Δ.Π.Θ

Εισαγωγή στην Επιστήμη των Υπολογιστών

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΑΡΑΓΩΓΗΣ & ΔΙΟΙΚΗΣΗΣ

3ο Φυλλάδιο Ασκήσεων

(Ενδεικτικές Λύσεις)

Dr. Αθανάσιος Μπαλαφούτης
Εργαστηριακό Διδακτικό Προσωπικό
Τομέας Συστημάτων Παραγωγής
Εργαστήριο Ρομποτικής και Αυτοματισμών
abalafou@pme.duth.gr
Γραφείο 304, τηλ.: 25410 – 79892

Εισαγωγή/Εκτύπωση Στοιχείων Πίνακα


```c
#include <stdio.h>

int main()
{
    int arr[5];
    for(int i = 0; i < 5; i++){
        printf("Δώσε το %do στοιχείο του πίνακα: ", i+1);
        scanf("%d", &arr[i]);
    }

    for (int i = 0; i < 5; i++)
        printf("%d ", arr[i]);

    return 0;
}
```

Άσκηση 1

Μονοδιάστατος πίνακας 50 θέσεων περιέχει τις ποσότητες παραγωγής (τύπου int) ενός προϊόντος.
Να γραφεί πρόγραμμα σε γλώσσα C που θα υλοποιεί τα παρακάτω :

1. Εισαγωγή των στοιχείων του πίνακα με χρήση τυχαίων αριθμών (περιοχή τιμών 0-100).
2. Εμφάνιση των στοιχείων του πίνακα.
3. Εύρεση της μέσης τιμής των στοιχείων του πίνακα.
4. Δημιουργία νέου πίνακα που θα περιέχει όλες τις ποσότητες των οποίων η διαφορά από τη μέση τιμή των στοιχείων του αρχικού πίνακα είναι μεγαλύτερη κατά 20% της μέσης τιμής.
5. Εμφάνιση των στοιχείων του νέου πίνακα καθώς και του πλήθους των στοιχείων του.

Άσκηση 1 - Λύση


```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main(){

    int A[50], new_A[50], random_x, lower, upper, k;
    float mo, sum;

    srand(time(0));
    lower = 0;
    upper = 100;

    for(int i = 0; i <50; i++){
        random_x = rand() % (upper - lower + 1) + lower;
        A[i] = random_x;
    }
    printf("50 τυχαίοι αριθμοί από 0-100: \n");
    for(int i = 0; i <50; i++)
       printf("%d \n", A[i]);

    sum = 0;
    for(int i = 0; i <50; i++)
       sum += A[i];

    mo = sum / 50;
    printf("Μέσος όρος: %.2f \n", mo);

    k = 0;
    for(int i = 0; i <50; i++){
        if(A[i] > mo + 0.2 * mo){
            new_A[k] = A[i];
            k++;
        }
    }
    printf("Πλήθος στοιχείων πίνακα: %d \n", k);
    for(int i = 0; i < k; i++)
       printf("%d \n", new_A[i]);
    return 0;
}
```

Άσκηση 2

Να γραφεί πρόγραμμα σε γλώσσα C που θα εισάγει N ακεραίους αριθμούς σε μονοδιάστατο πίνακα N θέσεων (N = γνωστό).
Στη συνέχεια θα εισάγεται ένας ακέραιος αριθμός z και το πρόγραμμα θα βρίσκει αν ο αριθμός υπάρχει στον πίνακα και σε ποια θέση.
Αν ο αριθμός υπάρχει περισσότερες από μία φορές να εμφανίζεται μόνον η θέση όπου εμφανίζεται για πρώτη φορά.

Άσκηση 2 - Λύση (1/2)


```c
#include <stdio.h>

#define N 5

int main(){
    int table[N], n, pos_n;

    for(int i = 0; i < N; i++){
        printf("Δώσε το %do στοιχείο του πίνακα: ", i+1);
        scanf("%d", &table[i]);
    }

    printf("Δώσε έναν ακέραιο: ");
    scanf("%d", &n);
    pos_n = -1;

    // ... συνεχίζεται ...
```

Άσκηση 2 - Λύση (2/2)


```
// ...

    for(int i = 0; i < N; i++){
        if(table[i] == n){
            pos_n = i;
            break;
        }
    }
    if(pos_n != -1)
       printf("Ο αριθμός %d βρέθηκε στη θέση: %d \n", n, pos_n + 1);
    else
       printf("Ο αριθμός %d δεν βρέθηκε \n", n);

    return 0;
}
```

Άσκηση 3

Να γραφεί πρόγραμμα σε γλώσσα C που θα εισάγει N ακεραίους αριθμούς σε μονοδιάστατο πίνακα N θέσεων (N = γνωστό).
Στη συνέχεια θα εισάγεται ένας ακέραιος αριθμός z και το πρόγραμμα θα βρίσκει αν ο αριθμός υπάρχει στον πίνακα και σε ποια θέση.
Αν ο αριθμός υπάρχει περισσότερες από μία φορές θα εμφανίζονται όλες οι θέσεις του πίνακα όπου υπάρχει ο αριθμός z.

Άσκηση 3 - Λύση (1/2)


```c
#include <stdio.h>

#define N 5

int main(){
    int table[N], n, pos_n;

    for(int i = 0; i < N; i++){
        printf("Δώσε το %do στοιχείο του πίνακα: ", i+1);
        scanf("%d", &table[i]);
    }

    printf("Δώσε έναν ακέραιο: ");
    scanf("%d", &n);
    pos_n = -1;

    // ... συνεχίζεται ...
```

Άσκηση 3 - Λύση (2/2)


```
// ...

    for(int i = 0; i < N; i++){
        if(table[i] == n){
            pos_n = i;
            printf("Ο αριθμός %d βρέθηκε στη θέση: %d \n", n, pos_n + 1);
        }
    }
    if(pos_n == -1)
       printf("Ο αριθμός %d δεν βρέθηκε \n");

    return 0;
}
```

Άσκηση 4

Να γραφεί πρόγραμμα σε γλώσσα C που θα εναλλάσσει τα περιεχόμενα ενός μονοδιάστατου πίνακα ακεραίων Ν θέσεων (Ν = γνωστό) ως εξής :

- Το 1ο στοιχείο με το Ν-οστό στοιχείο
- Το 2ο στοιχείο με το Ν-1 στοιχείο κλπ

Το πρόγραμμα θα εμφανίζει τον πίνακα πριν και μετά την εναλλαγή των στοιχείων του.

