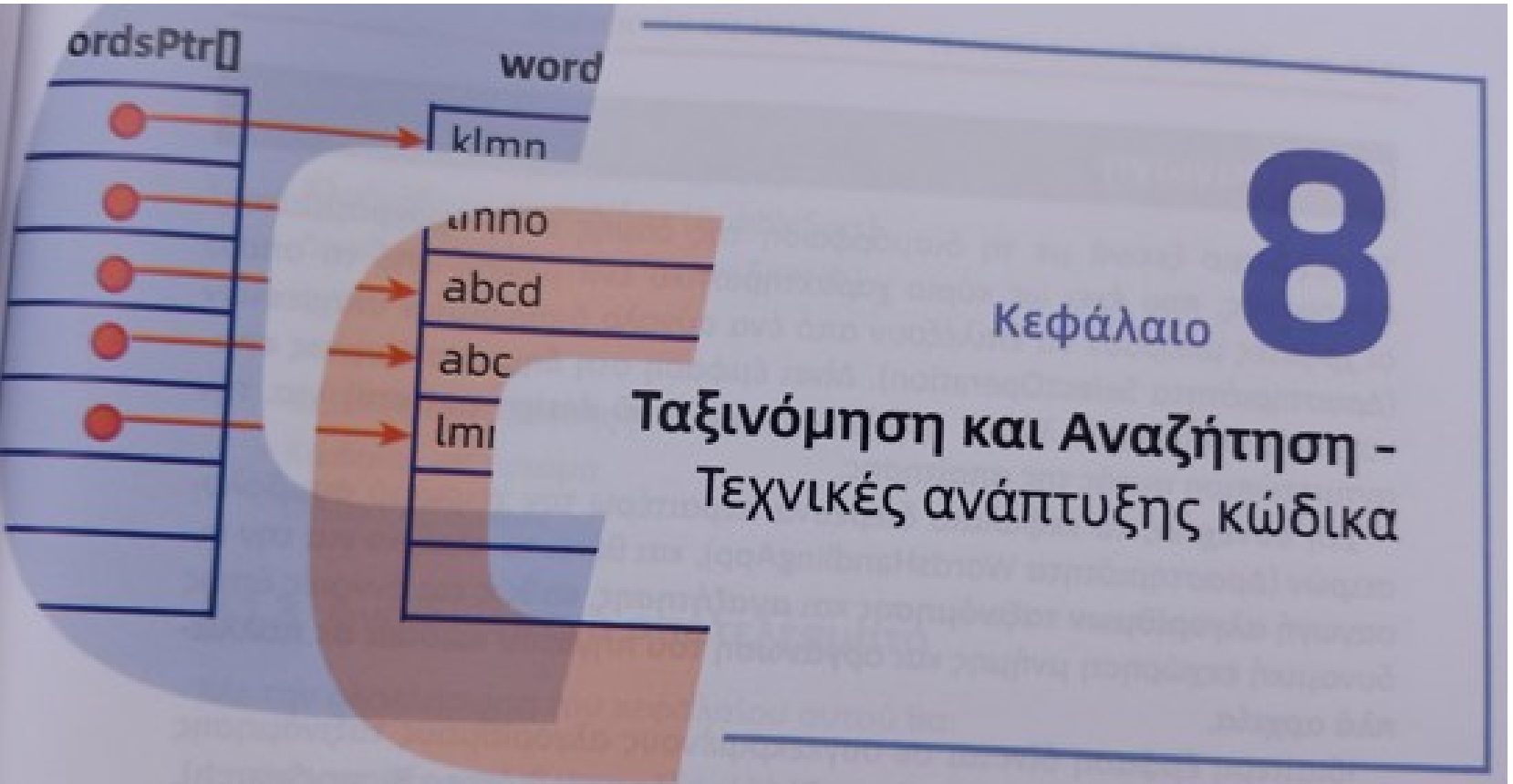
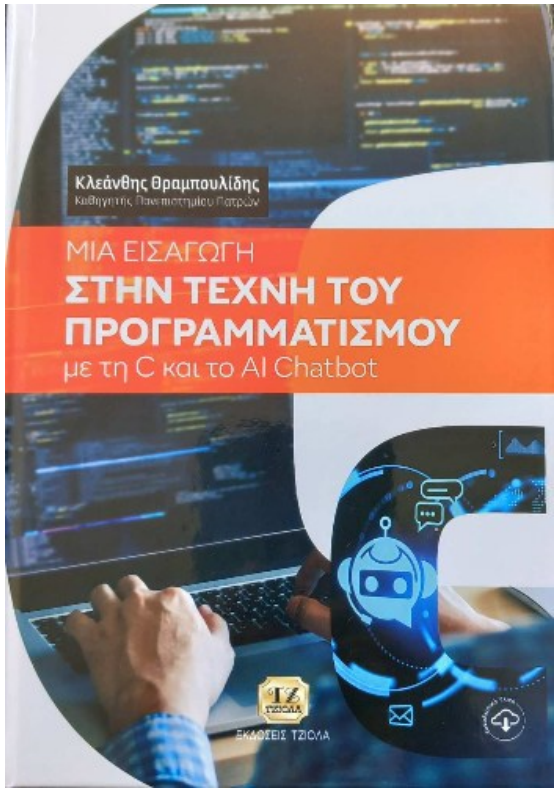
Εισαγωγή στον Προγραμματισμό (Introduction to Programming)

Ταξινόμηση και Αναζήτηση – Part B

A Step-By-Step Approach

https://sites.google.com/view/i2art-of-programming





Πηγή: Μια Εισαγωγή στην Τέχνη του Προγραμματισμού

Sorting and Searching

Στόχος Ενότητας

- Δίνει τη διαμόρφωση της δομής ενός προγράμματος/εφαρμογής, που έχει ως κύριο χαρακτηριστικό ένα **Μενού**, από το οποίο οι χρήστες μπορούν να επιλέξουν από ένα σύνολο διαθέσιμων υπηρεσιών. Παρουσιάζει μια επαναχρησιμοποιήσιμη λύση (μοτίβο σχεδιασμού-design pattern) για την αντιμετώπιση αυτής της απαίτησης.
- Τονίζει τη σημασία της **σταδιακής ανάπτυξης του αλγορίθμου** μέσα από μια επαναληπτική διαδικασία για την **αντιμετώπιση της πολυπλοκότητας**,
- Εισάγει τις τεχνικές της τμηματοποίησης (**partitioning**) και της από το γενικό προς το ειδικό ανάπτυξης (**top-down**).
- διερευνά περαιτέρω το χειρισμό συμβολοσειρών, και θέτει το πλαίσιο για την εισαγωγή **αλγορίθμων ταξινόμησης και αναζήτησης**, καθώς και έννοιες όπως **δυναμική εκχώρηση μνήμης** και **οργάνωση του πηγαίου κώδικα σε πολλαπλά αρχεία**.

Η ικανότητα να σκέφτεται κανείς αφαιρετικά και να διατυπώνει τη Λεκτική Περιγραφή, ενισχύει την αποτελεσματικότητα και τις δεξιότητες του προγραμματιστή, ιδιαίτερα κατά τον σχεδιασμό αλγορίθμων.

Η ενότητα

- **Δίνει έμφαση** σε συγκεκριμένους αλγόριθμους ταξινόμησης (Δραστηριότητα BubbleSort) και αναζήτησης (Δραστηριότητα BinarySearch), και στη διαδικασία υλοποίησης τους με τη μορφή επαναχρησιμοποιήσιμων συναρτήσεων.
- **Εστιάζει :**
  - στη σταδιακή ανάπτυξη της λειτουργικότητας μέσα από μια σειρά εκδόσεων, και,
  - στη βελτίωση της απόδοσης του αλγορίθμου.
- **Μελετά** την αξιοποίηση συναρτήσεων ταξινόμησης (qsort) και αναζήτησης (bsearch) της τυπικής βιβλιοθήκης
- **Εστιάζει** στο σημαντικό πλεονέκτημα που παρέχει η αξιοποίηση δείκτη συνάρτησης για να περάσουμε στις συναρτήσεις αυτές συγκεκριμένη λογική σύγκρισης που απαιτεί η εφαρμογή του αλγορίθμου.
- Προχωρά στην ανάπτυξη μιας γενικής συνάρτησης ταξινόμησης (bsort), η οποία υλοποιεί τον αλγόριθμο Bubble Sort και μπορεί να εφαρμοστεί σε πίνακες οποιουδήποτε τύπου

Οργάνωση Διάλεξης

- **Ταξινόμηση με χρήση της i2p βιβλιοθήκης**
  - Πίνακα ακεραίων
  - Πίνακα αλφαριθμητικών
- **Ταξινόμηση - Bubble Sort**
  - A step-by-step Approach για πίνακα ακεραίων
  - Ενναλακτικές Υλοποιήσεις (+Αναδρομικότητα)
  - Ταξινόμηση πίνακα Δεικτών σε ακεραίους
  - Ταξινόμηση Λέξεων
    - Ταξινόμηση πίνακα αλφαριθμητικών
    - Ταξινόμηση πίνακα Δεικτών σε αλφαριθμητικά
- **Αναζήτηση**
  - Linear - Binary
  - Σε πίνακα ακεραίων
  - Σε πίνακα αλφαριθμητικών
- Ταξινόμηση και Αναζήτηση με την τυπική βιβλιοθήκη

