ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

ΑΝΟΙΚΤΑ ακαδημαϊκά μαθήματα ΠΠ

Προγραμματισμός Η/Υ

Ενότητα 8: Ειδικά Θέματα Αλγορίθμων

Νίκος Καρακαπιλίδης, Καθηγητής
Δημήτρης Σαραβάνος, Καθηγητής
Πολυτεχνική Σχολή
Τμήμα Μηχανολόγων & Αεροναυπηγών Μηχανικών

Σκοποί ενότητας

- Κατανόηση της σύνταξης συναρτήσεων με βάση τα δεδομένα πινάκων
- Κατανόηση των μεθόδων της γραμμικής και δυαδικής αναζήτησης μιας παραμέτρου
- Κατανόηση των μεθόδων ταξινόμησης των παραμέτρων

Περιεχόμενα ενότητας

- Πέρασμα Πίνακα σε Συνάρτηση
- Προβλήματα Αναζήτησης
- Προβλήματα Ταξινόμησης

Μέρος 1ο

Πέρασμα Πίνακα σε Συνάρτηση

Πέρασμα Πίνακα σε Συνάρτηση

Πέρασμα Πίνακα σε Συνάρτηση

Πέρασμα πίνακα

Παράδειγμα:

```
int my_array[24];
my_function(my_array, 24);
```

- Δίνουμε το όνομα του πίνακα (χωρίς [ ])
- Συνήθως χρειάζεται να «περαστεί» και το μέγεθος του πίνακα

Γίνεται «πέρασμα δια αναφοράς» (call by reference)

Το όνομα του πίνακα είναι η διεύθυνση του πρώτου του στοιχείου

- Η συνάρτηση «γνωρίζει» που είναι αποθηκευμένος ο πίνακας

Πέρασμα Πίνακα σε Συνάρτηση

Πέρασμα στοιχείων πίνακα

Παράδειγμα:

```
my_other_function(my_array[3]);
```

- Δίνουμε το όνομα του στοιχείου

Γίνεται «πέρασμα δια τιμής» (call by value)

Πέρασμα Πίνακα σε Συνάρτηση

Δήλωση (ή πρωτότυπο) συνάρτησης

Παράδειγμα:

void modify_array(int b[], int array_size);

Τα ονόματα των παραμέτρων δεν είναι υποχρεωτικά στη δήλωση της συνάρτησης

- **int b[]** μπορεί να γραφεί απλά ως **int []**
- **int array_size** μπορεί να γραφεί απλά ως **int**

Επομένως, το παραπάνω παράδειγμα μπορεί να γραφεί ως:

void modify_array(int [], int);

Παράδειγμα

```
1    /* Example program
     Πέρασμα ολόκληρου πίνακα και στοιχείων πίνακα σε συναρτήσεις */
2      Passing arrays and individual array elements to functions */
3    #include <stdio.h>
4    #define SIZE 5
5
6    void modifyArray( int [], int );   /* function prototypes */
7    void modifyElement( int );
8
9    int main()
10   {
11      int a[ SIZE ] = { 0, 1, 2, 3, 4 }, i;
12
13      printf( "Effects of passing entire array call "
14              "by reference:\n\nThe values of the "
15              "original array are:\n" );
```

Παράδειγμα

```
16
17      for ( i = 0; i <= SIZE - 1; i++ )
18          printf( "%3d", a[ i ] );
19
20      printf( "\n" );
21      modifyArray( a, SIZE );                    /* call by reference */
22      printf( "The values of the modified array are:\n" );
23
24      for ( i = 0; i <= SIZE - 1; i++ )
25          printf( "%3d", a[ i ] );
26
27      printf( "\n\n\nEffects of passing array element call "
28              "by value:\n\nThe value of a[3] is %d\n", a[ 3 ] );
29      modifyElement( a[ 3 ] );                   /* call by value */
30      printf( "The value of a[ 3 ] is %d\n", a[ 3 ] );
```

Ολόκληροι πίνακες:
* Πέρασμα δια αναφοράς
* Μπορούν να τροποποιηθούν

Μεμονωμένα στοιχεία πίνακα:
* Πέρασμα δια τιμής
* Δεν μπορούν να τροποποιηθούν

Παράδειγμα

```
31      return 0;
32  }
33
34  void modifyArray( int b[], int size )      /* function definition */
35  {
36      int j;
37
38      for ( j = 0; j <= size - 1; j++ )
39          b[ j ] *= 2;
40  }
41
42  void modifyElement( int e )                /* function definition */
43  {
44      printf( "Value in modifyElement is %d\n", e *= 2 );
45  }
```

