γενικό λύκειο
νέων μουδανιών

ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ - ΠΙΝΑΚΕΣ

ΑΝΑΠΤΥΞΗ ΕΦΑΡΜΟΓΩΝ ΣΕ ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΟ ΠΕΡΙΒΑΛΛΟΝ

ΠΕΡΙΕΧΟΜΕΝΑ

ΓΙΑΤΙ ΧΡΕΙΑΖΟΜΑΣΤΕ ΠΙΝΑΚΕΣ (τροποποίηση της §9.1 του βιβλίου)....2
ΜΟΝΟΔΙΑΣΤΑΤΟΙ ΠΙΝΑΚΕΣ....3
ΠΡΟΣΠΕΛΑΣΗ....3
(διάβασμα, αρχικοποίηση, τροποποίηση, εμφάνιση, έλεγχος στοιχείων μονοδιάστατου πίνακα)....3
Υπολογισμός αθροίσματος - Μ.Ο. μονοδιάστατου πίνακα....4
Εύρεση min και max και θέσεων τους σε μονοδιάστατο πίνακα (παρ.1, σελ. 57 βιβ. Μαθητή)....4
ΤΑΞΙΝΟΜΗΣΗ....5
ΑΝΑΖΗΤΗΣΗ....5
ΑΝΤΙΓΡΑΦΗ - ΣΥΓΧΩΝΕΥΣΗ - ΔΙΑΧΩΡΙΣΜΟΣ....6
ΔΙΣΔΙΑΣΤΑΤΟΙ ΠΙΝΑΚΕΣ....7
Υπολογισμός αθροίσματος - Μ.Ο. δισδιάστατου πίνακα....8
Μέγιστα και ελάχιστα δισδιάστατου πίνακα....9
ΤΕΤΡΑΓΩΝΙΚΟΙ ΠΙΝΑΚΕΣ....11
ΠΑΡΑΛΛΗΛΟΙ ΠΙΝΑΚΕΣ....11
ΘΕΜΑ 4 ΑΕΠΠ ΠΑΝΕΛΛΗΝΙΩΝ ΕΞΕΤΑΣΕΩΝ 2004....11
ΛΥΜΕΝΑ ΠΑΡΑΔΕΙΓΜΑΤΑ....13

Διδάσκων Καθηγητής:

Γιώργος Μαλακούδης

e-mail: gmalakoudi@sch.gr

ΓΙΑΤΙ ΧΡΕΙΑΖΟΜΑΣΤΕ ΠΙΝΑΚΕΣ (τροποποίηση της §9.1 του βιβλίου)

Πολλά από τα προβλήματα που έχουμε αντιμετωπίσει μέχρι στιγμής απλά επεξεργάζονται μία σειρά δεδομένων.

Διαβάζουν ένα δεδομένο κάθε φορά, το εκχωρούν σε μία μεταβλητή, εκτελούν τους αντίστοιχους υπολογισμούς και στη συνέχεια επαναλαμβάνεται η ίδια διαδικασία μέχρι να τελειώσουν όλα τα δεδομένα.

Για παράδειγμα ένα τμήμα αλγορίθμου το οποίο διαβάζει τις θερμοκρασίες όλων των ημερών του μήνα, έστω 30, και υπολογίζει τη μέση θερμοκρασία, μπορεί πολύ απλά να γραφεί ως εξής

```
sum←0
ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 30
   ΔΙΑΒΑΣΕ Θ
   sum ← sum + Θ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΜΟ ← sum/30
```

Αυτό που παρατηρούμε, βέβαια, είναι ότι με τη χρησιμοποίηση μόνο μίας μεταβλητής για την αποθήκευση 30 διαδοχικών τιμών θερμοκρασίας (στο συγκεκριμένο παράδειγμα), ουσιαστικά οι τιμές "χάνονται" μετά από κάθε επανάληψη – η μόνη τελικά θερμοκρασία που "μένει" στη μεταβλητή *Θ* είναι αυτή της 30ης ημέρας, γιατί δε θα εκτελεστεί άλλη επανάληψη. Παρόλα αυτά, η δουλειά μας γίνεται χρησιμοποιώντας μόνο μία μεταβλητή, τη μεταβλητή *Θ,* το πρόβλημα λύνεται πολύ απλά και το αντίστοιχο πρόγραμμα είναι σύντομο και κατανοητό και πιο "εύκολο" για τον υπολογιστή να το εκτελέσει.

Αν όμως το πρόβλημα ζητούσε και το πλήθος των ημερών που η θερμοκρασία ήταν κατώτερη της μέσης, τότε η σύγκριση αυτή θα έπρεπε να γίνει μετά τον υπολογισμό της μέσης θερμοκρασίας. Αυτό σημαίνει ότι όλες οι θερμοκρασίες θα έπρεπε να ξαναδιαβαστούν για να συγκριθούν με τη μέση.

Μία λύση για να επιλύσουμε τέτοιου είδους προβλήματα είναι να καταχωρηθεί κάθε θερμοκρασία σε διαφορετική μεταβλητή, έτσι ώστε κάθε τιμή που εισάγεται να διατηρείται στη μνήμη και να μπορεί να συγκριθεί με τη μέση, αφού αυτή υπολογιστεί. Τότε όμως πρέπει να δημιουργηθούν 30 διαφορετικές μεταβλητές Θ1, Θ2,..., Θ30. Για να γραφεί το πρόγραμμα χρειάζονται τριάντα εντολές ΔΙΑΒΑΣΕ και τριάντα εντολές ΑΝ.

Αν και αυτή η λύση είναι σωστή και πρακτική για μικρό αριθμό δεδομένων, προφανώς δεν εξυπηρετεί την επεξεργασία μεγάλου αριθμού δεδομένων.

Η καλύτερη λύση στο πρόβλημα αυτό είναι η χρήση μεταβλητής με δείκτες, έννοια που είναι γνωστή από τα μαθηματικά και υλοποιείται στον προγραμματισμό με τη δομή δεδομένων του **πίνακα.** Χρησιμοποιείται λοιπόν μόνο ένα όνομα ***Θ[ ],*** που αναφέρεται και στις τριάντα διαφορετικές θερμοκρασίες.

Το όνομα του πίνακα καθορίζει μία ομάδα διαδοχικών θέσεων στη μνήμη. Κάθε συγκεκριμένη θέση μνήμης καλείται στοιχείο του πίνακα και προσδιορίζεται από την τιμή ενός δείκτη για τους μονοδιάστατους και δύο δεικτών για τους δισδιάστατους..

Τελικά όμως πότε θα χρησιμοποιούμε πίνακες στους αλγόριθμους μας;

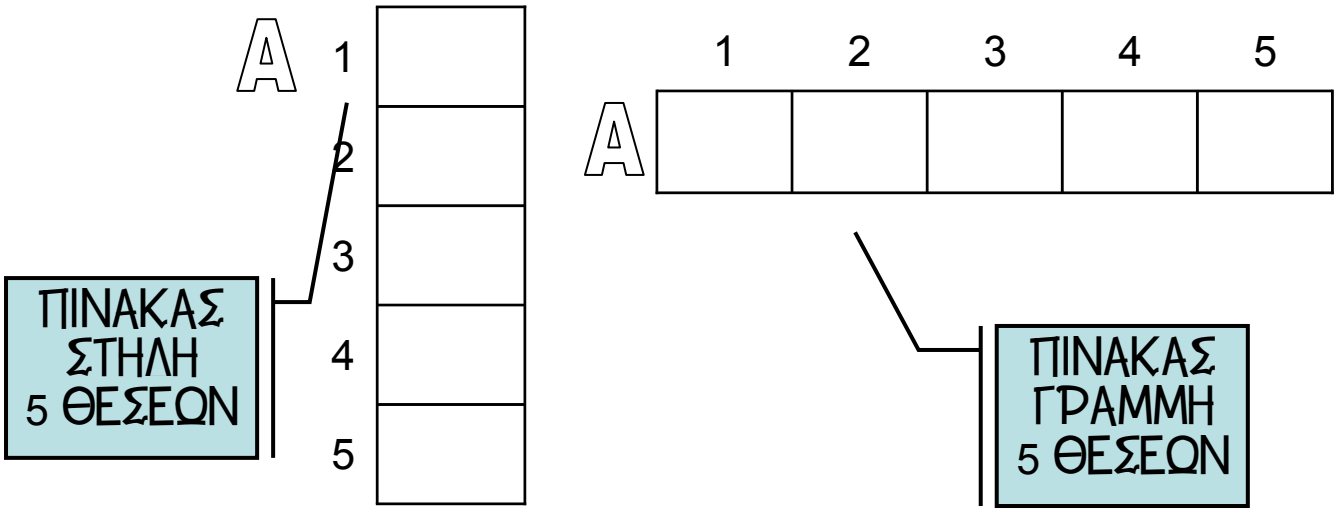
Πίνακες θα πρέπει να χρησιμοποιούμε όταν πραγματικά μας χρειάζεται όπως όταν κάποια δεδομένα θέλουμε να τα επεξεργαστούμε παραπάνω από μία φορά.

Επίσης πίνακες θα χρησιμοποιούμε όταν αναφέρεται ρητά από την εκφώνηση του προβλήματος.

Ουσιαστικά οι πίνακες είναι μια ομάδα μεταβλητών πού έχουν το ίδιο όνομα και χρησιμοποιούμε, και κάποιους δείκτες (ακέραιους αριθμούς), για να δείξουμε ποια από όλες αυτές ης μεταβλητές θέλουμε. Έτσι τα στοιχεία ενός πίνακα θα τα επεξεργαζόμαστε σαν να είναι μεταβλητές. Θα τους δίνουμε αρχική τιμή όπου χρειάζεται, θα εκχωρούμε σ' αυτά νέες τιμές, θα τα χρησιμοποιούμε σε εκφράσεις κ.ο.κ. Αν ένα στοιχείο ενός πίνακα δεν έχει πάρει καμία τιμή, τότε η τιμή του θα είναι απροσδιόριστη, όπως ακριβώς συμβαίνει και με τις μεταβλητές.