Ταξινόμηση με χρήση της i2p βιβλιοθήκης

i2p library

1. Ταξινόμηση (αύξουσα και φθίνουσα) **Πίνακα ακεραίων**

```
void sortIntArrayInc(int ar[],int numOfElements);
void sortIntArrayDec(int ar[],int numOfElements);
```

2. Ταξινόμηση (αύξουσα και φθίνουσα) **Πίνακα αλφαριθμητικών**

```
void sortStringArrayInc(char base[],int numOfElements, int strWidth);
void sortStringArrayDec(char base[],int numOfElements, int strWidth);
```

Hands-on : Ταξινόμηση Πίνακα ακεραίων

```
void sortIntArrayInc(int ar[],int numOfElements);
void sortIntArrayDec(int ar[],int numOfElements);
```

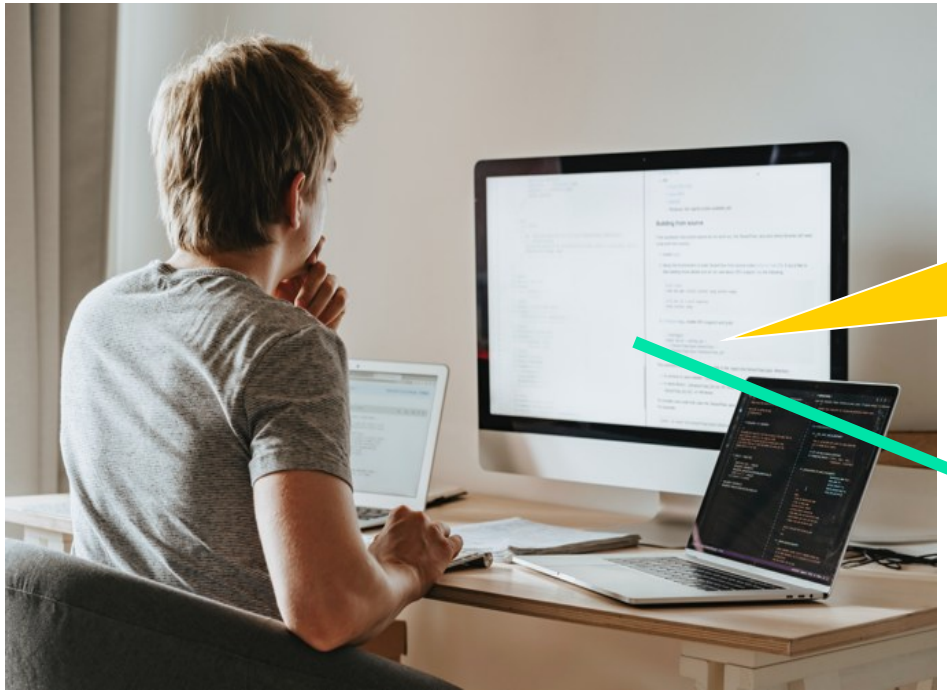
```
C:\Code\courses\I2P2023-24_
Array before sorting
9       4       12      2       17      21      8       10      3
Array status: Not sorted

Press any key to continue . . .

Array after incremental sort
2       3       4       8       9       10      12      17      21
Array status: Incrementaly sorted

Press any key to continue . . .

Array after decremental sort
21      17      12      10      9       8       4       3       2
Array status: Decrementaly sorted


--------------------------------
Process exited after 4.325 seconds with return value 0
Press any key to continue . . .
```

Αναπτύξτε ένα πρόγραμμα που να έχει την παραπλεύρως έξοδο κάνοντας χρήση των δύο παραπάνω συναρτήσεων ταξινόμησης.

Η διαπίστωση του αν είναι ταξινομημένος ο πίνακας και πως, να γίνεται από την μηχανή με την εκτέλεση κατάλληλης συνάρτησης

Hands-on : Ταξινόμηση Πίνακα αλφαριθμητικών

```
void sortStringArrayInc(char base[],int numOfElements, int strWidth);
void sortStringArrayDec(char base[],int numOfElements, int strWidth);

char ar[][10]={"paris","nikos","alekos","kostas",
               "basos","andreas","petros"};
```



Χρησιμοποιήστε τις δύο παραπάνω συναρτήσεις για να αναπτύξτε ένα πρόγραμμα που να έχει έξοδο ανάλογη με αυτή του προγράμματος που αναπτύξατε για ταξινόμηση ακεραίων.

Η διαπίστωση του αν είναι ταξινομημένος ο πίνακας και πως, να γίνεται από την μηχανή με την εκτέλεση κατάλληλης συνάρτησης

Οργάνωση Διάλεξης

- **Ταξινόμηση με χρήση της i2p βιβλιοθήκης**
  - Πίνακα ακεραίων
  - Πίνακα αλφαριθμητικών
- **Ταξινόμηση - Bubble Sort**
  - A step-by-step Approach για πίνακα ακεραίων
  - Ενναλακτικές Υλοποιήσεις (+Αναδρομικότητα)
  - Ταξινόμηση πίνακα Δεικτών σε ακεραίους
  - Ταξινόμηση Λέξεων
    - Ταξινόμηση πίνακα αλφαριθμητικών
    - Ταξινόμηση πίνακα Δεικτών σε αλφαριθμητικά
- **Αναζήτηση**
  - Linear - Binary
  - Σε πίνακα ακεραίων
  - Σε πίνακα αλφαριθμητικών
- Ταξινόμηση και Αναζήτηση με την τυπική βιβλιοθήκη

Δραστηριότητα 8.3 - BubbleSort

Αναπτύξτε ένα πρόγραμμα που θα χρησιμοποιεί μια συνάρτηση bubbleSort4ArrayOfInts, την οποία θα αναπτύξετε για ταξινόμηση πίνακα ακεραίων με βάση τον **αλγόριθμο φυσαλίδας**.

- Η Δραστηριότητα έχει **στόχο** την εξάσκηση με την βήμα-προς-βήμα διαδικασία ανάπτυξης συναρτήσεων ταξινόμησης που βασίζονται πάνω στον αλγόριθμο bubble sort (αλγόριθμο φυσαλίδας) για την ταξινόμηση αριθμών και συμβολοσειρών

Hands-on : H main

Αναπτύξτε ένα πρόγραμμα που θα χρησιμοποιεί μια συνάρτηση που υλοποιεί τον αλγόριθμο Bubble Sort για να ταξινομήσει ένα πίνακα ακεραίων.

Δώστε τη Λεκτική Περιγραφή αξιοποιώντας αφαιρετικότητα στις διεργασίες και στη συνέχεια γράψτε την main με στόχο το πρόγραμμα σας να έχει την παρακάτω έξοδο.

Δώστε μόνο τη δήλωση της bubbleSort

```
3       5       4       7       2       8       1       0       9       6
Array status: not sorted
Press any key to continue . . .
0       1       2       3       4       5       6       7       8       9
Array status: sorted

--------------------------------
```

Screenshot εκτέλεσης του ζητούμενου προγράμματος.

Ταξινόμηση πίνακα ακεραίων – Η main

void bubbleSort(int ar[], int numOfElements);

```
16  int ar[] = {3,5,4,7,2,8,1,0,9,6};
17  int main(int argc, char *argv[]) {
18
19      int numOfElements =sizeof(ar)/sizeof(int);
20      displayArray(ar,numOfElements);
21      arrayStatus(ar,numOfElements);
22      system("pause");
23      bubbleSort(ar, numOfElements);
24      displayArray(ar,numOfElements);
25      arrayStatus(ar,numOfElements);
26      return 0;
27  }
```

Αξιοποιήθηκαν οι τεχνικές της τμηματοποίησης (**partitioning**) και της από το γενικό προς το ειδικό ανάπτυξης (**top-down**).

Bubble Sort – Ορισμός

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

"Bubble 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 had 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." *Wikipedia*

Bubble Sort – Βασικές διεργασίες

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

"Bubble 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 had 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." *Wikipedia*

- **Δύο είναι οι βασικές διεργασίες που εκτελεί**
  - Σύγκριση δύο στοιχείων (**comparing** the current element with the one after it)
  - Εναλλαγή θέσης αν η σύγκριση το απαιτεί (**swapping** their values if needed)
- Με την **επαναλαμβανόμενη εκτέλεση** των δύο αυτών διεργασιών επιτυγχάνει την ταξινόμηση των στοιχείων του πίνακα
  - **repeatedly steps** through the input list element by element
  - These **passes through the list** are repeated

Bubble Sort – A step-by-step Approach

"Bubble 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 had 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." *Wikipedia*

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

- Στόχος μας να αναπτύξουμε την bubbleSort βήμα-προς-βήμα.
  - **Step 1** – **Υλοποιεί το κίτρινο μέρος της περιγραφής**

    Κάνει moveBubbleUp για τον πίνακα ar με στοιχεία numOfElements
  - **Step 2** – Υλοποιεί την επανάληψη του Step 1 για όλους τους υπο-πίνακες του ar μέχρι τον υπο-πίνακα με 2 στοιχεία (πράσινο μέρος περιγραφής).