Άσκηση 4 - Λύση


```c
#include <stdio.h>

#define N 5

int main(){
    int table[N], temp;

    for(int i = 0; i < N; i++){
        printf("Δώσε το %do στοιχείο του πίνακα: ", i+1);
        scanf("%d", &table[i]);
    }

    printf("Αρχικός Πίνακας: \n");

    for(int i = 0; i < N; i++)
       printf("%d \n", table[i]);
```

```c
    for(int i = 0; i < N / 2; i++){
        temp = table[i];
        table[i] = table[(N - 1) - i];
        table[(N - 1) - i] = temp;

    }

    printf("Πίνακας μετά την εναλλαγή: \n");
    for(int i = 0; i < N; i++)
       printf("%d \n", table[i]);

    return 0;
}
```

Άσκηση 5

Να γραφεί πρόγραμμα σε γλώσσα C που θα διαχωρίζει ένα δεδομένο πίνακα ακεραίων τιμών N θέσεων (N =γνωστό) σε δύο νέους πίνακες που θα περιέχουν αντίστοιχα τις άρτιες και τις περιττές τιμές του αρχικού πίνακα.
Το πρόγραμμα θα εμφανίζει τον αρχικό πίνακα, τους δύο νέους πίνακες καθώς και το πλήθος των στοιχείων τους.

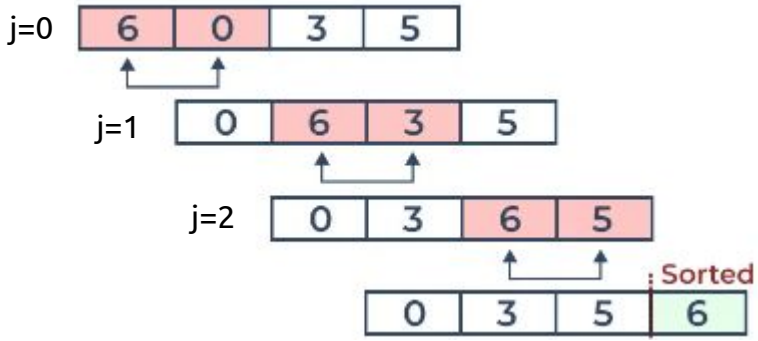
Άσκηση 5 - Λύση


```c
#include <stdio.h>
#define N 5
int main(){
    int table[N], even[N], odd[N], j, k;
    for(int i = 0; i < N; i++){
        printf("Δώσε το %do στοιχείο του πίνακα: ", i+1);
        scanf("%d", &table[i]);
    }
    j = 0; k = 0;
    for(int i = 0; i < N; i++){
        if(table[i] % 2 == 0){
            even[j] = table[i];
            j++;
        }
        else{
            odd[k] = table[i];
            k++;
        }
    }
```

```c
    printf("Αρχικός Πίνακας: \n");

    for(int i = 0; i < N; i++)
       printf("%d \n", table[i]);

    printf("Πλήθος Άρτιων: %d \n", j);

    for(int i = 0; i < j; i++)
       printf("%d \n", even[i]);

    printf("Πλήθος Περιττών: %d \n", k);

    for(int i = 0; i < k; i++)
       printf("%d \n", odd[i]);

    return 0;
}
```

Ταξινόμηση Πίνακα με Φυσαλίδα

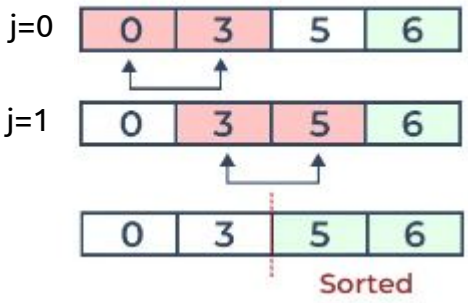
Βήμα 1ο: i = 0




```
for(int i = 0; i < N-1; i++){
    for(int j = 0; j < (N-1)-i; j++){
        if(table[j] > table[j+1]){
            temp = table[j];
            table[j] = table[j+1];
            table[j+1] = temp;
        }
    }
}
```

Ταξινόμηση Πίνακα με Φυσαλίδα

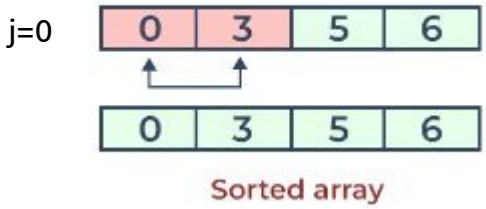
Βήμα 2ο: i = 1




```
for(int i = 0; i < N-1; i++){
    for(int j = 0; j < (N-1)-i; j++){
        if(table[j] > table[j+1]){
            temp = table[j];
            table[j] = table[j+1];
            table[j+1] = temp;
        }
    }
}
```

Ταξινόμηση Πίνακα με Φυσαλίδα

Βήμα 3ο: i = 2

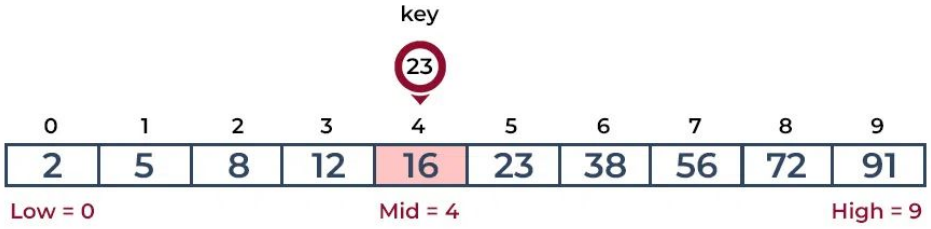



```
for(int i = 0; i < N-1; i++){
    for(int j = 0; j < (N-1)-i; j++){
        if(table[j] > table[j+1]){
            temp = table[j];
            table[j] = table[j+1];
            table[j+1] = temp;
        }
    }
}
```

