Ανάπτυξη εφαρμογών σε προγραμματιστικό περιβάλλον

Γ’ Λυκείου

ΚΕΦΑΛΑΙΟ 9

Πίνακες

Χρήστος Μουρατίδης - Έκδοση 2021
mouratx@yahoo.com
http://users.sch.gr/mouratx

Περιεχόμενα

Εισαγωγή

Σε πολλά προβλήματα χρειάζεται να επεξεργαστούμε μία ομάδα από ομοειδή δεδομένα για παράδειγμα, να υπολογίσουμε το μέσο όρο 20 ακέραιων αριθμών ή να βρούμε τον μεγαλύτερο βαθμό των μαθητών της τάξης.

Προκειμένου να γίνει η επεξεργασία αυτών των ομοειδών δεδομένων, αυτά θα πρέπει να έχουν αποθηκευτεί στη μνήμη. Η χρήση διαφορετικών μεταβλητών για καθένα από τα δεδομένα αυτά κάνει δύσχρηστη την πρόσβαση και επεξεργασία τους. Οι γλώσσες προγραμματισμού διαθέτουν τη δομή του πίνακα, όπου χρησιμοποιούμε ένα όνομα μεταβλητής και δείκτη για την πρόσβαση σε κάθε θέση του.[1]

Για παράδειγμα, αν θέλουμε να αποθηκεύσουμε 20 αριθμούς, αντί να χρησιμοποιήσουμε 20 διαφορετικές μεταβλητές χρησιμοποιούμε μία μεταβλητή πίνακα με 20 θέσεις.

Στο Κεφάλαιο αυτό θα δούμε μονοδιάστατους και δισδιάστατους πίνακες καθώς και κάποιες τυπικές επεξεργασίες πινάκων, όπως εύρεση αθροισμάτων, μέσων όρων, μεγίστου κ.λπ.

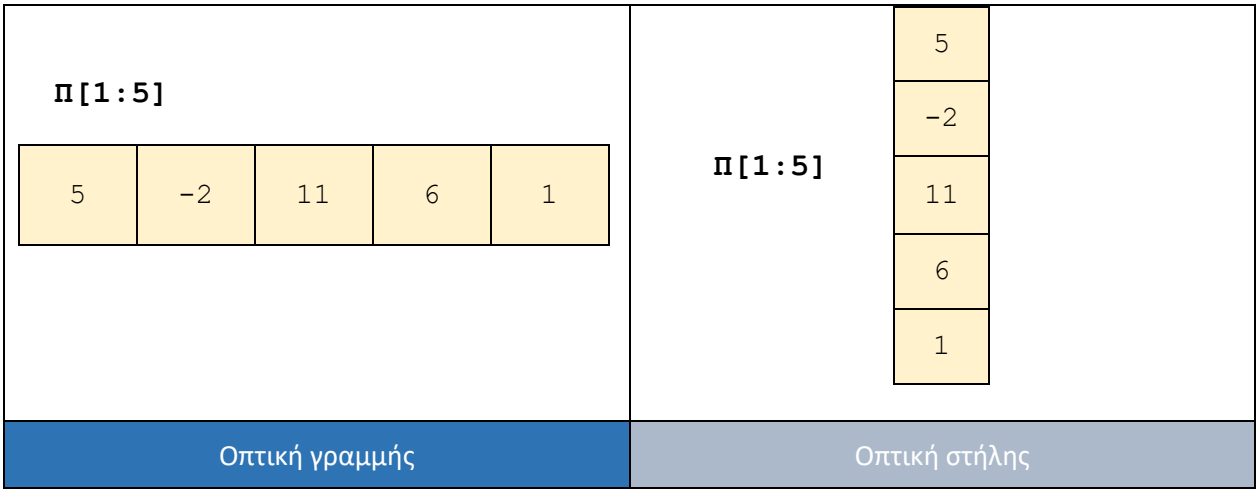
Μονοδιάστατοι πίνακες

Ένας μονοδιάστατος πίνακας είναι ένα **σύνολο συνεχόμενων θέσεων στη μνήμη**, όπου οπτικά αναπαρίσταται ως μία γραμμή ή μία στήλη.

> **Ορισμός** (*από το σχολικό βιβλίο*):
>
> Πίνακας είναι ένα **σύνολο αντικειμένων ίδιου τύπου**, τα οποία αναφέρονται με ένα κοινό όνομα. Κάθε ένα από τα αντικείμενα που απαρτίζουν τον πίνακα λέγεται **στοιχείο** του πίνακα.
>
> Η αναφορά σε ατομικά στοιχεία του πίνακα γίνεται με το όνομα του πίνακα ακολουθούμενο από ένα δείκτη.

[1] Στους πολυδιάστατους πίνακες χρησιμοποιούμε περισσότερους από έναν δείκτες θέσης.

Παρακάτω, βλέπουμε έναν πίνακα 5 θέσεων, με όνομα **Π**. Ο πίνακας περιέχει ακεραίους:



Στα επόμενα, θα χρησιμοποιήσουμε την οπτική γραμμής.

Κάθε θέση του πίνακα ονομάζεται στοιχείο του πίνακα και προσδιορίζεται από την τιμή ενός δείκτη. Με άλλα λόγια, χρησιμοποιούμε μία **ειδική μεταβλητή ή σταθερά**, που ονομάζεται **δείκτης θέσης**[2]**,** για να προσδιορίσουμε την θέση ενός πίνακα.

Π	5	-2	11	6	1
Θέσεις:	1η	2η	3η	4η`	5η

Για μονοδιάστατο πίνακα χρειάζεται ένας μόνο δείκτης θέσης για να προσδιορίσουμε ένα στοιχείο του. Ο δισδιάστατος, που θα δούμε αργότερα, χρειάζεται δύο δείκτες.

Το **περιεχόμενο μίας θέσης καθορίζεται από την εξής σύνταξη**:

```
Π[δείκτης]
```

Για παράδειγμα, `Π[1]` αναφέρεται στον αριθμό 5, `Π[3]` στον αριθμό 11.

[2] Είναι σύνηθες στον προγραμματισμό να χρησιμοποιούμε τα ονόματα `i, j, k` ως ονόματα μεταβλητών για δείκτες θέσης, σε συνδυασμό με εντολές επανάληψης.

Παραδείγματα επεξεργασίας μονοδιάστατου πίνακα

Διάβασμα στοιχείων

Να γραφεί πρόγραμμα στη ΓΛΩΣΣΑ που διαβάζει τους (μηνιαίους) μισθούς των 50 εργαζομένων μίας εταιρείας σε έναν πίνακα. Για κάθε μισθό που διαβάζεται πρέπει να ελέγχεται αν είναι θετικός αριθμός και κάτω του 3000.

Στις περισσότερες περιπτώσεις χρησιμοποιούμε μία εντολή **ΓΙΑ..ΑΠΟ..ΜΕΧΡΙ** για το διάβασμα (εισαγωγή) των δεδομένων σε έναν πίνακα, καθόσον ο αριθμός των θέσεών του είναι προκαθορισμένος.

```
ΠΡΟΓΡΑΜΜΑ Διάβασμα_μισθών_σε_πίνακα
ΜΕΤΑΒΛΗΤΕΣ
     ΠΡΑΓΜΑΤΙΚΕΣ: Μισθός[50]
     ΑΚΕΡΑΙΕΣ: i    !μεταβλητή για τον δείκτη θέσης.
ΑΡΧΗ
     !Είσοδος δεδομένων, εδώ των 50 μισθών σε πίνακα.
     ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 50

        !Τυπική περίπτωση ελέγχου εγκυρότητας.
        ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ

           ΓΡΑΨΕ 'Παρακαλώ, δώστε τον μισθό:'
           ΔΙΑΒΑΣΕ Μισθός [i]

           ΑΝ ΟΧΙ(Μισθός[i] > 0 ΚΑΙ Μισθός[i] < 3000) ΤΟΤΕ

             ΓΡΑΨΕ 'Ο μισθός που δώσατε δεν είναι έγκυρος.
                    & Παρακαλώ ξαναδώστε'
           ΤΕΛΟΣ_ΑΝ

        ΜΕΧΡΙΣ_ΟΤΟΥ Μισθός[i] > 0 ΚΑΙ Μισθός[i] < 3000

     ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

     !Επιβεβαιωτικό ολοκλήρωσης εισόδου.
     ΓΡΑΨΕ 'Η εισαγωγή των μισθών ολοκληρώθηκε.'
```

ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ

Ο **κύριος βρόχος** είναι ο **ΓΙΑ..ΑΠΟ..ΜΕΧΡΙ** όπου διαβάζονται επαναληπτικά ένας-ένας οι μισθοί. Στο εσωτερικό αυτού υπάρχει ένας **βρόχος ΜΕΧΡΙΣ_ΟΤΟΥ** για τον **έλεγχο του εισαγόμενου μισθού**. Όσο ο χρήστης δεν δίνει έγκυρο ποσό το πρόγραμμα δεν προχωράει και του ζητάει συνεχώς να εισάγει έγκυρο ποσό.
Στο Κεφάλαιο 8, αναφέραμε εκτεταμένα τις σχετικές περιπτώσεις ενδεδειγμένης χρήσης της **ΜΕΧΡΙΣ_ΟΤΟΥ** και καλό είναι ο αναγνώστης να τις έχει σοβαρά υπόψιν του.

Υπολογισμός αθροίσματος

Να γραφεί πρόγραμμα στη ΓΛΩΣΣΑ που διαβάζει τους (μηνιαίους) μισθούς των 50 εργαζομένων μίας εταιρείας σε έναν πίνακα. Σκοπός είναι να υπολογίσει το άθροισμα των μισθών ώστε να ξέρει η εταιρεία τί συνολικό ποσό καλείται να πληρώσει στο τέλος του μήνα. Υποθέτουμε ότι οι μισθοί που εισάγονται είναι σωστοί (δεν χρειάζεται έλεγχος εγκυρότητας).

Θα χρειαστούμε μία **μεταβλητή-συσσωρευτή (αθροιστή)**. Συνήθως, δίνουμε το **όνομα Sum**. Για κάθε μισθό που διαβάζουμε τον προσθέτουμε στον αθροιστή Sum.

```
ΠΡΟΓΡΑΜΜΑ Άθροισμα_μισθών_ενός_πίνακα
ΜΕΤΑΒΛΗΤΕΣ
     ΠΡΑΓΜΑΤΙΚΕΣ: Μισθός[50], Sum
     ΑΚΕΡΑΙΕΣ: i     !μεταβλητή για τον δείκτη θέσης.
ΑΡΧΗ

     !Δεν ξεχνάμε την αρχικοποίηση του αθροιστή.
     Sum ← 0

     !Είσοδος δεδομένων, εδώ των 50 μισθών σε πίνακα.
     ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 50

          ΓΡΑΨΕ 'Παρακαλώ, δώστε τον ', i, 'ο μισθό:'
          ΔΙΑΒΑΣΕ Μισθός[i]

     ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

     !Προσπελαύνει έναν-έναν μισθό από τον πίνακα και τον
     !προσθέτει στον αθροιστή.
     ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 50

          Sum ← Sum + Μισθός[i]
```

```
      ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

      !Εμφάνιση του αποτελέσματος στην οθόνη.
      ΓΡΑΨΕ 'Το συνολικό ποσό των μισθών είναι: ', Sum

ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Φυσικά, θα μπορούσαμε κατά το διάβασμα των μισθών να τους προσθέταμε αμέσως στον αθροιστή (με έναν βρόχο) αλλά εδώ ο σκοπός είναι να δείξουμε την επεξεργασία (άθροισμα) που πραγματοποιείται *όταν τα δεδομένα είναι ήδη στον πίνακα*.

Υπολογισμός μέσου όρου

Η εύρεση του μέσου όρου προϋποθέτει ότι έχουμε ήδη βρει το άθροισμα των δεδομένων στον πίνακα (ο αριθμός των δεδομένων είναι ο αριθμός των θέσεων του πίνακα).

Σε συνέχεια του προηγούμενου παραδείγματος (του υπολογισμού αθροίσματος), η εταιρεία επιθυμεί, επίσης, να μάθει και τον μέσο όρο των μισθών.

Ο κώδικας δεν διαφέρει και πολύ από τον προηγούμενο. Απλά, προσθέτουμε κώδικα για τον υπολογισμό του μέσου όρου.

