Προγραμματισμός Υπολογιστών

Διαχείριση πινάκων

Νικόλαος Ζ. Ζάχαρης
Καθηγητής

Πανεπιστήμιο Δυτικής Αττικής
Τμήμα Μηχανικών Πληροφορικής και Υπολογιστών

Πίνακας

Σε αρκετά προγράμματα χρειάζεται να αποθηκεύσουμε ομοειδή αντικείμενα, π.χ τις βαθμολογίες μιας τάξης, τις θερμοκρασίες μιας εβδομάδας κ.λπ. Όλα τα παραπάνω αντικείμενα είναι του ιδίου τύπου, για παράδειγμα πραγματικοί αριθμοί (float), και με τις μέχρι τώρα γνώσεις μας για όσα δεδομένα θέλουμε να αποθηκεύσουμε θα πρέπει να χρησιμοποιήσουμε το αντίστοιχο πλήθος μεταβλητών.

Για την αποθήκευση των 7 θερμοκρασιών θα πρέπει να χρησιμοποιήσουμε 7 μεταβλητές, όπως παρακάτω :

```
float wk1, wk2, wk3, wk4, wk5, wk6, wk7;
float mesos = (wk1 + wk2 + wk3 + wk4 + wk5 + wk6 + wk7) / 7.0;
```

Πρακτικά, δεν έχουμε πρόβλημα με την δήλωση πολλών μεταβλητών αλλά όταν το πλήθος μεγαλώνει τότε και η διαχείριση τους γίνεται αρκετά δύσκολη.

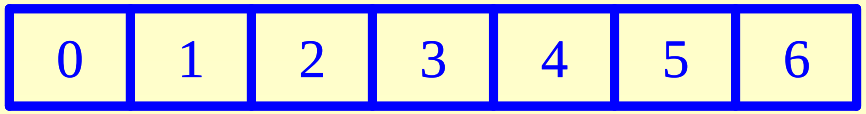
Η C όπως και οι άλλες γλώσσες προγραμματισμού για να αντιμετωπίσουν το παραπάνω πρόβλημα διαθέτουν την δομή του πίνακα. Η δήλωση ενός πίνακα στην γλώσσα C γίνεται ως εξής :

Τύπος_Δεδομένων Όνομα_Μεταβλητής[Πλήθος_Στοιχείων]

Στο προηγούμενο πρόβλημα, για να δηλώσουμε ένα πίνακα δεκαδικών αριθμών και χωρητικότητας επτά θέσεων που θα χρησιμοποιήσουμε για την αποθήκευση των επτά θερμοκρασιών, γράφουμε :

float wks[7];

Ένας πίνακας είναι ένα σύνολο από ομοειδή αντικείμενα τα οποία είναι αποθηκευμένα σε συνεχόμενες θέσεις. **Η αρίθμηση των θέσεων ξεκινά από το μηδέν μέχρι Πλήθος_Στοιχείων – 1.**



Η ανάγνωση και η απόδοση των τιμών σε μια θέση ένος πίνακα γίνεται όπως και στις υπόλοιπες μεταβλητές, με την μόνη διαφορά ότι θα πρέπει να δηλώνουμε και την αντίστοιχη θέση του πίνακα που αναφερόμαστε :

```
wks[0] = 1;    // Απόδοση της τιμής 1 στη θέση 0 του πίνακα

wks[2] = 32; // Απόδοση της τιμής 32 στη θέση 2 του πίνακα

float x;

x = wks[2] + wks[0]; // Απόδοση της τιμής 33 στη μεταβλητή x
```

Η αρχικοποίηση ενός πίνακα μπορεί να γίνει με έναν από τους παρακάτω τρόπους :

```
float wks[7] = {0.0, 3.4, 4.4, 7.9, 1.2, 6.4, 7.3};      είτε

float wks[] = {0.0, 3.4, 4.4, 7.9, 1.2, 6.4, 7.3};
```

