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

C Από τη Θεωρία στην Εφαρμογή

Β΄Έκδοση

Ενδεικτικές Ασκήσεις του Βιβλίου

Ενδεικτικές Ασκήσεις Βιβλίου

Ε.1 Ποια είναι ή έξοδος του παρακάτω προγράμματος?

```
#include <stdio.h>
int main()
{
      int i, a[] = {10, 20, 30, 40, 50};
      double b[] = {2.2, 1.94, 0.5, -1, -2};

      for(i = 0; a[i] = b[i]; i++)
            printf("%d ", a[i]);
      return 0;
}
```

Ε.2 Να γραφεί ένα πρόγραμμα το οποίο να διαβάζει τους ακεραίους κωδικούς 50 προϊόντων και να τους αποθηκεύει σε έναν πίνακα. 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.3 Ποια είναι η έξοδος του παρακάτω προγράμματος;

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

      ptr1 = &i;
      ptr2 = &j;

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

Ε.4 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.

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

Ε.5 Να γραφεί ένα πρόγραμμα το οποίο να διαβάζει τους βαθμούς 10 φοιτητών, να τους αποθηκεύει σε έναν πίνακα και να εμφανίζει τη μεγαλύτερη και τη μικρότερη τιμή, καθώς και τις θέσεις τους στον πίνακα. Το πρόγραμμα να ελέγχει αν ο εισαγόμενος βαθμός ανήκει στο [0, 10]. Η διαχείριση του πίνακα να γίνει με σημειογραφία δείκτη.

Ε.6 Ποιες τιμές αποκτούν τα στοιχεία του πίνακα arr στο παρακάτω πρόγραμμα;

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

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

Ε.7 Να γραφεί ένα πρόγραμμα το οποίο να εμφανίζει όλα τα πεζά γράμματα του λατινικού αλφαβήτου σε μία γραμμή, τα κεφαλαία γράμματα σε μία δεύτερη γραμμή και τους χαρακτήρες που αντιστοιχούν στα ψηφία 0-9 σε μία τρίτη γραμμή. Να χρησιμοποιήσετε μόνο έναν **for** βρόχο.

Ε.8 Ποια είναι η έξοδος του παρακάτω προγράμματος;

```
#include <stdio.h>
int main()
{
      char str1[] = "test", str2[] = "test";
      (str1 == str2) ? printf("One\n") : printf("Two\n");
      return 0;
}
```

Ε.9 Να γραφεί ένα πρόγραμμα το οποίο να υποχρεώνει τον χρήστη να εισάγει ένα αλφαριθμητικό με περισσότερους από πέντε χαρακτήρες και λιγότερους από εκατό και να το εμφανίζει. Μην χρησιμοποιήσετε την `gets()`.

Ε.10 Ποια είναι η έξοδος του παρακάτω προγράμματος;

```
#include <stdio.h>
#include <string.h>
int main()
{
	char str[10] = "test";
	printf("%d %s\n", *strcpy(str, "n")**strcpy(str+2, "xt"),
str);
	return 0;
}
```

Ε.11 Δημιουργήστε μία συνάρτηση που να δέχεται σαν παραμέτρους δύο δείκτες σε δύο πραγματικούς αριθμούς τύπου **double** και να επιστρέφει τον δείκτη στον πραγματικό αριθμό με τη μεγαλύτερη τιμή. Να γραφεί ένα πρόγραμμα το οποίο να διαβάζει δύο πραγματικούς και να εμφανίζει τον μεγαλύτερο από αυτούς με χρήση της συνάρτησης.

Ε.12 Ποια είναι η έξοδος του παρακάτω προγράμματος;

```
#include <stdio.h>
#include <string.h>
int main()
{
	char *ptr, str[] = "Text";
	int i;

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

Ε.13 Δημιουργήστε μία συνάρτηση που να δέχεται σαν παραμέτρους έναν πίνακα που περιέχει τις τιμές των εμπορευμάτων σε ένα κατάστημα και το μέγεθός του και να επιστρέφει την ελάχιστη τιμή, τη μέγιστη και τον μέσο όρο. Να γραφεί ένα πρόγραμμα το οποίο να διαβάζει τις τιμές μέχρι 100 εμπορευμάτων και να τις αποθηκεύει σε έναν πίνακα. Η εισαγωγή των τιμών να

σταματάει αν ο χρήστης εισάγει την τιμή -1. Το πρόγραμμα να εμφανίζει την ελάχιστη τιμή, τη μέγιστη και τον μέσο όρο με χρήση της συνάρτησης.

Ε.14 Ποια είναι η έξοδος του παρακάτω προγράμματος;