```
ΠΡΟΓΡΑΜΜΑ Μέσος_όρος_μισθών_ενός_πίνακα
ΜΕΤΑΒΛΗΤΕΣ
      ΠΡΑΓΜΑΤΙΚΕΣ: Μισθός[50], Sum, ΜΟ
      ΑΚΕΡΑΙΕΣ: i     !μεταβλητή για τον δείκτη θέσης.
ΑΡΧΗ

      !Δεν ξεχνάμε την αρχικοποίηση του αθροιστή.
      Sum ← 0

      !Είσοδος δεδομένων, εδώ των 50 μισθών σε πίνακα.
      ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 50

            ΓΡΑΨΕ 'Παρακαλώ, δώστε τον ', i, 'ο μισθό:'
            ΔΙΑΒΑΣΕ Μισθός[i]

      ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

      !Προσπελαύνει έναν-έναν μισθό από τον πίνακα και τον
      !προσθέτει στον αθροιστή.
      ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 50

            Sum ← Sum + Μισθός[i]
```

```
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

    !Υπολογισμός του μέσου όρου.
    ΜΟ ← Sum / 50

    !Εμφάνιση του αποτελέσματος στην οθόνη.
    ΓΡΑΨΕ 'Ο μέσος όρος των μισθών είναι: ', ΜΟ

ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Καταμέτρηση στοιχείων

Μία τυπική επεξεργασία σε έναν πίνακα είναι να μετρήσουμε το πλήθος κάποιων στοιχείων που πληρούν κάποιο/α κριτήριο/α.

Να γραφεί πρόγραμμα στη ΓΛΩΣΣΑ που διαβάζει τους (μηνιαίους) μισθούς των 50 εργαζομένων μίας εταιρείας σε έναν πίνακα. Σκοπός είναι να υπολογίσει πόσοι είναι οι «μεγάλοι» μισθοί, δηλαδή πόσοι είναι πάνω από ή ίσοι με 2000 €.

Εδώ, σε κάθε επανάληψη πρέπει να ελέγχουμε αν ο τρέχον μισθός (της i θέσης του πίνακα) ικανοποιεί το κριτήριο, δηλαδή να είναι ίσοι ή πάνω από 2000. Αν το ικανοποιεί τότε αυξάνουμε την τιμή μίας **μεταβλητής-μετρητή**.

```
ΠΡΟΓΡΑΜΜΑ Πλήθος_υψηλών_μισθών_ενός_πίνακα
ΜΕΤΑΒΛΗΤΕΣ
    ΠΡΑΓΜΑΤΙΚΕΣ: Μισθός[50]
    ΑΚΕΡΑΙΕΣ: i, μετρητής
ΑΡΧΗ

    !Δεν ξεχνάμε την αρχικοποίηση του μετρητή.
    μετρητής ← 0

    !Είσοδος δεδομένων, εδώ των 50 μισθών σε πίνακα.
    ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 50

        ΓΡΑΨΕ 'Παρακαλώ, δώστε τον ', i, 'ο μισθό:'
        ΔΙΑΒΑΣΕ Μισθός[i]

    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

    !Προσπελαύνει έναν-έναν μισθό από τον πίνακα και ελέγχει
    !αν ικανοποιεί το κριτήριο.
```

```
      ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 50

            ΑΝ Μισθός[i] >= 2000 ΤΟΤΕ
                  μετρητής ← μετρητής + 1
            ΤΕΛΟΣ_ΑΝ

      ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

      !Εμφάνιση του αποτελέσματος στην οθόνη.
      ΓΡΑΨΕ 'Το πλήθος των υψηλών μισθών είναι: ', μετρητής

ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Φυσικά, κριτήρια μπορούν να υπάρχουν και στις υπόλοιπες επεξεργασίες, του υπολογισμού αθροίσματος και μέσου όρου. Παρόμοια, για παράδειγμα, να βρούμε το συνολικό ποσό που πρέπει να πληρώσει η εταιρεία στους υψηλόμισθους (>=2000 €).

Εύρεση του μέγιστου στοιχείου

Να γραφεί πρόγραμμα στη ΓΛΩΣΣΑ που διαβάζει τα ονοματεπώνυμα και τις μηνιαίες πωλήσεις των 50 εργαζομένων μίας εταιρείας σε δύο πίνακες, αντίστοιχα. Στη συνέχεια, θα υπολογίσει κι εμφανίσει στην οθόνη ποιός εργαζόμενος πραγματοποίησε τις μεγαλύτερες πωλήσεις («*ο πωλητής του μήνα*»).

Στο Κεφάλαιο 3 έχουμε δει τον σχετικό αλγόριθμο εύρεσης μεγίστου στοιχείου.

- Θα χρειαστούμε μία μεταβλητή που θα κρατά το μέγιστο στοιχείο πώλησης (π.χ. `Max`). Η αρχική τιμή αυτής της μεταβλητής τίθεται από το πρώτο στοιχείο του πίνακα.

- Στη συνέχεια, θα σαρωθεί ο πίνακας των πωλήσεων διαδοχικά ένα προς ένα στοιχείο και θα συγκρίνεται με την μέγιστη πώληση. Αν το τρέχον στοιχείο του πίνακα είναι μεγαλύτερο από εκείνο της `Max` τότε στην `Max` θα εκχωρείται το τρέχον στοιχείο του πίνακα.

- Στο τέλος, η `Max` θα περιέχει το μέγιστο στοιχείο.

```
ΠΡΟΓΡΑΜΜΑ Ο_πωλητής_του_μήνα
ΜΕΤΑΒΛΗΤΕΣ
      ΠΡΑΓΜΑΤΙΚΕΣ: Πώληση[50], Max_Πώληση
      ΧΑΡΑΚΤΗΡΕΣ: Ονοματεπώνυμο[50], Max_Ονοματεπώνυμο
```

```
     ΑΚΕΡΑΙΕΣ: i
ΑΡΧΗ

     !Είσοδος δεδομένων, εδώ των 50 ονοματεπωνύμων και πωλήσεων
     !σε δύο πίνακες, αντίστοιχα.
     ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 50

          ΓΡΑΨΕ 'Παρακαλώ, δώστε το ', i, 'ο ονοματεπώνυμο:'
          ΔΙΑΒΑΣΕ Ονοματεπώνυμο[i]
          ΓΡΑΨΕ 'Παρακαλώ, δώστε την ', i, 'η πώληση:'
          ΔΙΑΒΑΣΕ Πώληση[i]

     ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ


     !Θέτουμε στις μεταβλητές Max το πρώτο στοιχείο από κάθε
     !πίνακα.
     Max_Ονοματεπώνυμο ← Ονοματεπώνυμο[1]
     Max_Πώληση ← Πώληση[1]

     !Προσπελαύνει μία-μία πώληση από τον πίνακα πωλήσεων και
     !ελέγχει αν η τρέχουσα πώληση είναι μεγαλύτερη της Max.
     ΓΙΑ i ΑΠΟ 2 ΜΕΧΡΙ 50

          ΑΝ Πώληση[i] > Max_Πώληση ΤΟΤΕ
               Max_Πώληση ← Πώληση[i]
               Max_Ονοματεπώνυμο ← Ονοματεπώνυμο[i]
          ΤΕΛΟΣ_ΑΝ

     ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

     !Εμφάνιση του αποτελέσματος στην οθόνη.
     ΓΡΑΨΕ 'Ο πωλητής του μήνα είναι ο/η: ', Max_Ονοματεπώνυμο
     ΓΡΑΨΕ 'και οι πωλήσεις που έκανε είναι: ', Max_Πώληση

ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Φυσικά, το πρόγραμμα απαιτεί βελτίωση διότι τίθεται το εξής **ερώτημα**:

- Τί γίνεται αν δύο ή περισσότεροι πωλητές/τριες πραγματοποιήσουν ισόποσες πωλήσεις; Το πρόγραμμα κρατάει μόνο έναν (ονοματεπώνυμο και πώληση στις αντίστοιχες `Max` μεταβλητές).
  Ο αναγνώστης καλείται να σκεφτεί μία λύση.

Εύρεση του ελάχιστου στοιχείου

Παρόμοιος είναι ο τρόπος εύρεσης του ελάχιστου στοιχείου. Ο αναγνώστης μπορεί να βασιστεί στο προηγούμενο παράδειγμα για να κατασκευάσει πρόγραμμα που θα υπολογίζει τον «*χειρότερο πωλητή του μήνα*».

Αναζήτηση στοιχείου

Η αναζήτηση στοιχείου είναι από τις συνηθέστερες επεξεργασίες ενός πίνακα. Δύο είναι οι πιο γνωστοί αλγόριθμοι:

1. **Σειριακή (ή ακολουθιακή) αναζήτηση**. Εφαρμόζεται υποχρεωτικά σε **μη ταξινομημένους πίνακες**. Θεωρείται απλή μέθοδος αλλά λιγότερη αποδοτική (*αρκετά χρονοβόρα για μεγάλους πίνακες*).

2. **Δυαδική αναζήτηση.** Εφαρμόζεται μόνο σε **ταξινομημένους πίνακες**. Θεωρείται πιο πολύπλοκη μέθοδος αλλά εξαιρετικά αποδοτική (*πολύ γρήγορη ακόμα και για μεγάλους πίνακες)*.

Σειριακή αναζήτηση

Να γραφεί πρόγραμμα στη ΓΛΩΣΣΑ που διαβάζει τα επώνυμα, τα ονόματα και τις μηνιαίες πωλήσεις των 50 εργαζομένων μίας εταιρείας σε τρεις πίνακες, αντίστοιχα. Στη συνέχεια, ζητάει από τον χρήστη να δώσει ένα επώνυμο και το πρόγραμμα αναζητά τον αντίστοιχο εργαζόμενο (ή εργαζόμενους αν τυγχάνει να έχουν το ίδιο επώνυμο) και τυπώνει στην οθόνη τα στοιχεία του (επώνυμο, όνομα και πωλήσεις).

Ο πίνακας δεν είναι ταξινομημένος κι επομένως θα εφαρμόσουμε αλγόριθμο σειριακής αναζήτησης. Τα στοιχεία ενός εργαζόμενου αποθηκεύονται ξεχωριστά σε 3 πίνακες: `Επώνυμο`, `Όνομα` και `Πώληση`, στην ίδια φυσικά θέση (π.χ. στη θέση `3` τα στοιχεία είναι τα `Επώνυμο[3]`, `Όνομα[3]` και `Πώληση[3]`).

Στο Κεφάλαιο 3 έχουμε περιγράψει τον αλγόριθμο της σειριακής αναζήτησης, τον οποίο θα χρησιμοποιήσουμε εδώ με δύο διαφοροποιήσεις:

