Εισαγωγή στον Προγραμματισμό

Ιστότοπος μαθήματος
https://eclass.upatras.gr/courses/CEID1247/

(CEID_NY131)

Ανάπτυξη εφαρμογής διαχείρισης Καταλόγου Διευθύνσεων (**Address Book**)

Kleanthis Thramboulidis
Prof. of Software and System Engineering
University of Patras
https://sites.google.com/site/thramboulidiskleanthis/

Ο στόχος

- Ταξινόμηση (sort)
- Αναζήτηση (search)
- Δυναμική διαχείριση μνήμης (dynamic memory allocation)
- Αποθήκευση (save) και ανάκληση (load) πληροφορίας από δίσκο (low level file handling)
- Δυναμικές Δομές Δεδομένων – Συνδεδεμένες λίστες
  - Απλά συνδεδεμένη λίστα (singly linked list)
  - Διπλά συνδεδεμένη λίστα (doubly linked list)

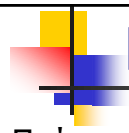
Οργάνωση Διάλεξης

- **Η εφαρμογή Address Book**
- Ταξινόμηση - Αναζήτηση
- Παραδείγματα ταξινόμησης
  - Ταξινόμηση πίνακα ακεραίων
  - Ταξινόμηση πίνακα αλφαριθμητικών
- Procedural and Data Abstraction στην εφαρμογή Address Book
- Εναλλακτικές υλοποιήσεις
  - Πίνακας εγγραφών
  - Πίνακας δεικτών σε εγγραφές
  - Απλά συνδεδεμένη λίστα
  - Διπλά συνδεδεμένη λίστα

Κατάλογος Διευθύνσεων

Address Book A-B-C

Name
Street Address
City: State: Zip Code:
Home Phone: Cell Phone:
Work Phone: Email:



Η εφαρμογή

Πρόγραμμα (εφαρμογή) διαχείρισης διευθύνσεων-τηλεφώνων.
(άσκηση 6 του κεφαλαίου 12 του βιβλίου Διαδικαστικός Προγραμματισμός – C.)

- Η εφαρμογή θα υποστηρίζει τις παρακάτω λειτουργίες:
  - Εισαγωγή νέας εγγραφής,
  - Εμφάνισης των εγγραφών
  - Τροποποίηση εγγραφής
  - Αποθήκευση των εγγραφών σε αρχείο
  - Ανάκληση εγγραφών από αρχείο
  - Ταξινόμηση των εγγραφών (με χρήση της qsort της βασικής βιβλιοθήκης)
  - Αναζήτηση εγγραφής (με χρήση της bsearch της βασικής βιβλιοθήκης)

Οργάνωση Διάλεξης

- Η εφαρμογή Address Book
- **Ταξινόμηση - Αναζήτηση**
- Παραδείγματα ταξινόμησης
  - Ταξινόμηση πίνακα ακεραίων
  - Ταξινόμηση πίνακα αλφαριθμητικών
- Procedural and Data Abstraction στην εφαρμογή Address Book
- Εναλλακτικές υλοποιήσεις
  - Πίνακας εγγραφών
  - Πίνακας δεικτών σε εγγραφές
  - Απλά συνδεδεμένη λίστα
  - Διπλά συνδεδεμένη λίστα

Ταξινόμηση - qsort

```
void qsort(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`**

The qsort() function is **an implementation of C.A.R. Hoare's quicksort algorithm**, a variant of partition-exchange sorting.
The algorithm is taken from *Algorithm Q* in D.E. Knuth's *Sorting and Searching*.

Κεφάλαιο 11, σελ. 352, ενότητα 11.4.5

Ταξινόμηση - qsort

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 qsort(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

Αναζήτηση - bsearch

```
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.

Κεφάλαιο 11, σελ. 352, ενότητα 11.4.5

Αναζήτηση - bsearch

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.

Οργάνωση Διάλεξης

- Η εφαρμογή Address Book
- Ταξινόμηση - Αναζήτηση
- **Παραδείγματα ταξινόμησης**
  - Ταξινόμηση πίνακα ακεραίων
  - Ταξινόμηση πίνακα αλφαριθμητικών
- Procedural and Data Abstraction στην εφαρμογή Address Book
- Εναλλακτικές υλοποιήσεις
  - Πίνακας εγγραφών
  - Πίνακας δεικτών σε εγγραφές
  - Απλά συνδεδεμένη λίστα
  - Διπλά συνδεδεμένη λίστα