    Στην ουσία επαναλαμβάνει την δουλειά του βήματος 1, δηλαδή το moveBubbleUp, για κάθε υπο-πίνακα του ar μέχρι αυτόν με 2 στοιχεία.
  - **Step 3** – Βελτίωση του βήματος 2 για να αποφύγουμε τις άσκοπες επαναλήψεις

Bubble Sort – Step 1 (BubbleSort Version 1(moveBubbleUp))

Δράση 8-11

Αναπτύξτε μια συνάρτηση bubbleSort που υλοποιεί το βήμα 1 (Step 1) δηλαδή τη διεργασία moveBubbleUp.

- **Step 1 – Υλοποιεί το κίτρινο μέρος της περιγραφής**

  Κάνει moveBubbleUp για τον πίνακα ar με στοιχεία numOfElements

1st Step – moveBubbleUp for ar (1/2)

"Bubble 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. (…)" *Wikipedia*

- Ονομάζουμε **moveBubbleUp** τη διεργασία που περιγράφεται παραπάνω με κίτρινο
- Αυτή είναι η πρώτη δουλειά που πρέπει να κάνει η bubbleSort
- Με βάση αυτό η bubbleSort διαμορφώνεται όπως παρακάτω

```
28  void bubbleSort(int ar[], int numOfElements){
29      moveBubbleUp(ar,numOfElements);
30      }
31
32  void moveBubbleUp(int ar[], int numOfElements){
```

Δώστε το σώμα της moveBubbleUp

1st Step – moveBubbleUp for ar (2/2)

Ο πίνακας

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

Screenshot εκτέλεσης 1ης έκδοσης

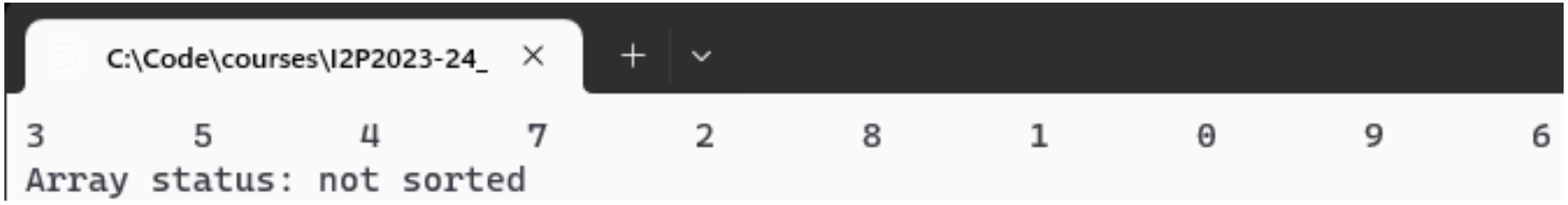
```
C:\Code\courses\I2P2023-24_
3       5       4       7       2       8       1       0       9       6
Array status: not sorted
Press any key to continue . . .
Iteration No2->3        4       5       7       2       8       1       0       9       6
Iteration No4->3        4       5       2       7       8       1       0       9       6
Iteration No6->3        4       5       2       7       1       8       0       9       6
Iteration No7->3        4       5       2       7       1       0       8       9       6
Iteration No9->3        4       5       2       7       1       0       8       6       9

End of array pass

Press any key to continue . . .
3       4       5       2       7       1       0       8       6       9
Array status: not sorted

--------------------------------
Process exited after 15.26 seconds with return value 0
Press any key to continue . . .
```

Το iteration είναι η επανάληψη μιας εκτέλεσης της σύγκρισης δυο στοιχείων και εναλλαγής θέσης τους αν απαιτείται.

Δώστε τον πηγαίο κώδικα για το βήμα αυτό (V1)

Next Step for bubbleSort ?

Ο πίνακας **πριν την εκτέλεση του Step 1**



Ο πίνακας **μετά την εκτέλεση του Step 1** (moveBubbleUp)

Bubble Sort – Step 2 (BubbleSort Version 2)

Δράση 8-12

Αναβαθμίστε την 1η έκδοση της συνάρτησης ώστε να υλοποιεί και τη λειτουργικότητα του 2ου βήματος.

- **Step 2** – Υλοποιεί την επανάληψη του Step 1 για όλους τους υπο-πίνακες του ar μέχρι τον υπο-πίνακα με 2 στοιχεία (πράσινο μέρος περιγραφής).

  Στην ουσία επαναλαμβάνει την δουλειά του βήματος 1, δηλαδή το moveBubbleUp, για κάθε υπο-πίνακα του ar μέχρι αυτόν με 2 στοιχεία.

Step 2 – MoveBubbleUp for all sub-arrays

Screenshot εκτέλεσης 2ης έκδοσης (1st part)

```
C:\Code\courses\I2P2023-24_
3       5       4       7       2       8       1       0       9       6
Array status: not sorted
Press any key to continue . . .
Iteration No2->3        4       5       7       2       8       1       0       9       6
Iteration No4->3        4       5       2       7       8       1       0       9       6
Iteration No6->3        4       5       2       7       1       8       0       9       6
Iteration No7->3        4       5       2       7       1       0       8       9       6
Iteration No9->3        4       5       2       7       1       0       8       6       9

End of array pass for 10 elements

Press any key to continue . . .
Iteration No3->3        4       2       5       7       1       0       8       6
Iteration No5->3        4       2       5       1       7       0       8       6
Iteration No6->3        4       2       5       1       0       7       8       6
Iteration No8->3        4       2       5       1       0       7       6       8

End of array pass for 9 elements

Press any key to continue . . .
Iteration No2->3        2       4       5       1       0       7       6
Iteration No4->3        2       4       1       5       0       7       6
Iteration No5->3        2       4       1       0       5       7       6
Iteration No7->3        2       4       1       0       5       6       7

End of array pass for 8 elements
```

Αναβαθμίστε τον πηγαίο κώδικα του βήματος 1 (V1) για να υλοποιήσει την εκτέλεση της λειτουργικότητας moveBubbleUp για όλους τους υπο-πίνακες του ar (V2).

Step 2 – MoveBubbleUp for all sub-arrays

Screenshot εκτέλεσης 2ης έκδοσης (last part)

```
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 . . .
```

Παρατηρήστε τις άσκοπες επαναλήψεις για 3 και 2 στοιχεία

Avoid meaningless iterations

Bubble Sort – Step 3 (BubbleSort Version 3)

Δράση 8-13

Αναβαθμίστε την 2η έκδοση της συνάρτησης ώστε να βελτιώσετε το χρόνο ταξινόμησης αποφεύγοντας άσκοπες επαναλήψεις της διεργασίας moveBubbleUp

- **Step 3** – Βελτίωση του βήματος 2 για να αποφύγουμε τις άσκοπες επαναλήψεις

Step 3 – Avoid meaningless iterations

Screenshot εκτέλεσης 3ης έκδοσης (last part)

```
Press any key to continue . . .
Iteration No2->2        1       3       0       4       5
Iteration No3->2        1       0       3       4       5

End of array pass for 6 elements

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

--------------------------------
Process exited after 12.04 seconds with return value 0
Press any key to continue . . .
```

Αναβαθμίστε τον πηγαίο κώδικα του βήματος 2 (V2) για να αποφύγετε την εκτέλεση της λειτουργικότητας moveBubbleUp για τους ταξινομημένους υπο-πίνακες του ar (V3).

Εναλλακτικές Υλοποιήσεις

1. Ένας βρόχος και συνάρτηση moveBubbleUp
2. Δύο βρόχοι (χωρίς τη συνάρτηση moveBubbleUp)
3. Αναδρομικότητα (recursion)

   Η bubbleSort καλεί τον εαυτό της.