- Ο αριθμός των στοιχείων του πίνακα είναι γνωστός (εδώ 50).
- Αν το στοιχείο προς εύρεση (εδώ το επώνυμο που ψάχνουμε, key) βρεθεί δεν σταματάμε τη σάρωση αλλά εμφανίζουμε τα στοιχεία του στην οθόνη και συνεχίζουμε μέχρι το τέλος του πίνακα (*διότι ενδέχεται να υπάρχει κι άλλος με το ίδιο επώνυμο*).

```
ΠΡΟΓΡΑΜΜΑ Σειριακή_αναζήτηση_στοιχείου_σε_πίνακα
ΜΕΤΑΒΛΗΤΕΣ
      ΠΡΑΓΜΑΤΙΚΕΣ: Πώληση[50]
      ΧΑΡΑΚΤΗΡΕΣ: Επώνυμο[50], Όνομα[50], key
      ΑΚΕΡΑΙΕΣ: i
      ΛΟΓΙΚΕΣ: βρέθηκε
ΑΡΧΗ

      !Είσοδος δεδομένων, εδώ των 50 επωνύμων, ονομάτων και
      !πωλήσεων σε τρεις πίνακες, αντίστοιχα.
      ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 50

            ΓΡΑΨΕ 'Παρακαλώ, δώστε τον ', i, ' εργαζόμενο.'
            ΓΡΑΨΕ 'Το επώνυμο, το όνομα και την πώλησή του:'
            ΔΙΑΒΑΣΕ Επώνυμο[i], Όνομα[i], Πώληση[i]

      ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ


      !Είσοδος του επωνύμου προς εύρεση, key.
      ΓΡΑΨΕ 'Παρακαλώ, δώστε το επώνυμο που ψάχνετε:'
      ΔΙΑΒΑΣΕ key

      !Αρχική τιμή της μεταβλητής-σημαίας βρέθηκε.
      βρέθηκε ← ΨΕΥΔΗΣ

      !Σαρώνει τον πίνακα Επώνυμο για να βρει το key.
      ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 50

        ΑΝ Επώνυμο[i] = key ΤΟΤΕ
           ΓΡΑΨΕ Επώνυμο[i], Όνομα[i], Πώληση[i]
           βρέθηκε ← ΑΛΗΘΗΣ
        ΤΕΛΟΣ_ΑΝ

      ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

      !Αν δεν βρέθηκε τότε εμφάνισε ένα σχετικό μήνυμα στην
      !οθόνη.
      ΑΝ ΟΧΙ βρέθηκε ΤΟΤΕ
```

```
            ΓΡΑΨΕ 'Το επώνυμο δεν βρέθηκε στον πίνακα'
        ΤΕΛΟΣ_ΑΝ

ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Δυαδική αναζήτηση

Όπως προείπαμε, **η δυαδική αναζήτηση εφαρμόζεται μόνο επί ενός ταξινομημένου πίνακα**.

Ο **αλγόριθμος**, γενικά, είναι ο εξής:

1. Βρίσκουμε το μεσαίο στοιχείο του πίνακα.
2. Αν το μεσαίο στοιχείο του πίνακα ισούται με αυτό που ψάχνουμε (key) τότε το βρήκαμε.
3. Αν το μεσαίο στοιχείο του πίνακα είναι μικρότερο από αυτό που ψάχνουμε (key) τότε πρέπει να ψάξουμε στο δεύτερο μισό (άνω μέρος) του πίνακα. Έτσι, η αναζήτηση θα περιοριστεί σε έναν νέο «νοητό πίνακα» που αποτελεί το δεύτερο μισό του αρχικού.
4. Αν το μεσαίο στοιχείο του πίνακα είναι μεγαλύτερο από αυτό που ψάχνουμε (key) τότε πρέπει να ψάξουμε στο πρώτο μισό (κάτω μέρος) του πίνακα. Έτσι, η αναζήτηση θα περιοριστεί σε έναν νέο «νοητό πίνακα» που αποτελεί το πρώτο μισό του αρχικού.
5. Συνέχισε με το βήμα 1 μέχρι να το βρεις ή μέχρι να μην υπάρχει πια άλλος «νοητός πίνακας».

Ακόμα και τεράστιος να είναι ο πίνακας η δυαδική αναζήτηση θα βρει ή όχι ένα στοιχείο σε ελάχιστα βήματα.

Να γραφεί πρόγραμμα στη ΓΛΩΣΣΑ που διαβάζει σε έναν πίνακα 10 ακεραίους αριθμούς σε αύξουσα διάταξη (δηλαδή, η εισαγωγή τους θα γίνεται από τον μικρότερο στον μεγαλύτερο). Στη συνέχεια, ζητάει από τον χρήστη να δώσει έναν ακέραιο και το πρόγραμμα τον αναζητάει μέσα στον ταξινομημένο πίνακα. Αν τον βρει, εμφανίζει τη θέση του και αν όχι εμφανίζει ένα κατάλληλο μήνυμα περί μη εύρεσης του ακεραίου.

Μερικές παρατηρήσεις:

- Το πλήθος των ακεραίων που θα διαβαστεί είναι προκαθορισμένο (εδώ 10), άρα η εισαγωγή τους στον πίνακα μπορεί να γίνει με μία επαναληπτική εντολή **ΓΙΑ..ΑΠΟ..ΜΕΧΡΙ**.

- Όμως, η αναζήτηση μπορεί να περιλαμβάνει μη καθορισμένο αριθμό βημάτων (*μπορεί να βρεθεί στην πρώτη επανάληψη, στην δεύτερη κ.ο.κ ή να μην βρεθεί καθόλου*). Σε αυτήν την περίπτωση θα χρησιμοποιήσουμε μία εντολή **ΌΣΟ..ΕΠΑΝΑΛΑΒΕ.**

```
ΠΡΟΓΡΑΜΜΑ Δυαδική_αναζήτηση_στοιχείου_σε_πίνακα
ΜΕΤΑΒΛΗΤΕΣ
     ΑΚΕΡΑΙΕΣ: Π[10], key, i, j , m
     ΛΟΓΙΚΕΣ: βρέθηκε
ΑΡΧΗ

     !Είσοδος δεδομένων, εδώ των 10 ακεραίων.
     ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 10

          ΓΡΑΨΕ 'Παρακαλώ, δώστε τον ', i, ' ακέραιο.'
          ΔΙΑΒΑΣΕ Π[i]

     ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ


     !Είσοδος του ακεραίου προς εύρεση, key.
     ΓΡΑΨΕ 'Παρακαλώ, δώστε τον ακέραιο που ψάχνετε:'
     ΔΙΑΒΑΣΕ key

     !Αρχική τιμή της μεταβλητής-σημαίας βρέθηκε.
     βρέθηκε ← ΨΕΥΔΗΣ

     !Αρχικές τιμές των δεικτών θέσης i και j.
     i ← 1
     j ← 10

     !Σαρώνει δυαδικά τον πίνακα Π για να βρει το key.
     ΟΣΟ i <= j ΚΑΙ (ΌΧΙ βρέθηκε)

       !Ο δείκτης m είναι στο μέσον του πίνακα.
       m ← (i + j) DIV 2

       !Αν το μεσαίο στοιχείο Π[m] ισούται με το key
       !τότε βρέθηκε.
       ΑΝ Π[m] = key ΤΟΤΕ
          ΓΡΑΨΕ 'Βρέθηκε στη θέση ' , m
          βρέθηκε ← ΑΛΗΘΗΣ

       !Αλλιώς αν το μεσαίο στοιχείο Π[m] είναι
       !μικρότερο από το key τότε πρέπει να ψάξουμε στο δεύτερο
       !μισό του πίνακα. Συνεπώς, ανεβάζουμε το κάτω όριο (i)
       !στη θέση m + 1.
```

```
        ΑΛΛΙΩΣ_ΑΝ  Π[m] < key ΤΟΤΕ
            i ← m + 1

            !Αλλιώς αν το μεσαίο στοιχείο Π[m] είναι
            !μεγαλύτερο από το key τότε πρέπει να ψάξουμε στο
            !πρώτο μισό του πίνακα. Συνεπώς, κατεβάζουμε το πάνω
            !όριο (j)στη θέση m - 1.
        ΑΛΛΙΩΣ
            j ← m - 1

        ΤΕΛΟΣ_ΑΝ

    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

    !Αν δεν βρέθηκε τότε εμφάνισε ένα σχετικό μήνυμα στην
    !οθόνη.
    ΑΝ ΟΧΙ βρέθηκε ΤΟΤΕ
        ΓΡΑΨΕ 'Ο ακέραιος ', key, ' δεν βρέθηκε στον πίνακα'
    ΤΕΛΟΣ_ΑΝ

ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

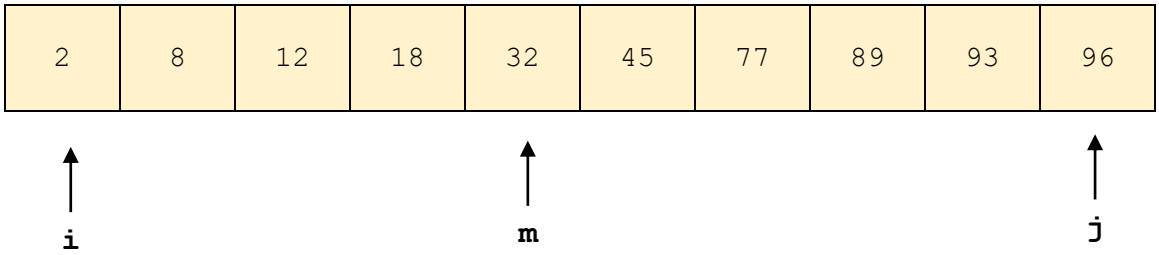
Εκτέλεση με δοκιμαστικά δεδομένα

Αρχικός πίνακας:

Π[10]	2	8	12	18	32	45	77	89	93	96

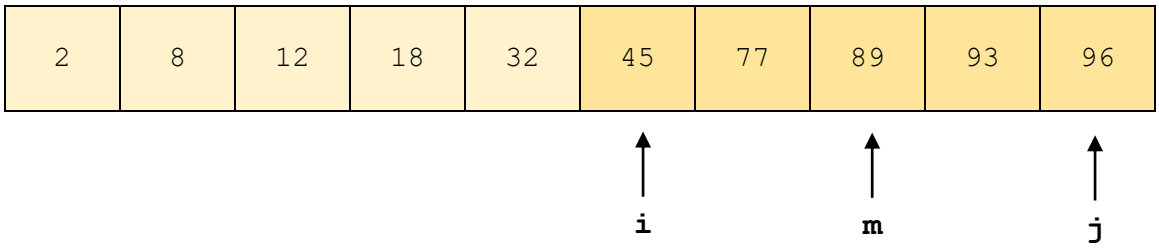
Key ← 93

- **<u>1η επανάληψη</u>:** i ← 1 , j ← 10, m ← (1+10) DIV 2 = 5



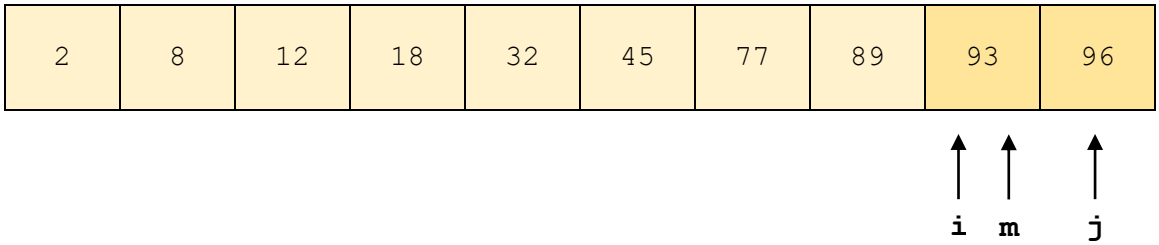
32 < 93, άρα θα συνεχίσουμε στο δεύτερο μισό (πάνω μέρος) του πίνακα
(**i** ← m + 1 = 5 + 1 = 6, **j** ← 10)

- **<u>2η επανάληψη</u>:** `i ← 6 , j ← 10, m ← (6+10) DIV 2 = 8`
  Ο «νοητός πίνακας» στον οποίο θα περιοριστεί η αναζήτηση φαίνεται με έντονο κίτρινο χρώμα.



`89 < 93,` άρα θα συνεχίσουμε στο δεύτερο μισό (πάνω μέρος) του «νοητού πίνακα»
`(`**`i`** `← m + 1 = 8 + 1 = 9,` **`j`** `← 10)`

- **<u>3η επανάληψη</u>:** `i ← 9 , j ← 10, m ← (9+10) DIV 2 = 9`
  Ο «νοητός πίνακας» στον οποίο θα περιοριστεί η αναζήτηση φαίνεται με έντονο κίτρινο χρώμα.



`93 = 93,` βρέθηκε.

Συμπέρασμα: Μέσα σε 3 επαναλήψεις βρήκαμε το ζητούμενο στοιχείο (`Key`). Αν χρησιμοποιούσαμε σειριακή αναζήτηση θα βρίσκαμε το `key` σε 9 επαναλήψεις. Γι' αυτό **η δυαδική αναζήτηση είναι ταχύτατη**.

Δισδιάστατοι πίνακες

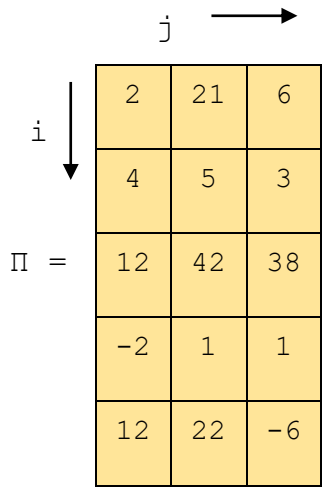
Αν και η αποθήκευση των δεδομένων σε έναν δισδιάστατο πίνακα γίνεται επίσης σε συνεχόμενες θέσεις στη μνήμη, η νοητική αναπαράστασή τους γίνεται με γραμμές και στήλες.[3]

Παρακάτω, βλέπουμε έναν πίνακα ακεραίων Π_{5x3} ή σε σύνταξη της ΓΛΩΣΣΑΣ `Π[5,3]`, όπου η **πρώτη διάσταση** `[5]` αναφέρεται στις **γραμμές** και η **δεύτερη διάσταση** `[3]` στις **στήλες**.

Π =	2	21	6
	4	5	3
	12	42	38
	-2	1	1
	12	22	-6

Για τον προσδιορισμό μίας θέσης του πίνακα χρειάζονται δύο δείκτες: Ο δείκτης γραμμής και ο δείκτης στήλης. Συμβατικά, για τον **δείκτη γραμμής** χρησιμοποιούμε το **όνομα `i`** και για τον **δείκτη στήλης** το **όνομα `j`**.

Έτσι, ο συμβολισμός **`Π[i,j]`** προσδιορίζει ένα **στοιχείο του πίνακα**. Π.χ. η θέση `Π[1,3]` περιέχει τον ακέραιο `6`.



[3] Συνιστάται στον αναγνώστη να έχει διαβάσει πρώτα το Κεφάλαιο 3 «Δομές δεδομένων και αλγόριθμοι».

