Δρ. Γ. Σ. Τσελίκης | Δρ. Ν. Δ. Τσελίκας

C Από τη Θεωρία στην Εφαρμογή

Δ' Έκδοση

| **500+** λυμένες ασκήσεις και παραδείγματα κλιμακούμενης δυσκολίας

| Ιδιαίτερη έμφαση σε σύνθετες έννοιες της γλώσσας όπως δείκτες, αρχεία και δυναμική διαχείριση μνήμης

| Εισαγωγή σε αλγορίθμους και δομές δεδομένων

| Εισαγωγή στις **C++** και **Java**

Ενδεικτικές Ασκήσεις Βιβλίου

Ε.1 Ποια είναι η έξοδος του παρακάτω προγράμματος;

```
#include <stdio.h>
int main(void)
{
	int i = 10, j = 20, k = 0;

	if(i = 40)
		printf("One ");
	if(j = 50)
		printf("Two ");
	if(k = 60)
		printf("Three ");
	if(k = 0)
		printf("Four ");
	printf("%d %d %d\n", i, j, k);
	return 0;
}
```

Ε.2 Σε ένα τεστ 50 ερωτήσεων πολλαπλής επιλογής η πρώτη απάντηση παίρνει τρεις βαθμούς, η δεύτερη έναν βαθμό και η τρίτη δύο βαθμούς. Να γραφεί ένα πρόγραμμα το οποίο να διαβάζει τις 50 απαντήσεις ενός εξεταζόμενου και να εμφανίζει τους συνολικούς του βαθμούς με χρήση της εντολής **switch**. Αν ο εξεταζόμενος επιλέξει τρεις φορές διαδοχικά την πρώτη απάντηση το πρόγραμμα να εμφανίζει τους συνολικούς του βαθμούς και να τερματίζει. Θεωρήστε ότι ο εξεταζόμενος δίνει έγκυρες απαντήσεις με αριθμούς στο [1, 3]. Ο παρακάτω κώδικας δεν είναι σωστός. Μπορείτε να βρείτε το γιατί;

```
#include <stdio.h>
int main(void)
{
	int i, ans, first_ans, points;

	points = first_ans = 0;
	for(i = 0; i <= 50; i++)
	{
		printf("Enter answer [1-3]: ");
		scanf("%d", &ans);

		switch(ans)
		{
			case 1:
				first_ans++;
				if(first_ans == 3)
				{
					printf("Candidate gets %d points in
total\n", points);
```

```
                                        return 0;/*Τερματισμός προγράμματος.*/
                                }
                                points += 3;
                        break;

                        case 2:
                                points += 1;
                        break;

                        case 3:
                                points += 2;
                        break;
                }
        }
        printf("Candidate gets %d points in total\n", points);
        return 0;
}
```

Ε.3 Ποια είναι η έξοδος του παρακάτω προγράμματος;

```
#include <stdio.h>
int main(void)
{
        unsigned char i;

        for(i = 3; i && i-1; i--)
                printf("%d ", i);
        for(; ++i;)
                printf("\n%d ", i);
        return 0;
}
```

Ε.4 Να γραφεί ένα πρόγραμμα το οποίο να εμφανίζει, αν υπάρχουν, τις ακέραιες λύσεις της εξίσωσης: `3x + 7y − 5z = 10`, όπου τα `x`, `y`, `z` είναι ακέραιοι στο [-30, 30]. Επίσης, αν υπάρχουν λύσεις, να εμφανίζει ξεχωριστά στο τέλος και αυτή στην οποία το άθροισμα των `x`, `y`, `z` έχει τη μικρότερη τιμή.

Ε.5 Να γραφεί ένα πρόγραμμα το οποίο να διαβάζει τους ακεραίους κωδικούς προϊόντων και να τους αποθηκεύει σε έναν πίνακα 50 θέσεων. Αν δεν υπάρχει άλλη διαθέσιμη θέση ή ο χρήστης εισάγει την τιμή `-1`, η εισαγωγή των κωδικών να τερματίζει. Το πρόγραμμα πριν αποθηκεύσει κάποιον κωδικό πρέπει να ελέγχει αν ήδη υπάρχει και μόνο αν δεν υπάρχει να τον αποθηκεύει στον πίνακα. Δηλαδή, όλα τα στοιχεία του πίνακα πρέπει να είναι διαφορετικά μεταξύ τους.

Ε.6 Να γραφεί ένα πρόγραμμα το οποίο να διαβάζει 100 ακεραίους και να τους αποθηκεύει σε έναν πίνακα. Στη συνέχεια, να βρίσκει το στοιχείο του πίνακα, ώστε όλα τα προηγούμενα από αυτό να έχουν μικρότερη τιμή και όλα τα επόμενα από αυτό να έχουν μεγαλύτερη τιμή. Αν υπάρχουν παραπάνω από ένα τέτοια στοιχεία, το πρόγραμμα να εμφανίζει το μεγαλύτερο από αυτά. Αν δεν υπάρχει κανένα τέτοιο στοιχείο, το πρόγραμμα να εμφανίζει ανάλογο μήνυμα. Το πρώτο και τελευταίο

στοιχείο του πίνακα να μην ελεγχθούν, αφού δεν υπάρχουν στοιχεία αριστερά του και δεξιά του, αντίστοιχα. Για παράδειγμα, αν θεωρήσουμε ότι τα στοιχεία του πίνακα είναι έξι και έχουν τις τιμές 2, 1, 3, 4, 10, 6 το πρόγραμμα πρέπει να εμφανίζει την τιμή 4, ενώ αν τα στοιχεία έχουν τις τιμές 1, 3, 3, 4, 4, 6, το πρόγραμμα πρέπει να εμφανίζει το μήνυμα ότι δεν υπάρχει κανένα τέτοιο στοιχείο.

Ε.7 Να γραφεί ένα πρόγραμμα το οποίο να προσομοιώνει ένα παιχνίδι κλήρωσης αριθμών με ηλεκτρονικό τρόπο. Κάθε κλήρωση αντιστοιχεί στη δημιουργία 10 τυχαίων θετικών ακεραίων στο [0, 100]. Ωστόσο, το πρόγραμμα να γραφεί με τέτοιο τρόπο ώστε οι κληρώσεις να είναι «πειραγμένες». Συγκεκριμένα, η απαίτηση του προγράμματος είναι σε κάθε κλήρωση οι 3 από τους 10 τυχερούς αριθμούς να έχουν εμφανιστεί στην προηγούμενη κλήρωση. Το πρόγραμμα να ρωτάει τον χρήστη αν θέλει να παίξει και αν απαντήσει όχι, να τερματίζει. Για απλότητα, θεωρήστε ότι ένας αριθμός που κερδίζει μπορεί να εμφανιστεί παραπάνω από μία φορά στην τυχερή δεκάδα.

Ε.8 Να συμπληρώσετε τα κενά στο πρόγραμμα (δεν επιτρέπεται να προσθέσετε άλλες εντολές ή μεταβλητές) ώστε να υλοποιούνται τα παρακάτω:

α. Να αντιγράφει τα στοιχεία του πίνακα `a[8]` στον πίνακα `b[4][2]`.
β. Να υπολογίζει το γινόμενο των στοιχείων του πίνακα `b`.
γ. Να εμφανίζει τα στοιχεία του πίνακα `b` αντίστροφα, δηλαδή ξεκινώντας από το «κάτω-δεξιά» στοιχείο και καταλήγοντας στο «πάνω-αριστερά».

```
#include <stdio.h>

#define ROWS 4
#define COLS 2

int main(void)
{
	int i, j, p, a[ROWS*COLS] = {1, 2, 3, 4, 5, 6, 7, 8},
b[ROWS][COLS];

	/* Πρώτη σχέση. */
	for(i = 0; i < ____; i++)
		for(j = 0; j < ____; j++)
			____ = ____;
	/* Δεύτερη σχέση. */
	p = ____;
	for(i = 0; i < ____; ____)
		p = ____;

	/* Τρίτη σχέση. */
	for(i = ____; ____; ____)
		for(j = ____; ____; ____)
			printf("b[%d][%d] = %d\n", ____, ____, ____);
	return 0;
}
```

Ε.9 Να γραφεί ένα πρόγραμμα το οποίο να διαβάζει ακεραίους και να τους αποθηκεύει σε έναν τετραγωνικό πίνακα (π.χ. 3×3). Στη συνέχεια, να ελέγχει αν ο πίνακας αποτελεί μαγικό τετράγωνο, δηλαδή αν το άθροισμα της κάθε γραμμής, στήλης και διαγωνίου του είναι το ίδιο.

Ε.10 Ποια είναι η έξοδος του παρακάτω προγράμματος;

```
#include <stdio.h>
int main(void)
{
	int *ptr1, *ptr2, i = 10, j = 20;

	ptr1 = &i;
	ptr2 = &j;

	ptr2 = ptr1;
	*ptr1 = *ptr1 + *ptr2;
	*ptr2 *= 2;
	printf("%d\n", *ptr1 + *ptr2);
	return 0;
}
```

Ε.11 Να συμπληρώσετε το παρακάτω πρόγραμμα χρησιμοποιώντας τον δείκτη p2 για να διαβάζετε συνεχώς βαθμούς φοιτητών, τον δείκτη p1 για να εμφανίσετε πόσοι φοιτητές πήραν βαθμό στο [5, 10] και τον δείκτη p3 για να εμφανίσετε τον μεγαλύτερο βαθμό. Αν ο χρήστης εισάγει την τιμή -1, η εισαγωγή των βαθμών να τερματίζεται. Οι μεταβλητές sum, max και grade να χρησιμοποιηθούν μόνο μία φορά.

```
#include <stdio.h>
int main(void)
{
	int *p1, sum;
	float *p2, *p3, grade, max; /* Δεν επιτρέπεται να δηλώσετε άλλες
μεταβλητές. */
	...
}
```

Ε.12 Να συμπληρώσετε το παρακάτω πρόγραμμα ώστε να διαβάζει δύο ακεραίους και να εμφανίζει το άθροισμα των ακεραίων που μεσολαβούν μεταξύ τους, χρησιμοποιώντας τους δείκτες p1, p2 και p3. Για παράδειγμα, αν ο χρήστης εισάγει τους ακεραίους 6 και 10, το πρόγραμμα να εμφανίζει 24 (7+8+9). Το πρόγραμμα να υποχρεώνει τον χρήστη να εισάγει τιμές μικρότερες από 100 και ο πρώτος ακέραιος να είναι μικρότερος από τον δεύτερο. Οι μεταβλητές i, j και sum να χρησιμοποιηθούν μόνο μία φορά.

```
#include <stdio.h>
int main(void)
{
```

```
        int *p1, *p2, *p3, i, j, sum; /* Δεν επιτρέπεται να δηλώσετε
άλλες μεταβλητές. */
        ...
}
```

