# Lesson 14 – Arrays

* One dimensional Arrays
* Basic Operations with arrays
* MAX – MIN item
* Linear search
* Binary search
* Sorting with Bubble Sort
* Sorting with Selection Sort

What students should know

**4h**

Ένα μεγάλο πρόβλημα στον προγραμματισμό είναι η διαχείριση πολλών δεδομένων. Όταν ένα πρόβλημα αποτελείται από ελάχιστα δεδομένα συνήθως 6-7 μεταβλητές είναι εύκολο να δηλωθούν και να χρησιμοποιηθούν. Τι θα συμβεί όταν υπάρχει ανάγκη χρήσης πολλών ομοειδών δεδομένων ταυτόχρονα; Για παράδειγμα 100 μαθητές και 100 βαθμοί πως μπορούν να δηλωθούν διακόσιες μεταβλητές με ονόματα και βαθμούς και πως μπορεί ένας προγραμματιστής να διαχειριστεί τόσα δεδομένα;

Μία από τις λύσεις που παρέχει η επιστήμη των υπολογιστών στο παραπάνω πρόβλημα είναι η χρήση των πινάκων. Γενικά με τον όρο πίνακα ορίζεται ένα σύνολο από δεδομένα ίδιου τύπου τα οποία τοποθετούνται το ένα μετά το άλλο στην μνήμη του υπολογιστή. Έτσι ο προγραμματιστής μπορεί να τα εντοπίζει απλά μετακινούμενος από τη μία θέση στην άλλη χωρίς να έχει ορίσει ξεχωριστά ονόματα.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***Grades*** | | 98 | 76 | 86 | 45 | 32 | 77 | 56 | 99 | 34 | 71 | 47 | 82 | 69 | 88 |
|  |  | 0 |  |  |  |  |  |  |  |  |  |  |  |  | 13 |

Picture 1 Πίνακας με όνομα “Grades” και 14 θέσεις για βαθμούς

Στην Picture 1 μπορείτε να δείτε ένα παράδειγμα πίνακα με 14 βαθμούς μαθητών σε ένα μάθημα. Όλοι οι βαθμοί είναι τοποθετημένοι μέσα σε συνεχόμενες θέσεις και υπάρχει μόνο ένα όνομα Grades. Προκειμένου να αναφερθεί ο προγραμματιστής σε θέσεις μνήμης ενός πίνακα αρκεί να αναφέρει το όνομα του και στη συνέχει να γράψει , μέσα σε παρένθεση, τον αριθμό του κελιού στον οποίο βρίσκεται το δεδομένο που χρειάζεται. Έτσι για να εμφανίσει π.χ. το πρώτο στοιχείο του πίνακα αρκεί να γράψει “Grades(0)” όπου 0 είναι η πρώτη θέση του πίνακα.

### Δήλωση πίνακα

Η δήλωση ενός πίνακα γίνεται όπως οι υπόλοιπες μεταβλητές.

**Private** Grades(14) **As** **Int**

Όπου Grades είναι το όνομα του πίνακα και ο αριθμός στην παρένθεση αντιπροσωπεύει το πλήθος των θέσεων του πίνακα. Από τη στιγμή που ένας πίνακας θα δηλωθεί ως κάποιας διάστασης δεν μπορεί να αλλάξει μέσα στον κώδικα εκτός και αν δηλωθεί ξανά με νέα διάσταση.

## Λειτουργίες σε πίνακες

### Εισαγωγή στοιχείων σε πίνακα.

Για να εισάγετε ένα στοιχείο σε πίνακα αρκεί να αναθέσετε στο αντίστοιχο στοιχείο του πίνακα μία τιμή. Π.χ.