Ταξινόμηση πίνακα ακεραίων

- Γράψτε ένα πρόγραμμα για την ταξινόμηση ενός πίνακα ακεραίων με τη χρήση της qsort.
- Σας δίνεται ενδεικτική έξοδος του προγράμματος.
- 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
```

Ταξινόμηση πίνακα ακεραίων

- Βοήθεια σε τρία βήματα
  - Πάρτε τη βοήθεια μόνο μετά από δική σας προσπάθεια και βήμα προς βήμα. Σε κάθε περίπτωση μετά την ολοκλήρωση της άσκησης δείτε και τα τρία βήματα.
- συνάρτηση **displayIntArray**
  - Σώμα συνάρτησης
    - https://eclass.upatras.gr/modules/document/file.php/CEID1247/Resources/displayIntArrayBody.jpg
  - Δήλωση (function prototype)
    - .../Resources/displayIntArrayFunctionPrototype.jpg
  - Κλήση συνάρτησης
    - .../Resources/displayIntArrayFunctionCall.jpg
- συνάρτηση **compInt**
  - Δήλωση (function prototype)
    - .../Resources/compIntFunctionPrototype.jpg
  - Σώμα συνάρτησης
    - ... /Resources/compIntFunctionBody.jpg
  - Κλήση συνάρτησης
    - Την συνάρτηση δεν την καλείτε εσείς αλλά η qsort()

Ταξινόμηση πίνακα αλφαριθμητικών

- Γράψτε ένα πρόγραμμα για την ταξινόμηση ενός πίνακα αλφαριθμητικών με τη χρήση της qsort.
- Σας δίνεται ενδεικτική έξοδος του προγράμματος.
- 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
```

Οργάνωση Διάλεξης

- Η εφαρμογή Address Book
- Ταξινόμηση - Αναζήτηση
- Παραδείγματα ταξινόμησης
  - Ταξινόμηση πίνακα ακεραίων
  - Ταξινόμηση πίνακα αλφαριθμητικών
- **Procedural and Data Abstraction** στην εφαρμογή Address Book
- Εναλλακτικές υλοποιήσεις
  - Πίνακας εγγραφών
  - Πίνακας δεικτών σε εγγραφές
  - Απλά συνδεδεμένη λίστα
  - Διπλά συνδεδεμένη λίστα

Procedural Abstraction

getAbEntry()

appendAbEntry(struct abEntry abe)

displayAbEntry()

displayAB()

decSort(void)

incSort(void)

findAbEntry( )

loadAddressBook()

saveAddressBook ()

Data Abstraction

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


Οργάνωση Διάλεξης

- Η εφαρμογή Address Book
- Ταξινόμηση - Αναζήτηση
- Παραδείγματα ταξινόμησης
  - Ταξινόμηση πίνακα ακεραίων
  - Ταξινόμηση πίνακα αλφαριθμητικών
- Procedural and Data Abstraction στην εφαρμογή Address Book
- **Εναλλακτικές υλοποιήσεις**
  - Πίνακας εγγραφών
  - Πίνακας δεικτών σε εγγραφές
  - Απλά συνδεδεμένη λίστα
  - Διπλά συνδεδεμένη λίστα

Version I – Array of structs

```
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 με στοιχεία abEntry

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

Αριθμός εγγραφών πίνακα

Μέγεθος σε bytes της εγγραφής abEntry

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

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.

Κεφάλαιο 11, σελ. 352, ενότητα 11.4.5

bsearch

```
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.

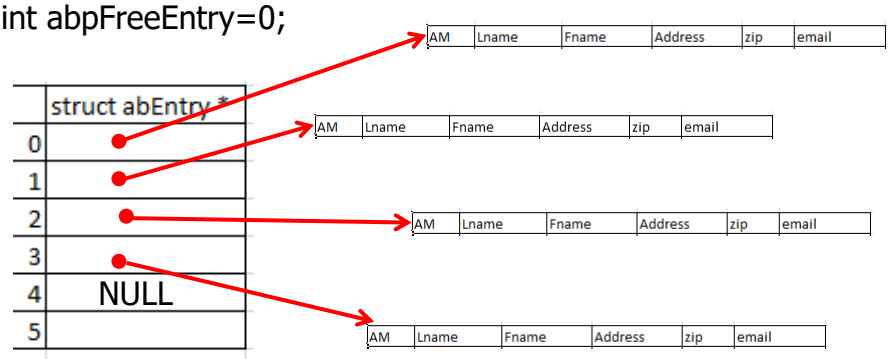
Οργάνωση Διάλεξης

- Η εφαρμογή Address Book
- Ταξινόμηση - Αναζήτηση
- Παραδείγματα ταξινόμησης
  - Ταξινόμηση πίνακα ακεραίων
  - Ταξινόμηση πίνακα αλφαριθμητικών
- Procedural and Data Abstraction στην εφαρμογή Address Book
- **Εναλλακτικές υλοποιήσεις**
  - Πίνακας εγγραφών
  - **Πίνακας δεικτών σε εγγραφές**
  - Απλά συνδεδεμένη λίστα
  - Διπλά συνδεδεμένη λίστα

