

Η γλώσσα C σε βάθος

3η βελτιωμένη έκδοση

Πλήρης οδηγός εκμάθησης της γλώσσας C
με εκ

Απαντήσεις ασκήσεων

*Απευθύνεται τόσο στο νέο σπουδαστή όσο και
στον έμπειρο προγραμματιστή, με πλήθος από
χαρακτηριστικά παραδείγματα, επεξηγηματικά
σχήματα και ασκήσεις*

Νίκος Μ. Χατζηγιαννάκης



Περιέχει
δωρεάν CD-ROM

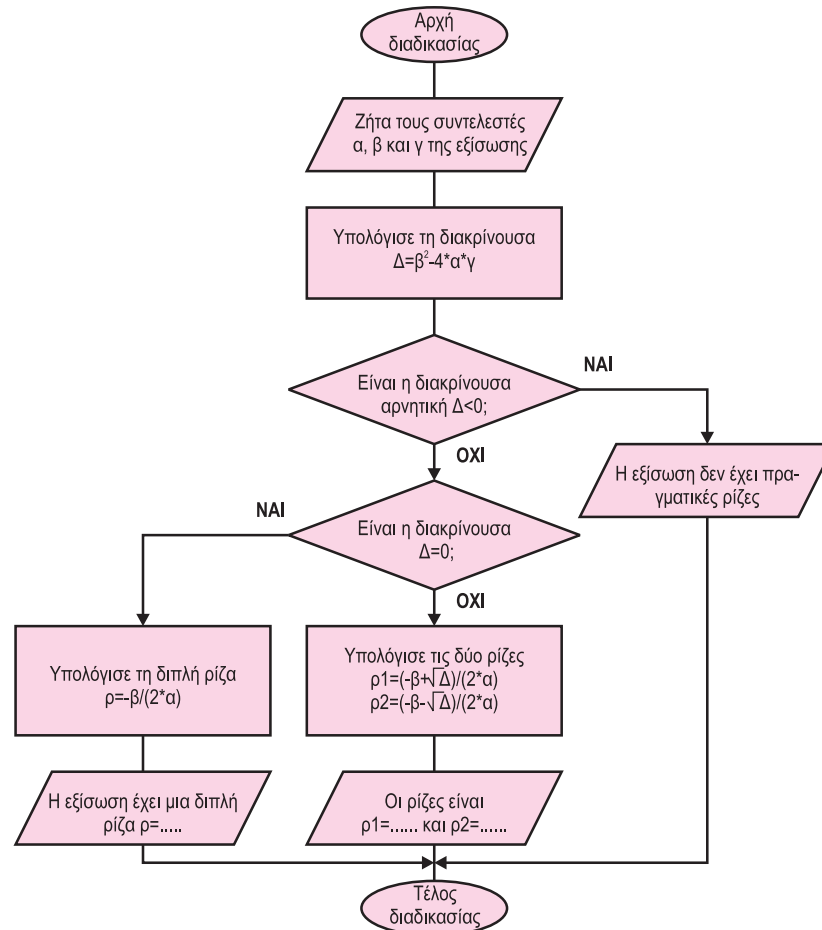
Προλογίζει ο τακτικός καθηγητής του τμήματος
Επιστήμης Υπολογιστών του Πανεπιστημίου Κρήτης
Δρ. Πάνος Τραχανιάς



Ασκήσεις Κεφαλαίου 1

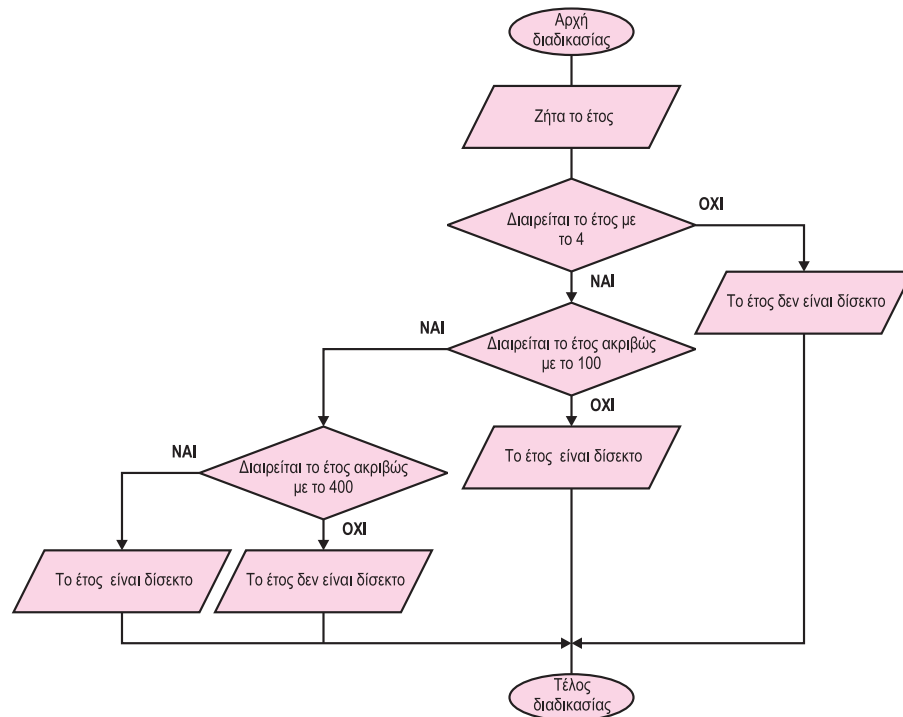
1.1

Νοίκεστράτος έχει ένα λογικό πρόγραμμα που υπολογίζει τις ρίζες της εξίσωσης $ax^2 + bx + c = 0$ της άσκησης μιας εξίσωσης δευτέρου βαθμού. ❖❖❖



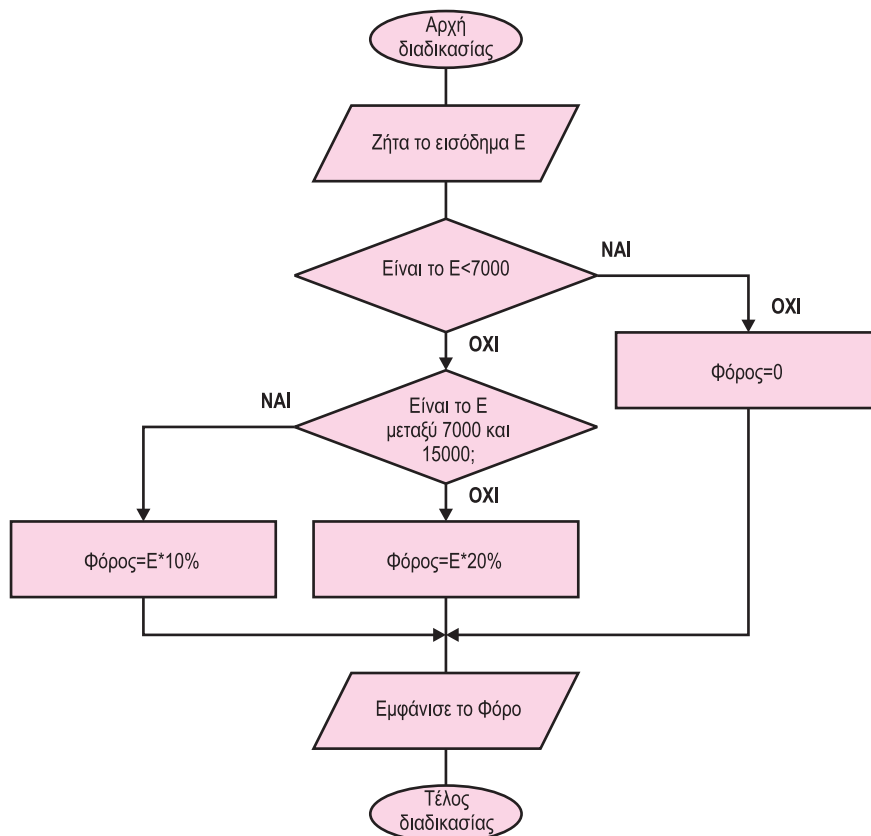
1.2

Αν γνωρίζουμε ότι: «Ο δίσκος είναι δίσκος έτος όταν διαιρείται ακριβώς με το 4 (ή το έτος που διαιρείται ακριβώς με το 100 δεν είναι δίσκος αν είναι διαιρετό με το 400), τότε ο αλγόριθμος στο παρακάτω διάγραμμα μπορεί να καθορίσει αν ένα έτος είναι δίσκος ή όχι. Να ζωγραφίσετε το έτος και να απαντήσετε αν είναι δίσκος ή όχι».



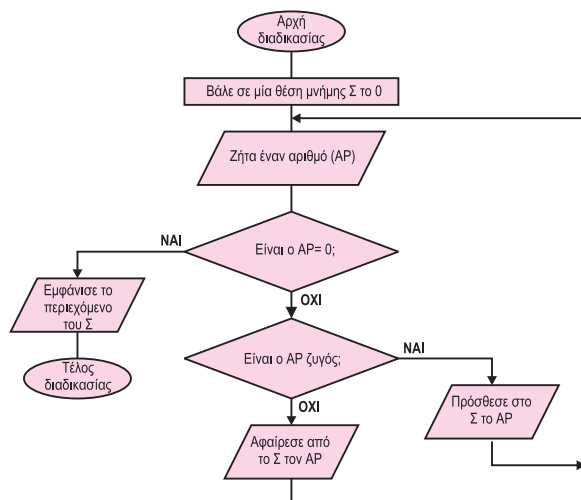
1.3

Αν κάποιος έχει εισόδημα (E) που είναι μικρότερο από 7000€ τότε ο φόρος στην περιουσία που έχει εισόδημα είναι 0%, ενώ αν είναι μεταξύ 7000€ και 15000€ τότε ο φόρος εισόδημα είναι 10%, ενώ αν είναι μεγαλύτερο από 15000€ τότε ο φόρος εισόδημα είναι 20%. Να γραφτεί ο αλγόριθμος που υπολογίζει το φόρο. *



1.4

Γράψτε το πρόγραμμα λογικό διάγραμμα. Η ακολουθία θα έχει 6 διατάξεις στην διαδοχή με τη σειρά τους αριθμούς 12, 3, 10, 7, 1, 4 και 0. *



Θα εμφανίσει το 15.

Στο Σ προσθέτει όλους τους ζυγούς 12, 10 & 4 και αφαιρεί όλους τους μονούς 3, 7 & 1. Οπότε το τελικό περιεχόμενο του Σ θα είναι:

$$12 - 3 + 10 - 7 - 1 + 4 = 15$$

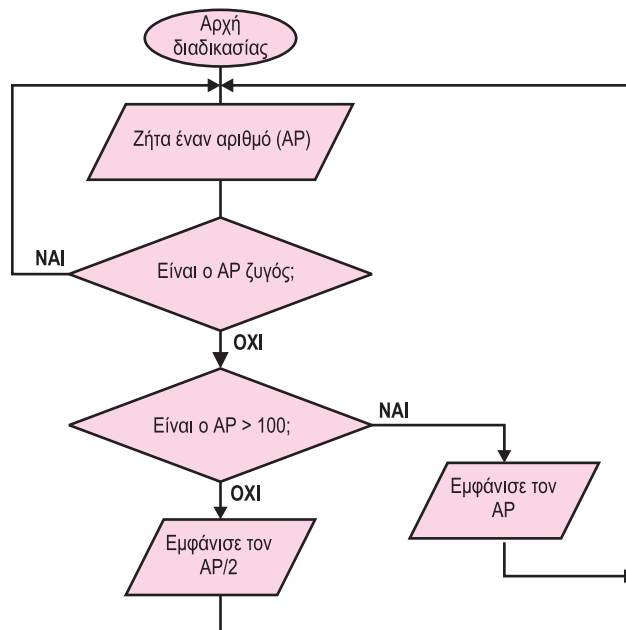
Μόλις δοθεί ο αριθμός 0, θα εμφανίσει το 15 και θα σταματήσει.

1.5 Ποια από τα παρακάτω αληθεύουν: ★

- ☐ Η C είναι μια γλώσσα με αυστηρό έλεγχο.
- ☐ Η C συναντάται συνήθως σε ερμηνευτική μορφή.
- ☐ Σε μια μεταβλητή δεν μπορούμε να αλλάξουμε το όνομά της.
- ☒ Οι τύποι δεδομένων μπορεί να διαφέρουν σε διαφορετικές γλώσσες προγραμματισμού.
- ☐ Το λογικό διάγραμμα εξαρτάται από τη γλώσσα προγραμματισμού που χρησιμοποιούμε.

1.6

Μελετήστε το παρακάτω λογικό διάγραμμα. Τι αποτέλεσμα θα έχει η διαδικασία αν δώσουμε τον αριθμό 12, 15, 145 και πάλι τον ίδιο αριθμό ξανά; Ποιες θα σταματήσει η διαδικασία; ★★



Αν δώσουμε το 12 δεν κάνει τίποτα (διότι είναι ζυγός) και συνεχίζει να περιμένει τον επόμενο. Αν δώσουμε το 15 εμφανίζει το μισό του (7.5) ενώ αν δώσουμε το 145 εμφανίζει τον ίδιο (145).

Η διαδικασία δεν σταματάει ποτέ, και έχει κάποιο αποτέλεσμα μόνο όταν δίνουμε μονούς αριθμούς.

Ασκήσεις Κεφαλαίου 2

2.1 Ένα πρόγραμμα ορίζεται με τον παρακάτω κώδικα:

```
main()
{
    int a,b,c=3;
    a=b=2;
    a=c+b;
}
```

Μεταβλητή	Τιμή
a	5
b	2
c	3

2.2 Ένα πρόγραμμα ορίζεται με τον παρακάτω κώδικα:

```
#define MM 23
main()
{
    const int c=3;
    int a,b;
    a=4+(b=2);
    b=c+b+MM;
}
```

Μεταβλητή	Τιμή
a	6
b	28
c	3

2.3 Αναφέρετε τα λάθη στον παρακάτω κώδικα:

```
#define MM 23;
main()
{
    const int c=3;
    int a,b;
    a=2;
    float d;
    d=4.3
    a=4+(b=2);
    MM=10;
    3=a;
    c=c+b+MM;
}
```

Η οδηγίες δεν τερματίζονται με ερωτηματικό (;).

Δεν επιτρέπεται δηλωτική πρόταση (float ...) μετά από εκτέλεση.

Δεν τερματίζεται με ερωτηματικό (;).

Η MM δεν είναι μεταβλητή. Δεν μπορεί να της ανατεθεί τιμή.

Το 3 δεν είναι μεταβλητή.

Στη c δεν μπορεί να ανατεθεί τιμή διότι έχει δηλωθεί ως const (μόνο ανάγνωσης).

2.4

Ένα πρόγραμμα σε μεταβλητές **a**, **b** και **c** μετά το τέλος του πρώτου τμήματος *

```
main()
```

```
{
```

```
    int a,b,c=3;
```

```
    a=b=2;
```

```
    a=c>b;
```

```
    b=b==1;
```

```
    c=printf("τέλος");
```

```
}
```

Οι **a** και **b** θα πάρουν την τιμή 2

Η **a** θα πάρει τιμή 1 διότι η παράσταση **c>b** ($3>2$) είναι αληθής.

Η **b** θα πάρει τιμή 0 διότι το **b==1** ($2==1$) είναι ψευδές.

Η **c** θα πάρει την τιμή 5, διότι η **printf()** επιστρέφει σαν τιμή το πλήθος των χαρακτήρων (5 έχει η λέξη "τέλος") που εμφανίζει στην οθόνη.

Μεταβλητή	Τιμή
a	1
b	0
c	5

2.5 Ποια από τα παρακάτω αληθεύουν: ★

- ☐ Δηλωτικές προτάσεις μπορούν να μπουν σε οποιοδήποτε σημείο του προγράμματος.
- ☒ Ένα πρόγραμμα της C μπορεί να περιέχει πολλά υποπρογράμματα (συναρτήσεις).
- ☒ Μια λογική παράσταση έχει τιμή 1 ή 0.
- ☐ Μια μεταβλητή στη C, πριν της δοθεί τιμή, έχει τιμή 0.
- ☒ Η οδηγία **#define** χρησιμοποιείται για να ορίσει μία σταθερά του προγράμματος μας.

2.6

Με δεδομένες τις τιμές των μεταβλητών **a**, **b** και **c** σε 5, 10 και 15 αντίστοιχα, συμπληρώστε την τιμή 0 ή αληθής, 1 ή ψευδής των παρακάτω λογικών παραστάσεων: ★ ★

Λογική παράσταση	Τιμή
a==(c-b)	1
a>b b>c	0
a==5 && c==15	1
a==5 && c>20	0

2.7

Θέλω να φτιάξω ένα πρόγραμμα να υπολογίζει το μέσο όρο των 3, 7, και 21 σε τρεις θέσεις μνήμης. Καταπνίξτε να υπολογίσει και να επιστρέφει σε μια ερώτηση το μέσο όρο τους: ★ ★

```
main()
```

```
{
```

```
    int a,b,c;
```

```
    float mo;
```

```
    a=3;
```

```
    b=7;
```

```
    c=21;
```

```
    mo=(a+b+c)/3.0;
```

```
}
```

2.8

Δεν επιτρέπονται οι ελληνικοί χαρακτήρες στα ονόματα των μεταβλητών, σταθερών, αρχείων, πομπών, κλπ.

```
main()
{
    int a,b,c;
    float mo;
    a=rand();
    b=rand();
    c=rand();
    mo=(a+b+c)/3.0;
}
```

2.9

Εντοπίστε το σφάλμα ή τα λάθη στον παρακάτω κώδικα.

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

#define ONE 12
#define TWO 78
#define ΤΕΣΣΕΡΑ 4

main()
{
    int c=3,a,b,γ;
    float c=5.6;
    b= ONE + TWO;
    a=printf("Η γλώσσα C σε βάθος");
    printf('Τέλος')
}
```

Το όνομα του αρχείου πρέπει να περικλείεται σε <.>

Δεν επιτρέπονται ονόματα σταθερών στα Ελληνικά

Δεν επιτρέπονται ονόματα μεταβλητών στα Ελληνικά

Η μεταβλητή **c** έχει ξαναδηλωθεί. Δεν μπορούμε να έχουμε δύο μεταβλητές με το ίδιο όνομα!

Η συμβολοσειρά **Τέλος** έπρεπε να περικλείεται σε διπλά εισαγωγικά: "Τέλος".

Ασκήσεις Κεφαλαίου 3

3.1

Να γραφτεί πρόγραμμα το οποίο να ζητήσει τρεις πραγματικούς αριθμούς από τον χρήστη, και να εμφανίσει τον μέσο όρό τους. * * *

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

main()
{
    float a,b,c,mo;
    printf("Δώσε τρεις αριθμούς:");
    scanf("%f %f %f",&a,&b,&c);
    mo=(a+b+c)/3;
    printf("Ο μέσος όρος είναι %f\n",mo);
}
```

3.2

Να γραφτεί πρόγραμμα το οποίο να επόμενα προγράμματα. *

```
main()
{
    int a=4,b=5;
    char ch;
    ch='A';
    printf("%d %d %c",a,b,ch);
    printf("%d %d %d\n",a,b,ch);
    printf("%d\n%d \n%c\n",a,b,ch);
    printf("Τέλος\n");
}
```

4 5 A 45 65
4
5
A
Τέλος

☞ Μετά το τέλος της πρώτης `printf()` δεν γίνεται αλλαγή γραμμής, οπότε τα αποτελέσματα της δεύτερης `printf()` εμφανίζονται στην ίδια γραμμή με τα αποτελέσματα της πρώτης

☞ Το 65 είναι ο ASCII κωδικός του χαρακτήρα 'A' (λατινικό).

