Γ Λυκείου

Ανάπτυξη εφαρμογών σε προγραμματιστικό περιβάλλον

ΜΟΝΟΔΙΑΣΤΑΤΟΙ- ΠΙΝΑΚΕΣ

Περιεχόμενα

Κατηφόρης Παναγιώτης
2019-2020

Πίνακες

Τι είναι ο πίνακας.

Είναι μια **στατική** δομή (περιέχει δηλαδή, ένα σταθερό σύνολο από κόμβους) που περιέχει στοιχεία του ίδιου τύπου (δηλαδή ακέραιοι, πραγματικοί κτλ.)

Ένας πίνακας έχει κάποιο (α)**όνομα**, το οποίο το ορίζει ο προγραμματιστής, και συγκεκριμένο (β)**μέγεθος** (αριθμών κόμβων). Τόσο το όνομα όσο και το μέγεθος ενός πίνακα δηλώνονται στην παράγραφο ΜΕΤΑΒΛΗΤΕΣ του προγράμματος.

Η αναφορά στα στοιχεία ενός πίνακα γίνεται με τη χρήση του **ονόματος** του πίνακα ακολουθούμενου από την τιμή ενός ή περισσότερων **δεικτών (indexes)** σε παρένθεση ή αγκύλη.

Ο δείκτης είναι μία μεταβλητή που μπορεί να έχει οποιοδήποτε δεκτό όνομα. Είναι σύνηθες όμως στον Προγραμματισμό ως δείκτες να χρησιμοποιούνται οι μεταβλητές i, j, k.

Προσοχή: Οι λειτουργίες **εισαγωγή** και **διαγραφή** κόμβων **δεν** υλοποιούνται στους πίνακες γιατί, οι πίνακες είναι στατική δομή δεδομένων

Είδη πινάκων.

Ένας πίνακας μπορεί να είναι μονοδιάστατος, δισδιάστατος, τρισδιάστατος και γενικά ν-διάστατος πίνακας. Παραδείγματα:

1. Μονοδιάστατος πίνακας (γραμμή) με τιμές ακέραιες

1	2	3	4	5	6	7	8	9
-456	45	367	0	-21	-6	123	1	20

2. Μονοδιάστατος πίνακας (στήλη) με τιμές χαρακτήρες

1	'παγίδα'
2	'αύριο'
3	''
4	'Τρίτη'
5	'Άκης'

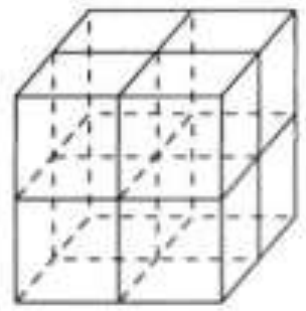
3. Δισδιάστατος πίνακας (3 γραμμών και 5 στηλών, 3Χ5)

	1	2	3	4	5
1	23,5	0,5	-1,4	17	2,0
2	-4	0,55	-0,82	1,111	55
3	3,14	2.723	1,86	-273	0,01

4. Δισδιάστατος τετραγωνικός πίνακας (3 γραμμών και 3 στηλών, 3Χ3)

	1	2	3
1	ΑΛΗΘΗΣ	ΨΕΥΔΗΣ	ΑΛΗΘΗΣ
2	ΨΕΥΔΗΣ	ΑΛΗΘΗΣ	ΑΛΗΘΗΣ
3	ΨΕΥΔΗΣ	ΨΕΥΔΗΣ	ΨΕΥΔΗΣ

5. Τρισδιάστατος πίνακας (2Χ2Χ2)



Παρατήρηση: Συχνά αποφεύγεται η χρήση πινάκων τριών ή περισσοτέρων διαστάσεων και τα προβλήματα αντιμετωπίζονται με χρήση μονοδιάστατων και δισδιάστατων πινάκων.

Μονοδιάστατοι πίνακες (μιας γραμμής ή μιας στήλης).

Ορισμός.

Ένας πίνακας ονομάζεται **μονοδιάστατος** όταν χρησιμοποιεί *ένα μόνο δείκτη* για την αναφορά των στοιχείων τους.

Οι κόμβοι σε ένα πίνακα συχνά αποκαλούνται θέσεις, κελιά, στοιχεία

Στο παράδειγμα βλέπουμε 2 πίνακες

πίνακας ακεραίων 5 κόμβων

ΠΙΝ	13	-27	8	46	-19
	1	2	3	4	5

πίνακας χαρακτήρων 4 κόμβων

Π	"Μαρία"	"Θωμάς"	"Χ1"	"31100"
	1	2	3	4

Το όνομα του πίνακα — **Μια τιμή του πίνακα** — **Ο δείκτης του πίνακα**

Ο πίνακας έχει **μοναδικό όνομα**, το οποίο δίνει ο προγραμματιστής (το όνομα ακολουθεί τους κανόνες που είδαμε στην ονοματολογία των μεταβλητών).

