



104

Αλγόριθμοι, Προγραμματισμός και Δομές Δεδομένων

Εργαστήριο – Κεφάλαιο 13

Δομές

Διδάσκουσα:
Σοφία Αλμπάνη



Άσκηση 13.1

13.1) Να γράψετε πρόγραμμα σε C το οποίο χρησιμοποιεί **δομές (struct)** και **δείκτες** για την αποθήκευση και διαχείριση στοιχείων φοιτητών. Συγκεκριμένα:

- Να ορίσετε **μία δομή Student** η οποία θα περιέχει:
 - το ονοματεπώνυμο του φοιτητή,
 - τον κωδικό του,
 - τον βαθμό του.
- Στη **συνάρτηση main**:
 - να δηλώνετε έναν πίνακα 100 στοιχείων τύπου Student.
 - για κάθε στοιχείο του πίνακα, το πρόγραμμα να διαβάζει από το πληκτρολόγιο το όνομα, τον κωδικό και τον βαθμό του φοιτητή, να αποθηκεύει τα δεδομένα στη δομή, και να τα εμφανίζει.
 - **A περίπτωση:** Με χρήση δεικτών θέσης (π.χ. stud[i]).
 - **B περίπτωση:** Με χρήση δείκτη σε δομή (struct Student *) για τη διαπέραση του πίνακα.

Σημείωση: Για την ανάγνωση των συμβολοσειρών να χρησιμοποιηθεί η δοθείσα συνάρτηση read_text.

Στόχος: Εξοικείωση με τη χρήση δομών και δεικτών για τη διαπέραση και διαχείριση πίνακα δομών.



Ενδεικτική Λύση 13.1 – Α περίπτωση

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int read_text(char str[], int size, int flag);
#define SIZE 100

struct Student
{
    char name[50];
    int code;
    float grd;
};

int main()
{
    struct Student stud[SIZE];
    int i;

    for (i = 0; i < SIZE; i++)
    {
        printf("\nEnter name: ");
        read_text(stud[i].name, sizeof(stud[i].name), 1);

        printf("Enter code: ");
        scanf("%d", &stud[i].code);

        printf("Enter grade: ");
        scanf("%f", &stud[i].grd);

        printf("\nN: %s C: %d G: %.2f\n",
            stud[i].name,
            stud[i].code,
            stud[i].grd);

        getchar();
    }
    return 0;}
```



Ενδεικτική Λύση 13.1 – Β περίπτωση

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int read_text(char str[], int size, int flag);

#define SIZE 100

struct Student
{
    char name[50];
    int code;
    float grd;
};

int main()
{
    struct Student *ptr, stud[SIZE];

    for(ptr = stud; ptr < stud+SIZE; ptr++) /* Με κάθε αύξηση, ο ptr δείχνει στο επόμενο στοιχείο του πίνακα, δηλαδή στην επόμενη δομή. */
    {
        printf("\nEnter name: ");
        read_text(ptr->name, sizeof(ptr->name), 1); // ισοδύναμο με (*ptr).name

        printf("Enter code: ");
        scanf("%d", &ptr->code);

        printf("Enter grade: ");
        scanf("%f", &ptr->grd);

        printf("\nN: %s C: %d G: %.2f\n", ptr->name, ptr->code, ptr->grd);
        getchar();
    }
    return 0;
}
```

Άσκηση 13.2

13.2) Να γράψετε πρόγραμμα σε **C** το οποίο διαχειρίζεται πληροφορίες για πόλεις, χρησιμοποιώντας **δομές (struct)** και πίνακα δομών. Συγκεκριμένα:

- Να ορίσετε μία **δομή City** η οποία θα περιέχει:
 - το όνομα της πόλης,
 - το όνομα της χώρας στην οποία ανήκει,
 - τον πληθυσμό της πόλης.
- Να δηλώσετε έναν πίνακα 100 στοιχείων τύπου City.
- Στη **συνάρτηση main** να διαβάζει από το πληκτρολόγιο, για κάθε πόλη:
 - το όνομα της πόλης,
 - τη χώρα στην οποία ανήκει,
 - τον πληθυσμό της.
- Στη συνέχεια, να ζητά από τον χρήστη: το όνομα μίας χώρας, και έναν ακέραιο αριθμό που αντιστοιχεί σε πληθυσμό.
- Να πραγματοποιείται αναζήτηση στον πίνακα των πόλεων και να εμφανίζονται:
 - τα ονόματα και οι πληθυσμοί όλων των πόλεων που ανήκουν στη συγκεκριμένη χώρα **και** έχουν πληθυσμό μεγαλύτερο από τον πληθυσμό που δόθηκε από τον χρήστη.
- Αν δεν βρεθεί καμία πόλη που να ικανοποιεί τα παραπάνω κριτήρια, να εμφανίζεται κατάλληλο μήνυμα.

Σημείωση: Για την ανάγνωση των συμβολοσειρών να χρησιμοποιηθεί η δοθείσα συνάρτηση `read_text`.

Στόχος: Υλοποίηση αναζήτησης σε πίνακα δομών με χρήση συμβολοσειρών και σύνθετων λογικών συνθηκών.



Ενδεικτική Λύση 13.2

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define SIZE 100
int read_text(char str[], int size, int flag);

struct City
{
    char name[50];
    char cntry[50];
    int pop;
};

int main(void)
{
    char cntry[50];
    int i, pop, found;
    struct City city[SIZE];

    for(i = 0; i < SIZE; i++)
    {
        printf("\nCity: ");
        read_text(city[i].name, sizeof(city[i].name), 1);

        printf("Country: ");
        read_text(city[i].cntry, sizeof(city[i].cntry), 1);

        printf("Population: ");
        scanf("%d", &city[i].pop);
        getchar();
    }
```

Συνέχεια:

```
printf("\nEnter country to search: ");
read_text(cntry, sizeof(cntry), 1);

printf("Population: ");
scanf("%d", &pop);

found = 0;
for(i = 0; i < SIZE; i++)
{
    if((strcmp(city[i].cntry, cntry) == 0) && (city[i].pop > pop))
    {
        found = 1;
        printf("C: %s P: %d\n", city[i].name, city[i].pop);
    }
}
if(found == 0)
printf("\nNone city is found\n");
}
```

Άσκηση 13.3

13.3) Να γράψετε πρόγραμμα σε C το οποίο μελετά τη χρήση δομών (struct) σε συνδυασμό με **συναρτήσεις που επιστρέφουν δείκτη σε δομή**. Συγκεκριμένα:

- Να ορίσετε **μία δομή Student** η οποία θα περιέχει:
 - το όνομα του φοιτητή,
 - τον κωδικό του,
 - τον βαθμό του.
- Να δηλώσετε μία **καθολική μεταβλητή** τύπου Student.
- Να υλοποιήσετε **συνάρτηση test** η οποία:
 - δεν δέχεται παραμέτρους,
 - επιστρέφει δείκτη σε δομή τύπου Student,
 - επιστρέφει τη διεύθυνση της καθολικής μεταβλητής.
- Στη **συνάρτηση main**:
 - να δηλώσετε μία μεταβλητή τύπου Student και έναν δείκτη που να δείχνει σε αυτή,
 - να καλέσετε τη συνάρτηση test,
 - να αντιγράψετε το περιεχόμενο της δομής που επιστρέφεται από τη συνάρτηση στη δομή της main,
 - να εμφανίσετε τα στοιχεία της δομής μετά την αντιγραφή.

Στόχος: Κατανόηση του τρόπου με τον οποίο μπορεί να επιστραφεί δείκτης σε δομή από συνάρτηση, καθώς και η αντιγραφή δεδομένων μεταξύ δομών.



Ενδεικτική Λύση 13.3

```
#include <stdio.h>
struct Student
{
    char name[50];
    int code;
    float grd;
};

struct Student *test(void), st;

int main()
{
    struct Student stud = {"somebody", 1111, 7.5}, *ptr = &stud;