ΜΟΝΟΔΙΑΣΤΑΤΟΙ ΠΙΝΑΚΕΣ

Ένας μονοδιάστατος πίνακας με *n* στοιχεία (το *n* σταθερό) μπορεί να αναπαρασταθεί οπτικά με δύο μορφές: α) ως πίνακας – γραμμή και β) ως πίνακας – στήλη. Για παράδειγμα, αν n=5, στο παρακάτω σχήμα φαίνονται οι δύο διαφορετικές οπτικές αναπαραστάσεις των πινάκων για καθεμία από τις δύο μορφές. Τονίζεται, ότι στην ουσία, δεν υπάρχει καμία απολύτως διαφορά μεταξύ των δύο μορφών μονοδιάστατων πινάκων. Απλά, ο διαφορετικός τρόπος απεικόνισης, θα δούμε ότι, μας βοηθάει όταν από δισδιάστατο πίνακα, με κάποια επεξεργασία, δημιουργούμε άλλους μονοδιάστατους πίνακες (π.χ. για την αποθήκευση αθροισμάτων κατά γραμμές ενός δισδιάστατου πίνακα χρησιμοποιούμε την αναπαράσταση ενός πίνακα – στήλη κλπ.)



ΠΡΟΣΠΕΛΑΣΗ

(διάβασμα, αρχικοποίηση, τροποποίηση, εμφάνιση, έλεγχος στοιχείων μονοδιάστατου πίνακα)

Η προσπέλαση ορίζεται ως εξής (Βιβλίο Μαθητή, σελ. 54): ***Προσπέλαση*** (access), πρόσβαση σε ένα κόμβο (μίας δομής δεδομένων) με σκοπό να εξετασθεί ή να τροποποιηθεί το περιεχόμενο του.

Στους πίνακες, εφόσον είναι στατικές δομές δεδομένων, ο αριθμός των κόμβων (ή κελιών όπως συνηθίζουμε αλλιώς να λέμε) είναι σταθερός και αμετάβλητος από την αρχή μέχρι το τέλος της λειτουργίας ενός προγράμματος. Ουσιαστικά, προσπελαύνουμε έναν πίνακα όταν δίνουμε εντολή να διαβάζονται κάποιες τιμές και να αποθηκεύονται στα κελιά του πίνακα (*διάβασμα* του πίνακα από τον υπολογιστή), όταν εκχωρούμε κάποιες τιμές απευθείας σε κάποια στοιχεία του πίνακα (τροποποίηση ή αρχικοποίηση πίνακα), όταν δίνουμε εντολή να εμφανίζεται κάποιο στοιχείο ή όλα τα στοιχεία του πίνακα (*εμφάνιση* στοιχείων πίνακα) και όταν με κάποια εντολή επιλογής (κυρίως με την ΑΝ) ελέγχουμε το περιεχόμενο των στοιχείων ενός πίνακα για να πάρουμε κάποια απόφαση.

Για να αναφερθούμε σε ένα στοιχείο ενός μονοδιάστατου πίνακα, δουλεύουμε όπως με τις μεταβλητές, με τη διαφορά ότι τώρα χρησιμοποιούμε το όνομα του πίνακα ακολουθούμενο από έναν δείκτη σε παρένθεση για να "δείξουμε" σε ποιο στοιχείο αναφερόμαστε. Π.χ., για τον παραπάνω πίνακα Α( ) 5 θέσεων, μπορούμε να δώσουμε μεμονωμένα τις εξής εντολές:

Διάβασε Α[3]	*η τιμή που πληκτρολογείται αποθηκεύεται στη θέση 3 του πίνακα Α, δηλ. Α(3)*
Α[2]←0	*η τιμή 0 εκχωρείται στο στοιχείο Α(2)*
Α[3]←Α[2]+2	*η τιμή του στοιχείου Α(2) + 2 εκχωρείται στο στοιχείο Α(3)*
Γράψε Α[2]	*η τιμή του στοιχείου Α(2) εμφανίζεται στην οθόνη*
Αν Α[1]mod 2 =0 τότε…	*ελέγχεται αν το Α(1) είναι άρτιος*

Στην περίπτωση που θέλουμε να προσπελάσουμε συνολικά όλα τα στοιχεία ενός μονοδιάστατου πίνακα, τότε αρκεί να "πλαισιώσουμε" με μία [ΓΙΑ i από 1 μέχρι n] την εντολή ή τις εντολές θέλουμε να εκτελεστούν για τον πίνακα n θέσεων. Για παράδειγμα, πάλι για τον προηγούμενο πίνακα Α[i] 5 θέσεων, θα έχουμε:

"Διάβασμα" των στοιχείων του πίνακα Α	Αρχικοποίηση όλων των στοιχείων του Α στο 0	Εμφάνιση όλων των στοιχείων του πίνακα Α	Υπολογισμός & εμφάνιση πλήθους άρτιων στοιχείων ενός πίνακα Α
Για i από 1 μέχρι 5 Διάβασε Α[i] Τέλος_επανάληψης	Για i από 1 μέχρι 5 Α[i]←0 Τέλος_επανάληψης	Για i από 1 μέχρι 5 Γράψε Α[i] Τέλος_επανάληψης	pl ← 0 Για i από 1 μέχρι 5 Αν Α[i] mod 2 = 0 τότε pl←pl+1 Τέλος_αν Τέλος_επανάληψης Γράψε 'άρτιοι= ', pl
Για i από 1 μέχρι 5 Διάβασε Β Α[i]←Β Τέλος_επανάληψης			

Είναι φανερό ότι από τα παραπάνω παραδείγματα, ότι με την τροποποίηση των παραμέτρων της ΓΙΑ, μπορούμε να αναφερθούμε με διαφορετική σειρά στα στοιχεία του πίνακα. Π.χ. αν γράψουμε

```
Για i από 5 μέχρι 1 με_βήμα -1
        Γράψε Α[i]
Τέλος_επανάληψης
```

τότε, προφανώς τα στοιχεία του πίνακα θα εμφανιστούν από τη θέση 5 προς τη θέση 1.

Υπολογισμός αθροίσματος - Μ.Ο. μονοδιάστατου πίνακα

Για να βρούμε το άθροισμα των στοιχείων ενός μονοδιάστατου πίνακα, εργαζόμαστε με τη γνωστή μεθοδολογία υπολογισμού αθροίσματος μιας σειράς δεδομένων. Ως γνωστόν, η εντολή υπολογισμού αθροίσματος μίας σειράς αθροιστέων είναι sum← sum + *αθροιστέος*. Εφόσον ο πίνακας είναι στατική δομή δεδομένων, έχει σταθερό πλήθος στοιχείων. Έτσι, σε έναν πίνακα table ( ) με n στοιχεία το άθροισμα των στοιχείων του sum θα ισούται με table(1)+ table(2)+ table(3)+…+ table(n-1)+ table(n). Χρησιμοποιώντας μία εντολή ΓΙΑ i από 1 μέχρι n … Τέλος επανάληψης που "πλαισιώνει" τον γενικό τύπο του αθροίσματος sum←sum + table(i) σχηματίζεται το ζητούμενο άθροισμα. Κατόπιν, ο υπολογισμός του ΜΟ είναι εύκολη υπόθεση (Ο πίνακας θεωρείται "γεμάτος" από το προηγούμενο τμήμα του αλγόριθμου):

```
sum←0
Για i από 1 μέχρι n
        sum←sum+table(i)
Τέλος_επανάληψης
ΜΟ←sum/n
```

Εύρεση min και max και θέσεων τους σε μονοδιάστατο πίνακα (παρ.1, σελ. 57 βιβ. Μαθητή)

Στην περίπτωση που ζητείται μέγιστο ή ελάχιστο στοιχείο μονοδιάστατου πίνακα, πάλι επεκτείνουμε τη μεθοδολογία που είχαμε μάθει. Συγκρίνουμε διαδοχικά, όλα τα στοιχεία του πίνακα με το max ή το min, άρα πάλι "πλαισιώνουμε" με ΓΙΑ. ΠΡΟΣΟΧΗ στην αρχικοποίηση των min και max! Θεωρούμε και πάλι την γενικευμένη περίπτωση ενός μονοδιάστατου πίνακα table( ) με n στοιχεία (Ο πίνακας θεωρείται "γεμάτος" από το προηγούμενο τμήμα του αλγόριθμου):

Με αρχικοποίηση των min, max στην πρώτη τιμή του πίνακα (ΠΡΟΤΕΙΝΕΤΑΙ)	Με αρχικοποίηση των min, max σε μία πολύ μεγάλη και μία πολύ μικρή τιμή αντίστοιχα
min←table[1] ; θέσημin←1 !ΑΡΧΙΚΟΠΟΙΗΣΗ ΜΙΝ! max←table[1] ; θέσημax←1 !ΑΡΧΙΚΟΠΟΙΗΣΗ ΜΑΧ! Για i από 2 μέχρι n Αν table[i]<min τότε !ΕΥΡΕΣΗ ΜΙΝ! min←table[i] θέσημin←i Τέλος_αν !ΞΕΧΩΡΙΣΤΕΣ ΑΝ! Αν table[i]>max τότε !ΕΥΡΕΣΗ ΜΑΧ! max←table[i] θέσημax←i Τέλος_αν Τέλος_επανάληψης	min←2^32 !ΑΡΧΙΚΟΠΟΙΗΣΗ ΜΙΝ! max←-2^32 !ΑΡΧΙΚΟΠΟΙΗΣΗ ΜΑΧ! Για i από 1 μέχρι n Αν table[i]<min τότε !ΕΥΡΕΣΗ ΜΙΝ! min←table[i] θέσημin←i Τέλος_αν !ΞΕΧΩΡΙΣΤΕΣ ΑΝ! Αν table[i]>max τότε !ΕΥΡΕΣΗ ΜΑΧ! max←table[i] θέσημin←i Τέλος_αν Τέλος_επανάληψης

ΤΑΞΙΝΟΜΗΣΗ

Ταξινόμηση (sorting) καλείται η διαδικασία κατά την οποία οι κόμβοι μιας δομής διατάσσονται κατά αύξουσα ή φθίνουσα σειρά.

Υπάρχουν πολλοί διαφορετικοί αλγόριθμοι ταξινόμησης, άλλοι πιο γρήγοροι και άλλοι πιο αργοί. Εμείς, στα πλαίσια του μαθήματος, μαθαίνουμε τον αλγόριθμο της ταξινόμησης ευθείας ανταλλαγής ή φυσαλίδας (bubble sort) όπως είναι γνωστότερος.