Για να αναφερθούμε στο περιεχόμενο (τιμή) ενός κόμβου χρησιμοποιούμε το **όνομα** του πίνακα και τον **δείκτη** (αριθμό θέσης) του κόμβου, δηλαδή

ΟΝΟΜΑ_ΠΙΝΑΚΑ [ΔΕΙΚΤΗΣ].

πχ.

ΠΙΝ[3] = το περιεχόμενο του 3ου κόμβου του πίνακα ΠΙΝ δηλ. **ΠΙΝ[3]** = 8.

Π[1] = το περιεχόμενο του 1ου κόμβου του πίνακα Π δηλ. **Π[1]** = "Μαρία".

Δήλωση πίνακα

Κάθε πίνακας δηλώνεται στο τμήμα δηλώσεων αναφέροντας τον τύπο του πίνακα (ακέραιος, πραγματικός, λογικός ή χαρακτήρας), το όνομά του και το μέγεθός του (αριθμό των στοιχείων του).

π.χ. Οι πίνακες των παραδειγμάτων θα δηλώνοντας ως εξής:

ΜΕΤΑΒΛΗΤΕΣ
ΑΚΕΡΑΙΕΣ: ΠΙΝ [5]
ΧΑΡΑΚΤΗΡΕΣ: Π[4]

Ενώ ένας μονοδιάστατος πραγματικός πίνακας 20 θέσεων με όνομα Χ, δηλώνεται

ΜΕΤΑΒΛΗΤΕΣ
ΠΡΑΓΜΑΤΙΚΕΣ: Χ [20]

Επεξεργασίες μονοδιάστατου πίνακα

1. Γέμισμα πίνακα από το πληκτρολόγιο (Διάβασμα πίνακα)

Ο αλγόριθμος

```
Αλγόριθμος Διάβασμα_Στοιχείων
!Πίνακας table 50 θέσεων
   Για i από 1 μέχρι 50
      Εμφάνισε "Δώσε το στοιχείο", i
      Διάβασε table[i]
   Τέλος_επανάληψης
Τέλος Διάβασμα_Στοιχείων
```

Το πρόγραμμα

```
ΠΡΟΓΡΑΜΜΑ Διάβασμα_Στοιχείων
   ΜΕΤΑΒΛΗΤΕΣ
      ΑΚΕΡΑΙΕΣ:table[50], i
   ΑΡΧΗ
      ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 50
         ΓΡΑΨΕ "Δώσε το στοιχείο", i
         ΔΙΑΒΑΣΕ table[i]
      ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
   ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Στις παρακάτω επεξεργασίες δίνεται μόνο ο αλγόριθμος για λόγους συντομίας.

2. Εμφάνιση των στοιχείων του πίνακα

```
Αλγόριθμος Εμφάνιση_Στοιχείων
Δεδομένα // table, N//
    Για i από 1 μέχρι N
        Εμφάνισε table[i]
    Τέλος_επανάληψης
Τέλος Εμφάνιση_Στοιχείων
```

3. Υπολογισμός του αθροίσματος

```
Αλγόριθμος Άθροισμα_Στοιχείων
Δεδομένα // table, N//
    Sum ← 0                          ! Αθροιστής
    Για i από 1 μέχρι N
        Sum ← sum + table[i]
    Τέλος_επανάληψης
    Εμφάνισε "Το άθροισμα είναι:", Sum
Τέλος Άθροισμα_Στοιχείων
```

4. Υπολογισμός του μέσου όρου

```
Αλγόριθμος Μέσος_Όρος
Δεδομένα // table, N//
    Sum ← 0                          ! Πρώτα θα υπολογίσουμε το άθροισμα
    Για i από 1 μέχρι N
        Sum ← sum + table[i]
    Τέλος_επανάληψης
    ! Ο Μ.όρος βρίσκεται διαιρώντας το άθροισμα δια του πλήθους
    ΜΟ←sum/N
    Εμφάνισε "Ο μέσος όρος είναι:", Sum
Τέλος Μέσος_Όρος
```

5. Υπολογισμός πλήθους των στοιχείων που ικανοποιούν μια ιδιότητα Κ