Grades(0) = 89

Η διαδικασία μπορεί να συνεχιστεί με το ίδιο τρόπο αλλά είναι πιο εύκολο να δημιουργήσετε μια επαναληπτική διαδικασία προκειμένου να «γεμίσει» τον πίνακα. Στην περίπτωση του πίνακα Grades στην B4X θα μπορούσε να είναι:

**Private** Grades(14) **As** **Int**

**For** i = 0 **To** 13

Grades(i) = Rnd(1,100)

**Next**

Ο παραπάνω κώδικας γεμίζει τον πίνακα με τυχαίους αριθμούς από το 1 έως το 100. Παρατηρήστε ότι η μέτρηση των θέσεων ξεκινά από τον αριθμό 0. Η μεταβλητή **i** η οποία προσδιορίζει κάθε φορά την θέση του πίνακα χρησιμοποιούμε ονομάζεται **δείκτης** του πίνακα. Μετακινώντας τον δείκτη i με μια επανάληψη μπορείτε να έχετε πρόσβαση σε κάθε θέση του πίνακα.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***Grades*** | | 89 |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | | 🡹 |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | i |  |  |  |  |  |  |  |  |  |  |  |  | 13 |

Picture 2 Εισαγωγή στοιχείων σε πίνακα

Ένας δεύτερος τρόπος εισαγωγής στοιχείων σε πίνακα είναι ο παρακάτω:

**Private** Grades() **As Int**

Grades **= Array As Int**(19,43,12,65,23,87,45,65,87,23,56)

Με το τρόπο αυτό το μέγεθος του πίνακα δεν καθορίζεται στη δήλωση του αλλά κατά τη διάρκεια γεμίσματος με την εισαγωγή των στοιχείων με την εντολή **Array.**

Είναι προφανές ότι μπορείτε να έχετε πίνακες οποιοδήποτε τύπου για παράδειγμα strings, floats κλπ αλλά πάντα χωρίς να αναμειγνύονται οι τύποι μεταξύ τους στον ίδιο πίνακα.

### Εμφάνιση στοιχείων σε πίνακα.

Χρησιμοποιώντας την εντολή Log εύκολα μπορείτε να τυπώσετε ένα στοιχείο ενός πίνακα ή ολόκληρο τον πίνακα χρησιμοποιώντας μία επανάληψη.

**Log**(Grades(0)) ‘ Shows the 1st item of Array Grades

**For** i = 0 **to** 13

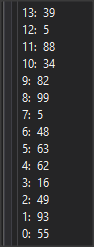
**Log**(i & “: “ & Grades(i))

**Next**

Στο παραπάνω παράδειγμα ξεκινά μια επανάληψη η οποία χρησιμοποιεί ένα δείκτη i για να εμφανίσει την τρέχουσα θέση του πίνακα.

Η προσπάθεια χρήσης δείκτη εκτός ορίων πίνακα οδηγεί σε κατάρρευση του προγράμματος και επομένως πρέπει να δίνετε ιδιαίτερη προσοχή στη χρήση των δεικτών και των ορίων του πίνακα.

### Εμφάνιση στοιχείων με ανάποδη σειρά

Ο παρακάτω κώδικας εμφανίζει τα στοιχεία του πίνακα από το τέλος προς την αρχή:

**For** i = 13 **to** 0 **Step** -1

**Log**(i & “: “ & Grades(i))

**Next**

Γενικά μπορείτε να μετακινήσετε όπως θέλετε τον δείκτη σας μέσα σε μια επανάληψη για να εμφανίσετε ή να χρησιμοποιήσετε όποια στοιχεία του πίνακα θέλετε.

### Εύρεση Αθροίσματος και Μέσου όρου στοιχείων πίνακα

Οι κανόνες που ισχύουν στις επαναληπτικές διαδικασίες για αλγοριθμικές τεχνικές ισχύουν γενικά και στους πίνακες με λίγες διαφοροποιήσεις. Έτσι για άθροισμα των στοιχείων απαιτείται μία μεταβλητή επιπλέον που θα κρατάει το άθροισμα συνήθως την ονομάζουμε “sum” και μια επανάληψη στο πίνακα.