3.3

Να γραφτεί πρόγραμμα το οποίο να ζητήσει τρεις αριθμούς και να υπολογίσει το άθροισμά, το γινόμενο, και το μέσο όρό τους. Το πρόγραμμα να μην γίνει επανάληψη για το α, να γίνει να υπολογιστεί και να γίνει το άθροισμα, γινόμενο, μέσος όρος, άθροισμα, γινόμενο, μέσος όρος. * * *

Δώσε τον πρώτο αριθμό: 6
Δώσε το δεύτερο αριθμό: 2
Δώσε τον τρίτο αριθμό: 10
Το άθροισμα των 6,2,10 είναι 18
Το γινόμενο των 6,2,10 είναι 120
Ο μέσος όρος των 6,2,10 είναι 6

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

main()
{
    float a,b,c,mo,gin,sum;
    printf("Δώσε τον πρώτο αριθμό:");
```

```
scanf("%f",&a);
printf("Δώσε τον δεύτερο αριθμό:");
scanf("%f",&b);
printf("Δώσε τον τρίτο αριθμό:");
scanf("%f",&c);
mo=(a+b+c)/3;
gin=a*b*c;
sum=a+b+c;
printf("Το άθροισμα των %f, %f, %f είναι %f\n",a,b,c,sum);
printf("Το γινόμενο των %f, %f, %f είναι %f\n",a,b,c,gin);
printf("Ο μέσος όρος των %f, %f, %f είναι %f\n",a,b,c,mo);
}
```

3.4 Πόσοτε έργο θα έχει το πρόγραμμα παραδείγματος 3.4

```
main()
{
    int a,b;
    float f;
    char ch;
    printf("%d %d %d\n",sizeof a,sizeof f,sizeof ch);
    scanf("%d %f %c",&a,&f,&ch);
    printf("%d %f %c\n",a,f,ch);
}
```

- ☞ Η πρώτη **printf()** εμφανίζει τα μεγέθη (σε bytes) των μεταβλητών **a**, **f** και **ch** δηλαδή το 4, το 4 και το 1 αντίστοιχα.
- ☞ Η **scanf()** ζητάει να πληκτρολογηθούν τρεις τιμές (δύο αριθμοί και ένας χαρακτήρας) από το πληκτρολόγιο και τις καταχωρίζει στις μεταβλητές **a**, **f** και **ch** αντίστοιχα.
- ☞ Η τελευταία **printf()** εμφανίζει τα περιεχόμενα των μεταβλητών **a**, **f** και **ch**.

3.5 Πόσοτε έργο θα λειτουργήσει το επόμενο πρόγραμμα 3.5

```
main()
{
    int a,b;
    scanf("%d %d",&a,&b);
    if(a>b)
        printf("%d\n",a);
    else
        printf("%d\n",b);
}
```

- ☞ Η **scanf()** ζητάει να πληκτρολογηθούν δύο αριθμοί από το πληκτρολόγιο και τους καταχωρίζει στις μεταβλητές **a**, και **b** αντίστοιχα.
- ☞ Η **if** ελέγχει αν η τιμή της μεταβλητής **a** είναι μεγαλύτερη από την τιμή της μεταβλητής **b**. Αν είναι, εμφανίζει την τιμή της **a** διαφορετικά την τιμή της **b**. Σε κάθε περίπτωση δηλαδή εμφανίζει τον μεγαλύτερο από τους δύο αριθμούς που δώσαμε.

3.6

Να γραφεί πρόγραμμα το οποίο να ειστάει τρεις ακέραιους αριθμούς και να εμφανίζει το μεγαλύτερο από αυτά. * *

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

main()
{
    int a,b,c,max;
    scanf("%d %d %d",&a,&b,&c);
    if(a>b)
        max=a;
    else
        max=b;
    if(max>c)
        printf("%d\n",max);
    else
        printf("%d\n",c);
}
```

Με την if στη **max** καταχωρίζεται ο μεγαλύτερος αριθμός μεταξύ των **a** και **b**.

Η δεύτερη if συγκρίνει την τιμή της **max** με την τιμή της **c** και εμφανίζει τη μεγαλύτερη. Η τιμή αυτή είναι η μεγαλύτερη από τους τρεις αριθμούς που δόθηκαν.

3.7 Ποια από τα παρακάτω αληθεύουν: ★

- ☒ Η **scanf()** χρειάζεται τις διευθύνσεις των μεταβλητών στις οποίες θα καταχωρίσει τα δεδομένα που θα πληκτρολογήσουμε.
- ☐ Το μέγεθος ενός τύπου δεδομένων στη C είναι πάντα το ίδιο και ανεξάρτητο από το σύστημα στο οποίο δουλεύουμε.
- ☒ Ο έλεγχος **if(a=5)** είναι πάντα αληθής.
- ☐ Αν μέσα σε ένα πρόγραμμα δεν υπάρχει κλήση της **exit()**, το πρόγραμμα δεν θα τερματιστεί ποτέ.
- ☒ Ο τελεστής **sizeof** μπορεί να εφαρμοστεί και σε τύπο δεδομένων π.χ. **sizeof(char)**.

3.8

Να γραφεί πρόγραμμα το οποίο να ειστάει πέντε ακέραιους αριθμούς και να εμφανίζει το άθροισμά τους. * *

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

main()
{
    int a,b;
    scanf("%d,%d",&a,&b);
    printf("%d\n",a+b);
}
```

Το , και το * ανάμεσα στα δύο %d αναγκάζουν την **scanf()** μετά τον πρώτο αριθμό να περιμένει να διαβάσει ένα κόμμα και ένα αστεράκι πριν να διαβάσει τον δεύτερο αριθμό.

3.9

Να γραφεί πρόγραμμα το οποίο θα διαβάσει την ακτίνα **r** ενός κύκλου και θα εμφανίζει την περιμέτρου και το εμβαδόν του (περίμετρος = 2 * π * **r**, εμβαδόν = π * **r**²). Το πρόγραμμα να χρησιμοποιεί το κρηστήρισμα εισόδου δεδομένων με τη μορφή "Enter radius: ". Η τιμή του π να ορίζεται ως σταθερά με τιμή 3.141592. * *

```
#include <stdio.h>
#include <stdlib.h>
#define pi 3.141592

main()
{
    float aktina,per,emvado;
    printf("Δώσε ακτίνα:");
    scanf("%f",&aktina);
    per=2*pi*aktina;
    emvado=pi*aktina*aktina;
    printf("Κύκλος με ακτίνα %f έχει περίμετρο %f και εμβαδό\n",aktina,per,emvado);
}
```

3.10 Να γραφτεί ένα πρόγραμμα στο οποίο θα ζητάει να δοθεί ο ακριβής αριθμός του ποσού και ο ποσοστό ΦΠΑ από όπου θα ληφθεί. Το πρόγραμμα θα υπολογίσει και θα εκτυπώσει το τελικό κόστος (ποσό + ΦΠΑ).

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

main()
{
    float poso,fpa,synolo;
    printf("Δώσε ποσό :");
    scanf("%f",&poso);
    printf("Δώσε ποσοστό ΦΠΑ :");
    scanf("%f",&fpa);
    synolo=poso+poso*fpa;
    printf("Το τελικό κόστος είναι: %f\n",synolo);
}
```

3.11 Το ερώτημα είναι ποιο είναι το άθροισμα από τον αριθμό 1 έως τον αριθμό 100. Αλλά ερώτημα και η γκερμανική απόδοση. Κάθε γερμανόφωνος δηλαδή 1 κοπιάει 0.23€, καθήκονας 100 70€ και καθήκονας 100 1€. Να γραφτεί ένα πρόγραμμα στο οποίο θα ζητάει να δοθεί ο αριθμός και το είδος των γερμανόφωνων που θέλουμε να αγοράσουμε και θα υπολογίσει το ποσό που πρέπει να πληρώσουμε.

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

main()
{
    int plithos;
    float poso=0;
    char eidos;
    printf("Δώσε πλήθος και είδος :");
    scanf("%d %c",&plithos,&eidos);
    if (eidos=='E') poso=plithos*0.23;
```

```

        if (eidos=='A') poso=plithos*0.70;
        if (eidos=='T') poso=plithos*0.15;
        printf("Το τελικό ποσό είναι %f\n",poso);
    }

```

3.12 Ο Δεσφής Μάκος Διανοστής (ΔΜΔ) μπορεί να επιβεβαιώσει τον τύπο B το βρέχει σε κενό και X το άλλο, σε μέτρο. Αν αγοράσει το όχημα το οποίο θα επιλεγεί να πωληθεί, οφείλει να πληρώσει το ποσό που ορίζει το πρόγραμμα. Το πρόγραμμα υπολογίζει το ποσό που οφείλει να πληρώσει ο ΔΜΔ. Το πρόγραμμα υπολογίζει το ποσό που οφείλει να πληρώσει ο ΔΜΔ, και το ποσό που οφείλει να πληρώσει ο ΔΜΔ. ***

```

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

main()
{
    float y,b,dms;
    printf("Δώσε ύψος και βάρος :");
    scanf("%f %f",&y,&b);
    dms = b/(y*y);
    printf("Ο ΔΜΔ με ύψος %f και βάρος %f είναι: %f\n",y,b,dms);
}

```

3.13 Να γραφεί πρόγραμμα το οποίο να διαβάζει τον αριθμό των λίτρων βενζίνης που βρέθηκε σε ένα αυτοκίνητο, και το ποσό σε ευρώ που πληρώθηκε. Το πρόγραμμα να υπολογίζει τη τιμή του λίτρου και να εμφανίζει τη φράση "Ακριβή βενζίνη", στην περίπτωση που η τιμή του λίτρου είναι περισσότερο από 1.3€, διαφορετικά να εμφανίζει τη φράση "Φτηνή βενζίνη". ***

```

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

main()
{
    float litra,poso,timi_ana_litro;
    printf("Δώσε αριθμό λίτρων :");
    scanf("%f",&litra);
    printf("Δώσε ποσό :");
    scanf("%f",&poso);
    timi_ana_litro=poso/litra;
    if (timi_ana_litro>1.3)
        printf("Ακριβή βενζίνη\n");
    else
        printf("Φτηνή βενζίνη\n");
}

```

Υπολογισμός της τιμής ανά λίτρο

Έλεγχος της τιμής του λίτρου

3.14 Να γραφεί πρόγραμμα το οποίο να διαβάζει από τον χρήστη τον αριθμό και να υπολογίζει το ποσό που οφείλει να πληρώσει ο χρήστης. Το ποσό που οφείλει να πληρώσει ο χρήστης να υπολογιστεί με βάση τον αριθμό που ορίζει το πρόγραμμα. Το πρόγραμμα να υπολογίζει το ποσό που οφείλει να πληρώσει ο χρήστης, και το ποσό που οφείλει να πληρώσει ο χρήστης. ***

```

#include <stdio.h>

```

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

```
main()
```

```
{
```

```
    int a,b,temp;
```

```
    printf("Δώσε δύο αριθμούς :");
```

```
    scanf("%d %d",&a,&b);
```

```
    printf("a=%d b=%d\n",a,b);
```

```
    temp=a;
```

```
    a=b;
```

```
    b=temp;
```

```
    printf("a=%d b=%d\n",a,b);
```

```
}
```

Διαβάζει δύο ακέραιους από το πληκτρολόγιο

Αντιμεταθέτει τα περιεχόμενα των μεταβλητών **a** και **b**.

Εμφανίζει τα περιεχόμενα των μεταβλητών **a** και **b**.

Ασκήσεις Κεφαλαίου 4

4.1

Στο επόμενο, τετραγωνικό πίνακα υποθέτουμε ότι το **x** έχει τιμή 100 πριν από την εκτέλεση κάθε παράστασης. Στον πίνακα, δίνουμε την τιμή του **x** και την τιμή της παράστασης μετά από την εκτέλεση της κάθε πρότασης. *

Πρόταση	Τιμή του x	Τιμή της παράστασης
x++;	101	100
++x;	101	101
x--;	99	100
--x;	99	99
x-x;	100	0

4.2

Το ακόλουθο πρόγραμμα έχει το ακόλουθο πρόγραμμα. *

```
main()
{
    int a,b,aa,bb,x,y;
    x = y = 100;
    a = ++x;
    b = y++;
    aa = ++x;
    bb = y++;
    printf("Η τιμή του a είναι %d\n",a);
    printf("Η τιμή του b είναι %d\n",b);
    printf("Η τιμή του aa είναι %d\n",aa);
    printf("Η τιμή του bb είναι %d\n",bb);
}
```

Η τιμή του a είναι 101
Η τιμή του b είναι 100
Η τιμή του aa είναι 102
Η τιμή του bb είναι 101

- ☞ Στη πρόταση **a=++x** η **x** θα αυξηθεί κατά 1 (101) και η **a** θα πάρει σαν τιμή την τιμή της παράστασης **++x** που είναι η **νέα** τιμή του **x** (101).
- ☞ Στη πρόταση **b=y++** η **y** θα αυξηθεί κατά 1 (101) και η **b** θα πάρει σαν τιμή την τιμή της παράστασης **y++** που είναι η τιμή του **y** πριν την αύξηση (100).
- ☞ Παρόμοια, στη πρόταση **aa=++x** η **aa** θα πάρει σαν τιμή την τιμή της παράστασης **++x** που είναι η **νέα** τιμή του **x** (102).
- ☞ Στη πρόταση **bb=y++** η **bb** θα πάρει σαν τιμή την τιμή της παράστασης **y++** που είναι η τιμή του **y** πριν την αύξηση (101 όπως είχε γίνει από την προηγούμενη **y++**).

4.3

Στο επόμενο, ορί η τιμή του **y** είναι 100 πριν από την εκτέλεση κάθε μιας από τις επόμενες παράστασεις. Πάντα, θα είναι οι τιμές που καταβλήθηκαν **x** και **y** με την εκτέλεση κάθε πρότασης. *

Παράσταση	Τιμή του x	Τιμή του y
x=y;	100	100
x = --y * 4;	396 (99*4)	99
x = y = y++;	100	101
x = y == 100;	1	100
x = y == y++;	0	101
x = y == ++y;	1	101

4.4 Δεδομένου του σπινέλου παρακάτω, γράψτε:

```
int x, y, z;  
x = 22;  
y = 40;  
z = 30;  
if (x == y) printf("x==y");  
  
// συμπληρώστε το κώδικα, ώστε να εκτυπώνεται αν x και y είναι ίσα  
// αν όχι * *  
  
x == y    αν το x είναι ίσο με το y  
x < z     αν το x είναι μικρότερο από το z  
x > z     αν το x είναι μεγαλύτερο από το z
```

```
if (x==z) printf("x==z");  
if (x<z)  
    printf("x<z");  
else  
    printf("x>z");
```

4.5 Να γραφτούν οι κώδικες, ώστε να εκτυπώνεται μια πρώτη σειρά, χρησιμοποιώντας τον τελεστή ++ * *

```
y = y + 1;  
z = x + y;  
x = x + 1;  
  
z = ++y + x++;
```

4.6 Ποιο είναι το αποτέλεσμα της εκτέλεσης του παρακάτω κώδικα; * *

```
15  → 1111  
52  → 110100  
0   → 0  
128 → 10000000
```

4.7 Ποιο είναι το αποτέλεσμα της εκτέλεσης του παρακάτω κώδικα; * *

```
1100111 → 103  
111     → 7  
1000000 → 64
```


4.8

Το αποτέλεσμα θα έχει το επόμενο πρόγραμμα: ★★

```
main()
{
    int a,b,c;
    a=5;
    b=8;
    printf("%d \n%d\n %d\n",a & b, a | b, a && b);
}
```

0
13
1

👉 Ο αντίστοιχος δυαδικός του 5 είναι 101 και του 8 1000 οπότε οι bitwise πράξεις **a & b** και **a | b** έχουν αποτέλεσμα 0000 (0) και 1101 (13) αντίστοιχα.

👉 Ο τελεστής **&&** είναι ο λογικός τελεστής (AND) και η λογική παράσταση **a && b** θα έχει αποτέλεσμα 1 (αληθές) δεδομένου ότι και τα δύο μέλη της (a και b) θεωρούνται αληθή (ως διάφορα του 0).

4.9 Ποια από τα επόμενα αληθεύουν: ★

- ☐ Το **i++** αυξάνει την τιμή του **i** κατά 1 ενώ το **++i** όχι.
- ☒ Οι τελεστές ++ και -- εφαρμόζονται **μόνο** σε μεταβλητές.
- ☒ Όταν κάνω μια πράξη bitwise AND (&) με το 0, το αποτέλεσμα θα είναι πάντα 0.
- ☐ Ο τελεστής ανάθεσης = έχει την πρώτη προτεραιότητα.
- ☒ Η παράσταση 5/2 έχει αποτέλεσμα τύπου **int** (το 2).

4.10

Υποθέτουμε ότι η τιμή του **x** είναι 5, του **y** είναι 100, του **a** είναι 0. Ποια από τις εκφράσεις είναι μίση από τις επόμενες παραστάσεις... Ποιες θα είναι οι τιμές των μεταβλητών **x** και **y** μετά από την εκτέλεση κάθε παραστάσης. ★★

Παράσταση	x	y	Παρατηρήσεις
$x = y > x \parallel a$	1	100	Προτεραιότητα έχει ο τελεστής OR (\parallel). Η έκφραση $x \parallel a$ έχει αποτέλεσμα 1 δεδομένου ότι το x θεωρείται αληθές (5). Η σύγκριση $y > 1$ είναι αληθής οπότε το x θα πάρει την τιμή 1.
$x = y \mid a;$	100	100	Η bitwise OR πράξη του y (δυαδικός 1100100) και του a (0000000) είναι 1100100 δηλαδή 100.
$y = x \& a;$	5	0	Η bitwise AND πράξη του x (δυαδικός 101) και του a (000) είναι 000 δηλαδή 0.
$x = x \& y;$	4	100	Η bitwise AND πράξη του x (δυαδικός 0000101) και του y (1100100) είναι 0000100 δηλαδή 4.
$x = x \mid y;$	101	100	Η bitwise OR πράξη του x (δυαδικός 0000101) και του y (1100100) είναι 1100101 δηλαδή 101.
$x = -x \&& y \parallel a;$	1	100	Ο τελεστής - έχει μεγαλύτερη προτεραιότητα, μετά ο AND (&&) και τέλος ο OR (\parallel). Το αποτέλεσμα της έκφρασης $-x \&& y$ είναι αληθές (1) δεδομένου ότι και τα δύο μέλη είναι αληθή (διάφορα του 0). Η έκφραση $1 \parallel a$ είναι αληθής δεδομένου ότι το πρώτο μέλος είναι αληθές. Επομένως το αποτέλεσμα της παράστασης το οποίο θα καταχωρηθεί στη x είναι το 1.

4.11

Μια ψηφιακή συσκευή για την εξέταση παρόμοιας ενός κτηρίου, επιστρέφει στην επιχείρηση έναν αριθμό αριθμών τα bits του οποίου έχουν το ακόλουθο νόημα: Το bit 7,6,5 και 4 προσφέρουν το σημάδι ελέγχου (0 πιθανόν, 1 αντιστοιχεί με τιμές από 0 μέχρι 15), το bit 3 και 2 το είδος του συναγερμού (0-κέντρο, 1-ορόφος, 2-επιπρόσθια, 3-κέντρο), το bit 1 δείχνει αν η συσκευή λειτουργεί κανονικά (0-πρόβλημα, 1-OK), ενώ το bit 0 δίνει την κατάσταση. Να γραφεί πρόγραμμα το οποίο θα πάρει τον αριθμό που επιστρέφει η συσκευή και θα εμφανίσει τον αντίστοιχο σημειωτικό (από 1 μέχρι 16) καθώς και το είδος του συναγερμού (π.χ. κέντρο). Στη περίπτωση που η συσκευή δεν λειτουργεί κανονικά δεν πρέπει να εμφανίσει τίποτα παρά μόνο τη φράση "Πρόβλημα στη συσκευή".

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

main()
{
    int ar;
    int ok,simeio,alarm;
    printf("Δώσε αριθμό από τη συσκευή:");
    scanf("%d",&ar);
    simeio=(ar&240)>>4;
    alarm=(ar&12)>>2;
    ok=(ar&2)>>1;
    if (ok==0)
    {
        printf("Πρόβλημα στη συσκευή\n");
        exit(1);
    }
    printf("Σημείο ελέγχου: %d Συναγερμός:",simeio+1);
    if (alarm==0) printf("OK\n");
    if (alarm==1) printf("Φωτιά\n");
    if (alarm==2) printf("Παραβίαση\n");
    if (alarm==3) printf("Καπνός\n");
}
```

Απομόνωση των τεσσάρων bit (7,6,5 & 4) και μετακίνηση τους στις τέσσερις πρώτες θέσεις του byte.

Απομόνωση των δύο bit (3 & 2) και μετακίνηση τους στις πρώτες θέσεις του byte.

Απομόνωση του δεύτερου bit.

👉 Ο αριθμός 240 είναι ο δυαδικός 11110000. Η bitwise πράξη AND με τον αριθμό **ar** χρησιμοποιείται για να απομονώσει τα bit 7,6,5,και 4. Η ολίσθηση δεξιά κατά 4 θέσεις μετακινεί τα απομονωμένα bit στις τέσσερις πρώτες θέσεις του byte ώστε να αποδώσουν τιμή από 0 μέχρι 15.

👉 Ο αριθμός 12 είναι ο δυαδικός 00001100. Η bitwise πράξη AND με τον αριθμό **ar** χρησιμοποιείται για να απομονώσει τα bit 3 και 2. Η ολίσθηση δεξιά κατά 2 θέσεις μετακινεί τα απομονωμένα bit στις πρώτες θέσεις του byte ώστε να αποδώσουν τιμή από 0 μέχρι 3.

👉 Ο αριθμός 2 είναι ο δυαδικός 00000010. Η bitwise πράξη AND με τον αριθμό **ar** χρησιμοποιείται για να απομονώσει το bit No 1 (το δεύτερο). Η ολίσθηση δεξιά μετακινεί το bit στη πρώτη θέση του byte ώστε να αποδώσει τιμή 1 ή 0.

4.12

Να γραφεί πρόγραμμα το οποίο να ζητάει από τον χρήστη να εισάγει έναν αριθμό και να ελέγξει αν ο αριθμός που εισήγαγε είναι άρτιος ή περιττός. 

```
#include <stdio.h>
#include <stdlib.h>
main()
{
    int ar;
    printf("Δώσε αριθμό :");
    scanf("%d",&ar);
    if (ar%2==0)
        printf("Ο αριθμός %d είναι ζυγός\n",ar);
    else
        printf("Ο αριθμός %d είναι μονός\n",ar);
}
```

Η παράσταση `ar%2` υπολογίζει το υπόλοιπο της διαίρεσης του αριθμού `ar` με το 2. Αν είναι 0 ο αριθμός είναι ζυγός, διαφορετικά είναι μονός.