```
Αλγόριθμος Πλήθος_στοιχείων
Δεδομένα // table, N//
    πλ← 0                                ! Μετρητής
    Για i από 1 μέχρι N
    ! Ελέγχουμε αν κάθε στοιχείο του πίνακα ικανοποιεί την ιδιότητα
        Αν table[i] ικανοποίει ιδιότητα Κ τότε
                πλ ←πλ+1
        τελος_αν
    Τέλος_επανάληψης
    Εμφάνισε "Το πλήθος είναι:", πλ
Τέλος Πλήθος_στοιχείων
```

6. Εύρεση μεγίστου – ελαχίστου του πίνακα

```
Αλγόριθμος Max_στοιχείων
Δεδομένα // table, N//
    max ← table[1]          ! Θέτω το max ίσο με το 1ο στοιχείο του πίνακα
    Για i από 2 μέχρι N     ! Σαρώνω τον πίνακα
        ! Ελέγχω αν κάποιο στοιχείο το είναι μεγαλύτερο του max
        Αν table[i] > max τότε
            ! Αν είναι κρατάω αυτό σαν max
            max ←  table[i]
        τέλος_αν
    Τέλος_επανάληψης
    Εμφάνισε "Το μέγιστο είναι:", max
Τέλος Max_στοιχείων
```

Παρατήρηση:

Για την εύρεση του ελαχίστου (min) το μόνο που αλλάζει είναι η συνθήκη του **αν** που γίνεται: **Π[i] < min** (όπου min είναι το ελάχιστο)

7. Εύρεση μέγιστου και θέσης μεγίστου - ελαχίστου

Πολλές φορές μας ενδιαφέρει να βρούμε όχι μονό το μεγαλύτερο στοιχείο του πίνακα αλλά και σε ποιο κόμβο (θέση) αυτό βρίσκεται

```
Αλγόριθμος Max_θέση_στοιχείων
Δεδομένα // table, N//
    max ← table[1]
    ! Στην μεταβλητή θmax κρατάω την θέση που βρέθηκε το max
    θmax←1      ! Της δίνω αρχική τιμή 1 λόγω της εντολής max ← table[1]
    Για i από 2 μέχρι 50
        Αν table[i] > max τότε
            max ← table[i]
            θmax← i
        τέλος_αν
    Τέλος_επανάληψης
    Εμφάνισε "Μέγιστο =", max
    Εμφάνισε "στη θέση:", θmax
Τέλος Max_θέση_στοιχείων
```

Παρατηρήσεις:

1. Για την εύρεση του ελαχίστου (min) και της θέσης το μόνο που αλλάζει είναι η συνθήκη που γίνεται: **Π[i] < min**.
2. Αν το μέγιστο στοιχείο του πίνακα υπάρχει σε **πολλές θέσεις** τότε:
   i. Ο παραπάνω αλγόριθμος εντοπίζει την **<u>πρώτη θέση</u>** στην οποία βρίσκεται.
   ii. Για να εντοπίσουμε την **<u>τελευταία θέση</u>** τότε το μόνο που αλλάζουμε είναι στη συνθήκη προσθέτοντας και το ίσον (=), δηλαδή, Π[i] >= max

   <u>Εναλλακτική λύση εντοπισμού της τελευταίας θέσης είναι:</u>

   Να χρησιμοποιήσουμε τον παραπάνω αλγόριθμο <u>σαρώνοντας</u> τον πίνακα ανάποδα δηλαδή, από το τέλος προς την αρχή. Η εντολή επανάληψης διαμορφώνεται:

   ```
   max ←table [N]
   θmax ← N
   Για i από N μέχρι 1 με βήμα -1
   ```

   iii. Για να εμφανίσουμε **<u>όλες τις θέσεις</u>** στις οποίες εμφανίζεται το μέγιστο στοιχείο θα πρέπει να συνδυάσουμε:

   - τον **αλγόριθμο 6** για να βρούμε το μέγιστο μαχ και
   - τον **αλγόριθμο 5** για να εντοπίσουμε τις θέσεις που υπάρχει το μέγιστο.

   Άρα ο αλγόριθμος που θα βρίσκει την μέγιστη τιμή του πίνακα και θα εμφανίζει και όλες τις θέσεις που πιθανόν βρίσκεται αυτή η τιμή διαμορφώνεται ως εξής:

```
Αλγόριθμος Max_Θέσειςmax
Δεδομένα // table, N//
! Αρχικά βρίσκουμε το μέγιστο max. (Αλγόριθμος 6)
    max ← table[1]
    Για i από 2 μέχρι N
        Αν table[i] > max τότε
            max ← table[i]
        τέλος_αν
    Τέλος_επανάληψης
    Εμφάνισε "Το μέγιστο είναι:", max
! Αμέσως μετά εντοπίζουμε τις θέσεις του πίνακα που η τιμή είναι ίση με το max (αλγόριθμος 5)
    Για i από 1 μέχρι N
        Αν table[i] = max τότε
                Εμφάνισε " θέση:", i
        τελος_αν
    Τέλος_επανάληψης
Τέλος Max_στοιχείων
```

iv. Ανάλογα ισχύουν και για τον εντοπισμό της θέσης ή των θέσεων του ελαχίστου (min)

8. Αναζήτηση του στοιχείου key στον πίνακα table.

Πολλές φορές μας ενδιαφέρει να βρούμε αν κάποια συγκριμένη τιμή υπάρχει μέσα σε ένα πίνακα και αν ναι σε θέση του πίνακα βρίσκεται. Χρειάζεται λοιπόν να κάνουμε **αναζήτηση**.

Υπάρχουν περισσότεροι του ενός αλγόριθμοι αναζήτησης. Η επιλογή του κατάλληλου αλγόριθμου αναζήτησης εξαρτάται από αν ο πίνακας:

- είναι ταξινομημένος ή όχι
- περιέχει στοιχεία που είναι όλα διάφορα μεταξύ τους ή όχι

Οι δύο πιο σημαντικοί αλγόριθμοι αναζήτησης είναι:

- Ο αλγόριθμός της **σειριακής ή γραμμικής** αναζήτησης
- Ο αλγόριθμος της **δυαδικής** αναζήτησης.

Η σειριακή ή γραμμική αναζήτηση είναι η πιο *απλή* αλλά και *λιγότερο αποτελεσματική* μέθοδος αναζήτησης. Δικαιολογείται η χρήση της στις περιπτώσεις όπου:

- ✓ Ο πίνακας δεν είναι ταξινομημένος
- ✓ Ο πίνακας είναι μικρού μεγέθους (n ≤ 20)
- ✓ Η αναζήτηση στον πίνακα γίνεται σπάνια.

Η δυαδική μέθοδος αναζήτησης είναι η σαφώς *πιο αποδοτικότερη* από την σειριακή αλλά θα πρέπει ο πίνακας να είναι *ταξινομημένος*.

Σειριακή ή Γραμμική Αναζήτηση

Η **λογική** αυτής της μεθόδου είναι η εξής:

Σαρώνουμε τον πίνακα και εξετάζουμε, σε κάθε θέση, αν το στοιχείο της θέσης είναι ίσο με αυτό που αναζητάμε. Αν ναι σταματάμε την διαδικασία και επιστρέφουμε κατάλληλο μήνυμα.

Για την υλοποίηση αυτής της μεθόδου θα χρειαστούμε:

- μια *λογικού τύπου μεταβλητή* την **εύρηκα** (ή OK ή done) η οποία, αρχικά είναι ΨΕΥΔΗΣ και θα γίνει ΑΛΗΘΗΣ ΜΟΝΟ αν το στοιχείο βρεθεί.
- Επίσης μια *ακέραια μεταβλητή* την **θ** (ή θέση ή position)στην οποία θα εκχωρήσουμε την θέση στην οποία βρέθηκε το στοιχείο. Αρχικά η μεταβλητή αυτή θα έχει τιμή 0

Ο αλγόριθμος της σειριακής αναζήτησης είναι ο ακόλουθος.

```
Αλγόριθμος Σειριακή_Αναζήτηση
! table το όνομα του πίνακα, Ν το μέγεθος του, key η τιμή που αναζητάμε)
Δεδομένα// table, key, N//
    εύρηκα ← ψευδής
    θ ← 0
    i ← 1
    Όσο (i <=N) και (εύρηκα = ψευδής) επανάλαβε
        Αν table[i] = key τότε
            θ ← i
            εύρηκα ← αληθής
        αλλιώς
            i←i+1
        Τέλος_αν
    Τέλος_επανάληψης
    Αν εύρηκα = αληθής τότε
        Εμφάνισε "Το",key,"βρέθηκε στη θέση:", θ
    Αλλιώς
        Εμφάνισε " Το ",key,"δεν υπάρχει "
    Τέλος_αν
Τέλος  Σειριακή_Αναζήτηση
```

Ένα πρόγραμμα που χρησιμοποιεί την σειριακή αναζήτηση