Παράδειγμα

output προγράμματος

```
Effects of passing entire array call by reference:

The values of the original array are:
  0  1  2  3  4
The values of the modified array are:
  0  2  4  6  8

Effects of passing array element call by value:


The value of a[3] is 6
Value in modifyElement is 12
The value of a[3] is 6
```

Μέρος 2ο

Προβλήματα Αναζήτησης

Προβλήματα Αναζήτησης

Γραμμική και Δυαδική Αναζήτηση

Γραμμική Αναζήτηση

Γενικά

Απλός αλγόριθμος, χρήσιμος για μικρούς και μη ταξινομημένους πίνακες

Παράδειγμα

Να γραφεί πρόγραμμα που να ψάχνει έναν πίνακα ακεραίων στοιχείων (όνομα πίνακα: **student_id**) για να βρει σε ποια θέση του περιέχεται κάποια συγκεκριμένη τιμή (έστω η τιμή **z**)

- Αν η τιμή περιέχεται σε πολλές θέσεις του πίνακα, θέλουμε να βρίσκουμε μόνο την πρώτη από αυτές

Το μέγεθος του πίνακα ορίζεται με τη σταθερά **STUDENT_NUM**

Γραμμική Αναζήτηση

Αλγόριθμος επίλυσης προβλήματος

Για κάθε στοιχείο του πίνακα

- εάν είναι ίσο με **z**
  - Φύλαξε θέση
  - Τερμάτισε την επανάληψη

Χειρότερη περίπτωση: εξέταση όλων των στοιχείων του πίνακα

```
int i;
...
for (i=0;i<STUDENT_NUM;++i)
   if (student_id[i]==z) break;
...
```

Γραμμική Αναζήτηση (χρήση συνάρτησης)

Παράδειγμα

Να γραφεί συνάρτηση που αναζητά μέσα σε έναν πίνακα ακεραίων στοιχείων **id_table** τον αριθμό μητρώου **id**. Το μέγεθος του πίνακα (αριθμός φοιτητών) περιέχεται στην παράμετρο **size**. Εάν βρεθεί ο αριθμός μητρώου που ψάχνουμε, να επιστραφεί η θέση του πίνακα που τον περιέχει. Αλλιώς, να επιστραφεί η τιμή του **size**.

```
int location_of_student(int id_table[], int size, int id)
{ int i;
    for(i=0;i<size;++i)
          if (student_id[i]==id)
                return i;
    return size;
}
```

Αναζήτηση και ενημέρωση

Παράδειγμα

Έστω δύο παράλληλοι πίνακες (ίδιο μέγεθος) όπου αποθηκεύονται οι αριθμοί μητρώου και οι βαθμοί των φοιτητών για κάποιο μάθημα (έστω **id_table** και **grade_table** αντίστοιχα)

Να γραφεί συνάρτηση που αναζητά μέσα στον πίνακα **id_table** κάποιον φοιτητή με αριθμό μητρώου **id** και στη συνέχεια ενημερώνει κατάλληλα τον πίνακα **grade_table** (μέσω κάποιας παραμέτρου **new_grade**)

Αναζήτηση και ενημέρωση

- Το μέγεθος των πινάκων (αριθμός φοιτητών) περιέχεται στην παράμετρο **size**
- Αν η ενημέρωση είναι επιτυχής, να επιστρέφεται «0». Αλλιώς (αν δεν βρεθεί ο συγκεκριμένος αριθμός μητρώου), να επιστρέφεται η τιμή «-1»

```
int
update_grade(int id_table[], float grade_table[],
                        int size, int id, float new_grade)
{
        int location = location_of_student(id_table,size,id);
        if (location == size)
                return -1;
        else
        {
                grade_table[location]= new_grade;
                return 0;
        }
}
```

Μέτρηση (χρήση const)

Παράδειγμα

Να γραφεί συνάρτηση που υπολογίζει τον αριθμό των φοιτητών που έχουν πάρει (σε κάποιο μάθημα) βαθμό από 8 και πάνω καθώς και τον αριθμό αυτών που έχουν αποτύχει (βαθμός μικρότερος του 5)

- Οι βαθμοί είναι αποθηκευμένοι σε πίνακα μεγέθους **size**

Μέτρηση (χρήση const)

```
void
grade_count(const int grade_table[], int size,
            int *above_eight, int *below_five)
{
        int i;
        *above_eight = *below_five = 0;
        for(i=0;i<size;++i)
                if (grade_table[i]>=8.0)
                        ++*above_eight;
                else if (grade_table[i]<5)
                        ++*below_five;
}
```

