ΕΡΓΑΣΤΗΡΙΟ 10:
Δομές και Αυτοαναφορικές Δομές

Στο εργαστήριο αυτό θα μελετήσουμε τη δυνατότητα που μας δίνει η C για να ομαδοποιούμε δεδομένα ορίζοντας δομές. Μέσω των δομών θα ορίσουμε συνδεδεμένες λίστες και δυαδικά δένδρα, που είναι ειδικές δομές οι οποίες ονομάζονται αυτοαναφορικές.

Άσκηση 1: Δομές και συναρτήσεις

1.1 Κατασκευάστε το αρχείο `point.c` και ορίστε σε αυτό τη δομή `point` που αποθηκεύει τις συντεταγμένες (τύπου `double`) ενός σημείου στο διδιάστατο χώρο.

1.2 Ορίστε τη συνάρτηση `struct point middle(struct point a, struct point b)` που υπολογίζει και επιστρέφει το σημείο που είναι το μέσο του ευθυγράμμου τμήματος με άκρα τα σημεία `a` και `b`.

1.3 Υπολογίστε το μέσο του ευθυγράμμου τμήματος με άκρα τα σημεία `(1.2,5.4)` και `(7.3,1.8)`.

Άσκηση 2: Δομές και δείκτες

2.1 Κατασκευάστε το αρχείο `person.c` και ορίστε σε αυτό τη δομή `person` που αποθηκεύει το όνομα, το επώνυμο και το πατρώνυμο ενός ατόμου ως εξής:

```
struct person {
   char *fname;
   char *lname;
   char *mname;
};
```

2.2 Κατασκευάστε τη συνάρτηση `struct person *person_init(char *firstname, char *lastname, char *middlename)` η οποία δέχεται σαν όρισμα 3 συμβολοσειρές, δεσμεύει χώρο για μία δομή τύπου `person`, την αρχικοποιεί κατάλληλα και επιστρέφει τη διεύθυνση της δομής αυτής στο όνομά της. Έπειτα καλέστε την από τη συνάρτηση `main()` με ορίσματα που απεικονίζουν τον πατέρα σας.

2.3 Ορίστε τη συνάρτηση `struct person *childof(struct person father, char *newname)`, έτσι ώστε να αρχικοποιεί τα στοιχεία ενός παιδιού του `father`, που έχει μικρό όνομα το `newname`, και να επιστρέφει στο όνομά της τη διεύθυνση μίας δομής `person` για το παιδί. Έπειτα, καλέστε την από τη συνάρτηση `main()` για να κατασκευάσετε τον εαυτό σας.

Άσκηση 3: Συνδεδεμένες λίστες

3.1 Κατασκευάστε το πρόγραμμα `grades.c` στο οποίο να ορίζεται μία αυτοαναφορική δομή λίστας ακεραίων αριθμών.

Ορισμός Συνδεδεμένης Λίστας

```
typedef struct listnode *Listptr;

struct listnode {
  ΤΔ data;
  Listptr next;
};
```

3.2 Κατασκευάστε τη συνάρτηση `void insert_at_start(Listptr *ptr, int grade)` που να προσθέτει έναν βαθμό στην αρχή της λίστας. Τροποποιήστε την `main()` του προγράμματος σας, ώστε να διαβάζει βαθμούς από το πληκτρολόγιο, μέχρι το τέλος της εισόδου, και να τους προσθέτει στην αρχή της λίστας.

3.3 Κατασκευάστε τη συνάρτηση `float average(Listptr ptr)` που δέχεται σαν όρισμα μία συνδεδεμένη λίστα, διασχίζει τα περιεχόμενα της και υπολογίζει τον μέσο όρο των βαθμών που έχουν αποθηκευθεί σε αυτήν.

Άσκηση 4: Δυαδικά δένδρα

4.1 Κατασκευάστε το πρόγραμμα `tree.c` στο οποίο να ορίζεται μία αυτοαναφορική δομή δυαδικού δένδρου ακεραίων αριθμών.

```
typedef struct tnode *Treeptr;

struct tnode {
  ΤΔ data;
  Treeptr left;
  Treeptr right;
};
```

4.2 Ένα ταξινομημένο δυαδικό δένδρο είναι ένα δυαδικό δένδρο στο οποίο κάθε κόμβος έχει στο αριστερό του υποδένδρο αριθμούς μικρότερους από τον ίδιο και στο δεξί του υποδένδρο αριθμούς μεγαλύτερους από τον ίδιο, και η ιδιότητα αυτή ισχύει τόσο για το αριστερό όσο και για το δεξί υποδένδρο.

Ορίστε την αναδρομική συνάρτηση `Treeptr addtree(Treeptr p, int x)` η οποία να προσθέτει έναν αριθμό `x` στο ταξινομημένο δυαδικό δένδρο `p`, διατηρώντας το ταξινομημένο, και να επιστρέφει στο όνομά της το νέο δένδρο. Δηλαδή:

- Αν ο αριθμός που περιέχει ο κόμβος στον οποίο δείχνει ο `p` είναι μεγαλύτερος του `x`, γίνεται κλήση της `addtree` για το αριστερό παιδί του `p`.
- Αν ο αριθμός που περιέχει ο κόμβος στον οποίο δείχνει ο `p` είναι μικρότερος του `x`, γίνεται κλήση της `addtree` για το δεξί παιδί του `p`.
- Αν ο αριθμός που περιέχει ο κόμβος στον οποίο δείχνει ο `p` είναι ίσος με το `x`, τότε δεν γίνεται καμία εισαγωγή στο δέντρο.
- Αν ο `p` είναι `NULL`, κατασκευάζεται ένας νέος κόμβος δένδρου, που περιέχει τον `x` και ο `p` τίθεται να δείχνει σε αυτόν τον νέο κόμβο.

Τροποποιήστε την `main()` του προγράμματος σας, ώστε να διαβάζει αριθμούς από το πληκτρολόγιο, μέχρι το τέλος της εισόδου, και να τους προσθέτει στο δένδρο.

4.3 Κατασκευάστε την αναδρομική συνάρτηση `void treeprint(Treeptr p)` που δέχεται σαν όρισμα ένα δυαδικό δένδρο και διασχίζει τα περιεχόμενα του σύμφωνα με την εξής λογική:

1. Αν το δέντρο είναι `NULL`, η συνάρτηση επιστρέφει.
2. Αν το δέντρο δεν είναι `NULL`:
   i. Καλείται η `treeprint` για να εκτυπωθεί το αριστερό παιδί του `p`.
   ii. Εκτυπώνεται η ρίζα του δέντρου.
   iii. Καλείται η `treeprint` για να εκτυπωθεί το δεξί παιδί του `p`.

Καλέστε την `treeprint`, από την `main()`, για το δυαδικό δένδρο που έχετε κατασκευάσει και παρατηρήστε την εκτύπωσή του.