Version II – Array of pointers to structs

struct abEntry *abp[MAX_ENTRIES];

int abpFreeEntry=0;



abpFreeEntry is 4

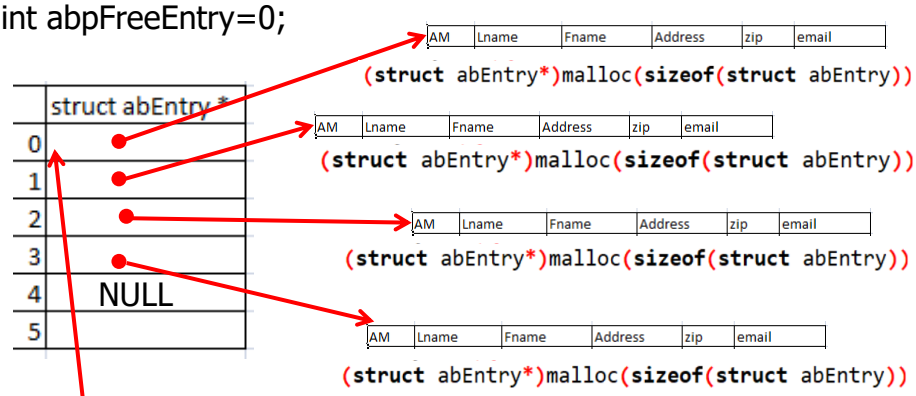
```
#include <stdlib.h>
#include <malloc.h>
void *malloc(size_t size);
void free(void *buffer);
```

Κεφάλαιο 9, σελ. 298, ενότητα 9.12.2

Data Abstraction (version II)

struct abEntry *abp[MAX_ENTRIES];

int abpFreeEntry=0;

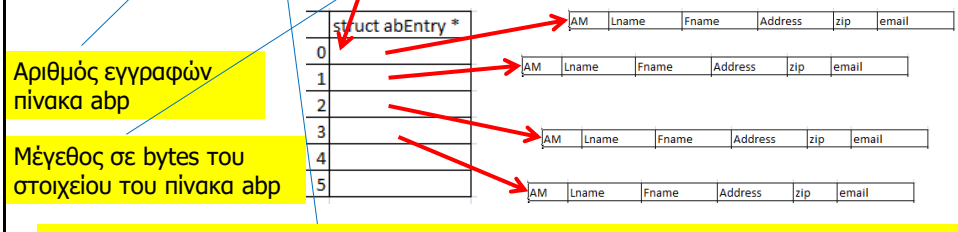


a [ ] void *a = (void*)abp;