Ε.13 Να συμπληρώσετε το παρακάτω πρόγραμμα χρησιμοποιώντας τους δείκτες `p1` και `p2` για να διαβάσετε και να αποθηκεύσετε στον πίνακα `arr` τους ακεραίους κωδικούς 100 προϊόντων, με τον περιορισμό ένας κωδικός να αποθηκεύεται στον πίνακα μόνο αν δεν έχει ήδη αποθηκευτεί. Αν ο χρήστης εισάγει την τιμή `-1`, η εισαγωγή των κωδικών να τερματίζει. Το πρόγραμμα να εμφανίζει τους κωδικούς πριν τερματίσει. Η διαχείριση του πίνακα να γίνει με σημειογραφία δείκτη και η μεταβλητή `arr` να χρησιμοποιηθεί μέχρι 5 φορές.

```
#include <stdio.h>

#define SIZE 100
int main(void)
{
        int *p1, *p2, arr[SIZE]; /* Δεν επιτρέπεται να δηλώσετε άλλες
μεταβλητές. */
        ...
}
```

Για την εισαγωγή των κωδικών, το πρόγραμμα να εμφανίζει μηνύματα με την παρακάτω μορφή:

```
Enter code_1:
Enter code_2:
...
```

Ε.14 Ποιες είναι οι τιμές των στοιχείων του πίνακα `arr` στο παρακάτω πρόγραμμα;

```
#include <stdio.h>
int main(void)
{
        int *ptr, arr[5] = {20};

        for(ptr = arr+1; ptr < arr+4; ptr++)
                *ptr = *(ptr-1) + *(ptr+1) + 1;
        return 0;
}
```

Ε.15 Να συμπληρώσετε το παρακάτω πρόγραμμα, το οποίο να χρησιμοποιεί τη μεταβλητή `p1` για να διαβάζει συνεχώς έναν ακέραιο και να εμφανίζει τη λέξη `test` τόσες φορές όσες και η απόλυτη τιμή του ακεραίου. Το πρόγραμμα να χρησιμοποιεί τη μεταβλητή `p2` για να εμφανίσει τον συνολικό αριθμό των λέξεων που εμφανίστηκαν στην οθόνη. Αν ο χρήστης εισάγει το `0`, η εισαγωγή των ακεραίων να σταματάει. Χρησιμοποιήστε μόνο **`for`** βρόχους.

```
#include <stdio.h>
int main(void)
{
	int **p1, **p2, *p3, *p4, i, num, times; /* Δεν επιτρέπεται να δηλώσετε άλλες μεταβλητές. */
	...
}
```

Ε.16 Τι κάνει το παρακάτω πρόγραμμα;

```
#include <stdio.h>
int main(void)
{
	int *ptr, arr[2][5] = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100};

	for(ptr = arr[1]+2; ptr < arr[1]+5; ptr++)
		*ptr = 0;
	return 0;
}
```

Ε.17 Να γραφεί ένα πρόγραμμα το οποίο να διαβάζει συνεχώς έναν χαρακτήρα και, εάν αυτός είναι πεζό γράμμα, να εμφανίζει τον αντίστοιχο κεφαλαίο, αλλιώς να τον εμφανίζει όπως είναι. Αν ο χρήστης εισάγει διαδοχικά τους χαρακτήρες `':'` και `'q'`, το πρόγραμμα να εμφανίζει πόσες φορές εισήχθησαν οι χαρακτήρες `'w'` και `'x'` και μετά να τερματίζει. Σημειώστε ότι στον ASCII κώδικα η διαφορά ενός πεζού γράμματος από το αντίστοιχο κεφαλαίο είναι ίση με 32.

Ε.18 Ποια είναι η έξοδος του παρακάτω προγράμματος;

```
#include <stdio.h>
int main(void)
{
	char str1[] = "test", str2[] = "test";

	(str1 == str2) ? printf("One\n") : printf("Two\n");
	return 0;
}
```

Ε.19 Να γραφεί ένα πρόγραμμα το οποίο να προσομοιώνει το δημοφιλές παιχνίδι της «κρεμάλας». Το πρόγραμμα να διαβάζει μία λέξη μέχρι 30 χαρακτήρες, η οποία θα αποτελεί τη μυστική λέξη. Στη συνέχεια, το πρόγραμμα να διαβάζει επαναληπτικά κάποιο γράμμα από τον παίκτη. Αν το γράμμα δεν περιέχεται στη μυστική λέξη, το πρόγραμμα να ενημερώνει τον παίκτη. Αν περιέχεται, να εμφανίζει τα γράμματα που έχουν βρεθεί και στα υπόλοιπα να βάζει τον χαρακτήρα της υπογράμμισης *(underscore)*. Το παιχνίδι ολοκληρώνεται όταν βρεθεί η λέξη ή αν ο παίκτης έχει κάνει 7 λανθασμένες προβλέψεις. Θεωρήστε ότι δεν επιτρέπεται ο χρήστης να εισάγει το ίδιο γράμμα πάνω από μία φορά.

Ε.20 Ποια είναι η έξοδος του παρακάτω προγράμματος;

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

int main(void)
{
	char str[] = "Text", *p = str;
	int i;

	for(i = 0; i < strlen(str)-1; i++, p++)
		printf("%c", p[i]);
	return 0;
}
```

Ε.21 Ο αλγόριθμος συμπίεσης δεδομένων RLE *(Run Length Encoding)* βασίζεται στην παρατήρηση ότι μέσα σε ένα σύνολο δεδομένων το ίδιο σύμβολο μπορεί να επαναλαμβάνεται πολλές φορές στη σειρά. Αυτή η ακολουθία επαναλήψεων του συμβόλου μπορεί να αντικατασταθεί από:

(α) έναν ακέραιο που δηλώνει το πλήθος των συνεχόμενων εμφανίσεων και
(β) το σύμβολο αυτό

Να γραφεί ένα πρόγραμμα το οποίο να διαβάζει ένα αλφαριθμητικό μέχρι 100 χαρακτήρες και να το συμπιέζει σύμφωνα με τον RLE αλγόριθμο. Τα ψηφία, καθώς και οι χαρακτήρες που εμφανίζονται μία φορά να μην συμπιέζονται.

Για παράδειγμα, το αλφαριθμητικό: `fffmmmm1234jjjjjjjjjx`
να συμπιεσθεί ως εξής: `3f4m123410jx`

Ε.22 Ποια είναι η έξοδος του παρακάτω προγράμματος;

```
#include <stdio.h>
#include <string.h>
int main(void)
{
	char str[] = "noteasy";

	printf("%s\n", str+(*(str+3)-1)-str[strlen(str+3)]);
	return 0;
}
```

Ε.23 Να συμπληρώσετε τα κενά στο πρόγραμμα (δεν επιτρέπεται να προσθέσετε άλλες εντολές ή μεταβλητές) ώστε η τιμή του a μετά την κλήση της `f()` να γίνεται ίση με τετράγωνο της τιμής που θα εισάγει ο χρήστης (π.χ. αν ο χρήστης εισάγει 4, το πρόγραμμα να εμφανίζει 16).

```
#include <stdio.h>

void f(____);
```

```
int main(void)
{
        int a;

        scanf("%d", &a);

        ____;  /* Συμπληρώστε την κλήση της συνάρτησης. */

        printf("%d", a);
        return 0;
}

void f(____)
{
        ____;  /* Συμπληρώστε το σώμα της συνάρτησης. */
}
```

Ε.24 Δημιουργήστε μία **void** συνάρτηση που να δέχεται σαν παραμέτρους τους συντελεστές ενός τριωνύμου και να επιστρέφει τις πραγματικές ρίζες του, αν υπάρχουν. Να γραφεί ένα πρόγραμμα το οποίο να διαβάζει τους συντελεστές ενός τριωνύμου (π.χ. a, b και c) και να το επιλύει με χρήση της συνάρτησης. Το πρόγραμμα να υποχρεώνει τον χρήστη να εισάγει μη μηδενική τιμή για το a. Υπενθυμίζεται ότι για τις ρίζες ενός τριωνύμου $ax^2 + bx + c$ με $a \neq 0$ ελέγχουμε την τιμή της διακρίνουσας $D = b^2 - 4ac$:

α) Αν $D > 0$ υπάρχουν δύο πραγματικές ρίζες $r_{1,2} = (-b \pm \sqrt{D})/2a$.
β) Αν $D = 0$ υπάρχει μία (διπλή) ρίζα $r = -b/2a$.
γ) Αν $D < 0$ δεν υπάρχει πραγματική ρίζα.

Για τον υπολογισμό της τετραγωνικής ρίζας χρησιμοποιήστε τη συνάρτηση `sqrt()` που δηλώνεται στο `math.h`.

Ε.25 Ποια είναι η έξοδος του παρακάτω προγράμματος;

```
#include <stdio.h>

int *test(int p[], int *ptr2);

int main(void)
{
        int *ptr, i = 1, arr[] = {10, 20, 30, 40, 50, 60, 70};

        ptr = test(arr+2, &i);
        printf("%d %d\n", arr[4], *ptr);
        return 0;
}

int *test(int p[], int *ptr2)
{
        p[2] = 200;
```

```
        return p+*ptr2;
}
```

Ε.26 Δημιουργήστε μία συνάρτηση η οποία να δέχεται σαν παραμέτρους δύο αλφαριθμητικά και με χρήση δεικτών να επιστρέφει 0 αν είναι ίδια ή τη διαφορά των πρώτων χαρακτήρων τους που δεν είναι ίδιοι. Να γραφεί ένα πρόγραμμα το οποίο να διαβάζει δύο αλφαριθμητικά μέχρι 100 χαρακτήρες και να εμφανίζει το αποτέλεσμα της σύγκρισής τους με χρήση της συνάρτησης.

Ε.27 Ποια είναι η έξοδος του παρακάτω προγράμματος;

```
#include <stdio.h>

float *test(float *ptr1, float *ptr2);

int main(void)
{
        float a = 1.2, b = 3.4;

        *test(&a, &b) = 5.6;
        printf("val1 = %.1f val2 = %.1f\n", a, b);

        *test(&a, &b) = 7.8;
        printf("val1 = %.1f val2 = %.1f\n", a, b);
        return 0;
}

float *test(float *ptr1, float *ptr2)
{
        if(*ptr1 < *ptr2)
                return ptr1;
        else
                return ptr2;
}
```

Ε.28 Ποια είναι η έξοδος του παρακάτω προγράμματος;