    *ptr = *test();
    printf("%s %d %d\n", ptr->name, ptr->code, ptr->grd);
    return 0;
}

struct Student *test(void)
{
    return &st;
}
```

Άσκηση 13.4

13.4) Να γράψετε πρόγραμμα σε C το οποίο χρησιμοποιεί δομές (struct) και **συναρτήσεις που επιστρέφουν δομή** για τη μετατροπή χρόνου. Συγκεκριμένα:

- Να ορίσετε μία **δομή Time** η οποία θα περιέχει:
 - τις ώρες,
 - τα λεπτά,
 - τα δευτερόλεπτα.
- Το πρόγραμμα να ζητά από τον χρήστη έναν ακέραιο αριθμό που αντιστοιχεί σε **συνολικά δευτερόλεπτα**.
- Να υλοποιήσετε **συνάρτηση mk_time** η οποία:
 - δέχεται ως παράμετρο **δείκτη σε ακέραιο** που αναπαριστά τα συνολικά δευτερόλεπτα,
 - υπολογίζει τις αντίστοιχες ώρες, λεπτά και δευτερόλεπτα,
 - επιστρέφει μία μεταβλητή τύπου Time που περιέχει τα αποτελέσματα.
- Στη **συνάρτηση main**:
 - να κληθεί η mk_time,
 - να αποθηκευτεί το αποτέλεσμα σε μεταβλητή τύπου Time,
 - και να εμφανιστούν τα επιμέρους στοιχεία του χρόνου σε μορφή Ώρες:Λεπτά:Δευτερόλεπτα.

Στόχος: Χρήση συναρτήσεων που επιστρέφουν δομές και η μετατροπή δεδομένων μέσω υπολογισμών.



Ενδεικτική Λύση 13.4

```
#include <stdio.h>
struct Time
{
    int hours;
    int mins;
    int secs;
};
struct Time mk_time(const int *s);

int main(void)
{
    int secs;
    struct Time t;

    printf("Enter seconds: ");
    scanf("%d", &secs);

    t = mk_time(&secs);
    printf("\nH:%d M:%d S:%d\n", t.hours, t.mins, t.secs);
    return 0;
}

struct Time mk_time(const int *s)
{
    struct Time tmp;

    tmp.hours = *s/3600;
    tmp.mins = (*s%3600)/60;
    tmp.secs = *s%60;
    return tmp;
}
```



Άσκηση 13.5

13.5) Να γράψετε πρόγραμμα σε C το οποίο χρησιμοποιεί **εμφωλευμένες δομές (struct)** και **συναρτήσεις με παραμέτρους δείκτες σε δομές** για τον υπολογισμό του εμβαδού ορθογωνίου. Συγκεκριμένα:

- Να ορίσετε μία **δομή Coord** η οποία θα αναπαριστά ένα σημείο στο καρτεσιανό επίπεδο και θα περιέχει:
 - τη συντεταγμένη x,
 - τη συντεταγμένη y.
- Να ορίσετε μία δεύτερη **δομή Rectangle** η οποία θα περιέχει:
 - δύο σημεία τύπου Coord, τα οποία αντιστοιχούν στα άκρα της διαγωνίου ενός ορθογωνίου.
- Να υλοποιήσετε **συνάρτηση rect_area** η οποία:
 - δέχεται ως παραμέτρους δείκτες σε δύο δομές Coord,
 - υπολογίζει το μήκος της βάσης και το ύψος του ορθογωνίου ως απόλυτες διαφορές των αντίστοιχων συντεταγμένων,
 - επιστρέφει το εμβαδόν του ορθογωνίου.
- Στη **συνάρτηση main**, να διαβάζει από το πληκτρολόγιο:
 - τις συντεταγμένες (x, y) του πρώτου σημείου της διαγωνίου,
 - τις συντεταγμένες (x, y) του δεύτερου σημείου της διαγωνίου.
 - να καλείται η rect_area και να εμφανίζεται το εμβαδόν του ορθογωνίου.

Στόχος: Κατανόηση εμφωλευμένων δομών και ο υπολογισμός γεωμετρικών μεγεθών με χρήση δεικτών σε δομές.



Ενδεικτική Λύση 13.5