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

Ταξινόμηση πίνακα abpEntry

```
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

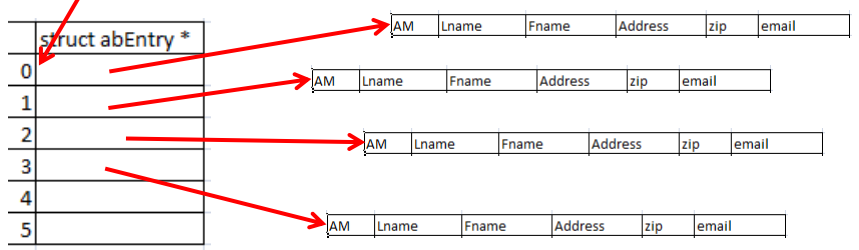
Συνάρτηση σύγκρισης του μέλους της abEntry με βάση το οποίο θέλουμε να γίνει η ταξινόμηση. Δέχεται ως ορίσματα δύο δείκτες σε δύο δείκτες σε εγγραφές abEntry. Θα συγκρίνει τα μέλη των εγγραφών με βάση τα οποία θέλουμε να γίνει η ταξινόμηση και θα επιστρέφει ανάλογα.

Κεφάλαιο 11, σελ. 352, ενότητα 11.4.5

bsearch

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

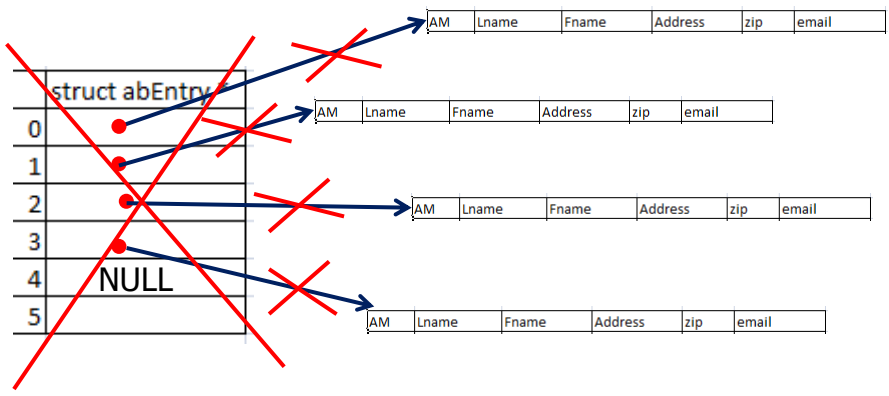
```
struct abEntry *keyPtr = &key;
struct abEntry key;
```



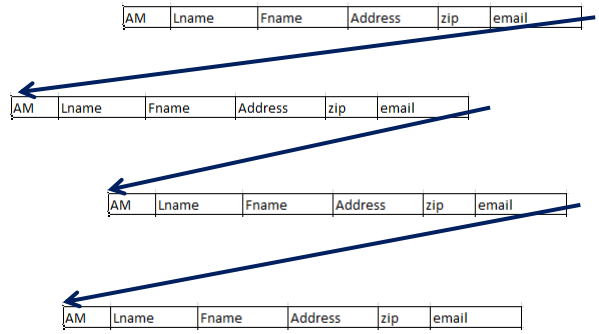
Οργάνωση Διάλεξης

- Η εφαρμογή Address Book
- Ταξινόμηση - Αναζήτηση
- Παραδείγματα ταξινόμησης
  - Ταξινόμηση πίνακα ακεραίων
  - Ταξινόμηση πίνακα αλφαριθμητικών
- Procedural and Data Abstraction στην εφαρμογή Address Book
- **Εναλλακτικές υλοποιήσεις**
  - Πίνακας εγγραφών
  - Πίνακας δεικτών σε εγγραφές
  - **Απλά συνδεδεμένη λίστα**
  - Διπλά συνδεδεμένη λίστα

Version III – singly linked list

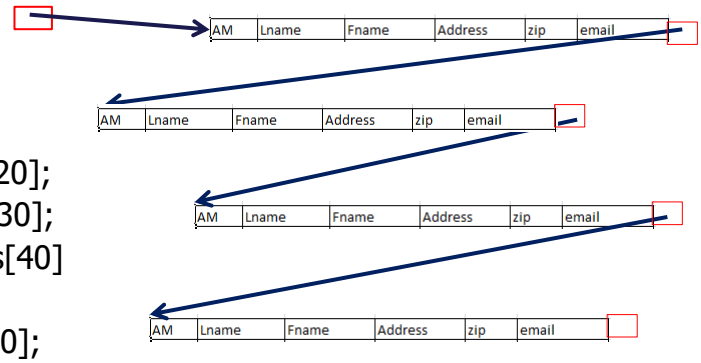


Version III – singly linked list



Version III – singly linked list

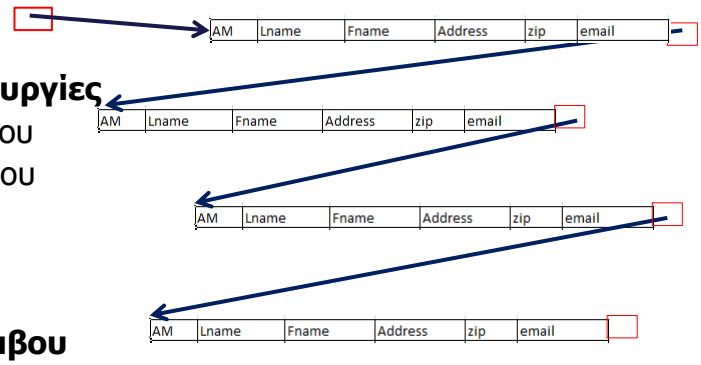
Απλά συνδεδεμένη λίστα




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

Singly linked list - Βασικές λειτουργίες

Απλά συνδεδεμένη λίστα



Βασικές λειτουργίες

Εισαγωγή κόμβου

Διαγραφή κόμβου

Εισαγωγή κόμβου

Εισαγωγή ως πρώτου στοιχείου

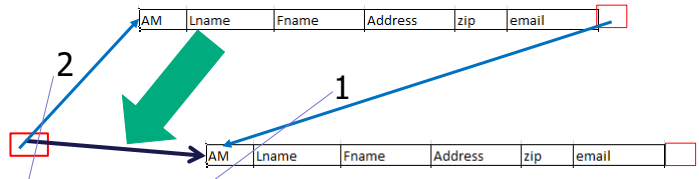
Εισαγωγή ως ενδιάμεσου στοιχείου

Εισαγωγή ως τελευταίου στοιχείου

Singly linked list - Insert node 1/3

