Εργαστήριο 6

Εαρινό Εξάμηνο 2010-2011

Στόχοι του εργαστηρίου

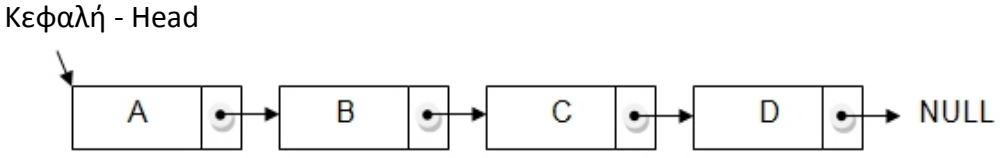
- Δομές δεδομένων
- Δυναμική δέσμευση μνήμης

Δομές Δεδομένων

Εισαγωγή

Μία δομή δεδομένων είναι μια συλλογή δεδομένων με κάποιες ιδιότητες η οποία προσφέρει εύκολη πρόσβαση σε αυτά τα δεδομένα. Έχουμε δει ως τώρα μία στατική δομή δεδομένων, τον πίνακα. Τι κάνουμε όμως στην περίπτωση που θέλουμε να αποθηκεύουμε ένα συνεχώς μεταβαλλόμενο αριθμό από δεδομένα;
Είναι δομές δεδομένων που μπορούν να αυξομειώνουν το μέγεθός τους, δηλαδή να αναπτύσσονται και να συρρικνώνονται στο χρόνο εκτέλεσης. Σε αυτό το εργαστήριο θα δούμε δύο σημαντικές δομές δεδομένων: η διασυνδεδεμένη λίστα και το δέντρο.

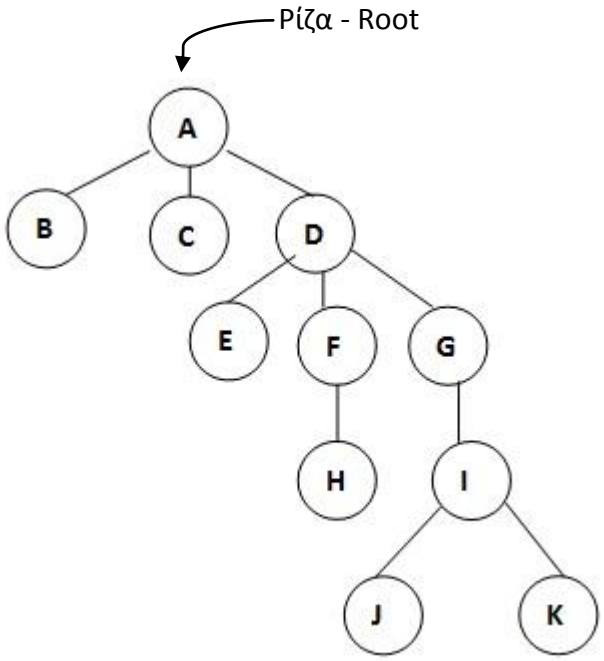
Μία **διασυνδεδεμένη λίστα** είναι μια γραμμική συλλογή δομών, που ονομάζονται κόμβοι, και συνδέονται με δείκτες. Μία συνδεδεμένη λίστα προσπελαύνεται μέσω ενός δείκτη στον πρώτο κόμβο της λίστας (κεφαλή - head). Κατά σύμβαση, ο δείκτης σύνδεσης στον τελευταίο κόμβο της λίστας έχει μηδενική τιμή (NULL) για να σηματοδοτήσει το τέλος της λίστας.



Ένα **δέντρο** είναι μία δομή δεδομένων που συναντάμε συχνά στην καθημερινή ζωή, όπως για παράδειγμα το γενεαλογικό δέντρο, η μορφή των αγώνων σε knockout πρωτάθλημα, … Στις εφαρμογές για Η/Υ, μια από τις γνωστότερες χρήσεις των δομών δέντρου είναι στην οργάνωση του συστήματος αρχείων.
Ένα δέντρο είναι ένα σύνολο κόμβων που συνδέονται από ακμές ή τόξα.
Ορίζεται αναδρομικά ως εξής:

1. Ένα δένδρο είναι είτε κενό είτε
2. αποτελείται από μια ρίζα (root), δηλαδή ένα κόμβο στον οποίο δεν καταλήγουν αλλά μόνο ξεκινούν ακμές, και 0 ή περισσότερα υποδέντρα T_1, T_2,, …, T_k, το καθένα ξεχωριστό από τα άλλα και από τη ρίζα. Υπάρχει μια ακμή από τη ρίζα στις ρίζες των T_1, T_2, …, T_k.

