Διάλεξη 1: Δομές Δεδομένων και Αλγόριθμοι - Εισαγωγή

Στην ενότητα αυτή θα μελετηθούν τα εξής επιμέρους θέματα:
Εισαγωγή στις έννοιες:

- Δεδομένα και Ενδόμνημη Αναπαράσταση τους
- Οργάνωση Δεδομένων και Δομές Δεδομένων
- Αλγόριθμοι και Πολυπλοκότητα

Διδάσκων: Παναγιώτης Ανδρέου

Συμβόλαιο Μαθήματος

- **Επίπεδο:** Προπτυχιακό
- **Πίστωση:** 7.0 μονάδες ECTS
- **Προαπαιτούμενα:**
  - ΕΠΛ034: Εισαγωγή στον Προγραμματισμό για ΗΜΜΥ
- **Μέθοδοι Διδασκαλίας**
  - Διαλέξεις (3 ώρες εβδομαδιαίως): Παράδοση Διδασ. Ύλης
  - Φροντιστήριο (1 ώρα εβδομαδιαίως): Ύλη / Θεωρητική Εξάσκηση
  - Εργαστήριο (2 ώρες εβδομαδιαίως): Πρακτική Εξάσκηση

Συμβόλαιο Μαθήματος

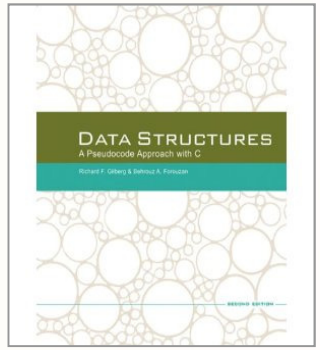
- **Αξιολόγηση**
  - 55% Τελική Εξέταση (1)
  - 20% Ενδιάμεση Εξέταση (1)

    Ημερ.: Δευτέρα, 15 Οκτωβρίου 2012 (7η Εβδ.)
  - 25% Ασκήσεις (5 συνολικά)

    Προγραμματιστικές, Θεωρητικές ή συνδυασμός

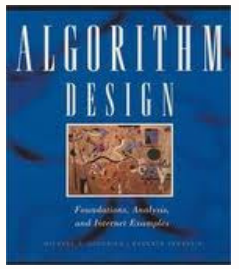
Βιβλιογραφία

Βασική Βιβλιογραφία



- Data Structures – A Pseudocode Approach with C, Richard F. Gilberg & Behrouz A. Fourouzan, 2nd Edition, Thomson, ISBN: 0-534-39080-3, ISBN-13: 9780534390808, 2005.