Ταξινόμηση Πίνακα με Φυσαλίδα

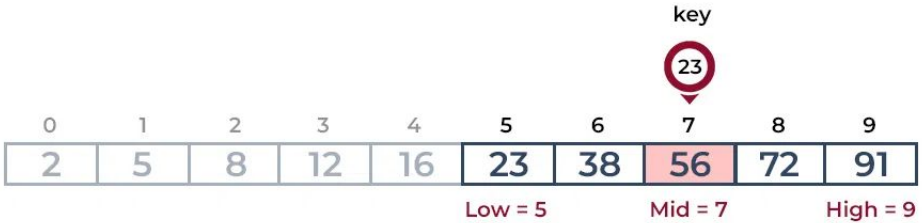

```c
#include <stdio.h>

#define N 10

int main(){
    int table[N], temp;

    for(int i = 0; i < N; i++){
      printf("Δώσε το %do στοιχείο του πίνακα: ", i+1);
      scanf("%d", &table[i]);
    }
    for(int i = 0; i < N-1; i++){
        for(int j = 0; j < (N-1)-i; j++){
            if(table[j] > table[j+1]){
                temp = table[j];
                table[j] = table[j+1];
                table[j+1] = temp;
            }
        }
    }
    printf("Ταξινομημένος Πίνακας: \n");
    for(int i = 0; i < N; i++)
        printf("%d \n", table[i]);

    return 0;
}
```

Δυαδική Αναζήτηση



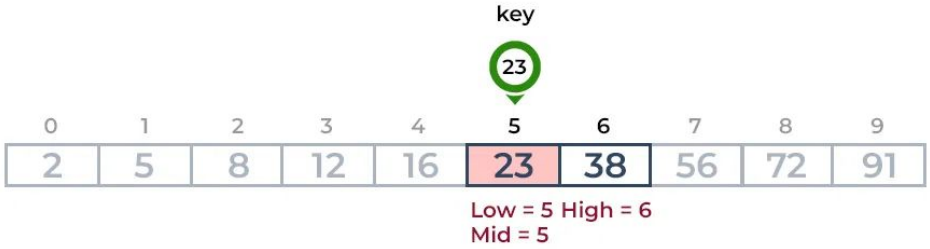

```
left = 0; right = N-1;

while(left <= right){
    int middle = (left + right) / 2;

    if(table[middle] == x){
      printf("Βρέθηκε στη θέση: %d \n", middle + 1);
      find = true;
      break;
    }
    if(table[middle] < x)
       left = middle + 1;
    else
       right = middle - 1;
}
```

Δυαδική Αναζήτηση




```
left = 0; right = N-1;

while(left <= right){
    int middle = (left + right) / 2;

    if(table[middle] == x){
      printf("Βρέθηκε στη θέση: %d \n", middle + 1);
      find = true;
      break;
    }
    if(table[middle] < x)
       left = middle + 1;
    else
       right = middle - 1;
}
```

Δυαδική Αναζήτηση




```
left = 0; right = N-1;

while(left <= right){
    int middle = (left + right) / 2;

    if(table[middle] == x){
      printf("Βρέθηκε στη θέση: %d \n", middle + 1);
      find = true;
      break;
    }
    if(table[middle] < x)
       left = middle + 1;
    else
       right = middle - 1;
}
```

Δυαδική Αναζήτηση


```c
#include <stdio.h>
#include <stdbool.h>

#define N 10

int main(){
    int table[N], temp, x, left, right;
    bool find = false;


    for(int i = 0; i < N; i++){
      printf("Δώσε το  %do  στοιχείο του πίνακα:  ", i+1);
      scanf("%d", &table[i]);
    }

   // ... εδώ θα πρέπει να γίνει ταξινόμηση  ...

    printf("Δώσε έναν αριθμό προς αναζήτηση:");
    scanf("%d", &x);
```

```c
 left = 0; right = N-1;

 while(left <= right){
     int middle = (left + right) / 2;

     if(table[middle] == x){
       printf("Βρέθηκε στη θέση: %d \n", middle + 1);
       find = true;
       break;
     }
     if(table[middle] < x)
        left = middle + 1;
     else
        right = middle - 1;
 }
 if(find == false)
     printf("Το x δεν βρέθηκε στον πίνακα  \n");

 return 0;
}
```

Άσκηση 6

Να γραφεί πρόγραμμα σε γλώσσα C που θα εισάγει N ακεραίους αριθμούς σε μονοδιάστατο πίνακα N θέσεων (N = γνωστό) και θα τον ταξινομεί κατά αύξουσα σειρά των στοιχείων του.

Στη συνέχεια θα εισάγεται ένας ακέραιος αριθμός Α και :

- αν ο αριθμός Α υπάρχει στον πίνακα θα βρίσκεται η θέση του, με τις λιγότερες δυνατές προσπάθειες (δυαδική αναζήτηση). Αν ο αριθμός υπάρχει περισσότερες από μία φορές θα εμφανίζεται μόνον η θέση όπου θα εντοπιστεί για πρώτη φορά.
- αν ο αριθμός Α δεν υπάρχει στον πίνακα να εισάγεται σε αυτόν στην κατάλληλη θέση ώστε ο πίνακας να διατηρείται ταξινομημένος.