4.13

Νο. πέρασε πρόγραμμα το οποίο να ζητάει το χρόνο σε δευτερόλεπτα, να το εμφανίζει, να το χωρίζει, να αντίζει, να μετατρένει και να αντίζει δευτερόλεπτα που είναι στην ώρα, στα λεπτά και το υπόλοιπο. **

```
#include <stdio.h>
#include <stdlib.h>
main()
{
    int deyter, ores, lepta, sec;
    printf("Δώσε δευτερόλεπτα :");
    scanf("%d",&deyter);
    ores=deyter/3600;
    lepta=(deyter-ores*3600)/ 60;
    sec=deyter%60;
    printf("Τα %d δευτερόλεπτα είναι:\n",deyter);
    printf("%d ώρες %d λεπτά και %d δευτερόλεπτα\n",ores,lepta,sec);
}
```

Η ακέραια διαίρεση `deyter/3600` υπολογίζει τις ώρες που περιέχονται στα δευτερόλεπτα που δόθηκαν.
Το υπόλοιπο των δευτερολέπτων (`deyter-ores*3600`) αν διαιρεθεί με το 60 αποδίδει τα υπόλοιπα λεπτά.
Τέλος τα δευτερόλεπτα που υπολείπονται υπολογίζονται από το υπόλοιπο της διαίρεσης των συνολικών δευτερολέπτων με το 60.

4.14

Νο. γράσει πρόγραμμα το οποίο να ζητάει τα βαθμολογία ενός μαθητή σε τρία μαθήματα. Το πρόγραμμα να υπολογίζει τον μέσο όρο, να επηρεάζονται οι βαθμολογίες. "Πέρασες" στην περίπτωση που είναι μεγαλύτερος ή ίσος από το 10, διαφορετικά να εμφανίζει "Κόπηκες". **

```
#include <stdio.h>
#include <stdlib.h>
main()
{
    float b1,b2,b3,mo;
    printf("Δώσε τρεις βαθμούς :");
    scanf("%f %f %f",&b1,&b2,&b3);
    mo=(b1+b2+b3)/3;
    if (mo>=10)
        printf("Πέρασες με %f\n",mo);
    else
        printf("Κόπηκες\n");
}
```

Υπολογισμός του μέσου όρου των τριών βαθμών.

Έλεγχος του μέσου όρου.

Ασκήσεις Κεφαλαίου 5

5.1

Να γραφεί πρόγραμμα το οποίο να δεχτεί ένα χαρακτήρα από το πληκτρολόγιο και να τον επεξεργαστεί ως εξής: *

- Αν ο χαρακτήρας είναι πεζός, να τον τυπώνει στην οθόνη.
- Εάν προκύψει αριθμητικό μήνισμα (0-9), να εμφανίζεται στην οθόνη Πατήθηκε ένα ψηφίο.
- Σε κάθε άλλη περίπτωση να μην κάνει τίποτα.

```
main()
{
    char ch;
    ch=getch();
    if ((ch>='a' && ch<='z') || (ch>='α' && ch<='ω'))
        putchar(ch);
    if (ch>='0' && ch<='9')
        printf("Πατήθηκε ένα ψηφίο\n");
}
```

➡ Η έκφραση `ch>='a' && ch<='z'` είναι αληθής όταν ο χαρακτήρας είναι πεζός λατινικός ενώ η έκφραση `ch>='α' && ch<='ω'` είναι αληθής όταν ο χαρακτήρας είναι πεζός ελληνικός.

5.2

Να γραφεί πρόγραμμα το οποίο να δεχτεί ένα χαρακτήρα από το πληκτρολόγιο και να τον επεξεργαστεί ως εξής: *

- Αν ο χαρακτήρας είναι αριθμητικός (αριθμός), να τον εμφανίζει όπως είναι.
- Αν είναι λατινικός (ή αραβικός ή κεφαλαίος), να εμφανίζει το επόμενο επόμενο χαρακτήρα, δηλαδή αν πατηθεί η κλειδαριά 'a' να εμφανιστεί το 'c' κ.λπ.

```
main()
{
    char ch;
    ch=getch();
    if ((ch>='a' && ch<='z') || (ch>='A' && ch<='Z'))
        putchar(ch+1);
    if (ch>='0' && ch<='9')
        putchar(ch);
}
```

➡ Η `if`, στη παραπάνω λύση, ελέγχει για λατινικούς μόνο χαρακτήρες πεζούς ή κεφαλαίους. Αν θέλαμε να περιλαμβάνει και τους ελληνικούς θα έπρεπε η λογική έκφραση να ήταν `((ch>='a' && ch<='z') || (ch>='A' && ch<='Z')) || ((ch>='α' && ch<='ω') || (ch>='Α' && ch<='Ω'))`.

➡ Η πρόταση `putchar(ch+1)` εμφανίζει τον επόμενο χαρακτήρα από το περιεχόμενο του `ch`.

5.3

Νο. πρέπει να γράψουμε το πρόγραμμα που εμφανίζει το αποικανό "hi" θα μπορούσε να είναι και "hi" και να κάνει την αντίστοιχη ενέργεια. ***

```
1-Εκτύπωσε την λέξη "Hello"
2-Εκτύπωσε τον αριθμό 2
3-Εκτύπωσε "bye bye"
4-Μην κάνεις τίποτα
Δώσε επιλογή:
```

```
main()
{
    char ch;
    printf("1-Εκτύπωσε την λέξη Hello\n");
    printf("2-Εκτύπωσε τον αριθμό 2\n");
    printf("3-Εκτύπωσε bye bye\n");
    printf("4-Μην κάνεις τίποτα\n");
    printf("Δώσε επιλογή:");
    ch=getch();
    if (ch=='1') printf("Hello\n");
    if (ch=='2') printf("2\n");
    if (ch=='3') printf("bye bye\n");
    if (ch!='1' && ch!='2' && ch!='3' && ch!='4')
        printf("Λάθος επιλογή");
}
```

5.4

Η απάντηση θα είναι το παρακάτω πρόγραμμα. ***

```
main()
{
    char ch,b='A';
    ch='A';
    if(ch==b)
        printf("NAI-1");
    else
        printf("OXI-1");
    if("A"=='A')
        printf("NAI-2");
    else
        printf("OXI-2");
}
```

NAI-1OXI-2

- ☞ Η λογική παράσταση **ch==b** είναι αληθής διότι και οι δύο μεταβλητές περιέχουν τον χαρακτήρα 'A'.
- ☞ Η λογική παράσταση **"A"=='A'** είναι ψευδής διότι το "A" είναι συμβολοσειρά ενώ το 'A' χαρακτήρας. Συγκρίνεται η διεύθυνση της συμβολοσειράς με τον κωδικό του χαρακτήρα.

5.5 Ποια από τα παρακάτω αληθεύουν: ★

- ☑ Μπορούμε να χειριζόμαστε τους χαρακτήρες σαν αριθμούς.
- ☑ Μια μεταβλητή τύπου **char** έχει μέγεθος ενός byte.
- ☐ Σε μια μεταβλητή χαρακτήρα δεν μπορούμε να καταχωρίσουμε έναν αριθμό.
- ☑ Οι συμβολοσειρές προσδιορίζονται από τη διεύθυνση όπου είναι αποθηκευμένος ο πρώτος τους χαρακτήρας.
- ☐ Μια συμβολοσειρά καταλαμβάνει τόσα byte όσοι **ακριβώς** είναι και οι χαρακτήρες που περιέχει.

5.6

Θεωρείται το πρόγραμμα το οποίο να εμφανίζει τους κωδικούς των χαρακτήρων a, * και του κενού διαστήματος. **

```
main()
{
    printf("Ο κωδικός του a είναι %d\n",'a');
    printf("Ο κωδικός του * είναι %d\n",'*');
    printf("Ο κωδικός του κενού είναι %d\n",' ');
}
```

5.7

Να γράψουμε πρόγραμμα το οποίο να εμφανίζει τους χαρακτήρες με κωδικούς 80, 125 και 192. **

```
main()
{
    printf("Ο χαρακτήρας με κωδικό 80 είναι %c\n",80);
    printf("Ο χαρακτήρας με κωδικό 125 είναι %c\n",125);
    printf("Ο χαρακτήρας με κωδικό 192 είναι %c\n",192);
}
```

5.8

Να γράψουμε το πρόγραμμα που εμφανίζει τον αριθμό a, τον χαρακτήρα ch και τη λέξη let. **

```
main()
{
    char ch=68, let='L';
    int a=2, b=4;
    a=ch+let;
    ch=++let;
    printf("a=%d ch=%c let=%c\n", ++a, ch, let);
}
```

a=145 ch=M let=M

- 👉 Η παράσταση **ch+let** έχει αποτέλεσμα 144 (68 + 76) δεδομένου ότι ο ASCII κωδικός του 'L' είναι 76.
- 👉 Η πρόταση **++let** αυξάνει τη **let** κατά 1 και την κάνει 77 που είναι ο ASCII κωδικός του 'M'.

5.9

Ποιο το διαφορά του 'A' με το "A"? **

- 👉 Το 'A' αναφέρεται στον χαρακτήρα 'A' και ισοδυναμεί με τον αριθμό 65 που είναι ο ASCII κωδικός του 'A'.
- 👉 Το "A" αναφέρεται σε μία συμβολοσειρά και ισοδυναμεί με τη διεύθυνση της πρώτης θέσης μνήμης στην οποία έχει καταχωρηθεί η συγκεκριμένη συμβολοσειρά (βλέπε σελίδα 107 του βιβλίου).

5.10

Παρασάφηνισε το πρόγραμμα στο οποίο υπάρχει ένα χαρακτήρας από το πληκράκι που πατάει ο χρήστης τον οποίο τον χαρακτήρα τον κωδικό και τον ASCII κωδικό του. **

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

main()
{
    char ch;
    printf("Πάτησε κάποιο χαρακτήρα :");
    ch=getch();
    printf("\nΠάτησες το %c με ASCII κωδικό %d\n",ch,ch);
}
```

Στη **printf()** και οι δύο παραστάσεις είναι ίδιες: ο χαρακτήρας **ch**. Με το **%c** την πρώτη φορά εμφανίζεται ως χαρακτήρας, ενώ τη δεύτερη (με το **%d**) ως αριθμός.

5.11

Νο, μπορείς να πάρεις το κωδικό ενός αριθμού βάζοντας τον αριθμό στον χαρακτήρα με κωδικό τον ASCII κωδικό. **

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

main()
{
    int ar;
    printf("Δώσε έναν αριθμό :");
    scanf("%d",&ar);
    printf("Ο χαρακτήρας με κωδικό %d είναι ο %c\n",ar,ar);
}
```

Στη **printf()** και οι δύο παραστάσεις είναι ίδιες: ο αριθμός **ar**. Με το **%d** την πρώτη φορά εμφανίζεται ως αριθμός, ενώ τη δεύτερη (με το **%c**) ως χαρακτήρας.

Ασκήσεις Κεφαλαίου 6

6.1

Να γραφεί πρόγραμμα το οποίο να ζητάει ένα τεκμήριο από το πληκτρολόγιο, να υπολογίζει το μέσο όρό τους και να το εμφανίζει με δύο δεκαδικά ψηφία (π.χ. 5.50) και συνολικά επτά θέσεις στην οθόνη. *

```
main()
{
    float a,b,mo;
    scanf("%f %f",&a,&b);
    mo=(a+b)/2;
    printf("MO=%7.2f\n",mo);
}
```

6.2

Να γραφεί πρόγραμμα το οποίο να ζητάει την ακτίνα ενός κύκλου. Να υπολογιστεί και το εμβαδόν του κύκλου που με βάση περιέχει. Να γραφεί πρόγραμμα σύνθεσης **pow()** με την εμφάνιση σε οθόνη. Η επιτ. του π 3.141592653589793 να δηλωθεί με συνάρτηση με την οθόνη **#define**. *

```
#include <math.h>
#define pi 3.141592653589793
main()
{
    double r,e;
    printf("Δώσε ακτίνα:");
    scanf("%lf",&r);
    e=pow(r,2)*pi;
    printf("Εμβαδον κύκλου ακτινας %f είναι %f\n",r,e);
}
```

6.3

Να αποτέλεσμα θα έχει το επόμενο πρόγραμμα: και γιατί *

```
main()
{
    float d;
    int a,b;
    a=5;
    b=6;
    d=(a+b)/2;
    printf("%f\n",d);
}
```

5

- ☞ Η παράσταση $(a+b)/2$ θα έχει αποτέλεσμα τύπου **int** διότι όλα τα μέλη της είναι τύπου **int**. Οπότε το αποτέλεσμα της θα είναι 5 και όχι 5.5 που θα ήταν το αναμενόμενο.
- ☞ Αν θέλαμε να υπολογιζόταν σωστά τότε θα έπρεπε να γραφεί ως $(a+b)/2.0$. Το 2.0 που είναι τύπου **float** "εξαναγκάζει" την όλη παράσταση να έχει αποτέλεσμα **float**, οπότε διατηρεί τα δεκαδικά της ψηφία.

6.4 Ποια από τα επόμενα αληθεύουν: ★

- ☑ Οι τελεστές ++ και -- δεν μπορούν να εφαρμοστούν σε μεταβλητές τύπου **float**. Αυτό ισχύει σύμφωνα με το πρότυπο ANSI, όμως αρκετοί μεταγλωττιστές το καταστρατηγούν και επιτρέπουν τη χρήση των τελεστών αυτών και σε μεταβλητές τύπου **float**.
- ☐ Οι μεταβλητές τύπου **double** αποθηκεύουν απεριόριστο αριθμό δεκαδικών ψηφίων.
- ☐ Με τη συνάρτηση **printf()** δεν μπορούμε να καθορίσουμε τον ακριβή αριθμό των δεκαδικών ψηφίων που θα εμφανίζονται στην οθόνη.
- ☑ Η παράσταση $1+1.0$ έχει αποτέλεσμα τύπου **float**.
- ☑ Η C δεν διαθέτει τελεστή για ύψωση σε δύναμη.

6.5

Να γραφτεί πρόγραμμα το οποίο να υπολογίζει το υπόλοιπο της ακεραίας διαίρεσης ενός δεκαδικού αριθμού με άλλο ακέραιο. Το πρόγραμμα θα τηρείται στο ακόλουθο ένα δεκαδικό και έναν ακέραιο και θα εμφανίζει το υπόλοιπο. Αν υπαίτιασε π.χ. θα δίνουμε τους αριθμούς 5.14 και 2 το αποτέλεσμα είναι το 1.14. ★★

```
main()
{
    float d, yp;
    int a,b;
    scanf("%f %d", &d, &a);
    b=d/a;
    yp=d-b*2;
    printf("%f\n", yp);
}
```

👉 Στη πρόταση **b=d/a**, η **b** είναι τύπου **int** οπότε θα αποθηκευτεί μόνο το ακέραιο τμήμα του αποτελέσματος της παράστασης **d/a**.

6.6

Να γραφτεί πρόγραμμα το οποίο να ζητάει δύο πραγματικούς αριθμούς και να εμφανίζει το γινόμενο τους με τον τρόπο που φαίνεται στο παρακάτω πλαίσιο. Οι αριθμοί θα πρέπει να εμφανίζονται με δύο δεκαδικά ψηφία και να καταλαμβάνουν εννέα θέσεις στην οθόνη. ★★

	10.50
x	2.00
<hr/>	
	21.00

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

main()
{
    float ar1,ar2;
    printf("Δώσε δύο αριθμούς :");
    scanf("%f %f", &ar1, &ar2);
    printf(" %9.2f\n", ar1);
    printf("x%9.2f\n", ar2);
    printf("=====\n");
    printf(" %9.2f\n", ar1*ar2);
}
```

}

6.7

Νο. 7. Κρατήστε ένα πρόγραμμα το οποίο να ζητάει τιμές για τα Α και Β και να υπολογίζει την τιμή της παράστασης που βρίσκεται στο επόμενο πλαίσιο. Το αποτέλεσμα να εκτυπώνεται στο όθρονη, ανάλογα με τη μεγαλύτερη δυνατή ακρίβεια σε δεκαδικά ψηφία. *

$$\frac{A}{A+B} \cdot \frac{B}{A-B} + \frac{A^{A+B}}{B^{A-B}}$$

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

```
main()
{
    double a,b,x;
    printf("Δώσε τιμή για το A :");
    scanf("%lf",&a);
    printf("Δώσε τιμή για το B :");
    scanf("%lf",&b);
    x=(a/(a+b))* (b/(a-b)) +pow(a,a+b)/pow(b,a-b);
    printf("Το αποτέλεσμα της παράστασης είναι: %f\n",x);
}
```

Εφόσον ζητείται η μεγαλύτερη δυνατή ακρίβεια, δηλώνονται μεταβλητές τύπου **double**.

Προσοχή στη **scanf()** η οποία χρησιμοποιεί το συνδυασμό **%lf** όταν πρόκειται για δεδομένα διπλής ακρίβειας (**double**).



Για την ύψωση σε δύναμη χρησιμοποιείται η συνάρτηση **pow()**. Η συνάρτηση **pow()** δηλώνεται στο αρχείο κεφαλίδας **math.h** το οποίο πρέπει να συμπεριληφθεί (με την οδηγία **include**) στο πρόγραμμά μας.

Ασκήσεις Κεφαλαίου 7

7.1

Νο. 7.1: Γράψτε ένα πρόγραμμα να ζητήσει το εισόδημα ενός πολίτη και να υπολογίσει το φόρο εισοδήματος με την εξής * *

- Αν το εισόδημα είναι κάτω από 5000, ο φόρος θα είναι 0.
- Αν το εισόδημα είναι από 5000 μέχρι 10000, ο φόρος θα είναι 5%.
- Αν το εισόδημα είναι πάνω από 10000 και μέχρι 30000, ο φόρος θα είναι 15%.
- Αν το εισόδημα είναι πάνω από 30000, ο φόρος θα είναι 35%.

```
main()
{
    float eis,foros;
    printf("Δώσε εισόδημα:");
    scanf("%f",&eis);
    if(eis<5000)
        foros=0;
    else if (eis>=5000 && eis<=10000)
        foros=eis*5/100;
    else if (eis>10000 && eis<=30000)
        foros=eis*15/100;
    else
        foros=eis*35/100;
    printf("Ο φόρος για εισόδημα %f είναι %f\n",eis,foros);
}
```

Η πρόταση αυτή θα εκτελεστεί όταν δεν ισχύει καμία από τις παραπάνω περιπτώσεις των εντολών if.

7.2

Νο. 7.2: Γράψτε το παρακάτω πρόγραμμα * *

```
main()
{
    int a,b;
    char ch;
    ch=getch();
    if((ch>='A') && (ch<='Z'))
        ++ch;
    else
        --ch;
    putchar(ch);
}
```

Περιμένει να πληκτρολογηθεί ένας χαρακτήρας τον οποίο καταχωρεί στη μεταβλητή **ch**.

Αν ο χαρακτήρας είναι κεφαλαίος λατινικός τότε αυξάνει το περιεχόμενο της **ch** κατά 1, διαφορετικά το μειώνει κατά 1.

Εμφανίζει τον χαρακτήρα με κωδικό **ch**, ο οποίος θα είναι ή ο επόμενος ή ο προηγούμενος από τον χαρακτήρα που δόθηκε αρχικά.



Για παράδειγμα, αν δώσουμε το 'B' θα εμφανιστεί το 'C' ενώ αν δώσουμε το 'b' θα εμφανιστεί το 'a'.

7.3

Πρέπει να λάβω υπόψη μου πρόγραμμα **

```
main()
{
    int a,b;
    a=getch();
    b='*';
    switch(a)
    {
        case 1:
            printf("%c\n",a);
            printf("-----\n");
        case b:
            printf("%d\n",b);
            break;
        case 'A':
            printf("aaaaaaaaaaaa");
            break;
        case 'A'+1:
            printf("telos");
            break;
        case 4:
            printf("4444444444");
    }
}
```

Η case ακολουθείται μόνο από σταθερές, και όχι μεταβλητές (b).

Η case ακολουθείται μόνο από σταθερές, και όχι παραστάσεις ('A'+1).

Η case πρέπει να τερματίζεται με :

7.4

Ποια συμπεριφορά στην οθόνη το επόμενο πρόγραμμα **

- ✦ Αν απαντήσω τον χαρακτήρα D (D=66)
- ✦ Αν απαντήσω τον χαρακτήρα A (A=65)
- ✦ Αν απαντήσω τον χαρακτήρα a
- ✦ Αν απαντήσω τον χαρακτήρα *

```
main()
{
    int a,b;
    b=44;
    a=getch();
    switch(a)
    {
        case 66:
            printf("%c\n",a);
            printf("-----\n");
        case 7:
            printf("%d\n",b);
            break;
        case 'A':
        case 'a':
            printf("aaaaaaaaaaaa");
    }
```

```

        break;
    default:
        printf("1234567890");
    }
}

```

👉 Στην περίπτωση που πληκτρολογηθεί ο χαρακτήρας 'B' (έχει ASCII κωδικό 66) υπάγεται στην πρώτη **case 66**: και θα εκτελεστούν οι προτάσεις της πρώτης **case**. Επειδή όμως οι προτάσεις της πρώτης **case** δεν τερματίζονται με εντολή **break**, θα εκτελεστούν και οι προτάσεις της **case 7**:

B

44

👉 Στην περίπτωση που πληκτρολογηθεί ο χαρακτήρας 'A' ή ο χαρακτήρας 'a' θα εκτελεστούν οι προτάσεις που ακολουθούν τις **case 'A'**: και **case 'a'**:

aaaaaaaaaaaa

👉 Στην περίπτωση που πληκτρολογηθεί ο χαρακτήρας '*', δεν υπάγεται σε καμία από τις περιπτώσεις **case** και θα εκτελεστούν οι προτάσεις της **default**:

1234567890

7.5

👉 Κάνει το πρόγραμμα πηγαίνοντας να πληκτρολογηθεί 'A', 'B', 'C' ή 'D' και 'K' όταν οι χαρακτήρες:

```

main()
{
    int a,b;
    char ch;
    a=ch=getch();
    if(ch>=65 && ch<='D')
    {
        switch(ch)
        {
            case 65:
                b=++a;
                ++b;
                break;
            case 66:
                b=a--;
            default:
                b=a+5;
        }
        printf("a=%d b=%d ch=%c\n",a,b,ch);
    }
    else if(ch=='*')
    {
        a=b=ch-1;
        printf("***%d*****%d*****\n",a,b);
    }
}

```

Το σώμα της **if** θα εκτελεστεί μόνο όταν ο χαρακτήρας είναι 'A','B','C' ή 'D'

Οι προτάσεις θα εκτελεστούν μόνο αν ο χαρακτήρας είναι '*'.

👉 Η πρόταση **a=ch=getch()**; περιμένει να πληκτρολογηθεί ένας χαρακτήρας τον οποίο αποθηκεύει τόσο στην μεταβλητή **ch** όσο και στην **a** (τον ASCII κωδικό του).

👉 Στην περίπτωση που πληκτρολογηθεί ο χαρακτήρας 'A' (έχει ASCII κωδικό 65) υπάγεται στην πρώτη **case 65**: και θα εκτελεστούν οι προτάσεις της πρώτης **case**. Η μεταβλητή **a** θα γίνει 66 τιμή που θα καταχωριστεί στη **b** ($b=++a$). Μετά η **b** θα πάρει τιμή 67 ($++b$).

a=66 b=67 ch=A

👉 Στην περίπτωση που πληκτρολογηθεί ο χαρακτήρας 'B' (έχει ASCII κωδικό 66) θα εκτελεστούν οι προτάσεις της **case 66**: αλλά και της **default**: γιατί δεν υπάρχει εντολή **break** στις προτάσεις της **case 66**:. Η μεταβλητή **a** θα γίνει 65 ($b=a--$) αλλά στη **b** καταχωρείται η τιμή πριν από τη μείωση (το 66). Μετά η **b** θα πάρει τιμή 70 ($b=a+5$).

a=65 b=70 ch=B

👉 Στην περίπτωση που πληκτρολογηθεί ο χαρακτήρας 'D' (έχει ASCII κωδικό 68) θα εκτελεστούν οι προτάσεις της **default**: και η **b** θα πάρει τιμή 73 ($b=a+5$).

a=68 b=73 ch=D

👉 Στην περίπτωση που πληκτρολογηθεί ο χαρακτήρας '*' (έχει ASCII κωδικό 42), τότε θα γίνουν οι προτάσεις της **else if**. Οι μεταβλητές **a** και **b** θα πάρουν τιμή 41 ($a=b=ch-1$).

*****41****41*******

👉 Στην περίπτωση που πληκτρολογηθεί ο χαρακτήρας 'K', δεν θα γίνει απολύτως τίποτα.

7.6

Να γραφεί το πρόγραμμα το οποίο να διακρίνει αν ένα έτος είναι δίσεκτο ή όχι σύμφωνα με τη μέθοδο ***.

- Θα ζητηθεί να πληκτρολογηθεί το έτος.
- Θα εμφανιστεί το λέξι "Δίσεκτο" αν το έτος είναι δίσεκτο και τη λέξη "Κανονικό" αλλιώς.
- Δίσεκτο είναι ένα έτος όταν διαιρείται ακριβώς με το 4. Όταν το έτος δεν διαιρείται ακριβώς με το 100 δεν είναι δίσεκτο. Αν διαιρείται ακριβώς με το 400 τότε είναι δίσεκτο.

```
main()
{
    int etos;
    printf("Δώσε έτος:");
    scanf("%d",&etos);
    if(etos%4 == 0)
    {
        if(etos%100 == 0)
        {
            if(etos%400 == 0)
                printf("Δίσεκτο\n");
            else
                printf("Κανονικό\n");
        }
        else
            printf("Δίσεκτο\n");
    }
    else
    {

```

Για να είναι ένα έτος δίσεκτο πρέπει να διαιρείται ακριβώς με το 4. Η παράσταση **etos%4** δίνει το υπόλοιπο της ακέραιας διαίρεσης με το 4 (βλέπε σελίδα 85 του βιβλίου).

Αν διαιρείται ακριβώς με το 100 δεν είναι δίσεκτο.

Εκτός αν διαιρείται ακριβώς με το 400, οπότε είναι δίσεκτο.

```
        printf("Κανονικό\n");
    }
}
```

7.7

Εάν ο χρήστης εισάγει το όπιο, ως επιλογή, να εμφανιστεί η λέξη **switch**. Εάν ο χρήστης εισάγει κάποιο άλλο να εμφανιστεί η αντίστοιχη ενέργεια. *

K->Να εμφανίζει τη λέξη "Καλημέρα"
 2->Να εμφανίζει τον αριθμό 2
 T->Να εμφανίζει "Τέλος"

Εάν η επιλογή δεν είναι αποδεκτή (εκτός από K,2, και T) τότε να εμφανίζει το μήνυμα "Λάθος επιλογή".

```
main()
{
    int a,b;
    char ch;
    ch=getch();
    switch(ch)
    {
        case 'K':
            printf("Καλημέρα\n");
            break;
        case '2':
            printf("2\n");
            break;
        case 'T':
            printf("Τέλος\n");
            break;
        default:
            printf("Λάθος επιλογή\n");
    }
}
```

7.8 Ποια από τα παρακάτω αληθεύουν: ★

- ☒ Η **case** πρέπει απαραίτητα να ακολουθείται από ακέραια σταθερά ή σταθερά χαρακτήρα.
- ☒ Οι εντολή **switch-case** μπορεί να αντικατασταθεί από ισοδύναμες εντολές **if-else if**.
- ☐ Οι εντολές **if-else if** μπορούν να αντικατασταθούν από ισοδύναμες εντολές **switch-case**.
- ☐ Η πρόταση **a==5** είναι ισοδύναμη με την **a=5**.
- ☐ Η εντολή **if** είναι η μοναδική εντολή που διαθέτει η C για τον έλεγχο λογικών παραστάσεων.

7.9

Μία εταιρεία που ασχολείται με τη μεταφορά των οχημάτων, έχει 30 και δίνει πρόσβαση σε κέρματα των 10, 20, 50 λεπτών καθώς και σε κέρματα των 1 και 2 ευρώ. Η εταιρεία κοστίζει 70 λεπτά. Να γραφτεί πρόγραμμα το

Επειδή να ζητήσει τον αριθμό των καφέδων και να εμφανίσει τα χρήματα σε λεπτά, ο χρήστης πρέπει να δώσει στη μεταβλητή `resta` το γινόμενο των 500 λεπτών με 70 λεπτά το καφέ. ***

```
#include <stdio.h>
#include <stdlib.h>
main()
{
    int kafedes, ker_10, ker_20, ker_50, ker_1, ker_2;
    int resta;
    printf("Δωσε αριθμό καφέδων:");
    scanf("%d", &kafedes);
    resta=500-kafedes*70;
    if (resta<0)
    {
        printf("Δεν φτάνουν τα χρήματα για τόσους καφέδες\n");
        exit(1);
    }
    ker_2=resta/200;
    resta=resta % 200;
    ker_1=resta/100;
    resta=resta % 100;
    ker_50=resta/50;
    resta=resta % 50;
    ker_20=resta/20;
    resta=resta % 20;
    ker_10=resta/10;
    resta=resta % 10;
    printf("Ρέστα\n");
    printf("Κέρματα 2€ :%d\n", ker_2);
    printf("Κέρματα 1€ :%d\n", ker_1);
    printf("Κέρματα 50λ :%d\n", ker_50);
    printf("Κέρματα 20λ :%d\n", ker_20);
    printf("Κέρματα 10λ :%d\n", ker_10);
}
```

Τα χρήματα υπολογίζονται σε λεπτά. Έτσι τα 5€ υπολογίζονται ως 500 Λεπτά.

Τα κέρματα των 2€ (200 λεπτά) προκύπτουν από την ακέραια διαίρεση του ποσού των ρέστων με το 200. Το υπόλοιπο ποσό προκύπτει από τη παράσταση `resta % 200`.

Η ίδια διαδικασία ακολουθείται και για τον υπολογισμό των υπολοίπων κερμάτων.

7.10

Ο Δεσφής, μάστερ ζωγράφος (ΔΜΣ) υπολογίζει τα χρήματα που θα πάρει από την πώληση ενός έργου του στο κατάστημα τέχνης. Ανάλογα με το είδος του ΔΜΣ και το είδος του έργου, εμφανίζει τον κατάλληλο αριθμό:

ΔΜΣ	Περιγραφή
μικρότερος από 18.5	Απολάσεις
από 18.5 και μικρότερος του 25	Κανονικός
από 25 και μικρότερος του 50	Χαρβακός
από 50 και πάνω	Παράσηκος

Να γραφεί πρόγραμμα σε C που να ζητάει να πληκτρολογήσουμε το Γ.Ο.Ν. και το είδος του έργου και να υπολογίζει τον ΔΜΣ. Το πρόγραμμα θα αναγράφει την τιμή του ΔΜΣ θα εμφανίζει τον χαρακτήρα που τον αντιστοιχεί (α, β, γ, δ) και θα βγαίνει. **


```
#include <stdio.h>
#include <stdlib.h>
main()
{
    float ypsos,baros,dms;
    printf("Δώσε ύψος και βάρος :");
    scanf("%f %f",&ypsos,&baros);
    dms = baros/(ypsos*ypsos);
    if (dms<18.5)
        printf("Λιποβαρής\n");
    else if (dms>=18.5 && dms<25)
        printf("Κανονικός\n");
    else if (dms>=25 && dms<30)
        printf("Υπέρβαρος\n");
    else
        printf("Παχύσαρκος\n");
}
```

7.11

Να γράψετε πρόγραμμα το οποίο θα ζητάει τους συντελεστές α, β και γ της εξίσωσης $a \cdot x^2 + b \cdot x + c = 0$. Το πρόγραμμα να δίνει στην οθόνη τη λύση ή προσανατολισμό πώς να τη βρούμε και το πώς ελέγχεται. Σε διαφορετική περίπτωση να εμφανίζει μήνυμα. Δεν υπάρχουν τελεστές $\sqrt{\quad}$ και \log .

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

main()
{
    float a,b,c,d,r1,r2;
    printf("Δωσε τους συντελεστές α β & γ:");
    scanf("%f %f %f",&a,&b,&c);
    d = b*b-4*a*c; //υπολογισμός διακρίνουσας
    if (d>0)
    {
        r1=(-b+sqrt(d))/(2*a);
        r2=(-b-sqrt(d))/(2*a);
        printf("R1=%f R2=%f\n",r1,r2);
    }
    else if (d==0)
    {
        r1=-b/(2*a);
        printf("Διπλή ρίζα R=%f\n",r1);
    }
    else
        printf("Δεν έχει πραγματικές ρίζες\n");
}
```

7.12

Μια εταιρεία κινητής τηλεφωνίας χρεώνει το SMS σύμφωνα με το παρακάτω σχήμα τιμολόγησης:

Πλήθος SMS	Τιμή ανά SMS
τα πρώτα 10	2 Λεπτά
τα επόμενα 50	1.5 Λεπτά
τα επόμενα 100	1.2 Λεπτά
όλα τα επόμενα	1 Λεπτό

Να γραφεί το πρόγραμμα που θα δέχεται από το χρήστη το πλήθος των SMS που στέλνει, το υπολογίζει και το εμφανίζει σε Euro, το ποσό που πρέπει να πληρώσει.

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

main()
{
    int sms;
    float poso;
    printf("Δώσε πλήθος sms :");
    scanf("%d",&sms);
    if (sms<=10)
        poso=sms*2;
    else if (sms<=60)
        poso=10*2 + (sms-10)*1.5;
    else if (sms<=160)
        poso=10*2 + 50*1.5 + (sms-60)*1.2;
    else
        poso=10*2 + 50*1.5 + 100*1.2 + (sms-160)*1;
    printf("Συνολικό ποσό σε euro: %f\n",poso/100);
}
```

7.13

Μια περιφερειακή εταιρεία θα πρέπει να διατηρεί ένα συγκεκριμένο ποσοστό πληρωτήτων που έχει βάλει, τουλάχιστον το 50% του ύψους του απαιτητικού. Εάν σε περίπτωση που έχει βάλει κάτω από το 50% σε ένα έτος, τότε θα πρέπει να αυξήσει το ποσό των θέσεων και τον αριθμό επιβατικών οχημάτων που θα προσφέρει, ενώ η εταιρεία έχει έσοδα ή ζημία από τη συγκεκριμένη πώληση.

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

main()
{
    int epivates,theseis;
    float pos;
    printf("Δώσε θέσεις και επιβάτες :");
    scanf("%d %d",&theseis,&epivates);
    pos=epivates*100.0/theseis; //υπολογισμός ποσοστού πληρότητας
    if (pos>=50)
```

```
printf("Κέρδος\n");
else if (pos<30)
    printf("Ζημία\n");
}
```

7.14

Ένα σιρόπι ανά μίλι με τέσσερις συνθήκες, και την ταχύτητα που έχει, μπορεί να είναι χαμηλή, μέτρια, κανονική ή υψηλή. Ο υπολογιστής του θέα,είναι η κατανάλωση (αυτή εκάστη μπορεί να χαρακτηριστεί με την πρώτη από αυτές):

Λίτρα ανά μίλι	Περιγραφή κατανάλωσης
από 0 μέχρι 0.9	πολύ χαμηλή
από 0.9 μέχρι 1.2	χαμηλή
από 1.2 μέχρι 1.8	κανονική
από 1.8 και πάνω	υψηλή

Να γράψετε πρόγραμμα το οποίο να ζητήσει το πλήθος των μιλίων που διένευσ το σιρόπι, την ταχύτητα (λίτρα που καταναλώνει ανά μίλι) και να εμφανίσει τη περιγραφή της κατανάλωσης σύμφωνα με τα παραπάνω πινάκω. **

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

main()
{
    float milia,litra,kat;
    printf("Δώσε μίλια και λίτρα :");
    scanf("%f %f",&milia,&litra);
    kat=litra/milia;           //υπολογισμός κατανάλωσης λίτρων ανά μίλι
    if (kat<=0.9)
        printf("Πολύ χαμηλή\n");
    else if (kat<=1.2)
        printf("Χαμηλή\n");
    else if (kat<=1.8)
        printf("Κανονική\n");
    else
        printf("Υψηλή\n");
}
```

Ασκήσεις Κεφαλαίου 8

8.1 Να γραφεί πρόγραμμα το οποίο να υπολογίζει το άθροισμα των αριθμών από 1 μέχρι το 1000. 

Με χρήση της εντολής **while** ...

```
main()
{
    int a,sum;
    a=sum=0;
    while(a<=1000)
    {
        sum=sum+a;
        a++;
    }
    printf("Το άθροισμα είναι %d\n",sum);
}
```

Κάθε φορά που εκτελείται η πρόταση **sum=sum+a**, στη **sum** προστίθεται η νέα τιμή της **a**.

Με χρήση της εντολής **for** ...

```
main()
{
    int a,sum;
    sum=0;
    for(a=0;a<=1000;a++)
    {
        sum=sum+a;
    }
    printf("Το άθροισμα είναι %d\n",sum);
}
```

8.2 Να γραφεί πρόγραμμα το οποίο να διαβάζει συνεχώς χαρακτήρες από το πληκτρολόγιο. Όταν πατηθεί το ESC, να σταματήσει και να εκτυπώσει το πλήθος των εισαγμένων και το αριθμό των εισαγμένων χαρακτήρων που είναι αριθμοί ή-letters. 

```
main()
{
    char ch;
    int lt=0,gr=0;
    do
    {
        ch=getch();
        if((ch>='A' && ch<='Ω') || (ch>='α' && ch<='ω'))
            ++gr;
        if((ch>='A' && ch<='Z') || (ch>='a' && ch<='z'))
            ++lt;
    } while(ch!=27);
}
```

```
printf("%d ελληνικοί χαρακτήρες\n",gr);
printf("%d λατινικοί χαρακτήρες\n",lt);
}
```

8.3

Πρόγραμμα που διαβάζει συνεχώς χαρακτήρες μέχρι να πατηθεί το <esc> (ASCII=27) και εμφανίζει το πλήθος των διαβασμένων χαρακτήρων.

```
main()
{
    char ch;
    int a,fl=0;
    ch=1;
    a=0;
    while(ch!=27)
    {
        ch=getch();
        if(ch=='*') fl=1;
        if(fl==1) ++a;
    }
    printf("%d χαρακτήρες\n",a);
}
```

Μόλις εντοπίσει το πρώτο αστεράκι (*) καταχωρίζει στο fl το 1.

Όταν το fl είναι 1 (που σημαίνει ότι έχει ήδη πατηθεί ένα αστεράκι) κάθε φορά που δίνεται ένας χαρακτήρας αυξάνει το a κατά 1.



Το πρόγραμμα ζητάει συνέχεια χαρακτήρες και σταματάει όταν πατηθεί το <esc>. Θα εμφανίσει στο τέλος το πλήθος των χαρακτήρων που πληκτρολογήθηκαν μετά από ένα αστεράκι (μετράει και το αστεράκι). Π.χ αν πληκτρολογηθούν οι χαρακτήρες: afgh6*ftg89 και μετά <esc>, θα εμφανίσει το 7 διότι από το * και μετά πληκτρολογήθηκαν 7 χαρακτήρες: *ftg89 και το <esc>).

8.4

Πρόγραμμα που διαβάζει συνεχώς αριθμούς μέχρι να πατηθεί το <esc> (ASCII=27) και εμφανίζει το πλήθος των διαβασμένων αριθμών.

```
main()
{
    int a,num,num1,num2;
    num1=num2=0;
    for (a=1;a<=100;++a)
    {
        scanf("%d",&num);
        switch(num % 2)
        {
            case 0:
                ++num2;
                break;
            case 1:
                ++num1;
                break;
        }
    }
    printf("num1=%d\n num2=%d\n",num1,num2);
}
```

Οι προτάσεις της for θα εκτελεστούν 100 φορές.

Η scanf() ζητάει έναν αριθμό και τον καταχωρεί στη μεταβλητή num.

Η παράσταση num % 2 υπολογίζει το υπόλοιπο της ακέραιας διαίρεσης του αριθμού με το 2.

Αν είναι 0 σημαίνει ότι ο αριθμός είναι ζυγός. Τότε αυξάνει την μεταβλητή num2 κατά 1.

Αν είναι 1 σημαίνει ότι ο αριθμός είναι μονός. Τότε αυξάνει την μεταβλητή num1 κατά 1.

- ✎ Με απλά λόγια, το πρόγραμμα ζητάει να δώσουμε 100 αριθμούς και στο τέλος εμφανίζει πόσους από αυτούς ήταν μονοί και πόσοι ζυγοί.
- ✎ Η μεταβλητή **num1** χρησιμοποιείται για να 'μετράει' τους μονούς αριθμούς ενώ η μεταβλητή **num2** χρησιμοποιείται για να 'μετράει' τους ζυγούς αριθμούς (βλέπε σελίδα 157 του βιβλίου).

8.5

Να γράψει πρόγραμμα το οποίο να κάνει το εξής: *

- ✎ Να ζητάει συνέχεια χαρακτήρες από το πληκτρολόγιο.
- ✎ Να σταματάει μόλις πληκτρολογήσουν δύο στερράκια (*).
- ✎ Να εμφανίζει πόσοι χαρακτήρες μεταβλήθηκαν μετά τον πρώτο και του δεύτερου αστεριού.

Ενώ παύση αν δοθούν οι χαρακτήρες * και *. Αν σταματήσει (επειδή πληκτρολογήσαν δύο στερράκια) και θα εμφανιστούν αριθμοί, οι δύο χαρακτήρες που δόθηκαν είναι χαρακτήρες, όχι αστερίκια.

```
main()
{
    char ch;
    int a, fl=0;
    a=0;
    while (fl<2)
    {
        ch=getch();
        if(ch=='*') fl++;
        if(fl==1 & ch!='*') ++a;
    }
    printf("%d χαρακτήρες\n", a);
}
```

Μόλις εντοπίσει ένα αστεράκι (*) αυξάνει την **fl** κατά 1. Έτσι στο πρώτο αστεράκι η **fl** θα γίνει 1 ενώ στο δεύτερο 2.

Όταν το **fl** είναι 1 (που σημαίνει ότι έχει πατηθεί το πρώτο αστεράκι) κάθε φορά που δίνεται ένας χαρακτήρας (και δεν είναι *) αυξάνει το **a** κατά 1.

- ✎ Το πρόγραμμα ζητάει συνέχεια χαρακτήρες και σταματάει όταν δοθούν δύο στερράκια (όταν το **fl** γίνει 2). θα εμφανίσει στο τέλος το πλήθος των χαρακτήρων που πληκτρολογήθηκαν μετά από το πρώτο αστεράκι.

8.6

Να γραφεί το επόμενο πρόγραμμα: *

```
main()
{
    int i, j;
    for (i=1; i<=10; ++i)
    {
        for (j=1; j<=i; ++j)
            printf("%d\n", j);
    }
}
```

1
1
2
1
2
3
1
2
3
4
.
.

- ✎ Ο εσωτερικός βρόχος εκτελείται για τιμές του **j** από το 1 μέχρι το **i** (το οποίο καθορίζεται από τον εξωτερικό βρόχο).
- ✎ Επομένως τη πρώτη φορά που το **i** είναι 1 ο εσωτερικός βρόχος θα εμφανίσει μόνο το 1. Τη δεύτερη φορά που το **i** είναι 2 ο εσωτερικός βρόχος θα εμφανίσει το 1 και το 2. Τη τρίτη φορά το 1 το 2 και το 3 κ.ο.κ.

Την τελευταία φορά που το **i** θα είναι 10, ο εσωτερικός βρόχος θα εμφανίσει τους αριθμούς από το 1 μέχρι το 10.

8.7

Να γράψουμε πρόγραμμα το οποίο θα εμφανίζει στην οθόνη όλους τους χαρακτήρες από το 'α' μέχρι την 'z' με κωδικό 32 μέχρι το 'z' με κωδικό 122. Πρέπει πάντα να γίνεται πρόγραμμα από το 32 μέχρι το 122, ώστε να εμφανίσει σε κάθε γραμμή της οθόνης 30 χαρακτήρες. * * *

```
main()
{
    int a;
    for (a=32;a<=255;++a)
    {
        putchar(a);
    }
}
```

Η **putchar()** εμφανίζει κάθε φορά τον χαρακτήρα με ASCII κωδικό την τιμή της **a**.

και για να αλλάζει γραμμή κάθε 30 χαρακτήρες ...