Σημασία χρήσης της const

- δηλώνει ότι δεν μπορεί να γίνει ανάθεση σε κάποια μεταβλητή (read only)
- Θέματα ασφάλειας
- Ιδιαίτερα χρήσιμη για παραμέτρους
- Π.χ., στο **grade_count** απαγορεύει την τροποποίηση των περιεχομένων του **grade_table**

Δυαδική αναζήτηση (binary search)

Για ταξινομημένους πίνακες

- Σε αύξουσα ή φθίνουσα σειρά

Συγκρίνει το μεσαίο στοιχείο ενός πίνακα (έστω **middle**) με το στοιχείο που ψάχνουμε (έστω **key**)

- Αν είναι ίσο, το στοιχείο βρέθηκε
- Αν **key < middle**, ψάχνει στο πρώτο μισό του πίνακα
- Αν **key > middle**, ψάχνει στο δεύτερο μισό του πίνακα
- Επανάληψη

Δυαδική αναζήτηση

Πολύ γρήγορος αλγόριθμος

- Το πολύ Ν βήματα, όπου 2^N > (αριθμός στοιχείων πίνακα)
  - Παράδειγμα: για πίνακα 30 στοιχείων χρειάζονται το πολύ 5 βήματα ($2^5 > 30$)

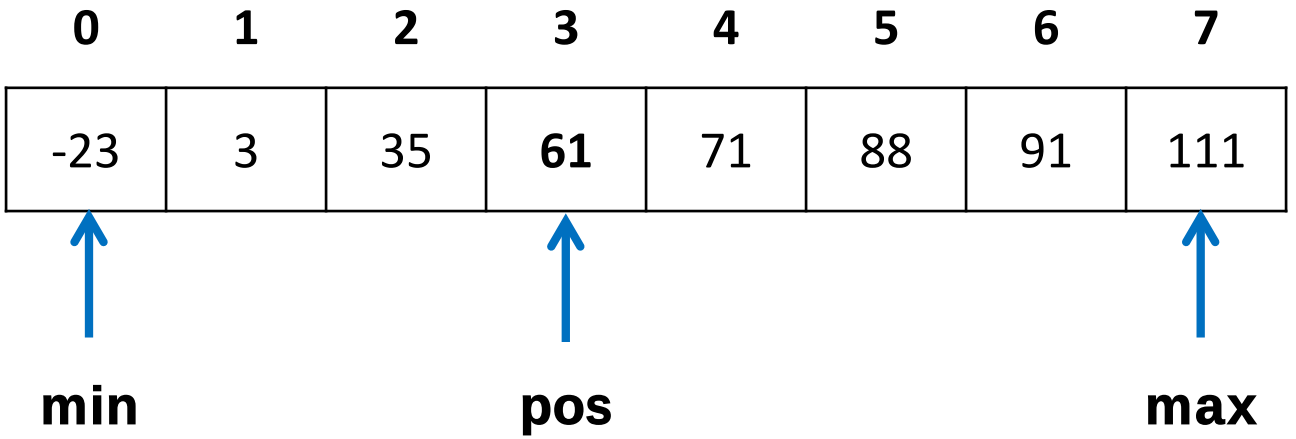
Μεσαίο στοιχείο πίνακα

- Περιττό και άρτιο πλήθος στοιχείων

Δυαδική αναζήτηση

Πρώτο παράδειγμα: Αναζήτηση 91

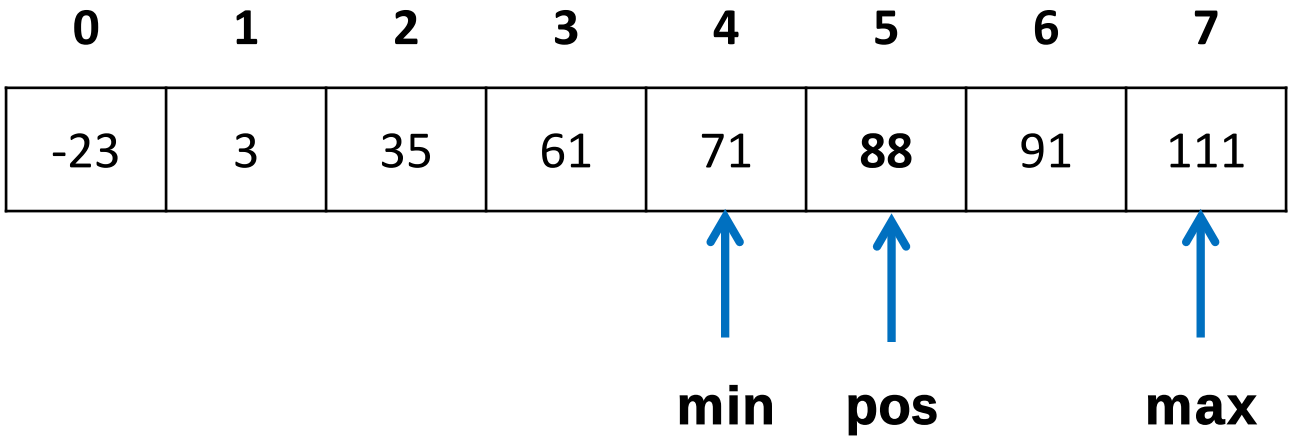
- Πρώτο βήμα



Δυαδική αναζήτηση