Άσκηση 6 - Λύση (1/2)


```c
#include <stdio.h>
#include <stdbool.h>
#define N 5
int main(){
    int table[N+1], temp, x, left, right;
    bool find = false;
    for(int i = 0; i < N; i++){
        printf("Δώσε το %do  στοιχείο του πίνακα: ", i+1);
        scanf("%d", &table[i]);
    }
    for(int i = 0; i < N-1; i++){
        for(int j = 0; j < (N-1)-i; j++){
            if(table[j] > table[j+1]){
                temp = table[j];
                table[j] = table[j+1];
                table[j+1] = temp;
            }
        }
    }   // ... συνεχίζεται ...
```

Άσκηση 6 - Λύση (2/2)


```
// ...
printf("Ταξινομημένος Πίνακας: \n");
for(int i = 0; i < N; i++)
   printf("%d \n", table[i]);

printf("Δώσε έναν αριθμό: ");
scanf("%d", &x);

left = 0; right = N-1;

while(left <= right){
    int middle = (left + right) / 2;

    if(table[middle] == x){
        printf("Βρέθηκε στη θέση: %d\n", middle + 1);
        find = true;
        break;
    }
        if(table[middle] < x)
           left = middle + 1;
        else
           right = middle - 1;
    }
    if(find == false){
        printf("Το x δεν βρέθηκε στον πίνακα \n");
        int last = N-1;
        while(table[last] > x){
            table[last+1] = table[last];
            last--;
        }
        table[last+1] = x;
        printf("Νέος πίνακας: \n");
        for(int i = 0; i < N+1; i++)
           printf("%d \n", table[i]);
    }
    return 0;
}
```

Άσκηση 7

Για ορισμένους ακέραιους αριθμούς ισχύει ότι το άθροισμα των ψηφίων του αριθμού διαιρεί τον ίδιο τον αριθμό
π.χ.
για τον αριθμό 1729 ισχύει : 1+7+2+9=19 και 1729/19=91.

Να γραφεί πρόγραμμα σε γλώσσα C που θα εισάγει σε έναν μονοδιάστατο πίνακα όλους αυτούς τους θετικούς ακέραιους που πληρούν την παραπάνω ιδιότητα και είναι μικρότεροι του 10000.

Ο αλγόριθμος θα εμφανίζει στο τέλος τον πίνακα καθώς και το πλήθος των στοιχείων του.

Άσκηση 7 - Λύση


```c
#include <stdio.h>

int main(){
    int table[10000], x, count, sum;

    count = 0;
    for(int i = 1; i < 10000; i++){
        sum = 0;
        x = i;
        while(x != 0){
            sum += x % 10;
            x = x / 10;
        }
        if(i % sum == 0){
            table[count] = i;
            count++;
        }
    }
```

```c
    printf("Πλήθος αριθμών: %d \n", count);

    for(int i = 0; i < count; i++)
        printf("%d \n", table[i]);

    return 0;
}
```

Άσκηση 8

Θεωρείστε έναν ακέραιο και θετικό αριθμό **n** που έχει **d** ψηφία.
Διαχωρίστε το τετράγωνο του αριθμού σε δύο τμήματα, το πρώτο από τα δεξιά που έχει **d** ψηφία και το δεύτερο από τα αριστερά που έχει **d** ή **d-1** ψηφία.
Προσθέστε αυτούς τους δύο αριθμούς και ελέγξτε αν το άθροισμά τους είναι ίσο με τον αρχικό αριθμό **n**.

Παραδείγματα:

9 ($9^2 = 81$, 8 + 1 = 9), 45 ($45^2 = 2025$, 20 + 25 = 45).

Να γραφεί πρόγραμμα σε γλώσσα C που θα εισάγει σε έναν μονοδιάστατο πίνακα όλους αυτούς τους αριθμούς που πληρούν την παραπάνω ιδιότητα και είναι μικρότεροι του 100.
Ο αλγόριθμος θα εμφανίζει στο τέλος τον πίνακα καθώς και το πλήθος των στοιχείων του.

Άσκηση 8 - Λύση


```c
#include <stdio.h>
#include <math.h>

int main(){
    int table[100], sq_i, x1, x2, count;

    count = 0;

    for(int i = 0; i < 100; i++){
        sq_i = pow(i, 2);
        if(i < 10){
            x1 = sq_i % 10;
            x2 = sq_i / 10;
        }
        else{
            x1 = sq_i % 100;
            x2 = sq_i / 100;
        }
        if(x1 + x2 == i){
            table[count] = i;
            count++;

        }
    }
    printf("Πλήθος αριθμών: %d \n", count);
    for(int i = 0; i < count; i++)
       printf("%d \n", table[i]);

    return 0;
}
```

Άσκηση 9

Να γραφεί πρόγραμμα σε γλώσσα C που θα βρίσκει και θα εμφανίζει όλες τις τετράδες ακεραίων αριθμών, έστω (*α, β, γ, δ*), που:

- ανήκουν στο διάστημα [1, 20]
- είναι διαφορετικές μεταξύ τους (δηλαδή να αποφύγετε την επανάληψη τετράδων όπως (*α, β, γ, δ*), (*α, γ, δ, β*), (*γ, α, β, δ*), κλπ), και
- Ικανοποιούν την ιδιότητα $\alpha^2 + \beta^2 + \gamma^2 = \delta^2$

Παράδειγμα : $1^2 + 4^2 + 8^2 = 9^2$, αλλά όχι $1^2 + 2^2 + 2^2 = 3^2$, διότι επαναλαμβάνεται το 2.

Επίσης:

1. Να βρεθεί και να εμφανιστεί το πλήθος αυτών των διαφορετικών μεταξύ τους τετράδων.
2. Να δημιουργηθεί ένας μονοδιάστατος αριθμητικός πίνακας ακεραίων που θα περιέχει τις πρώτες 10 τέτοιες τετράδες (αν υπάρχουν λιγότερες τότε θα περιέχει ακριβώς όσες υπάρχουν)

Άσκηση 9 - Λύση (1/2)


```c
#include <stdio.h>

int main(){

    int table[10][4], count = 0;

    for (int a = 1; a <= 20; ++a) {
        for (int b = a; b <= 20; ++b) {
            for (int c = b; c <= 20; ++c) {
                for (int d = c; d <= 20; ++d) {
                    if (a != b && a != c && a != d && b != c && b != d && c != d) {
                        if (a * a + b * b + c * c == d * d) {
                            if(count < 10){
                               table[count][0] = a;
                               table[count][1] = b;
                               table[count][2] = c;
                               table[count][3] = d;
                            }
                            count++;   // ... συνεχίζεται ...
```

Άσκηση 9 - Λύση (2/2)


```
// ...

                        }
                    }
                }
            }
        }
    }
    printf("Πλήθος τετράδων: %d \n", count);
    for(int i = 0; i < 10; i++)
       printf("(%d, %d, %d, %d)\n", table[i][0], table[i][1], table[i][2], table[i][3]);

    return 0;
}
```

Άσκηση 10

Να γραφεί πρόγραμμα σε γλώσσα C που θα βρίσκει όλες τις θετικές Πυθαγόρειες τριάδες:

$(m^2 - n^2 , 2mn, m^2 + n^2)$ για $1 <= m,n <= 30$.

