13. Άσκηση 13 – Ταξινόμηση και Αναζήτηση (Sorting and Searching)

Η άσκηση έχει στόχο την εισαγωγή βασικών εννοιών ταξινόμησης και αναζήτησης και την εξοικείωση με τις διαδικασίες ταξινόμησης και αναζήτησης στοιχείων σε πίνακα

Η **ενότητα 1** περιγράφει το αντικείμενο το οποίο θα πρέπει να ικανοποιεί η εφαρμογή.

Η **ενότητα 2** αναφέρεται στην αξιοποίηση έτοιμων συναρτήσεων για την ταξινόμηση πίνακα ακεραίων και αλφαριθμητικών. Δίνει έμφαση και στην ανάπτυξη κώδικα που θα έχει δεδομένη έξοδο στην οθόνη.

Η **ενότητα 3** περιγράφει εν συντομία τον αλγόριθμο ταξινόμησης bubble sort

Η **ενότητα 4** ζητά την ανάπτυξη ενός προγράμματος που θα χρησιμοποιεί την συνάρτηση bubbleSort() και θα μας προετοιμάσει για τον ορισμό της.

Η **ενότητα 5 αναφέρεται στη διαδικασία που θα πρέπει να ακολουθήσετε για την σταδιακή ανάπτυξη της bubbleSort()**. Περιγράφονται τρεις εκδόσεις οι οποίες υλοποιούν σταδιακά την λειτουργικότητα της ταξινόμησης με στόχο να οδηγηθούμε σε εναλλακτικές υλοποιήσεις. Η διαδικασία δίνει έμφαση και στην ανάπτυξη κώδικα που θα έχει δεδομένη έξοδο στην οθόνη.

Η **ενότητα 6** αναφέρεται σε εναλλακτικές υλοποιήσεις με βάση την βήμα προς βήμα ανάπτυξη της bubbleSort().

Η **ενότητα 7** αναφέρεται σε υλοποιήσεις της bubbleSort() για άλλους τύπους πινάκων.

Η **ενότητα 8** αναφέρεται στην αξιοποίηση της συνάρτησης binarySearch για αναζήτηση σε πίνακες ακεραίων και αλφαριθμητικών.

Αξιοποιήστε τον πίνακα περιεχομένων του pdf για να έχετε μια αφαιρετική εικόνα της δόμησης του κειμένου η οποία σας επιτρέπει να μετακινήστε εύκολα από ενότητα σε ενότητα.

13.1 Το αντικείμενο

Το αντικείμενο της ΕΑ είναι:

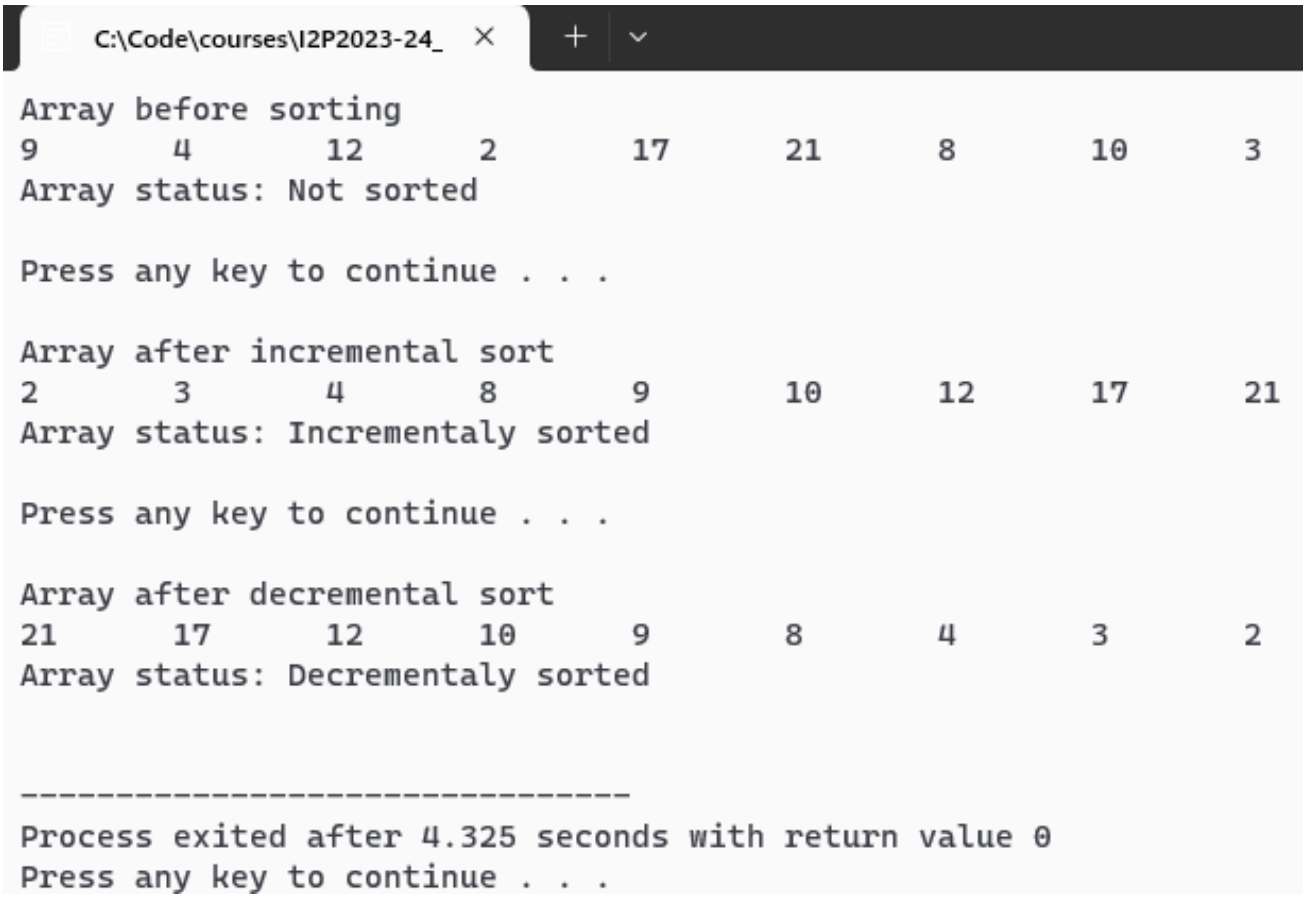
1) η αξιοποίηση έτοιμων συναρτήσεων ταξινόμησης και αναζήτησης από βιβλιοθήκη τρίτου κατασκευαστή (i2pLibrary),
2) η ανάπτυξη βήμα προς βήμα συνάρτησης που θα υλοποιεί την ταξινόμηση φυσαλίδας (bubble sort), και,
3) η αξιοποίηση έτοιμων συναρτήσεων ταξινόμησης και αναζήτησης της βασικής βιβλιοθήκης της C.

Η όλη διαδικασία είναι προσαρμοσμένη για την εξοικείωση και με την διαδικασία ανάπτυξης προγράμματος που θα έχει ζητούμενη έξοδο στην οθόνη.

13.2 Χρήση έτοιμων συναρτήσεων

13.2.1 Ταξινόμηση πίνακα ακεραίων

Αναθέστε στη μηχανή το έργο της δημιουργίας ενός πίνακα 12 ακεραίων, τον οποίο θα γεμίζει με τυχαίους αριθμούς (C rand() function, Generating random values in C) και στη συνέχεια θα τον ταξινομεί με αύξουσα και φθίνουσα σειρά κάνοντας χρήση συναρτήσεων της βιβλιοθήκης i2p . Ταυτόχρονα να δίνει στην οθόνη έξοδο ανάλογη αυτής του σχήματος 1. Η διαπίστωση του αν είναι ταξινομημένος ο πίνακας και πως, να γίνεται με την εκτέλεση κατάλληλης συνάρτησης.



Σχήμα 1. Έξοδος προγράμματος ταξινόμησης πίνακα ακεραίων.

13.2.2 Ταξινόμηση πίνακα αλφαριθμητικών

Αναθέστε στη μηχανή το έργο της δημιουργίας ενός πίνακα 8 αλφαριθμητικών, τον οποίο θα γεμίζει με τα αλφαριθμητικά τυχαίων τετραψήφιων ακέραιων αριθμών και στη συνέχεια θα τον ταξινομεί με αύξουσα και φθίνουσα σειρά κάνοντας χρήση συναρτήσεων της βιβλιοθήκης i2p . Ταυτόχρονα να δίνει στην οθόνη έξοδο ανάλογη αυτής του σχήματος 1, αλλά για αλφαριθμητικά. Η διαπίστωση του αν είναι ταξινομημένος ο πίνακας και πως, να γίνεται με την εκτέλεση κατάλληλης συνάρτησης.