Η μέθοδος της ταξινόμησης της φυσαλίδας συνίσταται στην επαναληπτική σύγκριση διαδοχικών στοιχείων ενός μονοδιάστατου πίνακα ανά δύο μεταξύ τους και στην αντιμετάθεση των τιμών τους – αν χρειάζεται, προκειμένου η "ελαφρύτερη" τιμή να ανέβει προς τα επάνω – αν πρόκειται για αύξουσα ταξινόμηση, ή προκειμένου η "βαρύτερη" τιμή να ανέβει προς τα επάνω – αν πρόκειται για φθίνουσα ταξινόμηση. Το κομμάτι του γενικευμένου αλγόριθμου της ταξινόμησης ενός μονοδιάστατου πίνακα n στοιχείων με το όνομα table έχει ως εξής (θεωρείται ότι πίνακας είναι "γεμάτος" από το προηγούμενο κομμάτι του αλγόριθμου):

Αύξουσα ταξινόμηση	**Φθίνουσα ταξινόμηση**
Για i **από** 2 **μέχρι** **n** **Για** **j** **από** **n** **μέχρι** **i** **με_βήμα** -1 **Αν** table[j-1] > table[j] **τότε** **αντιμετάθεσε** table[j-1], table[j] **Τέλος_αν** **Τέλος_επανάληψης** **Τέλος_επανάληψης**	**Για** i **από** 2 **μέχρι** **n** **Για** **j** **από** **n** **μέχρι** **i** **με_βήμα** -1 **Αν** table[j-1] < table[j] **τότε** **αντιμετάθεσε** table[j-1], table[j] **Τέλος_αν** **Τέλος_επανάληψης** **Τέλος_επανάληψης**

Τον αλγόριθμο αυτόν τον χρησιμοποιούμε ως έχει, με τη διαφοροποίηση, φυσικά, του ονόματος του πίνακα *table* και των θέσεων του *n,* ανάλογα με το όνομα και τις θέσεις του πίνακα που θέλουμε να ταξινομήσουμε.

Η "συνθήκη – κλειδί" που καθορίζει την ταξινόμηση είναι η συνθήκη στην εντολή ΑΝ. Εκεί, δίνουμε εντολή ποιον πίνακα θέλουμε να ταξινομήσουμε και κατά ποια σειρά (αύξουσα ή φθίνουσα). Ο αλγόριθμος της ταξινόμησης έχει κάποιες παραλλαγές που θα συζητήσουμε παρακάτω (π.χ. παράλληλοι πίνακες).

ΠΑΡΑΤΗΡΗΣΤΕ, επίσης, ότι από τη στιγμή που ο μονοδιάστατος πίνακας ταξινομηθεί, αυτόματα γνωρίζουμε το min και max του.

ΑΝΑΖΗΤΗΣΗ

Αναζήτηση (searching) ονομάζεται η διαδικασία, κατά την οποία προσπελαύνονται οι κόμβοι μιας δομής, προκειμένου να εντοπιστούν ένας ή περισσότεροι που έχουν μια δεδομένη ιδιότητα (π.χ. αν η τιμή τους ισούται με μία τιμή – κλειδί).

Όπως και με την ταξινόμηση, υπάρχουν πολλές διαφορετικές υλοποιήσεις αλγορίθμων αναζήτησης, άλλες πιο αποδοτικές και άλλες πιο χρονοβόρες και "δύσκολες" για τον υπολογιστή. Ο βασικός αλγόριθμος που μαθαίνουμε είναι αυτός της Σειριακής αναζήτησης (sequential search) για αναζήτηση μίας τιμής – κλειδιού σε έναν αταξινόμητο μονοδιάστατο πίνακα table n θέσεων. Παρόλα αυτά, έχοντας ως βάση αυτόν τον αλγόριθμο μπορούμε να επεκταθούμε και σε πιο πολύπλοκους αλγόριθμους αναζήτησης (που αφορούν π.χ. ταξινομημένους πίνακες).

ΜΗΝ ΞΕΧΝΑΤΕ ΑΥΤΟ ΠΟΥ ΛΕΕΙ ΣΤΟ ΒΙΒΛΙΟ ΣΑΣ: Ουσιαστικά, η ΤΑΞΙΝΟΜΗΣΗ είναι μία διαδικασία που εφαρμόζουμε για να γίνει πιο εύκολη την ΑΝΑΖΗΤΗΣΗ (θυμηθείτε και το παράδειγμα με τους τηλεφωνικούς καταλόγους).

Το κομμάτι του αλγορίθμου αναζήτησης μίας τιμής – κλειδιού key σε μία μόνο θέση (την πρώτη) ενός αταξινόμητου πίνακα table n θέσεων, έχει ως εξής:

Και αυτό το κομμάτι το μαθαίνουμε απ' έξω, με τη διαφοροποίηση, βέβαια, του ονόματος του πίνακα table, των θέσεων του n, αλλά και της τιμής κλειδιού key που αναζητούμε!

```
done ←ψευδής
position ← 0
i ← 1
Όσο (done=ψευδής) και (i<=n)   επανάλαβε
        Αν  table [i]=key τότε
                done  ← αληθής
                position ←i
        Αλλιώς
                i←i+1
        Τέλος_αν
Τέλος_επανάληψης
```

Αυτό το κομμάτι αλγορίθμου είναι πολύ σημαντικό, γιατί αφενός είναι η πρώτη φορά που βλέπουμε στην πράξη τη χρησιμοποίηση μίας λογικής μεταβλητής, και αφετέρου γιατί φαίνονται οι δυνατότητες της ΟΣΟ.

Τη λογική μεταβλητή done τη χρησιμοποιούμε για να "σηματοδοτήσουμε" αν η αναζήτηση μας είναι επιτυχής ή όχι. Έτσι, θεωρούμε ότι θα έχει τιμή *ψευδής* αν δε βρεθεί το κλειδί key που αναζητούμε, και ότι θα γίνει *αληθής* όταν βρεθεί. Παράλληλα, σε συνδυασμό με την ΟΣΟ, μας βοηθάει να διακόψουμε τις επαναλήψεις όταν βρεθεί το κλειδί και γίνει *αληθής,* έτσι ώστε να μη γίνουν άλλες ανούσιες για τον συγκεκριμένο πρόβλημα επαναλήψεις.

Η ΟΣΟ, επίσης, με τον έλεγχο ΚΑΙ της συνθήκης (i<=n), μας διασφαλίζει ότι όσο δε βρίσκεται το κλειδί key θα συνεχίζει να προσπελαύνει τα στοιχεία του πίνακα.

Η μεταβλητή position (ή αν θέλετε *θέση)* μας χρησιμεύει για την αποθήκευση της θέσης του στοιχείου που ψάχνουμε, αφού εκεί εκχωρείται το τρέχον i όταν η αναζήτηση είναι επιτυχής.

Στην περίπτωση, που δεν αναζητούμε μόνο ένα στοιχείο στον πίνακα με την τιμή key, αλλά περισσότερα, η μεταβλητή done δε μας χρειάζεται και η επανάληψη της αναζήτησης γίνεται τόσες φορές, όσα είναι και τα στοιχεία του πίνακα. Δηλ. θα έχει ως εξής:

με ΟΣΟ	με ΓΙΑ
position ← 0 **i ← 1** **Όσο i<=n επανάλαβε** **Αν table [i]=key τότε** **position ←i** **Γράψε position** **Τέλος_αν** **i←i+1** **Τέλος_επανάληψης**	**position ← 0** **Για i από 1 μέχρι n** **Αν table [i]=key τότε** **position ←i** **Γράψε position** **Τέλος_αν** **Τέλος_επανάληψης**

Η προσθήκη της εντολής **Γράψε position** κρίνεται απαραίτητη, έτσι ώστε κάθε φορά που βρίσκεται το αναζητούμενο κλειδί να εμφανίζεται στην οθόνη η θέση του. Αν δεν υπήρχε, μετά την εκτέλεση της αναζήτησης η μεταβλητή position θα έχει αποθηκευμένη μόνο την τιμή του τελευταίου ευρεθέντος στοιχείου στον πίνακα table, και οι προηγούμενες θέσεις θα έχουν "χαθεί".

Εναλλακτικά, σε αυτήν περίπτωση, μήπως θα μπορούσαμε να χρησιμοποιήσουμε έναν πίνακα position( ) για να αποθηκεύσουμε τις θέσεις που βρέθηκε η τιμή κλειδί; Πόσες θέσεις θα έπρεπε να έχει αυτός ο πίνακας; (Άσκηση)

ΑΝΤΙΓΡΑΦΗ – ΣΥΓΧΩΝΕΥΣΗ – ΔΙΑΧΩΡΙΣΜΟΣ

Αντιγραφή (copying) ονομάζεται η διαδικασία, κατά την οποία όλοι οι κόμβοι ή μερικοί από τους κόμβους μίας δομής αντιγράφονται σε μία άλλη δομή.

Π.χ. έστω ότι έχουμε έναν πίνακα Α( ) 100 θέσεων και θέλουμε να αντιγράψουμε τα πρώτα 50 στοιχεία του σε έναν νέο πίνακα Β( ) 50 θέσεων.

```
Για i από 1 μέχρι 50
        Β[i]←Α[i]
Τέλος_επανάληψης
```

Συγχώνευση (merging) ονομάζεται η διαδικασία, κατά την οποία δυο ή περισσότερες δομές συνενώνονται σε μία ενιαία δομή.

Π.χ. έστω ότι έχουμε δύο "γεμάτους" πίνακες Α( ) και Β( ) 50 και 20 θέσεων αντίστοιχα. Για να δημιουργήσουμε έναν νέο πίνακα Γ( ) προφανώς 70 θέσεων με πρώτα τα στοιχεία του Α( ), γράφουμε:

```
Για i από 1 μέχρι 50
        Γ(i)←Α(i)
Τέλος_επανάληψης

Για i από 1 μέχρι 20
        Γ[i+50]←Β[i]
Τέλος_επανάληψης
```

Διαχωρισμός (separation)είναι η αντίστροφη πράξη της συγχώνευσης.

Π.χ. από πίνακα 70 θέσεων να φτιάξουμε δύο των 35 θέσεων.

Για αυτές τις τρεις επεξεργασίες βλέπετε ασκήσεις

ΔΙΣΔΙΑΣΤΑΤΟΙ ΠΙΝΑΚΕΣ

Δισδιάστατοι είναι οι πίνακες που αποτελούνται από πολλές γραμμές και στήλες. Όλα τα στοιχεία του πίνακα πρέπει να είναι του ίδιου τύπου. Για να αναφερθούμε σε ένα στοιχείο του πίνακα χρησιμοποιούμε το όνομα του και 2 δείκτες, έναν για τις γραμμές κι έναν για τις στήλες, π.χ. Δ(1,1), Δ(11,3), όπως στο σταυρόλεξο. Ουσιαστικά μπορούμε να θεωρήσουμε το δισδιάστατο πίνακα ότι είναι πολλοί μονοδιάστατοι πίνακες ίδιων θέσεων μαζί, ο ένας κάτω ή δίπλα από τον άλλο.