Οι τριάδες αυτές θα καταχωρούνται διαδοχικά σε έναν μονοδιάστατο αριθμητικό πίνακα.
Στη συνέχεια θα εμφανίζονται όλες οι τριάδες που έχουν καταχωρηθεί στον πίνακα καθώς και το πλήθος αυτών των τριάδων.

Οι πυθαγόρειες τριάδες δίνονται από τον τύπο: $(m^2 - n^2)^2 + (2mn)^2 = (m^2 + n^2)^2$

Άσκηση 10 - Λύση (1/2)


```c
#include <stdio.h>
#include <math.h>

int main(){
    int table[2700], count, a, b, c;
    count = 0;
    for(int m = 1; m <= 30; m++){
        for(int n = 1; n <= 30; n++){
            a = pow(m, 2) - pow(n, 2);
            b = 2 * m * n;
            c = pow(m, 2) + pow(n, 2);
            if(pow(a, 2) + pow(b, 2) == pow(c, 2)){
                table[count] = a;
                table[count + 1] = b;
                table[count + 2] = c;
                count += 3;
            }
        }
    }   // ... συνεχίζεται ...
```

Άσκηση 10 - Λύση (2/2)


```
// ...

    printf("Πλήθος τριάδων: %d \n", count / 3);
    for(int i = 0; i < count; i += 3)
       printf("(%d, %d, %d)\n", table[i], table[i+1], table[i+2]);

    return 0;
}
```

Άσκηση 11

Να γραφεί πρόγραμμα σε γλώσσα C που θα απομακρύνει από ένα μονοδιάστατο αριθμητικό πίνακα ακεραίων Ν θέσεων (Ν = γνωστό) όλες τις πολλαπλές ίδιες τιμές δημιουργώντας ένα νέο πίνακα όπου κάθε αριθμός θα εμφανίζεται μόνο μία φορά.

Άσκηση 11 - Λύση


```c
#include <stdio.h>

#define N 5

int main(){
    int table[N], temp[N], count;

    for(int i = 0; i < N; i++){
        printf("Δώσε το %do στοιχείο του πίνακα: ", i+1);
        scanf("%d", &table[i]);
    }

    count = 0;
    for (int i = 0; i < N; i++){
        int j;
        for (j = 0; j < count; j++){
            if (table[i] == temp[j])
                break;
        }
        if (j == count){
            temp[count] = table[i];
            count++;
        }
    }

    for(int i = 0; i < count; i++)
       printf("%d \n", temp[i]);


    return 0;
}
```

Άσκηση 12

Να γραφεί πρόγραμμα σε γλώσσα C που θα εισάγει N ακεραίους αριθμούς σε μονοδιάστατο πίνακα N θέσεων (N = γνωστό) και θα βρίσκει το μεγαλύτερο στοιχείο καθώς και τη θέση που κατέχει στον πίνακα.
Σε περίπτωση που υπάρχουν περισσότερα από ένα μέγιστα στοιχεία θα βρίσκονται και θα εμφανίζονται όλες οι αντίστοιχες θέσεις, χωρίς να χρησιμοποιηθεί διαδικασία ταξινόμησης του πίνακα.

Άσκηση 12 - Λύση


```c
#include <stdio.h>

#define N 5

int main(){

    int table[N], max;

    for(int i = 0; i < N; i++){
        printf("Δώσε το  %do στοιχείο του πίνακα:  ", i+1);
        scanf("%d", &table[i]);
    }

    max = table[0];
    for(int i = 1; i < N; i++){
        if(table[i] > max)
           max = table[i];
    }
```

```c
    printf("Μεγαλύτερος αριθμός: %d \n", max);
    printf("Υπάρχει στις θέσεις: \n");

    for(int i = 0; i < N; i++){
        if(table[i] == max)
           printf("%d \n", i);
    }

    return 0;
}
```

Άσκηση 13

Να γραφεί πρόγραμμα σε γλώσσα C που θα εισάγει N ακεραίους αριθμούς σε μονοδιάστατο πίνακα N θέσεων (N = γνωστό).
Το πρόγραμμα θα βρίσκει το μεγαλύτερο στοιχείο, τη θέση που κατέχει στον πίνακα και στη συνέχεια το αμέσως μικρότερο διαφορετικό στοιχείο καθώς και τη θέση που κατέχει.

Σε περίπτωση που υπάρχουν περισσότερα από ένα στοιχεία σε κάθε μία από τις δύο αναζητήσεις θα βρίσκονται και θα εμφανίζονται όλα, χωρίς να χρησιμοποιηθεί διαδικασία ταξινόμησης του πίνακα.

Άσκηση 13 - Λύση


```c
#include <stdio.h>
#define N 5
int main(){
    int table[N], max1, max2;

    for(int i = 0; i < N; i++){
        printf("Δώσε το %do στοιχείο του πίνακα: ", i+1);
        scanf("%d", &table[i]);
    }
    max1 = table[0];
    max2 = max1;
    for(int i = 1; i < N; i++){
        if(table[i] > max1){
            max2 = max1;
            max1 = table[i];
        }
        else if(table[i] > max2 && table[i] != max1)
            max2 = table[i];
    }
```

```c
  printf("Μεγαλύτερος αριθμός: %d \n", max1);
  printf("Υπάρχει στις θέσεις: \n");
  for(int i = 0; i < N; i++){
      if(table[i] == max1)
         printf("%d \n", i);
  }
  if(max1 == max2)
      printf("Δεν υπάρχει 2ος μεγαλύτερος \n");
  else{
      printf("2ος μεγαλύτερος αριθμός: %d\n", max2);
      printf("Υπάρχει στις θέσεις: \n");
      for(int i = 0; i < N; i++){
          if(table[i] == max2)
             printf("%d \n", i);
      }
  }
  return 0;
}
```

Άσκηση 14

Πρώτοι αριθμοί (prime numbers) είναι εκείνοι που έχουν ως γνήσιους διαιρέτες τον εαυτό τους και την μονάδα. Ο αλγόριθμος του Ερατοσθένη βρίσκει όλους τους πρώτους αριθμούς που είναι μικρότεροι ή ίσοι ενός δοθέντος αριθμού Ν. Να γραφεί πρόγραμμα σε γλώσσα C που θα εμφανίζει όλους τους πρώτους αριθμούς μεταξύ 1 και 1000. Για την εύρεση να χρησιμοποιηθεί ο αλγόριθμος του Ερατοσθένη που περιγράφεται στη συνέχεια υπό μορφή βημάτων.