Από κάθε κόμβο ξεκινούν 0 ή περισσότερες ακμές. Σε κάθε κόμβο εκτός της ρίζας καταλήγει μία μόνο ακμή ενώ στην ρίζα δεν καταλήγει καμιά ακμή. Οι κόμβοι περιέχουν δεδομένα και οι ακμές επιβάλλουν μια ιεραρχική δομή στα δεδομένα.
Ο **βαθμός** (degree) ενός κόμβου είναι ο αριθμός των παιδιών του. Ο βαθμός ενός δένδρου είναι ο μέγιστος από τους βαθμούς των κόμβων του.



Διασυνδεδεμένη λίστα

Στην πρώτη άσκηση, θα χρησιμοποιήσουμε μία δομή δεδομένων που θα αποτελεί ένα κόμβο μιας διασυνδεδεμένης λίστας (linked list). Κάθε κόμβος θα αποτελείται από δύο λέξεις (των 32 bits καθεμία): έναν ακέραιο «data» που θα περιέχει την «πληροφορία χρήστη», κι έναν δείκτη σύνδεσης (pointer) «next» που θα περιέχει τη διεύθυνση του επόμενου κόμβου στη λίστα. Στον τελευταίο κόμβο της λίστας, next=0. Τα δύο στοιχεία (λέξεις) του κόμβου μας θα βρίσκονται σε διαδοχικές θέσεις της μνήμης. Επομένως, κάθε κόμβος θα έχει μέγεθος 2 x 4 = 8 bytes. Διεύθυνση ενός κόμβου είναι η διεύθυνση του πρώτου στοιχείου του, δηλαδή του στοιχείου με «μηδενικό offset», που για μας είναι το «data».

Δυαδικό δέντρο

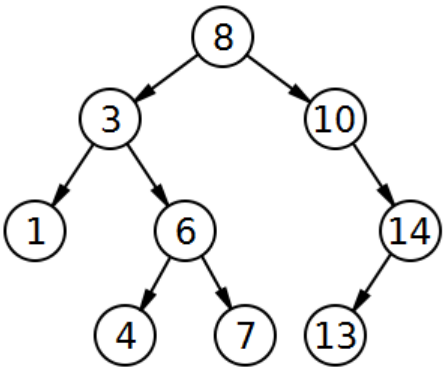
Τα δυαδικά δέντρα έχουν βαθμό 2, δηλαδή κάθε κόμβος μπορεί να έχει μέχρι δύο κόμβους παιδιά. Επομένως κάθε κόμβος μπορεί να έχει δύο υποδέντρα, το αριστερό υποδέντρο και το δεξί υποδέντρο. Κάθε κόμβος θα αποτελείται από τρεις λέξεις (των 32 bits καθεμία): έναν ακέραιο «data» που θα περιέχει την «πληροφορία χρήστη», κι δύο δείκτες σύνδεσης (pointer) «left» και «right» που θα περιέχουν τη διεύθυνση του αριστερού και του δεξιού υποδέντρου αντίστοιχα. Αν οι δείκτες left και right είναι μηδενικά, δηλαδή αν ένας κόμβος δεν έχει παιδιά, τότε ο κόμβος ονομάζεται φύλλο (leaf).
Επομένως, κάθε κόμβος θα έχει μέγεθος 3 x 4 = 12 bytes.

Δυαδικό δέντρο αναζήτησης

Τα δυαδικά δέντρα αναζήτησης (binary search trees) είναι δυαδικά δέντρα με τις παρακάτω ιδιότητες για κάθε κόμβο:

- όλοι οι κόμβοι του αριστερού παιδιού έχουν τιμές μικρότερες ή ίσες της τιμής του κόμβου
- όλοι οι κόμβοι του δεξιού παιδιού έχουν τιμές μεγαλύτερες ή ίσες της τιμής του κόμβου

Παράδειγμα:



Δυναμική Εκχώρηση Μνήμης (Dynamic Memory Allocation)