Π.χ. ένας δισδιάστατος πίνακας 12x4 (12 γραμμές και 4 στήλες) με όνομα Δ έχει ως εξής:

Δ

	1	2	3	4
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				

Όπως και στους μονοδιάστατους, μπορούμε κι εδώ να δώσουμε μεμονωμένα εντολές:

Διάβασε Δ(3,1) *η τιμή που πληκτρολογείται αποθηκεύεται στη θέση (3,1) του πίνακα Δ, δηλ. στην τρίτη γραμμή και πρώτη στήλη*

Δ(2,2)←0 *η τιμή 0 εκχωρείται στο στοιχείο Δ(2,2)*

Δ(3,1)←Δ(2,1)+2 *η τιμή του στοιχείου Δ(2,1) + 2 εκχωρείται στο στοιχείο Α(3,1)*

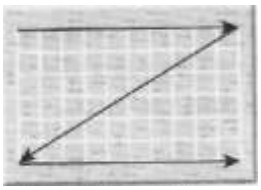
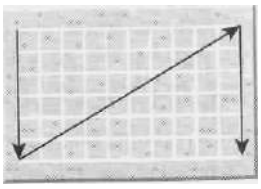
Γράψε Δ(2,4) *η τιμή του στοιχείου Δ(2,4) εμφανίζεται στην οθόνη*

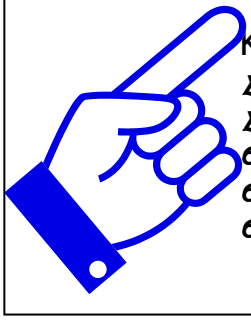
Αν Δ(1,1)mod 2 =0 τότε... *ελέγχεται αν το Δ(1,1) είναι άρτιος*

Όσον αφορά τις βασικές λειτουργίες επεξεργασίας επί ενός δισδιάστατου πίνακα που αφορούν όλα τα στοιχεία του, "πλαισιώνουμε" την εντολή (ή τις εντολές) με 2 ΓΙΑ (άρα θα έχουμε εμφωλευμένο βρόχο) έτσι ώστε να "σαρώσουμε" όλον τον πίνακα. Για λόγους σύμβασης, συνηθίζουμε να χρησιμοποιούμε τον μετρητή i για να αναφερόμαστε στις γραμμές και τον μετρητή j για τις στήλες. Για την περίπτωση του Πίνακα Δ θα έχουμε:

"Διάβασμα" των στοιχείων του πίνακα Δ με σάρωση κατά γραμμές	**"Διάβασμα" των στοιχείων του πίνακα Δ με σάρωση κατά στήλες**	**Εμφάνιση όλων των στοιχείων του πίνακα Δ κατά γραμμές**	**Εμφάνιση όλων των στοιχείων του πίνακα Δ κατά στήλες**

Για i από 1 μέχρι 12
  Για j από 1 μέχρι 4
  Διάβασε Δ[i,j]
  Τέλος_επανάληψης
Τέλος_επανάληψης
	Για j από 1 μέχρι 4
	  Για i από 1 μέχρι 12
	  Διάβασε Δ[i,j]
	  Τέλος_επανάληψης
	Τέλος_επανάληψης
		Για i από 1 μέχρι 12
		  Για j από 1 μέχρι 4
		  Γράψε Δ[i,j]
		  Τέλος_επανάληψης
		Τέλος_επανάληψης
			Για j από 1 μέχρι 4
			  Για i από 1 μέχρι 12
			  Γράψε Δ[i,j]
			  Τέλος_επανάληψης
			Τέλος_επανάληψης

Για i από 1 μέχρι 12
  Για j από 1 μέχρι 4
  Διάβασε Α
  Δ[i,j]←Α
  Τέλος_επανάληψης
Τέλος_επανάληψης
	Για j από 1 μέχρι 4
	  Για i από 1 μέχρι 12
	  Διάβασε Α
	  Δ[i,j]←Α
	  Τέλος_επανάληψης
	Τέλος_επανάληψης
			



ΚΑΝΟΝΑΣ: Στα στοιχεία του πίνακα αναφέρουμε πάντοτε πρώτα τις γραμμές και μετά τις στήλες Δ(i,j). Όταν θέλουμε να κινηθούμε κατά γραμμές, βάζουμε πρώτα την επαναληπτική διαδικασία που αναφέρεται στις γραμμές (Για i από 1 μέχρι m), ενώ, όταν θέλουμε να κινηθούμε κατά στήλες βάζουμε πρώτα την επαναληπτική διαδικασία που αναφέρεται στις στήλες (Για j από 1 μέχρι n).

Υπολογισμός αθροίσματος - Μ.Ο. δισδιάστατου πίνακα

Στην περίπτωση του υπολογισμού αθροίσματος στοιχείων δισδιάστατου πίνακα, έχουμε να αντιμετωπίσουμε κάπως πιο πολύπλοκες καταστάσεις σε σχέση με τους μονοδιάστατους πίνακες. Ο υπολογισμός του ολικού αθροίσματος των στοιχείων του πίνακα γίνεται με την επέκταση της μεθόδου που χρησιμοποιήσαμε και στους μονοδιάστατους, απλά πρέπει να χρησιμοποιήσουμε δυο ΓΙΑ, και στην καρδιά του εμφωλευμένου βρόχου να βάλουμε την εντολή εκχώρησης που υπολογίζει το άθροισμα. Για την γενικευμένη περίπτωση ενός πίνακα table με m γραμμές και n στήλες, το κομμάτι του αλγόριθμου θα έχει ως εξής (σάρωση κατά γραμμές στη συγκεκριμένη περίπτωση):

```
sum←0
Για i από 1 μέχρι m
  Για j από 1 μέχρι n
        sum←sum+table(i,j)
  Τέλος_επανάληψης
Τέλος_επανάληψης
MO←sum/(m*n)
```

Πολλές φορές, όμως, θα κληθούμε να υπολογίσουμε μερικά αθροίσματα κατά γραμμές ή κατά στήλες. Αυτό σημαίνει ότι για έναν πίνακα table mxn, υπάρχουν m αθροίσματα κατά γραμμές και n αθροίσματα κατά στήλες.

Σε αυτό το σημείο έχουμε δύο επιλογές:

Αν χρειαζόμαστε τα αθροίσματα για περαιτέρω επεξεργασία που επιβάλλει να τα έχουμε αποθηκευμένα (π.χ. ταξινόμηση), τότε θα πρέπει να τα τοποθετήσουμε σε νέους πίνακες Για να τα αποθηκεύσουμε αυτά χρειαζόμαστε δύο νέους μονοδιάστατους πίνακες, έστω όπως τους ονομάζει το βιβλίο έναν row[i] m θέσεων (που θα "τρέχει" με το i) και έναν col[j] n θέσεων (που θα "τρέχει" με το j).
Παράδειγμα, για τον πίνακα Δ 12x4 θα πρέπει να δημιουργηθούν δύο πίνακες, ο row[ ] 12 θέσεων και ο col[ ] 4 θέσεων.

Εύρεση μερικών αθροισμάτων και ΜΟ γραμμών και αποθήκευση τους στον πίνακα row( ) – σάρωση κατά γραμμές	**Εύρεση μερικών αθροισμάτων γραμμών και αποθήκευση τους στον πίνακα row( ) με την αρχικοποίηση του row( ) και υπολογισμό του ΜΟ( ) στα όρια του πρώτου βρόχου**
Για i **από** 1 **μέχρι m** row[i] ← 0 **Τέλος_επανάληψης** **Για i από 1 μέχρι m** **Για j από 1 μέχρι n** row[i] ← row[i] + table [i,j] **Τέλος_επανάληψης** **Τέλος_επανάληψης** **Για i από 1 μέχρι m** MOrow[i]←row[i]/n **Τέλος_επανάληψης**	**Για** i **από** 1 **μέχρι m** row[i] ← 0 **Για j από 1 μέχρι n** row[i] ← row[i] + table [i,j] **Τέλος_επανάληψης** MOrow[i]←row[i]/n **Τέλος_επανάληψης**

Εύρεση μερικών αθροισμάτων στηλών και αποθήκευση τους στον πίνακα col( ) – σάρωση κατά γραμμές	**Εύρεση μερικών αθροισμάτων στηλών και αποθήκευση τους στον πίνακα col( ) με την αρχικοποίηση του col( ) στα όρια του πρώτου βρόχου – σάρωση κατά στήλες**
Για j **από** 1 **μέχρι n** col[j] ← 0 **Τέλος_επανάληψης** **Για i από 1 μέχρι m** **Για j από 1 μέχρι n** col[j] ← col[j] + table [i,j] **Τέλος_επανάληψης** **Τέλος_επανάληψης** **Για j από 1 μέχρι n** MOcol[j]←col[j]/m **Τέλος_επανάληψης**	**Για j από 1 μέχρι n** col[j] ← 0 **Για** i **από** 1 **μέχρι m** col[j] ← col[j] + table [i,j] **Τέλος_επανάληψης** MOcol[j]←col[j]/m **Τέλος_επανάληψης**

Πλήρης αλγόριθμος εύρεσης ολικού αθροίσματος, και μερικών αθροισμάτων κατά γραμμές και κατά στήλες	
sum ← 0 **Για i από 1 μέχρι m** row[i] ← 0 **Τέλος_επανάληψης** **Για j από 1 μέχρι n** col [j] ← 0 **Τέλος_επανάληψης** **Για i από 1 μέχρι m** **Για j από 1 μέχρι n** sum ← sum + table[i,j] row[i] ← row[i] + table [i,j] col[j] ← col[j] + table[i,j] **Τέλος_επανάληψης** **Τέλος_επανάληψης**	

Αλλιώς_αν χρειάζεται απλά να τα υπολογίσουμε και να τα εμφανίσουμε γιατί έτσι μας ζητείται από την εκφώνηση, χωρίς να τα χρειαζόμαστε σε επόμενο βήμα του αλγόριθμου, τότε δεν είναι απαραίτητη η αποθήκευση των αθροισμάτων σε ξεχωριστούς πίνακες, αλλά δουλεύουμε, όπως και με τις απλές μεταβλητές (αυτό το κάνουμε για να μην "φορτώνουμε" τον υπολογιστή με δέσμευση πολύτιμης μνήμης που συνεπάγεται η χρήση πινάκων):