Πρώτο παράδειγμα: Αναζήτηση 91

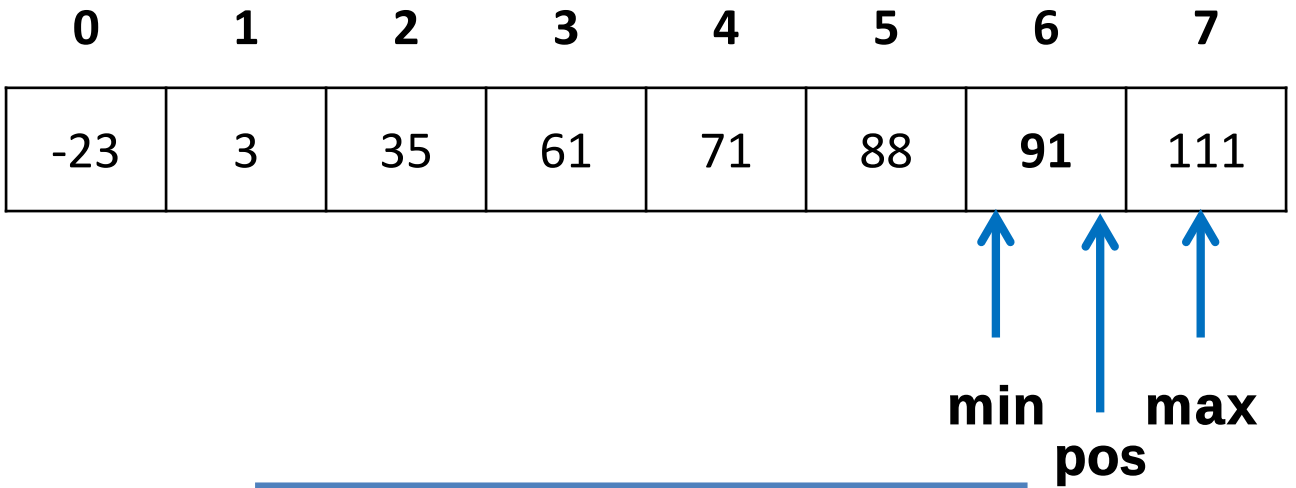
- Δεύτερο βήμα



Δυαδική αναζήτηση

Πρώτο παράδειγμα: Αναζήτηση 91

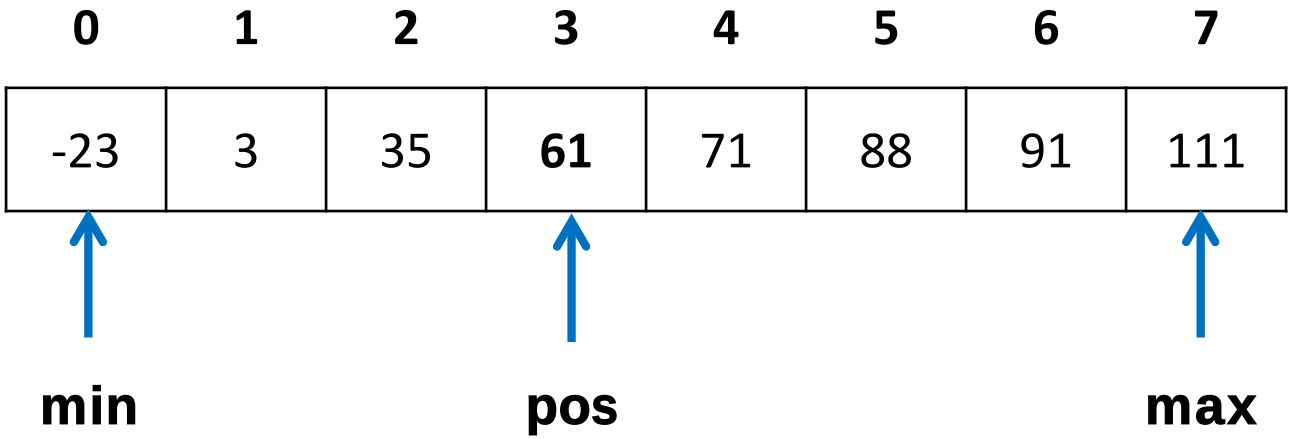
- Τρίτο βήμα



Δυαδική αναζήτηση

Δεύτερο παράδειγμα: Αναζήτηση 4

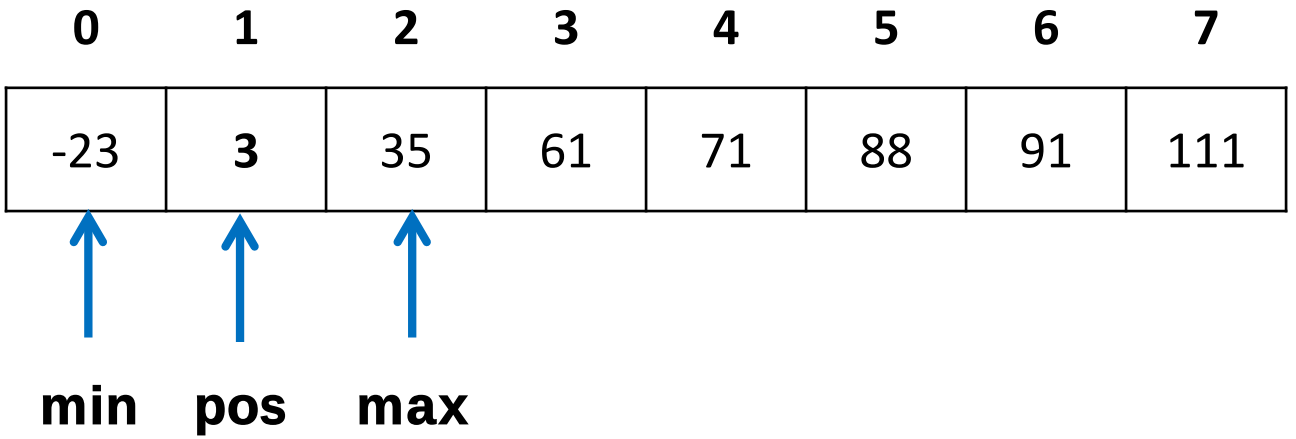
- Πρώτο βήμα



Δυαδική αναζήτηση

Δεύτερο παράδειγμα: Αναζήτηση 4

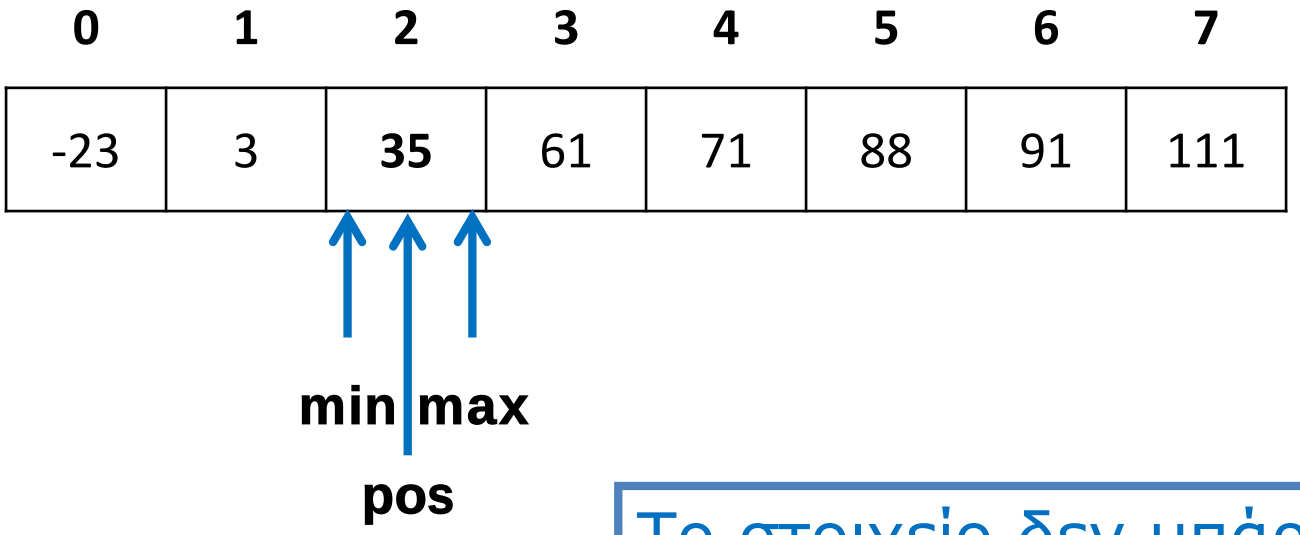
- Δεύτερο βήμα



Δυαδική αναζήτηση

Δεύτερο παράδειγμα: Αναζήτηση 4

- Τρίτο βήμα



Το στοιχείο δεν υπάρχει!!!