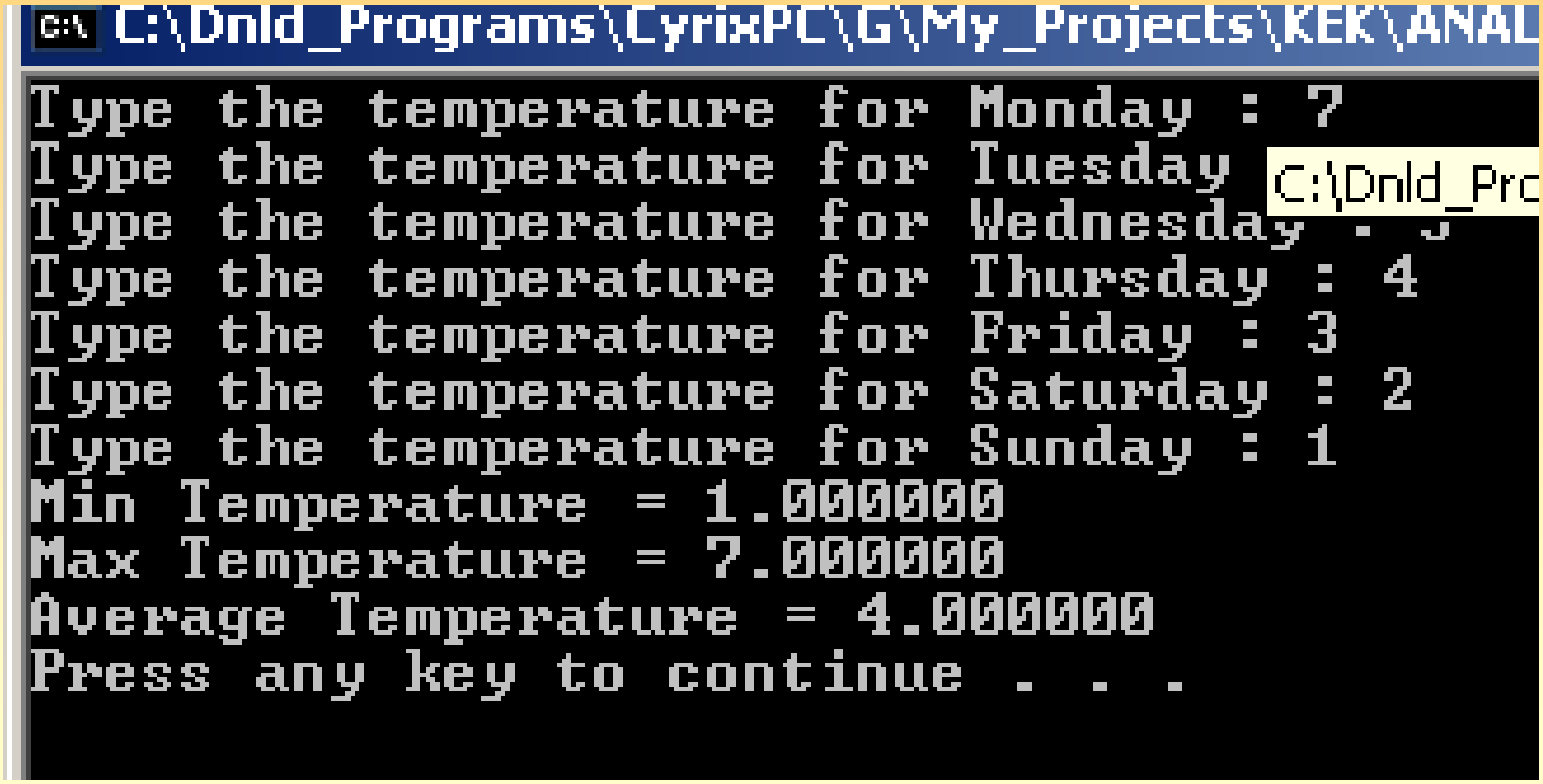
Σε ένα πίνακα χωρητικότητας 7 πραγματικών αριθμών να αναθέσετε σε όλες τις θέσεις του τιμές από το πληκτρολόγιο, να τις εμφανίσετε σε μια γραμμή στην οθόνη.

```
int main(int argc, char *argv[])  {
  int i;
  float pin[7];
  for (i = 0; i < 7; i++)   {
     printf("Type a float number : ");
     scanf("%f", &pin[i]);
  }

  printf("The contents of the array\n");
   for (i = 0; i < 7; i++)  {
      printf("%.2f\t", pin[i]);
  }
  return 0;
}
```