4. Ταξινόμηση πίνακα δεικτών σε ακεραίους **?**

```
int *ar[100];
```

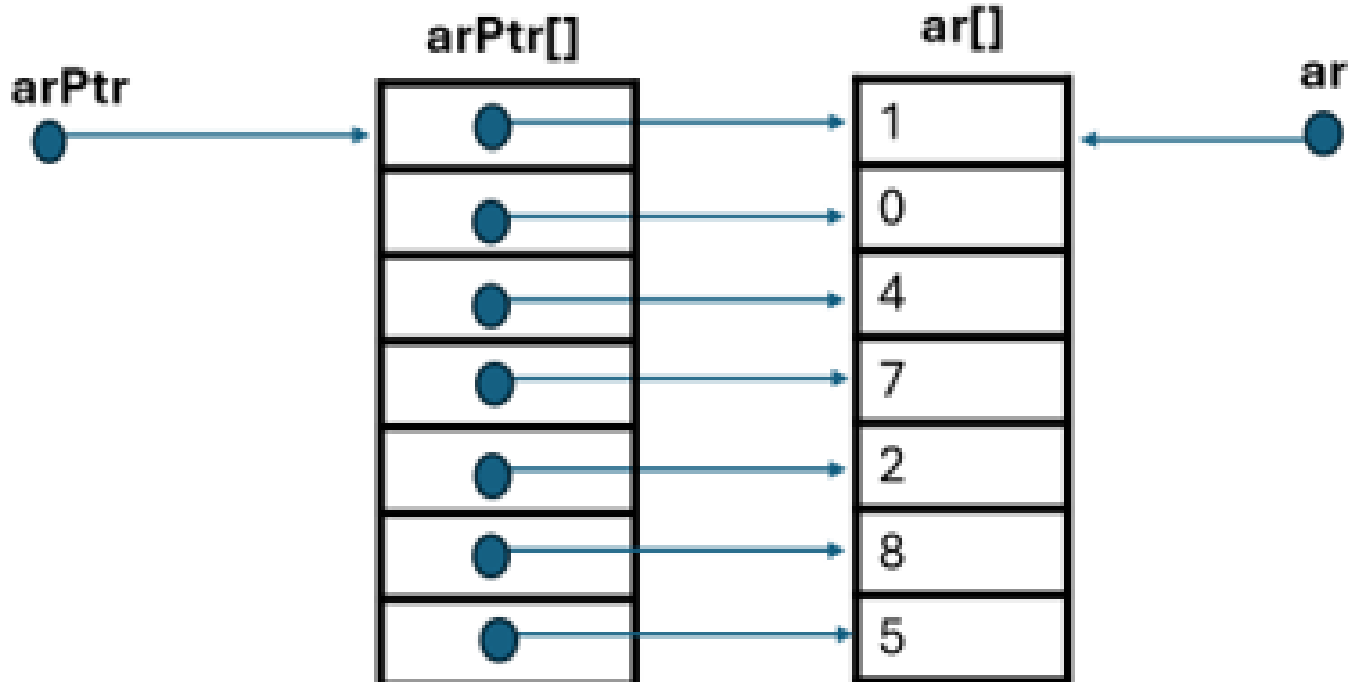
5. Ταξινόμηση Αλφαριθμητικών **?**

```
14  char ar[][10] = {"cde","abc","cef","abb","aab",
15                   "klm","kab","gbc","gab","aaa"};

    char *ar[] = {"cde","abc","cef","abb","aab",
                  "klm","kab","gbc","gab","aaa"};
```

Hands-on: Ταξινόμηση Πίνακα Δεικτών σε int

Δράση 8-15

Αναπτύξτε μια συνάρτηση που θα υλοποιεί τον αλγόριθμο BubbleSort για ταξινόμηση πίνακα δεικτών σε ακεραίους.



Σχήμα 8-4 Οι πίνακες ar και arPtr.

Hands-on: Ταξινόμηση Πίνακα Δεικτών σε int

```
11  int arr[] = {3,5,4,7,2,8,1,0,9,6};          //
12  //int arr[] = {1,0,4,7,2,8,5,3,9,6};    //4
13
14  int main(int argc, char *argv[]) {
15      int numOfElements;
16
17      numOfElements=sizeof(arr)/sizeof(int);
18
19      int *ar[numOfElements];
20      for(int i;i<numOfElements;i++)
21          ar[i]=&arr[i];
```

Δώστε τον πηγαίο κώδικα τροποποιώντας τον κώδικα της bubbleSort για πίνακα ακεραίων

Hands-on: Ταξινόμηση Πίνακα Συμβολοσειρών

Δράση 8-16

Αναπτύξτε μια συνάρτηση που θα υλοποιεί τον αλγόριθμο BubbleSort για ταξινόμηση πίνακα συμβολοσειρών.

```
char ar[][10] = {"cde","abc","cef","abb","aab",
                 "klm","kab","gbc","gab","aaa"};
```

```
C:\Code\courses\I2P2023-24_
cde     abc     cef     abb     aab     klm     kab     gbc     gab     aaa
Array status: not sorted
Press any key to continue . . .
Iteration No1->abc      cde     cef     abb     aab     klm     kab     gbc     gab     aaa
Iteration No3->abc      cde     abb     cef     aab     klm     kab     gbc     gab     aaa
Iteration No4->abc      cde     abb     aab     cef     klm     kab     gbc     gab     aaa
Iteration No6->abc      cde     abb     aab     cef     kab     klm     gbc     gab     aaa
Iteration No7->abc      cde     abb     aab     cef     kab     gbc     klm     gab     aaa
Iteration No8->abc      cde     abb     aab     cef     kab     gbc     gab     klm     aaa
Iteration No9->abc      cde     abb     aab     cef     kab     gbc     gab     aaa     klm
End of array pass for 10 Elements. Array is not sorted

Press any key to continue . . .
Iteration No2->abc      abb     cde     aab     cef     kab     gbc     gab     aaa
Iteration No3->abc      abb     aab     cde     cef     kab     gbc     gab     aaa
Iteration No6->abc      abb     aab     cde     cef     gbc     kab     gab     aaa
Iteration No7->abc      abb     aab     cde     cef     gbc     gab     kab     aaa
Iteration No8->abc      abb     aab     cde     cef     gbc     gab     aaa     kab
End of array pass for 9 Elements. Array is not sorted

Press any key to continue . . .
```

Hands-on: Ταξινόμηση Πίνακα * σε συμβολοσειρές

Δράση 8-17

Αναπτύξτε μια συνάρτηση που θα υλοποιεί τον αλγόριθμο BubbleSort για ταξινόμηση πίνακα δεικτών σε συμβολοσειρές.

```
char *ar[] = {"cde","abc","cef","abb","aab",
              "klm","kab","gbc","gab","aaa"};
```

```
cde     abc     cef     abb     aab     klm     kab     gbc     gab     aaa
Array status: not sorted
Press any key to continue . . .
Iteration No1->abc      cde     cef     abb     aab     klm     kab     gbc     gab     aaa
Iteration No3->abc      cde     abb     cef     aab     klm     kab     gbc     gab     aaa
Iteration No4->abc      cde     abb     aab     cef     klm     kab     gbc     gab     aaa
Iteration No6->abc      cde     abb     aab     cef     kab     klm     gbc     gab     aaa
Iteration No7->abc      cde     abb     aab     cef     kab     gbc     klm     gab     aaa
Iteration No8->abc      cde     abb     aab     cef     kab     gbc     gab     klm     aaa
Iteration No9->abc      cde     abb     aab     cef     kab     gbc     gab     aaa     klm
End of array pass for 10 Elements. Array is not sorted

Press any key to continue . . .
Iteration No2->abc      abb     cde     aab     cef     kab     gbc     gab     aaa
Iteration No3->abc      abb     aab     cde     cef     kab     gbc     gab     aaa
Iteration No6->abc      abb     aab     cde     cef     gbc     kab     gab     aaa
Iteration No7->abc      abb     aab     cde     cef     gbc     gab     kab     aaa
Iteration No8->abc      abb     aab     cde     cef     gbc     gab     aaa     kab
End of array pass for 9 Elements. Array is not sorted

Press any key to continue . . .
Iteration No1->abb      abc     aab     cde     cef     gbc     gab     aaa
Iteration No2->abb      aab     abc     cde     cef     gbc     gab     aaa
Iteration No6->abb      aab     abc     cde     cef     gab     gbc     aaa
Iteration No7->abb      aab     abc     cde     cef     gab     aaa     gbc
End of array pass for 8 Elements. Array is not sorted
```

Οργάνωση Διάλεξης

- **Ταξινόμηση με χρήση της i2p βιβλιοθήκης**
  - Πίνακα ακεραίων
  - Πίνακα αλφαριθμητικών
- **Ταξινόμηση - Bubble Sort**
  - A step-by-step Approach για πίνακα ακεραίων
  - Ενναλακτικές Υλοποιήσεις (+Αναδρομικότητα)
  - Ταξινόμηση πίνακα Δεικτών σε ακεραίους
  - Ταξινόμηση Λέξεων
    - Ταξινόμηση πίνακα αλφαριθμητικών
    - Ταξινόμηση πίνακα Δεικτών σε αλφαριθμητικά
- **Αναζήτηση**
  - Linear - Binary
  - Σε πίνακα ακεραίων
  - Σε πίνακα αλφαριθμητικών
- Ταξινόμηση και Αναζήτηση με την τυπική βιβλιοθήκη

Δραστηριότητα 8.6 - Binary Search

Μελετήστε τον **αλγόριθμο δυαδικής αναζήτησης** και αναπτύξτε μια συνάρτηση που τον υλοποιεί.

- «Binary search begins by comparing an element in the middle of the array with the target value. **If** the target value matches the element, its position in the array is returned. **If** the target value is less than the element, the search continues in the lower half of the array. **If** the target value is greater than the element, the search continues in the upper half of the array. … » wikipedia

Hands-on : Linear Search

Αναπτύξτε ένα πρόγραμμα σύμφωνα με το οποίο το σύστημα θα εκτελεί αναζήτηση με βάση τον αλγόριθμο linear search. Δώστε τη Λεκτική Περιγραφή για τη συνάρτηση αλγόριθμο linearSearch και αναπτύξτε ένα πρόγραμμα LinearSearchTester για αναζήτηση σε πίνακα ακεραίων.

Δώστε και μια εκδοχή για αναζήτηση σε πίνακα δεικτών σε ακεραίους.

