Πίνακες

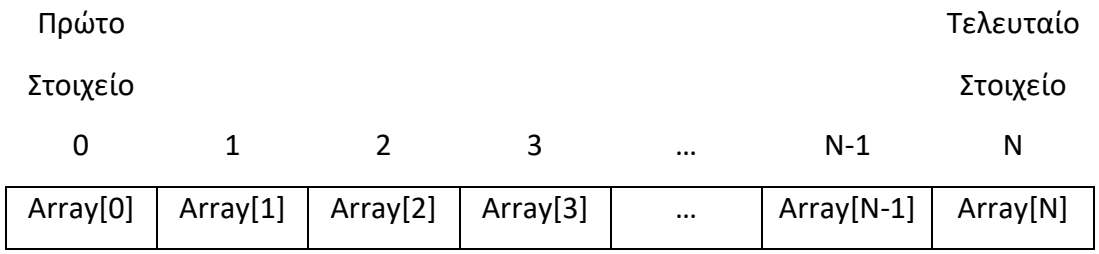
1.1. Μονοδιάστατοι-Πολυδιάστατοι Πίνακες

Οι πίνακες εκφράζουν μια μεταβλητή που έχει περισσότερες από μια θέσεις, δηλαδή μια μεταβλητή πολλαπλών θέσεων. Για το λόγο αυτό όταν ορίζουμε ένα πίνακα, χρησιμοποιούμε το όνομα της μεταβλητής και το δείκτη, που είναι ένας αριθμός και εκφράζει τη θέση του κάθε δεδομένου. Ένας πίνακας μπορεί να είναι μονοδιάστατος ή μεγαλύτερης διάστασης.

Ο πίνακας είναι ένα σύνολο δεδομένων ίδιου τύπου. Αποθηκεύει μια διαδοχική συλλογή στοιχείων του ίδιου τύπου σταθερού μεγέθους. Κάθε ένα από τα δεδομένα του πίνακα λέγεται στοιχείο του πίνακα. Όταν αναφερόμαστε σε ένα δεδομένο από τον πίνακα τότε γράφουμε το όνομα του πίνακα ακολουθούμενο από έναν δείκτη.

Όλοι οι πίνακες αποτελούνται από συνεχόμενες θέσεις μνήμης. Η χαμηλότερη διεύθυνση αντιστοιχεί στο πρώτο στοιχείο και η υψηλότερη διεύθυνση στο τελευταίο στοιχείο.

Αντί να δηλώνετε μεμονωμένες μεταβλητές, όπως number0, number1, ... και number99, δηλώνετε μια μεταβλητή πίνακα, π.χ. τη number και χρησιμοποιείτε number[0], number[1], ..., number[99] για να αναπαραστήσετε τις μεμονωμένες μεταβλητές που βρίσκονται στις θέσεις 0, 1, …, 99 του πίνακα.



Τα πλεονεκτήματα χρήσης πινάκων, είναι η χρήση πίνακα είναι ένας βολικός τρόπος για τη διαχείριση πολλών δεδομένων ίδιου τύπου, ενώ τα μειονεκτήματα είναι το γεγονός ότι απαιτούν μνήμη, η οποία δεσμεύεται από την αρχή του προγράμματος. Αυτό έχει σαν αποτέλεσμα, σε μεγάλα και σύνθετα προγράμματα την αδυναμία

εκτέλεσης τους, ενώ παράλληλα οι πίνακες περιορίζουν τις δυνατότητες του προγράμματος.

Χρησιμοποιούνται στη περίπτωση που έχουμε να διαχειριστούμε πολλά δεδομένα και πρέπει να είναι αποθηκευμένα για μεταγενέστερη επεξεργασία τους.

Για να δηλώσετε έναν πίνακα πρέπει να καθορίσετε τον τύπο των δεδομένων και το πλήθος των στοιχείων που απαιτούνται από έναν πίνακα ως εξής:

```
<τύπος_δεδομένων> Όνομα_Πίνακα[μέγεθος_πίνακα];
```

Για παράδειγμα, για να δηλώσετε έναν πίνακα 10 στοιχείων που ονομάζεται MyArray τύπου double, χρησιμοποιείται τη δήλωση:

```
double MyArray[10];
```

Μπορείτε να αρχικοποιήσετε έναν πίνακα, αρχικοποιώντας είτε ένα-προς-ένα τα στοιχεία που περιέχει ή χρησιμοποιώντας μια γενικής πρόταση αρχικοποίησης του, όπως για παράδειγμα:

```
double MyArray[5] = {1000.0, 2.0, 3.4, 7.0, 50.0};
```

Ο αριθμός των τιμών μεταξύ των αγκύλων { } δεν μπορεί να είναι μεγαλύτερος από τον αριθμό των στοιχείων που δηλώνουμε για τον πίνακα μεταξύ αγκύλων δήλωσης μεγέθους του πίνακα ([ ]).