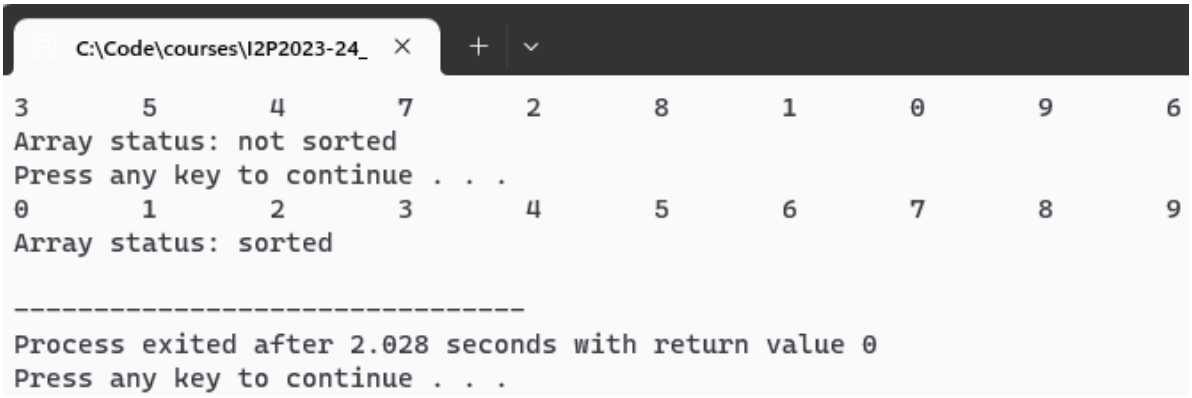
13.3 Ο αλγόριθμος ταξινόμησης bubble sort

«**Bubble sort**, sometimes referred to as **sinking sort**, is a simple sorting algorithm that repeatedly steps through the input list element by element, comparing the current element with the one after

it, swapping their values if needed. These passes through the list are repeated until no swaps have to be performed during a pass, meaning that the list has become fully sorted. The algorithm, which is a comparison sort, is named for the way the larger elements "bubble" up to the top of the list.» wikidedia.

13.4 Ένα πρόγραμμα που χρησιμοποιεί την συνάρτηση bubbleSort

Δώστε την δήλωση της συνάρτησης bubbleSort() και με βάση αυτή δώστε την main ενός προγράμματος που την αξιοποιεί για να εμφανίσει στην οθόνη κάτι ανάλογο του σχήματος 2. Αξιοποιείστε αφαιρετικότητα στις διεργασίες. Την έξοδο του προγράμματος σας θα έχετε τη δυνατότητα να ελέγξετε και επιδείξετε μετά τον ορισμό της bubbleSort().



Σχήμα 2. Ενδεικτική έξοδος προγράμματος ταξινόμησης πίνακα 10 ακεραίων.

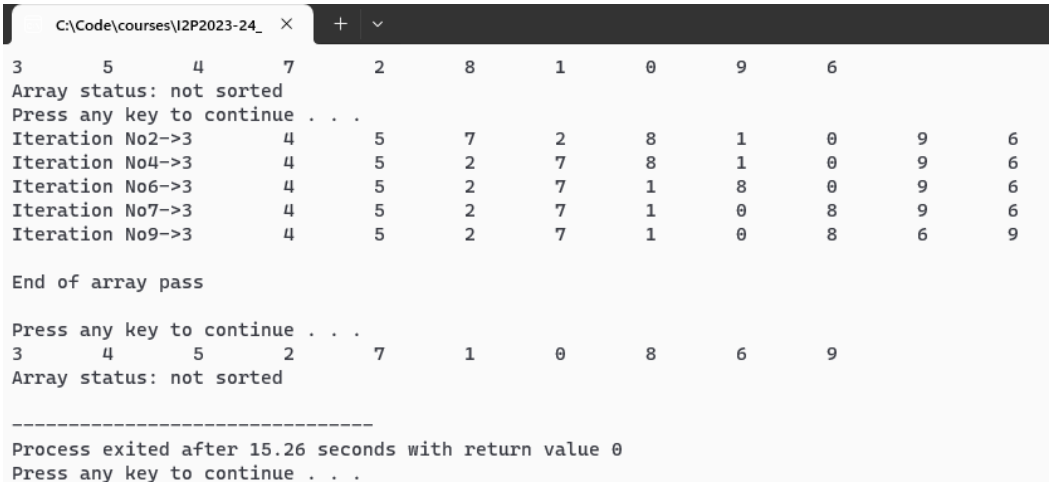
13.5 Ανάπτυξη συνάρτησης ταξινόμησης που υλοποιεί τον αλγόριθμο bubble sort

Η ανάπτυξη της συνάρτησης bubbleSort() θα γίνει μέσα από μία διαδικασία (σειρά βημάτων ανάπτυξης) που μας βοηθούν στην σταδιακή ανάπτυξη της συνάρτησης και την σε βάθος κατανόηση της. Ο κώδικας κάθε βήματος υλοποιεί μέρος της λειτουργικότητας του αλγορίθμου με στόχο τον κώδικα που υλοποιεί όλη την λειτουργικότητα. Παράλληλα μας δίνει τη δυνατότητα να δούμε ενναλακτικές υλοποιήσεις.