```
#include <stdio.h>

void swap(char *s1, char *s2);

int main(void)
{
        char *p[] = {"Shadow", "Play"};

        swap(p[0], p[1]);
        printf("%s %s\n", p[0], p[1]);
        return 0;
}

void swap(char *s1, char *s2)
{
        char *p;
```

```
        p = s1;
        s1 = s2;
        s2 = p;
}
```

Ε.29 Μία εταιρεία μέτρησης της τηλεθέασης καταγράφει για κάθε μέρα της εβδομάδας τον αριθμό των τηλεθεατών που παρακολούθησαν τα κεντρικά δελτία ειδήσεων πέντε τηλεοπτικών σταθμών. Να γραφεί ένα πρόγραμμα το οποίο να εκτελεί διαδοχικά τις παρακάτω ενέργειες:

α) να καλεί μία συνάρτηση, η οποία για κάθε τηλεοπτικό σταθμό να διαβάζει το όνομά του (μέχρι 50 χαρακτήρες) και τον αριθμό των τηλεθεατών που παρακολούθησαν το κεντρικό του δελτίο ειδήσεων για κάθε μέρα της εβδομάδας και να αποθηκεύει αυτά τα στοιχεία σε δύο κατάλληλους πίνακες, αντίστοιχα.
β) να καλεί μία συνάρτηση, η οποία για κάθε τηλεοπτικό σταθμό να υπολογίζει τον μέσο όρο των τηλεθεατών που παρακολούθησαν το κεντρικό του δελτίο ειδήσεων στη διάρκεια της εβδομάδας και να επιστρέφει αυτούς τους πέντε μέσους όρους (ένας για κάθε σταθμό).
γ) να καλεί μία συνάρτηση, η οποία να εμφανίζει τα ονόματα των σταθμών για τους οποίους ο μέσος όρος της τηλεθέασης του είναι μικρότερος από τον μέσο όρο της τηλεθεάσής του το Σαββατοκύριακο. Αν δεν υπάρχει τέτοιος σταθμός, η συνάρτηση να εμφανίζει ανάλογο μήνυμα.
δ) να καλεί μία συνάρτηση, η οποία να εμφανίζει τα ονόματα των σταθμών οι οποίοι παρουσιάζουν αύξηση της τηλεθεάσής τους καθόλη τη διάρκεια της εβδομάδας. Αν δεν υπάρχει τέτοιος σταθμός, η συνάρτηση να εμφανίζει ανάλογο μήνυμα.

Θεωρήστε ότι οι τιμές που θα εισάγει ο χρήστης είναι έγκυρες. Υπάρχουν δύο περιορισμοί. Ο τύπος επιστροφής όλων των συναρτήσεων να είναι **void** και η `main()` να περιέχει μόνο δηλώσεις μεταβλητών και κλήσεις συναρτήσεων, τίποτα άλλο.

Ε.30 Δημιουργήστε μία αναδρομική συνάρτηση που να δέχεται σαν παράμετρο έναν θετικό ακέραιο αριθμό n και να επιστρέφει το παραγοντικό του, χρησιμοποιώντας τον τύπο $n! = n \times (n-1)!$. Το παραγοντικό ενός ακεραίου n, όπου $n >= 1$, είναι το γινόμενο των ακεραίων από το 1 μέχρι και το n, δηλαδή $1 \times 2 \times 3 \times ... \times n$. Το παραγοντικό του 0 είναι 1 $(0!=1)$. Να γραφεί ένα πρόγραμμα το οποίο να διαβάζει έναν θετικό ακέραιο μικρότερο από 170 και να εμφανίζει το παραγοντικό του με χρήση της συνάρτησης.

Ε.31 Ποια είναι η έξοδος του παρακάτω προγράμματος;

```
#include <stdio.h>

void f(int num);
```

```
int main(void)
{
	f(3);
	return 0;
}

void f(int num)
{
	int i;

	if(num >= 1)
	{
		printf("\n");
		for(i = 0; i < num; i++)
			printf("One ");
		f(num/2);
		printf("\n***");
		f(num/2);
	}
}
```

Ε.32 Να ορίσετε τη δομή `Product` με πεδία: όνομα προϊόντος, κωδικός και τιμή.

α) Δημιουργήστε μία συνάρτηση που να δέχεται σαν παραμέτρους έναν πίνακα τέτοιων δομών, τον αριθμό των προϊόντων που είναι αποθηκευμένα στον πίνακα και τον κωδικό ενός προϊόντος. Η συνάρτηση να ελέγχει αν υπάρχει κάποια δομή στον πίνακα που να έχει τον ίδιο κωδικό και, αν υπάρχει, να επιστρέφει έναν δείκτη σε αυτή τη δομή, αλλιώς να επιστρέφει `NULL`.
β) Να γραφεί ένα πρόγραμμα που να χρησιμοποιεί αυτή τη δομή για να εισάγει ο χρήστης τα στοιχεία `100` προϊόντων. Τα προϊόντα με τιμή πάνω από 20 Ευρώ να αποθηκεύονται σε έναν πίνακα τέτοιων δομών. Στη συνέχεια, να διαβάζει τον κωδικό ενός προϊόντος, να καλεί τη συνάρτηση και, αν το προϊόν είναι καταχωρημένο, να εμφανίζει τα υπόλοιπα στοιχεία του.

Ε.33 Να ορίσετε μία δομή συντεταγμένων (π.χ. `Coord`) με πεδία τους δεκαδικούς αριθμούς `x` και `y` που δηλώνουν τις συντεταγμένες ενός σημείου. Να ορίσετε, επίσης, μία δομή ορθογωνίου παραλληλογράμμου (π.χ. `Rectangle`) με πεδία δύο δομές τύπου `Coord` (π.χ. `point_A` και `point_B`).

α) Δημιουργήστε μία συνάρτηση που να δέχεται σαν παραμέτρους τα δύο ακραία σημεία της μίας διαγωνίου ενός ορθογωνίου παραλληλογράμμου, το καθένα από τα οποία θα πρέπει να είναι δομή τύπου `Coord`. Η συνάρτηση να υπολογίζει και να επιστρέφει το εμβαδό του ορθογωνίου.
β) Να γραφεί ένα πρόγραμμα που να χρησιμοποιεί τη δομή `Rectangle` για να εισάγει ο χρήστης τις συντεταγμένες των δύο ακραίων σημείων της μίας διαγωνίου ενός ορθογωνίου παραλληλογράμμου και να εμφανίζει το εμβαδό του με χρήση της συνάρτησης.

Ε.34 Ένα ξενοδοχείο αποτελείται από 5 ορόφους και κάθε όροφος από 10 δωμάτια. Σε κάθε όροφο τα δωμάτια είναι αριθμημένα με τους αριθμούς από το 1 μέχρι το 10. Κάθε δωμάτιο μπορεί να είναι διαθέσιμο ή όχι και διαθέτει μία, δύο ή τρεις κλίνες. Να γραφεί ένα πρόγραμμα το οποίο να εκτελεί διαδοχικά τις παρακάτω ενέργειες:

α) να ορίσετε τη δομή `Room` με πεδία: τύπος δωματίου και το αν το δωμάτιο είναι διαθέσιμο ή όχι. Για τη διαχείριση των δωματίων, το πρόγραμμα να χρησιμοποιεί τον πίνακα `room[5][10]` όπου το κάθε στοιχείο του πίνακα είναι δομή τύπου `Room`. Το πρόγραμμα να καλεί μία συνάρτηση, η οποία να διαβάζει τον τύπο του κάθε δωματίου (οι τιμές είναι 1, 2 ή 3 και ο τύπος δηλώνει τον αριθμό των κλινών) και το αν είναι διαθέσιμο ή όχι (η τιμή 1 δηλώνει διαθεσιμότητα) και να αποθηκεύει αυτές τις τιμές στα στοιχεία του πίνακα.
β) να καλεί μία συνάρτηση η οποία να επιστρέφει τον συνολικό αριθμό των κλινών στο ξενοδοχείο. Το πρόγραμμα να εμφανίζει αυτόν τον αριθμό.
γ) να διαβάζει τον τύπο ενός δωματίου (1, 2 ή 3) και να καλεί μία συνάρτηση, η οποία να επιστρέφει το μεγαλύτερο πλήθος συνεχόμενων δωματίων με αυτόν τον τύπο (τα συνεχόμενα δωμάτια πρέπει να είναι παραπάνω από ένα), καθώς και σε ποιον όροφο βρίσκεται αυτό το συνεχόμενο πλήθος δωματίων. Θεωρήστε ότι το μεγαλύτερο πλήθος βρίσκεται μόνο σε έναν όροφο. Το πρόγραμμα να εμφανίζει αυτές τις τιμές. Αν δεν υπάρχουν παραπάνω από ένα συνεχόμενα δωμάτια με αυτόν τον τύπο σε κανέναν όροφο, το πρόγραμμα να εμφανίζει ανάλογο μήνυμα.
δ) να καλεί μία συνάρτηση, η οποία να επιστρέφει όλους τους αριθμούς των δωματίων που είναι διαθέσιμα σε όλους τους ορόφους. Δηλαδή, για κάθε αριθμό δωματίου, το πρόγραμμα να ελέγχει αν είναι διαθέσιμο σε όλους τους ορόφους και να εμφανίζει ανάλογο μήνυμα (π.χ. το δωμάτιο με αριθμό 3 είναι διαθέσιμο σε όλους τους ορόφους). Αν δεν υπάρχει τέτοιο δωμάτιο, το πρόγραμμα να εμφανίζει ανάλογο μήνυμα.

Θεωρήστε ότι οι τιμές που θα εισάγει ο χρήστης είναι έγκυρες. Επίσης, υπάρχει ο περιορισμός ο τύπος επιστροφής όλων των συναρτήσεων να είναι **`void`**.

Ε.35 Να ορίσετε μία δομή `Country` με πεδία: όνομα, πρωτεύουσα και πληθυσμός (σημ. τα ονόματα είναι μέχρι 100 χαρακτήρες). Θεωρήστε ότι ένα αρχείο κειμένου περιέχει τα στοιχεία χωρών. Η πρώτη γραμμή του αρχείου περιέχει τον αριθμό των χωρών και κάθε επόμενη περιέχει τα στοιχεία της χώρας με τη μορφή:

```
όνομα  πρωτεύουσα  πληθυσμός
```

Να γραφεί ένα πρόγραμμα το οποίο να διαβάζει το όνομα του αρχείου και να χρησιμοποιεί τη δομή για να αποθηκεύσει τα στοιχεία των χωρών σε μία δυναμικά δεσμευμένη μνήμη. Στη συνέχεια, να διαβάζει έναν αριθμό και να εμφανίζει τα στοιχεία των χωρών με μεγαλύτερο πληθυσμό από αυτόν τον αριθμό.

Ε.36 Υποθέστε ότι το δυαδικό αρχείο `test.bin` περιέχει τους βαθμούς ενός φοιτητή. Η πρώτη εγγραφή στο αρχείο δηλώνει το πλήθος των βαθμών. Να γραφεί ένα πρόγραμμα το οποίο να διαβάζει τους βαθμούς από το αρχείο (χρησιμοποιήστε τον τύπο **`float`**) και να τους αποθηκεύει σε μία δυναμικά δεσμευμένη μνήμη. Στη συνέχεια, να διαβάζει έναν αριθμό και να εμφανίζει τους βαθμούς με μεγαλύτερη τιμή από αυτόν.

Ε.37 Δημιουργήστε μία μακροεντολή που να υπολογίζει τον μεγαλύτερο από δύο αριθμούς. Να γραφεί ένα πρόγραμμα το οποίο να διαβάζει τέσσερις ακεραίους και να εμφανίζει τον μεγαλύτερό τους με χρήση της μακροεντολής.

Ε.38 Να ορίσετε τη δομή `Athlete` με πεδία: όνομα και την επίδοση του αθλητή στο αγώνισμα του ύψους (θεωρήστε ότι είναι ακέραιο πεδίο που αντιστοιχεί στην επίδοση σε εκατοστά). Να γραφεί ένα πρόγραμμα το οποίο να εκτελεί διαδοχικά τις παρακάτω ενέργειες:

α) να χρησιμοποιεί αυτή τη δομή για να εισάγει ο χρήστης στοιχεία `16` αθλητών, τα οποία να αποθηκεύονται σε δύο πίνακες τέτοιων δομών (αντιστοιχούν σε δύο ομίλους αθλητών) από `8` αθλητές στον καθένα. Για την εισαγωγή των τιμών σε κάθε πίνακα, το πρόγραμμα να καλεί συνάρτηση.
β) να καλεί συνάρτηση που να ταξινομεί τον κάθε όμιλο σε αύξουσα σειρά ανάλογα με την επίδοση του κάθε αθλητή.
γ) να καλεί συνάρτηση για τη δημιουργία της τελικής οκτάδας. Στην τελική φάση προκρίνεται ο πρώτος αθλητής του κάθε ομίλου και οι έξι αθλητές με τις καλύτερες επιδόσεις από όλους τους υπόλοιπους συμμετέχοντες. Θεωρήστε ότι όλες οι επιδόσεις είναι διαφορετικές μεταξύ τους. Η συνάρτηση να επιστρέφει την τελική οκτάδα.
δ) Το πρόγραμμα να εμφανίζει τα ονόματα και τις επιδόσεις της τελικής οκτάδας.

Υπάρχει ο περιορισμός ο τύπος επιστροφής και των τριών συναρτήσεων να είναι **`void`**.

Ε.39 Ποια είναι η έξοδος του παρακάτω προγράμματος;

```
#include <stdio.h>

void test(int **tmp);

int main(void)
{
	int *ptr, arr[] = {5, 10, 15};

	ptr = arr;
	test(&ptr);

	printf("%d ", *ptr);
```

```
        return 0;
}

void test(int **tmp)
{
        int i;

        i = *(*tmp)++;
        printf("%d ", i);

        i = ++(**tmp);
        printf("%d ", i);
}
```

Ε.40 Να γραφεί ένα πρόγραμμα το οποίο να υλοποιεί το δημοφιλές παιχνίδι τρίλιζα. Για την υλοποίηση να δηλώσετε έναν διδιάστατο πίνακα (π.χ. `tr[3][3]`), τον οποίο να αρχικοποιήσετε με τον χαρακτήρα '*'. Το πρόγραμμα να προτρέπει εναλλάξ τους παίκτες να εισάγουν τη θέση (γραμμή και στήλη) του πίνακα και να τοποθετεί σε αυτή τη θέση το σύμβολο του κάθε παίκτη. Για τον πρώτο παίκτη το σύμβολο είναι ο χαρακτήρας 'X' και για τον δεύτερο το 'O'. Πριν παίξει ο παίκτης, το πρόγραμμα να εμφανίζει την τρίλιζα. Αν κάποιος παίκτης συμπληρώσει μία τριάδα με τους ίδιους χαρακτήρες σε κάποια γραμμή, στήλη ή διαγώνιο του πίνακα, τότε το πρόγραμμα πρέπει να εμφανίζει ότι αυτός ο παίκτης είναι ο νικητής και να τερματίζει. Αν δεν υπάρχει νικητής, το πρόγραμμα να εμφανίζει ανάλογο μήνυμα. Το πρόγραμμα να ελέγχει ότι η θέση που επέλεξε ο κάθε παίκτης είναι έγκυρη, καθώς και ότι είναι ελεύθερη.

Για να ελέγξει το πρόγραμμα αν ο παίκτης που παίζει κέρδισε, να καλεί μία συνάρτηση η οποία να δέχεται σαν παράμετρο τον διδιάστατο πίνακα, τη γραμμή και τη στήλη που επέλεξε ο παίκτης και αν υπάρχει μία τριάδα με τους χαρακτήρες του σε κάποια γραμμή, στήλη ή διαγώνιο του πίνακα, τότε η συνάρτηση να επιστρέφει `1`, αλλιώς `0`.

Ενδεικτικές Απαντήσεις

Ε.1 *Απάντηση:*

Αυτό το πρόγραμμα αποτελεί ένα παράδειγμα λανθασμένης χρήσης του τελεστή = αντί του τελεστή == σε **if** συνθήκες. Συγκεκριμένα, στις παραπάνω **if** συνθήκες, δεν γίνεται έλεγχος ισότητας, αλλά εκχώρηση τιμών στις αντίστοιχες μεταβλητές. Δηλαδή, στο `i` εκχωρείται το `40`, στο `j` το `50` ενώ στο `k` εκχωρείται αρχικά το `60` και μετά το `0`. Άρα, οι τρεις πρώτες **if** συνθήκες είναι αληθείς (αφού εκχωρούνται μη μηδενικές τιμές στις `i`, `j` και `k`, αντίστοιχα), ενώ η τέταρτη **if** συνθήκη είναι ψευδής (αφού εκχωρείται το `0` στη μεταβλητή `k`). Επομένως, το πρόγραμμα εμφανίζει: `One Two Three 40 50 0`

Ε.2 *Απάντηση:*

Ένα λάθος είναι ότι η εντολή `points += 3;` πρέπει να πάει πριν το **if**, ώστε να προστεθούν οι βαθμοί της τρίτης διαδοχικής απάντησης πριν το πρόγραμμα τερματίσει με την **return**. Ένα δεύτερο λάθος είναι ότι δεν μηδενίζεται η `first_ans` στην περίπτωση που ο εξεταζόμενος επιλέξει τη δεύτερη ή την τρίτη απάντηση, ώστε να αρχίσει η μέτρηση των διαδοχικών πρώτων απαντήσεων από το μηδέν. Άρα, σε αυτές τις δύο **case** περιπτώσεις πρέπει να προστεθεί η εντολή `first_ans = 0`. Ένα τρίτο λάθος είναι ότι ο αριθμός των επαναλήψεων δεν είναι 50 αλλά 51. Απλά, διαγράφουμε το = και γράφουμε `i < 50` .

Ε.3 *Απάντηση:*

Ο πρώτος βρόχος τερματίζεται όταν η συνθήκη `i && i-1` γίνει ψευδής. Αυτό θα συμβεί όταν η τιμή του `i` γίνει ίση με `1`. Άρα, ο βρόχος εκτελείται δύο φορές και το πρόγραμμα εμφανίζει `3` και `2`. Πάμε στον δεύτερο. Η συνθήκη `++i` είναι ισοδύναμη με `++i != 0`. Πότε θα γίνει `0` το `i`; Επειδή ο τύπος του είναι **unsigned char**, στην επόμενη αύξηση από το `255`. Άρα, ο δεύτερος βρόχος εμφανίζει τις τιμές του `i` από `2` μέχρι και `255`.

Ε.4 *Απάντηση:*

```
#include <stdio.h>
int main(void)
{
	int x, y, z, sum, min_sum, min_x, min_y, min_z, flag;

	flag = 1;
	min_sum = 91; /* Θέτουμε μία μέγιστη τιμή σύμφωνα με το διάστημα
των λύσεων που καθορίζεται στην εκφώνηση. */
	for(x = -30; x <= 30; x++)
```

```
                for(y = -30; y <= 30; y++)
                        for(z = -30; z <= 30; z++)
                                if(3*x + 7*y - 5*z == 10)
                                {
                                        flag = 0;
                                        printf("Solution: %d %d %d\n", x, y,
z);
                                        sum = x+y+z;
                                        if(sum < min_sum)
                                        {
                                                min_x = x;
                                                min_y = y;
                                                min_z = z;
                                                min_sum = sum;
                                        }
                                }
        if(flag)
                printf("No solution\n");
        else
                printf("Values: %d %d %d\n", min_x, min_y, min_z);
        return 0;
}
```

Ε.5 *Απάντηση:*

```
#include <stdio.h>

#define SIZE 50
int main(void)
{
        int i, j, pos, num, found, code[SIZE];

        pos = 0;
        while(pos < SIZE)
        {
                printf("Enter code: ");
                scanf("%d", &num);
                if(num == -1)
                        break;

                found = 0;
                /* Η μεταβλητή pos μετράει πόσοι κωδικοί έχουν καταχωρηθεί
στον πίνακα, οπότε ο επόμενος βρόχος ελέγχει αν ο εισαγόμενος κωδικός
υπάρχει ήδη στον πίνακα. Αν υπάρχει, η μεταβλητή found γίνεται 1 και ο
βρόχος τερματίζεται. */
                for(j = 0; j < pos; j++)
                {
                        if(code[j] == num)
                        {
                                printf("Error: Code %d exists. ", num);
                                found = 1;
                                break;
                        }
                }
```

```
                /* Αν ο κωδικός δεν υπάρχει στον πίνακα, τον αποθηκεύουμε
και αυξάνουμε τον δείκτη θέσης κατά ένα. */
                if(found == 0)
                {
                        code[pos] = num;
                        pos++;
                }
        }
        printf("\nCodes: ");
        for(i = 0; i < pos; i++)
                printf("%d ", code[i]);
        return 0;
}
```

E.6 *Απάντηση:*

```
#include <stdio.h>

#define SIZE 100

int main(void)
{
        int i, j, max, arr[SIZE], found, all_less, all_more;

        found = 0;
        for(i = 0; i < SIZE ; i++)
        {
                printf("Enter number: ");
                scanf("%d", &arr[i]);
        }
        for(i = 1; i < SIZE-1; i++)
        {
                all_less = all_more = 1;
                for(j = 0; j < i; j++)
                {
                        if(arr[j] >= arr[i])
                        {
                                all_less = 0;
                                break; /* Αφού βρήκαμε ένα στοιχείο που δεν
ικανοποιεί τη συνθήκη δεν έχει νόημα να συνεχίσουμε την επανάληψη. */
                        }
                }
                if(all_less)  /* Αφού ικανοποιείται η πρώτη απαίτηση
συνεχίζουμε με τη δεύτερη απαίτηση. */
                {
                        for(j = i+1; j < SIZE; j++)
                        {
                                if(arr[j] <= arr[i])
                                {
                                        all_more = 0;
                                        break; /* Όπως πριν, αφού βρήκαμε ένα
στοιχείο που δεν ικανοποιεί τη συνθήκη δεν έχει νόημα να συνεχίσουμε
την επανάληψη. */
                                }
                        }
```