```
#include <stdio.h>

void test(int *arg);

int var = 100;

int main()
{
	int *ptr, i = 30;

	ptr = &i;
	test(ptr);
	printf("Val = %d\n", *ptr);
	return 0;
}

void test(int *arg)
{
	arg = &var;
}
```

Ε.15 Ποια είναι η έξοδος του παρακάτω προγράμματος;

```
#include <stdio.h>

int a = 4;

int main()
{
	if(a == 0)
		return 0;
	else
	{
		printf("%d ", a--);
		main();
	}
	return 0;
}
```

Ε.16 Να γραφεί ένα πρόγραμμα το οποίο να διαβάζει τα ορίσματα της γραμμής εντολών, να δεσμεύει μνήμη, να τα αποθηκεύει αντεστραμμένα σε αυτήν και

στη συνέχεια να τα εμφανίζει στην οθόνη. Για παράδειγμα, αν τα ορίσματα είναι `next` και `time` να αποθηκεύονται στη μνήμη σαν `txen` και `emit`.

Ε.17 Να ορίσετε μία δομή `city` με πεδία: όνομα, χώρα, πληθυσμός. Να γραφεί ένα πρόγραμμα που να χρησιμοποιεί αυτή τη δομή για να εισάγει ο χρήστης τα στοιχεία `5` πόλεων, τα οποία να αποθηκεύονται σε έναν πίνακα τέτοιων δομών. Στη συνέχεια, το πρόγραμμα να διαβάζει το όνομα μίας χώρας και έναν αριθμό και να εμφανίζει τις πόλεις της συγκεκριμένης χώρας που έχουν μεγαλύτερο πληθυσμό από τον ακέραιο που εισήγαγε ο χρήστης.

Ε.18 Θεωρείστε ότι κάθε γραμμή του αρχείου κειμένου `students.txt` περιέχει ονόματα φοιτητών (θεωρείστε ότι κάθε όνομα έχει λιγότερους από 100 χαρακτήρες) και τους βαθμούς τους σε δύο μαθήματα με την ακόλουθη μορφή:

```
John        Morne        7        8.12
Jack        Lommi        4.50     9
Peter       Smith        2        5.75
...
```

Να γραφεί ένα πρόγραμμα το οποίο να διαβάζει κάθε γραμμή του αρχείου `students.txt` και να αποθηκεύει σε ένα δεύτερο αρχείο `suc.txt` τα στοιχεία των φοιτητών που έχουν μέσο όρο μεγαλύτερο ή ίσο του `5`, ενώ σε ένα τρίτο αρχείο `fail.txt` τα στοιχεία των φοιτητών με μέσο όρο μικρότερο του `5`. Το πρόγραμμα να εμφανίζει τον αριθμό των επιτυχόντων και αποτυχόντων φοιτητών, πριν τερματίσει.

Ε.19 Υποθέστε ότι το δυαδικό αρχείο `test.bin` περιέχει τους βαθμούς ενός φοιτητή (αποθηκευμένοι σαν **`float`**). Να γραφεί ένα πρόγραμμα το οποίο να διαβάζει τους βαθμούς από το αρχείο, στη συνέχεια να διαβάζει έναν αριθμό και να εμφανίζει τους βαθμούς με μεγαλύτερη τιμή από αυτόν.

Ε.20 Ποια είναι ή έξοδος του παρακάτω προγράμματος?

```
#include <stdio.h>

#define no_main(type, name, text,num) type name() {printf(text);
return num;}

no_main(int, main, "No main() program", 0)
```

Ενδεικτικές Απαντήσεις

Ε.1 *Απάντηση:*

Η συνθήκη ελέγχου `a[i] = b[i]` είναι ισοδύναμη με την έκφραση (`a[i] = b[i]) != 0`. Αυτή η έκφραση σημαίνει ότι οι τιμές των στοιχείων του πίνακα `b` αντιγράφονται στα αντίστοιχα στοιχεία του πίνακα `a` και γίνεται έλεγχος αν η τιμή του `a[i]` έγινε `0`. Αν γίνει `0`, ο βρόχος τερματίζεται και δεν γίνεται άλλη αντιγραφή.

Όμως, αφού ο τύπος του πίνακα `a` είναι **`int`**, ενώ ο τύπος του πίνακα `b` είναι **`double`**, μόνο το ακέραιο μέρος των στοιχείων του πίνακα `b` θα εκχωρείται στα αντίστοιχα στοιχεία του πίνακα `a`. Άρα, με την αντιγραφή της τιμής `0.5` η τιμή του `a[2]` θα γίνει `0` και ο βρόχος θα τερματιστεί.

Επομένως, το πρόγραμμα εμφανίζει: `2 1`

Ε.2 *Απάντηση:*