```
main()
{
    int a,i=0;
    for (a=32;a<=255;++a)
    {
        putchar(a);
        i++;
        if((i%30 ==0)) putchar('\n');
    }
}
```

Η μεταβλητή **i** 'μετράει' τους χαρακτήρες που εμφανίζονται στην οθόνη.

Όταν η μεταβλητή **i** γίνεται πολλαπλάσιο του 30 ($i\%30 == 0$) τότε γίνεται αλλαγή γραμμής με την **putchar('\n')**.

8.8

Να γράψουμε πρόγραμμα το οποίο θα υπολογίζει το άθροισμα των κλάσμων $1/1 + 1/2 + 1/3 + 1/4 + 1/5 + \dots + 1/100$. * * *

```
main()
{
    int a;
    double sum;
    for (a=1;a<=100;++a)
    {
        sum=sum+1.0/a;
    }
    printf("Το άθροισμα είναι: %f\n", sum);
}
```

Το 1.0 χρειάζεται για να έχει η παράσταση **1.0/a** αποτέλεσμα τύπου **float**

Η πρόταση θα εκτελεστεί 100 φορές και κάθε φορά στη **sum** θα προστίθεται ένα νέο κλάσμα ($1/a$). Το **sum** θα περιέχει διαφορετική τιμή, από το 1 μέχρι το 100.



Βλέπε σελίδα 157 του βιβλίου.

8.9

Όσο πρέπει πρόγραμμα σε οποίο θα δοθεί ένας δυαδικός αριθμός και θα εμφανιστεί στον αντίστοιχο δυαδικό τον αριθμό. Για παράδειγμα, αν τον δώσουμε με το 5234 το δείχνουμε ότι 1010001110010. ***

```
main()
{
    int ar,p,yp;
    printf("Δώσε αριθμό:");
    scanf("%d",&ar);
    do
    {
        p=ar/2;
        yp=ar % 2;
        printf("%d",yp);
        ar=p;
    } while (p!=0);
    putchar('\n');
}
```

Στη **p** καταχωρείται το πηλίκο της διαίρεσης του αριθμού με το 2, ενώ στην **yp** το υπόλοιπο.

Στη θέση του αριθμού καταχωρείται το πηλίκο ώστε να επαναληφθεί η διαδικασία μέχρι το πηλίκο να γίνει 0.

Επιβάλλει αλλαγή γραμμής μετά από την εκτύπωση των δυαδικών ψηφίων

👉 Στη σελίδα 86 του βιβλίου αναλύεται διεξοδικά η διαδικασία μετατροπής ενός δυαδικού στον αντίστοιχο δυαδικό του.

👉 Με το παραπάνω πρόγραμμα ο δυαδικός αριθμός θα εμφανιστεί αντίστροφα, με πρώτο το λιγότερο σημαντικό του ψηφίο. Δηλαδή αν δώσουμε το 5234 αντί να δούμε το 1010001110010 θα δούμε το 0100111000101. Για να μπορέσουμε να δούμε σωστά τον αριθμό θα πρέπει να προσθέσουμε στο πρόγραμμα μας τη χρήση πινάκων η οποία όμως είναι αντικείμενο του κεφαλαίου 12.

8.10

Τι θα εμφανιστεί από το επόμενο πρόγραμμα. ***

```
main()
{
    int i,j,k=4;
    for (i=1;i<=10;i=i+2)
    {
        k++;
        for (j=1;j<5;++j)
            k=k+2;
    }
    printf("k=%d\n",k);
}
```

k=49

👉 Ο εξωτερικός βρόχος θα γίνει πέντε φορές (για i 1,3,5,7,9) ενώ ο εσωτερικός τέσσερις (για j 1,2,3,4).

👉 Η πρόταση **k++** θα εκτελεστεί πέντε φορές (γιατί ανήκει μόνο στον εξωτερικό βρόχο) ενώ η **k=k+2** είκοσι (5*4) φορές (γιατί ανήκει και στους δύο βρόχους).

👉 Κάθε φορά που εκτελείται η **k++** η **k** αυξάνει κατά 1 οπότε τελικά θα αυξηθεί κατά 5 εφόσον θα εκτελεστεί πέντε φορές. Κάθε φορά που εκτελείται η **k=k+2** η **k** αυξάνει κατά 2 οπότε τελικά θα αυξηθεί κατά 40 εφόσον θα εκτελεστεί είκοσι φορές.

👉 Η τελική τιμή του **k** θα είναι 4 + 5 + 40 = 49 (το 4 είναι η αρχική τιμή της k).

8.11

Το παρακάτω πρόγραμμα υπολογίζει το άθροισμα των ψηφίων του 1234. *******

```
main()
{
    int i, j=6, k=4;
    i=(k=k+2, j=k*10);
    printf("i=%d j=%d k=%d\n", i, j, k);
}
```

i=60 j=60 k=6

👉 Η παράσταση **k=k+2, j=k*10** έχει σαν αποτέλεσμα να γίνουν οι παραστάσεις **k=k+2** και **j=k*10** (οπότε στο j καταχωρείται το 60). Το αποτέλεσμα της όλης παράστασης είναι η τιμή της τελευταίας (δηλαδή το 60) η οποία καταχωρείται στο **i** (βλέπε σελίδα 153 του βιβλίου).

8.12

Το παρακάτω πρόγραμμα θα σταματήσει να δουλεύει όταν ο τυχαίος αριθμός που θα δούμε, εκτελείς αριθμούς, δεν θα έχουμε ποτέ, *******

```
main()
{
    int a;
    do
    {
        a=rand();
        if(a>=100) continue;
        printf("%d\n", a);
    } while(a!=0);
}
```

Στην **a** θα καταχωρηθεί ένας τυχαίος αριθμός.

Αν ο αριθμός είναι μεγαλύτερος ή ίσος με το 100 τότε η επόμενη **printf()** παρακάμπτεται και η διαδικασία επαναλαμβάνεται.

Η επαναληπτική διαδικασία θα σταματήσει όταν η **a** αποκτήσει τιμή 0.

👉 Το πρόγραμμα εμφανίζει τυχαίους αριθμούς μικρότερους από το 100.
 👉 Θα σταματήσει όταν ο τυχαίος αριθμός που επιστρέφει η **rand()** είναι 0.
 👉 Ο τελευταίος αριθμός που θα δούμε είναι το 0. Δεν θα δούμε ποτέ αριθμούς μεγαλύτερους ή ίσους από το 100.

8.13

Το παρακάτω πρόγραμμα το οποίο θα διαβεί έναν αριθμό θα βάλει στο **sum** το άθροισμα των ψηφίων του. Για παράδειγμα αν τον δώσουμε το 1234 θα εμφανίσει τον αριθμό 14 (5+2+3+4). *******

```
main()
{
    int ar, y, p, sum=0;
    printf("Δώσε αριθμό:");
    scanf("%d", &ar);
    do
    {
        y=ar % 10;
        p=ar/10;
        sum=sum+y;
        ar=p;
    } while (p!=0);
    printf("Το άθροισμα των ψηφίων είναι %d\n", sum);
}
```

Στη μεταβλητή **y** καταχωρείται το υπόλοιπο της διαίρεσης του αριθμού με το 10, ενώ στην **p** το πηλίκο.

Η διαδικασία συνεχίζεται αντικαθιστώντας κάθε φορά το **ar** με το πηλίκο μέχρι το πηλίκο να γίνει 0. Τα υπόλοιπα που υπολογίζονται από την παράσταση **ar%10** αποτελούν τα ψηφία του αριθμού και προστίθενται στη **sum**.

👉 Βλέπε το παράδειγμα Π8.5 στη σελίδα 162 του βιβλίου.

8.14 Να γραφεί πρόγραμμα το οποίο να εμφανίζει όλους τους αριθμούς από το 1

μέχρι το 100. Οι αριθμοί να εμφανίζονται ανά δεκάδα σε κάθε γραμμή της οθόνης. Η πρώτη γραμμή π.χ. από το 1 μέχρι το 10, η δεύτερη από το 11 μέχρι το 20 κ.τ.λ. ★

```
#include <stdio.h>

main()
{
    int i;
    for (i=1;i<=100;i++)
    {
        printf("%d ",i);
        if(i%10 == 0) putchar('\n');
    }
}
```

Κάθε φορά που το *i* γίνεται πολλαπλάσιο του 10 (10,20 ...) η `putchar('\n')` εξαναγκάζει σε μια αλλαγή γραμμής. Η παράσταση `i%10` υπολογίζει το υπόλοιπο της ακέραιας διαίρεσης του *i* με το 10, και είναι 0 μόνο όταν το *i* είναι πολλαπλάσιο του 10.

8.15 Ποια από τα επόμενα αληθεύουν: ★

- ☒ Ο βρόχος **do-while** εκτελείται τουλάχιστον μία φορά.
- ☐ Στην εντολή **for** τα τρία τμήματά της πρέπει να σχετίζονται μεταξύ τους.
- ☒ Η εντολή **goto** οδηγεί σε μη δομημένα προγράμματα.
- ☒ Η πρόταση **while(1)** έχει ως αποτέλεσμα τη συνεχή εκτέλεση ενός βρόχου.
- ☐ Ο τελεστής κόμμα (,) επιστρέφει την τιμή της πρώτης παράστασης.

8.16 Δίδονται οι βαθμοί ενός μαθητή. Να γραφεί πρόγραμμα το οποίο να ζητάει από το πληκτρολόγιο τους 100 βαθμούς να εμφανίζει τον μέσο όρο τους καθώς και τα δύο μεγαλύτερα βαθμολογία με δύο δεκάδικα ψηφία. ★★

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

main()
{
    int i;
    float mo,sum,bathmos,max1,max2;
    sum=0;
    for (i=1;i<=100;i++)
    {
        printf("Δώσε βαθμό %d ->",i);
        scanf("%f",&bathmos);
        sum=sum+bathmos;
        if (i==1)
            max1=bathmos;
        else if (i==2)
            max2=bathmos;
        else
        {
            if (bathmos>max1)
            {
```

Η αρχική τιμή των **max1** και **max2** πρέπει να είναι ο πρώτος και ο δεύτερος βαθμός αντίστοιχα.

Στη περίπτωση που ο βαθμός είναι μεγαλύτερος από τον **max1**, καταχωρίζεται ο **max1** στον **max2** και ο βαθμός στον **max1**.

```

        max2=max1;
        max1=bathmos;
    }
    else if (bathmos>max2)
    {
        max2=bathmos;
    }
}
}
mo=sum/100;
printf("MO=%5.2f\n",mo);
printf("MAX1=%5.2f\n",max1);
printf("MAX2=%5.2f\n",max2);
}

```

Στη περίπτωση που ο βαθμός είναι μεγαλύτερος από τον **max2**, καταχωρίζεται στον **max2**.

8.17 Να δοθεί η συνάρτηση $f(x) = x^4 - 5x^2 + 3$ να γραφεί πρόγραμμα που να επαναλάβει τις τιμές του x από 0 μέχρι 1 με βήμα 0.05. *

```

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
main()
{
    double fx,x;
    for (x=0.0;x<=1;x=x+0.05)
    {
        fx=pow(x,4)-5*pow(x,2)+3;
        printf("x=%lf  fx=%lf\n",x,fx);
    }
}

```

8.18 Πρωτεύοντες το επόμενο πρόγραμμα. *

```

main()
{
    int i,j;
    for (i=1,j=10;i<=j;++i,j--)
    {
        printf("%d %d\n",i,j);
    }
}

```

```

1 10
2 9
3 8
4 7
5 6

```

8.19 Ένας αριθμός θεωρείται έμφανος, όταν από τον αριθμό αφαιρεθεί ο τελευταίος ψηφίος του είναι ίσο με το τετράγωνό του. Π.χ. 100000. Αν γράψουμε πρόγραμμα το οποίο να ελέγξει το ποσό που δίνει ο πελάτης στο κατάστημα και να ελέγξει αν είναι έμφανος ή όχι και να δώσει μήνυμα αν είναι έμφανος ή όχι. Να δοθεί ένα πρόγραμμα που να ελέγξει το ποσό που δίνει ο πελάτης στο κατάστημα και να ελέγξει αν είναι έμφανος ή όχι. Να δοθεί ένα πρόγραμμα που να ελέγξει το ποσό που δίνει ο πελάτης στο κατάστημα και να ελέγξει αν είναι έμφανος ή όχι. *

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

main()
{
    int poso,sum=0,plithos,max,min;
    plithos=0;
    printf("Δώσε ποσό :");
    scanf("%d",&poso);
    sum=sum+poso;
    plithos++;
    max=min=poso;
    while (sum<100000)
    {
        printf("Δώσε ποσό:");
        scanf("%d",&poso);
        sum=sum+poso;
        plithos++;
        if (poso>max) max=poso;
        if (poso<min) min=poso;
    }
    printf("Πλήθος ατόμων: %d\n",plithos);
    printf("Συνολικό ποσό %d\n",poso);
    printf("Μεγαλύτερο ποσό: %d Μικρότερο ποσό: %d\n",max,min);
}
```

Ζητείται το πρώτο ποσό έξω από την επαναληπτική διαδικασία για να δοθούν αρχικές τιμές στις μεταβλητές **max** και **min**.

8.20

Με δεδομένη τη βελτιστοποίηση αλγόριθμου, να βρούμε τα πρώτα πολλαπλάσια του n που είναι άρτια, όπου n είναι ένας αριθμός που δίνεται από τον χρήστη. Να μην συμπεριληφθούν τα ζεύγη αριθμών για τον οποίο πληροστέ α και β όταν ένας και ένα από τους δύο συντελεστές έχει τιμή 0. Σε κάθε περίπτωση να γίνεται έλεγχος για πρόκληση δι-
 υαρίσματος.

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

main()
{
    int a,b;
    float d,r1,r2;
    for (a=-50;a<=50;a++)
    {
        for (b=-50;b<=50;b++)
        {
            if (a==0 || b==0) continue;
            d=b*b-4*a*3;
            if (d>=0)
```

Στην περίπτωση που ένας από τους δύο συντελεστές είναι 0, προχωράμε στην επόμενη επανάληψη.

```

    {
        r1=(-b+sqrt(d))/(2*a);
        r2=(-b-sqrt(d))/(2*a);
        printf("a=%d b=%d r1=%f r2=%f\n",a,b,r1,r2);
    }
    else
        printf("a=%d b=%d δεν έχει πραγματικές ρίζες\n",a,b);
}
}

```

Αν η διακρίνουσα είναι μεγαλύτερη ή ίση με 0 υπάρχουν πραγματικές ρίζες, διαφορετικά όχι.

8.21

Ο πρόγραμμα που ακολουθεί μετρά τον αριθμό των κοινών διαιρετών δύο θετικών ακεραίων m και n μέσω αλγορίθμου ΕΥΚΛΕΙΔΗ.

- Βήμα 1: Θέτουμε στο m τον μεγαλύτερο και στο n τον μικρότερο αριθμό.
 Βήμα 2: Στην περίπτωση που ο ένας των αριθμών m ή n είναι πολλαπλάσιο του άλλου, τότε ο αριθμός των κοινών διαιρετών είναι ο αριθμός των κοινών διαιρετών των δύο αριθμών.
 Βήμα 3: Διαίρεση του m με το n και ανάστροφο το υπόλοιπο.
 Βήμα 4: Αν $n=0$, ο αριθμός κοινών διαιρετών και το υπόλοιπο είναι 1.
 Βήμα 5: Αλλάζουμε την τιμή του m του n και την επαναλαμβάνουμε από το βήμα 2.

Ο αλγόριθμος που ακολουθεί μετρά τον αριθμό των κοινών διαιρετών δύο θετικών ακεραίων m και n μέσω αλγορίθμου ΕΥΚΛΕΙΔΗ.

```

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

main()
{
    int m,n,mkd,r,temp;
    printf("Δώσε δύο αριθμούς :");
    scanf("%d %d",&m,&n);
    if (n>m)
    {
        temp=m;
        m=n;
        n=temp;
    }
    if (n==0)
        mkd=m;
    else
    {
        do
        {
            r=m%n;
            if (r!=0)
            {
                m=n;
                n=r;
            }
        }
    }
}

```

```
        } while (r!=0);  
        mkd=n;  
    }  
    printf("Ο ΜΚΔ είναι %d\n",mkd);  
}
```

Ασκήσεις Κεφαλαίου 9

9.1

Ορίστε τη συνάρτηση που υπολογίζει το $\frac{a+b}{2}$.

```
main()
{
    double x,y;
    printf("*****\n");
    scanf("%f %f",&x,&y);
    printf("mo=%f\n",mo(x,y));
}

float mo(int a, int b);

{
    if(a==0 && b==0)
    {
        return 0;
    }
    else
    {
        float mesos;
        mesos=(a+b)/2.0;
    }
    return mesos;
}
```

Η `scanf()` χρειάζεται τις διευθύνσεις των μεταβλητών `x` και `y`. Επίσης επειδή είναι μεταβλητές τύπου `double` θα έπρεπε στο αλφαριθμητικό ελέγχου να ήταν `%lf` και όχι `%f` ως εξής:
`scanf("%lf %lf",&x,&y);`

Οι τυπικές παράμετροι της `mo()` είναι τύπου `int`, επομένως και τα ορίσματα όταν την καλούμε θα έπρεπε να είναι τύπου `int` και όχι `double` όπως οι μεταβλητές `x` και `y`.

Ο ορισμός της συνάρτησης δεν τερματίζεται με ερωτηματικό. Επίσης δεδομένου ότι η συνάρτηση δεν είναι τύπου `int`, θα έπρεπε να υπάρχει και πρόσθια δήλωση της συνάρτησης πριν από τη `main()` (βλέπε σελίδα 181 του βιβλίου).

Η μεταβλητή `mesos` είναι άγνωστη σε αυτό το τμήμα του κώδικα. Η εμφάνιση της `mesos` περιορίζεται στη σύνθετη πρόταση της `else`.

9.2

Να γράψετε συνάρτηση που να κάνει τα εξής:

- Να ελέγχει τη σειρά των παραμέτρων.
- Αν η πρώτη παράμετρος είναι 1, να επιστρέφει το άθροισμα των δύο άλλων παραμέτρων.
- Αν η πρώτη παράμετρος είναι 2, να επιστρέφει το μέγ. το γίνεται από των δύο άλλων παραμέτρων.
- Αν η πρώτη παράμετρος είναι 3, να επιστρέφει a_1 είναι το μέγ. της από των δύο άλλων παραμέτρων.
- Αν η πρώτη παράμετρος δεν είναι ούτε 1, ούτε 2, ούτε 3, να εμφανίζεται στην οθόνη μήνυμα λάθους. Διακριτική κλήση συνάρτησης και να σταματάει το πρόγραμμα με κώδικα εξόδου 1.

Να ελέγξετε εσείς τον τύπο της συνάρτησης και να γράψετε τον κώδικα της.

```
float calc(int a, float b, float c)
{
    switch(a)
    {
        case 1:
            return b+c;
            break;
        case 2:
```

Η πρώτη παράμετρος πρέπει να είναι τύπου `int` διότι η `switch` που ελέγχει τη τιμή της συντάσσεται μόνο με ακέραιες παραστάσεις (βλέπε σελίδα 128 του βιβλίου).

```

        return b*c;
        break;
    case 3:
        return (b+c)/2;
        break;
    default:
        printf("Αντικανονική κλήση συνάρτησης\n");
        exit(1);
    }
}

```

9.3

Ο ορίστηκε να υπολογίζει το άθροισμα `total()` και να δέχεται ως παράμετρο έναν αριθμό και να επιστρέφει ως τιμή το άθροισμα των αριθμών από το 1 μέχρι το `total(250)` να επιστρέφει το άθροισμα των αριθμών από το 1 μέχρι το 250.

```

int total(int ar)
{
    int i, sum=0;
    for(i=1; i<=ar; i++)
        sum=sum+i;
    return sum;
}

```

9.4

Ο ορίστηκε να υπολογίζει τον αριθμό των χαρακτήρων που εμφανίζονται στην κλήση με `test()`.

```

int test(char ch)
{
    char a;
    if (ch=='A' || ch=='a') return 0; // 'A' και 'a' αρχίζονται
    for(a='A'; a<=ch; a++)
        printf("%c", a);
    return 1;
}

```

Παραγίνει όταν την κληθεί με `test('J')`.