Άσκηση

Δημιουργείστε ένα πρόγραμμα που διαβάζει τις θερμοκρασίες μιας εβδομάδας και υπολογίζει την μικρότερη και μεγαλύτερη θερμοκρασία καθώς και το μέσο όρο των θερμοκρασιών. (Χρήση πίνακα, switch, continue)



```
int main(int argc, char *argv[]){
  float wks[7], min, max, mesos;
  int i;
  for(i = 0; i < 7; i++)  {
    printf("Type the temperature for ");
    switch(i)  {
    case 0 : printf("Monday : "); break;
    case 1 : printf("Tuesday : "); break;
    case 2 : printf("Wednesday : ");break;
    case 3 : printf( "Thursday : "); break;
    case 4 : printf( "Friday : "); break;
    case 5 : printf( "Saturday : "); break;
    case 6 : printf( "Sunday : "); break;
    }
    scanf("%f", &wks[i]);
  }

  min = wks[0];
  max = wks[0];
  mesos = wks[0];

  for(i = 1; i < 7; i++)  {
    mesos = mesos + wks[i];

      if(wks[i] < min)   {
        min = wks[i];
        continue;
      }

      if(wks[i] > max)
      {
        max = wks[i];
      }
   }

printf("Min Temp = %f\n", min);
printf("Max Temp = %f\n", max);
printf("Aver Temp = %f\n",
mesos/7.0);
}
```

Νικόλαος Ζ. Ζάχαρης

Άσκηση

Δημιουργείστε ένα πρόγραμμα που διαβάζει έναν ακέραιο αριθμό και εμφανίζει το πλήθος εμφάνισης του αριθμού σε ένα προκαθορισμένο πίνακα ακεραίων.

```
int grades[7] = {0, 4, 4, 5, 3, 8, 9};
```

```
Αρχή Αλγόριθμου
thesi = 1
Emfanisi = 0
Διάβασε Χ
Ενώ thesi <= 7 Επανέλαβε
    Αν Pinakas[thesi] = Χ  Τότε
        Emfanisi = Emfanisi  + 1
    Τέλος Αν
thesi = thesi + 1
Τέλος Επανάληψης
Εκτύπωσε “Ο Χ εμφανίζεται “ +  Emfanisi + “ φορές.”
Τέλος Αλγόριθμου
```

```
#define plithos 7

int main(int argc, char *argv[]){
   int Pinakas[plithos] = {0, 4, 4, 5, 3, 8, 9};
// int Pinakas[] = {0, 4, 4, 5, 3, 8, 9};  // alternative statement
  int X, thesi = 0, emfanisi = 0;

  printf("Type an integer number : ");
  scanf("%d", &X);

  for(thesi = 0; thesi < plithos; thesi++)  {
    if(Pinakas[thesi] == X)    {
       emfanisi++;
    }
  }
  printf("The %d appears %d times. ", X, emfanisi);
}
```

Η δήλωση 2 ή περισσοτέρων διαστάσεων σε ένα πίνακα στην γλώσσα C γίνεται όπως παρακάτω :

Τύπος_Δεδομένων Ονομα_Πίνακα[ΠρώτηΔιάσταση][Δεύτερη][...];

Για να δηλώσω ένα πίνακα ακεραίων με το όνομα **Pin** και τρείς διαστάσεις εκ των οποίων η πρώτη έχει 3 στοιχεία, η δεύτερη 4 στοιχεία και η τρίτη 7 θα πρέπει να γράψω :

```
int Pin[3][4][7];
```

Για να δηλώσω ένα πίνακα δεκαδικών με το όνομα **Grades** και δύο διαστάσεις εκ των οποίων η πρώτη έχει 5 στοιχεία, η δεύτερη 5 στοιχεία θα πρέπει να γράψω :

```
float Grades[5][5];
```

Πρόβλημα

Δηλώστε ένα πίνακα ακεραίων δύο διαστάσεων με 3 στοιχεία σε κάθε διάσταση και γεμίστε τις θέσεις από το πληκτρολόγιο.

```
int pin[3][3];
```

Απάντηση

Για να γεμίσουμε το πίνακα θα χρειαστούμε δύο επαναλήψεις : μία που θα διασχίσει όλες τις γραμμές και για την κάθε γραμμή, μία επιπλέον επανάληψη, που θα διασχίζει όλες τις στήλες. Μπορεί να γίνει και το αντίστροφο δηλαδή η πρώτη επανάληψη να διασχίζει τις στήλες και η δεύτερη τις γραμμές

```
for(i = 0; i < 3; i++)  {
        for(j = 0; j < 3; j++) {
            scanf("%d", &pin[i][j]);
         }
}
```

Πρόβλημα

Δηλώστε ένα πίνακα ακεραίων δύο διαστάσεων με 3 στοιχεία σε κάθε διάσταση και διαβάστε αριθμούς από το πληκτρολόγιο, για τις 2 πρώτες γραμμές και στήλες. Έστω ότι διαβάζω από το πληκτρολόγιο τους αριθμούς 4, 5, 2, 6, τότε ο πίνακας θα έχει την παρακάτω μορφή

4	5	
2	6	

Γεμίστε τις κενές θέσεις σε κάθε γραμμή και στήλη με το αντίστοιχο άθροισμα των στοιχείων.