```
			}
			if(all_less && all_more)
			{
				if(found == 0)
				{
					max = arr[i];
					found = 1;
				}
				else /* Σημαίνει ότι έχει βρεθεί και άλλο στοιχείο που ικανοποιεί τις δύο απαιτήσεις. */
				{
					if(arr[i] > max)
						max = arr[i];
				}
			}
		}
		if(found == 0)
			printf("Such element does not exist\n");
		else
			printf("The element with value %d satisfies the requirements\n", max);
		return 0;
}
```

E.7 *Απάντηση:*

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

#define SIZE 10

int main(void)
{
	int i, pos1, pos2, pos3, ans, arr[SIZE]; /* Για την αποθήκευση των τυχερών αριθμών χρησιμοποιούμε τον πίνακα arr. */

	srand(time(NULL));
	/* Πρώτη κλήρωση. */
	for(i = 0; i < SIZE; i++)
	{
		arr[i] = rand()%101;
		printf("%d ", arr[i]);
	}
	while(1)
	{
		printf("\nContinue to play? (0:No): ");
		scanf("%d", &ans);
		if(ans == 0)
			return 0;
		/* Επιλέγουμε τρεις τυχαίες θέσεις του πίνακα. Οι αριθμοί που είναι αποθηκευμένοι σε αυτές τις θέσεις θα εμφανιστούν και στην επόμενη κλήρωση. */
		pos1 = rand()%SIZE;
		do
		{
```

```
                pos2 = rand()%SIZE;
            } while(pos1 == pos2);

            do
            {
                pos3 = rand()%SIZE;
            } while((pos1 == pos3) || (pos2 == pos3));

            /* Επόμενη κλήρωση. */
            for(i = 0; i < SIZE; i++)
            {
                if((i == pos1) || (i == pos2) || (i == pos3))
                {
                    printf("%d ", arr[i]);
                    continue;
                }
                arr[i] = rand()%101;
                printf("%d ", arr[i]);
            }
        }
        return 0;
}
```

Σχόλια: Με αυτό το απλό παράδειγμα μίας «πειραγμένης» κλήρωσης θέλουμε να σας δείξουμε ότι ένα λογισμικό μπορεί να γραφεί με τέτοιο τρόπο, ώστε τα αποτελέσματα να είναι προκαθορισμένα. **Μακριά** λοιπόν από on-line στοιχηματικές εταιρείες που διαφημίζουν μεγάλα κέρδη σε ηλεκτρονικές κληρώσεις. Αυτή που σίγουρα στο τέλος θα είναι κερδισμένη, είναι η στοιχηματική εταιρεία.

Ε.8 *Απάντηση:*

Για την πρώτη σχέση, στο πρώτο κενό η απάντηση είναι `ROWS`, στο δεύτερο είναι `COLS` και η εκχώρηση είναι `b[i][j] = a[COLS*i + j];`. Για τη δεύτερη σχέση, στο πρώτο κενό η απάντηση είναι `1`, τα κενά στην επανάληψη είναι `ROWS` και `i++` και η εκχώρηση είναι `p = p * b[i][0] * b[i][1];`. Για την τρίτη σχέση, η πρώτη επανάληψη είναι `for(i = ROWS-1; i >= 0; i--)`, η δεύτερη είναι `for(j = COLS-1; j >= 0; j--)` και οι απαντήσεις για τα τρία κενά στην εμφάνιση είναι `i`, `j` και `b[i][j]`, αντίστοιχα.

Ε.9 *Απάντηση:*

```
#include <stdio.h>

#define ROWS 3 /* Αφού ο πίνακας είναι τετραγωνικός θα μπορούσαμε να
χρησιμοποιήσουμε μία σταθερά. Χρησιμοποιούμε δύο σταθερές για να είναι
πιο κατανοητό πότε κάνουμε υπολογισμούς σε γραμμή και πότε σε στήλη.
*/
#define COLS 3

int main(void)
{
```

```
	int i, j, sum, tmp, arr[ROWS][COLS];

	sum = tmp = 0;
	for(i = 0; i < ROWS; i++)
	{
		/* Βρίσκουμε τα αθροίσματα των στοιχείων των διαγωνίων. */
		for(j = 0; j < COLS; j++)
		{
			printf("Enter element arr[%d][%d]: ", i, j);
			scanf("%d", &arr[i][j]);
			if(i == j)
				sum += arr[i][j];
			if(i+j == ROWS-1) /* Ελέγχουμε αν το στοιχείο
βρίσκεται στη δευτερεύουσα διαγώνιο. */
				tmp += arr[i][j];
		}
	}
	if(sum != tmp)
	{
		printf("Not magic square -> Sum_main_diag: %d
Sum_sec_diag: %d\n", sum, tmp);
		return 0;
	}
	for(i = 0; i < ROWS; i++)
	{
		/* Αρχικοποίηση της μεταβλητής που υπολογίζει το άθροισμα
των στοιχείων της κάθε γραμμής. */
		tmp = 0;
		for(j = 0; j < COLS; j++)
			tmp += arr[i][j];

		if(sum != tmp)
		{
			printf("Not magic square -> Sum_row_%d: %d
Sum_diag: %d\n", i+1, tmp, sum);
			return 0;
		}
	}
	for(i = 0; i < COLS; i++)
	{
		tmp = 0; /* Αρχικοποίηση της μεταβλητής που υπολογίζει το
άθροισμα των στοιχείων της κάθε στήλης. */
		for(j = 0; j < ROWS; j++)
			tmp += arr[j][i];

		if(sum != tmp)
		{
			printf("Not magic square -> Sum_col_%d: %d
Sum_diag: %d\n", i+1, tmp, sum);
			return 0;
		}
	}
	printf("Magic square !!!\n");
	return 0;
}
```

Ε.10 *Απάντηση:*

Αφού ο ptr1 δείχνει στη διεύθυνση του i, το *ptr1 θα είναι ίσο με την τιμή του i. Με την εντολή ptr2 = ptr1; ο ptr2 δείχνει εκεί που δείχνει ο ptr1, δηλαδή στη διεύθυνση του i. Άρα, το *ptr2 θα είναι ίσο με την τιμή του i. Αφού και οι δύο δείκτες δείχνουν στη διεύθυνση του i, η εντολή *ptr1 = *ptr1 + *ptr2; ισοδυναμεί με i = i+i = 10+10 = 20. Παρόμοια, η εντολή *ptr2 *= 2; αναλύεται σε *ptr2 = 2*(*ptr2); που ισοδυναμεί με i = 2*i = 2*20 = 40. Τελικά, το πρόγραμμα εμφανίζει την τιμή της έκφρασης *ptr1 + *ptr2 = i+i = 40+40 = 80.

Ε.11 *Απάντηση:*

```
#include <stdio.h>
int main(void)
{
	int *p1, sum;
	float *p2, *p3, grade, max;

	p1 = &sum;
	*p1 = 0;

	p3 = &max;
	*p3 = 0;

	p2 = &grade;
	while(1)
	{
		printf("Enter grade: ");
		scanf("%f", p2);

		if(*p2 == -1)
			break;
		if(*p2 >=5 && *p2 <= 10)
		{
			(*p1)++;
			if(*p2 > *p3)
				*p3 = *p2;
		}
	}
	printf("%d students passed (max = %.2f)\n", *p1, *p3);
	return 0;
}
```

Ε.12 *Απάντηση:*

```
#include <stdio.h>
int main(void)
{
	int *p1, *p2, *p3, i, j, sum;

	p1 = &i;
	p2 = &j;
```

```
        p3 = &sum;
        *p3 = 0;

        do
        {
                printf("Enter two numbers (a < b < 100): ");
                scanf("%d%d", p1, p2);
        } while(*p1 >= *p2 || *p2 > 100);

        (*p1)++;
        while(*p1 < *p2)
        {
                *p3 += *p1;
                (*p1)++;
        }
        printf("Sum = %d\n", *p3);
        return 0;
}
```

E.13 *Απάντηση:*

```
#include <stdio.h>

#define SIZE 100

int main(void)
{
        int *p1, *p2, arr[SIZE];

        p1 = arr;
        while(p1 < arr+SIZE)
        {
                printf("Enter code_%d: ", p1-arr+1);
                scanf("%d", p1);
                if(*p1 == -1)
                        break;

                for(p2 = arr; p2 < p1; p2++) /* Ξεκινώντας από την αρχή
του πίνακα, ελέγχουμε αν ο κωδικός έχει ήδη αποθηκευτεί. */
                {
                        if(*p1 == *p2)
                        {
                                printf("Error: Code %d exists\n", *p1);
                                break;
                        }
                }
                /* Αν ο κωδικός δεν υπάρχει στον πίνακα αυξάνουμε τον
δείκτη. */
                if(p2 == p1)
                        p1++;
        }
        /* Εμφάνιση κωδικών. */
        for(p2 = arr; p2 < p1; p2++)
                printf("C: %d\n", *p2);
        return 0;
}
```

Ε.14 *Απάντηση:*

Με τη δήλωση του πίνακα `arr` η τιμή του πρώτου στοιχείου του γίνεται `20` και οι τιμές των υπολοίπων `0`. Με την εντολή `ptr = arr+1;` ο `ptr` δείχνει στο δεύτερο στοιχείο του πίνακα και κάθε φορά που αυξάνεται με την εντολή `ptr++` δείχνει στο επόμενο στοιχείο του. Σε κάθε επανάληψη του βρόχου η τιμή του τρέχοντος στοιχείου του πίνακα γίνεται ίση με το άθροισμα των τιμών του προηγούμενου στοιχείου, του επόμενου και της μονάδας. Για παράδειγμα, στην πρώτη επανάληψη η εντολή είναι ισοδύναμη με: `arr[1] = arr[0]+arr[2]+1 = 21;` Άρα, τα στοιχεία του πίνακα `arr[0]` έως και `arr[3]` θα έχουν τιμές από `20` έως και `23`, και το `arr[4]` ίση με `0`.

Ε.15 *Απάντηση:*

```
#include <stdio.h>
int main(void)
{
	int **p1, **p2, *p3, *p4, i, num, times;

	p3 = &num;
	p1 = &p3;

	p4 = &times;
	p2 = &p4;

	**p2 = 0;
	for(;;)
	{
		printf("Enter number: ");
		scanf("%d", *p1);
		if(**p1 == 0)
			break;
		if(**p1 < 0)
			**p1 = -**p1;
		for(i = 0; i < **p1; i++)
			printf("test\n");
		**p2 += **p1;
	}
	printf("Total number is = %d\n", **p2);
	return 0;
}
```

Ε.16 *Απάντηση:*

Με την εντολή `ptr = arr[1]+2;` ο `ptr` δείχνει στη διεύθυνση του `arr[1][2]`. Επομένως, το `*ptr` είναι ίσο με το `arr[1][2]`. Άρα, η εντολή `*ptr = 0` ισοδυναμεί με `arr[1][2] = 0`. Με την εντολή `ptr++`, ο `ptr` δείχνει στο επόμενο στοιχείο της συγκεκριμένης γραμμής. Για παράδειγμα, όταν αυξηθεί για πρώτη φορά θα δείχνει στη διεύθυνση του στοιχείου `arr[1][3]` και με την επόμενη αύξηση στη διεύθυνση

του arr[1][4]. Επομένως, το πρόγραμμα μηδενίζει τα τρία τελευταία στοιχεία της δεύτερης γραμμής.