```
#include <stdio.h>

#define SIZE 50

int main()
{
	int i, j, num, found, code[SIZE];

	i = 0;
	while(i < SIZE)
	{
		printf("Enter code: ");
		scanf("%d", &num);

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

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

E.3 *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 της μεταβλητής `i`. Επομένως, 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*(*ptr2);` ισοδυναμεί με `i = 2*i = 2*20 = 40`, ενώ η εντολή `i += *ptr1;` με `i = i+i = 40+40 = 80`.

Αφού ο `ptr2` εξακολουθεί να δείχνει στη διεύθυνση του `i`, το πρόγραμμα εμφανίζει: `Val = 80`.

E.4 *Απάντηση:*

```
#include <stdio.h>
int main()
{
        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.5 *Απάντηση:*

```
#include <stdio.h>

#define SIZE 10

int main()
{
	int i, max_pos, min_pos;
	float max, min, arr[SIZE];

	max = 0;
	min = 10;
	max_pos = min_pos = 0;
	for(i = 0; i < SIZE; i++)
	{
		do
		{
			printf("Enter grade: ");
			scanf("%f", arr+i);
		} while(*(arr+i) > 10 || *(arr+i) < 0); /* Έλεγχος
αν ο βαθμός ανήκει στο [0,10]. */
		if(*(arr+i) > max)
		{
			max = *(arr+i);
			max_pos = i;
		}
		if(*(arr+i) < min)
		{
			min = *(arr+i);
			min_pos = i;
		}
	}
	printf("Max grade is %.2f in pos #%d\n", max, max_pos);
```

```
	printf("Min grade is %.2f in pos #%d\n", min, min_pos);
	return 0;
}
```

Ε.6 *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. Ο **`for`** βρόχος τερματίζεται όταν η τιμή του `ptr` γίνει ίση με `arr+5`, δηλαδή όταν δείξει σε διεύθυνση μνήμης μετά το τελευταίο στοιχείο του πίνακα.

Κάθε φορά που εκτελείται ο **`for`** βρόχος 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 εκτέλεση του βρόχου η εντολή:

`*ptr = *(ptr-1) + *(ptr+1) + 1;` είναι ισοδύναμη με `arr[1] = arr[0] + arr[2] + 1 = 21;`

Άρα, τα στοιχεία του πίνακα `arr[0]` έως και `arr[3]` θα αποκτήσουν τιμές από `20` έως και `23`. Ωστόσο, δεν τελειώσαμε. Τι τιμή θα έχει το `arr[4]`;

Η τιμή του `arr[4]` θα είναι ίση με το άθροισμα του `arr[3]`, της μονάδας και της τυχαίας τιμής που βρίσκεται στη διεύθυνση μνήμης αμέσως μετά το `arr[4]`.

Ε.7 *Απάντηση:*

```
#include <stdio.h>
int main()
{
	char ch, end_ch;

	end_ch = 'z';
	for(ch = 'a'; ch <= end_ch; ch++)
	{
		printf("%c ", ch);
		if(ch == 'z')
		{
```

```
                    ch = 'A'-1; /* Αφαιρούμε 1, ώστε στην επόμενη
επανάληψη του βρόχου με την εντολή ch++ να γίνει ίση με 'A'. */
                    end_ch = 'Z'; /* Αλλάζουμε τον τερματικό
χαρακτήρα, ώστε ο βρόχος να εμφανίσει τα κεφαλαία γράμματα. */
                    printf("\n");
              }
              else if(ch == 'Z')
              {
                    ch = '0'-1;
                    end_ch = '9';
                    printf("\n");
              }
        }
        return 0;
}
```

E.8 *Απάντηση:*

Τα ονόματα των πινάκων χρησιμοποιούνται σαν δείκτες. Αν λοιπόν το `str1` δείχνει εκεί που δείχνει το `str2`, τότε το πρόγραμμα εμφανίζει `One`, αλλιώς `Two`. Δείχνουν τα `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`.

Ποια θα ήταν η έξοδος του προηγούμενου 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`.

E.9 *Απάντηση:*