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

Διαμορφώστε τον κώδικα έτσι ώστε να έχετε έξοδο ανάλογη με την παρακάτω

```
Linear search for Array of Pointers to Ints
Enter number to search for:6
6 found at index 9
Binary search
arPtr: 0       1       2       3       4       5       6       7       8       9
mid=4
mid=7
mid=5
mid=6
6 found at index 6
```

Hands-on : Binary Search

Δράση 8-18 BinarySearch για πίνακα δεικτών σε ακεραίους

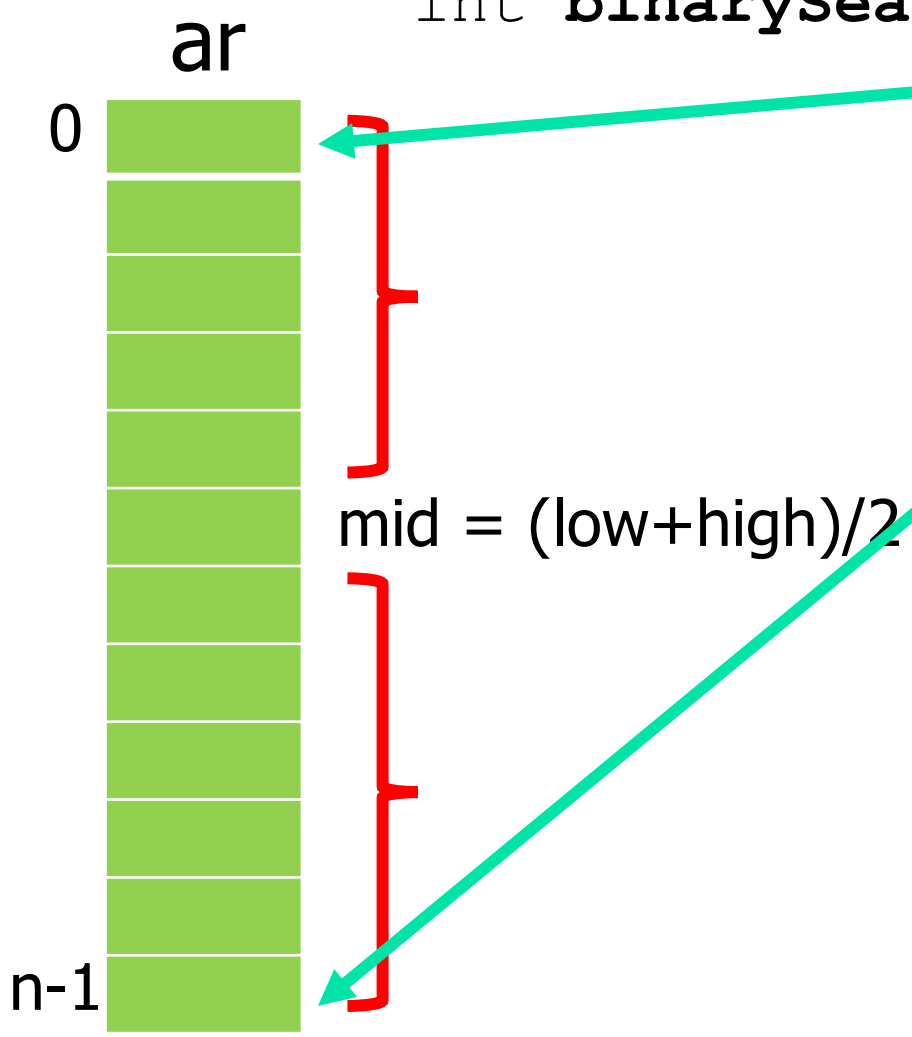
Δώστε τη Λεκτική Περιγραφή για τη συνάρτηση δυαδικής αναζήτησης και αναπτύξτε ένα πρόγραμμα BinarySearchTester για αναζήτηση σε πίνακα δεικτών σε ακεραίους.

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

Διαμορφώστε τον κώδικα έτσι ώστε να έχετε έξοδο ανάλογη με την παρακάτω

```
Linear search for Array of Pointers to Ints
Enter number to search for:6
6 found at index 9
Binary search
arPtr: 0       1       2       3       4       5       6       7       8       9
mid=4
mid=7
mid=5
mid=6
6 found at index 6
```

Binary Search




```
int binarySearch(int ar[], int low,
                 int high, int key);
```

```
mid=(low+high)/2;
if(ar[mid]==key)
        return mid;
if(ar[mid]>key)
        ....
if(ar[mid]<key)
        .....
```

Δώστε τον πηγαίο κώδικα της binarySearch() με και χωρίς αναδρομικότητά

Hands-on : Binary Search

Δράση 8-19 BinarySearch για πίνακα δεικτών σε συμβολοσειρές

Δώστε τη Λεκτική Περιγραφή για τη συνάρτηση δυαδικής αναζήτησης και αναπτύξτε ένα πρόγραμμα BinarySearchTester για αναζήτηση σε πίνακα δεικτών σε συμβολοσειρές.

```
char words[MAX_WORDS][MAX_WORD_LEN] = {"klmn", "lmno",
"abcd", "abc","lmn"};
```

```
BinarySearhcV2 - Linear search on Array of Pointers to Strings
words: klmn     lmno    abcd    abc     lmn
wordsPtr: klmn  lmno    abcd    abc     lmn
Enter word to search for:abcd
abcd found at index 2
Binary search on Array of Pointers to Strings
wordsPtr: abc   abcd    klmn    lmn     lmno
mid=2
mid=0
mid=1
abcd found at index 1
```

Διαμορφώστε τον κώδικα έτσι ώστε να έχετε έξοδο ανάλογη με της εικόνας

Αναζήτηση (Linear-Binary) 4String

```
6   int linearSearch(char *ar[],int numOfElements, char *key);
7   int binarySearch(char *ar[], int low, int high, char *key);
8
9   char *ar[] = {"cde","abc","cef","abb","aab",
10                "klm","kab","gbc","gab","aaa"};
11
12  int main(int argc, char *argv[]) {
13      int index=0;
14      int numOfElements =sizeof(ar)/sizeof(char *);
15      char key[] = "gbc";
```

Οργάνωση Διάλεξης

- **Ταξινόμηση με χρήση της i2p βιβλιοθήκης**
  - Πίνακα ακεραίων
  - Πίνακα αλφαριθμητικών
- **Ταξινόμηση - Bubble Sort**
  - A step-by-step Approach για πίνακα ακεραίων
  - Ενναλακτικές Υλοποιήσεις (+Αναδρομικότητα)
  - Ταξινόμηση πίνακα Δεικτών σε ακεραίους
  - Ταξινόμηση Λέξεων
    - Ταξινόμηση πίνακα αλφαριθμητικών
    - Ταξινόμηση πίνακα Δεικτών σε αλφαριθμητικά
- **Αναζήτηση**
  - Linear - Binary
  - Σε πίνακα ακεραίων
  - Σε πίνακα αλφαριθμητικών
- **Ταξινόμηση και Αναζήτηση με την τυπική βιβλιοθήκη**

Η συνάρτηση memcpy

```
void *memcpy(void *dest, const void * src, size_t n)
```

```
 8  int ar[] = {1,0,4,7,2,8,5,3,9,6};
 9  int arl[10];
10  char words[MAX_WORDS][MAX_WORD_LEN] = {"klmn", "lmno","abcd","abc","lmn"};
11  char wordsl[MAX_WORDS][MAX_WORD_LEN];
12
13  int main(int argc, char *argv[]){
14      printf("QsortHelperFunctionsV2- memcpy (Memory to Memory copy)\n");
15      memcpy(arl,ar,sizeof(ar));
16      printArrayOfInts("arl",arl,sizeof(arl)/sizeof(arl[0]));
17      memcpy(wordsl,words,sizeof(words));
18      printArrayOfStrings("wordl",wordsl,MAX_WORDS);
19      return 0;
20  }
```

Συμπεριφορά ως όρισμα (Δείκτης συνάρτησης- function pointer)

- Το **όνομα μιας συνάρτησης είναι δείκτης στο σώμα της συνάρτησης** και ως δείκτης μπορεί να περάσει ως όρισμα σε μία συνάρτηση.
- Μπορούμε να περάσουμε σε μια συνάρτηση func1 τον δείκτη μιας άλλης συνάρτησης func2.
- Τον δείκτη αυτόν μπορεί να χρησιμοποιήσει η func1 στο σώμα της για να καλέσει την func2.
- Με τον τρόπο αυτό έχουμε τη δυνατότητα να αλλάζουμε τη συμπεριφορά της συνάρτησης func1 κατά το χρόνο εκτέλεσης του προγράμματος.

```
void flexiblePrint(int *ar, int size,void(*print)(int *, int ))
```