Συμπερασματικά:

Το **περιεχόμενο μίας θέσης καθορίζεται από την εξής σύνταξη**:

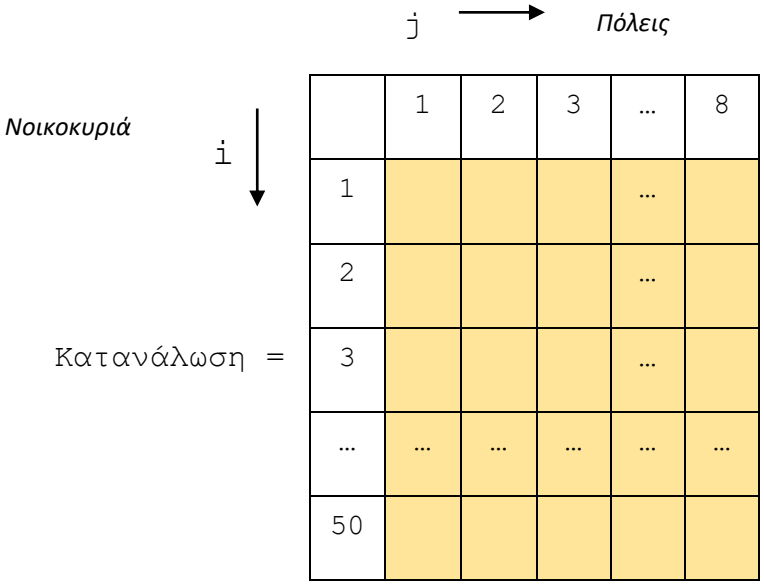
`Π[δείκτης γραμμής, δείκτης στήλης]`

Παραδείγματα επεξεργασίας δισδιάστατου πίνακα

Διάβασμα στοιχείων

Να γραφεί πρόγραμμα στη ΓΛΩΣΣΑ που διαβάζει την μηναία κατανάλωση ρεύματος 50 νοικοκυριών από 8 πόλεις της χώρας. Η κατανάλωση είναι ένας θετικός ακέραιος αριθμός και πρέπει να ελέγχεται αν είναι σωστός κατά την εισαγωγή του στον πίνακα.

Θα χρησιμοποιήσουμε έναν πίνακα με όνομα Κατανάλωση που θα έχει 50 γραμμές (για τα νοικοκυριά) και 8 στήλες (για τις πόλεις). Φυσικά, θα μπορούσε να θεωρηθεί και το αντίστροφο ως νοητική αναπαράσταση.



Στις περισσότερες περιπτώσεις χρησιμοποιούμε μία εξωτερική εντολή **ΓΙΑ..ΑΠΟ..ΜΕΧΡΙ** για τη σάρωση σε γραμμές και μία εσωτερική εντολή **ΓΙΑ..ΑΠΟ..ΜΕΧΡΙ** για τη σάρωση σε στήλες (δηλαδή, τις θέσεις της γραμμής).

Ο παρακάτω κώδικας είναι τυπικός για το διάβασμα στοιχείων δισδιάστατου πίνακα.

```
ΠΡΟΓΡΑΜΜΑ Διάβασμα_καταναλώσεων_ρεύματος_σε_πίνακα
ΜΕΤΑΒΛΗΤΕΣ
     ΑΚΕΡΑΙΕΣ: Κατανάλωση[50, 8] , i, j
ΑΡΧΗ
     !Εξωτερικός βρόχος για τις γραμμές.
     ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 50

        !Εσωτερικός βρόχος για τις στήλες (θέσεις της γραμμής).
        ΓΙΑ j ΑΠΟ 1 ΜΕΧΡΙ 8

            !Τυπική περίπτωση ελέγχου εγκυρότητας.
            ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ

              ΓΡΑΨΕ 'Παρακαλώ, δώστε την κατανάλωση:'
              ΔΙΑΒΑΣΕ Κατανάλωση [i, j]

              ΑΝ Κατανάλωση [i, j] <= 0 ΤΟΤΕ

                ΓΡΑΨΕ 'Η κατανάλωση που δώσατε δεν είναι
                     &  έγκυρη. Παρακαλώ ξαναδώστε'
               ΤΕΛΟΣ_ΑΝ

             ΜΕΧΡΙΣ_ΟΤΟΥ Κατανάλωση [i, j] > 0

          ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

     ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

     !Επιβεβαιωτικό ολοκλήρωσης εισόδου.
     ΓΡΑΨΕ 'Η εισαγωγή των καταναλώσεων ολοκληρώθηκε.'

ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

*(Στο Κεφάλαιο 8, αναφέραμε εκτεταμένα τις σχετικές περιπτώσεις ενδεδειγμένης χρήσης της **ΜΕΧΡΙΣ_ΟΤΟΥ** και καλό είναι ο αναγνώστης να τις έχει σοβαρά υπόψιν του).*

Πίνακας τιμών:

Ένα μικρό δείγμα **δοκιμαστικών δεδομένων** και του διαβάσματος των στοιχείων φαίνεται στον παρακάτω πίνακα τιμών:

i	j	Κατανάλωση [i,j] (εννοεί σε KW)
1	1	450
1	2	680
1	3	780
1	4	1340
1	5	560
1	6	590
1	7	1755
1	8	2400
2	1	1100
2	2	802
2	3	755
2	4	830
2	5	1004
2	6	1200
2	7	3400
2	8	1550
3	1	875
3	2	410
…	…	…
50	7	770
50	8	2100

Υπολογισμός αθροίσματος

Να γραφεί πρόγραμμα στη ΓΛΩΣΣΑ που διαβάζει την μηναία κατανάλωση ρεύματος 50 νοικοκυριών από 8 πόλεις της χώρας. Σκοπός είναι να υπολογίσει το άθροισμα των καταναλώσεων όλων των νοικοκυριών από όλες τις πόλεις. Υποθέτουμε ότι οι ακέραιοι αριθμοί των καταναλώσεων που εισάγονται είναι σωστοί (δεν χρειάζεται έλεγχος εγκυρότητας).

Θα χρειαστούμε μία **μεταβλητή-συσσωρευτή (αθροιστή)**. Συνήθως, δίνουμε το **όνομα Sum**. Για κάθε κατανάλωση που διαβάζουμε την προσθέτουμε στον αθροιστή Sum.

```
ΠΡΟΓΡΑΜΜΑ Σύνολο_κατανάλωσης_ρεύματος
ΜΕΤΑΒΛΗΤΕΣ
     ΑΚΕΡΑΙΕΣ: Κατανάλωση[50, 8] , i, j, Sum
ΑΡΧΗ
     !Δεν ξεχνάμε την αρχικοποίηση του αθροιστή.
     Sum ← 0
```

```
!Είσοδος δεδομένων.
!Εξωτερικός βρόχος για τις γραμμές.
ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 50

   !Εσωτερικός βρόχος για τις στήλες (θέσεις της γραμμής).
   ΓΙΑ j ΑΠΟ 1 ΜΕΧΡΙ 8

         ΓΡΑΨΕ 'Παρακαλώ, δώστε την κατανάλωση:'
         ΔΙΑΒΑΣΕ Κατανάλωση [i, j]

     ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

!Προσπελαύνει μία-μία κατανάλωση από τον πίνακα και την
!προσθέτει στον αθροιστή.
ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 50

   ΓΙΑ j ΑΠΟ 1 ΜΕΧΡΙ 8

      Sum ← Sum + Κατανάλωση[i,j]

   ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

!Εμφάνιση του αποτελέσματος στην οθόνη.
ΓΡΑΨΕ 'Η συνολική κατανάλωση είναι: ', Sum

ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Φυσικά, θα μπορούσαμε κατά το διάβασμα των καταναλώσεων να τις προσθέταμε αμέσως στον αθροιστή (τον πρώτο διπλό βρόχο) αλλά εδώ ο σκοπός είναι να δείξουμε την επεξεργασία (άθροισμα) που πραγματοποιείται *όταν τα δεδομένα είναι ήδη στον πίνακα*.

Υπολογισμός μέσου όρου

Η εύρεση του μέσου όρου προϋποθέτει ότι έχουμε ήδη βρει το άθροισμα των δεδομένων στον πίνακα (ο αριθμός των δεδομένων είναι ο αριθμός των θέσεων του πίνακα).

Σε συνέχεια του προηγούμενου παραδείγματος (του υπολογισμού αθροίσματος), η εταιρεία επιθυμεί, επίσης, να μάθει και τον μέσο όρο κατανάλωσης από όλα τα νοικοκυριά όλων των πόλεων.

Ο κώδικας δεν διαφέρει και πολύ από τον προηγούμενο. Απλά, προσθέτουμε κώδικα για τον υπολογισμό του μέσου όρου.

```
ΠΡΟΓΡΑΜΜΑ Μέσος_όρος_κατανάλωσης_ρεύματος
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: Κατανάλωση[50, 8] , i, j, Sum
  ΠΡΑΓΜΑΤΙΚΕΣ: ΜΟ
ΑΡΧΗ
     !Δεν ξεχνάμε την αρχικοποίηση του αθροιστή.
     Sum ← 0

  !Είσοδος δεδομένων.
  !Εξωτερικός βρόχος για τις γραμμές.
     ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 50

        !Εσωτερικός βρόχος για τις στήλες (θέσεις της γραμμής).
        ΓΙΑ j ΑΠΟ 1 ΜΕΧΡΙ 8

              ΓΡΑΨΕ 'Παρακαλώ, δώστε την κατανάλωση:'
              ΔΙΑΒΑΣΕ Κατανάλωση [i, j]

          ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

     ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

     !Προσπελαύνει μία-μία κατανάλωση από τον πίνακα και την
     !προσθέτει στον αθροιστή.
     ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 50

        ΓΙΑ j ΑΠΟ 1 ΜΕΧΡΙ 8

           Sum ← Sum + Κατανάλωση[i,j]

        ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

     ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

     !Υπολογισμός του μέσου όρου.
     ΜΟ ← Sum / 50

    !Εμφάνιση του αποτελέσματος στην οθόνη.
     ΓΡΑΨΕ 'Ο μέσος όρος κατανάλωσης είναι: ', ΜΟ
```

```
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Καταμέτρηση στοιχείων

Μία τυπική επεξεργασία σε έναν πίνακα είναι να μετρήσουμε το πλήθος κάποιων στοιχείων που πληρούν κάποιο/α κριτήριο/α.

Να γραφεί πρόγραμμα στη ΓΛΩΣΣΑ που διαβάζει την μηναία κατανάλωση ρεύματος 50 νοικοκυριών από 8 πόλεις της χώρας. Σκοπός είναι να υπολογίσει πόσοι είναι οι «μεγάλοι» καταναλωτές, δηλαδή πόσοι έχουν κατανάλωση ρεύματος πάνω από ή ίσο με 2000 KW.

Εδώ, σε κάθε επανάληψη πρέπει να ελέγχουμε αν η τρέχουσα κατανάλωση (της `[i,j]` θέσης του πίνακα) ικανοποιεί το κριτήριο, δηλαδή να είναι ίση ή πάνω από 2000. Αν το ικανοποιεί τότε αυξάνουμε την τιμή μίας **μεταβλητής-μετρητή**.

```
ΠΡΟΓΡΑΜΜΑ Πλήθος_μεγάλων_καταναλωτών_ρεύματος
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: Κατανάλωση[50, 8] , i, j, μετρητής
ΑΡΧΗ
     !Δεν ξεχνάμε την αρχικοποίηση του μετρητή.
     μετρητής ← 0

     !Είσοδος δεδομένων.
     !Εξωτερικός βρόχος για τις γραμμές.
     ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 50

        !Εσωτερικός βρόχος για τις στήλες (θέσεις της γραμμής).
        ΓΙΑ j ΑΠΟ 1 ΜΕΧΡΙ 8

              ΓΡΑΨΕ 'Παρακαλώ, δώστε την κατανάλωση:'
              ΔΙΑΒΑΣΕ Κατανάλωση [i, j]

        ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

     ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

     !Προσπελαύνει μία-μία κατανάλωση από τον πίνακα και ελέγχει
     !αν ικανοποιεί το κριτήριο.
     ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 50
```

```
        ΓΙΑ j ΑΠΟ 1 ΜΕΧΡΙ 8

           ΑΝ Κατανάλωση[i,j] >= 2000 ΤΟΤΕ
              μετρητής ← μετρητής + 1
           ΤΕΛΟΣ_ΑΝ    ]

        ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

     ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

     !Εμφάνιση του αποτελέσματος στην οθόνη.
     ΓΡΑΨΕ  'Το πλήθος των υψηλών καταναλώσεων είναι: ',
                                              & μετρητής

ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Εύρεση του μέγιστου στοιχείου

Να γραφεί πρόγραμμα στη ΓΛΩΣΣΑ που διαβάζει την μηναία κατανάλωση ρεύματος και το ονοματεπώνυμο ιδιοκτησίας 50 νοικοκυριών από 8 πόλεις της χώρας. Επίσης, διαβάζει και τα ονόματα των 8 πόλεων. Σκοπός είναι να υπολογίσει την μεγαλύτερη κατανάλωση, ποιό νοικοκυριό την έκανε και σε ποιά πόλη και όλες αυτές τις πληροφορίες να τις εμφανίσει στην οθόνη.