Το πρόγραμμά σας θα ζητάει και θα παίρνει κόμβους από το λειτουργικό σύστημα «δυναμικά», την ώρα που τρέχει (σε run-time). Για το σκοπό αυτό θα χρησιμοποιήστε την κλήση συστήματος (system call) «sbrk» (set break). Η κλήση αυτή «σπρώχνει» πιο πέρα (προς αύξουσες διευθύνσεις μνήμης) το σημείο «break», το όριο πριν από το οποίο οι διευθύνσεις μνήμης που γεννά το πρόγραμμα είναι νόμιμες, ενώ μετά από το οποίο (και μέχρι την αρχή της στοίβας) οι διευθύνσεις είναι παράνομες. Η κλήση συστήματος «sbrk» παίρνει σαν παράμετρο το πλήθος των νέων bytes που επιθυμείτε στον καταχωρητή $a0. Μετά την επιστροφή της κλήσης, ο καταχωρητής $v0 περιέχει την διεύθυνση του νέου block μνήμης, του

ζητηθέντος μεγέθους, που το σύστημα δίνει στο πρόγραμμά σας (έναν pointer). Σε περίπτωση που η διεύθυνση είναι μηδενική τότε έχει γεμίσει όλη η μνήμη και σε τέτοια περίπτωση το πρόγραμμά σας θα πρέπει να τερματίσει.

Για το επόμενο εργαστήριο (05-06/04/2011) έχετε να υλοποιήσετε τις 2 παρακάτω εργαστηριακές ασκήσεις. Την ώρα του εργαστηρίου θα εξετασθείτε προφορικά πάνω στους κώδικες που θα παραδώσετε.

Για την περιγραφή των παρακάτω ασκήσεων θα χρησιμοποιήσουμε τις εξής αναπαραστάσεις κόμβων:

Κόμβος λίστας:

```
typedef struct node {
      int data;
      struct node *next;
} nodeL;
```

Κόμβος δέντρου:

```
typedef struct btree {
      int data;
      struct btree *left, *right;
} nodeT;
```

Άσκηση 1 (7 μονάδες (5 + 2)) – Διασυνδεδεμένη λίστα

α) Να γραφεί μία συνάρτηση **nodeL* insertElement(nodeL *head, int data)** όπου head είναι δείκτη στην αρχή της διασυνδεδεμένης λίστας και data είναι ο ακέραιος που θέλουμε να εισάγουμε στην λίστα. Η λίστα πρέπει να είναι **ταξινομημένη κατά αύξουσα σειρά** με βάση την τιμή του data. Η συνάρτηση επιστρέφει την κεφαλή της λίστας.
Η main θα διαβάζει επαναληπτικά από το πληκτρολόγιο θετικούς αριθμούς (> 0) και θα τα εισάγει στην λίστα καλώντας την συνάρτηση insertElement. Μόλις δοθεί το 0, τότε η main σταματάει να ζητάει στοιχεία και εκτυπώνει την λίστα (ερώτημα β).

<u>Προσοχή:</u>

- Η συνάρτηση πρέπει να ελέγχει την τιμή που επιστρέφει η κλήση συστήματος για την δυναμική δέσμευση μνήμης. Σε περίπτωση που αυτή είναι μηδέν, τότε εκτυπώνει κατάλληλο μήνυμα λάθους και το πρόγραμμα τερματίζει.
- Να προσέχετε τις ειδικές περιπτώσεις όπως την εισαγωγή σε κενή λίστα.

β) Να γραφεί μία συνάρτηση **void printList (nodeL *head)** όπου head είναι δείκτη στην αρχή της διασυνδεδεμένης λίστας, η οποία θα εκτυπώνει το περιεχόμενο της λίστας στην οθόνη ως εξής:

```
List: (1 2 2 3 6 12 15)
```

Η main θα καλέσει την συνάρτηση printList μόλις τελειώσει η επανάληψη που διαβάζει και εισάγει τα στοιχεία στην ταξινομημένη λίστα.

Άσκηση 2 (3 μονάδες (2 + 1)) – Δυαδικό δέντρο αναζήτησης

α) Να γραφεί μία συνάρτηση **nodeT* insertElement(nodeT *root, int data)** όπου root είναι δείκτη στην ρίζα του δυαδικού δέντρου αναζήτησης και data είναι ο ακέραιος που θέλουμε να εισάγουμε στο **δυαδικό δέντρο αναζήτησης**. Η συνάρτηση επιστρέφει την ρίζα του δυαδικού δέντρου αναζήτησης.