```
#include <stdio.h>
#include <string.h>
int main()
{
      char str[100];
      int i, ch;

      printf("Enter text (> 5 && < 100): ");
      while(1)
```

```
	{
		i = 0;
		while((ch = getchar()) != '\n' && ch != EOF)
		{
			if(i < 99)
			{
				str[i] = ch;
				i++;
			}
		}
		str[i] = '\0';
		if(strlen(str) > 5)
			break;
		else
			printf("Enter text (> 5 && < 100): ");
	}
	printf("%s\n", str);
	return 0;
}
```

E.10 *Απάντηση:*

Η πρώτη `strcpy()` αντιγραφεί τους χαρακτήρες του αλφαριθμητικού `n`, δηλαδή τους χαρακτήρες `'n'` και `'\0'`, στα στοιχεία `str[0]` και `str[1]`, αντίστοιχα, και επιστρέφει τον `str` δείκτη. Άρα, η έκφραση `*strcpy(str, "n")` μπορεί να αντικατασταθεί από την `*str`, δηλαδή το `'n'`.

Η δεύτερη `strcpy()` αντιγραφεί το αλφαριθμητικό `xt` στην τρίτη θέση του πίνακα `str` και επιστρέφει τον `str+2` δείκτη. Άρα, η έκφραση `*strcpy(str+2, "xt")` μπορεί να αντικατασταθεί από την `*(str+2)`, δηλαδή το `'x'`.

Επομένως, το πρόγραμμα εμφανίζει το γινόμενο των ASCII τιμών των `'n'` και `'x'`, που είναι ίσο με `13200`.

Ωστόσο, δεν εμφανίζει `next`, αλλά μόνο τον χαρακτήρα `n`, γιατί η πρώτη `strcpy()` αντικατέστησε το `'e'` με `'\0'`.

E.11 *Απάντηση:*

```
#include <stdio.h>

double *max(double *ptr1, double *ptr2);

int main()
```

```
{
      double *ptr, i, j;

      printf("Enter numbers: ");
      scanf("%lf%lf", &i, &j);

      ptr = max(&i, &j);
      printf("The max of %f and %f is %f\n", i, j, *ptr);
      return 0;
}

double *max(double *ptr1, double *ptr2)
{
      if(*ptr1 > *ptr2)
            return ptr1;
      else
            return ptr2;
}
```

Σχόλια: Η `max()` συγκρίνει τα περιεχόμενα των δεικτών και επιστρέφει τον δείκτη στον πραγματικό με τη μεγαλύτερη τιμή. Ο δείκτης αυτός εκχωρείται στον δείκτη `ptr` και η `printf()` εμφανίζει το περιεχόμενό του. Θα μπορούσαμε να μην δηλώσουμε τη μεταβλητή `ptr` και να γράψουμε:

```
printf("The max value of %f and %f is %f\n", i, j, *max(&i, &j));
```

E.12 ***Απάντηση:***

Μάλλον, ετοιμαστείτε για **ήττα**.

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

1η επανάληψη (`i = 0`): Εμφανίζεται η τιμή του `ptr[0]`, δηλαδή το `'T'`.

2η επανάληψη (`i++ = 1`): Η τιμή του `ptr` έχει αυξηθεί κατά ένα, άρα τώρα ο `ptr` δείχνει στον δεύτερο χαρακτήρα του αλφαριθμητικού που είναι ο `'e'`. Αφού χειριζόμαστε το `ptr` σαν πίνακα, η τιμή του `ptr[0]` είναι `'e'` και η τιμή του `ptr[1]` είναι `'x'`. Άρα, το πρόγραμμα εμφανίζει `'x'`.

3η επανάληψη (`i++ = 2`): Ο `ptr` αυξάνεται και δείχνει στον τρίτο χαρακτήρα που είναι ο `'x'`. Επομένως, η τιμή του `ptr[0]` είναι `'x'`, η τιμή του `ptr[1]` είναι `'t'` και η τιμή του `ptr[2]` είναι `'\0'`. Άρα, το πρόγραμμα δεν εμφανίζει τίποτα.

4η επανάληψη (`i++ = 3`): Τώρα, ο `ptr` δείχνει στον τέταρτο χαρακτήρα του αλφαριθμητικού που είναι ο `'t'`. Επομένως, η τιμή του `ptr[0]` είναι `'t'`, η τιμή του `ptr[1]` είναι `'\0'` και οι τιμές των `ptr[2]` και `ptr[3]` είναι οι τυχαίες τιμές που βρίσκονται στη μνήμη. Άρα, το πρόγραμμα θα εμφανίσει έναν τυχαίο χαρακτήρα.

Τελικά, το πρόγραμμα εμφανίζει: `Tx(κενό)(τυχαίος χαρακτήρας)`

Ε.13 *Απάντηση:*