Άθροισμα στηλών

4 + 2 = 6

5 + 6 = 11

4	5	9
2	6	8
6	11	10

Άθροισμα γραμμών

4 + 5 = 9

2 + 6 = 8

Διαγώνιος 4 + 6 = 10

Νικόλαος Ζ. Ζάχαρης

Άθροισμα γραμμών

```
int main(int argc, char *argv[]) {
int i,j, x, t;
int pin[3][3];


for(i =0; i< 2; i++)  {
   t = 0;
   for(j = 0; j < 2; j++)    {
      scanf("%d", &pin[i][j]);
      t = t + pin[i][j];
   }
   pin[i][j] = t;
 }
```

Άθροισμα στηλών

```
x = 0;
 for(i = 0; i < 2; i++)  {
   t = 0;
   for(j = 0; j < 2; j++)  {
      t = t + pin[j][i];
      if (i == j)
         x = x + pin[j][i];
   }
   pin[j][i] = t;
 }
 pin[2][2] = x;
return 0; }
```

Ταξινόμηση και αναζήτηση

Σε πάρα πολλές εφαρμογές έχουμε αποθηκεύσει δεδομένα σε ένα πίνακα και θέλουμε να μάθουμε αν ένα στοιχείο εμπεριέχεται στο πίνακα ή όχι. Σε μερικές περιπτώσεις ενδιαφερόμαστε και για το πλήθος εμφάνισης ή τη θέση του στο πίνακα.

```
#define size 7

int main(int argc, char *argv[]){
   int Pinakas[plithos] = {0, 4, 4, 5, 3, 8, 9};
   int i, X, thesi = -1;

  printf("Type an integer number : ");
  scanf("%d", &X);

   for(i = 0; i < plithos; i++)  {
     if(Pinakas[i] == X)    {
       thesi = i;
       break;
    }
  }
```

```
 if(thesi < 0)  {
      printf("The %d not found.", X);
  }
  else
  {
     printf("The %d found at %d
                    position.", X, thesi);
  }

  return 0;
}
```

Νικόλαος Ζ. Ζάχαρης

Η σειριακή αναζήτηση είναι αρκετά εύκολη στην υλοποίηση αλλά είναι αρκετά αργή με αποτέλεσμα να μην ενδείκνυται για πίνακες με μεγάλο πλήθος στοιχείων. Επίσης και για μικρούς πίνακες δεν πρέπει να χρησιμοποιείται όταν η λειτουργία της αναζήτησης είναι αρκετά συχνή.

Η λειτουργία της αναζήτησης μέσα σε ένα πίνακα θα μπορούσε να είναι πολύ γρήγορη αν τα στοιχεία του πίνακα δεν ήταν σε τυχαία διάταξη αλλά ήταν ταξινομημένα. Με τον όρο ταξινόμηση εννοούμε την διάταξη των στοιχείων του πίνακα με ένα συγκεκριμένο τρόπο.

Υπάρχουν δύο είδη διάταξης :

Αύξουσα

-1	3	5	12	17

Φθίνουσα

17	12	5	3	1

Αν τα στοιχεία του πίνακα ήταν ταξινομημένα τότε με έναν απλό έλεγχο θα μπορούσαμε να πούμε αν υπάρχει περίπτωση να εμπεριέχεται το ζητούμενο στοιχείο μέσα στο πίνακα ή όχι.

Υπάρχουν αρκετές μέθοδοι για την ταξινόμηση ενός πίνακα, όπως η μέθοδος της Ευθείας Ανταλλαγής (Straight Exchange), της Φυσαλίδας (BubbleSort), της Γρήγορης Αναζήτησης (Quick Sort), κ.λπ. Οι διάφορές μεταξύ τους είναι στην ταχύτητα ταξινόμησης των στοιχείων του πίνακα.