Άθροισμα των στοιχείων της κάθε γραμμής του πίνακα table MxN	**Άθροισμα των στοιχείων της κάθε στήλης του πίνακα table MxN**
Αλγόριθμος άθροισμα_γραμμής Δεδομένα //table,M,N// Για i από 1 μέχρι M sum← 0 Για j από 1 μέχρι N sum← sum + table[i,j] Τέλος_επανάληψης Εμφάνισε 'Το άθροισμα της ', i, 'ης & γραμμής είναι: ', sum Τέλος_Επανάληψης Τέλος άθροισμα_γραμμής	Αλγόριθμος άθροισμα_στήλης Δεδομένα //table,M,N// Για j από 1 μέχρι N sum← 0 Για i από 1 μέχρι M sum← sum + table[i,j] Τέλος_επανάληψης Εμφάνισε 'Το άθροισμα της ', j, 'ης & στήλης είναι: ', sum Τέλος_Επανάληψης Τέλος άθροισμα_στήλης
Μέσος όρος των στοιχείων της κάθε γραμμής του πίνακα table MxN	**Μέσος όρος των στοιχείων της κάθε στήλης του πίνακα table MxN**
Αλγόριθμος Μέσος_γραμμής Δεδομένα //table,M,N// Για i από 1 μέχρι M sum←0 Για j από 1 μέχρι N sum←sum + table[i,j] Τέλος_επανάληψης MO←sum/N Εμφάνισε Ό μέσος όρος της', i, 'ης &γραμμής είναι:', MO Τέλος_Επανάληψης Τέλος Μέσος_γραμμής	Αλγόριθμος Μέσος_στήλης Δεδομένα //table,M,N// Για j από 1 μέχρι N sum←0 Για i από 1 μέχρι M sum←sum + table[i,j] Τέλος_επανάληψης MO←sum/M Εμφάνισε Ό μέσος όρος της', j, 'ης &στήλης είναι:', MO Τέλος_Επανάληψης Τέλος Μέσος_ στήλης

Μέγιστα και ελάχιστα δισδιάστατου πίνακα

Οι αλγόριθμοι και σε αυτήν την περίπτωση μοιάζουν πολύ αυτούς του υπολογισμού αθροισμάτων. Για την εύρεση ολικού μεγίστου και ελαχίστου δουλεύουμε με την κλασική ΑΝ μέσα στην καρδιά του διπλού εμφωλευμένου βρόχου:

Έτσι για έναν "γεμάτο" πίνακα table mxn, το κομμάτι αλγόριθμου που υπολογίζει το ολικό μέγιστο και ελάχιστο έχει ως εξής:

Με αρχικοποίηση των min, max στην πρώτη τιμή του πίνακα (ΠΡΟΤΕΙΝΕΤΑΙ)	**Με αρχικοποίηση των min, max σε μία πολύ μεγάλη και μία πολύ μικρή τιμή αντίστοιχα**

min←table[1,1] !ΑΡΧΙΚΟΠΟΙΗΣΗ MIN!
max←table[1,1] !ΑΡΧΙΚΟΠΟΙΗΣΗ MAX!
Για i από 1 μέχρι m
  Για j από 1 μέχρι n
  Αν table[i,j]<min τότε !ΕΥΡΕΣΗ MIN!
  min←table[i,j]
  Τέλος_αν !ΞΕΧΩΡΙΣΤΕΣ ΑΝ!
  Αν table[i,j]>max τότε !ΕΥΡΕΣΗ MAX!
  max←table[i,j]
  Τέλος_αν
  Τέλος_επανάληψης
Τέλος_επανάληψης
	min←2^31-1 !ΑΡΧΙΚΟΠΟΙΗΣΗ MIN!
	max←-2^31 !ΑΡΧΙΚΟΠΟΙΗΣΗ MAX!
	Για i από 1 μέχρι m
	  Για j από 1 μέχρι n
	  Αν table[i,j]<min τότε !ΕΥΡΕΣΗ MIN!
	  min←table[i,j]
	  Τέλος_αν !ΞΕΧΩΡΙΣΤΕΣ ΑΝ!
	  Αν table[i,j]>max τότε !ΕΥΡΕΣΗ MAX!
	  max←table[i,j]
	  Τέλος_αν
	  Τέλος_επανάληψης
	Τέλος_επανάληψης

Στην περίπτωση που μας ζητείται ο υπολογισμός μέγιστου και ελάχιστου κατά γραμμή ή κατά στήλη, έχουμε πάλι να αντιμετωπίσουμε το πρόβλημα, εάν θα πρέπει μόνο να τα υπολογίσουμε και να τα εμφανίσουμε ή αν θα πρέπει παράλληλα και να τα αποθηκεύσουμε για μελλοντική χρήση.

Μέγιστο και ελάχιστο στοιχείο της κάθε γραμμής του πίνακα με αρχικοποίηση στο πρώτο στοιχείο κάθε γραμμής– σάρωση κατά γραμμές	**Μέγιστο και ελάχιστο στοιχείο της κάθε γραμμής του πίνακα με αρχικοποίηση σε σταθερές τιμές και αποθήκευση τους σε νέους πίνακες maxrow( ) m θέσεων και minrow( ) m θέσεων**

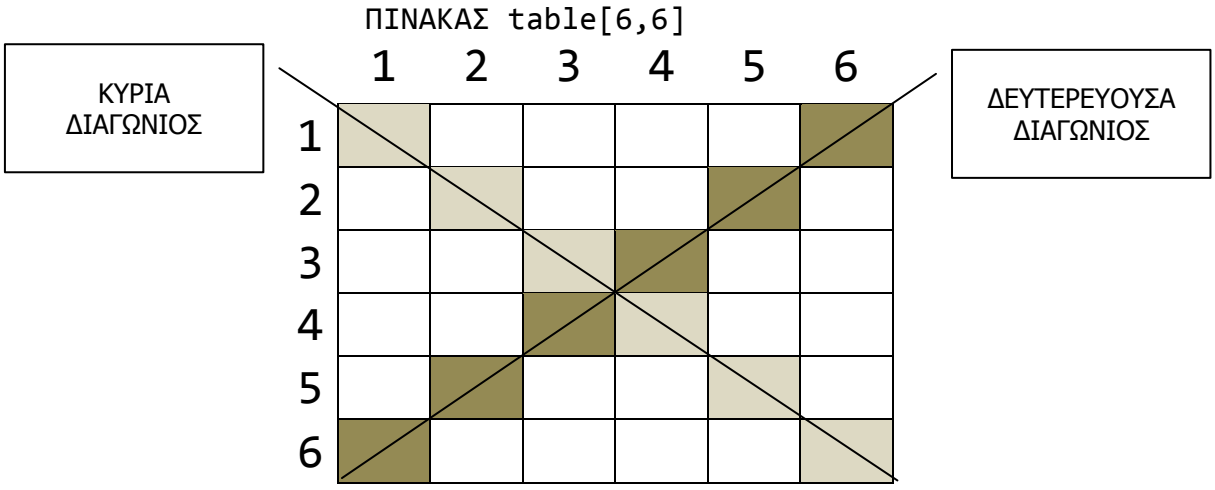
Αλγόριθμος Μέγιστο_Ελάχιστο_γραμμής
Δεδομένα //table, M, N//
Για i από 1 μέχρι M
  min←table[i,1]
  max←table[i,1]
  Για j από 1 μέχρι N
  Αν table[i,j] > max τότε
  max← table[i,j]
  Τέλος_αν
  Αν table[i,j] < min τότε
  min←table[i,j]
  Τέλος_αν
  Τέλος_επανάληψης
  Εμφάνισε 'Το μέγιστο της', i,' γραμμής είναι:', max
  Εμφάνισε 'Το ελάχιστο της', i,' γραμμής είναι:', min
Τέλος_Επανάληψης
Τέλος Μέγιστο_Ελάχιστο_γραμμής
	Αλγόριθμος Μέγιστο_Ελάχιστο_γραμμής
	Δεδομένα //table, M, N//
	Για i από 1 μέχρι M
	  minrow(i)←2^31-1 *!ή min←table[i,1]*
	  maxrow(i)←-2^31 *!ή max←table[i,1]*
	  Για j από 1 μέχρι N
	  Αν table[i,j] > maxrow(i) τότε
	  maxrow(i)← table[i,j]
	  Τέλος_αν
	  Αν table[i,j] < minrow(i) τότε
	  minrow(i)←table[i,j]
	  Τέλος_αν
	  Τέλος_επανάληψης
	Τέλος_Επανάληψης
	Αποτελέσματα //maxrow, minrow//
	Τέλος Μέγιστο_Ελάχιστο_γραμμής

Μέγιστο και ελάχιστο στοιχείο της κάθε στήλης του πίνακα με αρχικοποίηση σε σταθερές τιμές – σάρωση κατά στήλες	**Μέγιστο και ελάχιστο στοιχείο της κάθε στήλης του πίνακα με αρχικοποίηση στο πρώτο στοιχείο κάθε στήλης και αποθήκευση τους σε νέους πίνακες maxcol( ) n θέσεων και mincol ( ) n θέσεων**

Αλγόριθμος Μέγιστο_Ελάχιστο_στήλης
Δεδομένα //table, M, N//
Για j από 1 μέχρι N
  min←2^31-1
  max←-2^31
  Για i από 1 μέχρι M
  Αν table[i,j] > max τότε
  max← table[i,j]
  Τέλος_αν
  Αν table[i,j] < min τότε
  min←table[i,j]
  Τέλος_αν
  Τέλος_επανάληψης
  Εμφάνισε 'Το μέγιστο της', j,' στήλης είναι:', max
  Εμφάνισε 'Το ελάχιστο της', j,' στήλης είναι:', min
Τέλος_Επανάληψης
Τέλος Μέγιστο_Ελάχιστο_στήλης
	Αλγόριθμος Μέγιστο_Ελάχιστο_στήλης
	Δεδομένα //table, M, N//
	Για j από 1 μέχρι N
	  mincol(j)←table(1,j)
	  maxcol(j)←table(1,j)
	  Για i από 1 μέχρι M
	  Αν table[i,j] > maxcol(j) τότε
	  maxcol(j)← table[i,j]
	  Τέλος_αν
	  Αν table[i,j] < mincol(j) τότε
	  mincol(j)←table[i,j]
	  Τέλος_αν
	  Τέλος_επανάληψης
	Τέλος_Επανάληψης
	Αποτελέσματα //maxcol, mincol//
	Τέλος Μέγιστο_Ελάχιστο_γραμμής

ΤΕΤΡΑΓΩΝΙΚΟΙ ΠΙΝΑΚΕΣ

Οι τετραγωνικοί πίνακες είναι αυτοί που έχουν ίσο αριθμό γραμμών και στηλών. Στους τετραγωνικούς πίνακες δουλεύουμε όπως στους δισδιάστατους, με μοναδική προσθήκη την κύρια και την δευτερεύουσα διαγώνιο του πίνακα, οι οποίες ορίζονται μόνο στους τετραγωνικούς πίνακες.