Θα χρειαστούμε **3 πίνακες**:

- Έναν δισδιάστατο, με όνομα `Κατανάλωση[50,8]`, που θα περιέχει τις καταναλώσεις ρεύματος.
- Έναν δισδιάστατο, με όνομα `Ιδιοκτήτης[50,8]`, που θα περιέχει τα ονοματεπώνυμα των ιδιοκτητών των αντίστοιχων νοικοκυριών.
- Έναν μονοδιάστατο, με όνομα `Πόλη[8]`, που θα περιέχει τα ονόματα των αντίστοιχων πόλεων.

Ο αλγόριθμος για την εύρεση του μεγίστου στοιχείου σε έναν δισδιάστατο πίνακα έχει εξεταστεί στο Κεφάλαιο 3. Φυσικά, θα χρειαστούμε και κάποιες μεταβλητές που θα κρατούν την μέγιστη κατανάλωση (`Max_Κατανάλωση`), τον ιδιοκτήτη του αντίστοιχου νοικοκυριού (`Max_Ιδιοκτήτης`) και την πόλη που ανήκει (`Max_Πόλη`). Σε αυτές τις μεταβλητές βάζουμε το πρόθεμα `Max` για να ξεχωρίζουν.

```
ΠΡΟΓΡΑΜΜΑ Εύρεση_μέγιστης_κατανάλωσης
ΜΕΤΑΒΛΗΤΕΣ
```

```
  ΑΚΕΡΑΙΕΣ: Κατανάλωση[50,8] , i, j, Max_Κατανάλωση
  ΧΑΡΑΚΤΗΡΕΣ: Ιδιοκτήτης[50,8], Πόλη[8]
  ΧΑΡΑΚΤΗΡΕΣ: Max_Ιδιοκτήτης, Max_Πόλη
ΑΡΧΗ
     !Είσοδος δεδομένων.
     !Βρόχος για το διάβασμα των 8 πόλεων.
     ΓΙΑ j ΑΠΟ 1 ΜΕΧΡΙ 8

          ΓΡΑΨΕ 'Παρακαλώ, δώστε την ',j,' πόλη:'
          ΔΙΑΒΑΣΕ Πόλη[j]

     ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

     !Διάβασμα των καταναλώσεων και των ιδιοκτητών.
     !Εξωτερικός βρόχος για τις γραμμές.
     ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 50

        !Εσωτερικός βρόχος για τις στήλες (θέσεις της γραμμής).
        ΓΙΑ j ΑΠΟ 1 ΜΕΧΡΙ 8

             ΓΡΑΨΕ 'Παρακαλώ, δώστε την κατανάλωση:'
             ΔΙΑΒΑΣΕ Κατανάλωση [i, j]
             ΓΡΑΨΕ 'Παρακαλώ, δώστε τον ιδιοκτήτη:'
             ΔΙΑΒΑΣΕ Ιδιοκτήτης [i, j]

          ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

     ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

     !Θέτουμε στις μεταβλητές Max το πρώτο στοιχείο από κάθε
     !πίνακα.
     Max_Κατανάλωση ← Κατανάλωση[1,1]
     Max_Ιδιοκτήτης ← Ιδιοκτήτης[1,1]
     Max_Πόλη ← Πόλη[1]


     !Προσπελαύνει μία-μία κατανάλωση από τον πίνακα Κατανάλωση
     !και ελέγχει αν η τρέχουσα κατανάλωση είναι μεγαλύτερη της
     !Max.
     ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 50

        ΓΙΑ j ΑΠΟ 1 ΜΕΧΡΙ 8

           ΑΝ Κατανάλωση[i,j] > Max_Κατανάλωση ΤΟΤΕ
              Max_Κατανάλωση ← Κατανάλωση[i,j]
              Max_Ιδιοκτήτης ← Ιδιοκτήτης[i,j]
```

```
            Max_Πόλη ← Πόλη[j]
        ΤΕΛΟΣ_ΑΝ

    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

!Εμφάνιση του αποτελέσματος στην οθόνη.
ΓΡΑΨΕ 'Η μέγιστη κατανάλωση είναι: ', Max_Κατανάλωση
ΓΡΑΨΕ 'Ο ιδιοκτήτης είναι ο/η: ', Max_Ιδιοκτήτης
ΓΡΑΨΕ 'Και βρέθηκε στην πόλη: ', Max_Πόλη

ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Εύρεση του ελάχιστου στοιχείου

Παρόμοιος είναι ο τρόπος εύρεσης του ελάχιστου στοιχείου. Ο αναγνώστης μπορεί να βασιστεί στο προηγούμενο παράδειγμα για να κατασκευάσει πρόγραμμα που θα υπολογίζει την μικρότερη κατανάλωση, ποιός ιδιοκτήτης την έκανε και σε ποιά πόλη.

Αναζήτηση στοιχείου

Η αναζήτηση στοιχείου σε δισδιάστατο πίνακα δεν διαφέρει σημαντικά από εκείνον σε μονοδιάστατο πίνακα. Κι εδώ **εξαρτάται αν ο πίνακας είναι ταξινομημένος ή όχι**. Στην περίπτωση ταξινομημένου πίνακα προφανώς συνιστάται η χρήση της δυαδικής μεθόδου κι όχι της σειριακής[4].

Στο επόμενο παράδειγμα θα ασχοληθούμε μόνο με τη **σειριακή μέθοδο**.

Να γραφεί πρόγραμμα στη ΓΛΩΣΣΑ που διαβάζει τα 50 ονοματεπώνυμα των καταναλωτών ρεύματος από 8 πόλεις. Τα ονόματα των πόλεων διαβάζονται ξεχωριστά σε έναν μονοδιάστατο πίνακα. Θεωρούμε ότι κανείς καταναλωτής δεν έχει το ίδιο ονοματεπώνυμο με άλλον σε οποιαδήποτε πόλη. Στη συνέχεια, ζητάει από τον χρήστη να δώσει ένα

[4] Στο παράρτημα δίνουμε τον αλγόριθμο της δυαδικής αναζήτησης σε δισδιάστατο πίνακα.

ονοματεπώνυμο και το πρόγραμμα αναζητά τον αντίστοιχο καταναλωτή και τυπώνει στην οθόνη την πόλη που βρίσκεται.

Θα χρειαστούμε **2 πίνακες**:

- Έναν δισδιάστατο, με όνομα `Ονοματεπώνυμο[50,8]`, που θα περιέχει τα ονοματεπώνυμα των καταναλωτών ρεύματος.
- Έναν μονοδιάστατο, με όνομα `Πόλη[8]`, που θα περιέχει τα ονόματα των αντίστοιχων πόλεων.

Η αναζήτηση θα σταματάει όταν βρεθεί ο συγκεκριμένος καταναλωτής. Αυτό μπορεί να γίνει στο πρώτο στοιχείο του πίνακα, ενδιάμεσα ή στο τελευταίο στοιχείο του πίνακα. Συνεπώς, η χρήση του διπλού βρόχου **ΓΙΑ..ΑΠΟ..ΜΕΧΡΙ** είναι απαγορευτική και η χρήση του διπλού βρόχου **ΟΣΟ..ΕΠΑΝΑΛΑΒΕ** είναι μονόδρομος. Επίσης, θα χρειαστούμε μία λογική μεταβλητή-σημαία με όνομα `βρέθηκε`. Αν το στοιχείο βρεθεί, η τιμή της `βρέθηκε` γίνεται `ΑΛΗΘΗΣ` και ο διπλός βρόχος σταματάει αμέσως[5].

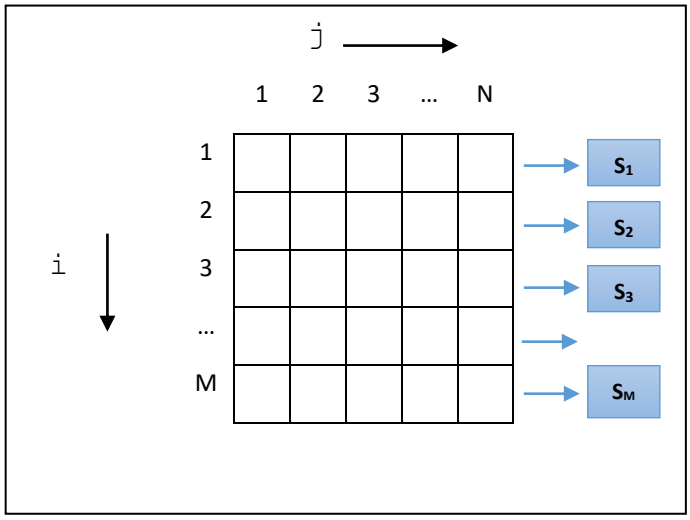
```
ΠΡΟΓΡΑΜΜΑ Σειριακή_αναζήτηση_καταναλωτή
ΜΕΤΑΒΛΗΤΕΣ
   ΑΚΕΡΑΙΕΣ: i, j
   ΧΑΡΑΚΤΗΡΕΣ: Ονοματεπώνυμο[50, 8], Πόλη[8], key
   ΛΟΓΙΚΕΣ: βρέθηκε
ΑΡΧΗ
      !Είσοδος δεδομένων.
      !Βρόχος για το διάβασμα των 8 πόλεων.
      ΓΙΑ j ΑΠΟ 1 ΜΕΧΡΙ 8

            ΓΡΑΨΕ 'Παρακαλώ, δώστε την ',j,' πόλη:'
            ΔΙΑΒΑΣΕ Πόλη[j]

      ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

      !Διάβασμα των ονοματεπωνύμων.
      !Εξωτερικός βρόχος για τις γραμμές.
      ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 50

         !Εσωτερικός βρόχος για τις στήλες (θέσεις της γραμμής).
         ΓΙΑ j ΑΠΟ 1 ΜΕΧΡΙ 8

               ΓΡΑΨΕ 'Παρακαλώ, δώστε το ονοματεπώνυμο:'
               ΔΙΑΒΑΣΕ Ονοματεπώνυμο [i, j]

           ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

[5] Σε γενικές γραμμές πρόκειται για υλοποίηση του αλγόριθμου σειριακής αναζήτησης μονοδιάστατου πίνακα, που εξετάσαμε στο Κεφάλαιο 3, στις δύο διαστάσεις.

```
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

    !Είσοδος του ονοματεπωνύμου προς εύρεση, key.
    ΓΡΑΨΕ 'Παρακαλώ, δώστε το ονοματεπώνυμο που ψάχνετε:'
    ΔΙΑΒΑΣΕ key

    !Αρχικές τιμές μεταβλητών πριν ξεκινήσει η αναζήτηση.
    i ← 1
    βρέθηκε ← ΨΕΥΔΗΣ

    !Εκινάει η αναζήτηση.
    !Σάρωση των γραμμών του πίνακα.
    ΟΣΟ i <= 50 ΚΑΙ (ΟΧΙ βρέθηκε) ΕΠΑΝΑΛΑΒΕ

       !Αρχικοποίηση του δείκτη j σε κάθε νέα γραμμή.
       j ← 1

       !Σάρωση των θέσεων (στηλών) της γραμμής.
       ΟΣΟ j <= 8 ΚΑΙ (ΟΧΙ βρέθηκε) ΕΠΑΝΑΛΑΒΕ

          ΑΝ Ονοματεπώνυμο[i,j] = key ΤΟΤΕ
              βρέθηκε ← ΑΛΗΘΗΣ
              ΓΡΑΨΕ 'Βρέθηκε στην πόλη ', Πόλη[j]
          ΑΛΛΙΩΣ
              j ← j + 1
          ΤΕΛΟΣ_ΑΝ

       ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

       !Αφού ολοκληρωθεί η σάρωση των θέσεων της γραμμής, πάμε
       !στην επόμενη γραμμή.
       i ← i + 1

    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

    !Εμφάνιση μηνύματος αν δεν βρέθηκε.
    ΑΝ (ΟΧΙ βρέθηκε) ΤΟΤΕ
       ΓΡΑΨΕ 'Το ονοματεπώνυμο ', key, ' δεν βρέθηκε σε καμία
                                                & πόλη.'
    ΤΕΛΟΣ_ΑΝ

ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Υπολογισμός αθροίσματος ανά γραμμή

Αρκετές φορές χρειάζεται να γίνει επεξεργασία των δεδομένων ανά γραμμή (ή στήλη). Στο παρακάτω **σχήμα** φαίνεται παραστατικά το **άθροισμα ανά γραμμή**:



Την αλγοριθμική λογική αυτού του είδους επεξεργασίας σε δισδιάστατο πίνακα την έχουμε εξετάσει στο Κεφάλαιο 3 «Δομές δεδομένων και αλγόριθμοι». Σαρώνουμε τις θέσεις της κάθε γραμμής με τον δείκτη `j` και οριζόντια βγαίνει ένα άθροισμα (`S1, S2, S3` κλπ).

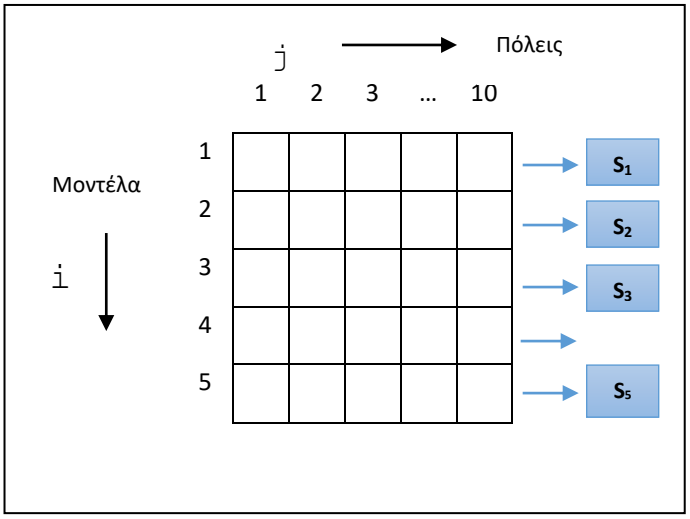
Μία εταιρεία πώλησης αυτοκινήτων πουλάει 5 μοντέλα του αυτοκινήτου της *Express Turbo 2.0*, μέσω των αντιπροσωπειών που διατηρεί σε 10 πόλεις της Ελλάδας. Να γραφεί πρόγραμμα στη `ΓΛΩΣΣΑ` που:

a) Διαβάζει τα ονόματα των μοντέλων και πόλεων σε αντίστοιχους πίνακες.

b) Διαβάζει την αξία των ετήσιων πωλήσεων όλων των μοντέλων από όλες τις πόλεις.

c) Υπολογίζει κι εμφανίζει το σύνολο των πωλήσεων κάθε μοντέλου. Το μήνυμα για κάθε μοντέλο θα είναι «Το σύνολο πωλήσεων του μοντέλου x είναι y, για το τρέχον έτος», όπου x είναι το όνομα του μοντέλου και y το σύνολο των πωλήσεων που υπολογίστηκε.

Στα προβλήματα με δισδιάστατους πίνακες είναι πολύ σημαντική η οπτική αναπάραστασή του δισδιάστατου πίνακα, ώστε να διευκολυνθούμε στην επίλυση. Δηλαδή, ποιά κατηγορία θα αποτελεί γραμμή και ποιά θα αποτελεί τη στήλη. Στην περίπτωσή μας, επειδή θέλουμε άθροισμα ανά μοντέλο αυτοκινήτου, η κατηγορία «μοντέλα» θα είναι οι γραμμές και η κατηγορία «πόλεις» οι στήλες, όπως φαίνεται στο παρακάτω σχήμα:



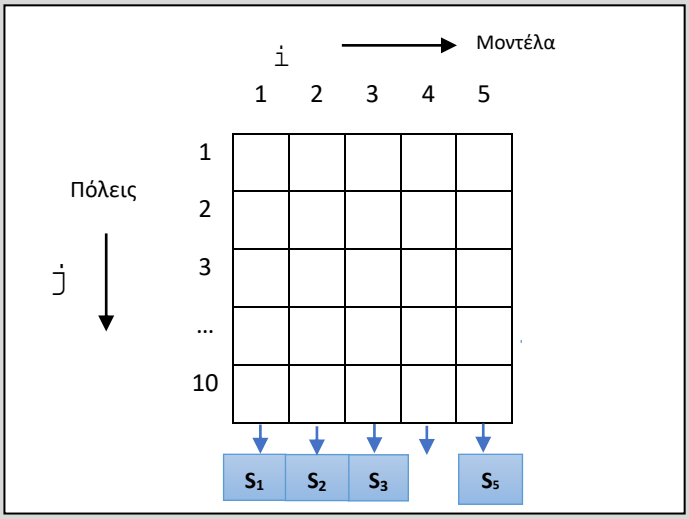
Θα ονομάσουμε τον πίνακα αυτόν «Πώληση».

Επίσης, θα χρειαστούμε για την αποθήκευση των ονομάτων των μοντέλων έναν μονοδιάστατο πίνακα 5 θέσεων. Θα τον ονομάσουμε «Μοντέλο». Παρόμοια και για τα ονόματα των πόλεων τον μονοδιάστατο πίνακα 10 θέσεων, με όνομα «Πόλη».

```
ΠΡΟΓΡΑΜΜΑ Σύνολο_πωλήσεων_ανά_μοντέλο_αυτοκινήτου
ΜΕΤΑΒΛΗΤΕΣ
   ΠΡΑΓΜΑΤΙΚΕΣ: Πώληση[5, 10], Sum
   ΧΑΡΑΚΤΗΡΕΣ: Μοντέλο[5], Πόλη[10]
   ΑΚΕΡΑΙΕΣ: i, j
ΑΡΧΗ

   !Είσοδος δεδομένων.
   !Αρχικά των ονομάτων των μοντέλων και πόλεων.
   ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 5

       ΓΡΑΨΕ 'Παρακαλώ, δώστε το ', i, 'ο μοντέλο:'
       ΔΙΑΒΑΣΕ Μοντέλο[i]

   ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

   ΓΙΑ j ΑΠΟ 1 ΜΕΧΡΙ 10

       ΓΡΑΨΕ 'Παρακαλώ, δώστε την ', j, 'η πόλη:'
       ΔΙΑΒΑΣΕ Πόλη[j]

   ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

   !Τώρα, το διάβασμα των πωλήσεων στον δισδιάστατο πίνακα.
   !Εξωτερικός βρόχος για τις γραμμές-μοντέλα.
   ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 5
```

```
        !Εσωτερικός βρόχος για τις στήλες-πόλεις.
        ΓΙΑ j ΑΠΟ 1 ΜΕΧΡΙ 10

            ΓΡΑΨΕ 'Παρακαλώ, δώστε την πώληση:'
            ΔΙΑΒΑΣΕ Πώληση[i, j]

        ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

    !Προσπελαύνει μία-μία πώληση από τον πίνακα και υπολογίζει
    !για κάθε γραμμή το άθροισμα.
    ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 5

        !Αρχικοποίηση του αθροίσματος σε κάθε νέα γραμμή.
        Sum ← 0

        ΓΙΑ j ΑΠΟ 1 ΜΕΧΡΙ 10

            Sum ← Sum + Πώληση[i,j]

        ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

        !Τελείωσε η γραμμή. Εμφάνισε το τρέχον άθροισμά της.
        ΓΡΑΨΕ 'Το σύνολο πωλήσεων του μοντέλου  ', Μοντέλο[i],
                        &  ' είναι ', Sum, ' για το τρέχον έτος'

    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Αν η οπτική αναπαράσταση ήταν αντίστροφη, δηλαδή η κατηγορία «πόλεις» ήταν σε γραμμή και η κατηγορία «μοντέλο» σε στήλη, τότε θα κάναμε υπολογισμό ανά στήλη.



Αυτό που θα άλλαζε στο πρόγραμμα θα ήταν:
α) Η δήλωση του πίνακα: ΠΡΑΓΜΑΤΙΚΕΣ: Πώληση[10,5]
β) Ο δείκτης i διατρέχει τις στήλες-μοντέλα και ο δείκτης j τις γραμμές-πόλεις.
γ) Η αναφορά στον πίνακα μέσα στο βρόχο είναι Πώληση[j,i].

```
!Εξωτερικός βρόχος για τις στήλες-μοντέλα.
 ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 5

     !Αρχικοποίηση του αθροίσματος σε κάθε νέα
     !στήλη.
     Sum ← 0

     !Εσωτερικός βρόχος για τις γραμμές-μοντέλα.
     ΓΙΑ j ΑΠΟ 1 ΜΕΧΡΙ 10

          Sum ← Sum + Πώληση[j,i]

     ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

     !Τελείωσε η στήλη. Εμφάνισε το τρέχον άθροισμά
     !της.
     ΓΡΑΨΕ 'Το σύνολο πωλήσεων του μοντέλου  ',
          & Μοντέλο[i], ' είναι ', Sum

ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

Συμπερασματικά: Πριν ξεκινήσουμε το γράψιμο του προγράμματος πρέπει να αποφασίσουμε την οπτική αναπαράσταση του δισδιάστατου πίνακα, την οποία την σχεδιάζουμε πρόχειρα στο χαρτί και αποφασίζουμε ποιά κατηγορία δεδομένων θα είναι γραμμή και ποιά στήλη.

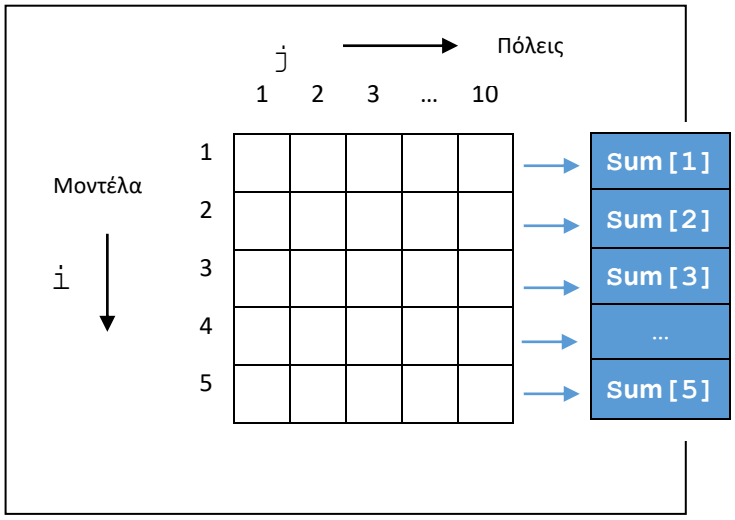
Στις παρούσες σημειώσεις, όταν το ζητούμενο είναι ένα άθροισμα, μέσος όρος κλπ.

	ανά κάποια *κατηγορία* (όπως στο παράδειγμα παραπάνω, πωλήσεις ανά *μοντέλο*) τότε κάνουμε την κατηγορία αυτή γραμμή κι έτσι υπολογίζουμε ανά γραμμή.

Υπολογισμός αθροίσματος ανά γραμμή και αποθήκευσή του σε ξεχωριστό πίνακα

Σε συνέχεια με το προηγούμενο παράδειγμα, η εταιρεία πώλησης αυτοκινήτων επιθυμεί να αποθηκεύσει το σύνολο των πωλήσεων κάθε μοντέλου κι όχι απλά να το εμφανίσει στην οθόνη. Στη συνέχεια, πρέπει το πρόγραμμα να υπολογίσει ποιό μοντέλο έκανε τις υψηλότερες ετήσιες πωλήσεις.

Εδώ, θα χρειαστούμε κι έναν **μονοδιάστατο πίνακα**, με όνομα, **Sum** 5 θέσεων (όσα και τα μοντέλα αυτοκινήτων) που θα περιέχει τα **αθροίσματα των γραμμών**, όπως φαίνεται στο σχήμα:



Αφού καταχωρηθούν τα αθροίσματα των γραμμών στον πίνακα `Sum`, στη συνέχεια είναι εύκολο να γράψουμε εκείνο το τμήμα του προγράμματος που βρίσκει το μέγιστο στοιχείο του `Sum` *(υλοποιούμε τον αλγόριθμο εύρεσης μεγίστου).*

Στο παρακάτω πρόγραμμα θα παραλείψουμε το τμήμα εκείνο που αφορά την είσοδο των δεδομένων στους πίνακες και είναι το ίδιο όπως και στο προηγούμενο παράδειγμα (θα γράψουμε μόνο τα σχόλια). Θα εστιάσουμε στην επεξεργασία των δεδομένων ώστε να βγουν τα οριζόντια αθροίσματα για αποθήκευση στον μονοδιάστατο πίνακα `Sum` και κατόπιν την εύρεση της υψηλότερης τιμής.

