



Θ.Ε. ΠΛΣ50 (2007-2008) – ΓΡΑΠΤΗ ΕΡΓΑΣΙΑ Ε3

Στόχος

Στην εργασία αυτή θα ασχοληθούμε με δυναμική δέσμευση μνήμης και σύνθετες δομές δεδομένων (λίστες).

Εισαγωγικά:

Σκοπός της εργασίας αυτής είναι να επαυξήσουμε τις δυνατότητες του προγράμματος Sudoku που κατασκευάσαμε στην εργασία Ε2, έτσι ώστε αυτό να επιτρέπει τον χειρισμό Sudoku μεταβλητών διαστάσεων (δηλαδή το ίδιο πρόγραμμα, χωρίς να χρειάζεται να επεμβούμε στον κώδικά του και να τροποποιήσουμε την τιμή της σταθεράς size, να μπορεί να χειρισθεί Sudoku διαστάσεων 4x4, 9x9, 16x16 κλπ). Ειδικότερα, επιθυμούμε μέσα από το κεντρικό μενού του προγράμματος ο χρήστης να μπορεί να διαβάζει, εμφανίζει, συμπληρώνει και αποθηκεύει Sudoku διαφόρων διαστάσεων. Για το σκοπό αυτό θα τροποποιήσουμε τόσο την αναπαράσταση των Sudoku, όσο και τις συναρτήσεις που υλοποιήσαμε για τον χειρισμό αυτών.

Ειδικότερα, όσον αφορά την αναπαράσταση, στην ενδεικτική λύση της εργασίας Ε2 χρησιμοποιήθηκε ο τύπος `sudoku_type`, ο οποίος ουσιαστικά αποτελεί έναν διδιάστατο πίνακα `short` μεγέθους `size x size`. Η αναπαράσταση αυτή δεν είναι επαρκής εάν θέλουμε να περιγράψουμε Sudoku διαφόρων διαστάσεων. Για το σκοπό αυτό προτείνεται να υιοθετηθεί η παρακάτω τροποποιημένη δομή:

```
struct sudoku_type
{
    int size;
    int* values;
};
```

Στην παραπάνω δομή κεντρικό ρόλο παίζει το πεδίο `size`, στο οποίο καταχωρείται η διάσταση του τρέχοντος Sudoku. Οι τιμές του τρέχοντος Sudoku, συμπληρωμένες ή και μη, διατηρούνται στον πίνακα ακεραίων που βρίσκεται στη διεύθυνση `values`. Ο κώδικας που ακολουθεί δημιουργεί ένα νέο sudoku με διάσταση 9 και αρχικοποιεί (μηδενίζει) τις τιμές όλων κελιών:

```
int i;
struct sudoku_type *s=(struct sudoku_type*) malloc(sizeof(struct sudoku_type));
s->size=9;
s->values=(int*) malloc(s->size*s->size*sizeof(int));

for(i=0;i<s->size*s->size;i++)
    s->values[i]=0;
```

Προσέξτε ότι το πεδίο `values` της δομής `sudoku_type` είναι δείκτης σε έναν πίνακα ακεραίων, για τον οποίο η μνήμη δεσμεύεται δυναμικά ανάλογα με την τιμή του πεδίου `size` του Sudoku. Έτσι χρειάζεται να γίνουν δύο διαφορετικές δεσμεύσεις μνήμης, μια για τη δομή `sudoku_type` (η οποία περιλαμβάνει μόνο έναν ακέραιο και έναν δείκτη σε ακέραιο) και μία για τον πίνακα των ακεραίων.

