Εργαστήριο #6: Δείκτες και Πίνακες

Στο εργαστήριο αυτό, θα μάθουμε για τους δείκτες της C και θα τους χρησιμοποιήσουμε για να κατασκευάσουμε συναρτήσεις που επιστρέφουν τιμές, μέσω ορισμάτων που είναι δείκτες, στις συναρτήσεις που τις καλούν. Επίσης, θα μάθουμε να ορίζουμε και να χρησιμοποιούμε πίνακες για να υλοποιούμε περισσότερο πολύπλοκες αλγοριθμικές διαδικασίες, θα δούμε τη σχέση τους με τους δείκτες και πώς να τους αντιμετωπίζουμε σαν τύπους δεδομένων (για να τους περνάμε σαν ορίσματα σε συναρτήσεις, κλπ.).

Άσκηση 1: Πέρασμα δεδομένων μέσω δεικτών - (myprog.c)

1.1 Γράψτε την παρακάτω συνάρτηση μέσα σ'ένα αρχείο `myprog.c` και μεταγλωττίστε το:

```
void badf(int x, int y, int sum, int diff) {
  sum = x + y;
  diff = x - y;
}
```

Γράψτε και μία συνάρτηση `main` μέσα σ' ένα αρχείο `myprog.c` η οποία να υλοποιεί την παρακάτω διαδικασία (εκφρασμένη σε ψευδογλώσσα) και μεταγλωττίστε το:

```
Όρισε τις ακέραιες μεταβλητές a, b, sum, diff

Θέσε a=5, b=4, sum=0, diff=0

Κάλεσε badf(a, b, sum, diff)

Τύπωσε sum, diff
```

Δημιουργήστε το εκτελέσιμο πρόγραμμα myprog και εκτελέστε το. Τι παρατηρείτε; Μπορείτε να εξηγήσετε το αποτέλεσμα;

1.2 Τροποποιήστε τη συνάρτηση `void badf(int x, int y, int sum, int diff)` ώστε οι `sum` και `diff` να είναι δείκτες σε ακεραίους, και μετονομάστε την σε `goodf`. Τροποποιήστε την main του προγράμματος ώστε να καλεί τη συνάρτηση `goodf` αντί για την `badf`. Τι παρατηρείτε;

1.3 Χρησιμοποιήστε τη συνάρτηση `scanf()` για να διαβάσετε τους ακέραιους αριθμούς `a`, `b` που εμφανίζονται στην main.

Υπενθυμίζουμε ότι η συνάρτηση `scanf()` διαβάζει από την είσοδο δεδομένα προς επεξεργασία. Συντάσσεται με αντίστοιχο τρόπο με την `printf()` δηλαδή:

```
scanf("%y", &var);
```

όπου `y` είναι σύμβολο που αντιστοιχεί στον τύπο δεδομένων της μεταβλητής var (π.χ, `d` για `int`, `f` για `float`, κλπ.). Παρατηρούμε ότι στο δεύτερο όρισμα περνιέται η διεύθυνση της μεταβλητής `var`, για να αποθηκευθεί η καταχώρηση και μετά το πέρας της κλήσης.

Άσκηση 2: Το κόσκινο του Ερατοσθένη - (sieve.c)

Το κόσκινο του Ερατοσθένη είναι ένας αλγόριθμος για την εύρεση όλων των πρώτων αριθμών σε ένα εύρος τιμών (από 2 έως N - 1). Είναι ένας από τους αρχαιότερους γνωστούς αλγορίθμους και οφείλεται στον Έλληνα φιλόσοφο και αστρονόμο Ερατοσθένη (276-194 π.Χ.). Ο αλγόριθμος εξετάζει διαδοχικά όλους τους ακεραίους και για κάθε αριθμό που συναντά διαγράφει όλα τα πολλαπλάσιά του (αφού σίγουρα δεν είναι πρώτοι).

Παρατηρήστε τα τρία πρώτα βήματα του αλγορίθμου, για N=20.

Όταν ξεκινά ο αλγόριθμος, όλοι οι αριθμοί θεωρούνται πιθανοί πρώτοι (αρχικοποιημένοι στο 1):

2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Ο «2» είναι πρώτος, διαγραφή των 4, 6, 8, 10, 12, 14, 16, 18, 20 (το σημειώνουμε με *):

2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0

Ο «3» είναι πρώτος, διαγραφή των 6, 9, 12, 15, 18:

2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	1	0	1	0	0	0	1	0	1	0	0	0	1	0	1	0

Ο «4» δεν εξετάζεται γιατί έχει αφαιρεθεί σε προηγούμενο βήμα.