```
#include <stdio.h>

void stat_arr(float arr[], int size, float *min, float *max,
float *avg);

int main()
{
	int i;
	float min, max, avg, arr[100];

	for(i = 0; i < 100; i++)
	{
		printf("Enter price: ");
		scanf("%f", &arr[i]);
		if(arr[i] == -1)
			break;
	}
	if(i == 0)
		return 0;
	/* Η μεταβλητή i δηλώνει πόσες τιμές αποθηκεύτηκαν στον
πίνακα. Π.χ. αν ο χρήστης δεν εισάγει την τιμή -1, τότε η τιμή
του i μετά το for βρόχο θα είναι ίση με 100. Αν ο χρήστης
εισάγει το -1, η τιμή του i δηλώνει τον αριθμό τους. */
	stat_arr(arr, i, &min, &max, &avg);
	printf("Max=%.2f Min=%.2f Avg=%.2f\n", max, min, avg);
	return 0;
}

void stat_arr(float arr[], int size, float *min, float *max,
float *avg)
{
	int i;
	float sum;

	sum = *min = *max = arr[0];
	for(i = 1; i < size; i++)
	{
```

```
            if(arr[i] > *max)
                  *max = arr[i];
            if(arr[i] < *min)
                  *min = arr[i];
            sum += arr[i];
      }
      *avg = sum/size;
}
```

Σχόλια: Αφού με την εντολή **`return`** μπορούμε να επιστρέψουμε το πολύ μία τιμή, μεταβιβάζουμε σαν ορίσματα δείκτες σε αντίστοιχες μεταβλητές για την αποθήκευση των τιμών.

Ε.14 *Απάντηση:*

Απαντήσατε `100`, έτσι δεν είναι; Παραδεχτείτε το, σας ξεγελάσαμε για άλλη μία φορά, ... ετοιμαστείτε για τις επόμενες.

Η `test()` δέχεται σαν όρισμα την τιμή της μεταβλητής `ptr` και όχι τη διεύθυνσή της. Άρα, οποιαδήποτε μεταβολή στην τιμή του `arg` δεν επηρεάζει την τιμή του `ptr` και το πρόγραμμα εμφανίζει: `Val = 30`

Ε.15 *Απάντηση:*

Παρατηρείστε ότι και η `main()` μπορεί να κληθεί με αναδρομικό τρόπο. Με κάθε κλήση της, η τιμή του `a` μειώνεται κατά `1`. Οι κλήσεις της `main()` θα σταματήσουν, όταν η τιμή του `a` γίνει `0`. Επομένως, το πρόγραμμα εμφανίζει: `4 3 2 1`

Ε.16 *Απάντηση:*

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

int main(int argc, char *argv[])
{
      char *rvs_str;
      int i, j, len;

      if(argc == 1)
      {
            printf("Missing string arguments ...\n");
            exit(1);
      }
```

```
	for(i = 1; i < argc; i++)
	{
		len = strlen(argv[i]);

		rvs_str = (char *) malloc(len+1); /* Δεσμεύουμε μία επιπλέον θέση για τον χαρακτήρα '\0'. */
		if(rvs_str == NULL)
		{
			printf("Error: Not available memory\n");
			exit(1);
		}
		for(j = 0; j < len; j++)
			rvs_str[j] = argv[i][len-1-j]; /* Ο τελευταίος χαρακτήρας είναι αποθηκευμένος στη θέση len-1. */
		rvs_str[j] = '\0'; /* Τερματισμός αλφαριθμητικού. */
		printf("Reverse of %s is: %s\n", argv[i], rvs_str);
		free(rvs_str);
	}
	return 0;
}
```