Εκθαλνεί όταν την κληθεί με `test('9')`.

Όταν θα επιστρέφει τιμή 0.

- 👉 Η συνάρτηση δέχεται σαν παράμετρο έναν χαρακτήρα.
- 👉 Αν ο χαρακτήρας δεν είναι λατινικός κεφαλαίος επιστρέφει τιμή 0. Αν είναι λατινικός κεφαλαίος, εμφανίζει όλους τους χαρακτήρες ξεκινώντας από το 'A' μέχρι τον χαρακτήρα της παραμέτρου και επιστρέφει τιμή 1.
- 👉 Στη περίπτωση που κληθεί με παράμετρο 'J' θα εμφανίσει όλους τους χαρακτήρες από το 'A' μέχρι το 'J' και θα επιστρέψει τιμή 1.
- 👉 Όταν κληθεί με παράμετρο '9' δεν θα εμφανίσει τίποτα και θα επιστρέψει τιμή 0.
- 👉 Θα επιστρέψει τιμή 0 όταν ο χαρακτήρας δεν είναι λατινικός κεφαλαίος.

9.5

Να γράψετε συνάρτηση η οποία θα εμφανίζει δέκα φορές τη φράση "Κατανοήστε τη γλώσσα C". *

```
void print_it1()
{
    int i;
    for(i=1;i<=10;i++)
        printf("Κατανοήστε τη γλώσσα C\n");
}

void print_it2(int num)
{
    int i;
    for(i=1;i<=num;i++)
        printf("Κατανοήστε τη γλώσσα C\n");
}
```

Η `print_it1()` εμφανίζει τη φράση "Κατανοήστε τη γλώσσα C" 10 φορές.

Η `print_it2()` εμφανίζει τη φράση "Κατανοήστε τη γλώσσα C" τόσες φορές όσες η τιμή της παραμέτρου (num) με την οποία καλείται.

👉 Οι συναρτήσεις δηλώνονται ως τύπου **void** δεδομένου ότι δεν επιστρέφουν τιμή.

9.6 Ποια από τα παρακάτω αληθεύουν: ★

- ☐ Μια συνάρτηση που δεν επιστρέφει τιμή πρέπει **υποχρεωτικά** να δηλωθεί ως τύπου **void**.
- ☒ Για να χρησιμοποιηθεί μια συνάρτηση βιβλιοθήκης πρέπει στον κώδικα του προγράμματος να συμπεριλάβουμε με **#include** το αρχείο κεφαλίδας στο οποίο δηλώνεται.
- ☐ Αν μια συνάρτηση δεν έχει εντολή **return**, δεν επιστρέφει ποτέ στο πρόγραμμα που την κάλεσε.
- ☒ Όταν καλούμε μια συνάρτηση, ο τύπος των ορισμάτων της πρέπει να είναι αντίστοιχος με τον τύπο των παραμέτρων της.
- ☒ Συναρτήσεις που δεν είναι τύπου **int** και ορίζονται μετά τη **main()** πρέπει να δηλωθούν και πριν από αυτήν (με πρόσθια δήλωση —forward declaration).

9.7

Ο ορισμένος τύπος του John Wallis (1616-1703) αποτελεί πρόβλημα το $\frac{2}{3}, \frac{4}{5}, \frac{6}{7}, \dots$. Η πρόβλεψη πρόβλεψε το οποίο χρησιμοποιείται στη τη σελίδα και υπολογίζει την τιμή του π. Το πρόγραμμα να χρησιμοποιεί συνάρτηση που να υπολογίζει τον μέσο όρο των μέσων της σειράς. Η συνάρτηση θα έχει ως παράμετρο τον τελεστή από τον οποίον να χρησιμοποιηθούν οι 1000 πρώτοι όροι. *

$$\frac{2}{3} = \frac{2}{3} \cdot \frac{4}{5} = \frac{4}{5} \cdot \frac{6}{7} = \frac{6}{7} \cdot \frac{8}{9} = \frac{8}{9} \cdot \frac{10}{11} = \frac{10}{11} \cdot \frac{12}{13} = \frac{12}{13} \cdot \frac{14}{15} = \frac{14}{15} \cdot \frac{16}{17} = \frac{16}{17} \cdot \frac{18}{19} = \frac{18}{19} \cdot \frac{20}{21} = \frac{20}{21} \cdot \frac{22}{23} = \frac{22}{23} \cdot \frac{24}{25} = \frac{24}{25} \cdot \frac{26}{27} = \frac{26}{27} \cdot \frac{28}{29} = \frac{28}{29} \cdot \frac{30}{31} = \frac{30}{31} \cdot \frac{32}{33} = \frac{32}{33} \cdot \frac{34}{35} = \frac{34}{35} \cdot \frac{36}{37} = \frac{36}{37} \cdot \frac{38}{39} = \frac{38}{39} \cdot \frac{40}{41} = \frac{40}{41} \cdot \frac{42}{43} = \frac{42}{43} \cdot \frac{44}{45} = \frac{44}{45} \cdot \frac{46}{47} = \frac{46}{47} \cdot \frac{48}{49} = \frac{48}{49} \cdot \frac{50}{51} = \frac{50}{51} \cdot \frac{52}{53} = \frac{52}{53} \cdot \frac{54}{55} = \frac{54}{55} \cdot \frac{56}{57} = \frac{56}{57} \cdot \frac{58}{59} = \frac{58}{59} \cdot \frac{60}{61} = \frac{60}{61} \cdot \frac{62}{63} = \frac{62}{63} \cdot \frac{64}{65} = \frac{64}{65} \cdot \frac{66}{67} = \frac{66}{67} \cdot \frac{68}{69} = \frac{68}{69} \cdot \frac{70}{71} = \frac{70}{71} \cdot \frac{72}{73} = \frac{72}{73} \cdot \frac{74}{75} = \frac{74}{75} \cdot \frac{76}{77} = \frac{76}{77} \cdot \frac{78}{79} = \frac{78}{79} \cdot \frac{80}{81} = \frac{80}{81} \cdot \frac{82}{83} = \frac{82}{83} \cdot \frac{84}{85} = \frac{84}{85} \cdot \frac{86}{87} = \frac{86}{87} \cdot \frac{88}{89} = \frac{88}{89} \cdot \frac{90}{91} = \frac{90}{91} \cdot \frac{92}{93} = \frac{92}{93} \cdot \frac{94}{95} = \frac{94}{95} \cdot \frac{96}{97} = \frac{96}{97} \cdot \frac{98}{99} = \frac{98}{99} \cdot \frac{100}{101} = \frac{100}{101}$$

```
double calc(int ar);

main()
{
    double p;
```

```

    p=calc(1000)*2;
    printf("π=%f\n",p);
}

double calc(int ar)
{
    int i;
    double p1;
    p1=1.0;
    for(i=2;i<=ar;i=i+2)
        p1=p1*pow(i,2)/((i-1)*(i+1));
    return p1;
}

```

9.8

[illegible]

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

main()
{
    int apo,eos,xr;
    printf("Από έτος:");
    scanf("%d",&apo);
    printf("Έως έτος:");
    scanf("%d",&eos);
    for (xr=apo;xr<=eos;xr++)
    {
        if(disekto(xr)) printf("%d Δίσεκτο\n",xr);
    }
}

int disekto(int et)
{
    if(et%4 == 0)
    {
        if(et%100 == 0)
        {
            if(et%400 == 0)
                return 1;
            else
                return 0;
        }
        else
            return 1;
    }
    else
        return 0;
}
```

```

        return 1;
    }
    else
        return 0;;
}

```

9.9

Ο ΔΜΣ (Δείκτης Σωματικού ΔΜΣ) υπολογίζεται από την τύπο B/Y^2 , όπου B το βάρος σε κιλά και Y το ύψος σε μέτρα. Ανάλογα με την τιμή του ΔΜΣ ένα άτομο χαρακτηρίζεται σύμφωνα με τον παρακάτω πίνακα:

ΔΜΣ	Περιγραφή
μικρότερος από 18.5	Λιποβάρης
από 18.5 και μικρότερος του 25	Κανονικός
από 25 και μικρότερος του 30	Υπερβαρής
από 30 και πάνω	Πολυβαρής

Να γράψετε πρόγραμμα το οποίο θα ζητάει να πληρωθούν τα βάρος και το ύψος διαδοχικών ατόμων. Το πρόγραμμα ζητάει ΔΜΣ και θα εμφανίζεται μερικές φορές ο αριθμός που χάνει (π.χ. 18.5 ή 30). Το πρόγραμμα θα σταματάει στο δοκίμιο 0, από το οποίο το βάρος είτε είναι το 0 ή ο 0 υπολογισμός του ΔΜΣ μπορεί να εκτελεστεί είτε να συντηρηθεί. Επειδή είναι μάλλον τετριμμένη εργασία, θα πρέπει να υλοποιηθεί η είσοδος των δεδομένων. ***

```

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

float dms(float y, float b);
void display(float d);

main()
{
    float ypsos,baros;
    do
    {
        printf("Δώσε ύψος και βάρος :");
        scanf("%f %f",&ypsos,&baros);
        if (ypsos==0 || baros==0) break;
        display(dms(ypsos,baros));
    } while (1);
}

float dms(float y, float b)
{
    float d;
    d = b/(y*y);
    return d;
}

void display(float d)
{
    if (d<18.5)
        printf("Λιποβαρής\n");
    else if (d>=18.5 && d<25)

```

Πρόσθετες δηλώσεις συναρτήσεων

Στη συνάρτηση display() μεταβιβάζεται ο ΔΜΣ που επιστρέφει η συνάρτηση dms().

```

        printf("Κανονικός\n");
    else if (d>=25 && d<30)
        printf("Υπέρβαρος\n");
    else
        printf("Παχύσαρκος\n");
}

```

9.10

Με δεδομένη την ολνέονη συνάρτηση $f(x) = x^4 - 10x^2 + 2$, να γραφτεί πρόγραμμα που να υπολογίζει τις τιμές που παίρνει η συνάρτηση για τις τιμές x από 0 μέχρι 1 με βήμα 0.05. Να εμφανίζονται οι τιμές της x και της εκτιμήσεως των δεκαδικών ψηφίων. Η τιμή της x να υπολογίζεται από εκχώρηση στην μεταβλητή στο πρόγραμμα σου. ❏

```

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

float f(float xx);

main()
{
    float x;
    for (x=0.0; x<=1; x=x+0.05)
    {
        printf("x=%6.3f  fx=%6.3f\n", x, f(x));
    }
}

float f(float xx)
{
    return pow(xx, 4) - 10 * pow(xx, 2) + 2;
}

```

9.11

Ο ΦΠΑ ενός προϊόντος μπορεί να ανήκει σε τέσσερις κατηγορίες:

Κατηγορία	Ποσοστό ΦΠΑ
1	0.00
2	0.06
3	0.13
4	0.19

Να γραφτεί πρόγραμμα το οποίο θα ζητάει να πληρωθεί ο αριθμός της κατηγορίας ΦΠΑ και το κόστος ΦΠΑ για το προϊόν. Το πρόγραμμα θα πρέπει να εμφανίζει το συνολικό κόστος της αγοράς καθώς και το ποσό του ΦΠΑ για όλα τα προϊόντα που αγοράστηκαν. Ο υπολογισμός του ΦΠΑ πρέπει να γίνεται με βάση τον συνολικό αριθμό των μεταβλητών που ορίζονται μέσα στο πρόγραμμα σου. ❏

```

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

float fpa(float p, int k);

```

```
main()
{
    int i,kat_fpa,plithos;
    float timi,synoliko_kostos=0,synoliko_fpa=0,poso,poso_fpa;
    for (i=1;i<=10;i++)
    {
        printf("Δώσε πλήθος τιμη και κατ-ΦΠΑ για το προϊόν %d >",i);
        scanf("%d %f %d",&plithos,&timi,&kat_fpa);
        poso=plithos*timi;
        poso_fpa=fpa(poso,kat_fpa);
        synoliko_fpa=synoliko_fpa+poso_fpa;
        synoliko_kostos=synoliko_kostos+poso+poso_fpa;
    }
    printf("Συνολικό κόστος: %f\n",synoliko_kostos);
    printf("Συνολικό ΦΠΑ: %f\n",synoliko_fpa);
}

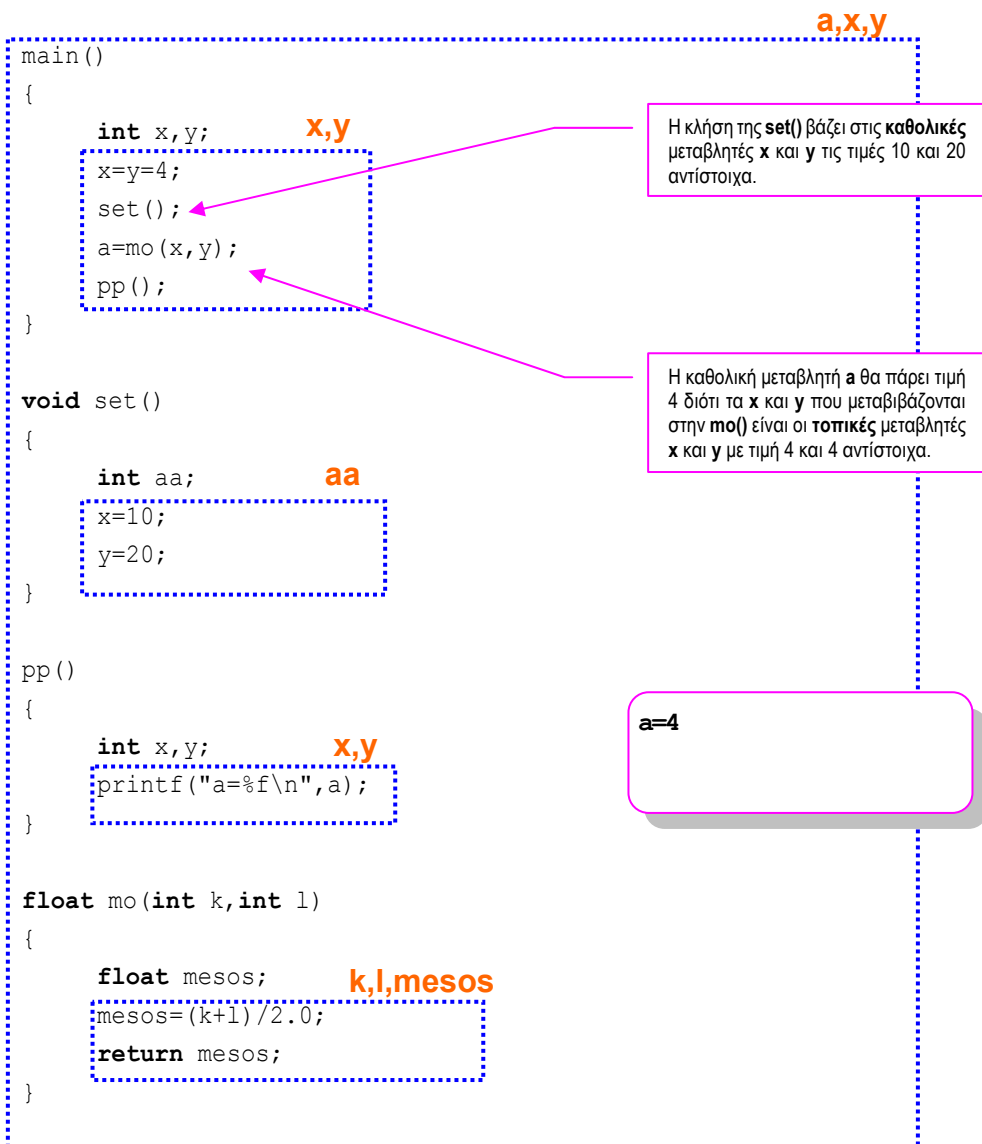
float fpa(float p, int k)
{
    float poso_fpa=0;
    switch(k)
    {
        case 1:
            poso_fpa=0;
            break;
        case 2:
            poso_fpa=p*0.06;
            break;
        case 3:
            poso_fpa=p*0.13;
            break;
        case 4:
            poso_fpa=p*0.19;
            break;
        default:
            printf("Lathos kathgoria FPA\n");
    }
    return poso_fpa;
}
```

Ασκήσεις Κεφαλαίου 10

10.1

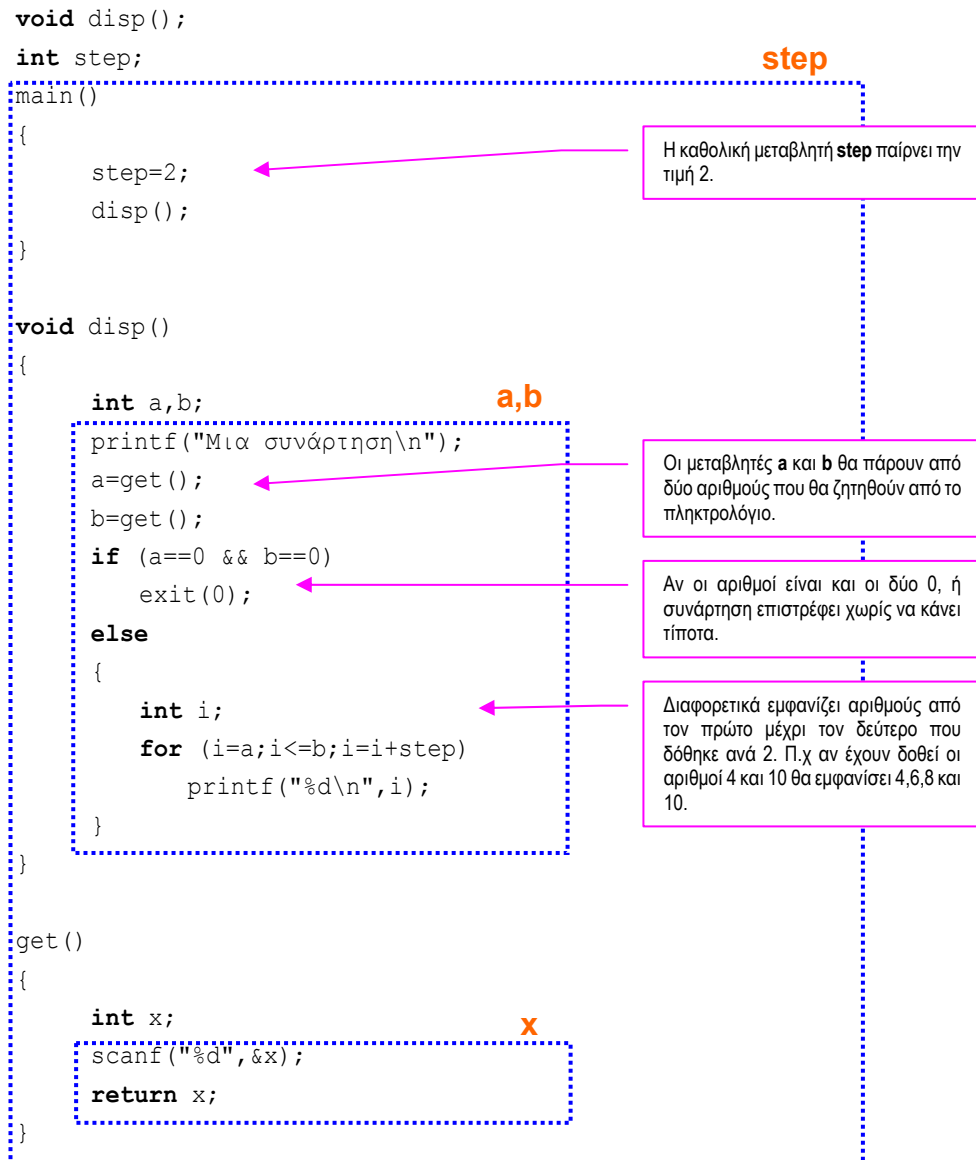
Ποια θα είναι τα αποτελέσματα του παρακάτω προγράμματος; Σημειώστε τα ποσοστά εμπέδησης κάθε μεταβλητής.

```
void set();
float a;
float mo();
int x,y;
```



10.2

Το παρακάτω πρόγραμμα στο οποίο οι μεταβλητές `a` και `b` δίδονται με ορόσημα και οι αριθμοί των μεταβλητών `x` και `y`.



👉 Με απλά λόγια το παραπάνω πρόγραμμα ζητάει δύο αριθμούς και εμφανίζει όλους τους αριθμούς από τον πρώτο που δόθηκε μέχρι τον δεύτερο ανά 2. Στην περίπτωση που και οι δύο αριθμοί είναι 0, δεν κάνει τίποτα.

10.3

Ο πρόγραμμα 10.3 δείχνει τον έλεγχο των παραμέτρων της συνάρτησης `disp()`.