Βοηθητική Βιβλιογραφία



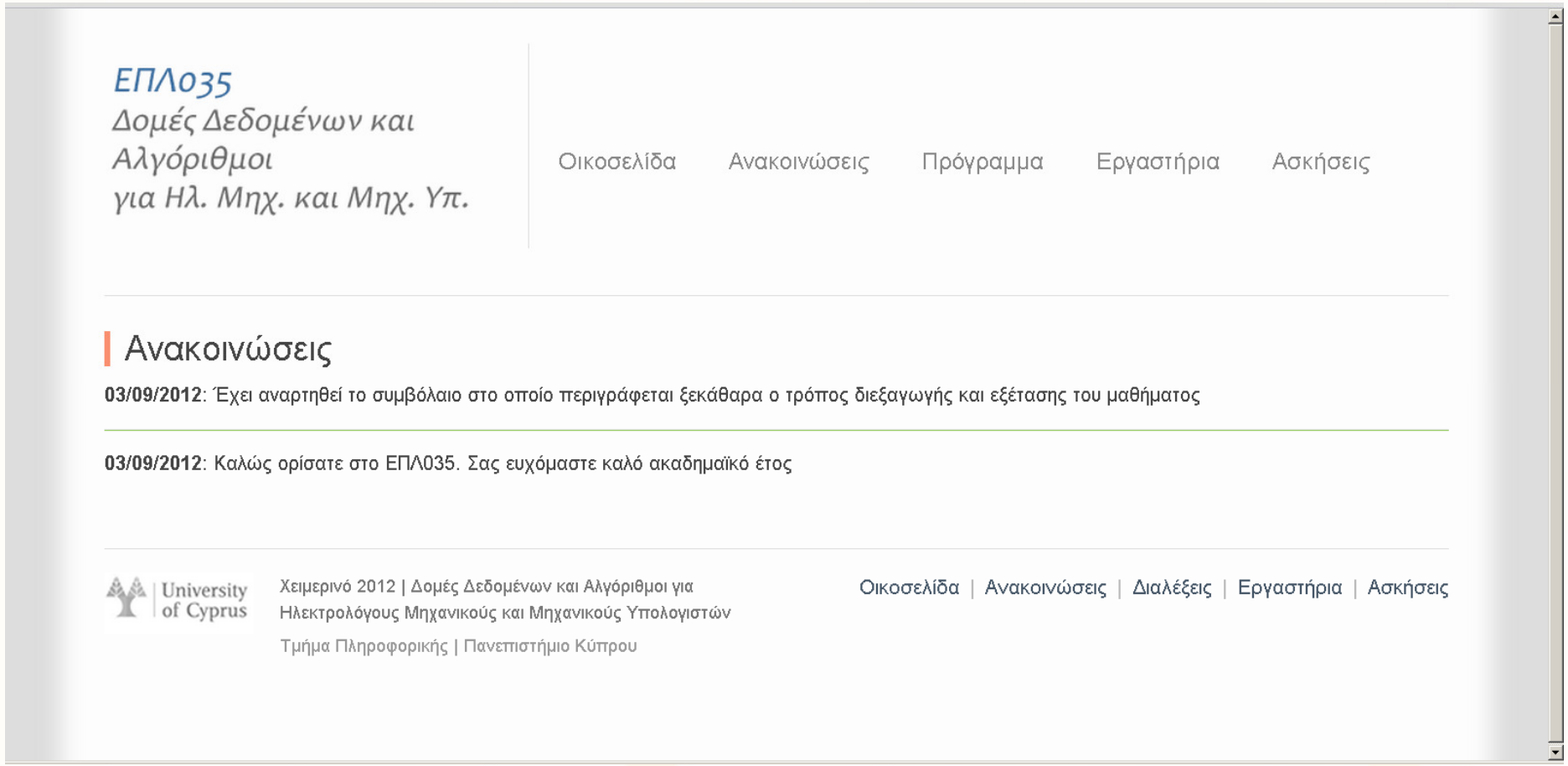
- Σημειώσεις Μαθήματος και Συνοδευτικό Υλικό
- Algorithm Design: Foundations, Analysis, and Internet Examples, Michael T. Goodrich and Roberto Tamassia, ISBN-10: 0471383651, ISBN-13: 978-0471383659, 2001.

- K.N. King, *C Programming: A Modern Approach*, Second Edition, ISBN-10: 0393979504, ISBN-13: 978-0393979503, 832 pages, W. W. Norton & Company, 2008.

- Νικόλαος Μισυρλής, Δομές Δεδομένων με C, ISBN 960-92031-1-6, 2002.

Ιστοσελίδα Μαθήματος

http://www.cs.ucy.ac.cy/courses/EPL035/



Δεδομένα

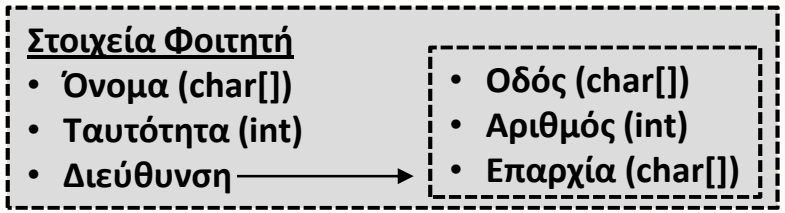
- **Τα δεδομένα βρίσκονται στο χαμηλότερο επίπεδο αφαιρετικότητας**
- **Από τα δεδομένα προκύπτει η πληροφορία και η γνώση.**

- Τα δεδομένα που χρησιμοποιούμε **δεν έχουν πάντοτε τον ίδιο βαθμό πολυπλοκότητας**, π.χ., "Στοιχεία Φοιτητή" vs. "Ηλικία Ατόμου"
- Μερικά μπορούν να αναλυθούν σε απλούστερα συστατικά, ενώ άλλα δεν μπορούν. Δεδομένα που δεν μπορούν να αναλυθούν λέγονται ***πρωτογενή δεδομένα*** **– primitive data types (π.χ. int, char)**
- Τα δεδομένα ενός προβλήματος συνήθως δεν είναι μια άμορφη συλλογή στοιχείων. Τις περισσότερες φορές μπορούν να εκφραστούν με γνωστές μαθηματικές δομές, π.χ.
  - σαν ένα απλό σύνολο (διακριτών στοιχείων): {2, 3, 5, 7, 11, 13}
  - σαν ένα διάνυσμα (διατεταγμένων στοιχείων): (2, 3, 5, 7, 11, 13)
  - σαν ένας πίνακας : $\begin{pmatrix} 2 & 3 & 5 \\ 7 & 11 & 13 \end{pmatrix}$