Εάν παραλείψετε το μέγεθος του πίνακα, δημιουργείται ένας πίνακας αρκετά μεγάλος ώστε να διατηρεί τη δήλωση αρχικοποίησης, επομένως, εάν γράψετε:

```
double MyArray[] = {1000.0, 2.0, 3.4, 7.0, 50.0};
```

θα δημιουργήσετε ακριβώς τον ίδιο πίνακα όπως παραπάνω.

Μια γενική πρόταση για την εκχώρηση ενός μεμονωμένου στοιχείου στις θέσεις των στοιχείων του πίνακα είναι η ακόλουθη:

```
MyArray[4] = 50.0;
```

δήλωση η οποία εκχωρεί στο 5ο στοιχείο του πίνακα την τιμή 50.0.

Όλοι οι πίνακες έχουν το 0 ως δείκτη του πρώτου τους στοιχείου που ονομάζεται επίσης δείκτης βάσης και ο τελευταίος δείκτης ενός πίνακα θα είναι το συνολικό μέγεθος του πίνακα μείον 1.

Για την προσπέλαση ενός μονοδιάστατου πίνακα Ν θέσεων απαιτείται η χρήση μιας δομής επανάληψης

Ένα παράδειγμα δήλωσης και προσπέλασης πίνακα είναι:

```
#include <stdio.h>

int main () {

   int n[ 10 ];
   int i,j;
```

```
      for ( i = 0; i < 10; i++ ) {
         n[ i ] = i + 100;
      }

      for (j = 0; j < 10; j++ ) {
         printf("Element[%d] = %d\n", j, n[j] );
      }

      return 0;
   }
```

Το οποίο όταν εκτελεστεί θα μας παράγει σαν έξοδο το ακόλουθο:

Element[0] = 100

Element[1] = 101

Element[2] = 102

Element[3] = 103

Element[4] = 104

Element[5] = 105

Element[6] = 106

Element[7] = 107

Element[8] = 108

Element[9] = 109

Ένας πίνακας μπορεί να είναι πολυδιάστατος, δηλαδή δισδιάστατος, τρισδιάστατος και γενικά ν-διάστατος πίνακας. Όσο αφορά τους δισδιάστατους πίνακες αν το μέγεθος των δύο διαστάσεων είναι ίσο, δηλαδή είναι NxN, τότε ο πίνακας λέγεται τετραγωνικός.

Στη περίπτωση αυτή μας ενδιαφέρει αν τα δεδομένα καταχωρηθούν στο πίνακα ανά γραμμή ή ανά στήλη, οπότε έχουμε για παράδειγμα εισαγωγή από το πληκτρολόγιο 20 ακέραιων και καταχώρηση στον πίνακα Α[10,2].

Για να δηλώσετε έναν δισδιάστατο ακέραιο πίνακα μεγέθους [x][y], θα γράφατε:

```
<τύπος_δεδομένων> Όνομα_Πίνακα[πλήθος_γραμμών][πλήθος_στηλών];
```

ή

```
<τύπος_δεδομένων> Όνομα_Πίνακα[x][y];
```

Ένας δισδιάστατος πίνακας μπορεί να θεωρηθεί ως ένας πίνακας που θα έχει x αριθμό γραμμών και y αριθμό στηλών. Ένας δισδιάστατος πίνακας Α, ο οποίος περιέχει τρεις γραμμές και τέσσερις στήλες μπορεί να παρουσιαστεί ως εξής:

	Στήλη 0	Στήλη 1	Στήλη 2	Στήλη 3

Γραμμή 0	A[0][0]	A[0][1]	A[0][2]	A[0][3]
Γραμμή 1	A[1][0]	A[1][1]	A[1][2]	A[1][3]
Γραμμή 2	A[2][0]	A[2][1]	A[2][2]	A[2][3]

Οι πολυδιάστατοι πίνακες μπορούν να αρχικοποιηθούν καθορίζοντας τιμές σε αγκύλες για κάθε γραμμή. Ακολουθεί ένας πίνακας με 3 γραμμές και κάθε γραμμή έχει 4 στήλες:

```
int a[3][4] = {
      {0, 1, 2, 3},
      {4, 5, 6, 7},
      {8, 9, 10, 11}
};
```