**Private** intSum **As Int** = 0

**Private fltAverage as Float**

**For** i = 0 **to** 13

intSum **=** intSum **+** Grades(i)

**Next**

fltAverage = intSum / 14

**Log**(intSum & fltAverage)

### Εύρεση Μέγιστου και Ελάχιστου

Η εύρεση του μέγιστου ή του ελάχιστου στοιχείου ενός πίνακα κάνει χρήση μιας επιπλέον μεταβλητής που συνήθως ονομάζεται Max ή Min αντίστοιχα. Στη μεταβλητή αυτή αρχικά αναθέτει ο προγραμματιστής την πρώτη τιμή του πίνακα και στη συνέχεια ελέγχει όλες τις υπόλοιπες τιμές με το Max (ή Min). Αν βρεθεί στοιχείο μεγαλύτερο του Max (ή μικρότερο του Min) τότε αντικαθίσταται το Max (ή το Min) με το νέο στοιχείο.

**Private** intMax, intMin **As Int**

intMax = Grades(0)

intMin = Grades(0)

**For** i = 0 To 13

**If** intMax < Grades(i) **Then**

intMax = Grades(i)

**End** **If**

**If** intMin > Grades(i) **Then**

intMin = Grades(i)

**End** **If**

**Next**

**Log**("Max = " & intMax)

**Log**("Min = " & intMin)

## Αλγόριθμοι Αναζήτησης

Η αναζήτηση ενός στοιχείου σε πίνακα αναφέρεται στην σάρωση ενός πίνακα προκειμένου να βρούμε κάποιο στοιχείο που ικανοποιεί κάποια συνθήκη.

Από τους αλγόριθμους αναζήτησης θα δούμε τη σειριακή και τη δυαδική.

### Σειριακή Αναζήτηση

Είναι ο πιο εύκολος αλλά και ο πιο αργός τρόπος αναζήτησης. Αφορά την σάρωση όλων των στοιχείων ενός πίνακα προκειμένου να βρεθεί το στοιχείο που αναζητείται. Στο παρακάτω κώδικα εμφανίζονται οι θέσεις του πίνακα που περιέχουν την τιμή key.

'Find all positions with key value

**For** i = 0 **To** 999

**If** Grades(i) = key **Then**

**Log**("found in : " & i & "position")

**End If**

**Next**

Όταν απαιτείται να βρεθεί η πρώτη θέση που εμφανίζεται μία τιμή θα πρέπει να δηλωθεί μία λογική μεταβλητή (found) της οποίας την τιμή θα αντιστρέψουμε αν βρεθεί το στοιχείο.

**Private** found **As Boolean** = False

i = 0

**Do While Not** (found) **And** i <= 999

**If** Grades(i) = key **Then**

**Log**("found in : " & i & "position")

found = True

**End If**

i = i + 1

**Loop**

**If Not(**found) **then**

**Log**(“Not found”)

**End If**

### Δυαδική Αναζήτηση

Η δυαδική Αναζήτηση εφαρμόζεται μόνο σε ταξινομημένους πίνακες. Η βασική φιλοσοφία της μεθόδου αφορά τον έλεγχο του μεσαίου στοιχείου. Αν το στοιχείο που αναζητούμε είναι μικρότερο από το μεσαίο τότε συνεχίζεται η αναζήτηση στον πάνω μισό πίνακα διαφορετικά στον κάτω μισό. Στο παρακάτω παράδειγμα υπάρχει πίνακας Grades με 10 τιμές ταξινομημένες κατά αύξουσα σειρά.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Binary Search** | | | | | | | | | |
| key value = 80 | | | | | | | | | |
| *1* | 47 | 🡨Up | 47 |  | 47 |  | 47 |  |  |
| *2* | 58 |  | 58 |  | 58 |  | 58 |  |  |
| *3* | 62 |  | 62 |  | 62 |  | 62 |  |  |
| *4* | 69 |  | 69 |  | 69 |  | 69 |  |  |
| *5* | 74 | 🡨Cent | 74 |  | 74 |  | 74 |  |  |
| *6* | 79 |  | 79 | 🡨Up | 79 | 🡨Up, Cent | 79 |  |  |
| *7* | 80 |  | 80 |  | 80 | 🡨Bot | 80 | 🡨Up, Cent, Bot |  |
| *8* | 83 |  | 83 | 🡨Cent | 83 |  | 83 |  |  |
| *9* | 88 |  | 88 |  | 88 |  | 88 |  |  |
| *10* | 95 | 🡨Bot | 95 | 🡨Bot | 95 |  | 95 |  |  |
|  | 1 |  | 2 |  | 3 |  | 4 |  |  |