Ταξινόμηση Φυσσαλίδας

Η μέθοδος ξεκινά από την παρατήρηση ότι αν είχαμε ένα πίνακα με δύο θέσεις το μόνο που έπρεπε να κάνουμε είναι να συγκρίνουμε τα στοιχεία μεταξύ τους και αν δεν ήταν στην σωστή θέση – ανάλογα με το είδος της ταξινόμησης – τότε θα έπρεπε να τα αντιμεταθέσουμε. Έστω ο παρακάτω πίνακας που θέλουμε να τον ταξινομήσουμε σε αύξουσα διάταξη

7	5

Σε αυτή την περίπτωση μια απλή σύγκριση είναι αρκετή μεταξύ του πρώτου και του δεύτερου στοιχείου. Όμως αν είχαμε τρία στοιχεία τότε με την πρώτη

7	5	4

σύγκριση θα αλλάζε θέση το 5 με το 7. Αυτό που πετύχαμε είναι να μετακινήσουμε το 7 πιο κοντά στην σωστή θέση. Αν κάνουμε ακόμα μια σύγκριση το 7 με το 4 θα πετύχουμε το επιθυμητό αποτέλεσμα πού είναι

5	7	4

όχι να ταξινομήσουμε το πίνακα αλλά να φέρουμε τον πραγματικά μεγαλύτερο αριθμό στην σωστή θέση.

5	4	7

Νικόλαος Ζ. Ζάχαρης

Την προηγούμενη διαδικασία μπορούμε να την περιγράψουμε ως εξής :

Σύγκρινε κάθε θέση του πίνακα με την επόμενη της.

Αν τα στοιχεία στις αντίστοιχες θέσεις δεν είναι στην σωστή σειρά, τότε κάνε αντιμετάθεση των στοιχείων μεταξύ των δύο θέσεων.

Συνέχισε την διαδικασία μέχρι το τέλος του πίνακα - 1.

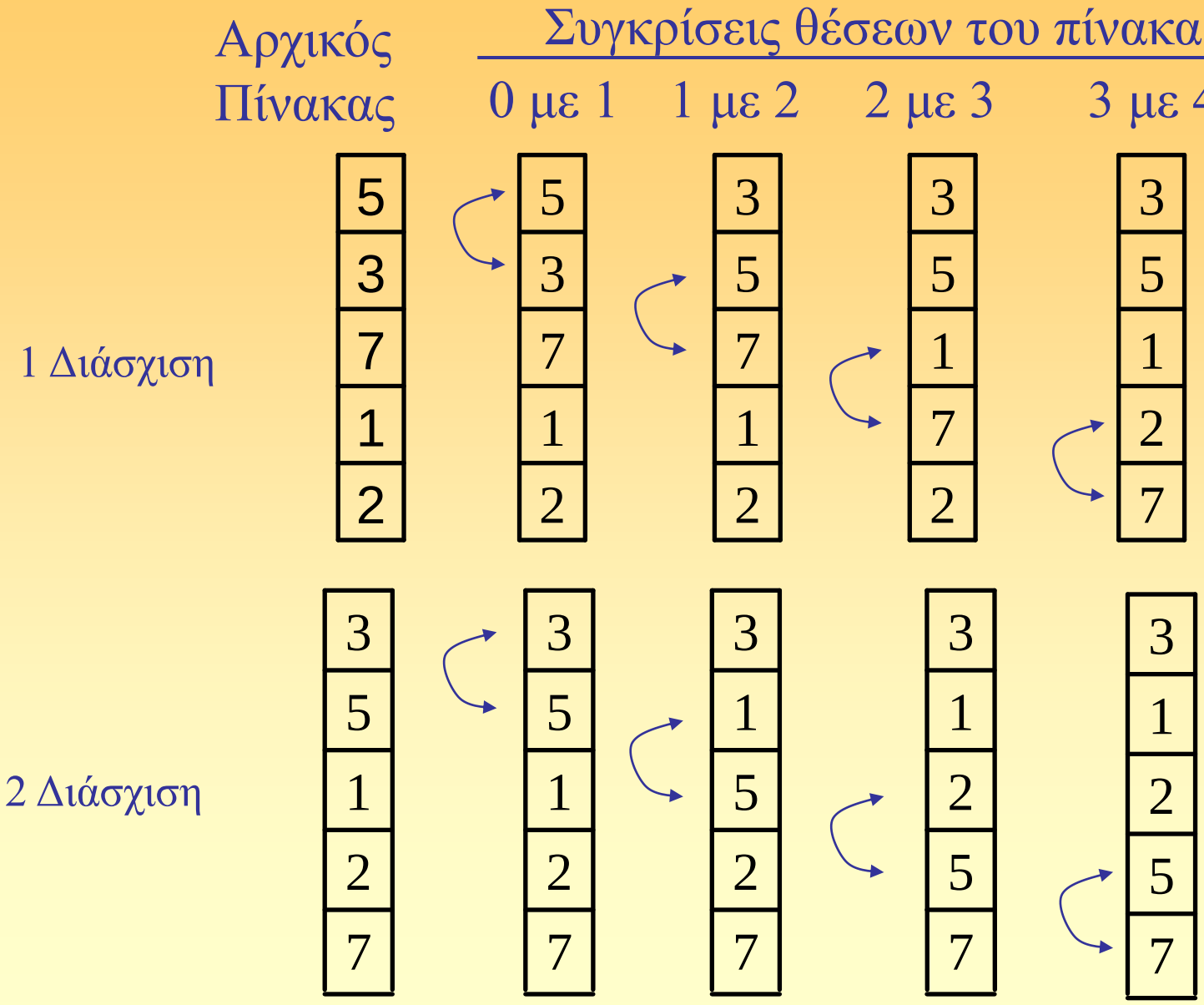
Στο προηγούμενο παράδειγμα εφαρμόζοντας την παραπάνω διαδικασία καταλήξαμε, στην παρακάτω μορφή:

5	4	7

Ο πίνακας δεν είναι ταξινομημένος, ακόμη. Αυτό που χρειάζεται είναι, αφού κάναμε την πρώτη διάσχιση και φέραμε το μεγαλύτερο στοιχείο στην σωστή θέση να επαναλάβουμε την διαδικασία ώστε να φέρουμε το αμέσως μικρότερο στοιχείο στην σωστή θέση και ούτω καθ' εξής. Δηλαδή να κάνουμε συγκρίσεις από την αρχή, το 5 με το 4 και να τα αντιμεταθέσουμε.

4	5	7

Επίδειξη της λειτουργίας της BubbleSort σε πίνακα με 5 θέσεις.



Νικόλαος Ζ. Ζάχαρης

Αλγόριθμος BubbleSort

```
Αρχή Αλγόριθμου BubbleSort
Για j Από 1 μέχρι Πλήθος_Στοιχείων_Πίνακα Με Βήμα 1
    Για i Από 0 μέχρι Πλήθος_Στοιχείων_Πίνακα - 1 Με Βήμα 1
        Αν Πινακας[i] > Πίνακας [i+1] Τότε
            Αντιμετάθεσε Πινακας[i], Πίνακας [i+1]
        Τέλος Αν
    Τέλος Επανάληψης
Τέλος Επανάληψης
Τέλος Αλγόριθμου
```

```
int main(int argc, char *argv[]){
  int i, j, temp, pin[5];

  for(i = 0; i < 5; i++){
    printf("Type a number : ");
    scanf("%d", &pin[i]);
  }
  printf("\nBefore Sort\n");
  for(i = 0; i < 5; i++){
    printf("%d\t", pin[i]);
  }
```

Θα πρέπει να τρέξουμε την επανάληψη για ένα στοιχείο λιγότερο 5–1 γιατί διαφορετικά θα προσπελάσουμε στοιχεία εκτός των ορίων του πίνακα.

```
 for(j = 1; j < 5; j++)
     for(i = 0; i < 4; i++)
       if(pin[i] > pin[i+1]){
          temp = pin[i];
          pin[i] = pin[i+1];
          pin[i+1] = temp;
       }

  printf("\nAfter Sort\n");
  for(i = 0; i < 5; i++){
    printf("%d\t", pin[i]);
  }

  return 0;
}
```

Νικόλαος Ζ. Ζάχαρης

Βελτιστοποίηση του αλγορίθμου (1-2)

Ο αλγόριθμός σε κάθε επανάληψη τοποθετεί το κάθε στοιχείο του πίνακα στη σωστή θέση του. Για παράδειγμα, αν κάνουμε ταξινόμηση των στοιχείων σε αύξουσα σειρά, τότε ο αλγόριθμος με τη πρώτη επανάληψη θα τοποθετήσει το μεγαλύτερο στοιχείο, στο τέλος του πίνακα.