```
ΠΡΟΓΡΑΜΜΑ Σειριακή_Αναζήτηση
ΜΕΤΑΒΛΗΤΕΣ
    ΑΚΕΡΑΙΕΣ:table[20],key,i,θ
    ΛΟΓΙΚΕΣ: εύρηκα
ΑΡΧΗ
! Αρχικά διαβάζω τον πίνακα
    ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 20
        ΓΡΑΨΕ 'Δώσε μια τιμή'
        ΔΙΑΒΑΣΕ table[i]
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
! διαβάζω επίσης και την τιμή αναζήτησης key
    ΓΡΑΨΕ 'Δώσε την τιμή αναζήτησης'
    ΔΙΑΒΑΣΕ key
! Ψάχνω αν η τιμή key βρίσκεται στον πίνακα table
    εύρηκα ← ψευδής
    θ ← 0
    i ← 1
    ΟΣΟ (i <=20) ΚΑΙ (εύρηκα = ψευδής) ΕΠΑΝΑΛΑΒΕ
        ΑΝ table[i] = key ΤΟΤΕ
            θ ← i
            εύρηκα ← αληθής
        ΑΛΛΙΩΣ
            i←i+1
        ΤΕΛΟΣ_ΑΝ
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
! Διατυπώνω την απάντηση
    ΑΝ εύρηκα = αληθής ΤΟΤΕ
        ΓΡΑΨΕ 'Το',key,'βρέθηκε στη θέση:', θ
    ΑΛΛΙΩΣ
        ΓΡΑΨΕ ' Το ',key,'δεν υπάρχει '
    ΤΕΛΟΣ_ΑΝ

ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Δυαδική Αναζήτηση

Η **λογική** αυτής της μεθόδου είναι η εξής:

Βήμα 1ο:Βρίσκουμε την θέση του μεσαίου κελίου του πίνακα (mid).

Βήμα 2ο: Συγκρίνουμε το στοιχείο αναζήτησης (key) με το μεσαίο κελί του πίνακα.

Βήμα 3ο Αν είναι ίσα (ΠΙΝ[mid] = key) αυτό, τότε το βρήκαμε και τελειώσαμε.

Βήμα 4ο Αν το key είναι μεγαλύτερο από το μεσαίο (ΠΙΝ[mid] < key)τότε περιοριζόμαστε στο πάνω μισό του πίνακα.

Βήμα 5ο Αν το key είναι μικρότερο από το μεσαίο (ΠΙΝ[mid] > key)τότε περιοριζόμαστε στο κάτω μισό του πίνακα.

Βήμα 6ο: Αν το key δεν βρέθηκε (Βήμα 4ο και Βήμα 5ο) τότε επαναλαμβάνουμε τα προηγούμενα βήματα αρχίζοντας από το Βήμα 1ο, στο τμήμα του πίνακα που ορίσθηκε από τα βήματα 4 ή 5

Θα χρειαστούμε μια *λογικού τύπου μεταβλητή* **εύρηκα** η οποία θα γίνει ΑΛΗΘΗΣ αν το στοιχείο βρεθεί. Επίσης μια *ακέραια μεταβλητή* **θ** στην οποία εκχωρούμε την θέση στην οποία βρέθηκε το στοιχείο key που ψάχνουμε.

Ο αλγόριθμος της δυαδικής αναζήτησης είναι ο ακόλουθος.

```
αλγόριθμος Δυαδική_αναζήτηση
    !table μονοδιάστατος πίνακας Ν θέσεων, key το στοιχείο αναζήτησης
    !Δεχόμαστε ότι ο πίνακας table είναι ταξινομημένος σε αύξουσα διάταξη
δεδομένα // table, N, key //
    ! Οι μεταβλητές left & right ορίζουνε το τμήμα του πίνακα στο οποίο αναζητούμε το key
    ! Δεχόμαστε ότι ο πίνακας table είναι ταξινομημένος σε αύξουσα διάταξη
    ! Αρχικά τους δίνουμε κατάλληλες τιμές ώστε να ψάχνουμε σε ολόκληρο τον πίνακα
        left ← 1                        ! αριστερό άκρο του πίνακα
        right ← N                       ! δεξιό άκρο του πίνακα
        θ ←  0                          ! θέση του στοιχείου
        εύρηκα ← ψευδής
        όσο (left<=right) και (εύρηκα=ψευδής) επανάλαβε
            mid ← (left+right) div 2    ! υπολογίζω την θέση του μεσαίου στοιχείου
            αν table[mid]=key τότε      ! βρέθηκε στην θέση mid
                    θ← mid;
                    εύρηκα ← αληθής;
            αλλιώς
                    αν table[mid]<key τότε   !περιορίζομαι στο πάνω μισό του πίνακα
                        left ← mid+1
                    αλλιώς
                        right← mid-1         ! περιορίζομαι στο κάτω μισό του πίνακα
                    Τέλος_αν
            Τέλος_αν
        Τέλος_επανάληψης
        Αν εύρηκα = αληθής τότε
            Εμφάνισε "Το", key , "υπάρχει στη θέση:", θ
        Αλλιώς
            Εμφάνισε "Το", key , " δεν υπάρχει "
        Τέλος_αν
Τέλος Δυαδική_Αναζήτηση
```