1. Αρχικά το πρώτο και τελευταίο κελί του πίνακα καταχωρούνται στις μεταβλητές Up και Bot αντίστοιχα. Στη συνέχεια προσδιορίζεται το κέντρο του πίνακα ως το πηλίκο της ακέραιας διαίρεσης (Up+Bot)/2.
2. Ελέγχεται η τιμής της θέσης Cent με το key και αν αυτό είναι μικρότερο τότε μεταφέρεται το Bot πάνω από το Cent. Σε αντίθετη περίπτωση μεταφέρεται το Up κάτω από το Cent. Προσδιορίζεται το νέο Cent με τον ίδιο τρόπο όπως και στο πρώτο βήμα.
3. Επαναλαμβάνονται τα παραπάνω βήματα μέχρι να βρεθεί το στοιχείο ή η θέση Up να γίνει μεγαλύτερη από την Bot.

### Ταξινόμηση

Υπάρχουν αρκετοί αλγόριθμοι ταξινόμησης ενός πίνακα που μπορείτε να χρησιμοποιήσετε. Ενδεικτικά αναφέρονται η ταξινόμηση φυσαλίδας (Bubble Sort) και η ταξινόμηση με επιλογή (Selection Sort).

#### Η ταξινόμηση φυσαλίδας

Η ταξινόμηση φυσαλίδα χρησιμοποιεί μία τεχνική περασμάτων προκειμένου κάθε φορά να μεταφέρει το μικρότερο ή μεγαλύτερο στοιχείο ενός πίνακα στην κορυφή.

Παράδειγμα Bubble Sort

Δίνεται ο παρακάτω πίνακας με όνομα Grades με 5 βαθμούς ακεραίων.

**Private** Grades() **As Int**

Grades = Array As Int(65,12,19,43,23)

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1st Pass | | | | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1 | 65 |  | 65 |  | 65 |  | 65 |  | 65 | 🡨 k-1 | 12 |  |
| 2 | 12 |  | 12 |  | 12 |  | 12 | 🡨k-1 | 12 | 🡨k | 65 |  |
| 3 | 19 |  | 19 |  | 19 | 🡨k-1 | 19 | 🡨k | 19 |  | 19 |  |
| 4 | 43 |  | 43 | 🡨k-1 | 23 | 🡨k | 23 |  | 23 |  | 23 |  |
| 5 | 23 | 🡨 k | 23 | 🡨 k | 43 |  | 43 |  | 43 |  | 43 |  |
|  |  |  | 1 |  | 2 |  | 3 |  | 4 |  |  |  |

Αρχικά ο αλγόριθμος ξεκινά από την τελευταία θέση του πίνακα και συγκρίνει διαδοχικά με την προηγούμενη

1. Η πρώτη σύγκριση γίνεται με τις τιμές των θέσεων 5, 4 όπου το στοιχείο του Grades(5) είναι μικρότερο του Grades(4) και έτσι τα δύο κελιά αλλάζουν μεταξύ τους τιμές (swap).
2. Στο επόμενο βήμα συγκρίνονται οι θέσεις 4,3 όπου το στοιχείο Grades(4) δεν είναι μικρότερο του Grades(3) και δεν αλλάζει κάτι στον πίνακα μας.
3. Για τις θέσεις 3,2 η 3η θέση δεν είναι μικρότερη της 2ης και δεν αλλάζουν οι τιμές στον πίνακα.
4. Και στο τελευταίο βήμα συγκρίνεται η 2η με την 1η θέση και προκύπτει πάλι αντιμετάθεση οπότε και ολοκληρώνεται η διαδικασία.

Το πρώτο πέρασμα υλοποιείται με την παρακάτω επανάληψη.