Προσέξτε επίσης ότι ο πίνακας ακεραίων `values` δεν δηλώνεται πουθενά ως διδιάστατος πίνακας (όπως θα θέλαμε να είναι), με αποτέλεσμα ουσιαστικά να συμπεριφέρεται ως μονοδιάστατος. Με άλλα λόγια, αυτό σημαίνει ότι για να έχουμε πρόσβαση στα στοιχεία του πρέπει να υπολογίσουμε τη "διεύθυνσή" τους μέσα σε αυτόν. Αυτό γίνεται ως εξής: Αν θεωρήσουμε ότι τα κελιά του Sudoku τοποθετούνται στον πίνακα κατά σειρές, τότε το στοιχείο `i,j`, δηλαδή το στοιχείο που βρίσκεται στην `i` σειρά και την `j` στήλη του Sudoku βρίσκεται στη θέση `(i-1)*size+(j-1)` του πίνακα `values`, όπου `size` η διάσταση του Sudoku. Ο παρακάτω κώδικας ζητά από τον χρήστη να δώσει πρώτα τη γραμμή και τη



στήλη ενός στοιχείου και στη συνέχεια την τιμή αυτού και το καταχωρεί στην σωστή θέση του πίνακα values.

```
int R,C,V;

printf("Δώσε τη γραμμή: ");
scanf("%d",&R);
printf("Δώσε τη στήλη: ");
scanf("%d",&C);
printf("Δώσε την τιμή: ");
scanf("%d",&V);

s->values[(R-1)*s->size+(C-1)]=V;
```

Σημειώστε ότι αφαιρείται 1 από τους αριθμούς γραμμής και στήλης που δίνει ο χρήστης, μιας και για τον χρήστη η πρώτη γραμμή και η πρώτη στήλη του Sudoku έχουν αύξοντα αριθμό 1, ενώ για τη γλώσσα C η πρώτη θέση ενός πίνακα είναι η θέση 0.

Θέμα 1: Ανάγνωση / Αποθήκευση Sudoku

Κατά την αποθήκευση ενός Sudoku σε αρχείο θα πρέπει να δηλώνεται η διάσταση του Sudoku. Κάτι τέτοιο μπορεί να επιτευχθεί εάν στην αρχή κάθε αρχείου υπάρχει ένας ακέραιος αριθμός που δηλώνει τη διάσταση. Επιπλέον, με δεδομένο ότι πλέον η διάσταση ενός Sudoku μπορεί να είναι αυθαίρετα μεγάλη (π.χ. 25 ή 36 ή 49 κλπ), η αναπαράσταση με συνδυασμό αριθμών και χαρακτήρων που προτάθηκε στην εργασία Ε2 δεν λειτουργεί. Αντί αυτής προτείνεται η παρακάτω αναπαράσταση για ένα Sudoku διάστασης 9, η οποία όμως γενικεύεται εύκολα σε μεγαλύτερες διαστάσεις:

9

-2	3	-8	0	-5	0	-1	0	-7
4	7	-6	0	0	-2	0	0	5
-1	-9	5	-7	0	3	0	0	-8
0	6	-2	0	0	-8	0	-5	1
-5	8	0	0	-6	1	0	0	-3
3	-4	1	-9	2	5	-7	8	6
-6	0	0	0	0	-4	8	-7	-2
0	0	0	-1	0	0	-5	3	4
-8	0	-4	0	-3	7	-6	1	-9

Στην παραπάνω αναπαράσταση, ο αρχικός αριθμός 9 στην πρώτη σειρά δηλώνει τη διάσταση του Sudoku, οι αρνητικοί αριθμοί δηλώνουν αριθμούς που υπάρχουν αρχικά στο Sudoku, οι θετικοί αριθμοί δηλώνουν αριθμούς που συμπλήρωσε ο χρήστης και τα μηδενικά δηλώνουν αριθμούς που δεν έχουν ακόμη συμπληρωθεί. Σημειώνεται ότι ανάλογη σύμβαση για τους αριθμούς που υπάρχουν εξ αρχής στο Sudoku και αυτούς που συμπληρώνει ο χρήστης καλό είναι να υιοθετηθεί και για την αναπαράσταση του Sudoku στη μνήμη, δηλαδή στον πίνακα values της δομής sudoku_type. Επίσης, στο αρχείο αποθήκευσης οι αριθμοί διαχωρίζονται με στηλοθέτες (χαρακτήρας '\t'), ενώ για λόγους ομοιομορφίας υπάρχει στηλοθέτης και μετά τον τελευταίο αριθμό κάθε σειράς.