Δυαδική αναζήτηση

Κώδικας

```
int binary_search(const int table[], int size, int item)
{
   int min = 0, max = size – 1, pos= (min + max)/2;
   while(min <= max){
        if (item > table[pos])
                min = pos + 1;
        else if (item < table[pos])
                max = pos – 1;
        else                    /* βρέθηκε */
                return pos;
        pos = (min + max) / 2;
    }
   return -1;
}
```

Μέρος 3ο

Προβλήματα Ταξινόμησης

Προβλήματα Ταξινόμησης

Ταξινόμηση Φυσαλίδας και Επιλογής

Ταξινόμηση

Γενικά

- Πολύ σπουδαία διεργασία
- Αύξουσα ή φθίνουσα σειρά

Ταξινόμηση φυσαλίδας (bubble sort)

- Σύγκριση διαδοχικών ζευγαριών στοιχείων ενός πίνακα
  - Αν είναι στην επιθυμητή σειρά, καμία αλλαγή
  - Αν όχι, αντίστρεψε τη σειρά τους (ενάλλαξε τα περιεχόμενά τους)
- Επανάληψη

Αλγόριθμος bubble sort

- Απαιτεί διπλό βρόχο
- Σειρά ταξινόμησης ως παράμετρος

Ταξινόμηση φυσαλίδας

0	1	2	3	4
25	**3**	35	111	-40

0	1	2	3	4
3	25	35	111	-40

0	1	2	3	4
3	25	35	111	-40

0	1	2	3	4
3	25	35	**111**	**-40**

0	1	2	3	4
3	25	35	-40	111

Ταξινόμηση φυσαλίδας

0	1	2	3	4
3	25	35	-40	111

0	1	2	3	4
3	25	35	-40	111

0	1	2	3	4
3	25	**35**	**-40**	111

0	1	2	3	4
3	25	-40	35	111

0	1	2	3	4
3	25	-40	35	111

Ταξινόμηση φυσαλίδας

0	1	2	3	4
3	25	-40	35	111

0	1	2	3	4
3	**25**	**-40**	35	111

0	1	2	3	4
3	-40	25	35	111

0	1	2	3	4
3	-40	25	35	111

0	1	2	3	4
3	-40	25	35	111

Ταξινόμηση φυσαλίδας

0	1	2	3	4
3	**-40**	25	35	111

0	1	2	3	4
-40	3	25	35	111

0	1	2	3	4
-40	3	25	35	111

0	1	2	3	4
-40	3	25	35	111

0	1	2	3	4
-40	3	25	35	111

Ταξινόμηση φυσαλίδας

0	1	2	3	4
-40	3	25	35	111

0	1	2	3	4
-40	3	25	35	111

0	1	2	3	4
-40	3	25	35	111

0	1	2	3	4
-40	3	25	35	111

0	1	2	3	4
-40	3	25	35	111

Ταξινομημένο!!

Ταξινόμηση φυσαλίδας

Παρατηρήσεις στον αλγόριθμο

- Κάθε προσπέλαση καθορίζει τη θέση ενός τουλάχιστο στοιχείου
  - Η πρώτη προσπέλαση καθορίζει το μεγαλύτερο στοιχείο
  - Η δεύτερη το αμέσως μικρότερο
  - κ.ο.κ.

Ταξινόμηση φυσαλίδας

- Ο αριθμός των στοιχείων που εξετάζονται μετά από κάθε προσπέλαση μειώνεται κατά ένα
  - Γιατί;

- Τερματισμός αλγορίθμου (ταξινόμηση στοιχείων)
  - Εάν σε μια προσπέλαση δεν γίνει καμία εναλλαγή στοιχείων