Με τη δεύτερη επανάληψη θα τοποθετήσει το αμέσως μικρότερο στοιχείο, στη προτελευταία θέση και ούτω καθεξής.

Πριν

```
for(j = 1; j < 5; j++)
   for(i = 0; i < 4; i++)
     if(pin[i] > pin[i+1]){
        temp = pin[i];
        pin[i] = pin[i+1];
        pin[i+1] = temp;
     }
```

Μετά

```
for(j = 1; j < 5; j++)
   for(i = 0; i < 5-j; i++)
     if(pin[i] > pin[i+1]){
        temp = pin[i];
        pin[i] = pin[i+1];
        pin[i+1] = temp;
     }
```

Βελτιστοποίηση του αλγορίθμου (2-2)

Αν ο πίνακας είναι ήδη ταξινομημένος τότε δεν θα "μπεί" ποτέ μέσα στο if για να κάνει αντιμετάθεση των στοιχείων του πίνακα.

```
int j = 1;
int sorted;

do{
    sorted = 1;
    for(i = 0; i < 5-j; i++) {
      if(pin[i] > pin[i+1]){
        temp = pin[i];
        pin[i] = pin[i+1];
        pin[i+1] = temp;
        sorted=0;
      }
    }
    j++;
 }while(!sorted);
```

Κάνουμε μία ατελείωτη επανάληψη όπου αρχικά θεωρούμε ότι ο πίνακας είναι ταξινομημένος (sorted=1). Αν φτάσει στη δήλωση while με τιμή 1 τότε θα σταματήσει η επανάληψη, γιατί η συνθήκη θα γίνει false (0).

Αν όμως ο πίνακας είναι αταξινόμητος τότε θα πρέπει να γίνει αντιμετάθεση των στοιχείων και θα μπει στη δήλωση ελέγχου if και η μεταβλητή sorted θα αλλάξει τιμή σε 0. Εν συνεχεία θα φτάσει στη δήλωση while με τιμή 0 και θα συνεχίσει η επανάληψη γιατί η συνθήκη είναι αληθής true (1).

Ο αλγόριθμός σταματά μόλις ταξινομηθεί ο πίνακας

Δυαδική Αναζήτηση (Binary Search)

Η σειριακή αναζήτηση εξετάζει ένα προς ένα όλα τα στοιχεία του πίνακα. Σε αυτή την περίπτωση τα στοιχεία δεν χρειάζεται να είναι διατεταγμένα με κάποια σειρά. Αν όμως ήταν διατεταγμένα τότε θα μπορούσε να επιταχύνουμε την αναζήτηση με το εξής απλό τρόπο :

Ας θεωρήσουμε ότι αναζητούμε τον αριθμό 12

Καταρχήν βρίσκουμε την μεσαία θέση του πίνακα και εξετάζουμε την τιμή που περιέχει.

-1	3	5	12	17

Μεσαία Θέση = (Αρχή + Τέλος) / 2 = (0 + 4) / 2 = 2

Ο πίνακας στην θέση 2 περιέχει την τιμή 5. Εν συνεχεία συγκρίνουμε τον αριθμό που αναζητούμε (το 12) με το 5. Αν είναι ίσοι τότε διακόπτουμε την αναζήτηση. Αν είναι μικρότερος τότε το νέο διάστημα έρευνας είναι από 0 έως Μεσαία θέση - 1.

Αν είναι μεγαλύτερος τότε το νέο διάστημα έρευνας είναι από

Μεσαία θέση + 1 έως τέλος του πίνακα

-1	3	5	12	17

$$12 > 5$$

Στην περίπτωση μας είναι μεγαλύτερος οπότε το νέο διάστημα έρευνας είναι το από θέση 3 έως 4, όπου είναι το τέλος του πίνακα

Επαναλαμβάνουμε την διαδικασία και βρίσκουμε την μεσαία θέση για το νέο διάστημα έρευνας.

Μεσαία Θέση = (Αρχή + Τέλος) / 2 = (3 + 4) / 2 = 3.5

Ακέραιο μέρος είναι το 3. Άρα η μεσαία θέση είναι η τρίτη

Ο πίνακας στην θέση 3 περιέχει την επιθυμητή τιμή 12 οπότε τερματίζει και η αναζήτηση.

