ΜΥΥ502 – ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΣΥΣΤΗΜΑΤΩΝ

Εργαστήριο #2
25/10/2016

Ι. «ΖΕΣΤΑΜΑ» ΜΕ ΔΕΙΚΤΕΣ (POINTERS)

1. Σας δίνεται ο παρακάτω κώδικας. Για κάθε μία εντολή printf προσδιορίστε τι εκτυπώνει και γιατί.

```
int a[10] = {0,2,4,6,8,7,6,4,2,0};
int *pa = &a[1], *pb = &a[8], *pc;
printf("1. *(a + 6) = %d\n", *(a + 6));
printf("2. pb - pa = %d\n", pb - pa);
printf("3. pb[1] = %d\n", pb[1]);
printf("4. &pb[1] = %p\n", &pb[1]);   /* For pointers better use %p, not %d */
printf("5. *pa += 3 = %d\n", *pa += 3);
printf("6. *(pb -= 3) = %d\n", *(pb -= 3));
```

2. Οι εντολές printf (4–6), θέλουμε να τυπώνουν αντίστοιχα το περιεχόμενο του ένατου κελιού του πίνακα a, το περιεχόμενο του τέταρτου κελιού του πίνακα a, και τη διεύθυνση του πέμπτου κελιού του πίνακα a. Κάντε τις απαραίτητες διορθώσεις.
3. Συμπληρώστε τον κώδικα ώστε ο pc να δείχνει στο μηδενικό κελί του πίνακα a, και να τυπώνει την τιμή του. Στον κώδικά σας πρέπει να κάνετε χρήση δεικτών.

ΙΙ. ΕΥΡΕΣΗ ΜΕΓΙΣΤΟΥ, ΕΛΑΧΙΣΤΟΥ ΚΑΙ ΜΕΣΟΥ ΟΡΟΥ ΜΕ ΜΙΑ ΚΛΗΣΗ

Σας ζητείται να υλοποιήσετε μία **συνάρτηση** η οποία υπολογίζει και το *μέγιστο* στοιχείο και το *ελάχιστο* στοιχείο ενός πίνακα ο οποίος της δίνεται ως παράμετρος, ενώ επίσης επιστρέφει και τον *μέσο όρο* των στοιχείων. Συγκεκριμένα, το πρωτότυπο της συνάρτησης θα είναι ως εξής:

```
double minmaxavg(double array[], int num, < συμπληρώστε παραμέτρους >);
```

όπου array είναι ο πίνακας που θα δοθεί, num είναι το πλήθος των στοιχείων του, και η τιμή που επιστρέφεται από τη συνάρτηση είναι ο μέσος όρος των στοιχείων. Θα χρειαστεί να βάλετε δύο ακόμα κατάλληλες παραμέτρους ώστε η συνάρτηση να υπολογίζει το μέγιστο και το ελάχιστο στοιχείο.

- Η main() πρέπει να ορίζει τον πίνακα και να τον αρχικοποιεί κατά τη δήλωση.
- Δεν επιτρέπονται καθολικές (global) μεταβλητές.
- **Οι εκτυπώσεις των αποτελεσμάτων γίνονται μέσα στη** main().

ΙΙΙ. ΕΥΡΕΣΗ ΣΤΟΙΧΕΙΟΥ

Υλοποιήστε μία συνάρτηση search() η οποία δέχεται τρεις παραμέτρους: α) έναν πίνακα ακεραίων, β) το μέγεθός του και γ) έναν ακόμα αριθμό. Η συνάρτηση θα ψάχνει να βρει αν ο αριθμός αυτός υπάρχει μέσα στον πίνακα.

- Αν υπάρχει, πρέπει να επιστρέψει *δείκτη* προς το αντίστοιχο στοιχείο του πίνακα
- Αν δεν υπάρχει, πρέπει να επιστρέψει *NULL*.

Η main(), μετά την κλήση της συνάρτησης, και εφόσον δεν έλαβε NULL, θα πρέπει να τυπώνει την τιμή του στοιχείου καθώς και *ποια θέση έχει στον πίνακα* χρησιμοποιώντας μόνο τον δείκτη που επέστρεψε η search().

ΙV. ΣΥΝΕΝΩΣΗ (MERGE) ΤΑΞΙΝΟΜΗΜΕΝΩΝ ΠΙΝΑΚΩΝ

Υλοποιήστε μία συνάρτηση void mergearrays(int *A, int na, int *B, int nb, int *C) η οποία καλείται με ορίσματα 3 πίνακες A (με na στοιχεία), B (με nb στοιχεία) και C (με τουλάχιστον na+nb στοιχεία). Οι πίνακες A και B υποτίθεται ότι έχουν τα στοιχεία τους ταξινομημένα σε αύξουσα σειρά και ο στόχος είναι στον πίνακα C να αντιγραφούν όλα τα στοιχεία των A και B έτσι ώστε και ο πίνακας C να είναι ταξινομημένος σε αύξουσα σειρά. Προσπαθήστε να χρησιμοποιήσετε μόνο δείκτες.

(Επιπλέον ασκήσεις στην επόμενη σελίδα)

V. ΕΠΙΠΛΕΟΝ ΕΞΑΣΚΗΣΗ: ΤΑΞΙΝΟΜΗΣΗ (SORTING)

Σε αυτή την άσκηση καλείστε να ταξινομήσετε έναν πίνακα ακεραίων με τον αλγόριθμο της φυσαλίδας (bubblesort). Ο χρήστης θα δίνει ως είσοδο έναν πίνακα ακεραίων και στο τέλεος θα εμφανίζεται ως έξοδος ο ίδιος ο πίνακας ταξινομημένος. Για την υλοποίηση πρέπει να φτιάξετε τις παρακάτω 3 συναρτήσεις. Για κάθε μία θα πρέπει να αποφασίσετε τι ορίσματα θα παίρνει και τι θα επιστρέφει.