- Δημιουργήστε μια λίστα των αριθμών (δηλ. 1-Ν)
- Ο αριθμός 2 είναι πρώτος αλλά όχι τα πολλαπλάσιά του (4,6,8…). Διαγράψτε όλα τα πολλαπλάσια του 2 (δηλ. 4,6,8,…)
- Βρείτε τον πρώτο κατά σειρά αριθμό που απέμεινε στη λίστα (δηλ. 3,…) μετά από αυτούς που έχουν διαγραφεί, ελέγξτε αν είναι πρώτος και διαγράψτε όλα τα πολλαπλάσιά του.
- Επαναλάβατε το βήμα 3 μέχρι να βρεθεί ο πρώτος ακέραιος αριθμός που δεν έχει απορριφθεί και το τετράγωνό του είναι μεγαλύτερο του Ν (συνθήκη τερματισμού).
- Όλοι οι αριθμοί που απομένουν στη λίστα είναι οι πρώτοι αριθμοί μεταξύ 2 και Ν.

Άσκηση 14 - Λύση


```c
#include <stdio.h>
#include <stdbool.h>

int main(){

    bool prime[1000];

    for(int i = 0; i <= 1000; i++)
        prime[i] = true;

    for(int i = 2; i * i <= 1000; i++){
        if(prime[i] == true){
            for(int j = i * i; j <=1000; j +=i)
                prime[j] = false;
        }
    }

    for(int i = 1; i <=1000; i++){
        if(prime[i] == true)
            printf("%d \n", i);
    }

    return 0;
}
```

Άσκηση 15

Να γραφεί πρόγραμμα σε γλώσσα C που θα εισάγει σε μονοδιάστατο πίνακα 300 θέσεων τις ποσότητες παραγωγής ενός προϊόντος για ένα έτος.
Κατά σειρά σε κάθε 25 θέσεις αντιστοιχεί ένας μήνας του έτους δηλ. οι θέσεις 1-25 αντιστοιχούν στον Ιανουάριο, οι θέσεις 26-50 στον Φεβρουάριο κλπ.
Να γίνει επίσης εισαγωγή της επιθυμητής μηνιαίας ποσότητας παραγωγής για κάθε έναν από τους 12 μήνες. Το πρόγραμμα θα βρίσκει και θα εμφανίζει:

- σε ποια ημέρα κάθε μήνα η συνολική μηνιαία ποσότητα παραγωγής υπερβαίνει την επιθυμητή
- ποια είναι η μηνιαία πλεονάζουσα παραγωγή.

Η εισαγωγή των στοιχείων στους πίνακες μπορεί να γίνει με τυχαίους αριθμούς.

Άσκηση 15 - Λύση (1/3)


```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <stdbool.h>

int main(){

    int products[300], months[12], lower, upper, random_x, sum, prev_m;
    bool print;

    srand(time(0));
    lower = 1000;
    upper = 3000;

    for(int i = 0; i <12; i++){
        random_x = rand() % (upper - lower + 1) + lower;
        months[i] = random_x;
    }
```

Άσκηση 15 - Λύση (2/3)


```
lower = 100;
upper = 200;
for(int i = 0; i <300; i++){
    random_x = rand() % (upper - lower + 1) + lower;
    products[i] = random_x;
}

sum = 0;
prev_m = 0;
print = true;
```

Άσκηση 15 - Λύση (3/3)


```
    for(int i = 0; i < 300; i++){
        int m = i / 25;
        int d = i % 25;
        if(prev_m == m){
            sum += products[i];
            if(sum > months[m] && print == true){
               printf("Υπέρβαση - Μήνας: %d, Ημέρα: %d \n", m+1, d+1);
               print = false;
            }
        }
        else{
            printf("Πλεόνασμα - Μήνας: %d, Ποσό: %d \n", m, sum - months[m-1]);
            prev_m = m; sum = 0;
            print = true;
        }
    }
    printf("Πλεόνασμα - Μήνας: 12, Ποσό: %d \n", sum - months[11]);
    return 0;
}
```

Άσκηση 16

Να γραφεί πρόγραμμα σε γλώσσα C που θα εισάγει σε δύο μονοδιάστατους πίνακες N ζεύγη ακεραίων αριθμών (N = γνωστό), έστω M, P που αντιστοιχούν στον αριθμό μήνα M (αποδεκτές τιμές 1-12) και σε μία ποσότητα παραγωγής P που παρήχθη τον μήνα M (P>0 και P>>N).

Για κάθε μήνα υπάρχουν περισσότερα από ένα και άγνωστα σε πλήθος ζεύγη τιμών.
Τα δεδομένα εισάγονται με τυχαία σειρά, δηλαδή δεν εισάγονται κατά σειρά δεδομένα του μήνα 1 μετά του μήνα 2 κλπ.

Να βρεθεί σε ποιόν μήνα του έτους ξεκινώντας από την αρχή του έτους η συνολική παραγωγή υπερβαίνει μια δεδομένη τιμή X.