Αν παρατηρήσουμε τους δείκτες των στοιχείων της κύριας διαγωνίου, ([1,1], [2,2], [3,3], [4,4], [5,5], [6,6]) θα καταλάβουμε ότι ένα στοιχείο ανήκει στην κύρια διαγώνιο, όταν ο δείκτης της γραμμής και της στήλης είναι ο ίδιος.Ι

Το table[i.j] ανήκει στην κύρια διαγώνιο αν i = j όπου table τετραγωνικός πίνακας NxN.

Αν κάνουμε το ίδιο για τη δευτερεύουσα διαγώνιο που τα στοιχεία της είναι στο παράδειγμα μας τα [1,6], [2,5], [3,4], [4,3], [5,2] και το [6,1], θα καταλάβουμε ότι το άθροισμα των δεικτών των στοιχείων της δευτερεύουσας διαγωνίου είναι κατά 1 μεγαλύτερο από τη διάσταση του πίνακα.

Το table[i,j] ανήκει στην δευτερεύουσα διαγώνιο αν i+j = N+1 όπου table τετραγωνικός πίνακας NxN.

Τα στοιχεία της κύριας διαγωνίου μπορούμε να πούμε ότι είναι τα table[i,i] όπου το i παίρνει τιμές από 1 μέχρι N για τον τετραγωνικό πίνακα table[N,N]

Τα στοιχεία της δευτερεύουσας διαγωνίου μπορούμε να πούμε ότι είναι τα table[i,N+1-i] όπου το i παίρνει τιμές από 1 μέχρι N για τον τετραγωνικό πίνακα table[N,N]

ΠΑΡΑΛΛΗΛΟΙ ΠΙΝΑΚΕΣ

Στους πίνακες έχουμε αδυναμία να αποθηκεύσουμε διαφορετικού τύπου δεδομένα. Παρόλα αυτά, πολλές φορές χρειάζεται να συσχετίζουμε π.χ. ονόματα (χαρακτήρες) με αριθμούς. Για να προσπεράσουμε αυτήν την αδυναμία των πινάκων, χρησιμοποιούμε του λεγόμενους παράλληλους πίνακες. Είναι πίνακες ακριβώς του ίδιου μεγέθους για τους οποίους υπάρχει αντιστοιχία των δεδομένων τους ανά θέση. Συνήθως τους χρησιμοποιούμε σε προβλήματα ταξινόμησης και αναζήτησης. Βλέπετε παραδείγματα.

ΘΕΜΑ 4 ΑΕΠΠ ΠΑΝΕΛΛΗΝΙΩΝ ΕΞΕΤΑΣΕΩΝ 2004

Μία εξαιρετική άσκηση που επεκτείνεται σε σχεδόν όλα όσα χρειαζόμαστε να ξέρουμε για τους πίνακες, ήταν το τέταρτο θέμα των εξετάσεων 2004. Ας δούμε κάποιους τρόπους επίλυσης:

ΘΕΜΑ 4ο
Για την πρώτη φάση της Ολυμπιάδας Πληροφορικής δήλωσαν συμμετοχή 500 μαθητές. Οι μαθητές διαγωνίζονται σε τρεις γραπτές εξετάσεις και βαθμολογούνται με ακέραιους βαθμούς στη βαθμολογική κλίμακα από 0 έως και 100.
Να γράψετε αλγόριθμο ο οποίος:
α. Να διαβάζει τα ονόματα των μαθητών και να τα αποθηκεύει σε μονοδιάστατο πίνακα. Μονάδες 2
β. Να διαβάζει τους τρεις βαθμούς που έλαβε κάθε μαθητής και να τους αποθηκεύει σε δισδιάστατο πίνακα. Μονάδες 2
γ. Να υπολογίζει το μέσο όρο των βαθμών του κάθε μαθητή. Μονάδες 4
δ. Να εκτυπώνει τα ονόματα των μαθητών και δίπλα τους το μέσο όρο των βαθμών τους ταξινομημένα με βάση τον μέσο όρο κατά φθίνουσα σειρά. Σε περίπτωση ισοβαθμίας η σειρά ταξινόμησης των ονομάτων να είναι αλφαβητική. Μονάδες 7
ε. Να υπολογίζει και να εκτυπώνει το πλήθος των μαθητών με το μεγαλύτερο μέσο όρο. Μονάδες 5

Παρατήρηση: Θεωρείστε ότι οι βαθμοί των μαθητών είναι μεταξύ του 0 και του 100 και ότι τα ονόματα των μαθητών είναι γραμμένα με μικρά γράμματα.

Το βασικό είναι να συνειδητοποιήσουμε (και να φτιάξουμε στο πρόχειρο μας) τους πίνακες που πρέπει να χρησιμοποιήσουμε. Διαβάζουμε καλά όλη την εκφώνηση για να πάρουμε μία συνολική εικόνα και μετά αντιμετωπίζουμε κάθε ερώτημα ξεχωριστά (άσχετα αν στην τελική επίλυση μπορούμε να συμπτύξουμε τον αλγόριθμο).

ερώτημα α) Έχουμε 500 μαθητές, άρα για να αποθηκεύσουμε τα ονόματα τους χρειαζόμαστε έναν μονοδιάστατο πίνακα 500 θέσεων τύπου χαρακτήρα. Έστω ότι τον ονομάζουμε ΟΝ( ).

```
Άρα:        Για i από 1 μέχρι 500
                Διάβασε ΟΝ[ i ]
            Τέλος_επανάληψης
```

ερώτημα β) Θέλουμε για κάθε έναν από τους 500 μαθητές να αποθηκεύσουμε 3 βαθμούς. Άρα, χρειαζόμαστε δισδιάστατο πίνακα 500x3 (όπου κάθε γραμμή αντιστοιχεί σε έναν μαθητή και κάθε στήλη σε ένα μάθημα). Ας ονομάζουμε αυτόν τον πίνακα Β(,).

```
Για i από 1 μέχρι 500
    Για j από 1 μέχρι 3
        Διάβασε Β [ i, j]
    Τέλος_επανάληψης
Τέλος_επανάληψης
```

ερώτημα γ) Θέλουμε τους ΜΟ κάθε μαθητή, άρα θέλουμε 500 μέσους όρους. Προφανώς ο κάθε ΜΟ προκύπτει από το άθροισμα των βαθμών κάθε μαθητή διά του 3. Άρα, έχουμε κλασική περίπτωση εύρεσης μερικών αθροισμάτων κατά γραμμές. Επειδή από το επόμενο ερώτημα μας ζητείται φθίνουσα ταξινόμηση ως προς τους ΜΟ, θα πρέπει να τους αποθηκεύσουμε σε έναν νέο πίνακα. Ας χρησιμοποιήσουμε την ονοματολογία του βιβλίου και ας ονομάσουμε τον πίνακα που θα αποθηκεύσουμε τα αθροίσματα row( ) και τον πίνακα των μέσων όρων ΜΟ( ). Και οι δύο έχουν 500 θέσεις, όσοι και οι μαθητές.

```
Για i από 1 μέχρι 500
  row[i] ← 0
      Για j από 1 μέχρι 3
          row[i]←row[i]+B[I,j]
      Τέλος_επανάληψης
  ΜΟ[i] ← ΜΟ[i] / 3
Τέλος_επανάληψης
```

ερώτημα δ) Εδώ έχουμε περίπτωση ταξινόμησης σε παράλληλους πίνακες. Πρέπει, αφενός να ταξινομήσουμε τον πίνακα ΜΟ( ) και ταυτόχρονα να αναδιατάξουμε τα στοιχεία του πίνακα ΟΝ( ), ώστε ο κάθε βαθμός να πηγαίνει ζευγάρι με τον αντίστοιχο μαθητή. Το δυσκολότερο σημείο είναι όταν έχουμε ισοβαθμία οπότε θα πρέπει τα στοιχεία του πίνακα ΟΝ( ) για τα οποία οι βαθμοί είναι ίδιοι να τα αλλάξουμε θέση ώστε να υπάρχει αλφαβητική σειρά στα ονόματα των ισοβαθμούντων .

```
Για i από 2 μέχρι 500
    Για j από 500 μέχρι i με βήμα -1
        Αν ΜΟ[j-1]<ΜΟ[j] τότε                  !Ταξινομούμε τον πίνακα ΜΟ
            αντιμετάθεσε ΜΟ[j-1], ΜΟ[j]
            αντιμετάθεσε ΟΝ[j-1], ΟΝ[j]        !Αναδιατάσσουμε τα στοιχεία του ΟΝ
        Αλλιώς_αν ΜΟ[j-1]= ΜΟ[j] τότε          !Ελέγχουμε την περίπτωση ισοβαθμίας του ΜΟ
            Αν ΟΝ[j-1]>ΟΝ[j] τότε               !Αν το j-1 όνομα είναι "μεγαλύτερο" του j-οστου
                αντιμετάθεσε ΟΝ[j-1], ΟΝ[j]     !τότε βάλε το j-οστο όνομα επάνω
            Τέλος_αν
        Τέλος_αν
    Τέλος_επανάληψης
Τέλος_επανάληψης
```

ερώτημα ε) Εδώ έχουμε περίπτωση αναζήτησης. Ξέρουμε ότι εφόσον ο ΜΟ( ) είναι ταξινομημένος κατά φθίνουσα σειρά, το ΜΟ(1) είναι το max. Για να μετρήσουμε το πλήθος των μαθητών που έχουν βαθμολογία ίση με το max πρέπει να χρησιμοποιήσουμε μια Αν και μία μεταβλητή pl. Μπορούμε, επομένως να γράψουμε:

```
pl←1
Για i από 2 μέχρι 500
        Αν ΜΟ[i]=ΜΟ[1] τότε
            pl←pl+1
        Τέλος_αν
Τέλος_επανάληψης
```

Η λύση αυτή, παρόλο που είναι σωστή, δεν είναι η καλύτερη δυνατή, κι αυτό γιατί θα αναγκάσουμε τον υπολογιστή να κάνει 499 επαναλήψεις όπως και να έχει. Μια πιο σωστή λύση είναι με ΟΣΟ:

```
pl←1
i←2
Οσο ΜΟ[i]=ΜΟ[1] ΚΑΙ i <=500 επανάλαβε
        pl←pl+1
        i←i+1
Τέλος_επανάληψης
```

Έτσι, θα σταματήσουν οι επαναλήψεις όταν βρεθεί βαθμός που είναι μικρότερος του ΜΟ[1]=max.
Θα μπορούσαμε να χρησιμοποιήσουμε και μία λογική μεταβλητή done που να γίνεται ψευδής όταν δεν υπάρχει άλλος μαθητής με βαθμολογία ίση με max=ΜΟ(1). Π.χ. θα μπορούσαμε να γράψουμε:

```
i←2; pl←1; done←αληθής
Όσο done=αληθής ΚΑΙ i<=500 επανάλαβε
  Αν ΜΟ[i]=ΜΟ[1] τότε
        pl←pl+1
        i←i+1
  Αλλιώς
        done←ψευδής
  Τέλος_αν
Τέλος_επανάληψης
```

⟺

```
i←2; pl←1; done←αληθής
Όσο done=αληθής ΚΑΙ i<=500 επανάλαβε
  Αν ΜΟ[i]<>ΜΟ[1] τότε
        done←ψευδής
  Αλλιώς
        pl←pl+1
        i←i+1
  Τέλος_αν
Τέλος_επανάληψης
```

ΣΗΜΑΝΤΙΚΟ: Αν κολλήσουμε σε κάποιο υποερώτημα, αυτό δε μας αποθαρρύνει και προχωράμε στο επόμενο (βαθμολογούνται ξεχωριστά). Αν όλα έχουν πάει καλά και έχουμε επιλύσει στο πρόχειρο τα υποερωτήματα του θέματος, προχωράμε σε έναν έλεγχο ορθότητας του αλγορίθμου. Χρησιμοποιώντας τυχαία δεδομένα, βλέπουμε αν ο αλγόριθμος μας λειτουργεί. Κάνουμε ένα τελευταίο ρετουσάρισμα, π.χ. βάζουμε Γράψε πριν τις Διάβασε και μεταφέρουμε τον αλγόριθμο στο καθαρό.