```
typedef struct item ITEM;
typedef ITEM *PITEM;

void insert_item(PITEM *head_ptr, PITEM p)
{
PITEM p_cur;

if(*head_ptr = = NULL) {   /* η λίστα είναι άδεια */
        *head_ptr = p;          /* εισάγει το στοιχείο στη λίστα */
        return;
        }
....
```

Singly linked list - Insert node 2/3




```
...
/* το στοιχείο πρέπει να τοποθετηθεί πρώτο */
if(strcmp(p->name,(*head_ptr)->name)<0){
p->next = *head_ptr;
*head_ptr = p;
 return;
        }

....
```


Singly linked list - Insert node 3/3

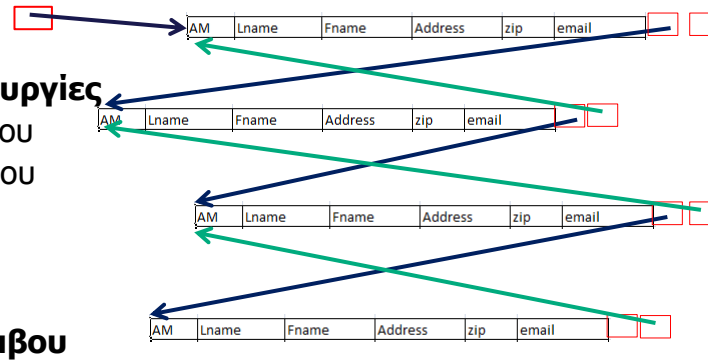
```
...

/* βρίσκει τη θέση του νέου στοιχείου στη λίστα */
p_cur = *head_ptr;
while((p_cur->next!=NULL) && strcmp(p_cur->next->name,p->name)<0)
        p_cur = p_cur->next;

/* το εισάγει στη λίστα μετά το p_cur*/
p->next = p_cur->next;
p_cur->next = p;
}
```

Οργάνωση Διάλεξης

- Η εφαρμογή Address Book
- Ταξινόμηση - Αναζήτηση
- Παραδείγματα ταξινόμησης
  - Ταξινόμηση πίνακα ακεραίων
  - Ταξινόμηση πίνακα αλφαριθμητικών
- Procedural and Data Abstraction στην εφαρμογή Address Book
- **Εναλλακτικές υλοποιήσεις**
  - Πίνακας εγγραφών
  - Πίνακας δεικτών σε εγγραφές
  - Απλά συνδεδεμένη λίστα
  - **Διπλά συνδεδεμένη λίστα**

Version IV – doubly linked list

Διπλά συνδεδεμένη λίστα



Βασικές λειτουργίες
Εισαγωγή κόμβου
Διαγραφή κόμβου

Εισαγωγή κόμβου
Εισαγωγή ως πρώτου στοιχείου
Εισαγωγή ως ενδιάμεσου στοιχείου
Εισαγωγή ως τελευταίου στοιχείου

Οργάνωση Διάλεξης

- Η εφαρμογή Address Book
- Ταξινόμηση - Αναζήτηση
- Παραδείγματα ταξινόμησης
  - Ταξινόμηση πίνακα ακεραίων
  - Ταξινόμηση πίνακα αλφαριθμητικών
- Procedural and Data Abstraction στην εφαρμογή Address Book
- Εναλλακτικές υλοποιήσεις
  - Πίνακας εγγραφών
  - Πίνακας δεικτών σε εγγραφές
  - Απλά συνδεδεμένη λίστα
  - Διπλά συνδεδεμένη λίστα
- **Low level file handling**

Low level file handling

```
#include <fcntl.h>
#include <sys\types.h>
#include <sys\stat.h>
#include <io.h>
```

int **open** *(const char *filename, int flags[, mode_t mode]);*

int **read** *(int filedes, void *buffer, size_t size);*

int **write** *(int filedes, const void *buffer, size_t size)*

int **close** *(int filedes)*

int **lseek** *(long int)*

TEXT vs. BINARY

```
struct abEntry{
        int am;
        char lname[4];
        } ab[2] = {{14,"abc\n"}, {16,"def\n"}};
```

In TEXT mode \n is saved as 0D 0A

File:

abc
def

In BINARY mode \n is saved as 0A

File:

abc def

See example in DevCpp