Ταξινόμηση φυσαλίδας με μείωση του αριθμού στοιχείων που εξετάζονται

0	1	2	3	4
25	**3**	35	111	-40

0	1	2	3	4
3	25	35	111	-40

0	1	2	3	4
3	25	35	111	-40

0	1	2	3	4
3	25	35	**111**	**-40**

0	1	2	3	4
3	25	35	-40	111

Ταξινόμηση φυσαλίδας με μείωση του αριθμού στοιχείων που εξετάζονται

0	1	2	3	4
3	25	35	-40	111

0	1	2	3	4
3	25	35	-40	111

0	1	2	3	4
3	25	**35**	**-40**	111

0	1	2	3	4
3	25	-40	35	111

Ταξινόμηση φυσαλίδας με μείωση του αριθμού στοιχείων που εξετάζονται

0	1	2	3	4
3	25	-40	35	111

0	1	2	3	4
3	**25**	**-40**	35	111

0	1	2	3	4
3	-40	25	35	111

Ταξινόμηση φυσαλίδας με μείωση του αριθμού στοιχείων που εξετάζονται

0	1	2	3	4
3	**-40**	25	35	111

0	1	2	3	4
-40	3	25	35	111

Ταξινομημένο**!!**

Ταξινόμηση φυσαλίδας

Κώδικας 1

```
void
bubble_sort(int table[], int size)
{
        int pass, j, hold;
        for ( pass = 1; pass <= size - 1; pass++ )
                for ( j = 0; j <= size - 2; j++ )
                        if ( table[j] > table[j+1] ) {
                                hold = table[j];
                                table[j] = table[j+1];
                                        table[j+1] = hold;
                        }
}
```

Ταξινόμηση φυσαλίδας

Κώδικας 2

```
void
bubble_sort(int table[], int size)
{
    int sorted, len = size – 1, k;
    do{
        sorted = 1;     /* έστω ότι ο πίνακας είναι ταξινομημένος */
        for (k = 0; k < len; k++)
                        if (table[k]<table[k+1]){
                                swap(&table[k],&table[k+1]);
                                sorted = 0;
                        }
                        --len;
    } while (!sorted  &&  len > 0);
}
```

Ταξινόμηση φυσαλίδας

Κώδικας 3

```
void
bubble_sort(int compare(int,int), int table[], int size)
{
    int sorted, len = size – 1, k;
    do{
        sorted = 1;    /* έστω ότι ο πίνακας είναι ταξινομημένος */
        for (k = 0; k < len; k++)
                    if (compare(table[k],table[k+1])){
                        swap(&table[k],&table[k+1]);
                        sorted = 0;
                }
                --len;
    } while (!sorted  &&  len > 0);
}
```

Για καθορισμό αύξουσας η φθίνουσας σειράς

Ταξινόμηση φυσαλίδας

```
int less(int x, int y) { return x < y; }
int greater(int x, int y) { return x > y; }

#define SIZE 5

int
main () {
    int table[SIZE] = {9,3,5,1,7};

    bubble_sort(less, table, 5);
    display_table(table,SIZE);
    bubble_sort(greater, table,5);
    display_table(table,SIZE);
}
```

Ταξινόμηση επιλογής

Βασική ιδέα:

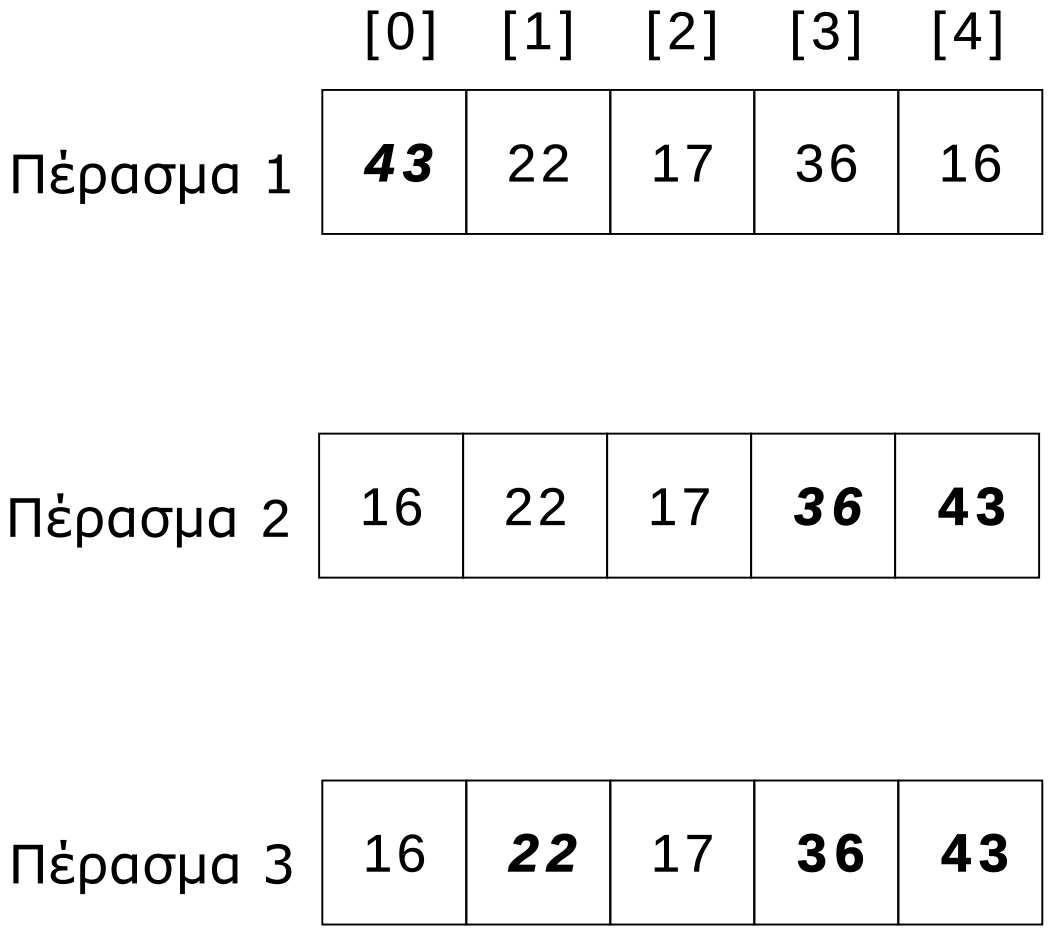
Σε έναν πίνακα στοιχείων, βρες το μεγαλύτερο στοιχείο του και κάνε εναλλαγή θέσεων (swap) αυτού με το τελευταίο στοιχείο του πίνακα