ΛΥΜΕΝΑ ΠΑΡΑΔΕΙΓΜΑΤΑ

Παράδειγμα 1 (Βασικές διεργασίες σε μονοδιάστατο πίνακα)
Σε ένα θερμοκήπιο ο υπάλληλος καταγράφει τις θερμοκρασίες του χώρου ανά μία ώρα.
Να γίνει αλγόριθμος ο οποίος: α) Να διαβάζει τις θερμοκρασίες για μία ημέρα. β) Να υπολογίζει τη μέγιστη και την ελάχιστη θερμοκρασία της ημέρας, γ) Να υπολογίζει τη μέση θερμοκρασία της ημέρας. δ) Να υπολογίζει πόσες θερμοκρασίες είναι μεγαλύτερες και πόσες μικρότερες από τη μέση θερμοκρασία.

Λύση
Επειδή τις τιμές που θα δώσει ο χρήστης θα πρέπει να τις επεξεργαστούμε πάνω από μία φορά, για να μη χρειαστεί να τις ξαναδώσει, θα τις αποθηκεύσουμε σε έναν πίνακα. Αν δεν υπήρχε το δ υποερώτημα, η χρήση πίνακα δεν θα ήταν αναγκαία.
Επομένως θα δημιουργήσουμε τον πίνακα ΘΕΡΜ, ο οποίος θα είναι μονοδιάστατος 24 θέσεων, αφού 24 είναι οι μετρήσεις κατά τη διάρκεια της ημέρας. Στην συνέχεια θα βρούμε το μέγιστο, το ελάχιστο και το μέσο όρο του πίνακα. Κατόπιν θα βρούμε πόσα στοιχεία του πίνακα είναι μεγαλύτερα και πόσα μικρότερα από τον μέσο όρο.

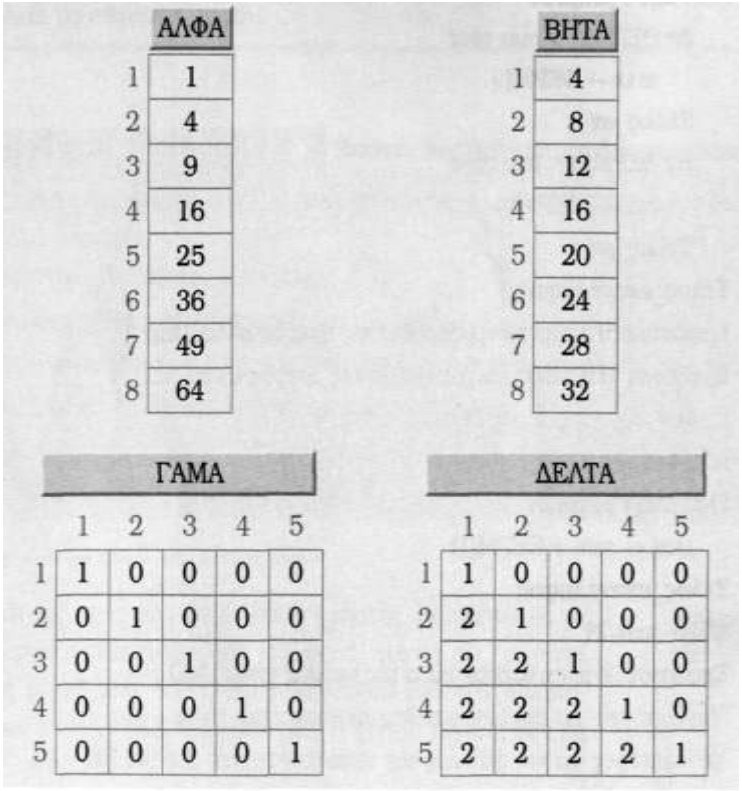
```
Αλγόριθμος Θερμοκήπιο
!Εισαγωγή δεδομένων
Για i από 1 μέχρι 24
        Εμφάνισε 'Δώστε την', i, 'θερμοκρασία'
        Διάβασε ΘΕΡΜ[i]
Τέλος_επανάληψης
!Υπολογισμός της μέγιστης και της ελάχιστης θερμοκρασίας
max ← ΘΕΡΜ[1]
min ← ΘΕΡΜ[1]
Για i από 1 μέχρι 24
        Αν ΘΕΡΜ[i] > max τότε
                max ← ΘΕΡΜ[i]
        Τέλος_αν
        Αν ΘΕΡΜ[i] < min τότε
                min← ΘΕΡΜ[i]
        Τέλος_αν
Τέλος_επανάληψης
Εμφάνισε 'Η μέγιστη θερμοκρασία της ημέρας είναι:', max
Εμφάνισε 'Η ελάχιστη θερμοκρασία της ημέρας είναι:', min
!Υπολογισμός της μέσης θερμοκρασίας
sum←0
Για i από 1 μέχρι 24
        sum ← sum + ΘΕΡΜ[i]
Τέλος_επανάληψης
ΜΟ ← sum/24
Εμφάνισε 'Η μέση θερμοκρασία της ημέρας είναι:', ΜΟ
!Υπολογισμός του πλήθους των θερμοκρασιών που είναι μεγαλύτερες και του πλήθους που είναι μικρότερες από τον ΜΟ
πάνω←0
κάτω←0
Για i από 1 μέχρι 24
        Αν ΘΕΡΜ[i] >ΜΟ τότε
                πάνω ← πάνω + 1
        αλλιώς_αν ΘΕΡΜ[i] < ΜΟ τότε
                κάτω ← κάτω + 1
Τέλος_αν
Τέλος_επανάληψης
Εμφάνισε 'Πάνω από την μέση θερμοκρασία είναι:', πάνω, 'τιμές'
Εμφάνισε 'Κάτω από την μέση θερμοκρασία είναι:', κάτω, 'τιμές'
Τέλος Θερμοκήπιο
```

	ΘΕΡΜ
1	
2	
3	
.	.
.	.
.	.
22	
23	
24	

ΣΥΜΒΟΥΛΗ: *Σε ασκήσεις με πίνακες πάντοτε να κάνετε έναν πρόχειρο πίνακα είτε με υποθετικά στοιχεία είτε χωρίς, ώστε να μπορείτε να κατανοείτε πιο εύκολα τη μορφή του πίνακα και τις ενέργειες που θέλετε να κάνετε σ' αυτόν.*

Παράδειγμα 2 (Δημιουργία πινάκων)

Να γίνει αλγόριθμος ο οποίος να δημιουργεί τους παρακάτω πίνακες (ένας για τον κάθε πίνακα).



	ΑΛΦΑ
1	1
2	4
3	9
4	16
5	25
6	36
7	49
8	64

	ΒΗΤΑ
1	4
2	8
3	12
4	16
5	20
6	24
7	28
8	32

ΓΑΜΑ	1	2	3	4	5
1	1	0	0	0	0
2	0	1	0	0	0
3	0	0	1	0	0
4	0	0	0	1	0
5	0	0	0	0	1

ΔΕΛΤΑ	1	2	3	4	5
1	1	0	0	0	0
2	2	1	0	0	0
3	2	2	1	0	0
4	2	2	2	1	0
5	2	2	2	2	1

Λύση

Σε κάθε έναν από τους πίνακες αυτούς θα πρέπει να βρούμε την σχέση βάσει της οποίας εισάγονται τα στοιχεία στον πίνακα.

ΑΛΦΑ: Ο πίνακας ΑΛΦΑ είναι μονοδιάστατος, 8 θέσεων, ο οποίος, αν παρατηρήσουμε, 8α δούμε ότι σε κάθε θέση του το στοιχείο που περιέχει είναι το τετράγωνο του δείκτη της θέσης αυτής.

Αλγόριθμος ΑΛΦΑ
Για i **από** 1 **μέχρι 8**
  ΑΛΦΑ[i] ← i*i
Τέλος επανάληψης
Τέλος ΑΛΦΑ

ΒΗΤΑ: Ο ΒΗΤΑ είναι και αυτός μονοδιάστατος, 8 θέσεων. Αν παρατηρήσουμε τα στοιχεία που περιέχει θα δούμε ότι αυτά είναι το τετραπλάσιο τον δείκτη της θέσης αυτής.