Άσκηση 16 - Λύση (1/3)


```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define N 100
#define X 50000

int main(){

    int months[N], product[N], lower, upper, random_x, s;

    srand(time(0));
    lower = 1;
    upper = 12;

    for(int i = 0; i <N; i++){
        random_x = rand() % (upper - lower + 1) + lower;
        months[i] = random_x;
    }
```

Άσκηση 16 - Λύση (2/3)


```
lower = 500;
upper = 1000;

for(int i = 0; i <N; i++){
    random_x = rand() % (upper - lower + 1) + lower;
    product[i] = random_x;
}

int sum[13] = {0};

for(int i = 0; i <N; i++){
    int m = months[i];
    sum[m] += product[i];
}
```

Άσκηση 16 - Λύση (3/3)


```
    s = 0;

    for(int i = 1; i < 13; i++){
        s += sum[i];
        if(s > X){
            printf("Η παραγωγή θα ξεπεράσει την τιμή %d, το μήνα: %d \n", X, i);
            break;
        }
    }
    if(s <= X)
       printf("Η παραγωγή δεν ξεπέρασε την τιμή %d \n", X);

    return 0;
}
```

Άσκηση 17

Η παραγωγή ενός προϊόντος για Ν περιόδους παραγωγής (Ν = γνωστό) καταχωρείται σε ένα μονοδιάστατο πίνακα ακεραίων θετικών αριθμών Α. Να γραφεί πρόγραμμα σε γλώσσα C που :

- θα εισάγει στον πίνακα Α τα στοιχεία πραγματοποιώντας όλους τους απαραίτητους ελέγχους ώστε οι αριθμοί να είναι θετικοί.
- θα δημιουργεί στη συνέχεια ένα νέο πίνακα Β που θα περιέχει στις αντίστοιχες θέσεις την τρέχουσα μέση τιμή των στοιχείων που έχουν εισαχθεί.
- θα βρίσκει και θα εμφανίζει πόσες και ποιες περιόδους η παραγωγή διαφέρει κατά μέγιστο 10% της τρέχουσας μέσης τιμής της ίδιας περιόδου.

ΠΑΡΑΔΕΙΓΜΑ για Ν=6

Περίοδος	1	2	3	4	5	6
Παραγωγή	100	80	90	80	200	170
Τρέχουσα μέση τιμή	100	90	90	87.5	110	120
Διαφορά <10%	ναι	όχι	ναι	ναι	όχι	όχι

Άσκηση 17 - Λύση (1/2)


```c
#include <stdio.h>
#define N 6

int main(){
     int A[N], count;
     float B[N], sum, metaboli;

    for(int i = 0; i < N; i++){
        do{
            printf("Δώσε την παραγωγή για την  %dη περίοδο: ", i+1);
            scanf("%d", &A[i]);
        } while (A[i] <= 0);
    }

    sum = 0;
    for(int i = 0; i < N; i++){
        sum += A[i];
        B[i] = sum / (i + 1);
    }
```

Άσκηση 17 - Λύση (2/2)


```
    printf("Η παραγωγή διαφέρει κατά μέγιστο 10%% της τρέχουσας μέσης τιμής, τις περιόδους: ");

    count = 0;

    for(int i = 0; i < N; i++){
        metaboli = (A[i] - B[i])/A[i] * 100;
        if (fabs(metaboli) < 10){
            printf("%d ", i+1);
            count++;
        }
    }
    printf("\nΠλήθος περιόδων: %d \n", count);

    return 0;
}
```

Άσκηση 18

Η χρονική διάρκεια λειτουργίας και η χρονική διάρκεια συντήρησης μιας εργαλειομηχανής (σε ώρες) είναι γνωστές για κάθε μια από Ν χρονικές περιόδους (Ν = γνωστό). Να γραφεί πρόγραμμα σε γλώσσα C που θα εισάγει τα στοιχεία αυτά σε δύο μονοδιάστατους πίνακες. Ο υπεύθυνος παραγωγής επιθυμεί η συνολική χρονική διάρκεια συντήρησης να μην υπερβαίνει το 15% της συνολικής χρονικής διάρκειας λειτουργίας της εργαλειομηχανής. Για τους λόγους αυτούς θέλει να γνωρίζει ποια είναι για κάθε χρονική περίοδο η τρέχουσα συνολική χρονική διάρκεια λειτουργίας και η τρέχουσα συνολική χρονική διάρκεια συντήρησης. Το πρόγραμμα θα πρέπει να εμφανίζει τα δεδομένα καθώς και τις τιμές που υπάρχουν στο παράδειγμα που ακολουθεί, μέχρι εκείνη τη χρονική περίοδο στην οποία η συνολική χρονική διάρκεια συντήρησης γίνεται για πρώτη φορά μικρότερη του 15% της συνολικής χρονικής διάρκειας λειτουργίας.

Παράδειγμα:

ΧΡΟΝΙΚΗ ΠΕΡΙΟΔΟΣ	1	2	3	**4**	5	
Λειτουργία	150	110	140	130	…	
Τρέχουσα συνολική λειτουργία	150	260	400	530		
Συντήρηση	28	22	20	5	…	
Τρέχουσα συνολική συντήρηση	28	50	70	75		
% συνολ. συντήρηση / συν. λειτουργία	18.6	19.2	17.5	**14.1**		

Άσκηση 18 - Λύση (1/2)


```c
#include <stdio.h>

#define N 5

int main(){
    int operation[N], maintenance[N];
    float total_oper, total_maint, pososto;

    for(int i = 0; i < N; i++){
        printf("Χρόνος λειτουργίας για την %dη περίοδο: ", i+1);
        scanf("%d", &operation[i]);
        printf("Χρόνος συντήρησης για την %dη περίοδο: ", i+1);
        scanf("%d", &maintenance[i]);
    }

    total_oper = 0;
    total_maint = 0;
```

Άσκηση 18 - Λύση (2/2)


```
    for(int i = 0; i < N; i++){
       total_oper += operation[i];
       total_maint += maintenance[i];
       pososto = total_maint / total_oper * 100;

       printf("Περίοδος: %d \t  Λειτουργία: %d \t Τρέχ. συν. λειτουργία: %.0f \t",
             i+1, operation[i], total_oper);
       printf("Συντήρηση: %d \t Τρέχ. συν. συντήρηση: %.2f \t Ποσοστό: %.2f \n",
             maintenance[i], total_maint, pososto);

       if(pososto < 15)
          break;
    }

   return 0;
}
```