Το παραπάνω Sudoku λυμένο θα αποθηκευόταν ως εξής:

9								
-2	3	-8	6	-5	9	-1	4	-7
4	7	-6	8	1	-2	3	9	5
-1	-9	5	-7	4	3	2	6	-8
9	6	-2	3	7	-8	4	-5	1
-5	8	7	4	-6	1	9	2	-3
3	-4	1	-9	2	5	-7	8	6
-6	1	3	5	9	-4	8	-7	-2
7	2	9	-1	8	6	-5	3	4
-8	5	-4	2	-3	7	-6	1	-9

Σας δίνεται λοιπόν έτοιμος ο κώδικας μιας συνάρτησης που διαβάζει ένα ημιτελές ή ένα λυμένο sudoku από αρχείο.

```
/* read a sudoku from a file */
/* and store it at sudoku data structure, where 0 = empty cell, */
/* negative = given number, positive = player's number */
int read_puzzle(struct sudoku_type *sudoku, FILE *in) {
    int i,j;
    int error;
    error=fscanf(in, "%d", &sudoku->size);
    if (error!=1)
        return 0;
    sudoku->values=(int*) malloc(sudoku->size*sudoku->size*sizeof(int));
    if (sudoku->values==NULL)
        return 0;
    for (i=0;i<sudoku->size;i++)
        for (j=0;j<sudoku->size;j++)
        {
            error=fscanf(in, "%d/t", &sudoku->values[(i-1)*sudoku->size+j-1]);
            if (error !=1)
                return 0;
        }
    return 1;
}
```

Η παραπάνω συνάρτηση θεωρεί ότι οι αρνητικοί αριθμοί τόσο στο αρχείο ανάγνωσης όσο και στη μνήμη υποδηλώνουν τις αρχικές τιμές ενός Sudoku.

Με βάση τα παραπάνω κατασκευάστε λοιπόν μια συνάρτηση

```
void write_puzzle(sudoku_type sudoku, FILE *out);
```

που αποθηκεύει ένα Sudoku (λυμένο ή μη) σε αρχείο. Όπως και η read_puzzle, έτσι και η write_puzzle θα πρέπει να επιστρέφει μηδενική τιμή σε περίπτωση σφάλματος και μη μηδενική τιμή διαφορετικά.



Θέμα 2: Εμφάνιση Sudoku

Κατασκευάστε μια συνάρτηση

```
void display_puzzle(struct sudoku_type *sudoku);
```

η οποία δέχεται ως είσοδο έναν δείκτη σε δομή τύπου struct sudoku_type και εμφανίζει στην οθόνη το σχετικό Sudoku. Η εμφάνιση του Sudoku στην οθόνη μπορεί να είναι σύμφωνα με τις οδηγίες που έχουν δοθεί στην εργασία E2, με μόνη τη διαφορά ότι οι αριθμοί στα κελιά μπορούν να διαχωρίζονται με στηλοθέτες (ώστε να επιτρέπονται πολυψήφιοι αριθμοί), όπως π.χ. φαίνεται παρακάτω:

(2)	3	(8)		0	(5)	0		(1)	0	(7)
4	7	(6)		0	0	(2)		0	0	5
(1)	(9)	5		(7)	0	3		0	0	(8)

0	6	(2)		0	0	(8)		0	(5)	1
(5)	8	0		0	(6)	1		0	0	(3)
3	(4)	1		(9)	2	5		(7)	8	6

(6)	0	0		0	0	(4)		8	(7)	(2)
0	0	0		(1)	0	0		(5)	3	4
(8)	0	(4)		0	(3)	7		(6)	1	(9)

Θέμα 3: Συμπλήρωση Sudoku

Κατασκευάστε μια συνάρτηση

```
int play(sudoku_type* sudoku);
```

που επαναλαμβανόμενα δέχεται τιμές από το χρήστη για τα κενά κελιά του sudoku, μέχρι να επιλυθεί το sudoku. Για τη συνάρτηση που θα κατασκευάσετε ισχύουν όλες οι οδηγίες του αντίστοιχου θέματος της εργασίας E2.