```
#include <stdio.h>
```

```
struct Coord
```

```
{
    double x; /* Τετμημένη σημείου. */
    double y; /* Τεταγμένη σημείου. */
};
```

```
struct Rectangle
```

```
{
    struct Coord point_A; /* Πρώτο σημείο διαγωνίου. */
    struct Coord point_B; /* Δεύτερο σημείο διαγωνίου. */
};
```

```
double rect_area(const struct Coord *c1, const struct Coord *c2);
```

```
int main(void)
```

```
{
    struct Rectangle rect;

    printf("Enter the x and y coords of the first point: ");
    scanf("%lf%lf", &rect.point_A.x, &rect.point_A.y);

    printf("Enter the x and y coords of the second point: ");
    scanf("%lf%lf", &rect.point_B.x, &rect.point_B.y);

    printf("Area: %f\n",
        rect_area(&rect.point_A, &rect.point_B));
    return 0;
}
```

```
double rect_area(const struct Coord *c1, const struct Coord *c2)
{
    double base, height;
    if(c1->x > c2->x)
        base = c1->x - c2->x;
    else
        base = c2->x - c1->x;

    if(c1->y > c2->y)
        height = c1->y - c2->y;
    else
        height = c2->y - c1->y;
    return base*height; /* Επιστροφή εμβαδού παρ/μου. */
}
```

Άσκηση 13.6

13.6) Να γράψετε πρόγραμμα σε C το οποίο διαχειρίζεται στοιχεία φοιτητών με χρήση **δομών (struct)**, **πινάκων δομών** και **συναρτήσεων**.

- Να ορίσετε μία δομή **Student** η οποία θα περιέχει: 1) το όνομα του φοιτητή, 2) τον κωδικό του, 3) τον βαθμό του.
- Να υλοποιήσετε συνάρτηση **sort_by_grade** η οποία:
 - δέχεται ως παραμέτρους τον πίνακα φοιτητών και το πλήθος τους,
 - ταξινομεί τους φοιτητές κατά **φθίνουσα σειρά βαθμού**.
- Να υλοποιήσετε συνάρτηση **show_students** η οποία:
 - δέχεται ως παραμέτρους τον πίνακα φοιτητών, το πλήθος τους και τον μέσο όρο βαθμολογίας,
 - εμφανίζει τα στοιχεία (όνομα, κωδικός, βαθμός) μόνο των φοιτητών που έχουν βαθμό **μεγαλύτερο από τον μέσο όρο**,
 - διακόπτει την εμφάνιση μόλις συναντήσει φοιτητή με βαθμό μικρότερο ή ίσο του μέσου όρου.
- Να δηλώσετε έναν **πίνακα δομών Student** μέγιστου μεγέθους 100 στοιχείων.
- Στη συνάρτηση **main** να διαβάζει επαναληπτικά δεδομένα φοιτητών από το πληκτρολόγιο ως εξής:
 - να ζητά τον βαθμό του φοιτητή (τιμές από 0 έως 10),
 - αν δοθεί βαθμός ίσος με **-1**, η διαδικασία εισαγωγής να τερματίζεται,
 - διαφορετικά, να διαβάζονται το όνομα και ο κωδικός του φοιτητή και να αποθηκεύονται στον πίνακα.
 - κατά την εισαγωγή, να υπολογίζεται το άθροισμα των βαθμών.
 - Αν δεν εισαχθεί κανένας φοιτητής, το πρόγραμμα να τερματίζει.
 - να καλείται η **sort_by_grade** και να υπολογίζεται ο μέσος όρος βαθμολογίας,
 - να καλείται η **show_students** για την εμφάνιση των αποτελεσμάτων.

*Σημείωση: Για την ανάγνωση των συμβολοσειρών να χρησιμοποιηθεί η δοθείσα συνάρτηση **read_text**.*

Στόχος: Ταξινόμηση και φιλτράρισμα πίνακα δομών με χρήση συναρτήσεων και μέσου όρου.



Ενδεικτική Λύση 13.6 (α)