Ένα πρόγραμμα που χρησιμοποιεί την σειριακή αναζήτηση.

```
ΠΡΟΓΡΑΜΜΑ δυαδική_αναζήτηση
ΜΕΤΑΒΛΗΤΕΣ
    ΑΚΕΡΑΙΕΣ: table[20], θ, left, right, mid, key, i
    ΛΟΓΙΚΕΣ: εύρηκα
ΑΡΧΗ
    ! Αρχικά διαβάζω τον πίνακα δίνοντας κάθε φορά μια μεγαλύτερη τιμή από την προηγούμενη, ώστε ο πίνακας table να προκύψει ταξινομημένος σε αύξουσα διάταξη.
    ΓΡΑΨΕ 'ΠΡΟΣΟΧΗ: Δώσε τα στοιχεία του πίνακα σε αύξουσα διάταξη'
    ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 20
        ΓΡΑΨΕ 'Δώσε το',i,' στοιχείο του πίνακα'
        ΔΙΑΒΑΣΕ A[i]
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
    ! διαβάζω επίσης και την τιμή αναζήτησης key
    ΓΡΑΨΕ 'Δώσε τιμή για αναζήτηση: '
    ΔΙΑΒΑΣΕ key
    left ← 1
    Right ← 20
    θ ← 0
    εύρηκα ← ΨΕΥΔΗΣ
    ΟΣΟ (left <= right) ΚΑΙ (εύρηκα = ΨΕΥΔΗΣ) ΕΠΑΝΑΛΑΒΕ
            mid ← (left + right) DIV 2
            ΑΝ table[mid] = key ΤΟΤΕ
                θ ← mid
                εύρηκα ← ΑΛΗΘΗΣ
            ΑΛΛΙΩΣ
                ΑΝ table[mid] < key ΤΟΤΕ
                    left ← mid + 1
                ΑΛΛΙΩΣ
                    right ← mid – 1
                ΤΕΛΟΣ_ΑΝ
            ΤΕΛΟΣ_ΑΝ
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
    ΑΝ εύρηκα ΤΟΤΕ
            ΓΡΑΨΕ "Το στοιχείο,", key, "υπάρχει στη θέση:", θ
    ΑΛΛΙΩΣ
            ΓΡΑΨΕ "Το στοιχείο,", key, " δεν υπάρχει"
    ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

9. Ταξινόμηση μονοδιάστατου πίνακα.

Η τακτοποίηση των κόμβων μίας δομής με μία ιδιαίτερη σειρά είναι μία πολύ σημαντική λειτουργία που ονομάζεται ταξινόμηση (sorting) ή διάταξη (ordering).

Συνήθως η σειρά αυτή είναι η αύξουσα τάξη της τιμής των μεγεθών προς ταξινόμηση. Μπορεί όμως σε κάποια περίπτωση να χρειαστεί να κάνουμε φθίνουσα ταξινόμηση.

Σκοπός της ταξινόμησης είναι να διευκολυνθεί στη συνέχεια η αναζήτηση των στοιχείων του ταξινομημένου πίνακα

Υπάρχουν αρκετοί αλγόριθμοι ταξινόμησης. Οι πιο σημαντικοί είναι:

1. η ταξινόμηση *φυσαλίδας* ή *ευθείας ανταλλαγής*
2. η ταξινόμηση με *επιλογή*,
3. η ταξινόμηση με *παρεμβολή*
4. η *γρήγορη ταξινόμηση*

Η επιλογή του καλύτερου αλγόριθμου ταξινόμησης εξαρτάται από:

1ον .κυρίως από το πλήθος των κελιών του

2ον την αρχική διάταξη των στοιχειών του (τελείως αταξινόμητος ή μερικώς ταξινομημένος)

Ταξινόμηση φυσαλίδας ή ευθείας ανταλλαγής

- Είναι ο πιο **απλός** και ταυτόχρονα ο πιο **αργός** αλγόριθμος ταξινόμησης
- Βασίζεται στην αρχή της σύγκρισης γειτονικών στοιχείων του πίνακα και ανταλλαγή τους μέχρι να διαταχθούν όλα σε μια σειρά.
- Η **λογική** αυτής της μεθόδου είναι η εξής: κάνουμε διαδοχικές σαρώσεις στον πίνακα. Σε κάθε σάρωση, το μικρότερο στοιχείο (για αύξουσα) μετακινείται προς την κορυφή του πίνακα. Αφού γίνουν όλες οι σαρώσεις θα επιτευχθεί η ταξινόμηση.
- Ο αλγόριθμος της ταξινόμησης φυσαλίδας

Αλγόριθμος Ταξινόμηση_Φυσαλίδας_Αύξουσα

! table ο πίνακα, N το μέγεθος του

Δεδομένα// table, N//

Για i **από** 2 **μέχρι** N *! Θα κάνουμε N-1 σαρώσεις στον table*

! σε κάθε μία σάρωση θα κάνουμε ένα αριθμό συγκρίσεων

Για j **από** N **μέχρι** i **με_βήμα** -1

Αν table [j-1] > table [j] **τότε**

!αντιμεταθέτουμε τα στοιχεία! table [j-1] και table [j]

temp ← table [j-1]

table [j-1] ← table [j]

table [j] ← temp

Τέλος_αν

Τέλος_επανάληψης

Τέλος_επανάληψης

Αποτελέσματα//table//

Τέλος Ταξινόμηση_Φυσαλίδας_Αύξουσα

- *Ένα πρόγραμμα που κάνει αύξουσα ταξινόμηση.*