```
ΠΡΟΓΡΑΜΜΑ Σύνολο_πωλήσεων_ανά_μοντέλο_αυτοκινήτου_v2
ΜΕΤΑΒΛΗΤΕΣ
   ΠΡΑΓΜΑΤΙΚΕΣ: Πώληση[5, 10], Sum[5], Max_Sum
   ΧΑΡΑΚΤΗΡΕΣ: Μοντέλο[5], Πόλη[10], Max_Μοντέλο
   ΑΚΕΡΑΙΕΣ: i, j
ΑΡΧΗ

   !Είσοδος δεδομένων.
   !Αρχικά των ονομάτων των μοντέλων και πόλεων.

   ...

   !Τώρα, το διάβασμα των πωλήσεων στον δισδιάστατο πίνακα.

    ...

   !Προσπελαύνει μία-μία πώληση από τον πίνακα και υπολογίζει
   !για κάθε γραμμή το άθροισμα.
    ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 5

       !Αρχικοποίηση του αθροίσματος σε κάθε νέα γραμμή.
       Sum[i] ← 0

       ΓΙΑ j ΑΠΟ 1 ΜΕΧΡΙ 10

          Sum[i] ← Sum[i] + Πώληση[i,j]

       ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

     ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

     !Τώρα, υλοποιούμε τον αλγόριθμο εύρεσης μεγίστου σε
     !μονοδιάστατο πίνακα ώστε να βρούμε ποιό μοντέλο έκανε
     !την υψηλότερη πώληση.
     Max_Sum = Sum[1]
     Max_Μοντέλο = Μοντέλο[1]

     ΓΙΑ i ΑΠΟ 2 ΜΕΧΡΙ 5

        ΑΝ Sum[i] > Max_Sum ΤΟΤΕ

           Max_Sum ← Sum[i]
           Max_Μοντέλο ← Μοντέλο[i]


        ΤΕΛΟΣ_ΑΝ

     ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

```
    !Έξοδος αποτελέσματος στην οθόνη.
    ΓΡΑΨΕ 'Το μοντέλο ' , Max_Μοντέλο, ' έκανε τις υψηλότερες
          & πωλήσεις ', Max_Sum,' για το τρέχον έτος'

ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Εκτός από άθροισμα ανά γραμμή, η επεξεργασία μπορεί να είναι η εύρεση του μέσου όρου ανά γραμμή, η υψηλότερη/χαμηλότερη τιμή ανά γραμμή κλπ. Η λογική παραμένει η ίδια.

Τυπικές επεξεργασίες πινάκων

Στο Κεφάλαιο 3, «Δομές δεδομένων και Αλγόριθμοι» αναφέρουμε εκτενώς τις συνήθεις (τυπικές) επεξεργασίες σε έναν πίνακα με παραδείγματα σε ψευδογλώσσα. Μεταξύ άλλων, στο παράρτημα του Κεφαλαίου 3, υπάρχει κι ένας σχετικός αλγόριθμος για τη συγχώνευση δύο μονοδιάστατων πινάκων (*ο αναγνώστης, μπορεί να υλοποιήσει εκείνον τον αλγόριθμο στη ΓΛΩΣΣΑ*).

Ερωτήσεις κατανόησης

1. Τί είναι ένας μονοδιάστατος πίνακας;
2. Σε τί χρησιμεύει γενικά η δομή ενός πίνακα;
3. Τί είναι ο δείκτης ενός πίνακα;
4. Στον συμβολισμό `Π[i]` ποιό είναι το όνομα του πίνακα και ποιός ο δείκτης θέσης;
5. Ο συμβολισμός `Π[5]` σε τί αναφέρεται;
6. Ο φίλος σας ισχυρίζεται ότι η δομή του πίνακα είναι στατική, δηλαδή το μέγεθός του προσδιορίζεται στο τμήμα δηλώσεων μεταβλητών και δεν αλλάζει κατά τη διάρκεια που εκτελείται το πρόγραμμα. Έχει δίκιο;

7. Ο φίλος σας, πάλι, ισχυρίζεται ότι τα στοιχεία του πίνακα αποθηκεύονται σε συνεχόμενες θέσεις στη μνήμη RAM. Έχει δίκιο;

8. Ένας πίνακας μπορεί να περιέχει στοιχεία διαφορετικού τύπου (π.χ. μίξη ακεραίων-πραγματικών αριθμών) ή όλα τα στοιχεία πρέπει να είναι του ιδίου τύπου (π.χ. όλοι ακέραιοι);

9. Ποιοί είναι οι πιο γνωστοί αλγόριθμοι αναζήτησης σε πίνακα και σε ποιές περιπτώσεις χρησιμοποιείται ο καθένας;

10. Ποιές είναι οι τυπικές επεξεργασίες σε έναν πίνακα;

11. Στις περισσότερες περιπτώσεις, ποιά είναι η καταλληλότερη εντολή επανάληψης για την προσπέλαση/επεξεργασία των στοιχείων ενός πίνακα;

12. Στα μαθηματικά, ένας **ταυτοτικός πίνακας NxN** είναι εκείνος που όλα τα **στοιχεία της κυρίας διαγωνίου είναι ίσα με 1** και **όλα τα υπόλοιπα είναι ίσα με 0**, όπως φαίνεται στο παρακάτω σχήμα:

Π_{5x5} =

1	0	0	0	0
0	**1**	0	0	0
0	0	**1**	0	0
0	0	0	**1**	0
0	0	0	0	**1**

Ο πίνακας έχει δηλωθεί ως:

```
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: Π[5,5]
```

και υποθέτουμε ότι έχουμε διαβάσει όλα τα στοιχεία του.

Γράψτε ένα τμήμα προγράμματος που εξετάζει αν ο πίνακας που διαβάστηκε είναι ταυτοτικός ή όχι.

Παράρτημα

Αντιγραφή στοιχείων από δισδιάστατο πίνακα σε μονοδιάστατο

Δίνεται ένας πίνακας A[15,10] που περιέχει ακεραίους και θέλουμε να αντιγράψουμε όλα τα στοιχεία του τα οποία διαιρούνται ακριβώς με το 3 σε έναν νέο μονοδιάστατο πίνακα B[150]. Γράψτε ένα πρόγραμμα στη ΓΛΩΣΣΑ που

α) Διαβάζει τους ακεραίους στον πίνακα A

β) Κατόπιν κάνει την επεξεργασία της αντιγραφής των κατάλληλων στοιχείων στον πίνακα B.

γ) Επίσης, το πρόγραμμα να μας πληροφορήσει για το πόσα στοιχεία αντιγράφηκαν.

Θεωρητικά, όλα τα στοιχεία του πίνακα A είναι δυνατόν να διαιρούνται ακριβώς με το 3, άρα το πλήθος των θέσεων του B θα είναι 15 x 10 = 150.

Μετά το τυπικό διάβασμα των στοιχείων του δισδιάστατου πίνακα A, ξεκινάει ένας διπλός βρόχος που προσπελαύνει ένα-ένα στοιχείο του A και το ελέγχει αν ικανοποιεί το κριτήριο της αντιγραφής (δηλαδή, αν διαιρείται ακριβώς με το 3). Αν ναι, τότε το αντιγράφει στην κατάλληλη θέση του B.

```
ΠΡΟΓΡΑΜΜΑ Αντιγραφή_στοιχείων_από_δισδιάτατο_σε_μονοδιάστατο
ΜΕΤΑΒΛΗΤΕΣ
     ΑΚΕΡΑΙΕΣ: A[15, 10] , B[150], i, j, k
ΑΡΧΗ
     !Είσοδος δεδομένων (απάντηση στο α).
     !Εξωτερικός βρόχος για τις γραμμές.
     ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 15

        !Εσωτερικός βρόχος για τις στήλες (θέσεις της γραμμής).
        ΓΙΑ j ΑΠΟ 1 ΜΕΧΡΙ 10

          ΓΡΑΨΕ 'Παρακαλώ, δώστε τον ',(i-1)*10+j, 'ο ακέραιο:'
          ΔΙΑΒΑΣΕ A[i, j]

          ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

     ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

     !Η επεξεργασία της αντιγραφής από τον A στον B.
```

```
        'απάντηση στο β)
        'Αρχική τιμή στον δείκτη θέσης του Β. Δεν γνωρίζουμε από
        'πριν πόσα στοιχεία θα αντιγραφούν.
        k ← 0

        ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 15

            ΓΙΑ j ΑΠΟ 1 ΜΕΧΡΙ 10

                ΑΝ A[i,j] MOD 3 = 0 ΤΟΤΕ

                    k ← k + 1
                    B[k] ← A[i,j]

                ΤΕΛΟΣ_ΑΝ

            ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

        ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

        !Τα στοιχεία αντιγράφηκαν. Το πλήθος τους είναι ίσο με την
        !τιμή του k. (απάντηση στο γ).
        ΓΡΑΨΕ  k, ' στοιχεία του Α αντιγράφηκαν στον Β'

ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Δυαδική αναζήτηση σε δισδιάστατο πίνακα

Παρακάτω, υλοποιούμε στη ΓΛΩΣΣΑ έναν αλγόριθμο δυαδικής αναζήτησης σε πίνακα δύο διαστάσεων. Όπως θα δούμε, πρόκειται για προσαρμογή του αλγόριθμου από μονοδιάστατο σε δισδιάστατο πίνακα.

Η λογική του αλγορίθμου είναι η εξής:

1. Βρίσκουμε το μεσαίο στοιχείο του πίνακα. Σε αυτό το σημείο, θεωρούμε τον πίνακα γραμμή-γραμμή, δηλαδή σε έναν πίνακα Π[5,3] το 1ο στοιχείο είναι το Π[1,1], το 2ο το Π[1,2], το 3ο το Π[1,3], το 4ο το Π[2,1] κλπ. Με άλλα λόγια, τον θεωρούμε ως ένα σύνολο θέσεων (όπως και τον μονοδιάστατο) γραμμή προς γραμμή.

   Σε έναν πίνακα, για παράδειγμα Π[5,3], αρχικά, το σύνολο θέσεων (flat) είναι 15 και το μεσαίο στοιχείο είναι το 8ο ((1+15) DIV 2).

2. Τώρα, πρέπει να αντιστοιχίσουμε τη θέση του μεσαίου στοιχείου από το σύνολο θέσεων (flat) στην πραγματική θέση `[i,j]` του πίνακα (2 dimensional).

3. Αν το μεσαίο στοιχείο του πίνακα ισούται με αυτό που ψάχνουμε (`key`) τότε το βρήκαμε.

4. Αν το μεσαίο στοιχείο του πίνακα είναι μικρότερο από αυτό που ψάχνουμε (`key`) τότε πρέπει να ψάξουμε στο δεύτερο μισό (άνω μέρος) του πίνακα. Έτσι, η αναζήτηση θα περιοριστεί σε έναν νέο «νοητό πίνακα» που αποτελεί το δεύτερο μισό του αρχικού.

5. Αν το μεσαίο στοιχείο του πίνακα είναι μεγαλύτερο από αυτό που ψάχνουμε (`key`) τότε πρέπει να ψάξουμε στο πρώτο μισό (κάτω μέρος) του πίνακα. Έτσι, η αναζήτηση θα περιοριστεί σε έναν νέο «νοητό πίνακα» που αποτελεί το πρώτο μισό του αρχικού.

6. Συνέχισε με το βήμα 1 μέχρι να το βρεις ή μέχρι να μην υπάρχει πια άλλος «νοητός πίνακας».

Θα χρειαστούμε:

- Τους κλασσικούς δείκτες `i` και `j`, για τον προσδιορισμό μίας θέσης του πίνακα.

- Δύο έξτρα δείκτες, θα τους ονομάσουμε `left` και `right`, για τον προσδιορισμό του «νοητού» πίνακα, μέσα στον οποίο γίνεται η αναζήτηση του `key`. Αυτοί οι δείκτες λειτουργούν στο σύνολο θέσεων (flat) αυτού του νοητού πίνακα.