Δομές Δεδομένων

- **Οι δομές δεδομένων είναι συλλογές πρωτόγονων δεδομένων, που συνδυάζονται για να σχηματίσουν πολυπλοκότερα δεδομένα.**

- Χρησιμοποιώντας τους αρχέγονους τύπους δεδομένων, μπορούμε να συνθέσουμε πιο πολύπλοκες δομές



- Θα χρησιμοποιήσουμε τον όρο ***πίνακας*** για να περιγράψουμε μια συλλογή στοιχείων τα οποία θα ονομάσουμε ***κόμβους*** ή ***καταχωρήματα***.
- Ένας κόμβος μπορεί να είναι ***απλός***, δηλαδή να αποτελείται από ένα μόνο πρωτόγονο δεδομένο, ή να είναι ***σύνθετος***, οπότε αποτελείται από δύο ή περισσότερα ***πεδία***. Τα πεδία ενός κόμβου μπορεί να αντιπροσωπεύουν πρωτόγονα δεδομένα διαφόρων τύπων.

Παράδειγμα Δομής Φοιτητή στην C

```
struct address{
   char street[50];
 int number;
};
struct student{
 char name[30];
 int id;
 struct address saddress;
};
```

Δεδομένα (διεύθυνση και μήκος)

- Ένα πεδίο ή ένα καταχώρημα χαρακτηρίζεται από τη ***διεύθυνση*** του και το ***μήκος*** του. Η διεύθυνση ενός πεδίου ή ενός καταχωρήματος είναι η διεύθυνση της πρώτης του κυψελίδας μνήμης και το μήκος του είναι ο αριθμός κυψελίδων από τις οποίες αποτελείται.

Address ← 4 bytes →*	Value
3669504	1
3669508	2
3669512	3
3669524	A
3669536	5

int a[] μήκος=3

Παραδείγματα στην C

```
int x = 5;
printf("x=%d\tAddress of x=%d\n",
        x, &x);
int c = 'A';
printf("c=%c\tAddress of c=%d\n",
        c, &c);
int a[]={1,2,3};
for(int i=0; i<3; i++){
   printf("a[%d]=%d\tAddress of
   a[%d]=%d\n", i,a[i],i,&a[i]);
}
```

- Η διεύθυνση του πρώτου πεδίου και ολόκληρου του πίνακα είναι **3669504**. Η διεύθυνση του δεύτερου καταχωρήματος **3669508** κ.ο.κ.

* Ισχύει για x86 αρχιτεκτονικές με σύνολο διευθύνσεων $2^{32} \approx 4x10^9$.
Για x64 αρχιτεκτονικές είναι 8 bytes με σύνολο διευθύνσεων $2^{64} = 18x10^{18}$!!!

Ενδόμνημη Παράσταση Δομών Δεδομένων

- Για να αναφερθούμε σε ένα καταχώρημα ή σε ένα πεδίο, χρησιμοποιούμε συνήθως συμβολικά ονόματα (όπως σε γλώσσες προγραμματισμού η ονομασία μεταβλητών αντιστοιχεί σε μία λέξη της μνήμης, π.χ., array).
- Για να πραγματοποιηθεί μια δομή δεδομένων, απαιτείται η **συσχέτιση** των **συμβολικών ονομάτων** με αντίστοιχα τμήματα της μνήμης του υπολογιστή. Αυτό μπορεί να γίνει με δύο τρόπους:

1. Διαδοχική (Στατική) Χορήγηση Μνήμης

Παράδειγμα: Αποθήκευση πίνακα κατά γραμμές ή/και στήλες

Address	Value
3143928	1
3143932	2
3143936	3
3143940	4
3143944	5
3143948	6

Παράδειγμα στην C

```
int b[][3]={{1,2,3},
            {4,5,6}};
for(int i=0; i<2; i++){
  for(int j=0; j<3; j++){
    printf("Address of
            b[%d][%d]=%d\n",
            i,j,&b[i][j]);
   }
}
```

Τυπώνει

```
Address of b[0][0]= 3143928
Address of b[0][1]= 3143932
Address of b[0][2]= 3143936
Address of b[1][0]= 3143940
Address of b[1][1]= 3143944
Address of b[1][2]= 3143944
```