```
ΠΡΟΓΡΑΜΜΑ Ταξινόμηση_Φυσαλίδας
ΜΕΤΑΒΛΗΤΕΣ
    ΑΚΕΡΑΙΕΣ:table[20],i,j,temp
ΑΡΧΗ
    ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 20
        ΔΙΑΒΑΣΕ table[i]
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
    ΓΙΑ i ΑΠΟ 2 ΜΕΧΡΙ 20
        ΓΙΑ j ΑΠΟ 20 ΜΕΧΡΙ i ΜΕ ΒΗΜΑ -1
                ΑΝ table [j-1] > table [j] ΤΟΤΕ
                    temp ← table [j-1]
                    table [j-1] ← table [j]
                    table [j] ← temp
                ΤΕΛΟΣ_ΑΝ
        ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
    ΓΡΑΨΕ 'Εκτύπωση ταξινομημένου πίνακα'
    ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 20
        ΓΡΑΨΕ table[i]
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

➢ *Κάνοντας τον αλγόριθμο φυσαλίδας «έξυπνο»*

Αν θέλουμε να κάνουμε τον παραπάνω αλγόριθμό πιο «έξυπνο», ώστε να σταματάει την διαδικασία αν ο πίνακας ταξινομηθεί πριν γίνουν όλες οι σαρώσεις θα πρέπει::

- ✓ Να χρησιμοποιήσουμε μια λογική μεταβλητή λογικού τύπου την **ΟΚ** με τιμή ΨΕΥΔΕΣ αν ο πίνακας είναι αταξινόμητος και ΑΛΗΘΕΣ αν ο πίνακας είναι ταξινομημένος.
- ✓ Να αντικαταστήσουμε την εξωτερική ΓΙΑ με μια ΟΣΟ.

Ο αλγόριθμος φυσαλίδας θα γίνει:

```
Αλγόριθμος Έξυπνη_Ταξινόμηση_Φυσαλίδας
Δεδομένα// table, N//
    i ← 2
    OK ← ψευδές
    Όσο (i <= N) και (OK = ψευδές) επανάλαβε
        OK ← αληθές
        Για j από N μέχρι i με_βήμα -1
            Αν table [j-1] > table [j] τότε
                temp ← table [j-1]
                table [j-1] ← table [j]
                table [j] ← temp
            Τέλος_αν
        Τέλος_επανάληψης
        i ← i + 1
    Τέλος_επανάληψης
Αποτελέσματα//table//
Τέλος Έξυπνη_Ταξινόμηση_Φυσαλίδας
```

Ταξινόμηση με επιλογή

Η μέθοδος ταξινόμησης με επιλογή είναι αρκετά πιο αποδοτική, ειδικά αν ο πίνακας είναι μερικώς ταξινομημένος.