```
#include <stdio.h>, #include <stdlib.h>, #include <string.h>

#define SIZE 100
struct Student
{
    char name[50];
    int code;
    float grd;
};

void sort_by_grade(struct Student studs[], int num_studs);
void show_students(const struct Student studs[], int num_studs, float avg_grd);
int read_text(char str[], int size, int flag);
int main(void)
{
    int i;
    float sum_grd;
    struct Student studs[SIZE];
    sum_grd = 0;
    for(i = 0; i < SIZE; i++)
    {
        printf("\nGrade [0-10]: ");
        scanf("%f", &studs[i].grd);
        if(studs[i].grd == -1)
            break;

        sum_grd += studs[i].grd;
        getchar();

        printf("Name: ");
        read_text(studs[i].name, sizeof(studs[i].name), 1);

        printf("Code: ");
        scanf("%d", &studs[i].code);
    }
```

```
if(i == 0)
    return 0;
    sort_by_grade(studs, i); /* Ταξινόμηση δομών σύμφωνα με την τιμή
του πεδίου grd. Η μεταβλητή i δείχνει πόσοι φοιτητές εισήχθησαν. */
    show_students(studs, i, sum_grd/i); /* Το τελευταίο όρισμα είναι
ο μέσος όρος των βαθμών. */
    return 0;
}
```



Ενδεικτική Λύση 13.6 (β)

```
void sort_by_grade(struct Student studs[], int num_studs)
{
    int i, j;
    struct Student tmp;

    for(i = 0; i < num_studs-1; i++)
    {
        /* Γίνεται σύγκριση του πεδίου grd κάθε δομής με το αντίστοιχο πεδίο των υπολοίπων δομών και αν βρεθεί κάποια με
        μεγαλύτερη τιμή, τότε οι δομές αντιμετατίθενται. */
        for(j = i+1; j < num_studs; j++)
        {
            if(studs[i].grd < studs[j].grd)
            {
                tmp = studs[i];
                studs[i] = studs[j];
                studs[j] = tmp;
            }
        }
    }
}

void show_students(const struct Student studs[], int num_studs, float avg_grd)
{
    int i;

    for(i = 0; i < num_studs; i++)
    {
        if(studs[i].grd > avg_grd)
            printf("N: %s C: %d G: %f\n", studs[i].name, studs[i].code, studs[i].grd);
        else
            return; /* Αφού οι δομές έχουν ταξινομηθεί σε φθίνουσα σειρά και βρήκαμε κάποιον φοιτητή με βαθμό
            μικρότερο ή ίσο από τον μέσο όρο, δεν χρειάζεται να συνεχίσουμε την επανάληψη. */
    }
}
```

Άσκηση 13.7

13.7) Να γράψετε πρόγραμμα σε C το οποίο προσομοιώνει την επεξεργασία μιας έγχρωμης εικόνας με χρήση **δομών (struct)** και **δισδιάστατων πινάκων δομών**. Συγκεκριμένα:

- Να ορίσετε μία **δομή Pixel** η οποία θα αναπαριστά ένα εικονοστοιχείο (pixel) σε μορφή **RGB** και θα περιέχει τις συνιστώσες κόκκινου, πράσινου και μπλε χρώματος.
- Να υλοποιήσετε **συνάρτηση rotate_right_90** η οποία:
 - δέχεται ως παραμέτρους την αρχική εικόνα και έναν πίνακα για το αποτέλεσμα,
 - περιστρέφει την εικόνα κατά 90° δεξιόστροφα,
 - αποθηκεύει το αποτέλεσμα στον πίνακα tmp.
- Στη **συνάρτηση main**:
 - Να δηλώσετε έναν **δισδιάστατο πίνακα img διαστάσεων ROWS × COLS** που θα αναπαριστά την αρχική εικόνα, καθώς και έναν **δεύτερο πίνακα tmp διαστάσεων COLS × ROWS** για την αποθήκευση της περιστραμμένης εικόνας.
 - Να αρχικοποιήσετε την αρχική εικόνα με τυχαίες τιμές χρώματος στο διάστημα [0, 255] για κάθε pixel.
 - Να εμφανίσετε την αρχική εικόνα στην οθόνη, εμφανίζοντας για κάθε pixel τις τιμές RGB.
 - Να εμφανίσετε τη νέα (περιστραμμένη) εικόνα στην οθόνη.

Στόχος: Κατανόηση της διαχείρισης δισδιάστατων πινάκων δομών.



Ενδεικτική Λύση 13.7 (α)

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

#define ROWS    3
#define COLS    5

struct Pixel /* RGB μορφή (Red-Green-Blue). */
{
    unsigned char red; /* Τιμή στο [0, 255]. */
    unsigned char green;
    unsigned char blue;
};

void rotate_right_90(const struct Pixel img[][COLS], struct Pixel tmp[][ROWS]);

int main(void)
{
    int i, j;
    struct Pixel img[ROWS][COLS], tmp[COLS][ROWS];

    srand(time(NULL));
    /* Δημιουργούμε τυχαία χρώματα. */
    for(i = 0; i < ROWS; i++)
    {
        for(j = 0; j < COLS; j++)
        {
            img[i][j].red = rand()%256;
            img[i][j].green = rand()%256;
            img[i][j].blue = rand()%256;
        }
    }
}
```



