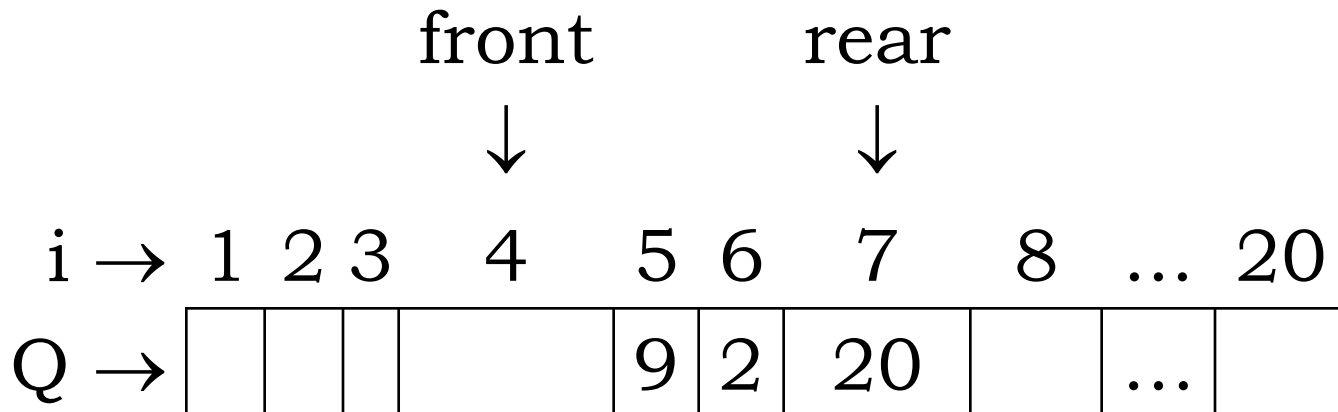
Ουρές – [10]



Εξαγωγή αριθμού: $k=5$

$\text{front} = \text{mod}(\text{front}+1, 20) = \text{mod}(3+1, 20) = 4$

Άρα $k=Q(\text{front})=Q(4)=5$



Μεταβλητές Εισόδου/Εξόδου

ΟΝΟΜΑ ΑΡΧΕΙΟΥ: 09-enqueue-dequeue.c

ΜΕΤΑΒΛΗΤΕΣ ΕΙΣΟΔΟΥ (enqueue):

Q - Πίνακας (τύπου double) στον οποίο θα αποθηκευτούν τα στοιχεία (N) της ουράς.

nofelem - Το πλήθος των στοιχείων της Q (τύπου int)

front - Δείκτης στην αρχή της ουράς (τύπου int).

ΜΕΤΑΒΛΗΤΕΣ ΕΞΟΔΟΥ (dequeue):

Q - Πίνακας (τύπου double) στον οποίο θα αποθηκευτούν τα στοιχεία της νέας ουράς.

done - Επιστρέφει 1 αν έγινε με επιτυχία η εισαγωγή του k και 0 διαφορετικά (τύπου int).

k - Ο αριθμός προς εξαγωγή (τύπου double).

Filename: [09-enqueue-dequeue.c](#)