Η **λογική** αυτής της μεθόδου είναι η εξής:

Αρχικά βρίσκουμε το μικρότερο στοιχείο του πίνακα και το ανταλλάσσουμε με το 1ο στοιχείο του. Μετά βρίσκουμε το αμέσως μικρότερο και το ανταλλάσσουμε με το 2ο στοιχείο του. Μετά βρίσκουμε το αμέσως μικρότερο και το ανταλλάσσουμε με το 3ο στοιχείο του κ.ο.κ.

Η παραπάνω λογική αφορά αύξουσα ταξινόμηση. Για φθίνουσα ταξινόμηση , αντί να βρίσκουμε το μικρότερο στοιχείο, βρίσκουμε το μεγαλύτερο.

Ακολουθεί ο αλγόριθμος της ταξινόμησης με επιλογή

Αλγόριθμος Ταξινόμηση_Με_Επιλογή_Αύξουσα
! table ο πίνακα, Ν το μέγεθος του
Δεδομένα // table, N //
! Θα κάνουμε Ν-1 σαρώσεις του πίνακα table
Για i **από** 1 **μέχρι** N-1
! αρχικοποιούμε την μεταβλητή θ με τον δείκτη του 1ου στοιχείου της νέας σάρωσης
θ ← i
! διατρέχουμε τον table από το 2ο στοιχείο κάθε σάρωσης έως το τέλος του
Για j από i+1 **μέχρι** N
! συγκρίνουμε τα στοιχεία των θέσεων θ και j
Αν table[θ] > table[j] **Τότε**
θ ← j
τέλος_αν
Τέλος_Επανάληψης
!αντιμεταθέτουμε τα στοιχεία των θέσεων θ και i
βοηθ← table[θ]
table[θ] ← table[i]
table[i] ← βοηθ
Τέλος_επανάληψης
Αποτελέσματα //table//
Τέλος Ταξινόμηση_Με_Επιλογή_Αύξουσα

Και ένα πρόγραμμα που κάνει ταξινόμηση με επιλογή

ΠΡΟΓΡΑΜΜΑ Ταξινόμηση_Με_Επιλογή
ΜΕΤΑΒΛΗΤΕΣ
ΑΚΕΡΑΙΕΣ: Α**[**20**]**, θ**,** βοηθ**,** i**,** j
ΑΡΧΗ
ΓΙΑ i **ΑΠΟ** 1 **ΜΕΧΡΙ** 20
ΔΙΑΒΑΣΕ Α**[**i**]**
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΓΙΑ i **ΑΠΟ** 1 **ΜΕΧΡΙ** 19
θ ← i
ΓΙΑ j **ΑΠΟ** i **+** 1 **ΜΕΧΡΙ** 20
ΑΝ table[θ] **>** Α**[**j**] ΤΟΤΕ**
θ ← j
ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
βοηθ← Α**[**θ**]**
Α**[**θ**]** ← Α**[**i**]**
Α**[**i**]** ← βοηθ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΓΡΑΨΕ 'Εκτύπωση με ταξινομημένα τα στοιχεία'
ΓΙΑ i **ΑΠΟ** 1 **ΜΕΧΡΙ** 20
ΓΡΑΨΕ Α**[**i**]**
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ

Πλεονεκτήματα – Μειονεκτήματα χρήσης των πινάκων

- **Πλεονεκτήματα.** Είναι ένας βολικός και εύκολος τρόπος διαχείρισης πολλών δεδομένων του ίδιου τύπου.
- **Μειονεκτήματα.**
  α. Οι πίνακες απαιτούν μνήμη. Ο πίνακας δεσμεύει από την αρχή του προγράμματος πολλές θέσεις μνήμης. Έτσι, η χρήση πολλών πινάκων σε ένα πρόγραμμα μπορεί να οδηγήσει και σε αδυναμία εκτέλεσης του.

  β. Οι πίνακες περιορίζουν τις δυνατότητες του προγράμματος. Επειδή ο πίνακας είναι στατική δομή το μέγεθος του πρέπει να δηλώνεται στην αρχή του προγράμματος και παραμένει σταθερό κατά την εκτέλεση του προγράμματος.
- **Πότε πρέπει να χρησιμοποιώ πίνακες;**
  Όταν τα δεδομένα που εισάγονται σε ένα πρόγραμμα πρέπει να παραμείνουν στη μνήμη RAM μέχρι το τέλος της εκτέλεσης, τότε η χρήση πινάκων βοηθάει ή είναι απαραίτητη για την επίλυση του προβλήματος.