E.17 ***Απάντηση:***

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

#define MAX_CITIES  5

struct city
{
/* Θεωρούμε ότι τα ονόματα είναι λιγότεροι από 30 χαρακτήρες. */
	char name[30];
	char country[30];
	int pop;
};

int main()
{
	int i,pop;
	char country[30];
	struct city cities[MAX_CITIES];

	for(i = 0; i < MAX_CITIES; i++)
	{
		printf("\n***** Enter city data:\n");

		printf("City name: ");
		scanf("%s", cities[i].name);

		printf("Country name: ");
		scanf("%s", cities[i].country);
```

```
            printf("Population: ");
            scanf("%d", &cities[i].pop);
      }
      printf("\nCountry to search and population: ");
      scanf("%s%d", country, &pop);

      for(i = 0; i < MAX_CITIES; i++)
        if((cities[i].pop > pop) && (strcmp(cities[i].country,
country) == 0))
            printf("%s (%d)\n", cities[i].name, cities[i].pop);

      return 0;
}
```

E.18 *Απάντηση:*

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
      FILE *fp_in, *fp_suc, *fp_fail;
      char fnm[100], lnm[100];
      int suc_stud, fail_stud;
      double grd1, grd2;

      fp_in = fopen("students.txt", "r");
      if(fp_in == NULL)
      {
            printf("Error: File can't be loaded\n");
            exit(1);
      }
      fp_suc = fopen("suc.txt", "w");
      if(fp_suc == NULL)
      {
            printf("Error: File_1 can't be created\n");
            exit(1);
      }
      fp_fail = fopen("fail.txt", "w");
      if(fp_fail == NULL)
      {
            printf("Error: File_2 can't be created\n");
            exit(1);
      }
      suc_stud = fail_stud = 0;
      while(1)
      {
            if(fscanf(fp_in,"%s%s%lf%lf",   fnm,   lnm,   &grd1,
&grd2) != 4)
                  break;
```

```
            if((grd1+grd2)/2 >= 5)
            {
                  fprintf(fp_suc,"%s  %s  %f  %f\n",  fnm,  lnm,
grd1, grd2);
                  suc_stud++;
            }
            else
            {
                  fprintf(fp_fail,"%s  %s  %f  %f\n",  fnm,  lnm,
grd1, grd2);
                  fail_stud++;
            }
      }
      printf("Failed: %d Succeeded: %d\n", fail_stud, suc_stud);
      fclose(fp_suc);
      fclose(fp_fail);
      fclose(fp_in);
      return 0;
}
```

Ε.19 *Απάντηση:*

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
      FILE *fp;
      int i, grd_num;
      float grd, *grd_arr;

      fp = fopen("test.bin", "rb");
      if(fp == NULL)
      {
            printf("Error: File can't be loaded\n");
            exit(1);
      }
      fseek(fp, 0, SEEK_END);
      grd_num = ftell(fp) / sizeof(float); /* Αφού ο δείκτης
θέσης του αρχείου είναι στο τέλος του με την ftell()
υπολογίζουμε το μέγεθος του αρχείου σε οκτάδες. Αφού κάθε βαθμός
αποθηκεύεται σαν float, το αποτέλεσμα της διαίρεσής τους είναι
το πλήθος των βαθμών που είναι αποθηκευμένοι στο αρχείο. */
      fseek(fp, 0, SEEK_SET);
      grd_arr = (float *) malloc(grd_num * sizeof(float)); /*
Δέσμευση μνήμης για την αποθήκευση των βαθμών. */
      if(grd_arr == NULL)
      {
            printf("Error: Not available memory\n");
            exit(1);
      }
```

```
	/* 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\n");

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

Ε.20 *Απάντηση:*

Και η καλύτερη για το τέλος. Παράνοια έτσι; Τηλέφωνο στη γραμμή εξαφάνισης, χάσαμε την `main()`.

Και όμως, το πρόγραμμα τρέχει. Ο προεπεξεργαστής αντικαθιστά το `type` με **`int`**, το `name` με `main`, το `text` με `"No main() program"` και το `num` με `0`. Επομένως, ο προεπεξεργαστής μεταφράζει την `no_main()` σε:

```
int main() {printf("No main() program"); return 0;}
```

οπότε, το πρόγραμμα εκτελείται κανονικά και εμφανίζει:

```
No main() program
```

Τι θα γινόταν αν το πρώτο όρισμα της μακροεντολής ήταν **`void`** αντί για **`int`**;

Ο μεταγλωττιστής θα εμφάνιζε μήνυμα λάθους, αφού μία **`void`** συνάρτηση δεν επιτρέπεται να επιστρέφει τιμή.