Η main θα διαβάζει επαναληπτικά από το πληκτρολόγιο θετικούς αριθμούς (> 0) και θα τα εισάγει στο δυαδικό δέντρο αναζήτησης καλώντας την συνάρτηση insertElement. Μόλις δοθεί το 0, τότε η main σταματάει να ζητάει στοιχεία και εκτυπώνει το δυαδικό δέντρο αναζήτησης(ερώτημα β).

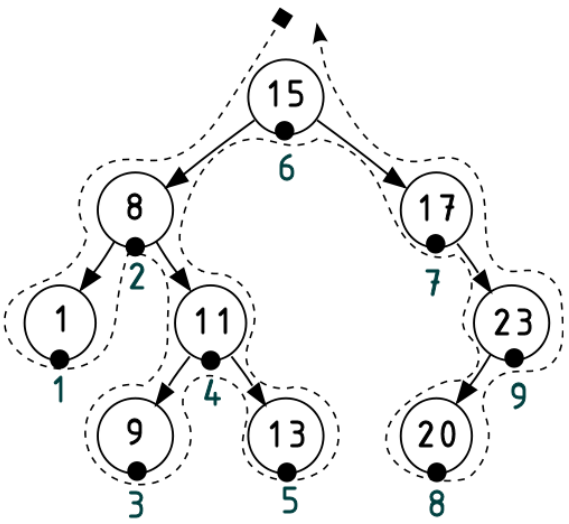
Προσοχή:

- Η συνάρτηση πρέπει να ελέγχει την τιμή που επιστρέφει η κλήση συστήματος για την δυναμική δέσμευση μνήμης. Σε περίπτωση που αυτή είναι μηδέν, τότε εκτυπώνει κατάλληλο μήνυμα λάθους και το πρόγραμμα τερματίζει.
- Να προσέχετε τις ειδικές περιπτώσεις όπως την εισαγωγή σε κενό δέντρο.

β) Να γραφεί μία **αναδρομική** συνάρτηση **void printTreeInOrder(nodeT *root)** όπου root είναι δείκτη στην ρίζα του δυαδικού δέντρου αναζήτησης. Η συνάρτηση αυτή εκτυπώνει το περιεχόμενο του δέντρου σε αύξουσα διάταξη με βάση τις ιδιότητες του δυαδικού δέντρου αναζήτησης και της in-order διαπέραση δέντρου.

Η **in-order διαπέραση** δέντρου εκτελεί τα παρακάτω αναδρομικά βήματα σε κάθε κόμβο:

1. Διαπέραση του αριστερού υποδέντρου
2. Επίσκεψη ρίζας
3. Διαπέραση του δεξιού υποδέντρου



Η συνάρτηση θα εκτυπώνει το περιεχόμενο της λίστας στην οθόνη ως εξής:

```
Tree: (1 8 9 11 13 15 17 20 23)
```

Η main θα καλέσει την συνάρτηση printTreeInOrder μόλις τελειώσει η επανάληψη που διαβάζει και εισάγει τα στοιχεία στο δυαδικό δέντρο αναζήτησης.

Θα πρέπει να στέλνετε με email τις λύσεις των εργαστηριακών ασκήσεων σας στους διδάσκοντες στο ce134lab@gmail.com.

Το email σας θα πρέπει να περιέχει ως attachment **ένα zip file** με τον κώδικα σας.

Κάθε διαφορετική άσκηση στην εκφώνηση θα βρίσκεται και σε διαφορετικό asm file. **Το όνομα των asm files θα ΠΡΕΠΕΙ να αρχίζει με το ΑΕΜ σας.**
Για παράδειγμα, το lab2.zip θα περιέχει 3 asm files, ένα για κάθε μία από τις ασκήσεις του lab2, με ονόματα 999_lab2a.asm, 999_lab2b.asm, 999_lab2c.asm για τον φοιτητή με ΑΕΜ 999.

Το email σας θα έχει Subject: CE134, lab N, Section X (N ο αριθμός του lab, N=2 ..., και X=1 έως 7).
Το email σας θα έχει body: το όνομα σας και το ΑΕΜ σας.

Θα πρέπει να στέλνετε το email σας πριν βγείτε από την εξέταση του εργαστηρίου.