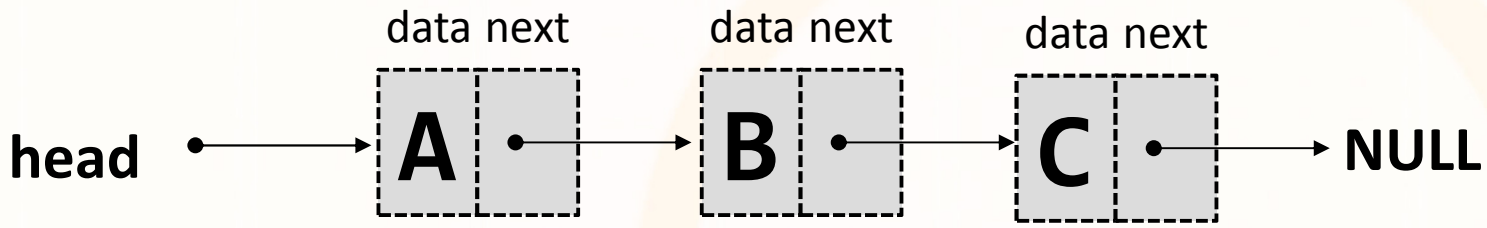
Ενδόμνημη Παράσταση Δομών Δεδομένων

2. Συνδετική Χορήγηση Μνήμης

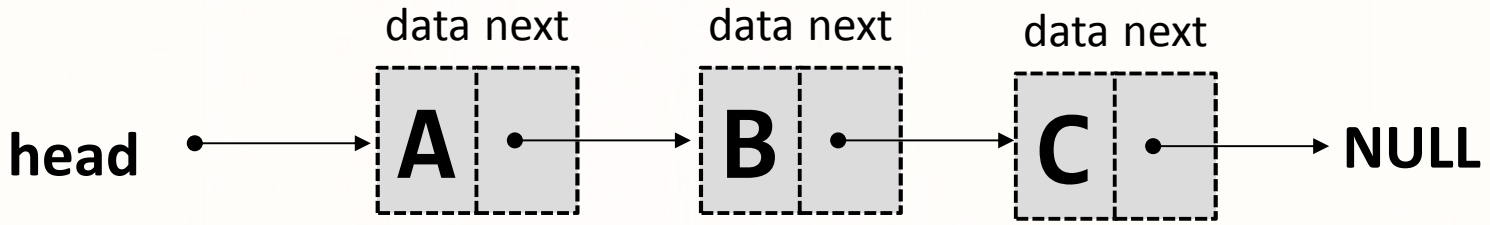
Ένα πεδίο μπορεί να παριστά τη διεύθυνση ενός άλλου πεδίου. Έτσι κατά τη **συνδετική χορήγηση** μνήμης κάθε κόμβος περιέχει πληροφορία σχετικά με το που **βρίσκεται κάποιος άλλος κόμβος της δομής** (π.χ., ο επόμενος, ο προηγούμενος).

Για την αποθήκευση αυτής της πληροφορίας κάθε κόμβος χρειάζεται ένα **ειδικό πεδίο** (**pointer**). Το πεδίο αυτό είναι χαρακτηριστικό της συνδετικής χορήγησης μνήμης.

Παράδειγμα: Απλά Συνδεδεμένη λίστα (linked-list)



Παράδειγμα: Απλά Συνδεδεμένη λίστα



Παράδειγμα στην C

```
struct node{
   char data;
   struct node * next;
};

struct node n1, n2, n3;
n1.data='A';
n2.data='B';
n3.data='C';

n1.next=&n2;
n2.next=&n3;
n3.next=NULL;

printf("n1 address=%d \tdata=%c
\tnext=%d\n",&n1,n1.data, n1.next);
printf("n2 address=%d \tdata=%c
\tnext=%d\n",&n2,n2.data, n2.next);
printf("n3 address=%d \tdata=%c
\tnext=%d\n",&n3,n3.data, n3.next);
```

Node	Address	data	next
n2	3143928	B	3143948
	3143932		
n1	3143936	A	3143928
	3143940		
	3143944		
n3	3143948	C	0

Οι κόμβοι βρίσκονται σε τυχαίες θέσεις μνήμης

Εκτυπώνει

```
n1 address=2751252      data=A   next=2751236
n2 address=2751236      data=B   next=2751220
n2 address=2751220      data=C   next=0
```

Αλγόριθμοι

Τι είναι ένας αλγόριθμος;

Abu Jafar Mohammed ibn Musa Al-Khowarizmi (790-840)

Αλγόριθμος είναι μια πεπερασμένη ακολουθία εντολών, αυστηρά καθορισμένων και εκτελέσιμων σε πεπερασμένο (μετρήσιμο) χρόνο, οι οποίες αν ακολουθηθούν επιτυγχάνεται κάποιο επιθυμητό αποτέλεσμα.