**Private** Grades() **As** Int

**Private** temp **As** **Int**

Grades = **Array** **As** **Int**(19,43,12,65,23)

**For** k = 4 **To** 1 **Step** -1

**If** Grades(k) **<** Grades(k-1) **Then**

temp **=** Grades(k)

Grades(k) **=** Grades(k-1)

Grades(k-1) **=** temp

**End If**

**Next**

Στο του κώδικα έχει μεταφερθεί το μικρότερο στοιχείο στην κορυφή του πίνακα.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2nd Pass | | | | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1 | 12 |  | 12 |  | 12 |  | 12 |  | 12 |  |  |  |
| 2 | 65 |  | 65 |  | 65 |  | 65 | 🡨k-1 | 19 |  |  |  |
| 3 | 19 |  | 19 |  | 19 | 🡨k-1 | 19 | 🡨k | 65 |  |  |  |
| 4 | 23 |  | 23 | 🡨k-1 | 23 | 🡨k | 23 |  | 23 |  |  |  |
| 5 | 43 | 🡨 k | 43 | 🡨 k | 43 |  | 43 |  | 43 |  |  |  |
|  |  |  | 1 |  | 2 |  | 3 |  |  |  |  |  |

Στο δεύτερο πέρασμα του πίνακα εκτελείται η ίδια διαδικασία με τη διαφορά ότι οι έλεγχοι μεταξύ των κελιών καταλήγουν μέχρι την 2η θέση αφού ήδη στην πρώτη έχει ανέβει το μικρότερο στοιχείο.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 3rd Pass | | | | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1 | 12 |  | 12 |  | 12 |  | 12 |  |  |  |  |  |
| 2 | 19 |  | 19 |  | 19 |  | 19 |  |  |  |  |  |
| 3 | 65 |  | 65 |  | 65 | 🡨k-1 | 23 |  |  |  |  |  |
| 4 | 23 |  | 23 | 🡨k-1 | 23 | 🡨k | 54 |  |  |  |  |  |
| 5 | 43 | 🡨 k | 43 | 🡨 k | 43 |  | 43 |  |  |  |  |  |
|  |  |  | 1 |  | 2 |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 4rth Pass | | | | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1 | 12 |  | 12 |  | 12 |  |  |  |  |  |  |  |
| 2 | 19 |  | 19 |  | 19 |  |  |  |  |  |  |  |
| 3 | 23 |  | 23 |  | 23 |  |  |  |  |  |  |  |
| 4 | 54 |  | 54 | 🡨k-1 | 43 |  |  |  |  |  |  |  |
| 5 | 43 | 🡨 k | 43 | 🡨 k | 54 |  |  |  |  |  |  |  |
|  |  |  | 1 |  |  |  |  |  |  |  |  |  |

Τα διαδοχικά περάσματα συνεχίζονται μεχρί να ολοκληρωθεί η ταξινόμηση του πίνακα. Παρατηρήστε ότι κάθε φορά ελέγχονται λιγότερες θέσεις αφού σε κάθε πέρασμα το μικρότερο ανεβαίνει στην επιφάνεια (bubble).

Γενικότερα τα περάσματα αυτά είναι όσο το μέγεθος του πίνακα -1. Στο παράδειγμα για πίνακα 5 θέσεων πραγματοποιήθηκαν 4 περάσματα. Ο ολοκληρωμένος κώδικας είναι ο παρακάτω:

**Private** Grades() **As** Int

**Private** temp **As** **Int**

Grades = **Array** **As** **Int**(19,43,12,65,23)

**For** i = 1 **to** 4 ‘i counts the different pass

**For** k = 4 **To** i **Step** -1

**If** Grades(k) **<** Grades(k-1) **Then**

temp **=** Grades(k)

Grades(k) **=** Grades(k-1)

Grades(k-1) **=** temp

**End If**

**Next**

**Next**

#### Η ταξινόμηση με επιλογή

Η ταξινόμηση με επιλογή χρησιμοποιεί την εύρεση του ελάχιστου στοιχείου του πίνακα το οποίο και τοποθετεί στην κορυφή του. Υλοποιείται σύμφωνα με τα παρακάτω βήματα