Ε.17 *Απάντηση:*

```
#include <stdio.h>
int main(void)
{
	char ch, last_ch;
	int sum1, sum2;

	last_ch = sum1 = sum2 = 0;
	while(1)
	{
		printf("Enter character: ");
		scanf("%c", &ch);

		if(last_ch == ':' && ch == 'q') /* Αν ο τελευταίος χαρακτήρας που εισήγαγε ο χρήστης είναι ο ':' και αυτός που εισήγαγε τώρα είναι ο 'q', η εισαγωγή χαρακτήρων τερματίζει. */
			break;
		else if(ch >= 'a' && ch <= 'z')
			printf("%c\n", ch-32); /* Εμφάνιση αντίστοιχου κεφαλαίου χαρακτήρα. */
		else
			printf("%c\n", ch);

		last_ch = ch; /* Στη μεταβλητή last_ch αποθηκεύεται ο τελευταίος χαρακτήρας που εισήγαγε ο χρήστης. */
		if(ch == 'w')
			sum1++;
		else if(ch == 'x')
			sum2++;
		getchar();
	}
	printf("%c:%d times, %c:%d times\n", 'w', sum1, 'x',sum2);
	return 0;
}
```

Ε.18 *Απάντηση:*

Αφού τα ονόματα των πινάκων χρησιμοποιούνται σαν δείκτες, η έκφραση str1 == str2 ελέγχει αν οι δύο δείκτες δείχνουν στην ίδια διεύθυνση μνήμης, και όχι αν οι αντίστοιχοι πίνακες έχουν το ίδιο περιεχόμενο. Δείχνουν οι str1 και str2 στην ίδια θέση μνήμης;

Όχι βέβαια. Οι μεταβλητές str1 και str2 είναι πίνακες, για τους οποίους έχει δεσμευτεί διαφορετική μνήμη. Ναι μεν τα περιεχόμενα των πινάκων είναι ίδια, αλλά αποθηκεύονται σε διαφορετικές θέσεις μνήμης. Επομένως, το πρόγραμμα θα εμφανίσει Two.

Όμως, ποια θα ήταν η έξοδος του προγράμματος αν γράφαμε:

```
(*str1 == *str2) ? printf("One\n") : printf("Two\n");
```

Αφού το `str1` δείχνει στο πρώτο στοιχείο του πίνακα, το `*str1` είναι ίσο με `'t'`. Παρομοίως, το `*str2` είναι και αυτό ίσο με `'t'`. Άρα, το πρόγραμμα θα εμφάνιζε `One`.

Ε.19 *Απάντηση:*

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

#define ERROR_TRIES 7

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

int main(void)
{
	char ch, secret[30], hide[30] = {0};
	int i, found, len, error, correct;

	printf("Enter secret word: ");
	len = read_text(secret, sizeof(secret), 1);

	for(i = 0; i < len; i++)
		hide[i] = '_';

	error = correct = 0;
	while(error < ERROR_TRIES)
	{
		printf("Enter character: ");
		ch = getchar();
		found = 0;
		for(i = 0; i < len; i++)
		{
			if(secret[i] == ch)
			{
				hide[i] = ch;
				found = 1;
				correct++;
				if(correct == len)
				{
					printf("Secret word is found !!!\n");
					return 0;
				}
			}
		}
		if(found == 0)
		{
			error++;
			printf("Error, %c does not exist. You've got %d
more attempts\n", ch, ERROR_TRIES - error);
```

```
                }
                else
                        printf("%s\n", hide);
                getchar(); /* Παραλείπουμε το '\n', από την προηγούμενη
getchar(). */
        }
        printf("Sorry, the secret word was %s\n", secret);
        return 0;
}

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;
}
```

E.20 *Απάντηση:*

Αφού η `strlen()` επιστρέφει `4`, ο βρόχος θα εκτελεστεί τρεις φορές. Ας αναλύσουμε τις επαναλήψεις:

1η επανάληψη (`i = 0`): Εμφανίζεται η τιμή του `p[0]`, δηλαδή το `T`.
2η επανάληψη (`i++ = 1`): Αφού η τιμή του `p` έχει αυξηθεί κατά ένα, ο `p` δείχνει στον δεύτερο χαρακτήρα του αλφαριθμητικού που είναι ο `'e'`. Αφού χειριζόμαστε το `p` σαν πίνακα, η τιμή του `p[0]` είναι `'e'` και η τιμή του `p[1]` είναι `'x'`. Άρα, το πρόγραμμα εμφανίζει `x`.
3η επανάληψη (`i++ = 2`): Ο `p` αυξάνεται και δείχνει στον τρίτο χαρακτήρα που είναι ο `'x'`. Επομένως, η τιμή του `p[0]` είναι `'x'`, το `p[1]` είναι `'t'` και το `p[2]` ίσο με `'\0'`. Άρα, το πρόγραμμα δεν εμφανίζει τίποτα.

Τελικά, το πρόγραμμα εμφανίζει: `Tx`

Ε.21 *Απάντηση:*

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

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

int main(void)
{
	char str[100];
	int i, len, cnt;

	printf("Original text: ");
	len = read_text(str, sizeof(str), 1);

	printf("Compressed text: ");
	i = 0;
	while(i < len)
	{
		cnt = 1;
		if(str[i] < '0' || str[i] > '9') /* Σύμφωνα με την εκφώνηση να μην γίνεται συμπίεση ψηφίων. */
		{
			while(str[i+cnt] == str[i]) /* Έλεγχος αν ο συγκεκριμένος χαρακτήρας, δηλαδή ο str[i], επαναλαμβάνεται στις επόμενες θέσεις. */
				cnt++;

			if(cnt == 1)
				printf("%c", str[i]);
			else
				printf("%d%c", cnt, str[i]);
		}
		else
			printf("%c", str[i]);

		i += cnt;
	}
	return 0;
}
```

Ε.22 *Απάντηση:*

Αφού η έκφραση `*(str+3)` ισοδυναμεί με `str[3]`, δηλαδή με `'e'`, η τιμή της έκφρασης `*(str+3)-1` είναι ίση με την τιμή `str[3]-1`, δηλαδή ίση με `'e'-1 = 'd'`. Η `strlen()` επιστρέφει το μήκος του αλφαριθμητικού μετά τον τρίτο χαρακτήρα. Άρα, επιστρέφει την τιμή 4. Επομένως, η έκφραση ισοδυναμεί με `str+'d'-'a' = str+3` και το πρόγραμμα εμφανίζει `easy`.

Ε.23 *Απάντηση:*

Αφού θέλουμε η `f()` να αλλάζει την τιμή του `a`, πρέπει να της μεταβιβάσουμε τη διεύθυνση του `a`. Αφού το `a` είναι ακέραιος, η παράμετρος της `f()` είναι δείκτης σε ακέραιο (π.χ. `int *p`). Στο δεύτερο κενό, για την κλήση της `f()` γράφουμε `f(&a)`. Τέλος, στο σώμα της `f()` χρησιμοποιούμε τον δείκτη και γράφουμε `*p = *p * *p;`.

Ε.24 *Απάντηση:*

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

void find_roots(double a, double b, double c, double *r1, double *r2,
int *code);

int main(void)
{
	int code;
	double a, b, c, r1, r2;

	do
	{
		printf("Enter coefficients (a<>0): ");
		scanf("%lf%lf%lf", &a, &b, &c);
	} while(a == 0);

	find_roots(a, b, c, &r1, &r2, &code);
	if(code == 2)
		printf("Two roots: %f %f\n", r1, r2);
	else if(code == 1)
		printf("One root: %f\n", r1);
	else
		printf("Not real roots\n");
	return 0;
}

void find_roots(double a, double b, double c, double *r1, double *r2,
int *code)
{
	double d;

	d = b*b-4*a*c;
	if(d > 0)
	{
		*code = 2;
		*r1 = (-b+sqrt(d))/(2*a);
		*r2 = (-b-sqrt(d))/(2*a);
	}
	else if(d == 0)
	{
		*code = 1;
		*r1 = *r2 = -b/(2*a);
```

```
        }
        else
                *code = 0;
}
```

Σχόλια: Αφού ο τύπος επιστροφής της συνάρτησης είναι **void** χρησιμοποιούμε παραμέτρους δείκτες, ώστε η συνάρτηση να επιστρέψει τις επιθυμητές τιμές.

Ε.25 *Απάντηση:*

Όταν καλείται η `test()` έχουμε `p = arr+2 = &arr[2]` και `ptr2 = &i`. Άρα, το `p[0]` είναι ίσο με `arr[2]`, το `p[1]` ίσο με `arr[3]` και το `p[2]` ίσο με `arr[4]`. Επομένως, με την εντολή `p[2] = 200;` το `arr[4]` γίνεται 200. Αφού ο `ptr2` δείχνει στη διεύθυνση του `i`, η εντολή `p+*ptr2;` είναι ισοδύναμη με `p+i = p+1`. Αφού ο `p` δείχνει στο τρίτο στοιχείο του πίνακα `arr`, η `test()` επιστρέφει έναν δείκτη στο τέταρτο στοιχείο του. Αφού αυτός ο δείκτης αποθηκεύεται στη μεταβλητή `ptr`, η τιμή του `*ptr` θα είναι ίση με `arr[3]`. Επομένως, το πρόγραμμα εμφανίζει: `200 40`

Ε.26 *Απάντηση:*

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

int str_cmp(const char *str1, const char *str2);
int read_text(char str[], int size, int flag);

int main(void)
{
        char str1[100], str2[100];
        int i;

        printf("Enter first string: ");
        read_text(str1, sizeof(str1), 1);

        printf("Enter second string: ");
        read_text(str2, sizeof(str2), 1);

        i = str_cmp(str1, str2);
        if(i == 0)
                printf("%s = %s\n", str1, str2);
        else if(i < 0)
                printf("%s < %s\n", str1, str2);
        else
                printf("%s > %s\n", str1, str2);
        return 0;
}

int str_cmp(const char *s1, const char *s2)
{
        while(*s1 == *s2)
        {
```

```
            if(*s1 == '\0')
                  return 0;
            s1++;
            s2++;
      }
      return *s1-*s2;
}
```

Σχόλια: Όταν βρεθούν δύο ανόμοιοι χαρακτήρες ο βρόχος τερματίζεται και η συνάρτηση επιστρέφει τη διαφορά τους. Η συνάρτηση `str_cmp()` αποτελεί ένα παράδειγμα υλοποίησης της συνάρτησης βιβλιοθήκης `strcmp()`.

E.27 *Απάντηση:*

Η `test()` επιστρέφει τον δείκτη στην παράμετρο με τη μικρότερη τιμή. Στην πρώτη της κλήση, έχουμε `ptr1 = &a`, άρα `*ptr1 = a = 1.2`. Παρομοίως, `ptr2 = &b`, άρα `*ptr2 = b = 3.4`. Επομένως, η `test()` επιστρέφει τον δείκτη `ptr1`. Αφού η `test()` επιστρέφει τον δείκτη στη μεταβλητή `a`, η εντολή `*test(&a, &b) = 5.6;` κάνει το `a` ίσο με `5.6` και το πρόγραμμα εμφανίζει: `val1 = 5.6 val2 = 3.4`

Στη δεύτερη κλήση της `test()`, αφού `*ptr1 = a = 5.6` και `*ptr2 = b = 3.4`, η `test()` επιστρέφει τον δείκτη `ptr2`. Παρόμοια με πριν, αφού τώρα η `test()` επιστρέφει τον δείκτη στη μεταβλητή `b`, η τιμή του `b` γίνεται ίση με `7.8` και το πρόγραμμα εμφανίζει: `val1 = 5.6 val2 = 7.8`.