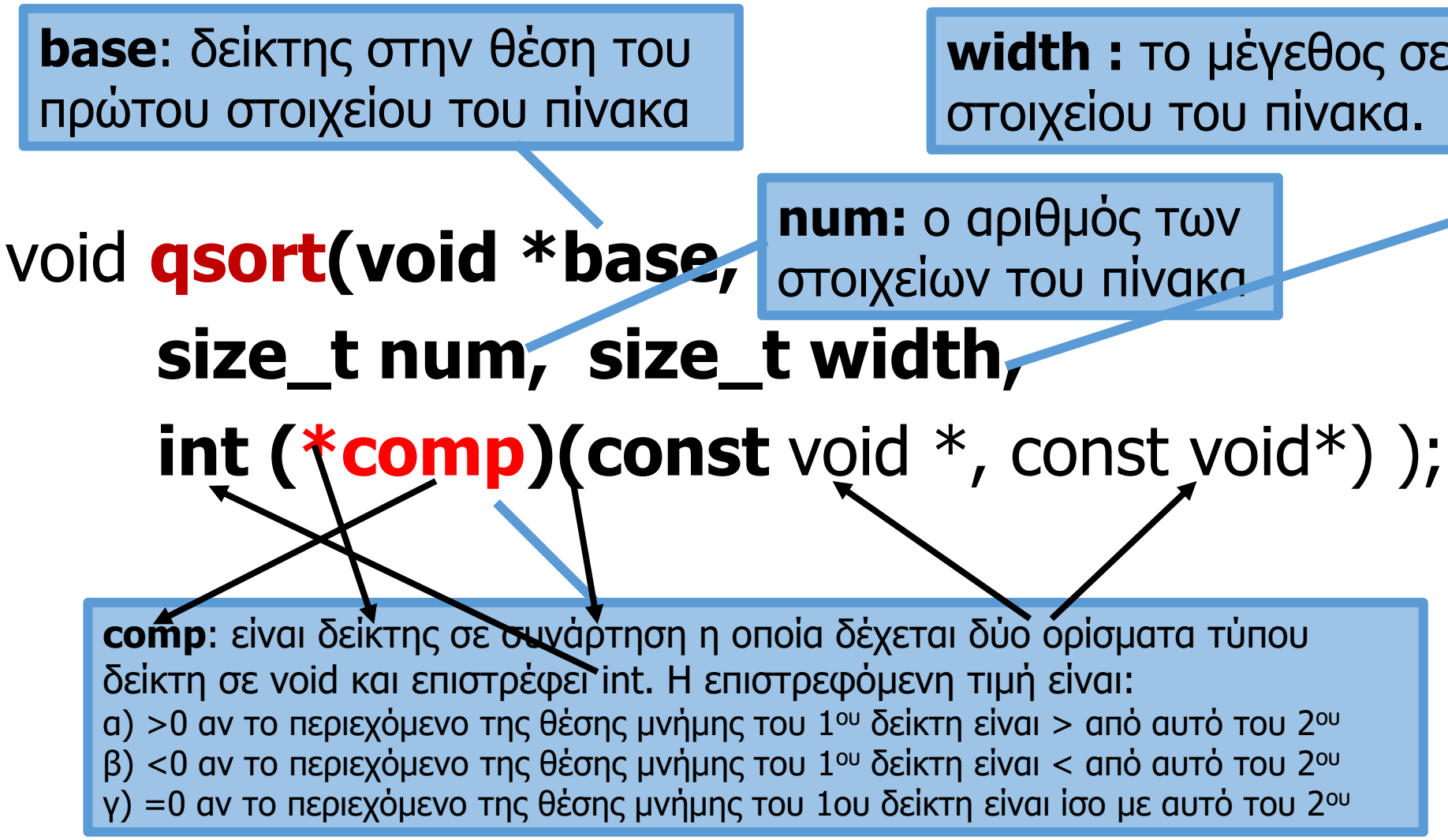
Hands-on : BubbleSort

Δράση 8-20 Μια BubbleSort για ταξινόμηση κάθε τύπου πίνακα

Αν έχετε ολοκληρώσει επιτυχώς τις παραπάνω δράσεις ταξινόμησης, έχετε το υπόβαθρο που απαιτείται για να εξετάσετε το ενδεχόμενο ορισμού συνάρτησης που θα υλοποιεί τον αλγόριθμο BubbleSort για ταξινόμηση πίνακα στοιχείων οποιουδήποτε τύπου. Είναι ένα δύσκολο έργο. Σημειώστε τις παρατηρήσεις σας και τολμήστε να περάσετε σε κώδικα.

Αν συναντήσετε δυσκολίες, είναι πολύ πιθανό, η ενότητα 8.8 θα σας βοηθήσει πολύ να την ολοκληρώσετε. Προς το παρόν συνεχίστε με την qsort.

Η συνάρτηση qsort



Hands-on : qsort

Δράση 8-21 Η qsort για ταξινόμηση πίνακα ακεραίων

Αναπτύξτε ένα μικρό πρόγραμμα που θα χρησιμοποιεί τη συνάρτηση qsort για να ταξινομήσει ένα πίνακα ακεραίων.

```
Array before sorting
[0]->9
[1]->4
[2]->12
[3]->2
[4]->17
[5]->21
[6]->8
[7]->10
[8]->3

Array after sorting
[0]->2
[1]->3
[2]->4
[3]->8
[4]->9
[5]->10
[6]->12
[7]->17
[8]->21
```

Hands-on : qsort για ταξινόμηση πίνακα ακεραίων

Δράση 8-21 Η qsort για ταξινόμηση πίνακα ακεραίων

Αναπτύξτε ένα μικρό πρόγραμμα που θα χρησιμοποιεί τη συνάρτηση qsort για να ταξινομήσει ένα πίνακα ακεραίων.

- Οδηγίες:
  - Γράψτε μια συνάρτηση **displayIntArray** η οποία θα τυπώνει τον πίνακα με την μορφή που δίνεται παραπλεύρως.
  - Γράψτε μια συνάρτηση **compInt** η οποία
    - α) θα έχει το function prototype που ορίζει η qsort και
    - β) θα κάνει cast σε int * τα ορίσματα της, θα συγκρίνει τα περιεχόμενα των θέσεων μνήμης που αυτά δείχνουν και θα επιστρέφει: 0 αν είναι ίσα, >0 αν το πρώτο είναι μεγαλύτερο και <0 αν το πρώτο είναι μικρότερο.
  - Τροποποιήστε την compInt ώστε η qsort να ταξινομεί με φθίνουσα σειρά

```
Array before sorting
[0]->9
[1]->4
[2]->12
[3]->2
[4]->17
[5]->21
[6]->8
[7]->10
[8]->3
```

```
Array after sorting
[0]->2
[1]->3
[2]->4
[3]->8
[4]->9
[5]->10
[6]->12
[7]->17
[8]->21
```

Hands-on : qsort για ταξινόμηση πίνακα ακεραίων

```
18 void displayIntArray(int ar[], int s){
19     int i=0;
20     for(i=0;i<s;i++)
21         printf("[%d]->%d\n",i,ar[i]);
22 }
23
24 int compInt(const void *a, const void*b){
25     if(*(int*)a==*(int*)b)
26         return 0;
27     return *(int*)a>*(int*)b?1:-1;
28 //  return *(int*)a - *(int*)b;
29 }
```

```
Array before sorting
[0]->9
[1]->4
[2]->12
[3]->2
[4]->17
[5]->21
[6]->8
[7]->10
[8]->3
```

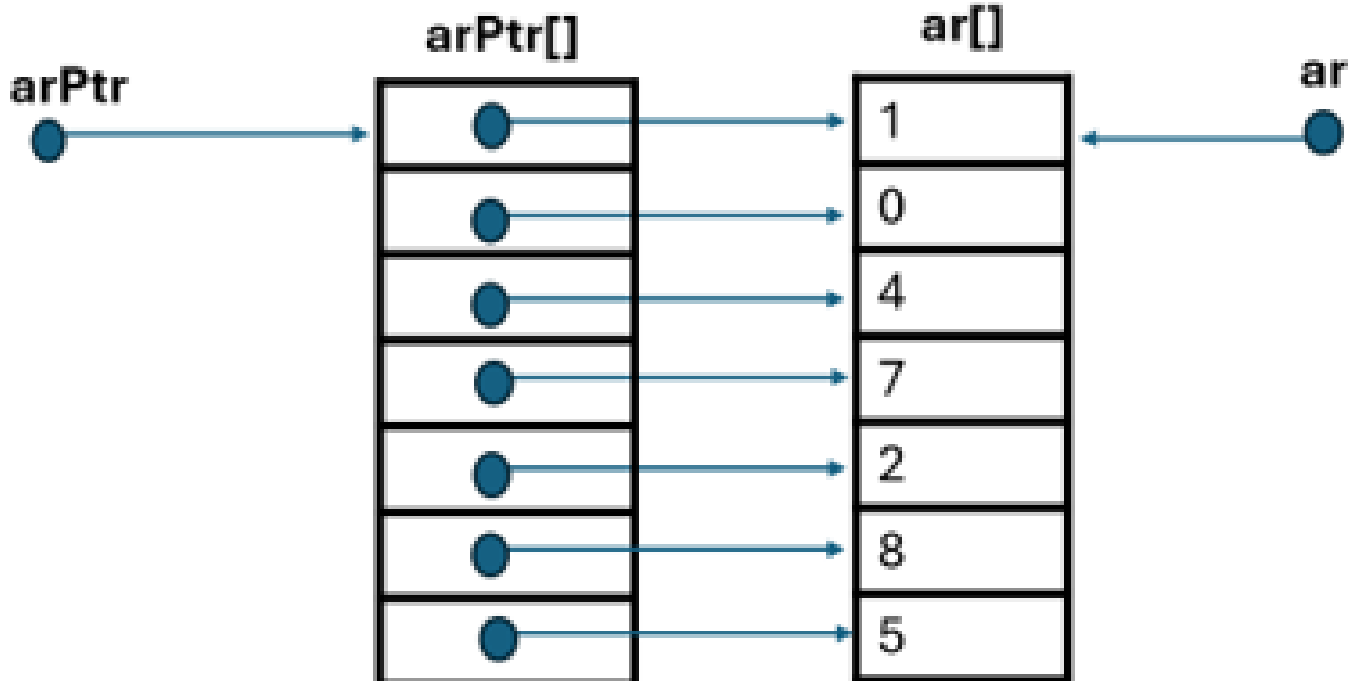
```
Array after sorting
[0]->2
[1]->3
[2]->4
[3]->8
[4]->9
[5]->10
[6]->12
[7]->17
[8]->21
```

Η έκφραση κάνει cast τον δείκτη a από δείκτη σε void σε δείκτη σε int και παίρνει το περιεχόμενο του.