Ορίστε έναν πίνακα N θέσεων και αρχικοποιήστε τον με μονάδες. Η διάσταση του πίνακα να ορίζεται μέσω #define με τιμή ίση με 50. Υλοποιήστε το κόσκινο του Ερατοσθένη σύμφωνα με τον αλγόριθμο όπως εκφράζεται παρακάτω σε ψευδογλώσσα:

```
Για i=2 έως N-1 επανάλαβε
  Θέσε A[i]=1

Για i=2 έως N-1 επανάλαβε
  Αν A[i]!=0
    Για j=2*i έως N-1 με βήμα i επανάλαβε
      Θέσε A[j]=0

Για i=2 έως N-1 επανάλαβε
  Αν το A[i]==1 τύπωσε "i is a prime number"
```

Εκτελέστε το πρόγραμμά σας και επιβεβαιώστε την ορθότητα του αποτελέσματος.

Άσκηση 3: Πίνακες και αριθμητική δεικτών - (pointers.c)

3.1 Ορίστε τη συνάρτηση `void print_array(int *array, int n)` που δέχεται σαν όρισμα έναν πίνακα ακεραίων και τη διάστασή του και εκτυπώνει τα περιεχόμενα του σε μία γραμμή χωρισμένα με στηλογνώμονα (tab).

3.2 Δημιουργήστε το πρόγραμμα `pointers.c` που δημιουργεί έναν πίνακα 8 θέσεων και κάνει πράξεις πάνω σε αυτόν, ακολουθώντας το παρακάτω τμήμα κώδικα:

```
01: int i, a[8], *pa;
02:
03: for (i=0; i<8; i++)
04:   a[i] = i*i;
05:
06: pa = &a[0];
07: a[6] = *(a+4);
08: *(pa+3) = a[5];
09: a[0] = *((pa++)+2);
10: *((++pa)+5) = a[1];
11: *(&a[5]-1) = *(--pa);
```

3.3 Εκτυπώστε τα περιεχόμενα του πίνακα, χρησιμοποιώντας τη συνάρτηση `print_array()` μετά από τα βήματα 04, 07, 08, 09, 10 και 11 για να παρακολουθήσετε την επίδραση των εντολών στον πίνακα.

Άσκηση 4: Πίνακες και συναρτήσεις - judgement.c

Σε ένα αγώνισμα υπάρχουν 10 κριτές. Η βαθμολογία του κάθε αγωνιζόμενου βγαίνει από τον μέσο όρο 8 κριτών, αφού εξαιρεθεί αυτός με τη μεγαλύτερη και αυτός με τη μικρότερη βαθμολογία (για να αποφευχθούν φαινόμενα μεροληψίας, είτε υπέρ είτε κατά ενός αθλητή).

4.1 Ορίστε τη συνάρτηση `int max_array(int *array, int n)` που επιστρέφει στο όνομά της το μέγιστο στοιχείο του πίνακα ακεραίων A διάστασης n.

4.2 Ορίστε τη συνάρτηση `int min_array(int *array, int n)` που επιστρέφει στο όνομά της το ελάχιστο στοιχείο του πίνακα ακεραίων A διάστασης n.

4.3 Ορίστε τη συνάρτηση `int sum_array(int *array, int n)` που επιστρέφει στο όνομά της το άθροισμα των στοιχείων του πίνακα ακεραίων A διάστασης n.

4.4 Κατασκευάστε το πρόγραμμα `judgement.c` το οποίο να διαβάζει τις 10 βαθμολογίες με χρήση της scanf() και να τυπώνει τη βαθμολογία με ακρίβεια 2 δεκαδικών ψηφίων σύμφωνα με τον τύπο:

```
Βαθμός = (Άθροισμα - Μέγιστο - Ελαχιστο) / 8
```

Έστω ένα αρχείο `grades.txt` με τις παρακάτω βαθμολογίες:

```
8 2 3 9 0 10 8 3 4 5
```

Παράδειγμα εκτέλεσης ακολουθεί:

```
$ ./judgement  < grades.txt
Sum: 52
Min: 0
Max: 10
Average: 5.25
```