Απαραίτητα κριτήρια:

- Υπάρχει είσοδος και έξοδος
- Καθορισμός εντολών (όχι ασάφειες).
- Περατότητα (να διεκπεραιώνει τον στόχο).

Επίδοση
δεν είναι απαραίτητη αλλά άκρως επιθυμητή

Εξίσωση Wirth: **Αλγόριθμοι + Δεδομένα = Προγράμματα**

Δηλαδή, δοθέντος ενός **προβλήματος**, ένας αλγόριθμος παρέχει τις **οδηγίες** σύμφωνα με τις οποίες τα ***δεδομένα*** του προβλήματος ***μετασχηματίζονται*** και ***συνδυάζονται*** για να προκύψει η **λύση του προβλήματος**.

Παράδειγμα Αλγοριθμικού Προβλήματος

Το πρόβλημα επιλογής (selection problem)

- **Πρόβλημα: Έστω ότι έχουμε n αριθμούς και θέλουμε να προσδιορίσουμε τον k-οστό πιο μεγάλο.**

  π.χ. Έστω οι αριθμοί {5, 72, 3, 4, 1, 9, 65} (72,65,9,5,4,3,1)

  Ο 2^{os} πιο μεγάλος είναι ο 65, ο 6^{os} πιο μεγάλος είναι ο 3, κοκ.

- Αυτό το πρόβλημα είναι γνωστό ως το **πρόβλημα επιλογής (the selection problem).** Υπάρχουν διάφοροι 'εύκολοι' τρόποι λύσης:
  1. **Sort-Based:** 'Διαβάζουμε' τους n αριθμούς σε μια λίστα, a. Ταξινομούμε τη λίστα από το μεγαλύτερο στο μικρότερο με βάση κάποιο αλγόριθμο ταξινόμησης. Επιστρέφουμε το (k-1)οστό στοιχείο της λίστας, δηλαδή το a[k-1].
  2. **Buffer-based:** 'Διαβάζουμε' τους k πρώτους αριθμούς σε μια λίστα a. Ταξινομούμε τη λίστα από το μεγαλύτερο στο μικρότερο δηλ για k=3 έχουμε a={72,5,3}. Μετά, επεξεργαζόμαστε τους υπόλοιπους n-k αριθμούς ως εξής: αν ένα στοιχείο είναι πιο μικρό από το a[k-1] το αγνοούμε, διαφορετικά το τοποθετούμε στη σωστή θέση της λίστας. Όταν η διαδικασία αυτή τελειώσει, επιστρέφουμε το k-οστό στοιχείο της λίστας, δηλαδή το a[k-1].

Στόχος Μαθήματος

- Κατά την εκτέλεση ενός αλγόριθμου **η δομή που έχουν τα δεδομένα παίζει πολύ μεγάλο ρόλο**.
- Επίσης το να γράφουμε **ένα σωστό πρόγραμμα δεν είναι αρκετό**. Ιδιαίτερα, όταν το σύνολο αρχικών δεδομένων είναι μεγάλου μεγέθους, ο ***χρόνος εκτέλεσης*** ενός προγράμματος είναι κύριας σημασίας.
- Στο μάθημα αυτό θα μάθουμε να i) **υπολογίζουμε το χρόνο εκτέλεσης αλγόριθμων** και να ii) **συγκρίνουμε την αποδοτικότητα διαφορετικών αλγόριθμων**, **προτού τους υλοποιήσουμε.** Θα μελετήσουμε επίσης μεθόδους βελτίωσης της ταχύτητας προγραμμάτων.

Κεντρικός στόχος του μαθήματος είναι η μελέτη δομών δεδομένων, αναπαράστασής τους στη μνήμη ενός υπολογιστή, και αλγόριθμων που τις δημιουργούν και τις επεξεργάζονται.

Επίδοση Αλγορίθμων

- Για την αξιολόγηση των διαφόρων αλγορίθμων χρησιμοποιούνται διάφορα μεγέθη.
- n: μέγεθος του προβλήματος
- T(n): Ο χρόνος εκτέλεσης