Να γραφεί πρόγραμμα στη `ΓΛΩΣΣΑ` που διαβάζει σε έναν πίνακα `Π[4,5]` ακεραίους αριθμούς σε αύξουσα διάταξη (δηλαδή, η εισαγωγή τους θα γίνεται από τον μικρότερο στον μεγαλύτερο). Στη συνέχεια, ζητάει από τον χρήστη να δώσει έναν ακέραιο και το πρόγραμμα τον αναζητάει μέσα στον ταξινομημένο πίνακα. Αν τον βρει, εμφανίζει τη θέση του και αν όχι εμφανίζει ένα κατάλληλο μήνυμα περί μη εύρεσης του ακεραίου.

```
ΠΡΟΓΡΑΜΜΑ Δυαδική_αναζήτηση_στοιχείου_σε_δισδιάστατο_πίνακα
ΜΕΤΑΒΛΗΤΕΣ
     ΑΚΕΡΑΙΕΣ: Π[4,5], key, i, j , m, left, right
     ΛΟΓΙΚΕΣ: βρέθηκε
ΑΡΧΗ

     !Είσοδος δεδομένων, εδώ των 20 ακεραίων.
     !Εξωτερικός βρόχος για τις γραμμές.
     ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 4
        !Εσωτερικός βρόχος για τις στήλες (θέσεις της γραμμής).
```

```
    ΓΙΑ j ΑΠΟ 1 ΜΕΧΡΙ 5

      ΓΡΑΨΕ 'Παρακαλώ, δώστε τον ',(i-1)*5+j, 'ο ακέραιο:'
      ΔΙΑΒΑΣΕ Π[i, j]

      ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

  ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

  !Είσοδος του ακεραίου προς εύρεση, key.
  ΓΡΑΨΕ 'Παρακαλώ, δώστε τον ακέραιο που ψάχνετε:'
  ΔΙΑΒΑΣΕ key

  !Αρχική τιμή της μεταβλητής-σημαίας βρέθηκε.
  βρέθηκε ← ΨΕΥΔΗΣ

  !Αρχικές τιμές των δεικτών θέσης left και right. Θεωρούμε
  !τον πίνακα ως σύνολο θέσεων από το 1 έως το 4 * 5 = 20.
  left ← 1
  right ← 20

  !Σαρώνει δυαδικά τον πίνακα Π για να βρει το key.
  ΟΣΟ left <= right ΚΑΙ (ΌΧΙ βρέθηκε)

    !Ο δείκτης m είναι στο μέσον του πίνακα.
    m ← (left + right) DIV 2

    !Ο δείκτης m δείχνει τη θέση στο σύνολο θέσεων (flat)
    !του πίνακα. Τώρα, πρέπει να αντιστοιχηθεί στην
    !πραγματική θέση [i,j] του δισδιάστατου πίνακα.
    i ← m DIV 5
    j ← m MOD 5

    ΑΝ j > 0 ΤΟΤΕ
       i ← i + 1
    ΑΛΛΙΩΣ
       j ← 5
    ΤΕΛΟΣ_ΑΝ

    !Αν το μεσαίο στοιχείο Π[i,j] ισούται με το key
    !τότε βρέθηκε.
    ΑΝ  Π[[i,j] = key   ΤΟΤΕ
       ΓΡΑΨΕ 'Βρέθηκε στην γραμμή' , i, ' και στήλη ',j
       βρέθηκε ← ΑΛΗΘΗΣ

    !Αλλιώς αν το μεσαίο στοιχείο Π[i,j] είναι
    !μικρότερο από το key τότε πρέπει να ψάξουμε στο δεύτερο
```

```
        !μισό του πίνακα. Συνεπώς, ανεβάζουμε το κάτω όριο
        !(left) στη θέση m + 1.
        ΑΛΛΙΩΣ_ΑΝ  Π[[i,j]  < key ΤΟΤΕ
           left ← m + 1

        !Αλλιώς αν το μεσαίο στοιχείο Π[[i,j] είναι
        !μεγαλύτερο από το key τότε πρέπει να ψάξουμε στο
        !πρώτο μισό του πίνακα. Συνεπώς, κατεβάζουμε το πάνω
        !όριο (right)στη θέση m - 1.
        ΑΛΛΙΩΣ
           right ← m - 1

        ΤΕΛΟΣ_ΑΝ

     ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

     !Αν δεν βρέθηκε τότε εμφάνισε ένα σχετικό μήνυμα στην
     !οθόνη.
     ΑΝ  ΟΧΙ βρέθηκε  ΤΟΤΕ
          ΓΡΑΨΕ 'Ο ακέραιος ', key, ' δεν βρέθηκε στον πίνακα'
     ΤΕΛΟΣ_ΑΝ

ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Εκτέλεση με δοκιμαστικά δεδομένα

Αρχικός πίνακας:

j →

Π[4,5]				
2	8	10	12	16
22	28	35	36	39
41	45	50	56	58
62	65	69	77	78

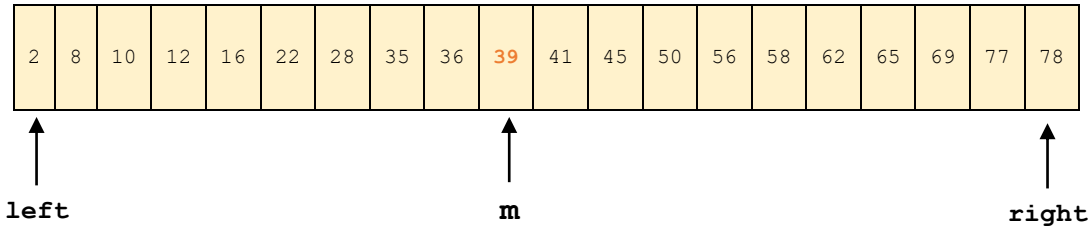
i ↓

Η άποψη του πίνακα ως σύνολο θέσεων γραμμή προς γραμμή (flat):

2	8	10	12	16	22	28	35	36	39	41	45	50	56	58	62	65	69	77	78

Key ← 65

- **<u>1η επανάληψη</u>:** `left ← 1 , right ← 20, m ← (1+20) DIV 2 = 10`

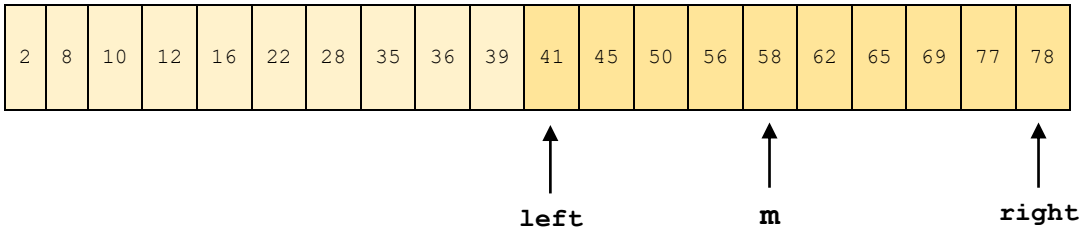


Πραγματική θέση στον Π: `i ← 10 DIV 5 = 2`
`j ← 10 MOD 5 = 0` γίνεται `j ← 5`
`Π[2,5]=39`

`39 < 65,` άρα θα συνεχίσουμε στο δεύτερο μισό (πάνω μέρος) του πίνακα
(**left** ← `m + 1 = 10 + 1 = 11,` **right** ← `20`)

- **<u>2η επανάληψη</u>:** `left ← 11 , right ← 20, m ← (11+20) DIV 2 = 15`

Το σύνολο θέσεων στον οποίο θα περιοριστεί η αναζήτηση φαίνεται με έντονο κίτρινο χρώμα.

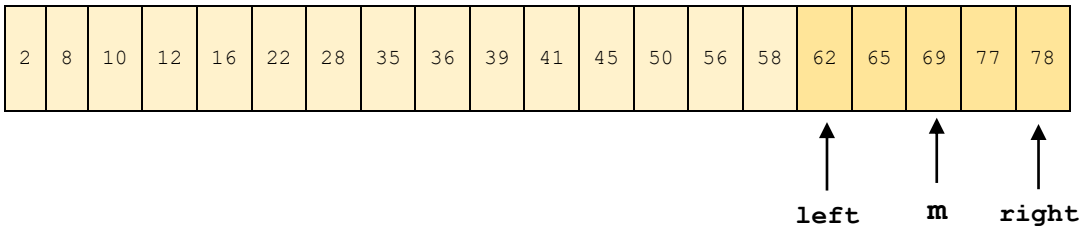


Πραγματική θέση στον Π: `i ← 15 DIV 5 = 3`
`j ← 15 MOD 5 = 0` γίνεται `j ← 5`
`Π[3,5]=58`

`58 < 65,` άρα θα συνεχίσουμε στο δεύτερο μισό (πάνω μέρος) του πίνακα
(**left** ← `m + 1 = 15 + 1 = 16,` **right** ← `20`)

- **<u>3η επανάληψη</u>:** `left ← 16 , right ← 20, m ← (16+20) DIV 2 = 18`

  Το σύνολο θέσεων στον οποίο θα περιοριστεί η αναζήτηση φαίνεται με έντονο κίτρινο χρώμα.

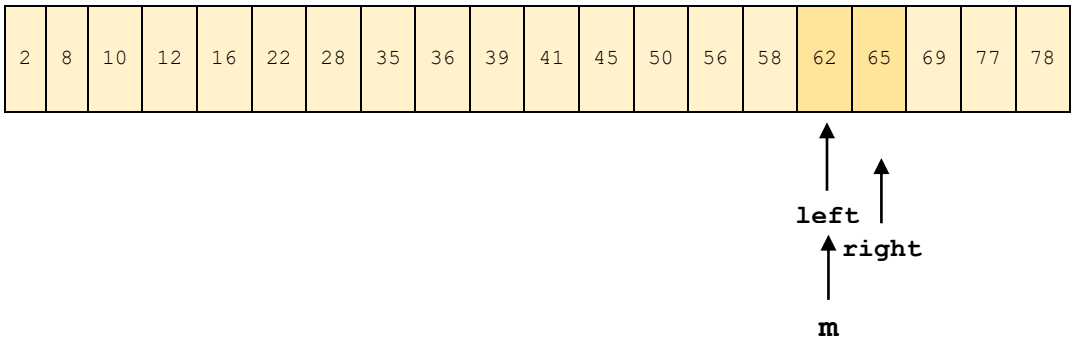


Πραγματική θέση στον Π: `i ← 18 DIV 5 = 3`
`j ← 18 MOD 5 = 3 > 0` γίνεται `i ← 4`
`Π[4,3]=69`

`69 < 65,` άρα θα συνεχίσουμε στο πρώτο μισό (κάτω μέρος) του πίνακα
(**left** ← 16, **right** ← 18−1=17)

- **<u>4η επανάληψη</u>:** `left ← 16 , right ← 17, m ← (16+17) DIV 2 = 16`

  Το σύνολο θέσεων στον οποίο θα περιοριστεί η αναζήτηση φαίνεται με έντονο κίτρινο χρώμα.

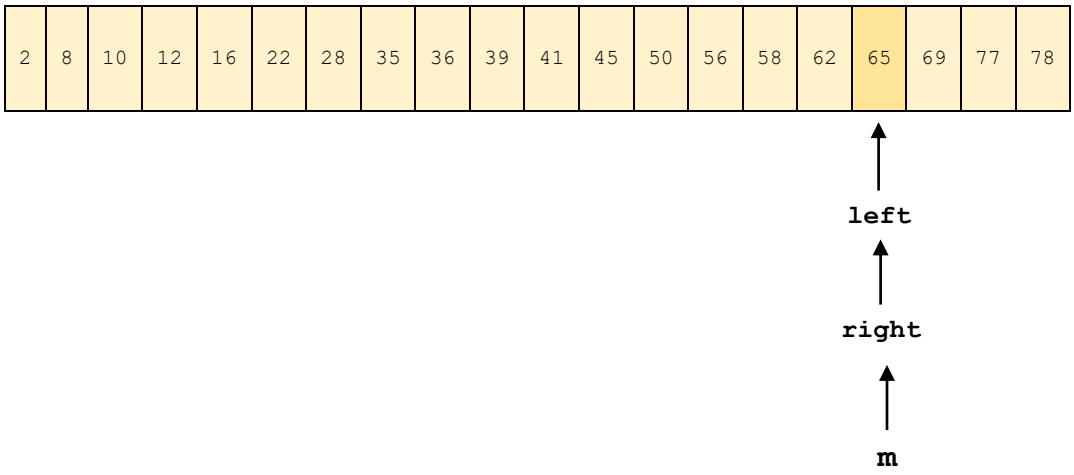


Πραγματική θέση στον Π: `i ← 16 DIV 5 = 3`
`j ← 16 MOD 5 = 1 > 0` γίνεται `i ← 4`
`Π[4,1]=62`

`62 < 65,` άρα θα συνεχίσουμε στο δεύτερο μισό (πάνω μέρος) του πίνακα
(**left** ← 16+1=17, **right** ← 17)

- **5η επανάληψη:** `left ← 17 , right ← 17, m ← (17+17) DIV 2 = 17`

  Το σύνολο θέσεων στον οποίο θα περιοριστεί η αναζήτηση φαίνεται με έντονο κίτρινο χρώμα.



Πραγματική θέση στον Π: `i ← 17 DIV 5 = 3`
`j ← 17 MOD 5 = 2 > 0` γίνεται `i ← 4`

Π[4,2]=65

`65 = 65,` **βρέθηκε**.

ΤΕΛΟΣ ΣΗΜΕΙΩΣΕΩΝ ΚΕΦΑΛΑΙΟΥ 9