- `compare()`: Δέχεται δύο μεταβλητές x, y και συγκρίνει ποια είναι μεγαλύτερη. Το αποτέλεσμα θα είναι 1 αν ο πρώτος αριθμός είναι μεγαλύτερος, 0 αν είναι ίσες οι μεταβλητές και -1 αν ο πρώτος αριθμός είναι ο μικρότερος.
- `swap()`: Δέχεται δύο μεταβλητές και κάνει εναλλαγή των τιμών τους (*πέρασμα με αναφορά*).
- `bubblesort()`: Η συγκεκριμένη συνάρτηση είναι που πρέπει να υλοποιεί την ταξινόμηση. Θα δέχεται ως όρισμα ένα πίνακα και θα πρέπει να ταξινομεί τα στοιχεία του πίνακα κατά αύξουσα σειρά. Για την υλοποίησή της πρέπει να χρησιμοποιήσετε τις δύο παραπάνω συναρτήσεις. Στο πλαίσιο που ακολουθεί σας δίνεται με μορφή ψευδοκώδικα η bubblesort:

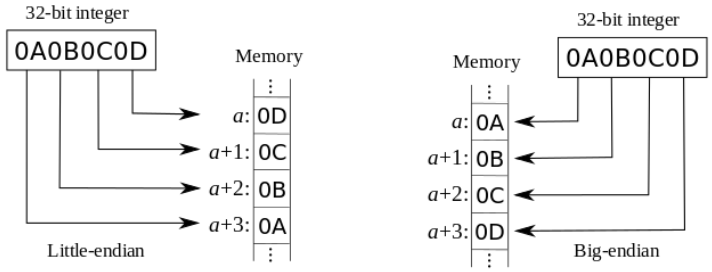
Αλγόριθμος BubbleSort

```
Είσοδος: πίνακας ακεραίων A, μεγέθους N
Έξοδος: πίνακας A ταξινομημένος
Για i από 0 έως N-1
        Για j από N-1 έως i
                Αν A[j] < A[j-1] (μέσω της compare()) τότε
                        αντάλλαξε A[j] και A[j-1] (μέσω swap())
                τέλος_αν
        τέλος_για
τέλος_για
```

VI. ΕΠΙΠΛΕΟΝ ΕΞΑΣΚΗΣΗ: ENDIANESS

Προσπαθήστε να λύσετε το πρόβλημα που αναφέρεται και στις διαφάνειες των διαλέξεων: *πώς μπορώ με δείκτες να καταλάβω αν ο υπολογιστής μου χρησιμοποιεί αποθήκευση μεγάλου ή μικρού άκρου;*

Όπως γνωρίζετε από το μάθημα της Αρχιτεκτονικής Υπολογιστών, τα 4 bytes τα οποία καταλαμβάνει ένας 32-μπιτος αριθμός, μπορούν να αποθηκευτούν με δύο διαφορετικούς τρόπους. Ανάλογα με ποιο από τα 4 bytes αποθηκεύεται πρώτο (δηλ. στη μικρότερη διεύθυνση στη μνήμη), έχουμε είτε την αποθήκευση μικρού άκρου (little-endian) είτε την αποθήκευση μεγάλου άκρου (big endian), όπως φαίνεται στα παρακάτω σχήματα.



Μπορείτε να σκεφτείτε μία λύση κάνοντας χρήση δεικτών;

VII. ΕΠΙΠΛΕΟΝ ΕΞΑΣΚΗΣΗ: ΑΝΤΙΣΤΡΟΦΗ ΣΥΜΒΟΛΟΣΕΙΡΑΣ ΜΕ ΔΕΙΚΤΕΣ

Δημιουργείστε ένα πρόγραμμα το οποίο διαβάζει ένα string (συμβολοσειρά) από το πληκτρολόγιο και το τυπώνει αντεστραμμένο. Για το σκοπό αυτό θα πρέπει να υλοποιηθούν οι παρακάτω δύο συναρτήσεις

1. `char *string_end(char *str);` η οποία δέχεται ένα δείκτη σε συμβολοσειρά και επιστρέφει έναν δείκτη στον τελευταίο χαρακτήρα της (αυτόν πριν το `'\0'`).
2. `void string_reverse(char *str);` η οποία δέχεται ένα δείκτη σε συμβολοσειρά την οποία και αντιστρέφει, δηλαδή ο πρώτος χαρακτήρας γίνεται τελευταίος, ο δεύτερος γίνεται προτελευταίος κλπ. Για την υλοποίηση θα πρέπει να χρησιμοποιήσετε δύο δείκτες: έναν `start` που θα δείχνει στην αρχή του string και έναν `end` που θα δείχνει στο τέλος του (το οποίο το βρίσκετε μέσω κλήσης στην `string_end()`) και να ακολουθήσετε τον παρακάτω αλγόριθμο:

Αντιστροφή συμβολοσειράς

```
μέχρι να διασταυρωθούν οι δυο δείκτες
   εναλλαγή των χαρακτήρων που δείχνουν οι δείκτες start και end
   ο start προχωράει στον επόμενο και ο end στον προηγούμενο χαρακτήρα
τέλος_μέχρι
```