Οι εσωτερικές αγκύλες που υποδεικνύουν την προβλεπόμενη γραμμή, είναι προαιρετικές και μπορεί η παραπάνω δήλωση να εκφραστεί ισοδύναμα:

```
int a[3][4] = {0,1,2,3,4,5,6,7,8,9,10,11};
```

Η προσπέλαση πολυδιάστατων πινάκων πραγματοποιείται με εμφωλευμένες επαναλήψεις, όπως για παράδειγμα η ακόλουθη υλοποίηση για έναν διδιάστατο πίνακα:

```
#include <stdio.h>

int main () {

   int a[5][2] = { {0,0}, {1,2}, {2,4}, {3,6},{4,8}};
   int i, j;

   for ( i = 0; i < 5; i++ ) {

      for ( j = 0; j < 2; j++ ) {
         printf("a[%d][%d] = %d\n", i,j, a[i][j] );
      }
   }

   return 0;
}
```

ΕΡΓΑΣΙΕΣ

1. Να γραφεί ένα πρόγραμμα το οποίο να διαβάζει ακεραίους και να τους αποθηκεύει σε έναν τετραγωνικό 5×5 πίνακα. Στη συνέχεια, να διαβάζει έναν ακέραιο στο [1, 5] (π.χ. x) και να αποθηκεύει σε έναν δεύτερο πίνακα (καταλλήλων διαστάσεων) τα στοιχεία του πρώτου, εκτός από τα στοιχεία της x-γραμμής και της x-στήλης. Να χρησιμοποιήσετε συναρτήσεις για τις επαναλαμβανόμενες ενέργειες.

2. Δημιουργήστε πρόγραμμα στην C το οποίο θα παίρνει ως είσοδο ένα μητρώο(ΠΙΝΑΚΑ) 3Χ3 και θα εμφανίζει:
   1) Τα διαγώνια στοιχεία (0 σε όλα τα υπόλοιπα στοιχεία)
   2) Το άνω τριγωνικό μητρώο & κάτω τριγωνικό μητρώο (0 σε όλα τα υπόλοιπα στοιχεία)
   3) Το άθροισμα των στοιχείων για τα 1-3.
   4) Το ανάστροφο (αν Α το μητρώο εισόδου τότε θα εμφανίζει το ΑΤ):

   Υπόδειγμα: Ένα Α μητρώο είναι της μορφής που φαίνεται στο ακόλουθο σχήμα:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

   Διαγώνιος πίνακας, άνω τριγωνικό μητρώο, κάτω τριγωνικό μητρώο:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{bmatrix} \qquad \begin{bmatrix} 1 & 2 & 3 \\ 0 & 5 & 6 \\ 0 & 0 & 9 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & 0 \\ 4 & 5 & 0 \\ 7 & 8 & 9 \end{bmatrix}$$

   Αναστροφή (Transpose):

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \qquad A^T = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$

   Δηλαδή $(A^T)_{i,j} = A_{j,i}$

3. Να γραφεί ένα πρόγραμμα το οποίο να διαβάζει τα ονόματα 20 βιβλιοπωλείων, 500 τίτλους βιβλίων και τον αριθμό των αντιτύπων που έχει για κάθε τίτλο το κάθε βιβλιοπωλείο (θεωρήστε ότι όλα τα ονόματα είναι μέχρι 100 χαρακτήρες το καθένα). Το πρόγραμμα να χρησιμοποιεί κατάλληλους πίνακες για την αποθήκευση των δεδομένων. Στη συνέχεια, το πρόγραμμα να διαβάζει τον τίτλο ενός βιβλίου και να εμφανίζει τα ονόματα των βιβλιοπωλείων στα οποία υπάρχουν διαθέσιμα αντίτυπα και πόσα είναι αυτά. Τέλος, να διαβάζει το όνομα ενός βιβλιοπωλείου και να εμφανίζει το πλήθος των βιβλίων που διαθέτει. Να χρησιμοποιήσετε συναρτήσεις για τις επαναλαμβανόμενες ενέργειες.

4. Γράψτε ένα πρόγραμμα το οποίο θα δέχεται ως είσοδο ένα πίνακα αλφαριθμητικών. Κάθε αλφαριθμητικό περιέχει ένα σύνολο από λέξεις χωρισμένα

από ένα ή περισσότερα κενά (μπορείτε για απλότητα να θεωρήσετε ότι κάθε πρόταση είναι μία λέξη, έχει όμως νόημα για εξάσκηση να κάνετε κώδικα και για την περίπτωση που κάθε πρόταση έχει πιο πολλές από μία λέξεις). Το πρόγραμμα θα μετράει το μήκος των λέξεων αυτών και θα εμφανίζει το ιστόγραμμα των μηκών με γραφικό τρόπο ως εξής:

```
Word Length             |NumberofOccurrences
1                       |
2                       | *******
3                       | **********
4                       | ****
5                       | ***
6                       | *****
7                       | ***
8                       | ***
9                       | *
10                      |
11                      |
12                      |
13                      |
Diogenis>
```

Επεκτάσεις:

- Φροντίστε να μην προσμετρούνται τα σημεία στίξης (τελείες, παρενθέσεις κ.λπ.) στο μήκος της λέξης.