Θέμα 4: Κατασκευή ολοκληρωμένου προγράμματος

Στόχος αυτού του θέματος είναι η κατασκευή ενός ολοκληρωμένου προγράμματος με κεντρικό μενού διαχείρισης των Sudoku. Ειδικότερα, το πρόγραμμα αυτό θα έχει τη δυνατότητα να διατηρεί στη μνήμη περισσότερα από ένα Sudoku (διαφορετικής ενδεχομένως διάστασης το καθένα), και ο χρήστης να επιλέγει με ποιο θα εργαστεί. Το κεντρικό μενού του προγράμματος θα έχει τις παρακάτω επιλογές:

1. Ανάγνωση Sudoku από αρχείο
2. Επιλογή Sudoku (εμφάνιση του ID της τρέχουσας επιλογής ή "None" εάν δεν υπάρχει τέτοια).
3. Εμφάνιση τρέχοντος Sudoku
4. Συμπλήρωση τρέχοντος Sudoku
5. Αποθήκευση τρέχοντος Sudoku
6. Διαγραφή τρέχοντος Sudoku από τη μνήμη



Για τη διατήρηση περισσότερων του ενός Sudoku στη μνήμη θα χρησιμοποιηθεί μια απλά συνδεδεμένη λίστα. Θα χρειαστεί λοιπόν να τροποποιηθεί η δομή `sudoku_type` έτσι ώστε να επιτρέπει το σχηματισμό συνδεδεμένων λιστών. Η νέα τροποποιημένη δομή `sudoku_type` θα έχει ως εξής:

```
struct sudoku_type
{
    int id;
    int size;
    int* values;
    struct sudoku_type* next;
};
```

Στην παραπάνω ενημερωμένη δομή έχουν προστεθεί δύο πεδία: το πεδίο `next` που αποτελεί δείκτη σε δομή του ίδιου τύπου (δίνοντάς μας έτσι τη δυνατότητα να δημιουργούμε συνδεδεμένες λίστες) και το πεδίο `id`, το οποίο θα χρησιμοποιηθεί για την αντιστοίχιση ενός μοναδικού αριθμού σε κάθε Sudoku (όπως συμβαίνει σε όλα τα περιοδικά με προβλήματα Sudoku, όπου κάθε Sudoku έχει το δικό του αύξοντα αριθμό).

Λόγω της προσθήκης του πεδίου `id` στην ενημερωμένη δομή `sudoku_type`, η τιμή του πεδίου αυτού θα πρέπει να γράφεται στα και να διαβάζεται από τα αρχεία αποθήκευσης στον δίσκο των διαφόρων Sudoku. Κάτι τέτοιο μπορεί να γίνει προσθέτοντας έναν ακόμη αριθμό στην αρχή κάθε αρχείου περιγραφής ενός Sudoku (π.χ. σε ξεχωριστή σειρά, αμέσως μετά τη διάσταση του Sudoku). Θα πρέπει λοιπόν να τροποποιήσετε τις συναρτήσεις ανάγνωσης/εγγραφής Sudoku κατάλληλα.

Για τη διατήρηση των πολλαπλών Sudoku στη μνήμη θα χρειαστεί να κατασκευαστεί μια συνδεδεμένη λίστα. Θα πρέπει λοιπόν το πρόγραμμα να διατηρεί στη μνήμη έναν δείκτη που να δείχνει στην αρχή αυτής της λίστας. Επιπλέον θα πρέπει να διατηρείται και ένας δείκτης που να δείχνει στο τρέχον Sudoku, δηλαδή στο Sudoku με το οποίο εργαζόμαστε την τρέχουσα στιγμή. Έστω λοιπόν οι δύο αυτοί δείκτες:

```
struct sudoku_type *sudoku_list;
struct sudoku_type *current_sudoku;
```

Επανερχόμενοι στο κεντρικό μενού του ολοκληρωμένου προγράμματος, οι διάφορες επιλογές του θα εκτελούν τις παρακάτω λειτουργίες:

1. Ανάγνωση Sudoku από αρχείο: Θα ζητείται από το χρήστη το όνομα ενός αρχείου και θα φορτώνεται το αντίστοιχο Sudoku στη μνήμη. Το φορτωθέν Sudoku θα τοποθετείται στην αρχή της συνδεδεμένης λίστας και θα γίνεται τρέχον.
2. Επιλογή Sudoku: Θα ζητείται από τον χρήστη να δώσει τον αύξοντα αριθμό (`id`) ενός Sudoku και στη συνέχεια θα γίνεται σειριακή αναζήτηση στη λίστα των Sudoku για εύρεση του πρώτου Sudoku με αύξοντα αριθμό ίδιο με το δοθέντα αριθμό. Εάν βρεθεί κάποιο Sudoku, τότε ο δείκτης `current_sudoku` θα δείχνει προς αυτό. Ειδικά ο δείκτης `current_sudoku` θα γίνεται NULL.
3. Εμφάνιση τρέχοντος Sudoku: Εάν `current_sudoku != NULL`, θα εμφανίζεται το τρέχον Sudoku.
4. Συμπλήρωση τρέχοντος Sudoku: Εάν `current_sudoku != NULL`, θα καλείται η συνάρτηση συμπλήρωσης για το τρέχον Sudoku.
5. Αποθήκευση τρέχοντος Sudoku: Εάν `current_sudoku != NULL`, θα ζητείται από το χρήστη το όνομα ενός αρχείου όπου και θα αποθηκεύεται το τρέχον Sudoku.
6. Διαγραφή τρέχοντος Sudoku από τη μνήμη: Εάν `current_sudoku != NULL`, το τρέχον Sudoku θα διαγράφεται από τη μνήμη, ενός θα γίνεται η κατάλληλη συντήρηση της λίστας των Sudoku. Η μεταβλητή `current_sudoku` θα παίρνει την τιμή NULL. Προσοχή: Για καλύτερη διαχείριση της μνήμης θα πρέπει πρώτα να διαγράφεται η μνήμη του πίνακα ακεραίων του πεδίου `values` και στη συνέχεια η μνήμη της δομής `sudoku_type`.



Κριτήρια αξιολόγησης:

Θέμα 1: Ανάγνωση / Αποθήκευση Sudoku	15
Θέμα 2: Εμφάνιση Sudoku	15
Θέμα 3: Συμπλήρωση Sudoku	20
Θέμα 4: Κατασκευή ολοκληρωμένου προγράμματος	
- Μενού	10
- Επιλογή τρέχοντος Sudoku	15
- Διαγραφή τρέχοντος Sudoku	15
Γενική εικόνα (ευανάγνωστος κώδικας, σχολιασμός, κλπ.)	10
ΣΥΝΟΛΟ	100

Ο συνολικός βαθμός θα διαιρεθεί δια 10, ώστε να προκύψει ο τελικός βαθμός της εργασίας (με μέγιστη τιμή το 10).

Τρόπος – Ημερομηνία Παράδοσης

Η εργασία σας θα πρέπει να έχει φτάσει στον Καθηγητή-Σύμβουλό σας μέχρι την Κυριακή 2/12/2007 ώρα 23:59.

Περιμένουμε όλες οι εργασίες να αποσταλούν μέσω Email και να είναι γραμμένες σε επεξεργαστή κειμένου MSWord (διευκρινήσεις, σχόλια και παραδοχές που θεωρείτε απαραίτητα να συμπεριλάβετε στην εργασία σας). Τα τμήματα κώδικα και τα τμήματα δεδομένων θα βρίσκονται σε ξεχωριστά αρχεία και θα υπάρχουν αναφορές σ' αυτά από το κείμενο της εργασίας. Στον Καθηγητή-Σύμβουλό σας, σε κάθε περίπτωση, στέλνετε ΕΝΑ μόνο αρχείο (συμπιεσμένο).

Δεν θα δοθεί παράταση στην παράδοση της εργασίας πέραν της ως άνω αναφερόμενης ημέρας και ώρας, για κανένα λόγο. Την Τρίτη 4/12/2007 ώρα 13:00, θα δημοσιευθεί ενδεικτική επίλυση της εργασίας στο διαδίκτυο (portal).

Καλή Επιτυχία!!!