Άσκηση 19

Η ποσότητα παραγωγής ενός προϊόντος και η ποσότητα του προϊόντος που απορρίπτεται από τον ποιοτικό έλεγχο είναι γνωστές για Ν σε πλήθος ώρες (Ν = γνωστό). Η μονάδα παραγωγής του προϊόντος επιθυμεί η συνολική ποσότητα που απορρίπτεται να μην υπερβαίνει το 20% της συνολικής ποσότητας παραγωγής για όλη την χρονική περίοδο των Ν ωρών. Ωστόσο διάφορα γεγονότα δεν επιτρέπουν στην παραγωγή να επιτύχει από την πρώτη ώρα λειτουργίας τον επιθυμητό στόχο. Για τους λόγους αυτούς ο προϊστάμενος της παραγωγής θέλει να γνωρίζει ποιο είναι για κάθε ώρα το ποσοστό % απόρριψης της τρέχουσας συνολικής παραγωγής.
Να γραφεί πρόγραμμα σε γλώσσα C που θα υλοποιεί τα παρακάτω:

1. θα εισάγει σε δύο μονοδιάστατους πίνακες ακεραίων θετικών αριθμών Ν τιμές που αφορούν την ποσότητα παραγωγής και την αντίστοιχη απορριπτόμενη ποσότητα για Ν ώρες παραγωγής. Κατά την εισαγωγή πρέπει να γίνονται οι απαραίτητοι έλεγχοι εγκυρότητας τιμών.
2. θα εμφανίζει στην οθόνη τις τιμές που έχουν εισαχθεί καθώς και τις υπολογιζόμενες τιμές που υπάρχουν στο παρακάτω παράδειγμα μέχρι την ώρα (αν αυτή υπάρχει) κατά την οποία η συνολική απορριπτόμενη ποσότητα παραγωγής γίνεται για πρώτη φορά μικρότερη του 20% της συνολικής ποσότητας παραγωγής.

Άσκηση 19

Παράδειγμα:

ΩΡΑ	1	2	3	4	5	6	7	…
Ποσότητα παραγωγής	100	150	140	80	110			…
Τρέχουσα συνολ. ποσότητα παραγωγής	100	250	390	470	580			
Απορριπτόμενη ποσότητα	25	45	35	5	3			…
Τρέχουσα συνολ. απορ. ποσότητα παραγωγής	25	70	105	110	113			
ποσοστό απόρριψης %	25	28	27	23	19.5			

Άσκηση 19 - Λύση (1/2)


```c
#include <stdio.h>

#define N 5

int main(){
    int production[N], error[N];
    float total_prod, total_error, pososto;


    for(int i = 0; i < N; i++){
        printf("Ποσότητα παραγωγής για την %dη ώρα: ", i+1);
        scanf("%d", &production[i]);
        printf("Απορριπτόμενη ποσότητα για την %dη ώρα: ", i+1);
        scanf("%d", &error[i]);
    }
```

Άσκηση 19 - Λύση (2/2)


```
    total_prod = 0;
    total_error = 0;

    for(int i = 0; i < N; i++){
        total_prod += production[i];
        total_error += error[i];
        pososto = total_error / total_prod * 100;
        printf("Ώρα: %d \t Ποσότ. Παραγ.: %d \t Τρέχουσα συν. ποσότ.: %.0f \t",
              i+1, production[i], total_prod);
        printf("Απόρ. Ποσότ.: %d \t Τρέχουσα απόρ. ποσότ: %.0f \t Ποσοστό: %.2f \n",
              error[i], total_error, pososto);

        if(pososto < 20)
           break;
    }
    return 0;
}
```

Ταξινόμηση με εισαγωγή


```c
#include <stdio.h>
#define N 8

int main(){
    int j, key;
    int arr[] = { 4, 3, 2, 10, 12, 1, 5, 6 };
    for (int i = 1; i < N; i++) {
        key = arr[i];
        j = i - 1;
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
    }
     for(int i = 0; i < N; i++)
        printf("%d \n", arr[i]);
    return 0;
}
```

Άσκηση 24

Να γίνει εισαγωγή τιμών σε μονοδιάστατο πίνακα ακεραίων θετικών αριθμών στην περιοχή [1-999], N στοιχείων (N = γνωστό). Η εισαγωγή των αριθμών μπορεί να γίνει με οποιονδήποτε τρόπο (εντολή scanf, χρήση συνάρτησης δημιουργίας τυχαίων αριθμών), αρκεί να υπάρχει έλεγχος εγκυρότητας τιμών.
Να δημιουργηθεί νέος πίνακας που θα περιλαμβάνει τις διαφορές των στοιχείων του αρχικού πίνακα, ανά ζεύγη, κατ' απόλυτη τιμή, ως εξής (διαφορά του 1ου με το 2ο στοιχείο, του 3ου στοιχείου με το 4ο στοιχείο κ.ο.κ.), ταξινομημένες σε αύξουσα διάταξη, χωρίς να χρησιμοποιηθεί διαδικασία ταξινόμησης.
Αν ο αριθμός N είναι περιττός τότε θα υπολογιστούν οι διαφορές για τα πρώτα N-1 στοιχεία.

<u>Παράδειγμα</u>:

Αρχικός πίνακας , N=10 :

37	71	19	25	53	19	8	5	77	24

Διαφορές: $|37 - 71| = 34$, $|19 - 25| = 6$, $|53 - 19| = 34$, $|8 - 5| = 3$, $|77 - 24| = 53$,

Νέος πίνακας σε αύξουσα διάταξη:

3	6	34	34	53

Άσκηση 24 - Λύση


```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define N 10

int main(){

    int A[N], random_x, lower, upper, j, current, k;

    srand(time(0));
    lower = 1;
    upper = 999;

    for(int i = 0; i <N; i++){
        random_x = rand() % (upper - lower + 1) + lower;
        A[i] = random_x;
    }
```

```c
    int new_A[N/2] = {0};
    j = 0;
    for (int i = 0; i < N; i += 2){
       current = abs(A[i] - A[i+1]);

       k = j-1;
       while(k >= 0 && new_A[k] > current){
           new_A[k+1] = new_A[k];
           k--;
       }
       new_A[k+1] = current;
       j++;
    }

    for(int i = 0; i < N/2; i++)
       printf("%d \n", new_A[i]);

    return 0;
}
```