Αν στο προηγούμενο παράδειγμα, που επιλέξαμε το ακέραιο μέρος του 3.5, αντί για το 3 θεωρήσουμε το 4 σαν μεσαία τιμή πάλι στο ίδιο αποτέλεσμα θα καταλήξουμε :

Ο πίνακας στην θέση 4 περιέχει την τιμή 17 η οποία είναι μεγαλύτερη από το 12 οπότε το νέο διάστημα είναι το Αρχη = 3 (από το προηγούμενο βήμα) και η Μεσαία θέση – 1, δηλαδή 4 -1 = 3

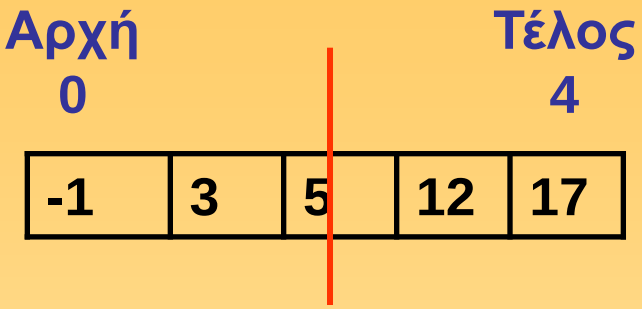
Επαναλαμβάνουμε την διαδικασία και βρίσκουμε την μεσαία θέση για το νέο διάστημα έρευνας.

Μεσαία Θέση = (Αρχή + Τέλος) / 2 = (3 + 3) / 2 = 3.

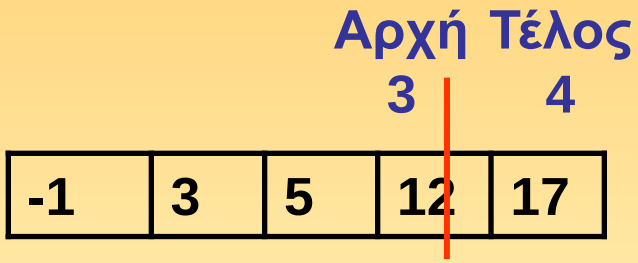
Άρα η μεσαία θέση είναι η τρίτη

Ο πίνακας στην θέση 3 περιέχει την επιθυμητή τιμή 12 οπότε τερματίζει και η αναζήτηση.

Συγκεντρωτικά

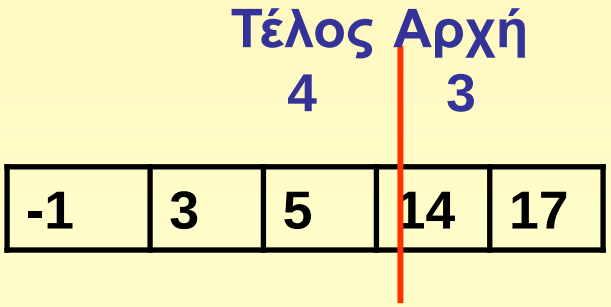


Μεσαία Θέση = (0 + 4) / 2 = 2



Μεσαία Θέση = (3 + 4) / 2 = 3.5

Ο αλγόριθμος σταματά όταν βρεθεί ο αριθμός ή όταν δεν ισχύει η συνθήκη Αρχή <= Τέλος. Για παράδειγμα αν αναζητούμε το 12 αλλά στο πίνακα στην θέση 3, αντί του 12 ήταν το 14, τότε το τέλος του νέου διαστήματος θα ήταν Μεσαία Θέση - 1



Αρχή 3 (από το προηγούμενο βήμα)

Τέλος = (Μέση – 1) = 2

Αρχή ΔΕΝ ΕΙΝΑΙ <= Τέλος οπότε το 12 δεν βρέθηκε στο πίνακα

Νικόλαος Ζ. Ζάχαρης

Υλοποίηση του BinarySearch

```
int main(int argc, char *argv[])
{
int pin[5] = { 1, 5, 7, 12, 25 };
int key = 12;
int found = -1, low = 0, high = 4;
int mid;

  while (1)
  {
     mid = (low + high) / 2;

     if(low > high)
        break;
     if(pin[mid] == key)     {
        found = mid;
        break;
     }
     else
     {
        if(pin[mid] > key)
           high = mid-1;
        else
           low = mid+1;
     }
  } // end of while

  printf("Found : %d",found);
  return 0;
}
```