```
int step;
main()
{
    step=2;
    disp(4);
}

void disp(st)
{
    int a,b,c;
    printf("Αυτή είναι μια συνάρτηση\n");
    a=get();
    c=x;
    b=get();
    if(a==0 && b==0)
        exit(0);
    else
    {
        int i;
        for(i=a;i<=b;i=i+step)
            printf("%d\n",i);
    }
    printf("i=%d\n",i);
    printf("step=%d\n",step);
}

get()
{
    float x;
    scanf("%f",&x);
    return x;
}
```

Δεν δηλώνεται ο τύπος της παραμέτρου `st`. Επίσης δεδομένου ότι η συνάρτηση `disp()` δεν είναι τύπου `int`, θα έπρεπε να υπάρχει και πρόσθια δήλωση της συνάρτησης πριν από τη `main()` (βλέπε σελίδα 181 του βιβλίου).

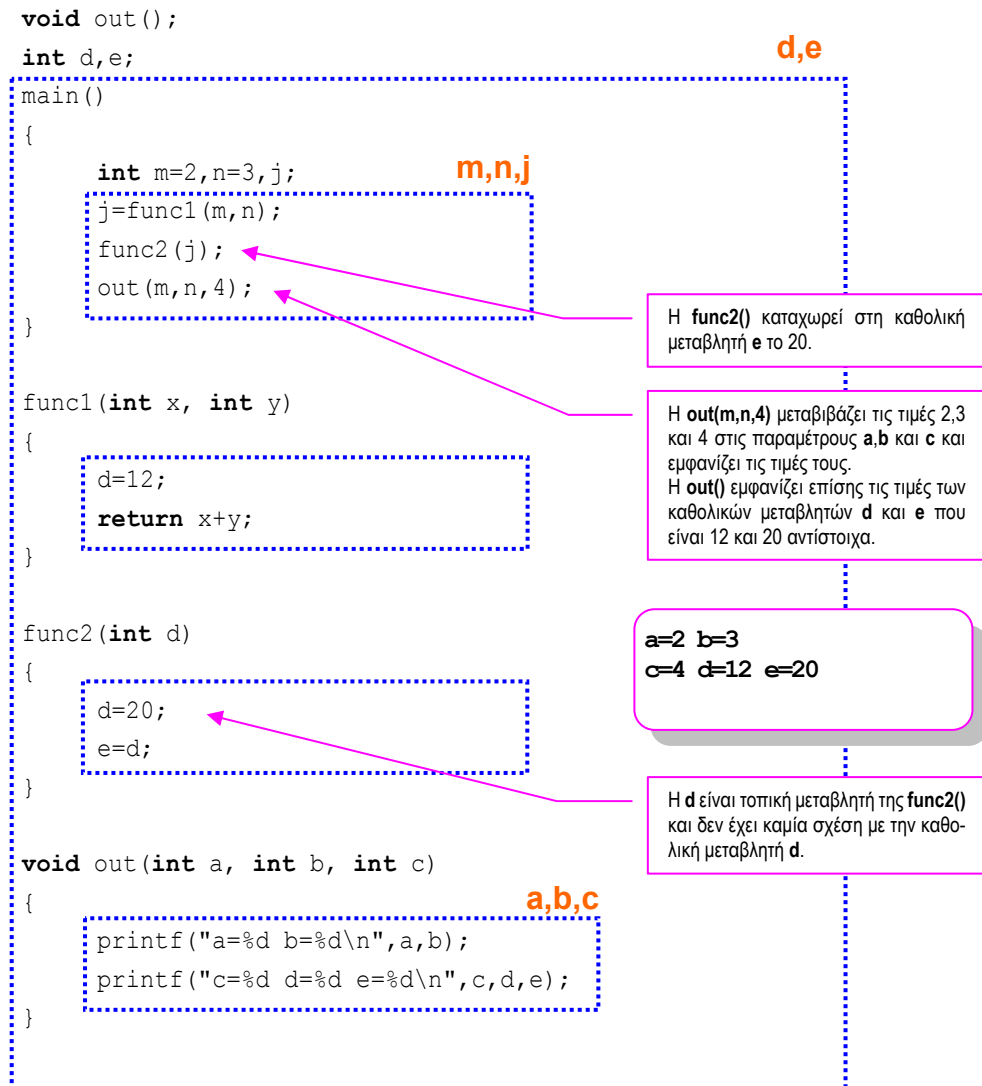
Η μεταβλητή `x` είναι άγνωστη στη συνάρτηση `disp()`. Η `x` είναι τοπική μεταβλητή της `get()`.

Η μεταβλητή `i` είναι άγνωστη σε αυτό το σημείο. Η `i` έχει εμβέλεια μόνο μέσα στη σύνθετη πρόταση της `else`.

Με τη `return x`, η `get()` επιστρέφει τιμή τύπου `float` ενώ η συνάρτηση έχει δηλωθεί τύπου `int` (όταν δεν δηλωθεί συγκεκριμένος τύπος για μια συνάρτηση, θεωρείται τύπου `int`).

10.4

Τι αποτέλεσμα θα έχει το επόμενο πρόγραμμα; Να σχεδιάσετε απεικόνιση των μεταβλητών. * *



10.5 Το static σε μια θύλη έχει το επόμενο πρόγραμμα. ***

```
int x=10;
void out1();
void out2();
void out3();

main()
{
    int i;
    for(i=1;i<=5;i++) out1();
    for(i=1;i<=5;i++) out2();
    for(i=1;i<=5;i++) out3();
}

void out1()
{
    static int x=4;
    printf("%d\n",x++);
}

void out2()
{
    static int x;
    x=4;
    printf("%d\n",x++);
}

void out3()
{
    printf("%d\n",x++);
}
```

Κάθε φορά που καλείται η **out1()** εμφανίζει τον επόμενο αριθμό από αυτόν που εμφάνισε την προηγούμενη φορά (5,6,7). Την πρώτη φορά που καλείται η **out1()**, εμφανίζει το 4.

4
5
6
7
8
4
4
4
4
4
10
11
12
13
14

Η πρόταση **static int x=4;** εκτελείται μόνο στην πρώτη κλήση της **out1()** και η **x** παίρνει αρχική τιμή 4 μόνο την πρώτη φορά (βλέπε σελίδα 207 του βιβλίου).

Κάθε φορά που καλείται η **out2()** εμφανίζει το 4 διότι κάθε φορά η **x** παίρνει την τιμή 4 (**x=4**).

Η **x** είναι καθολική μεταβλητής με αρχική τιμή 10. Κάθε φορά που εκτελείται η **out3()** εμφανίζει την τιμή της και την αυξάνει κατά 1 (10,11,12).

10.6 Στο πρόγραμμα παρακάτω, κάθε φορά που θα καλείται να επιστρέψει ως τιμή ένα χαρακτήρα μετά από τηρου, σύμφωνα με τη σχέση λογική. ***

- Την πρώτη φορά που θα κληθεί θα επιστρέψει το 'a', τη δεύτερη το 'b' και γ.ο.κ. κάθε φορά που θα κληθεί θα επιστρέφει το επόμενο γράμμα του αλφάβητου σύμφωνα με την προηγούμενη φορά. Όταν θα επιστρέψει το 'z' ο επόμενος θα είναι πάλι το 'a' κ.ο.κ.
- Στο μη κλησιμότητα θα επιστρέφει χαρακτήρα '\0' που ορίζεται ως την τελική παραμετρο.

```
char next_char()
{
    static char ch='a';
    if(ch>'z') ch='a';
    return ch++;
}
```

10.7 Ποια από τα επόμενα αληθεύουν: ★

- ☐ Η εμβέλεια μιας στατικής μεταβλητής είναι όση και μιας καθολικής μεταβλητής.
- ☒ Η παράμετρος μιας συνάρτησης αποτελεί και τοπική μεταβλητή της συνάρτησης.
- ☒ Οι στατικές μεταβλητές διατηρούν την τιμή τους ανάμεσα στις κλήσεις μιας συνάρτησης.
- ☒ Μια μεταβλητή μπορεί να δηλωθεί αμέσως μετά την αριστερή αγκύλη μιας εντολής **for**.
- ☒ Στις τοπικές μεταβλητές, μόλις λήξει η εμβέλειά τους, χάνουν ταυτόχρονα και τα περιεχόμενά τους.

10.8 Να γράψετε πρόγραμμα το οποίο θα ελάμβανε ως τρεις πρώτους αριθμούς. Πρόσθεστος ένας αριθμός ο οποίος διατηρεί μόνο με τον αριθμό που και το πολλαπλασιάζει παράδειγμα 12*7=84. Το πρόγραμμα θα πρέπει να χρησιμοποιεί μια συνάρτηση η οποία κάθε φορά που καλείται θα πρέπει να επιστρέφει τον επόμενο πρώτο αριθμό. Να μην χρησιμοποιήσετε καθολικές μεταβλητές.

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

main()
{
    int i;
    for (i=1;i<=100;i++)
    {
        printf("Επόμενος πρώτος %d \n",next_first());
    }
}

int next_first()
{
    static int num=0;
    int i,found;
    while (found)
    {
        found=0;
        num=num+1;
        for (i=2;i<=num/2;i++)
            if (num%i==0) found=1;
    }
    return num;
}
```

Κάθε φορά που καλείται η συνάρτηση **next_first()** επιστρέφει τον επόμενο πρώτο αριθμό.

Η **num** διατηρεί την τιμή της σε κάθε κλήση της συνάρτησης. Κάθε φορά αυξάνεται μέχρι να εντοπιστεί ο επόμενος πρώτος αριθμός.

Στη περίπτωση που βρεθεί αριθμός να διαιρεί ακριβώς τον **num**, ο **num** αυξάνεται κατά 1 και η διαδικασία επαναλαμβάνεται. Στη περίπτωση που εντοπιστεί αριθμός που δεν διαιρείται με κανέναν άλλο (από το 2 μέχρι το μισό του) η **found** παραμένει 0 και η διαδικασία σταματάει. Η συνάρτηση επιστρέφει τη τιμή του **num**.

10.9

Σε αυτό το πρόγραμμα χρησιμοποιούμε τη χρήση καθολικών μεταβλητών. Μπορεί να γίνει χρήση τοπικών μεταβλητών. * *

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

int num=0;

main()
{
    int i;
    for (i=1;i<=100;i++)
    {
        printf("Επόμενος πρώτος %d \n",next_first());
    }
}

int next_first()
{
    int i,found;
    while (found)
    {
        found=0;
        num=num+1;
        for (i=2;i<=num/2;i++)
            if (num%i==0) found=1;
    }
    return num;
}
```

Η **num** δηλώνεται ως καθολική μεταβλητή.

Η **num** ως καθολική πλέον μεταβλητή, διατηρεί την τιμή της σε κάθε κλήση της συνάρτησης. Κάθε φορά αυξάνεται μέχρι να εντοπιστεί ο επόμενος πρώτος αριθμός.

10.10

Σε αυτό το πρόγραμμα χρησιμοποιούμε καθολικές μεταβλητές. Οι συνάρτησεις πρέπει να χρησιμοποιούν αυτές τις μεταβλητές για να έχουν επηρεάσει την κατάσταση. * *

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

int x,y,aa,gg;
void out();

main()
{
    int m,n;
    scanf("%d %d",&m,&n);
    x=m;
    y=n;
    aa=add();
    gg=gin();
    out();
}

int add()
{

```

Οι καθολικές μεταβλητές **x,y,aa** και **gg** χρησιμοποιούνται για τη μεταβίβαση πληροφοριών στις συναρτήσεις.

Οι συναρτήσεις δεν έχουν παραμέτρους αλλά χρησιμοποιούν τις καθολικές μεταβλητές.

```

        return x+y;
    }

    int gin()
    {
        return x*y;
    }

    void out()
    {
        printf("Αθροισμα=%d\n", aa);
        printf("Γινόμενο=%d\n", gg);
    }

```

10.11 Είναι πιο σημαντικό να χρησιμοποιήσουμε καθολικές μεθόδους για τη μεταβίβαση πληροφοριών στις συναρτήσεις ή είναι καλύτερο να περάσουμε ερωτήματα με παραμέτρους μέσω παραμέτρων λειτουργίας στην ορισμένη τους;

- 👉 **Φυσικά μέσω παραμέτρων !** Με τον τρόπο αυτό μια συνάρτηση είναι ανεξάρτητη από το υπόλοιπο πρόγραμμα και δεν μπορεί ούτε να επηρεαστεί αλλά και ούτε να επηρεάσει κατά λάθος τιμές μεταβλητών του υπολοίπου προγράμματος.
- 👉 Κατ' εξαίρεση και με μεγάλη προσοχή μπορεί να χρησιμοποιήσουμε καθολικές μεταβλητές για δεδομένα στα οποία έχουν πρόσβαση σχεδόν όλες οι συναρτήσεις του προγράμματος μας για να αποφύγουμε τη χρήση πολλών παραμέτρων. Αυτό επιτρέπεται κατά εξαίρεση και μόνο σε μικρά προγράμματα στα οποία ο έλεγχος είναι σχετικά πιο εύκολος.

10.12 Να γράψετε συνάρτηση η οποία να εμφανίζει στην οθόνη ένα τετράγωνο το οποίο θα σχηματίζεται από έναν χαρακτήρα επιλογής και θα κομμάτιζόταν (σε χαρακτήρες) και οριζόντια (σε γραμμές). Το οριζόντιο τετράγωνο και ο χαρακτήρας θα μεταβιβάζονταν ως παραμέτρους στη συνάρτηση. Η συνάρτηση θα πρέπει να έχει δύο επιπλέον παραμέτρους η οποία στη περίπτωση που έχει τιμή 1 το τετράγωνο θα είναι γεμάτο με τον χαρακτήρα και στη περίπτωση που έχει τιμή 0 θα εμφανίζεται μόνο το περίγραμμα. Επίσης να γράψετε στο πρόγραμμά σας το οποίο να καλεί τη συνάρτησή σας και να εμφανίζει στην οθόνη τα τετράγωνα που ορίζονται ως εξής:

```

aaaaaaaaa
a      a
a      a
a      a
a      a
a      a
a      a
aaaaaaaaa

****
****
****
****
****

```

```

#include <stdio.h>

#include <stdlib.h>

void plaisio(int yps, int pl, char ch, int fill);

main()
{
    plaisio(7,8,'a',0);
    printf("\n");
    plaisio(5,4,'*',1);
}

```

```

}
void plaisio(int yps, int pl, char ch, int fill)
{
    int i, j;
    for (i=1; i<=pl; i++)
        printf("%c", ch);
    printf("\n");
    for (i=1; i<=yps-2; i++)
    {
        printf("%c", ch);
        for (j=1; j<=pl-2; j++)
        {
            if (fill)
                printf("%c", ch);
            else
                printf(" ");
        }
        printf("%c\n", ch);
    }
    for (i=1; i<=pl; i++)
        printf("%c", ch);
    printf("\n");
}

```

Εμφανίζεται η πρώτη γραμμή.

Εμφανίζονται οι ενδιάμεσες γραμμές. (yps-2).

Στη περίπτωση που η μεταβλητή **fill** έχει τιμή 1 οι ενδιάμεσες γραμμές γεμίζουν με τον χαρακτήρα, διαφορετικά έχουν κενά δημιουργώντας μόνο ένα περίγραμμα.

Εμφανίζεται η τελευταία γραμμή.

10.13 Παράδειγμα 10.13: Η συνάρτηση παραγοντικού (factorial) υπολογίζει το γινόμενο όλων των αριθμών από το 1 μέχρι το x. Η μεταβλητή **pp** υπολογίζεται με πολλαπλασιασμό των αριθμών από το 1 μέχρι το x. Το πρόγραμμα στο οποίο ακολουθεί η συνάρτηση παραγοντικού και η συνάρτηση **do_it** που χρησιμοποιεί τη συνάρτηση παραγοντικού, είναι το ακόλουθο:

```


#include <stdio.h>
#include <stdlib.h>
void do_it(char ch, int y);
main()
{
    int n;
    for (n=1; n<=10; n++)
        printf("%2d!= %d\n", n, paragontiko(n));
}
int paragontiko(int x)
{
    int i, pp=1;
    for (i=1; i<=x; i++) pp=pp*i;
    return pp;
}

```

Στη μεταβλητή **pp** υπολογίζεται το γινόμενο όλων των αριθμών από το 1 μέχρι το x: 1*2*3*4 *x

10.14

```
void do_it(char ch, int y)
{
    int i;
    for (i=1; i<=y; i++) printf("%c", ch);
    putchar('\n');
}
```

 Η συνάρτηση εμφανίζει τον χαρακτήρα της πρώτης παραμέτρου τόσες φορές όσο η τιμή της δεύτερης παραμέτρου. Π.χ. `do_it('*', 20)` θα εμφανίζει 20 αστεράκια στην οθόνη.

Ασκήσεις Κεφαλαίου 11

11.1 Τοποθετήστε τον κώδικα που μεταβλητές **a**, **b** και **c** μετά το τέλος του επόμενου προγράμματος:

```
main()
{
    int a,b,c,*m,*p;
    a=100;
    b=50;
    m=&a;
    p=&b;
    c=*p + *m;
    (*p)++;
    p=m;
    (*p)--;
}
```

Ο δείκτης **m** θα πάρει τη διεύθυνση της **a** και ο **p** τη διεύθυνση της **b**.

Η **c** θα πάρει τιμή 150 (100+50) δεδομένου ότι η παράσταση ***p** αναφέρεται στη μεταβλητή **a** και η παράσταση ***m** αναφέρεται στη μεταβλητή **b**.

Η παράσταση αυτή θα αυξήσει κατά 1 την θέση στην οποία 'δείχνει' ο δείκτης **p** (δηλαδή την **b**). Μετά από την πρόταση αυτή, η τιμή της **b** θα είναι 101.

Στο δείκτη **p** καταχωρείται το περιεχόμενο του δείκτη **m** δηλαδή η διεύθυνση της **a**. Τώρα και οι δύο δείκτες 'δείχνουν' στη μεταβλητή **a**. Η πρόταση **(*p)--** θα μειώσει το περιεχόμενο της **a** κατά 1.

Μεταβλητή	Τιμή
a	99
b	51
c	150

11.2 Τοποθετήστε τον κώδικα που έχετε συμπληρώσει:

```
main()
{
    char *p;
    int a=5,b=10;
    p="a,b=%d,%d\n";
    printf(p,a,b);
}
```

Τι θα έκανε αν αντί για,

```
printf(p,a,b);
```

είχαμε

```
printf(p+4,a,b);
```


11.3 Να συμπληρωθεί το επόμενο πρόγραμμα

```
main()
{
    char *p;
    p="αρνάκι άσπρο και παχύ";
    .....
}
```

Να συμπληρωθεί κατάλληλα, ώστε να ζητάει ένα γράμμα και μετά να μετράει πόσες φορές υπάρχει το γράμμα αυτό στο σύνολο χαρακτήρων "αρνάκι άσπρο και παχύ". ★★

```
main()
{
    char *p, ch;
    int ar=0;
    p="αρνάκι άσπρο και παχύ";
    ch=getch();
    while (*p!='\0')
    {
        if (*p==ch) ar++;
        p++;
    }

    printf("Το γράμμα %c υπάρχει %d φορές\n", ch, ar);
}
```

Η επαναλαμβανόμενη διαδικασία θα σταματήσει όταν ο δείκτης **p** φτάσει να 'δείχνει' στον χαρακτήρα τερματισμού '\0'.

Η μεταβλητή **ar** 'μετράει' τους χαρακτήρες της συμβολοσειράς που ισούνται με τον χαρακτήρα **ch** που πληκτρολογήθηκε.

Αυξάνει τον δείκτη **p** ώστε να δείχνει στον επόμενο χαρακτήρα.

👉 Ο δείκτης **p** αρχικά περιέχει τη διεύθυνση της πρώτης θέσης μνήμης που καταλαμβάνει η συμβολοσειρά "αρνάκι άσπρο και παχύ". Δηλαδή 'δείχνει' στον πρώτο χαρακτήρα της συμβολοσειράς.

👉 Να έχουμε υπ' όψη ότι κάθε συμβολοσειρά τερματίζεται με τον χαρακτήρα τερματισμού '\0'.

11.4 Να συμπληρωθεί στην οθόνη κατά το εκτέλεση το επόμενο πρόγραμμα. ★★

```
main()
{
    char *p;
    p="αρνάκι άσπρο και παχύ";
    while (*p!='\0')
    {
        if (*p==' ')
            putchar('\n');
        else
            putchar(*p);
        p++;
    }
}
```

αρνάκι
άσπρο
και
παχύ

Όταν ο δείκτης **p** 'δείξει' σε χαρακτήρα διαστήματος τότε αλλάζει γραμμή στην οθόνη, διαφορετικά εμφανίζει τον χαρακτήρα που 'δείχνει' ο **p**.

Αυξάνει τον δείκτη **p** ώστε να δείχνει στον επόμενο χαρακτήρα.

11.5 Μαθαίνουμε το επόμενο πρόγραμμα

```
main()
{
    int a,b,c,*p1,*p2,*p3;
    p1=&a;
    p2=&b;
    p3=&c;
    c=a+b;
}
```

Να σημειωθεί πως και η δήλωση, όπως και η χρήση της **scanf()** απαιτούνται να τονίζεται στις μεταβλητές **a** και **b** αντίστοιχα. Η πρόταση **c=a+b;** το θροισμα των **a** και **b** το οποίο καταχωρίζεται στη **c** και να επισημανθεί ότι περιέχει μόνο τις **c** στην οθόνη. Σε κάθε περίπτωση από τις νέες προτάσεις δεν πρέπει να εμφανίζονται το άθροισμα των μεταβλητών **a**, **b** και **c**.

```
main()
{
    int a,b,c,*p1,*p2,*p3;
    p1=&a;
    p2=&b;
    p3=&c;
    scanf("%d %d",p1,p2);
    *p3=*p1 + *p2;
    printf("Το περιεχόμενο της c είναι %d\n",*p3);
}
```

Οι μεταβλητές **p1** και **p2** περιέχουν τις διευθύνσεις των μεταβλητών **a** και **b** αντίστοιχα.