Hands-on : qsort (πίνακα δεικτών σε int)

Δράση 8-22 Η qsort για ταξινόμηση πίνακα δεικτών σε ακεραίους

Αναπτύξτε ένα πρόγραμμα που θα χρησιμοποιεί τη συνάρτηση qsort για να ταξινομήσει ένα πίνακα δεικτών σε ακεραίους.



Σχήμα 8-4 Οι πίνακες ar και arPtr.

Hands-on : qsort για ταξινόμηση πίνακα συμβολοσειρών

Δράση 8-23 Η qsort για ταξινόμηση πίνακα συμβολοσειρών

Αναπτύξτε ένα πρόγραμμα που θα χρησιμοποιεί τη συνάρτηση qsort για να ταξινομήσει ένα πίνακα συμβολοσειρών. Στο ίδιο πρόγραμμα αξιοποιήστε τις συναρτήσεις της τυπικής βιβλιοθήκης για σειριακή και δυαδική αναζήτηση σε πίνακες.

- Οδηγίες:
  - Γράψτε μια συνάρτηση **displayStringArray** η οποία θα τυπώνει τον πίνακα με την μορφή που δίνεται παραπλεύρως.
  - Γράψτε μια συνάρτηση **compString** η οποία
    - α) θα έχει το function prototype που ορίζει η qsort και
    - β) θα κάνει cast σε char * τα ορίσματα της, θα συγκρίνει τα περιεχόμενα των θέσεων μνήμης που αυτά δείχνουν και θα επιστρέφει: 0 αν είναι ίσα, >0 αν το πρώτο είναι μεγαλύτερο και <0 αν το πρώτο είναι μικρότερο.
  - Τροποποιήστε την compString ώστε η qsort να ταξινομεί με φθίνουσα σειρά

```
Array before sorting
[0]->paris
[1]->nikos
[2]->alekos
[3]->kostas
[4]->basos
[5]->andreas
[6]->petros

Array after sorting
[0]->alekos
[1]->andreas
[2]->basos
[3]->kostas
[4]->nikos
[5]->paris
[6]->petros
```

Hands-on : qsort για ταξινόμηση πίνακα συμβολοσειρών

QsortArrayString.c

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

void displayStringArray(char ar[][10], int s);
int compString(const void *, const void*);
char ar[][10]={"paris","nikos","alekos","kostas","basos","andreas","petros"};


int main(int argc, char *argv[]) {
    printf("Array before sorting\n");
    displayStringArray(ar,sizeof(ar)/sizeof(ar[0]));

    qsort(ar,sizeof(ar)/sizeof(ar[0]),sizeof(ar[0]),compString);

    printf("\nArray after sorting\n");
    displayStringArray(ar,sizeof(ar)/sizeof(ar[0]));
    return 0;
}

```

```
Array before sorting
[0]->paris
[1]->nikos
[2]->alekos
[3]->kostas
[4]->basos
[5]->andreas
[6]->petros

Array after sorting
[0]->alekos
[1]->andreas
[2]->basos
[3]->kostas
[4]->nikos
[5]->paris
[6]->petros
```

Hands-on : qsort για ταξινόμηση πίνακα συμβολοσειρών

```
13      qsort(ar,sizeof(ar)/sizeof(ar[0]),sizeof(ar[0]),compString);
14
15      printf("\nArray after sorting\n");
16      displayStringArray(ar,sizeof(ar)/sizeof(ar[0]));
17      return 0;
18  }
19
20  void displayStringArray(char ar[][10], int s){
21      int i=0;
22      for(i=0;i<s;i++)
23          printf("[%d]->%s\n",i,ar[i]);
24  }
25
26  int compString(const void *a, const void*b){
27      return strcmp((char*)a,(char*)b);
28  }
```

Δραστηριότητα 8.5 - BubbleSort4Arrays

Αναπτύξτε μια συνάρτηση **bsort** ανάλογη της qsort της τυπικής βιβλιοθήκης με τη διαφορά ότι θα υλοποιεί τον αλγόριθμο bubble sort.

Επιλέξτε να δουλέψετε για την ανάπτυξη της bsort με ένα πίνακα ακεραίων.
Μια καλή επιλογή είναι να χρησιμοποιήσετε ως βάση τον πηγαίο κώδικα της 3ης έκδοσης της BubbleSort

Hands-on : qsort για ταξινόμηση πίνακα Δομών

Δράση 8-25 (9-19) Η qsort για ταξινόμηση πίνακα Δομών

Αναπτύξτε ένα πρόγραμμα που θα χρησιμοποιεί τη συνάρτηση qsort για να ταξινομήσει ένα πίνακα δομών.

Address Book

Να εκτελεστεί μετά την ενότητα 9 (Δομές)

```
struct abEntry{
    int am;
    char lname[20];
    char fname[30];
    char address[40];
    long int zip;
    char email[20];
    };
```

```
struct abEntry ab[MAX_ENTRIES];
int abFreeEntry=0;
```

Η τάξη του πρώτου ελεύθερου στοιχείου του πίνακα ab.

Πίνακας ab

	AM	Lname	Fname	Address	zip	email
0						
1						
2						
3						
4						
5						

Ταξινόμηση πίνακα Δομών