5. Θεωρήστε δύο συμβολοσειρές Α και Β καθεμία μέγιστου μήκους Μ-1. Κάθε συμβολοσειρά θεωρούμε ότι μοντελοποιεί ένα κείμενο συνεπώς αποτελείται από μία ακολουθία λέξεων με κάθε λέξη να αποτελείται από συνεχόμενες εμφανίσεις αριθμητικών και αλφαβητικών χαρακτήρων, όλων του Αγγλικού αλφαβήτου. Υποθέτουμε ότι οι λέξεις οριοθετούνται με ένα ή περισσότερα κενά και ότι γίνεται διάκριση μεταξύ κεφαλαίων και πεζών γραμμάτων. Να υλοποιήσετε κώδικα που διαβάζει το περιεχόμενο των Α και Β και για κάθε λέξη της Α δημιουργεί μία ακολουθία με τις θέσεις εμφάνισής της στη Β (ως θέση εμφάνισης μία λέξης εννοούμε τον αριθμό του κελιού του Β στο οποίο η λέξη αρχίζει να εμφανίζεται). Η υλοποίηση της ακολουθίας γίνεται είτε με λίστα, είτε με πίνακα. Να χρησιμοποιήσετε συναρτήσεις για τις επαναλαμβανόμενες ενέργειες.

   Στη συνέχεια ο κώδικας για κάθε διακριτή λεξη του Α, εκτυπώνει τις θέσεις εμφάνισης της στο Β.

   Υπόδειγμα: Για την εύρεση των λέξεων που εμφανίζονται σε μία συμβολοσειρά προτείνεται να χρησιμοποιηθεί η συνάρτηση char * strtok(char *s, char * ct) της

βιβλιοθήκης <string.h>. Για την σύγκριση αλφαριθμητικων προτεινεται η συνάρτηση int strcmp(cs, ct) της βιβλιοθήκης <string.h>.

6. Να γραφεί πρόγραμμα που θα ζητάει από το χρήστη να του δώσει τις διαστάσεις για να κατασκευάσει ένα δισδιάστατο πίνακα ακεραίων. Μόλις ο χρήστης δώσει τις δύο διαστάσεις, το πρόγραμμα θα δεσμεύει δυναμικά μνήμη για τον πίνακα με χρήση της malloc και θα διαβάζει από τον χρήστη μια ακέραια τιμή για κάθε στοιχείο του πίνακα. Στη συνέχεια, να κατασκευαστεί συνάρτηση, η οποία θα καλείται από το πρόγραμμα και να εκτελεί τα παρακάτω:

   i. θα υπολογίζει το άθροισμα κάθε στήλης του πίνακα και θα το αποθηκεύει σ' ένα νέο πίνακα.

   ii. να επιστρέφει το νέο πίνακα στο κύριο πρόγραμμα.

   Το κύριο πρόγραμμα θα πρέπει να εκτυπώνει τα στοιχεία του νέου πίνακα που δημιουργήθηκε απ' τη συνάρτηση.

7. Δημιουργήστε κώδικα που να υλοποιεί το ακόλουθο (γνωστό) "παιχνίδι" λέξεων: αρχικά ζητείται από τον πρώτο χρήστη να εισαγάγει μια λέξη. Στη συνέχεια η οθόνη καθαρίζεται και εμφανίζονται χαρακτήρες '_' στη θέση των γραμμάτων της λέξης καθώς και ένας μετρητής που δηλώνει πόσες προσπάθειες απομένουν στο δεύτερο χρήστη, που καλείται να μαντέψει τη λέξη δίνοντας έναν χαρακτήρα κάθε φορά. Αν ο εισαγόμενος χαρακτήρας υπάρχει εμφανίζεται στην κατάλληλη θέση της λέξης, αν δεν υπάρχει ο μετρητής μειώνεται κατά 1 και αν έχει επιλεχθεί ήδη εμφανίζεται ένα ενημερωτικό μήνυμα. Το παιχνίδι τελειώνει όταν βρεθούν όλα τα γράμματα της λέξης ή όταν μηδενιστεί ο μετρητής (οπότε εμφανίζονται τα κατάλληλα μηνύματα στην οθόνη).