E.28 *Απάντηση:*

Η `swap()` δέχεται σαν ορίσματα τις τιμές των δεικτών, όχι τις διευθύνσεις τους. Άρα, η αντιμετάθεση δεν έχει καμία επίδραση και το πρόγραμμα εμφανίζει `Shadow Play`. Αυτό που θέλουμε να κάνετε είναι να τροποποιήσετε το πρόγραμμα, ώστε να εμφανίζει `Play Shadow`. Για να δούμε, μπορείτε; Υπόδειξη, αλλάξτε τον τύπο των παραμέτρων σε... βρείτε το.

E.29 *Απάντηση:*

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

#define CHANNELS 5
#define DAYS 7
#define SIZE 50

void read_chnl_info(char name[][SIZE], int view_num[][DAYS]);
void calc_chnl_view(const int view_num[][DAYS], float view_avg[]);
void show_chnl(const int view_num[][DAYS], const char name[][SIZE],
const float view_avg[]);
void check_day_inc(const int view_num[][DAYS], const char
name[][SIZE]);
```

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

int main(void)
{
	char name[CHANNELS][SIZE];
	int i, j, view_num[CHANNELS][DAYS];
	float view_avg[CHANNELS];

	read_chnl_info(name, view_num);
	calc_chnl_view(view_num, view_avg);
	show_chnl(view_num, name, view_avg);
	check_day_inc(view_num, name);
	return 0;
}

void read_chnl_info(char name[][SIZE], int view_num[][DAYS])
{
	int i, j;

	for(i = 0; i < CHANNELS; i++)
	{
		printf("\nEnter name of channel_%d: ", i+1);
		read_text(name[i], sizeof(name[i]), 1);

		for(j = 0; j < DAYS; j++)
		{
			printf("Enter viewers of channel_%d at day_%d: ",
i+1, j+1);
			scanf("%d", &view_num[i][j]);
		}
		getchar();
	}
}

void calc_chnl_view(const int view_num[][DAYS], float view_avg[]) /*
Για την επιστροφή των μέσων όρων μεταβιβάζουμε στη συνάρτηση τον πίνακα
view_avg και σε αυτόν θα αποθηκεύσουμε τις μέσες τηλεθεάσεις. */
{
	int i, j;
	float sum;

	for(i = 0; i < CHANNELS; i++)
	{
		sum = 0;
		for(j = 0; j < DAYS; j++)
			sum += view_num[i][j];

		view_avg[i] = sum/DAYS;
	}
}

void show_chnl(const int view_num[][DAYS], const char name[][SIZE],
const float view_avg[])
{
	int i, j, found;
```

```
	found = 0;
	for(i = 0; i < CHANNELS; i++)
	{
		if(view_avg[i] < (view_num[i][DAYS-1] + view_num[i][DAYS-
2])/2.0)
		{
			found = 1;
			printf("Average weekday viewers of channel %s are
more than the average weekend viewers\n", name[i]);
		}
	}
	if(found == 0)
		printf("No channel has more average weekday viewers than
the average weekend viewers\n");
}

void check_day_inc(const int view_num[][DAYS], const char name[][SIZE])
{
	int i, j, ch_inc, found;

	found = 0;
	for(i = 0; i < CHANNELS; i++)
	{
		ch_inc = 1;
		for(j = 0; j < DAYS-1; j++)
		{
			if(view_num[i][j] >= view_num[i][j+1])
			{
				ch_inc = 0;
				break;
			}
		}
		if(ch_inc == 1)
		{
			found = 1;
			printf("Channel %s has a daily increase in
viewers\n", name[i]);
		}
	}
	if(found == 0) /* Δεν υπάρχει σταθμός με αύξηση της τηλεθέασης
καθόλη τη διάρκεια της εβδομάδας. */
		printf("No channel has a daily increase in viewers\n");
}
```

Ε.30 *Απάντηση:*

```
#include <stdio.h>

double fact(int num);

int main(void)
{
	int num;

	do
	{
```

```
                printf("Enter a positive integer less than 170: ");
                scanf("%d", &num);
        } while(num < 0 || num > 170);

        printf("Factorial of %d is %e\n", num, fact(num));
        return 0;
}

double fact(int num)
{
        if((num == 0) || (num == 1))
                return 1;
        else
                return num * fact(num-1);
}
```

Σχόλια: Παρατηρήστε ότι για μεγάλες τιμές της num οι κλήσεις της fact() αυξάνουν σημαντικά και επομένως και ο χρόνος υπολογισμού της τελικής τιμής. Σε αυτή την περίπτωση, η λύση με τον **for** βρόχο στην αντίστοιχη άσκηση του βιβλίου υπολογίζει το παραγοντικό του αριθμού πιο γρήγορα.

E.31 *Απάντηση:*

Με την πρώτη κλήση της f() το πρόγραμμα εμφανίζει τρία One. Μετά, καλείται πάλι η f() με όρισμα 1, ενώ η printf() και η επόμενη κλήση της f() με όρισμα 1 θα γίνουν όταν τερματιστεί η προηγούμενη κλήση. Άρα, το πρόγραμμα εμφανίζει ένα One, καλείται η f() με όρισμα 0, η οποία επιστρέφει. Τώρα, καλείται η printf() που δεν είχε εκτελεστεί προηγουμένως και μετά πάλι η f() με όρισμα 0. Στη συνέχεια, καλούνται οι printf() και η f() με όρισμα 1. Το πρόγραμμα εμφανίζει ένα One και, όπως πριν, καλείται η f() με όρισμα 0, η οποία επιστρέφει. Μετά, καλείται η printf() και μετά πάλι η f() με όρισμα 0. Άρα, τελικά το πρόγραμμα εμφανίζει:

```
One One One
One
***
***
One
***
```

E.32 *Απάντηση:*

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

#define SIZE 100

struct Product
{
        char name[50];
```

```
        int code;
        float prc;
};

struct Product *find_prod(struct Product prod[], int num_prods, int
code); /* Δήλωση συνάρτησης που επιστρέφει έναν δείκτη σε δομή τύπου
Product. */
int read_text(char str[], int size, int flag);

int main(void)
{
        int i, cnt, code;
        float prc;
        struct Product *ptr, prod[SIZE];

        cnt = 0;
        for(i = 0; i < SIZE; i++)
        {
                printf("\nPrice: ");
                scanf("%f", &prc);

                if(prc > 20)
                {
                        prod[cnt].prc = prc;

                        getchar();

                        printf("Name: ");
                read_text(prod[cnt].name, sizeof(prod[cnt].name),1);

                        printf("Code: ");
                        scanf("%d", &prod[cnt].code);

                        cnt++;
                }
        }
        if(cnt == 0)
        {
                printf("None product is stored\n");
                return 0;
        }
        printf("\nEnter code to search: ");
        scanf("%d", &code);

        ptr = find_prod(prod, cnt, code);
        if(ptr == NULL)
                printf("\nNo product with code = %d\n", code);
        else
                printf("\nN:%s C:%d P:%.2f\n", ptr->name, code, ptr->prc);
        return 0;
}

struct Product *find_prod(struct Product prod[], int num_prods, int
code)
{
        int i;
```

```
	for(i = 0; i < num_prods; i++)
	{
		if(prod[i].code == code)
			return &prod[i]; /* Αν βρεθεί ο κωδικός του προϊόντος, η συνάρτηση τερματίζει άμεσα και επιστρέφει τη διεύθυνση της δομής. */
	}
	return NULL; /* Αν φτάσουμε σε αυτό το σημείο σημαίνει ότι ο κωδικός του προϊόντος δεν βρέθηκε, οπότε η συνάρτηση επιστρέφει NULL. */
}
```

Ε.33 *Απάντηση:*

```
#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. */
}
```

Ε.34 *Απάντηση:*

```
#include <stdio.h>

#define FLOORS 5
#define ROOMS 10

struct Room
{
        int free;
        int type;
};

void init_rooms(struct Room room[][ROOMS]);
void calc_beds(const struct Room room[][ROOMS], int* beds);
void same_type(const struct Room room[][ROOMS], int type, int* max_seq,
int* floor);
void all_free(const struct Room room[][ROOMS], int r_num[]);

int main(void)
{
        int i, beds, type, floor, max_seq, found, r_num[ROOMS] = {0};
        struct Room room[FLOORS][ROOMS];

        init_rooms(room);

        calc_beds(room, &beds);
        printf("\nTotal number of beds is %d\n", beds);

        printf("Enter type: ");
        scanf("%d", &type);
        same_type(room, type, &max_seq, &floor);
        if(floor == -1) /* Τιμή που υποδεικνύει ότι δεν υπάρχουν
συνεχόμενα δωμάτια του συγκεκριμένου τύπου. */
                printf("\nThere is no room sequence of type %d\n", type);
        else
                printf("\nMaximum sequence of those room types is %d in
floor %d\n", max_seq, floor+1);

        all_free(room, r_num);
        found = 0;
        for(i = 0; i < ROOMS; i++)
        {
                if(r_num[i])
                {
                        found = 1;
                        printf("Room number %d is free in all floors\n",
r_num[i]);
                }
        }
        if(found == 0)
                printf("No room number is free in all floors\n");
        return 0;
```

```
}

void init_rooms(struct Room room[][ROOMS])
{
	int i, j;

	for(i = 0; i < FLOORS; i++)
		for(j = 0; j < ROOMS; j++)
		{
			printf("\nEnter room availability (1 for free): ");
			scanf("%d", &room[i][j].free);

			printf("Enter room type: ");
			scanf("%d", &room[i][j].type);
		}
}

void calc_beds(const struct Room room[][ROOMS], int* beds) /* Αφού ο τύπος επιστροφής είναι void χρησιμοποιούμε δείκτη για την επιστροφή της τιμής. */
{
	int i, j, cnt;

	cnt = 0;
	for(i = 0; i < FLOORS; i++)
	{
		for(j = 0; j < ROOMS; j++)
		{
			if(room[i][j].type == 1)
				cnt++;
			else if(room[i][j].type == 2)
				cnt += 2;
			else
				cnt += 3;
		}
	}
	*beds = cnt;
}

void same_type(const struct Room room[][ROOMS], int type, int* max_seq, int* floor) /* Τώρα, χρησιμοποιούμε δύο δείκτες για την επιστροφή των δύο τιμών. */
{
	int i, j, cnt, seq, fl;

	seq = 1;
	fl = -1;
	for(i = 0; i < FLOORS; i++)
	{
		cnt = 0;
		for(j = 0; j < ROOMS; j++)
		{
			if(room[i][j].type == type)
				cnt++;
			else
			{
				if(cnt > seq)
```