1. Επιλογή του ελάχιστου στοιχείου
2. Ανταλλαγή του ελάχιστου με το πρώτο στοιχείο
3. Επανάληψη των βημάτων 1 και 2 για τα υπόλοιπα στοιχεία του πίνακα

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1 | 65 |  | 65 | swap | 12 |  | 12 |  | 12 |  | 12 |  | 12 |  | 12 |  |
| 2 | 12 | 🡨min | 12 | 65 |  | 65 | swap | 19 |  | 19 |  | 19 |  | 19 |  |
| 3 | 19 |  | 19 |  | 19 | 🡨min | 19 | 65 |  | 65 | swap | 23 |  | 23 |  |
| 4 | 43 |  | 43 |  | 23 |  | 23 |  | 23 | 🡨min | 23 | 65 |  | 65 | swap |
| 5 | 23 |  | 23 |  | 43 |  | 43 |  | 43 |  | 43 |  | 43 | 🡨min | 43 |
|  | 1 |  | 2 |  | 1 |  | 2 |  | 1 |  | 2 |  | 1 |  | 2 |  |

**Private** Grades() **As Int**

Grades = **Array As Int**(65,12,19,43,23)

**Private** intMin, intMinPos **As Int**

**For** k = 0 **To** 4

intMin = Grades(k)

intMinPos = k

**For** i = k **To** 4 ‘Find the minimum from position k to 5th

**If** intMin > Grades(i) **Then**

intMin = Grades(i)

intMinPos = i

**End If**

**Next**

Grades(intMinPos) = Grades(k)

Grades(k) = intMin

**Next**

## Ασκήσεις

1. Να κατασκευάσετε πρόγραμμα που θα γεμίζει ένα πίνακα με όνομα **A** 50 θέσεων με τυχαίους ακέραιους αριθμούς από το 1 έως 100.
2. Υπολόγισε και εμφάνισε το άθροισμα των στοιχείων του πίνακα A που βρίσκονται σε ζυγές θέσεις
3. Αν το άθροισμα των 25 πρώτων θέσεων του πίνακα είναι ίσο με το άθροισμα των 25 τελευταίων στοιχείων εμφάνισε το μήνυμα «Ίσα αθροίσματα».
4. Αν A(1)=A(50), A(2)=A(49), A(3)=A(48)… A(25)=A(26), τότε εμφάνισε το μήνυμα «Πίνακας συμμετρικός»
5. Βρείτε ποιο είναι το μέγιστο στοιχείο καθώς και τις θέσεις του πίνακα που αυτό εντοπίζεται.
6. Να δημιουργήσετε υποπρόγραμμα που να ταξινομεί τον πίνακα Α
7. Να δημιουργήσετε υποπρόγραμμα που να δέχεται ένα πίνακα και ένα ακέραιο αριθμό και να εφαρμόζει δυαδική αναζήτηση για τον αριθμό αυτό στον πίνακα. Τέλος να τη θέση που βρέθηκε ο αριθμός ή 0.
8. Με τη βοήθεια των δύο προηγούμενων υποπρογραμμάτων να κάνετε ταξινόμηση στον Πίνακα Α και στη συνέχεια να αναζητήσετε το στοιχείο 67 και να εμφανίσετε κατάλληλα μηνύματα αν εντοπίστηκε ή όχι.
9. Η μέση θερμοκρασία ανά μέρα για ένα έτος αποθηκεύονται σε ένα πίνακα **Temperature**(365). Αν θεωρήσετε ότι οι θερμοκρασίες αυτές είναι ακέραιες τιμές μεταξύ 1 και 40 oC να βρεθεί και να εμφανιστεί η συχνότητα εμφάνισης κάθε θερμοκρασίας.
10. Στον προηγούμενο πίνακα **Temperature** να βρεθεί η δεύτερη μεγαλύτερη θερμοκρασία του έτους.