Ενδεικτική Λύση 13.7 (β)

```
printf("*** Original Image ***\n\n");
for(i = 0; i < ROWS; i++)
{
    for(j = 0; j < COLS; j++)
    {
        printf("(%3d,%3d,%3d) ", img[i][j].red, img[i][j].green, img[i][j].blue);
    }
    printf("\n");
}
rotate_right_90(img, tmp); /* Κλήση της συνάρτησης περιστροφής. */

printf("\n*** Rotated Image ***\n\n");
for(i = 0; i < COLS; i++)
{
    for(j = 0; j < ROWS; j++)
    {printf("(%3d,%3d,%3d) ", tmp[i][j].red, tmp[i][j].green, tmp[i][j].blue);
    }
    printf("\n");
}
return 0;
}

void rotate_right_90(const struct Pixel img[][COLS], struct Pixel tmp[][ROWS])
{
    int i, j, k = 0;
    for(i = ROWS-1; i >= 0; i--)
    {
        for(j = 0; j < COLS; j++)
        {tmp[j][i] = img[k][j];
        }
        k++;
    }
}
```



Η συνάρτηση read_text:

```
int read_text(char str[], int size, int flag)
{
    int len;

    if(fgets(str, size, stdin) == NULL)
    {
        printf("Error: fgets() failed\n");
        exit(EXIT_FAILURE);
    }

    len = strlen(str);
    if(len > 0)
    {
        if(flag && (str[len - 1] == '\n'))
        {
            str[len - 1] = '\0';
            len--;
        }
    }
    else
    {
        printf("Error: No input\n");
        exit(EXIT_FAILURE);
    }

    return len;
}
```

Συνοπτικά

Κατηγορία	Συντακτικό	Περιγραφή
Ορισμός δομής	<code>struct S { ... };</code>	Νέος σύνθετος τύπος δεδομένων
Μεταβλητή δομής	<code>struct S s;</code>	Δήλωση δομής
Δείκτης σε δομή	<code>struct S *p;</code>	Δείχνει σε δομή
Πρόσβαση μέλους	<code>s.x</code> ή <code>p->x</code>	Πρόσβαση σε πεδία
Πίνακας δομών	<code>struct S a[N];</code>	Πολλές δομές
Διέλευση με δείκτη	<code>p = a; p < a+N;</code>	Διαπέραση πίνακα
Πέρασμα σε συνάρτηση	<code>f(s) / f(&s)</code>	Αντιγραφή / αναφορά
Επιστροφή δομής	<code>return s;</code>	Επιστροφή τιμής
Επιστροφή δείκτη	<code>return &s;</code>	Επιστροφή αναφοράς
Εμφωλευμένες δομές	<code>struct A{struct B;}</code>	Δομή μέσα σε δομή
Αντιγραφή δομών	<code>s1 = s2;</code>	Αντιγραφή όλων
<code>const</code> δομή	<code>const struct S*</code>	Μόνο ανάγνωση