Αλγόριθμος ΒΗΤΑ
Για i από 1 μέχρι 8
  ΒΗΤΑ[i] ← 4*i
Τέλος_επανάληψης
Τέλος ΒΗΤΑ

ΓΑΜΑ: Ο πίνακας αυτός είναι ένας τετραγωνικός πίνακας, του οποίου όλα τα στοιχεία είναι μηδέν, εκτός από τα στοιχεία της κύριας διαγωνίου τα οποία είναι ίσα με 1.

```
Αλγόριθμος ΓΑΜΑ
Για i από 1 μέχρι 5
   Για j από 1 μέχρι 5
      Αν i = j τότε
            ΓΑΜΑ[i,j]← 1
      αλλιώς
            ΓΑΜΑ[i,j] ←0
      Τέλος_αν
   Τέλος_επανάληψης
Τέλος_επανάληψης
Τέλος ΓΑΜΑ
```

ΔΕΛΤΑ: Ο πίνακας αυτός είναι ένας τετραγωνικός πίνακας, του οποίου τα στοιχεία της κυρίας διαγωνίου είναι ίσα με 1, αυτά πού είναι πάνω από την κύρια διαγώνιο είναι ίσα με 0 και αυτά που είναι κάτω από την κύρια διαγώνιο είναι ίσα με 2.

```
  Αλγόριθμος ΔΕΛΤΑ
  Για i από 1 μέχρι 5
       Για j από 1 μέχρι 5
          Αν i = j τότε
               ΔΕΛΤΑ[i,j] ← 1
          αλλιώς_αν  i < j τότε
               ΔΕΛΤΑ[i,j] ← 0
           αλλιώς
              ΔΕΛΤΑ[i,j] ← 2 Τέλος_αν
       Τέλος_επανάληψης
Τέλος_επανάληψης
Τέλος ΔΕΛΤΑ
```

Παράδειγμα 3 (Εύρεση μεγίστου σε παράλληλους πίνακες)

Σε κάποιους αγώνες ακοντισμού, οι καλύτερες επιδόσεις των 8 αθλητών που παίρνουν μέρος καταχωρούνται στον πίνακα ΕΠΙΔ. Στον πίνακα NAME είναι καταχωρημένα τα ονόματα των αθλητών. Οι πίνακες αυτοί είναι φτιαγμένοι έτσι ώστε η επίδοση και το όνομα ενός αθλητή να βρίσκονται σε Θέσεις με τον ίδιο δείκτη στους δυο πίνακες.
Να γίνει αλγόριθμος που θα εμφανίζει το όνομα του αθλητή που παίρνει το χρυσό μετάλιο.

Λύση

Οι δύο πίνακες που χρησιμοποιούνται στον συγκεκριμένο αλγόριθμο είναι παράλληλοι. Αυτό που πρέπει να κάνουμε είναι να βρούμε την μέγιστη επίδοση και σε ποια θέση βρίσκεται. Άρα, θα βρούμε το μέγιστο στοιχείο του πίνακα ΕΠΙΔ και σε ποια θέση βρίσκεται, ώστε να εμφανίσουμε το όνομα του αθλητή που βρίσκεται στην ίδια θέση στον πίνακα NAME.


```
Αλγόριθμος Ακοντισμός
Δεδομένα //ΕΠΙΔ, NAME//
max ← ΕΠΙΔ[1]
θέσημax ← 1
Για i από 1 μέχρι 8
    Αν ΕΠΙΔ[i] > max τότε
            max ← ΕΠΙΔ[i]
            Θέσημax ← i
     Τέλος_αν
Τέλος_επανάληψης
Εμφάνισε 'Το χρυσό μετάλλιο παίρνει ο', NAME[θέσημax]
Τέλος Ακοντισμός
```

Παρατήρηση: Η αρχική τιμή της μεταβλητής θέσημax είναι υποχρεωτική γιατί, αν ο νικητής ήταν ο πρώτος αθλητής, τότε δεν θα έπαιρνε ποτέ καμία τιμή και έτσι ο αλγόριθμος σε αυτή την περίπτωση δεν θα δούλευε σωστά.

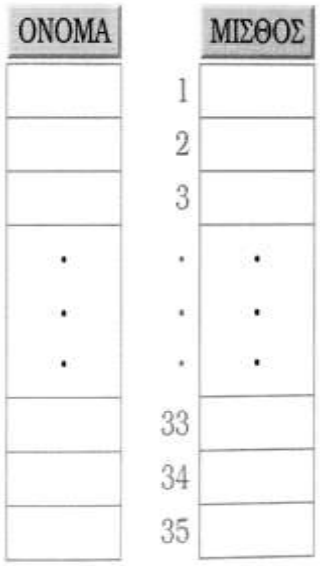
Παράδειγμα 4 (Αναζήτηση σε παράλληλους πίνακες)

Σε μια εταιρεία που απασχολεί 35 υπαλλήλους, για τη διευκόλυνση του ταμία έχουνε φτιάξει δυο πίνακες, τον πίνακα ΟΝΟΜΑ και τον πίνακα ΜΙΣΘΟΣ, οι οποίοι στις ίδιες τιμές δείκτη περιέχουν το όνομα και το μισθό αντίστοιχα ενός υπαλλήλου. Έτσι την ημέρα των πληρωμών, ο ταμίας πληκτρολογεί το όνομα του υπαλλήλου και του εμφανίζεται ο μισθός που παίρνει ο συγκεκριμένος υπάλληλος.
Να αναπτύξετε έναν αλγόριθμο με τον οποίο να μπορεί να δουλεύει το πρόγραμμα.

Λύση

Στον συγκεκριμένο αλγόριθμο θέλουμε να κάνουμε αναζήτηση σε παράλληλους πίνακες.

Θα κάνουμε αναζήτηση του ονόματος στον πίνακα ΟΝΟΜΑ και θα εμφανίσουμε το μισθό που βρίσκεται στον πίνακα ΜΙΣΘΟΣ στην ίδια θέση με αυτήν που βρήκαμε το όνομα. Άρα πρόκειται για έναν αλγόριθμο με απλή εφαρμογή της σειριακής αναζήτησης.




```
Αλγόριθμος Ταμείο
Δεδομένα //ΟΝΟΜΑ, ΜΙΣΘΟΣ//
Εμφάνισε 'Δώστε το όνομα του υπαλλήλου'
Διάβασε name
i←1
done ← ΨΕΥΔΗΣ
pos← 0
Όσο (i <= 35) ΚΑΙ (done = ΨΕΥΔΗΣ) επανάλαβε
    Αν ΟΝΟΜΑ[i] = name τότε
            done ← ΑΛΗΘΗΣ
             pos ← i
    αλλιώς
            i←i + 1
     Τέλος_αν
Τέλος_επανάληψης
Αν done = ΨΕΥΔΗΣ τότε
    Εμφάνισε 'Δεν υπάρχει υπάλληλος με τέτοιο όνομα'
αλλιώς
    Εμφάνισε 'Ο μισθός του υπαλλήλου είναι:', ΜΙΣΘΟΣ[pos]
Τέλος_αν
Τέλος Ταμείο
```

Παράδειγμα 5 (Ταξινόμηση σε παράλληλους πίνακες)

Στους σχολικούς αγώνες μιας περιφέρειας λαμβάνουν μέρος στους τελικούς των 100 μέτρων 16 παιδιά. Τα ονόματα των παιδιών είναι καταχωρημένα στον πίνακα NAME. Για την εύκολη επεξεργασία των αποτελεσμάτων οι διοργανωτές έφτιαξαν και τον πίνακα ΕΠΙΔ που περιέχει τις επιδόσεις των 16 παιδιών. Οι πίνακες NAME και ΕΠΙΔ είναι φτιαγμένοι έτσι ώστε το όνομα ενός παιδιού και η επίδοση του να βρίσκονται σε θέσεις με τους ίδιους δείκτες στους δυο πίνακες.

Να αναπτύξετε έναν αλγόριθμο ο οποίος να εμφανίζει το όνομα του παιδιού που παίρνει το χάλκινο μετάλλιο.

Λύση

Τα βασικά στοιχεία που πρέπει να προσέξουμε σε αυτό τον αλγόριθμο είναι:

1. Επειδή στο αγώνισμα των 100 μέτρων όσο μικρότερη είναι η επίδοση τόσο πιο καλή είναι, θα ψάξουμε να βρούμε την τρίτη μικρότερη επίδοση στον πίνακα ΕΠΙΔ.
2. Η εύρεση της μικρότερης επίδοσης είναι εύκολη. Όταν όμως θέλουμε να βρούμε την τρίτη μικρότερη, θα πρέπει να ταξινομήσουμε τον πίνακα ΕΠΙΔ κατά αύξουσα σειρά, ώστε η τρίτη καλύτερη επίδοση να βρεθεί στην τρίτη θέση του πίνακα.
3. Ο πίνακας ΕΠΙΔ που θέλουμε να ταξινομήσουμε είναι παράλληλος πίνακας με τον πίνακα NAME. Επομένως, για να μην μπερδέψουμε την αντιστοίχιση που υπάρχει μεταξύ των δύο πινάκων, θα πρέπει κάθε φορά που κάνουμε μια αλλαγή στον πίνακα ΕΠΙΔ που θέλουμε να ταξινομήσουμε, την ίδια αλλαγή να την κάνουμε και στον πίνακα NAME. Έτσι, για να εμφανίσουμε το όνομα του παιδιού που παίρνει το χάλκινο μετάλλιο, αρκεί να εμφανίσουμε το όνομα που θα βρίσκεται στην τρίτη δέση του πίνακα NAME, μετά την ταξινόμηση.

Συνεπώς, με βάση τα παραπάνω, ο αλγόριθμος είναι εφαρμογή της φυσαλίδας σε παράλληλους πίνακες και έχει ως εξής:

```
Αλγόριθμος Αγώνες
Δεδομένα //ΕΠΙΔ, NAME//
Για i από 2 μέχρι 16
      Για j από 16 μέχρι i με_βήμα -1
         Αν ΕΠΙΔ[j-1] > ΕΠΙΔ[j] τότε
             !Αντιμετάθεση των στοιχείων του ΕΠΙΔ
            temp1 ← ΕΠΙΔ[j]
            ΕΠΙΔ[j] ← ΕΠΙΔ[j-1]
            ΕΠΙΔ[j-1] ← temp1
             !Αντιμετάθεση των στοιχείων του NAME
             temp2 ← NAME[j]
             NAME [j] ← NAME [j-1]
             NAME[j-l] ← temp2
         Τέλος_αν
      Τέλος_επανάληψης
Τέλος_επανάληψης
Εμφάνισε Το χάλκινο μετάλλιο παίρνει ο:', NAME[3]
Τέλος Αγώνες
```

	NAME
1	
2	
3	
.	.
.	.
.	.
14	
15	
16	

	ΕΠΙΔ
1	
2	
3	
.	.
.	.
.	.
14	
15	
16	

ΑΝΑΦΟΡΕΣ - ΒΙΒΛΙΟΓΡΑΦΙΑ

1. "Ανάπτυξη Εφαρμογών Σε Προγραμματιστικό Περιβάλλον", Α. Βακάλη, Η. Γιαννόπουλου, Ν. Ιωαννίδη, Χ. Κοιλιά, Κ. Μάλαμα, Ι. Μανωλόπουλου, Π. Πολίτη, ΟΕΔΒ
2. "Ανάπτυξη Εφαρμογών Σε Προγραμματιστικό Περιβάλλον Γ' Ενιαίου Λυκείου", Α. Βερνάρδου, Λ. Μπότσαρη, Εκδόσεις Ζήτη