ΜΥΥ502 – ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΣΥΣΤΗΜΑΤΩΝ

Εργαστήριο #4
8/11/2016

I. ΖΕΣΤΑΜΑ (I)

Να γραφτεί ένα πρόγραμμα που ζητά από τον χρήστη το μέγεθος ενός πίνακα, για αποθήκευση ακεραίων. Στη συνέχεια, δεσμεύει δυναμικά χώρο για τον πίνακα αυτό, τον γεμίζει με τυχαίους ακεραίους, και υπολογίζει τον μέσο όρο τους. Στην τελική εκτύπωση, να τυπώνετε τα στοιχεία του πίνακα και τον μέσο όρο τους.

Για να χρησιμοποιήσετε (ψευδο)τυχαίους αριθμούς, πρέπει να κάνετε:

```
#include <time.h>
#include <stdlib.h>
```

Επιπλέον, κάπου στην αρχή της `main()` θα πρέπει να έχετε την εντολή:

```
srand(time(NULL));
```

Από κει και στο εξής, για να παίρνετε τυχαίους αριθμούς από το 0 μέχρι το `MAXNUM-1`, αρκεί να κάνετε:

```
num = rand() % MAXNUM;
```

II. ΖΕΣΤΑΜΑ (II) – ΣΥΜΠΛΗΡΩΣΤΕ ΤΑ ΚΕΝΑ

Δίνεται το πρόγραμμα `lab4-2-fill.c` το οποίο δεν είναι ολοκληρωμένο και πρέπει να το ολοκληρώσετε στα σημεία που αναγράφεται σε σχόλια '`FILL HERE`'. Σκοπός του είναι να ζητά από τον χρήστη το μέγεθος ενός πίνακα, τον οποίο δεσμεύει δυναμικά και γεμίζει με κάποιον αριθμό. Στη συνέχεια ζητά από τον χρήστη ένα νέο μέγεθος για τον πίνακα και του το αλλάζει, φροντίζοντας να μηδενίσει τα επιπλέον στοιχεία, εφόσον το νέο μέγεθος είναι μεγαλύτερο από το παλιό.

III. MANUAL PAGES & STRDUP

Τα manual pages (εντολή τερματικού `man`) είναι το "ευαγγέλιο" που θα πρέπει να συμβουλεύεστε αν δεν γνωρίζετε πώς λειτουργεί μία εντολή τερματικού ή και μία συνάρτηση βιβλιοθήκης. Έχουν το επιπλέον πλεονέκτημα ότι *δεν χρειάζεται internet για τη λειτουργία τους.*

Το `<string.h>` παρέχει πολλές ακόμα συναρτήσεις, πέραν των σημαντικών που είδαμε στις διαλέξεις. Για παράδειγμα, υπάρχει η συνάρτηση `strdup()` η οποία, δοσμένης μίας συμβολοσειράς, παράγει ένα πιστό αντίγραφό της: δεσμεύει όσο χώρο χρειάζεται και στη συνέχεια αντιγράφει εκεί τους χαρακτήρες. Για περισσότερες πληροφορίες εκτελέστε:

```
man strdup
```

και θα σας εμφανιστεί το manual page της συνάρτησης. Κοιτάξτε το πρωτότυπό της, την ενότητα DESCRIPTION και την ενότητα RETURN VALUE για να δείτε ακριβώς πώς πρέπει να είναι και πώς πρέπει να λειτουργεί η συνάρτηση.

Στη συνέχεια, υλοποιήστε τη δική σας `my_strdup()` (δεν μπορείτε να την ονομάσετε `strdup()` μιας και αυτή ήδη υπάρχει), η οποία θα πρέπει να λειτουργεί ακριβώς όπως η `strdup()`.

IV. ΑΠΟ ΤΗΝ ΠΕΡΣΙΝΗ ΠΡΟΟΔΟ

Φτιάξτε ένα πρόγραμμα στο οποίο η `main()` δέχεται ορίσματα και αθροίζει την αξία όλων των χαρακτήρων που είναι αριθμητικά ψηφία. Συγκεκριμένα:

1. για κάθε όρισμά της, ελέγχει ποια γράμματα είναι ψηφία (δηλαδή χαρακτήρες από `'0'` έως `'9'`),
2. αθροίζει την αριθμητική αξία όλων αυτών των ψηφίων (το `'0'` έχει αξία 0, το `'1'` έχει αξία 1, κλπ),
3. Εκτυπώνει το άθροισμα και τερματίζει. Αν ο χρήστης δεν έδωσε ορίσματα, να εκτυπώνει 0.

Για παράδειγμα, η εκτέλεση:

```
» ./a.out Perasa To MYY502 Me 10
```

θα πρέπει να τυπώσει 8 ($= 5 + 0 + 2 + 1 + 0$).

V. ΕΠΙΠΛΕΟΝ ΕΞΑΣΚΗΣΗ: ΤΡΙΓΩΝΟ PASCAL

Το γνωστό *τρίγωνο του Pascal* είναι μια τριγωνική διάταξη των δυωνυμικών συντελεστών. Για παράδειγμα, για $n = 5$ έχει ως εξής:

```
1
1   1
1   2   1
1   3   3   1
1   4   6   4   1
```

Πρόκειται για έναν κάτω τριγωνικό πίνακα n γραμμών όπου τα στοιχεία της πρώτης στήλης (στήλη 0) και της διαγωνίου είναι ίσα με 1, ενώ οποιοδήποτε άλλο στοιχείο στη γραμμή i και στη στήλη j ισούται με το άθροισμα του ακριβώς από πάνω στοιχείου (δηλ. στη γραμμή $i-1$, στήλη j) και του από πάνω και αριστερά (δηλ. στη γραμμή $i-1$, στήλη $j-1$).

Ζητείται να φτιάξετε συνάρτηση `int **pascal(int n);` η οποία θα κατασκευάζει *δυναμικά* το τρίγωνο του Pascal με n γραμμές και θα το επιστρέφει. Κάθε γραμμή του πίνακα θα πρέπει να έχει χώρο για ακριβώς όσα στοιχεία χρειάζονται (δηλ. η γραμμή i θα πρέπει να έχει $i+1$ στοιχεία).

Η `main()` θα περιμένει ως μοναδικό όρισμα (... argc, argv) το n, θα καλεί την `pascal()` για να φτιαχτεί το τρίγωνο, θα το τυπώνει στην οθόνη, και θα απελευθερώνει την μνήμη που δεσμεύτηκε.