- Το αποτέλεσμα είναι μια μη ταξινομημένη λίστα της οποίας το μέγεθος είναι κατά 1 μικρότερο από το μέγεθος της αρχικής λίστας, η οποία ακολουθείται από μια «ταξινομημένη» λίστα μεγέθους 1
- Επανάληψη του παραπάνω βήματος για τη μη ταξινομημένη λίστα έχει ως αποτέλεσμα μια μη ταξινομημένη λίστα της οποίας το μέγεθος είναι κατά 2 μικρότερο από το μέγεθος της αρχικής, η οποία ακολουθείται από μια ταξινομημένη λίστα μεγέθους 2. κ.ο.κ.

Τερματισμός αλγορίθμου (ταξινόμηση στοιχείων):

- Όταν το μέγεθος της μη ταξινομημένης λίστας γίνει 1

Ταξινόμηση επιλογής



Βρες το μεγαλύτερο των πρώτων **πέντε** στοιχείων και κάνε το εναλλαγή με το στοιχείο **array[4]**

Βρες το μεγαλύτερο των πρώτων **τεσσάρων** στοιχείων και κάνε το εναλλαγή με το στοιχείο **array[3]**

Βρες το μεγαλύτερο των πρώτων **τριών** στοιχείων και κάνε το εναλλαγή με το στοιχείο **array[2]**

Ταξινόμηση επιλογής

	[0]	[1]	[2]	[3]	[4]
Πέρασμα 4	16	***17***	**22**	**36**	**43**

Βρες το μεγαλύτερο των πρώτων **δύο** στοιχείων και κάνε το εναλλαγή με το στοιχείο **array[1]**

16	**17**	**22**	**36**	**43**

Ταξινομημένο**!!**

Ταξινόμηση επιλογής

Κώδικας

```
void selection_sort(int size, int table[])
{
    int i;
    int index_of_largest, last, temp;
    for (last = size-1; last >= 1; last--) {
          index_of_largest = 0;
          for (i = 1; i <= last; i++)
                    if (table[i] > table[index_of_largest])
                              index_of_largest = i;
                    /* end if */
          /* end for */
          temp = table[last];
          table[last] = table[index_of_largest];
          table[index_of_largest] = temp;
     }
}
```

Τέλος Ενότητας

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στο πλαίσιο του εκπαιδευτικού έργου των διδασκόντων.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Πατρών**» έχει χρηματοδοτήσει μόνο την αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Σημείωμα Αναφοράς

Νίκος Καρακαπιλίδης, Δημήτρης Σαραβάνος. «Προγραμματισμός Η/Υ. Ειδικά θέματα Αλγορίθμων». Έκδοση: 1.0. Πάτρα 2014. Διαθέσιμο από τη δικτυακή διεύθυνση: https://eclass.upatras.gr/courses/MECH1207/

Σημείωμα Αδειοδότησης

Διατήρηση Σημειωμάτων

Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει:

- το Σημείωμα Αναφοράς
- το Σημείωμα Αδειοδότησης
- τη δήλωση Διατήρησης Σημειωμάτων
- το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει)

μαζί με τους συνοδευόμενους υπερσυνδέσμους.

Σημείωμα Χρήσης Έργων Τρίτων