   Προτείνεται:

   i. όλοι οι εισαγόμενοι χαρακτήρες να μετατρέπονται σε μικρά,

   ii. ο έλεγχος ύπαρξης ενός χαρακτήρα σε μια λέξη να γίνεται με την κλήση συνάρτησης η οποία θα λαμβάνει ως ορίσματα τη λέξη και τον χαρακτήρα προς αναζήτηση και θα επιστρέφει 0 αν δεν υπάρχει ή 1 αν υπάρχει.

   Υπόδειγμα: Ο καθαρισμός της οθόνης γίνεται με την κλήση system("clear"); (Unix) της συνάρτησης system της βιβλιοθήκης <stdlib.h> (για τα Windows (εκτέλεση σε γραμμή εντολών) η κλήση είναι system("cls")).

8. Να γραφεί ένα πρόγραμμα το οποίο να προσομοιώνει ένα απλό σύστημα κράτησης δωματίων σε ένα ξενοδοχείο χρησιμοποιώντας πίνακα δύο διαστάσεων για την αποθήκευση των κρατήσεων (0 σημαίνει ελεύθερο δωμάτιο και 1 σημαίνει κράτηση). Θεωρήστε ότι το ξενοδοχείο έχει 8 πτέρυγες με 20 δωμάτια σε κάθε

πτέρυγα. Το πρόγραμμα να δίνει τη δυνατότητα στον χρήστη να επιλέξει τις παρακάτω λειτουργίες:

(α) Εμφάνιση των ελεύθερων δωματίων σε κάθε πτέρυγα.

(β) Κράτηση δωματίου. Το πρόγραμμα να διαβάζει τον αριθμό της πτέρυγας και να δεσμεύει το πρώτο ελεύθερο δωμάτιο σε αυτή την πτέρυγα. Αν δεν υπάρχει ελεύθερο δωμάτιο, το πρόγραμμα να προτρέπει τον χρήστη να εισάγει νέα πτέρυγα μέχρι να γίνει η κράτηση.

(γ) Ακύρωση κράτησης. Το πρόγραμμα να διαβάζει τον αριθμό της πτέρυγας και τον αριθμό του δωματίου και να ακυρώνει την κράτηση.

(δ) Τερματισμός προγράμματος.

Τα βήματα (α), (β), (γ), (δ) να υλοποιηθούν με χρήση συναρτήσεων.

Για την εύρεση των λέξεων που εμφανίζονται σε μία γραμμή του πίνακα να χρησιμοποιηθεί η συνάρτηση `char * strtok(char *s, char * ct)` της βιβλιοθήκης `<string.h>`. Η `strtok` χωρίζει τη συμβολοσειρά `s` σε λέξεις που οριοθετούνται με χαρακτήρες του `ct`. Η πρώτη κλήση της `strtok` έχει σαν όρισμα την συμβολοσειρά που θέλουμε να διαχωρίσουμε σε λέξεις, και επιστρέφει την πρώτη λέξη. Για να βρούμε τις επόμενες λέξεις πρέπει να καλέσουμε επαναλαμβανόμενα τη `strtok` με πρώτο όρισμα το `NULL`, έως ότου τελικά η `strtok` επιστρέψει `NULL` κάτι που σημαίνει ότι δεν υπάρχει άλλη λέξη.

Στην περίπτωση μας θα έχουμε ως `ct` το " " (οριοθετούμε λέξεις με το κενό). Η `strtok` επηρεάζει το περιεχόμενο του αρχικού ορίσματος της (`s`), οπότε θα πρέπει να κληθεί σε ένα αντίγραφο (δημιουργημένο λόγου χάρη με `strcpy`) της γραμμής του πίνακα που θέλουμε να διαχωρίσουμε σε λέξεις. Ενδεικτικός κώδικας επεξεργασία του κειμένου με κωδικό `i` για εύρεση των λέξεων που τον συνιστούν είναι ο εξής:

```
char *word;           // λέξη
char temp[M];         // συμβολοσειρά για αντίγραφο κειμένου
strcpy(temp, document_table[i]); // αντιγραφή κειμένου με κωδικό
i
word = strtok(temp, " " ); // πρώτη λέξη κειμένου με κωδικό
i

while (word != NULL ) // ενόσω υπάρχουν λέξεις
{
/* ΕΔΩ ΓΡΑΦΕΤΕ ΤΟΝ ΚΩΔΙΚΑ ΠΟΥ ΕΠΕΞΕΡΓΑΖΕΤΑΙ ΤΗ ΛΕΞΗ word */

word = strtok(NULL, " " ); // επόμενη λέξη του κειμένου με κωδικό
i
}
```