```
void qsort(void *base,
        size_t num,  size_t width,
        int (*comp)(const void *, const void*) );
```

	AM	Lname	Fname	Address	zip	email
0						
1						
2						
3						
4						
5						

Αριθμός εγγραφών πίνακα

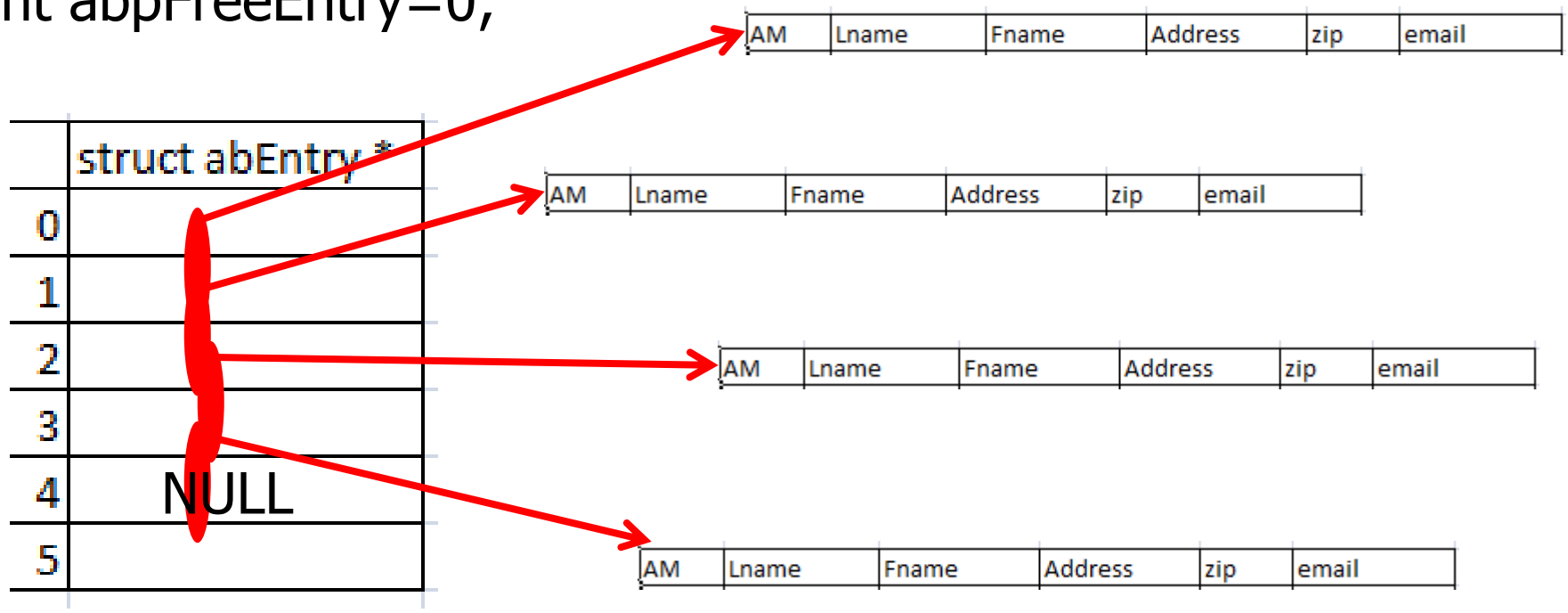
Μέγεθος σε bytes της εγγραφής abEntry

Συνάρτηση σύγκρισης του μέλους της abEntry με βάση το οποίο θέλουμε να γίνει η ταξινόμηση. Δέχεται ως ορίσματα δύο δείκτες σε δύο εγγραφές του πίνακα. Θα συγκρίνει τα μέλη με βάση τα οποία θέλουμε να γίνει η ταξινόμηση και θα επιστρέφει ανάλογα.

Ταξινόμηση πίνακα δεικτών σε Δομές 1/3

struct abEntry *abp[MAX_ENTRIES];

int abpFreeEntry=0;



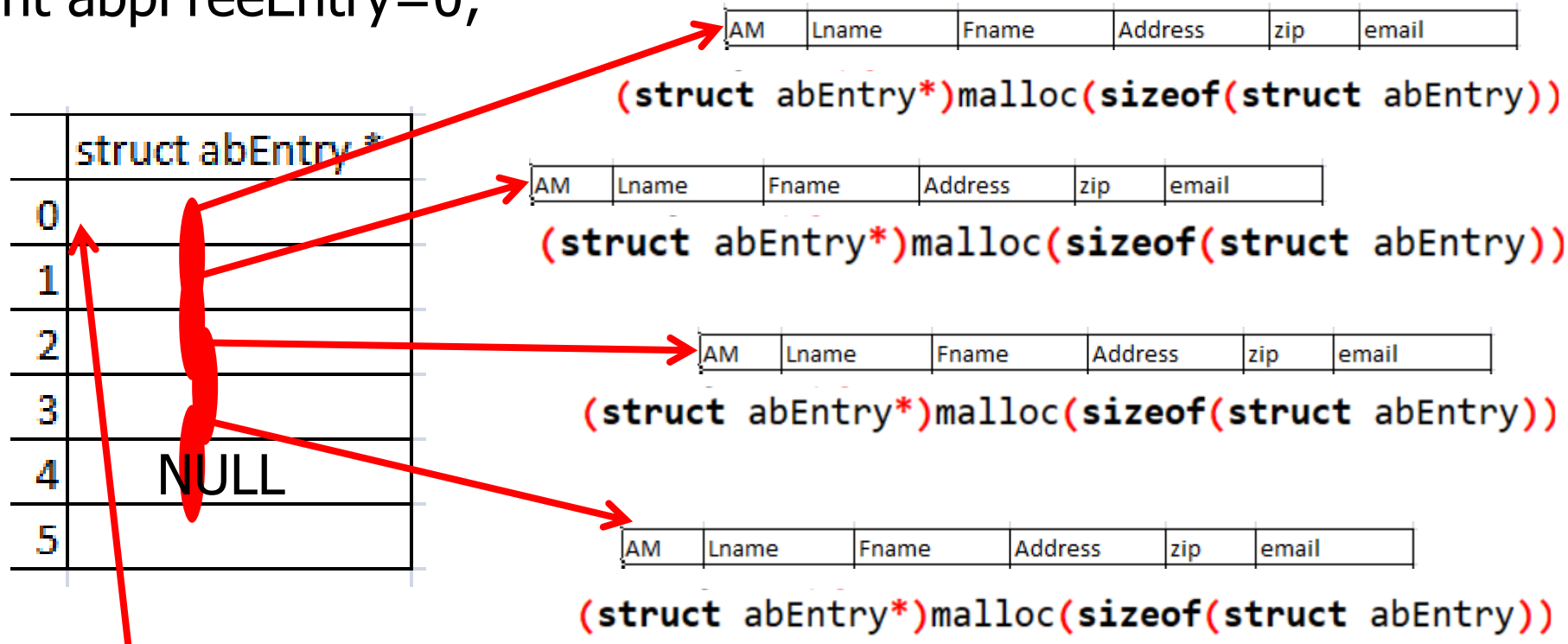
abpFreeEntry is 4

```
#include <stdlib.h>
#include <malloc.h>
void *malloc(size_t size);
void free(void *buffer);
```

Ταξινόμηση πίνακα δεικτών σε Δομές 2/3

struct abEntry *abp[MAX_ENTRIES];

int abpFreeEntry=0;

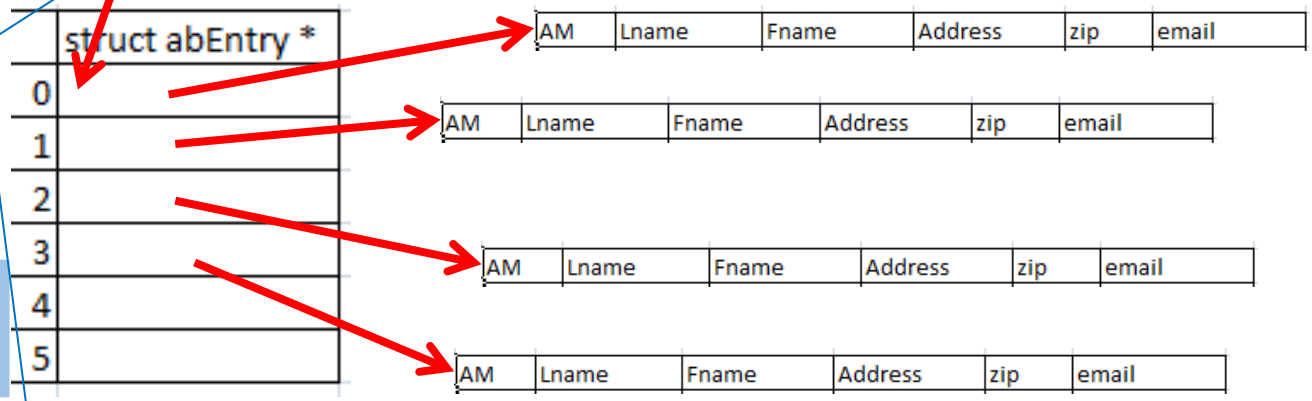


a void *a = (void*)abp;

```
(*(struct abEntry**)a)->am
```

Ταξινόμηση πίνακα δεικτών σε Δομές 3/3

```
void qsort(void *base,
        size_t num,  size_t width,
        int (*comp)(const void *, const void*) );
```



Αριθμός εγγραφών πίνακα abp

Μέγεθος σε bytes του στοιχείου του πίνακα abp

Συνάρτηση σύγκρισης του μέλους της abEntry με βάση το οποίο θέλουμε να γίνει η ταξινόμηση. Δέχεται ως ορίσματα δύο δείκτες σε δύο δείκτες σε εγγραφές abEntry. Θα συγκρίνει τα μέλη των εγγραφών με βάση τα οποία θέλουμε να γίνει η ταξινόμηση και θα επιστρέφει ανάλογα.

Αναζήτηση – bsearch() 1/2

```
void *bsearch(const void *key,
        void *base, size_t num,  size_t width,
        int (*comp)(const void *, const void*) );
```

Συνάρτηση της βασικής βιβλιοθήκης η οποία δηλώνεται στο αρχείο **`stdlib.h`**

Τα στοιχεία του πίνακα θα πρέπει να έχουν ταξινομηθεί προηγουμένως με την ίδια συνάρτηση comp.

Αναζήτηση – bsearch() 2/2

key: δείκτης στο κλειδί με βάση το οποίο θα γίνει η αναζήτηση

base: δείκτης στην θέση του πρώτου στοιχείου του πίνακα

num: ο αριθμός των στοιχείων του πίνακα

width : το μέγεθος σε bytes του στοιχείου του πίνακα.

```
void *bsearch(const void *key
        void *base, size_t num,  size_t width,
        int (*comp)(const void *, const void*) );
```

comp: είναι δείκτης σε συνάρτηση η οποία δέχεται δύο ορίσματα τύπου δείκτη σε void και επιστρέφει int. Η επιστρεφόμενη τιμή είναι:
α) >0 αν το περιεχόμενο της θέσης μνήμης του 1ου δείκτη είναι > από αυτό του 2ου
β) <0 αν το περιεχόμενο της θέσης μνήμης του 1ου δείκτη είναι < από αυτό του 2ου
γ) =0 αν το περιεχόμενο της θέσης μνήμης του 1ου δείκτη είναι ίσο με αυτό του 2ου

comp: Συνάρτηση σύγκρισης του key με τις εγγραφές του πίνακα που δείχνει η base.

Αναζήτηση σε πίνακα Δομών

```
void* bsearch(const void *key,
              void *base,
              size_t num,
              size_t width,
              int (*comp)(const void*, const void*));
```

AM	Lname	Fname	Address	zip	email

	AM	Lname	Fname	Address	zip	email
0						
1						
2						
3						
4						
5						

Συνάρτηση σύγκρισης του key με τις εγγραφές του πίνακα που δείχνει η base.

Αναζήτηση σε πίνακα * σε Δομές

```
void* bsearch(const void *key,
              void *base,
              size_t num,
              size_t width,
              int (*comp)(const void*, const void*));
```

```
struct abEntry *keyPtr = &key;
```

AM	Lname	Fname	Address	zip	email

```
struct abEntry key;
```

	struct abEntry *
0	
1	
2	
3	
4	
5	

AM	Lname	Fname	Address	zip	email

AM	Lname	Fname	Address	zip	email

AM	Lname	Fname	Address	zip	email

AM	Lname	Fname	Address	zip	email