Η διαδικασία ανάπτυξης που υιοθετούμε αποτελείται από τρία βήματα με το καθένα να έχει ως αποτέλεσμα την ανάπτυξη μιας έκδοσης της bubbleSort(). **Τα τρία βήματα δουλέψαμε στην αίθουσα και δώσαμε τον σχετικό κώδικα** (διαφάνειες I2P_SortAndSearch.pdf).

13.5.1 1η έκδοση της bubbleSort()

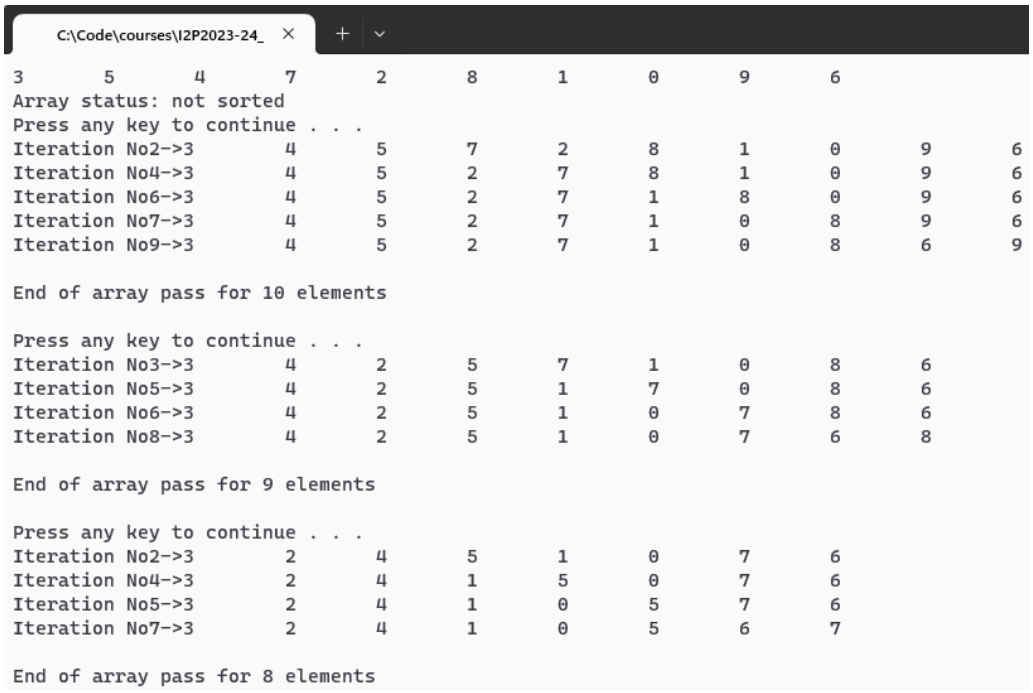
Δώστε την 1η έκδοση της bubbleSort() η οποία **υλοποιεί μόνο** την μετακίνηση του μεγαλύτερου στοιχείου του πίνακα στην τελευταία θέση του. Αναπτύξτε ένα πρόγραμμα που θα αξιοποιεί την 1η αυτή έκδοση της bubbleSort() την οποία θα τροποποιήσετε κατάλληλα ώστε το πρόγραμμα σας να εμφανίζει στην οθόνη κάτι ανάλογο του σχήματος 3.



Σχήμα 3. Ενδεικτική έξοδος προγράμματος που αξιοποιεί την κατάλληλα διαμορφωμένη συνάρτηση bubbleSort() που υλοποιεί την λειτουργικότητα του βήματος 1.

13.5.2 2η έκδοση της bubbleSort()

Δώστε την 2η έκδοση της bubbleSort() η οποία θα επαναλαμβάνει την λειτουργικότητα του βήματος 1 για όλους τους υπο-πίνακες του αρχικού πίνακα μέχρι αυτόν με δύο στοιχεία. Αναπτύξτε ένα πρόγραμμα που θα αξιοποιεί την 2η αυτή έκδοση της bubbleSort() για να εμφανίζει στην οθόνη κάτι ανάλογο του σχήματος 4. Παρατηρήστε τις άσκοπες επαναλήψεις (Σχήμα 5). Αυτές μας οδηγούν στην ανάπτυξη της 3ης έκδοσης.



