Δομές Δεδομένων και Τεχνικές Προγραμματισμού (Τμήμα Αρτίων ΑΜ)

Εργαστήριο 3: Δυαδικά Δέντρα Αναζήτησης

Στο εργαστήριο αυτό θα χρησιμοποιήσουμε τον κώδικα που παρουσιάστηκε στην ένατη ενότητα διαλέξεων (Δυαδικά Δέντρα Αναζήτησης) και βρίσκεται στην ιστοσελίδα του μαθήματος: http://cgi.di.uoa.gr/~k08/code.html . Ο κώδικας υλοποιεί διάφορες λειτουργίες των δυαδικών δέντρων αναζήτησης που συζητήσαμε στην τάξη. Μελετήστε τον κώδικα προσεκτικά και κατανοήστε τη λειτουργία του. Μεταγλωττίστε τον και εκτελέστε τον. Μετά κάνετε τις παρακάτω ασκήσεις.

1. Στην άσκηση αυτή θα θυμηθούμε την έννοια του **δείκτη σε μία συνάρτηση** από το πρώτο μάθημα προγραμματισμού. Στη C, όπως έχουμε δείκτες προς δεδομένα διαφόρων τύπων, έτσι μπορούμε να έχουμε και δείκτες σε συναρτήσεις. Μια που ο μεταγλωττιστής της C δεσμεύει μνήμη για να αποθηκεύσει τον κώδικα μιας συνάρτησης, το να έχουμε ένα δείκτη προς αυτές τις θέσεις μνήμης είναι λογικό. Η γενική μορφή της δήλωσης ενός δείκτη σε μια συνάρτηση έχει την εξής μορφή:

   ```
   return_type (*pointer_name)(parameter_type1
     parameter_name1, …, parameter_typeN parameter_nameN).
   ```

   Για να είναι σωστή η δήλωση ενός δείκτη προς μία συνάρτηση, θα πρέπει να ταιριάζει με τον τύπο επιστροφής της συνάρτησης και τη λίστα των τυπικών παραμέτρων της. Παρακάτω δίνεται ένα μικρό πρόγραμμα που παρουσιάζει την χρήση δείκτη σε συνάρτηση. Μελετήστε το πρόγραμμα προσεκτικά. Μεταγλωττίστε το και εκτελέστε το. Αφού κατανοήσετε τη έννοια του δείκτη σε συνάρτηση, μπορείτε να συνεχίσετε με την άσκηση 2.

```
/* Το παράδειγμα αυτό προέρχεται από το βιβλίο «C: Από τη
θεωρία στην εφαρμογή» των Γ. Σ. Τσελίκη και Ν.Δ. Τσελίκα, 2η
έκδοση, σελ. 252. */

#include <stdio.h>

void test(int a);

int main()
{
```

```
   void (*ptr)(int a);
   /* Η μεταβλητή ptr δηλώνεται ως δείκτης σε μία συνάρτηση, η οποία δέχεται μια ακέραια παράμετρο και δεν επιστρέφει τίποτα. */

   int i=10;

   ptr=test;
   /* Τώρα ο δείκτης ptr δείχνει στη διεύθυνση της συνάρτησης test. Όπως το όνομα ενός πίνακα μπορεί να χρησιμοποιηθεί σαν δείκτης, έτσι και το όνομα μιας συνάρτησης μπορεί να χρησιμοποιηθεί σαν δείκτης στη διεύθυνσή της.*/

   ptr(i);
   /* Η κλήση αυτή είναι ισοδύναμη με την (*ptr)(i); που τονίζει περισσότερο το γεγονός ότι ο ptr είναι δείκτης προς μία συνάρτηση. */

   return 0;
}

void test(int a)
{
   printf("Val = %d\n", 2*a);
}
```

2. Θεωρείστε τις συναρτήσεις διάσχισης δυαδικού δέντρου `Preorder`, `Inorder` και `Postorder` (αρχείο `BinarySearchTreeImplementation.c`). Τροποποιήστε τις συναρτήσεις αυτές ώστε να παίρνουν μια δεύτερη τυπική παράμετρο `void (*Visit)(TreeEntry x)` που είναι ένας δείκτης σε μια συνάρτηση η οποία παίρνει μια τυπική παράμετρο τύπου `TreeEntry`. Η συνάρτηση αυτή πρέπει να καλείται αντί της `printf` στο σώμα των συναρτήσεων διάσχισης. Παρατηρήστε ότι τώρα οι συναρτήσεις διάσχισης είναι πιο γενικές επειδή μπορούν να καλούνται με διαφορετικές τιμές της παραμέτρου `Visit`, και να εκτελούν κάθε φορά κάτι διαφορετικό σε κάθε κόμβο του δέντρου που επισκέπτονται. Γράψτε μια συνάρτηση `void PrintElement(TreeEntry x)` και μια συνάρτηση `void PrintElementPlusOne(TreeEntry x)` οι οποίες να μπορούν να χρησιμοποιηθούν σαν δεύτερο όρισμα στις κλήσεις των συναρτήσεων διάσχισης (αντιστοιχούν δηλαδή στην παράμετρο `Visit`). Η `PrintElement` τυπώνει την τιμή

της παραμέτρου `x` ενώ η `PrintElementPlusOne` τυπώνει την τιμή της έκφρασης `x+1`. Καλέστε τις νέες συναρτήσεις διάσχισης από το κυρίως πρόγραμμα (συνάρτηση `main`) ώστε να τεστάρετε τη λειτουργία τους.

3. Θεωρείστε την αναδρομική συνάρτηση `TreeSearch` (αρχείο `BinarySearchTreeImplementation.c`). Γράψτε μια ισοδύναμη συνάρτηση που να χρησιμοποιεί επανάληψη αντί για αναδρομή. Τεστάρετε την λειτουργία της με κατάλληλες κλήσεις από το κυρίως πρόγραμμα. Η πράξη που μόλις κάνατε λέγεται tail recursion elimination και μπορεί να γίνει πάντα όταν η αναδρομική κλήση είναι η τελευταία εντολή που εκτελείται σε μια αναδρομική διαδικασία.

4. Γράψτε δύο συναρτήσεις `TreeMinimum` και `TreeMaximum` οι οποίες θα παίρνουν σαν όρισμα ένα δείκτη σε ένα δυαδικό δέντρο αναζήτησης και θα επιστρέφουν ένα δείκτη στον κόμβο με το μικρότερο και μεγαλύτερο κλειδί αντίστοιχα. Τεστάρετε την λειτουργία των συναρτήσεων αυτών με κατάλληλες κλήσεις από το κυρίως πρόγραμμα.

5. Θεωρήστε τη συνάρτηση `InsertTree` η οποία εισάγει ένα νέο κόμβο σε ένα δυαδικό δέντρο αναζήτησης. Ο κόμβος αυτός έχει δημιουργηθεί στη συνάρτηση `main` και η `InsertTree` έχει σαν δεύτερο όρισμα ένα δείκτη στον νέο κόμβο. Τροποποιήστε τη συνάρτηση `InsertTree` ώστε το δεύτερο της όρισμα να είναι ένα κλειδί. Η νέα συνάρτηση θα δημιουργεί ένα νέο κόμβο, θα θέτει σαν τιμή του πεδίου `entry` την τιμή του κλειδιού και θα εισάγει το νέο κόμβο στο δέντρο.