```
                                {
                                        seq = cnt;
                                        fl = i;
                                }
                                cnt = 0;
                        }
                }
                if(cnt > seq)
                {
                        seq = cnt;
                        fl = i;
                }
        }
        *max_seq = seq;
        *floor = fl;
}

void all_free(const struct Room room[][ROOMS], int r_num[]) /* Επειδή δεν ξέρουμε πόσα είναι τα διαθέσιμα δωμάτια, μεταβιβάζουμε στη συνάρτηση έναν πίνακα για την αποθήκευση των αριθμών τους. */
{
        int i, j, cnt;

        cnt = 0;
        for(i = 0; i < ROOMS; i++)
        {
                for(j = 0; j < FLOORS; j++)
                {
                        if(room[j][i].free != 1)
                                break;
                }
                if(j == FLOORS) /* Αν ισχύει, σημαίνει ότι το δωμάτιο με τον αριθμό i+1 είναι διαθέσιμο σε όλους τους ορόφους και αποθηκεύουμε τον αριθμό στον πίνακα. */
                {
                        r_num[cnt] = i+1;
                        cnt++;
                }
        }
}
```

E.35 *Απάντηση:*

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int read_text(char str[], int size, int flag);

typedef struct
{
      char name[100];
      char capital[100];
      int pop;
} Country;

int main(void)
```

```
{
	FILE *fp;
	Country *cntr;
	char str[100];
	int i, num_cntr, pop;

	printf("Enter file name: ");
	read_text(str, sizeof(str), 1);

	fp = fopen(str, "r");
	if(fp == NULL)
	{
		printf("Error: fopen() failed\n");
		exit(EXIT_FAILURE);
	}
	if(fscanf(fp, "%d", &num_cntr) != 1)
	{
		fclose(fp);
		printf("Error: fscanf() failed\n");
		exit(EXIT_FAILURE);
	}
	/* Δυναμική δέσμευση μνήμης, για να αποθηκεύσουμε τα στοιχεία
των χωρών. */
	cntr = (Country*) malloc(num_cntr * sizeof(Country));
	if(cntr == NULL)
	{
		fclose(fp);
		printf("Error: Not available memory\n");
		exit(EXIT_FAILURE);
	}
	for(i = 0; i < num_cntr; i++)
		if(fscanf(fp,  "%s%s%d",  cntr[i].name,  cntr[i].capital,
&cntr[i].pop) != 3)
		{
			fclose(fp);
			printf("Error: fscanf() read error\n");
			exit(EXIT_FAILURE);
		}
	fclose(fp);

	printf("Enter population: ");
	scanf("%d", &pop);
	for(i = 0; i < num_cntr; i++)
		if(cntr[i].pop >= pop)
			printf("%s		%s\t%d\n",		cntr[i].name,
cntr[i].capital, cntr[i].pop);
	free(cntr);
	return 0;
}
```

E.36 *Απάντηση:*

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
```

```
	FILE *fp;
	int i, grd_num;
	float *grd_arr, grd;

	fp = fopen("test.bin", "rb");
	if(fp == NULL)
	{
		printf("Error: fopen() failed\n");
		exit(EXIT_FAILURE);
	}
	if(fread(&grd_num, sizeof(int), 1, fp) != 1)
	{
		fclose(fp);
		printf("Error: fread() failed\n");
		exit(EXIT_FAILURE);
	}
	grd_arr = (float*) malloc(grd_num * sizeof(float)); /* Δέσμευση
μνήμης για την αποθήκευση των βαθμών. */
	if(grd_arr == NULL)
	{
		fclose(fp);
		printf("Error: Not available memory\n");
		exit(EXIT_FAILURE);
	}
	/* Διάβασμα των βαθμών και έλεγχος ότι δεν συνέβη κάποιο λάθος.
*/
	if(fread(grd_arr, sizeof(float), grd_num, fp) == grd_num)
	{
		printf("Enter grade: ");
		scanf("%f", &grd);

		for(i = 0; i < grd_num; i++)
			if(grd_arr[i] > grd)
				printf("%f\n", grd_arr[i]);
	}
	else
		printf("Error: fread() failed to read grades\n");

	free(grd_arr);
	fclose(fp);
	return 0;
}
```

E.37 *Απάντηση:*

```
#include <stdio.h>

#define max(a, b) ((a) > (b) ? (a) : (b))

int main(void)
{
	int i, j, k, l;

	printf("Enter numbers: ");
	scanf("%d%d%d%d", &i, &j, &k, &l);
	printf("Max = %d\n", max(max(max(i, j), k), l));
```

```
	return 0;
}
```

Σχόλια: Ο προεπεξεργαστής μεταφράζει τις μακροεντολές ξεκινώντας από την πιο εσωτερική. Υπάρχουν και άλλοι τρόποι, για παράδειγμα, θα μπορούσαμε να γράψουμε `max(max(i, j), max(k, l));`.

Ε.38 *Απάντηση:*

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

#define SIZE 8

struct Athlete
{
	char name[50];
	int score;
};

void fill_group(struct Athlete ath[]);
void sort_group(struct Athlete ath[]);
void final_group(const struct Athlete ath_fir[], const struct Athlete
ath_sec[], struct Athlete ath_fin[]);
int read_text(char str[], int size, int flag);

int main(void)
{
	int i;
	struct Athlete ath_fir[SIZE], ath_sec[SIZE], ath_fin[SIZE];

	printf("First group\n");
	printf("------------\n");
	fill_group(ath_fir);

	printf("\nSecond group\n");
	printf("------------\n");
	fill_group(ath_sec);

	sort_group(ath_fir);
	sort_group(ath_sec);
	final_group(ath_fir, ath_sec, ath_fin);

	printf("\nFinal group\n");
	printf("------------\n");
	for(i = 0; i < SIZE; i++)
		printf("%d.%s  %d\n",  i+1,  ath_fin[i].name,  ath_fin[i].
score);

	return 0;
}

void fill_group(struct Athlete ath[])
{
```

```
	int i;

	for(i = 0; i < SIZE; i++)
	{
		printf("Name_%d: ", i+1);
		read_text(ath[i].name, sizeof(ath[i].name), 1);

		printf("Score: ");
		scanf("%d", &ath[i].score);

		getchar();
	}
}

void sort_group(struct Athlete ath[])
{
	int i, j;
	struct Athlete tmp;

	for(i = 0; i < SIZE-1; i++)
	{
		for(j = i+1; j < SIZE; j++)
		{
			if(ath[i].score < ath[j].score)
			{
				tmp = ath[i];
				ath[i] = ath[j];
				ath[j] = tmp;
			}
		}
	}
}

void final_group(const struct Athlete ath_fir[], const struct Athlete
ath_sec[], struct Athlete ath_fin[]) /* Μεταβιβάζουμε στη συνάρτηση τον
πίνακα ath_fin για την αποθήκευση της τελικής οκτάδας. */
{
	int i, j, k;

	ath_fin[0] = ath_fir[0];
	ath_fin[1] = ath_sec[0];
	i = j = 1;

	for(k = 2; k < SIZE; k++)
	{
		if(ath_fir[i].score > ath_sec[j].score)
		{
			ath_fin[k] = ath_fir[i];
			i++;
		}
		else
		{
			ath_fin[k] = ath_sec[j];
			j++;
		}
	}
```

```
}
```

Ε.39 *Απάντηση:*

Όταν καλείται η `test()` έχουμε `tmp = &ptr`, άρα το `*tmp` είναι ίσο με `ptr`. Άρα, η έκφραση `i = *(*tmp)++` είναι ισοδύναμη με `i = *ptr++;` Αυτή η εντολή πρώτα αποθηκεύει στο `i` την τιμή του `*ptr = *arr = arr[0] = 5` και μετά αυξάνει την τιμή του `ptr` ώστε να δείξει στο στοιχείο `arr[1]`. Επομένως, το πρόγραμμα εμφανίζει 5.

Στη συνέχεια, έχουμε `i = ++(**tmp) = ++(*ptr) = ++arr[1];` Τώρα, πρώτα αυξάνεται η τιμή του `arr[1]` και γίνεται 11 και μετά αυτή αποθηκεύεται στο `i`. Άρα, το πρόγραμμα εμφανίζει 11. Μετά τον τερματισμό της `test()`, αφού ο `ptr` δείχνει στο `arr[1]`, το πρόγραμμα εμφανίζει 11.

Ε.40 *Απάντηση:*

```
#include <stdio.h>

#define TIE    0
#define WIN    1

int check_win(const char tr[][3], int r, int c);

int main(void)
{
	char tr[3][3];
	int i, j, row, col, rslt, cells, play_id;

	cells = 0;
	play_id = 1;

	for(i = 0; i < 3; i++)
		for(j = 0; j < 3; j++)
			tr[i][j] = '*';

	while(1)
	{
		printf("\n");
		for(i = 0; i < 3; i++) /* Πριν παίξει ο παίκτης,
εμφανίζουμε τον πίνακα της τρίλιζας. */
		{
			for(j = 0; j < 3; j++)
				printf("%c  ", tr[i][j]);
			printf("\n");
		}
		while(1)
		{
			printf("\nPlayer_%d enter row and col: ", play_id);
			scanf("%d%d", &row, &col);

			row--; /* Αφαιρούμε 1, γιατί η αρίθμηση στους
πίνακες αρχίζει από το 0. */
			col--;
```

```
                        if(row < 0 || row > 2 || col < 0 || col > 2)
                                printf("\nError: wrong dimensions\n");
                        else if(tr[row][col] != '*')
                                printf("\nError: this cell is filled\n");
                        else
                                break;
                }
                if(play_id == 1)
                {
                        tr[row][col] = 'X';
                        play_id = 2;
                }
                else
                {
                        tr[row][col] = 'O';
                        play_id = 1;
                }
                cells++;
                if(cells > 4) /* Απλά, αφού πρώτα και οι δύο παίκτες έχουν παίξει από δύο φορές ο καθένας έχει νόημα να ελέγξουμε αν ο παίκτης κέρδισε. */
                        rslt = check_win(tr, row, col);

                if((rslt == WIN) || (cells == 9))
                        break;
        }
        if(rslt == WIN)
                printf("\nPlayer_%d won!\n", (play_id == 1) ? 2 : 1); /* Επειδή έχει αλλάξει η τιμή του play_id μετά το παίξιμο του παίκτη κάνουμε αυτόν τον έλεγχο για να εμφανίσουμε τον αριθμό του παίκτη που κέρδισε. */
        else
                printf("\nNobody wins!\n");
        return 0;
}

int check_win(const char tr[][3], int r, int c)
{
        if(tr[r][0] == tr[r][1] && tr[r][1] == tr[r][2])
                return WIN;
        else if(tr[0][c] == tr[1][c] && tr[1][c] == tr[2][c])
                return WIN;
        else if(tr[1][1] != '*' && tr[0][0] == tr[1][1] && tr[1][1] == tr[2][2])
                return WIN;
        else if(tr[1][1] != '*' && tr[0][2] == tr[1][1] && tr[1][1] == tr[2][0])
                return WIN;
        else
                return TIE;
}
```