Σχήμα 4. Τμήμα ενδεικτικής εξόδου προγράμματος που αξιοποιεί την κατάλληλα διαμορφωμένη συνάρτηση bubbleSort() που υλοποιεί την λειτουργικότητα του βήματος 2.

13.5.3 3η έκδοση της bubbleSort()

Δώστε την 3η έκδοση της bubbleSort() η οποία θα βελτιώνει την λειτουργικότητα του βήματος 2 για να αποφύγουμε τις άσκοπες επαναλήψεις. Αναπτύξτε ένα πρόγραμμα που θα αξιοποιεί την 3η αυτή έκδοση της bubbleSort() για να εμφανίζει στην οθόνη κάτι ανάλογο του σχήματος 5 όπου βέβαια δεν θα έχουμε άσκοπες επαναλήψεις.

```
Iteration No1->1        2       0       3       4
Iteration No2->1        0       2       3       4

End of array pass for 5 elements

Press any key to continue . . .
Iteration No1->0        1       2       3

End of array pass for 4 elements

Press any key to continue . . .

End of array pass for 3 elements

Press any key to continue . . .

End of array pass for 2 elements

Press any key to continue . . .
0       1       2       3       4       5       6       7       8       9
Array status: sorted

--------------------------------
Process exited after 17.62 seconds with return value 0
Press any key to continue . . .
```

Σχήμα 5. Τμήμα ενδεικτικής εξόδου προγράμματος που αξιοποιεί την κατάλληλα διαμορφωμένη συνάρτηση bubbleSort() που υλοποιεί την λειτουργικότητα του βήματος 2 και εμφανίζει άσκοπες επαναλλήψεις.

13.6 Εναλλακτικές υλοποιήσεις της bubbleSort() για πίνακα ακεραίων

Μελετήστε εναλλακτικές υλοποιήσεις της συνάρτησης bubbleSort().

Δώστε τις παρακάτω υλοποιήσεις για την bubbleSort() για πίνακα ακεραίων.

1. Αξιοποιεί ένα βρόχο επανάληψης και την συνάρτηση moveBubbleUp()
2. Αξιοποιεί δύο βρόχους επανάληψης (χωρίς τη συνάρτηση moveBubbleUp())
3. Αξιοποιεί αναδρομικότητα (recursion).

Αναδρομικότητα έχουμε όταν μια συνάρτηση καλεί τον εαυτό της.

13.7 Υλοποιήσεις της bubbleSort() για άλλους τύπους πινάκων

Διαμορφώστε κατάλληλα την bubbleSort() για ταξινόμηση διαφορετικών τύπων πινάκων.

13.7.1 Ταξινόμηση πίνακα δεικτών σε ακεραίους **[προαιρετικό]**

Αναπτύξτε ένα πρόγραμμα που θα ταξινομεί ένα πίνακα δεικτών σε ακεραίους.

13.7.2 Ταξινόμηση πίνακα αλφαριθμητικών

Αναπτύξτε ένα πρόγραμμα που θα ταξινομεί ένα πίνακα αλφαριθμητικών.

13.7.3 Ταξινόμηση πίνακα δεικτών σε αλφαριθμητικά **[προαιρετικό]**

Αναπτύξτε ένα πρόγραμμα που θα ταξινομεί ένα πίνακα δεικτών σε αλφαριθμητικά.

13.7.4 Ταξινόμηση πίνακα δομών

Αναπτύξτε ένα πρόγραμμα που θα ταξινομεί ένα πίνακα δομών Student, όπου Student ορίζεται όπως παρακάτω:

```
typedef struct student {
   char fname[20];
   char lname[30];
   int am;
} Student;
```

Δώστε α) ταξινόμηση με βάση το lname, και β) ταξινόμηση με βάση το am.

[Προαιρετικό]

Μπορείτε να ορίσετε 2 πίνακες δεικτών σε δομές και να ταξινομήσετε τον έναν με βάση το am και τον άλλο με βάση το lname. Με τον τρόπο αυτό θα έχετε τον πίνακα δομών σας στην αρχική κατάσταση και επιπλέον θα έχετε τη δυνατότητα να βλέπετε τους φοιτητές ταξινομημένους και ως προς τον αριθμό μητρώου και προς το επίθετο.

13.8 Αξιοποίηση της συνάρτησης binarySearch

Τροποποιήστε τα προγράμματα **BubbleSortArrayOfInts** και **BubbleSortArrayOfStrings** έτσι ώστε να δίνουν τη δυνατότητα στον χρήστη αναζήτησης ακεραίου ή αλφαριθμητικού αντίστοιχα στους πίνακες με την χρήση της binarySearch().