Το ***p3** αναφέρεται στη μεταβλητή **c**, το ***p1** αναφέρεται στη μεταβλητή **a** και , το ***p2** αναφέρεται στη μεταβλητή **b**. Η πρόταση αυτή είναι ισοδύναμη με την **c=a+b;**

11.6 Μαθαίνουμε την επόμενη πρόταση, στην οποία θέτουμε τον ταινία με την αντίστοιχη επεξεργασία *

```
int a,b,c,*p,*m;
```

Πρόταση	Ερμηνεία
p=&a	Καταχώρισε στην μεταβλητή δείκτη p τη διεύθυνση της μεταβλητής a
*p=23	Καταχώρισε στην μεταβλητή που δείχνει ο δείκτης p (δηλαδή στην a) το 23.
m=&b	Καταχώρισε στην μεταβλητή δείκτη m τη διεύθυνση της μεταβλητής b
c=*p + *m	Πρόσθεσε το περιεχόμενο της μεταβλητής στην οποία δείχνει ο p (της a) με το περιεχόμενο της μεταβλητής στην οποία δείχνει ο m (της b) και το αποτέλεσμα καταχώρισέ το στη μεταβλητή c .
m=p	Καταχώρισε στην μεταβλητή δείκτη m το περιεχόμενο της μεταβλητής δείκτη p . Τώρα και οι δύο μεταβλητές δείκτη m και p 'δείχνουν' στη μεταβλητή a .
p++	Αύξησε το περιεχόμενο της μεταβλητής δείκτη p κατά 4! (βλέπε αριθμητική των δεικτών σελίδα 226 του βιβλίου).
(*p)++	Αύξησε το περιεχόμενο της μεταβλητής στην οποία 'δείχνει' ο δείκτης p (δηλαδή της a) κατά 1. Ισοδυναμεί με την πρόταση a++ ;
m=120	Λάθος. Σε μία μεταβλητή δείκτη μπορούμε να καταχωρήσουμε μόνο διεύθυνση και όχι μια αριθμητική σταθερά.

11.7

Ποιο αποτέλεσμα στην οθόνη θα το αποκτήσει το πρόγραμμα; *

```
main()
{
    char *p,*m;
    p="αρνάκι άσπρο και παχύ";
    m=p;
    while(*p!='\0') p++;
    --p;
    while(p>=m) putchar(*(p--));
}
```

ύχαπ ιακ ορπόά ικόνρα

Η πρόταση αυτή αυξάνει τον δείκτη **p** μέχρι να 'δείξει' τον χαρακτήρα τερματισμού '\0'.

Ο δείκτης **p** μειώνεται ώστε να δείχνει τον τελευταίο χαρακτήρα.

Η πρόταση αυτή εμφανίζει τους χαρακτήρες από τον τελευταίο μέχρι τον πρώτο.

☞ Το πρόγραμμα εμφανίζει τους χαρακτήρες της συμβολοσειράς με αντίστροφη σειρά: από τον τελευταίο στον πρώτο.

☞ Η μεταβλητή **m** χρησιμοποιείται για να 'κρατήσει' την αρχική διεύθυνση της συμβολοσειράς, δεδομένου ότι τον δείκτη **p** τον μεταβάλλουμε.

11.8

Ποιο από τα ακόλουθα αποτελέσματα θα αποκτήσει το πρόγραμμα; *

```
int a,b,c=0,*ptr,*m;
a=b=10;
1 ptr=&a;
  *ptr=34;
2 m=&b;
3 c=*ptr + *m;
  ptr=m;
4 *ptr=100;
5 m++;
```

Στη μεταβλητή δείκτη **ptr** καταχωρείται η διεύθυνση της **a** (1000).

Στη μεταβλητή που δείχνει ο δείκτης **ptr** (δηλαδή στην **a**) καταχωρείται το 34.

Στη μεταβλητή δείκτη **m** καταχωρείται η διεύθυνση της **b** (3000).

Ισοδυναμεί με **c=a+b**; Στη **c** θα καταχωρηθεί το 44

Η **ptr** περιέχει τώρα τη διεύθυνση της **b**. Ισοδυναμεί **b=100**;

Επειδή ο δείκτης **m** είναι τύπου **int**, θα αυξηθεί κατά 4.

1	2	3	4	5
<div>διευθύνσεις</div> <div> <div>a</div><div>10</div><div>1000</div> </div> <div> <div>ptr</div><div>1000</div><div>2000</div> </div> <div> <div>b</div><div>10</div><div>3000</div> </div> <div> <div>c</div><div>0</div><div>4000</div> </div> <div> <div>m</div><div></div><div>5000</div> </div>	<div> <div>a</div><div>34</div><div>1000</div> </div> <div> <div>ptr</div><div>1000</div><div>2000</div> </div> <div> <div>b</div><div>10</div><div>3000</div> </div> <div> <div>c</div><div>0</div><div>4000</div> </div> <div> <div>m</div><div>3000</div><div>5000</div> </div>	<div> <div>a</div><div>34</div><div>1000</div> </div> <div> <div>ptr</div><div>1000</div><div>2000</div> </div> <div> <div>b</div><div>10</div><div>3000</div> </div> <div> <div>c</div><div>44</div><div>4000</div> </div> <div> <div>m</div><div>3000</div><div>5000</div> </div>	<div> <div>a</div><div>34</div><div>1000</div> </div> <div> <div>ptr</div><div>3000</div><div>2000</div> </div> <div> <div>b</div><div>100</div><div>3000</div> </div> <div> <div>c</div><div>44</div><div>4000</div> </div> <div> <div>m</div><div>3000</div><div>5000</div> </div>	<div> <div>a</div><div>34</div><div>1000</div> </div> <div> <div>ptr</div><div>3000</div><div>2000</div> </div> <div> <div>b</div><div>100</div><div>3000</div> </div> <div> <div>c</div><div>44</div><div>4000</div> </div> <div> <div>m</div><div>3004</div><div>5000</div> </div>

11.9 Ποια από τα επόμενα αληθεύουν: ★

- ☑ Ένας δείκτης μπορεί να περιέχει **μόνο** τη διεύθυνση μιας θέσης μνήμης.
- ☑ Ο τελεστής * χρησιμοποιείται για να έχουμε πρόσβαση σε μια θέση μνήμης μέσω ενός δείκτη ο οποίος περιέχει τη διεύθυνσή της.
- ☐ Αν εφαρμόσουμε τον τελεστή ++ σε μια μεταβλητή δείκτη, αυτή αυξάνεται κατά 1.
- ☑ Το μέγεθος μιας μεταβλητής δείκτη εξαρτάται από το σύστημα στο οποίο εκτελείται το πρόγραμμά μας.
- ☐ Μια μεταβλητή δείκτη **char** και μια μεταβλητή δείκτη **int** έχουν διαφορετικό μέγεθος.
- ☑ Στη C η διαβίβαση με αναφορά επιτυγχάνεται μέσω παραμέτρων - δεικτών.

11.10 Η θα εμφανιστεί στην οθόνη από το επόμενο πρόγραμμα: ★★

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

main()
{
    char *ptr;
    ptr="Η γλώσσα C σε βάθος";
    printf("%d\n", space(ptr));
}

int space(char *p)
{
    int c=0;
    while(*p != '\0')
    {
        if(*p == ' ') c++;
        p++;
    }
    return c;
}
```

Η συμβολοσειρά περιέχει τέσσερα κενά διαστήματα.

Η συνάρτηση **space()** μετράει το πλήθος των χαρακτήρων του διαστήματος που υπάρχουν στη συμβολοσειρά η διεύθυνση της οποίας διαβιβάζεται στη παράμετρο **p**. Στη συγκεκριμένη περίπτωση η συνάρτηση θα επιστρέψει τον αριθμό 4 ο οποίος θα εμφανιστεί στην οθόνη.

11.11 Το γράφει στην οθόνη ή θα γίνει να δαχτύλο σε περίπτωση που τη διεύθυνση του συμβολοσειράς θα να εμφανίσει μόνο τους αριθμητικούς χαρακτήρες (0-9) και θα εμφανιστεί στην οθόνη ο αριθμός: ★★

```
int digits(char *p)
{
    int c=0;
    while(*p != '\0')
    {
        if(*p >='0' && *p <='9') putchar(*p);
        p++;
    }
    return c;
}
```

Ο χαρακτήρας εμφανίζεται μόνο στη περίπτωση που είναι αριθμητικός (μεταξύ 0 και 9).

11.12 Πράγματι παρακάτω συνάρτηση: ***

```
int is_same(char *p1, char *p2)
{
    int yes=1;
    do
    {
        if (*p1!=*p2) yes=0;
    } while(*(p1++) != '\0' && *(p2++) !='\0');
    return yes;
}
```

Η συνάρτηση δέχεται ως παραμέτρους διευθύνσεις σε δύο συμβολοσειρές. Η συνάρτηση επιστρέφει τη τιμή 1 στη περίπτωση που οι δύο συμβολοσειρές περιέχουν ακριβώς τους ίδιους χαρακτήρες διαφορετικά επιστρέφει τιμή 0.

Για παράδειγμα, εάν η συνάρτηση `is_same()` κληθεί από τον παρακάτω κώδικα:

```
main()
{
    printf("%d\n", is_same("Νίκος", "Νίκος"));
    printf("%d\n", is_same("Νίκας", "Νίκος"));
}
```

Θα εμφανίζει τη τιμή 1 στην πρώτη κλήση της και την τιμή 0 στη δεύτερη.

Ασκήσεις Κεφαλαίου 12

12.1

Γεγονότα: Ορίστε `lexi1` και `lexi2` δύο πίνακες που ορίζονται ως εξής:

```
main()
{
    char lexi1[80], lexi2[80];
    gets(lexi1);
    gets(lexi2);
}
```

- Να γραφεί συνάρτηση η οποία να εμφανίζει στην οθόνη τα κοινά γράμματα των δύο λέξεων από μία φορά το καθένα.
- Να γραφεί συνάρτηση η οποία να διαγράφει από τον `lexi1` όσους χαρακτήρες περιέχονται στον `lexi2`.
- Να γραφεί συνάρτηση η οποία να βρίσκει αν η λέξη `lexi2` υπάρχει μέσα στη `lexi1`. Να επιστρέφει 0 αν δεν υπάρχει και, αν υπάρχει, να επιστρέφει τον αριθμό της θέσης μνήμης του `lexi1` από την οποία αρχίζει η `lexi2`.

//Εύρεση κοινών χαρακτήρων

```
void common(char pin1[], char pin2[])
```

```
{
    char koina[80]="";
    int i,j,k,found;
    for(i=0;i<strlen(pin1);i++)
    {
        for(j=0;j<strlen(pin2);j++)
        {
            found=0;
            if(pin1[i]==pin2[j])
            {
                for(k=0;k<strlen(koina);k++)
                    if(pin1[i]==koina[k]) found=1;
                if(!found)
                {
                    koina[strlen(koina)]=pin1[i];
                    koina[strlen(koina)+1]='\0';
                }
            }
        }
    }
    puts(koina);
}
```

Ο πίνακας `koina[]` χρησιμοποιείται για την καταχώριση των κοινών χαρακτήρων των δύο λέξεων.

Συγκρίνεται κάθε χαρακτήρας του πίνακα `pin1[]` με όλους τους χαρακτήρες του πίνακα `pin2[]`.

Στην περίπτωση που ο χαρακτήρας είναι κοινός, τότε ελέγχεται αν ήδη υπάρχει στον πίνακα `koina[]`. Αν δεν υπάρχει καταχωρείται στον `koina[]` διαφορετικά δεν γίνεται τίποτα.

- 👉 Η παραπάνω συνάρτηση ελέγχει έναν-έναν χαρακτήρα του πίνακα `pin1[]` με όλους τους χαρακτήρες του πίνακα `pin2[]`. Αν βρεθεί κοινός χαρακτήρας τον καταχωρίζει στον πίνακα `koina[]`.

👉 Για να εξασφαλίσουμε ότι οι κοινοί χαρακτήρες εντοπίζονται μόνο μία φορά και δεν υπάρχουν διπλοί στον πίνακα **koina[]**, πριν από την καταχώριση τους στον πίνακα ελέγχεται αν ο χαρακτήρας υπάρχει ήδη στον πίνακα. Αν υπάρχει **δεν** καταχωρίζεται για δεύτερη φορά.

//Διαγραφή χαρακτήρων

```
void del(char pin1[], char pin2[])
{
    int i, j, k;
    for(i=0; i<strlen(pin1); i++)
    {
        for(j=0; j<strlen(pin2); j++)
        {
            if(pin1[i]==pin2[j])
            {
                for(k=i; k<strlen(pin1); k++)
                    pin1[k]=pin1[k+1];

            }
        }
    }
}
```

Στη περίπτωση που βρεθεί ο στον πίνακα **pin2[]**, θα πρέπει να διαγραφεί.

Όλοι οι χαρακτήρες μετά από αυτόν τον χαρακτήρα και μέχρι το τέλος του πίνακα **pin1[]** αντιγράφονται στην προηγούμενη θέση.

👉 Η παραπάνω συνάρτηση ελέγχει έναν-έναν χαρακτήρα του πίνακα **pin1[]** με όλους τους χαρακτήρες του πίνακα **pin2[]**. Αν βρεθεί κοινός χαρακτήρας τον διαγράφει από τον πίνακα **pin1[]**.

👉 Μόλις εντοπιστεί ο χαρακτήρας που πρόκειται να διαγραφεί, ακολουθείται η ακόλουθη διαδικασία: Όλοι οι χαρακτήρες από μετά από αυτόν τον χαρακτήρα και μέχρι το τέλος του πίνακα **pin1[]** αντιγράφονται στην προηγούμενη θέση. Π.χ αν είναι να διαγραφεί ο χαρακτήρας στο **pin1[6]**, θα αντιγραφεί ο **pin1[7]** στον **pin1[6]**, ο **pin1[8]** στον **pin1[7]**, ο **pin1[9]** στον **pin1[8]** κ.ο.κ μέχρι το τέλος των χαρακτήρων του **pin1[]**. Με αυτόν τον τρόπο διαγράφεται ο χαρακτήρας και μετακινούνται όλοι οι υπόλοιποι μια θέση πιο πάνω.

//Εντοπισμός χαρακτήρων

```
int find(char pin1[], char pin2[])
{
    int i, j, k, found;
    for(i=0; i<strlen(pin1); i++)
    {
        found=i+1;
        for(j=0; j<strlen(pin2); j++)
        {
            if(pin1[i+j]!=pin2[j])
            {
                found=0;
                break;
            }
        }
        if(found) return found;
    }
}
```

Η **found** παίρνει αρχική τιμή **i+1** υποδεικνύοντας τον α/α της θέσης μνήμης από την οποία ξεκινάει κάθε φορά η αναζήτηση (την πρώτη φορά 1, τη δεύτερη 2 κ.ο.κ).

Ξεκινώντας από τη θέση **i** του πίνακα **pin1[]** συγκρίνει μία-μία τις θέσεις μνήμης του **pin2[]**, μέχρι να βρει διαφορετικό χαρακτήρα ή μέχρι να τελειώσει ο **pin2[]**.

Στην περίπτωση που η **found** δεν έχει γίνει 0 στον προηγούμενο βρόχο, σημαίνει ότι εντοπίστηκε το σύνολο χαρακτήρων του **pin2[]** μέσα στον **pin1[]** και μάλιστα στη θέση **found**.

```

    }
    return 0;
}

```

👉 Η παραπάνω συνάρτηση ξεκινάει από κάθε ένα χαρακτήρα του πίνακα `pin1[]` και ελέγχει όλους τους χαρακτήρες του πίνακα `pin2[]` να εντοπίσει αν συμπίπτουν με τους αντίστοιχους του πίνακα `pin1[]`. Π.χ. αν ξεκινήσει από τον `pin1[5]` θα συγκρίνει τον `pin1[5]` με τον `pin2[0]`, τον `pin1[6]` με τον `pin2[1]`, τον `pin1[7]` με τον `pin2[2]` κ.ο.κ, μέχρι να βρει διαφορετικό χαρακτήρα ή να τελειώσει ο `pin2[]`.

👉 Όταν τελειώσει μια τέτοια σύγκριση χωρίς να βρεθεί διαφορετικός χαρακτήρας σημαίνει ότι εντοπίστηκε η συμβολοσειρά του `pin2[]` μέσα στον `pin1[]`.

12.2

Η παρακάτω συνάρτηση με όνομα `convert()` θα στείλει να μετατρέπεται τους κεφαλαίους χαρακτήρες ενός πεζού χαρακτήρα σε πεζό, ή κεφαλαίους ή πεζούς σε κεφαλαίους όπου περιόριστος. * * *

```

int convert(char *str, int sel)

```

Η παράμετρος `str` είναι ο δείκτης σε `char` και δείχνει στον πίνακα χαρακτήρων που θα μετατραπεί. Η `sel` καθορίζει τη κατεύθυνση της αλλαγής ως εξής:

- Αν η `sel` έχει τιμή 1, η συνάρτηση θα μετατρέψει τα πεζά σε κεφαλαία.
- Αν η `sel` έχει τιμή 0, η συνάρτηση θα μετατρέψει τα κεφαλαία σε πεζά.
- Η συνάρτηση θα επιστρέφει ως τιμή τον αριθμό των χαρακτήρων που μετατράπηκαν.

```

{
    int cnt=0;
    while(*str!='\0')
    {
        if(*str>='a' && *str<='z' && sel==1)
        {
            *str=*str-32;
            cnt++;
        }
        if(*str>='A' && *str<='Z' && sel==0)
        {
            *str=*str+32;
            cnt++;
        }
        str++;
    }
    return cnt;
}

```

Η μεταβλητή `cnt` 'μετράει' τους χαρακτήρες που μετατράπηκαν.

Στην περίπτωση που ο χαρακτήρας είναι πεζός, και η `sel==1`, αφαιρεί το 32 μετατρέποντας τον σε κεφαλαίο (οι κωδικοί των πεζών και των κεφαλαίων χαρακτήρων διαφέρουν κατά 32).

Στην περίπτωση που ο χαρακτήρας είναι κεφαλαίος, και η `sel==0`, προσθέτει το 32 μετατρέποντας τον σε πεζό.

Ο δείκτης `str` αυξάνεται ώστε να δείχνει στον επόμενο χαρακτήρα. Η επαναλαμβανόμενη διαδικασία θα σταματήσει όταν ο δείκτης `str` 'δείξει' στο τέλος των χαρακτήρων (δηλαδή στο '\0').

12.3

Η παρακάτω συνάρτηση με όνομα `blablabla()` θα επανασυνθέσει τις λέξεις. * * *

```

char *blablabla(char *str1, char *str2, int num)

```

```

{
    int i=0;

```



```

while((str1[i] != '\0') && (i<num))
{
    str2[i]=str1[i];
    i++;
}
str2[num]='\0';
return str2;
}

```

👉 Η συνάρτηση 'αντιγράφει' τους **num** πρώτους χαρακτήρες του πίνακα **str1[]** στον **str2[]**. Τερματίζει τον **str2[]** μετά τον **num** χαρακτήρα και επιστρέφει έναν δείκτη στον **str2[]**.

👉 Π.χ αν ο πίνακας **str1[]** περιέχει τη συμβολοσειρά "abcdeDFGGG123", και ο πίνακας **str2[]** τη συμβολοσειρά "nikosMITILINI", και καλέσουμε τη συνάρτηση **blablabla(str1,str2,5)**, ο πίνακας **str2[]** θα γίνει "abcde" και η συνάρτηση θα επιστρέψει έναν δείκτη στον πίνακα **str2[]**. Δηλαδή με τη πρόταση **puts(blablabla(str1,str2,5))** θα εμφανιζόταν οι χαρακτήρες "abcde" στην οθόνη.

12.4

👉 Π.χ αν κάποιος καλέσει τη συνάρτηση **blablabla()** της παρακάτω μορφής, όπου το πρώτο ορίσμα που ακολουθεί καλό είναι επιλεγεί "Παπατρέ", τότε η συνάρτηση θα επιστρέψει " * * *".

```

main()
{
    char lexi1[40], lexi2[40];
    puts("Δωσε μια λέξη");
    gets(lexi1);
    blablabla(lexi1,lexi2,7);
    puts(lexi2);
}

```

👉 Στον πίνακα **lexi2[]** θα αντιγραφούν οι 7 πρώτοι χαρακτήρες του **lexi1[]**. Οπότε η **puts(lexi2)** θα εμφανίσει τους χαρακτήρες "Παπατρέ".

12.5

👉 Π.χ αν κάποιος καλέσει το πρόγραμμα * * *

```

char func1(char p);
void func2(char *p);
void func3(char *p, int num);
main()
{
    char a[10]="BENETIA", *ptr;
    printf("%c\n", func1(a[6]));
    func2(a);
    func2(&a[5]);
    func2(a+5);
    func3(a,5);
    func3(a+3,2);
}

```

B
BENETIA
IA
IA
TENEB
TE

Πίνακας a

a[0]	B
a[1]	E
a[2]	N
a[3]	E
a[4]	T
a[5]	I
a[6]	A
a[7]	\0
a[8]	
a[9]	

Πριν συνεχίσετε, δείτε πρώτα την επεξήγηση των συναρτήσεων που ακολουθεί στην επόμενη σελίδα!

- ☞ Στη `printf()` η συνάρτηση `func1()` καλείται με παράμετρο τον χαρακτήρα 'A' και επιστρέφει τον 'B'.
- ☞ Στη πρόταση `func2(a)` η συνάρτηση καλείται με παράμετρο την διεύθυνση του πίνακα `a`. Θα εμφανίσει όλους τους χαρακτήρες του πίνακα: "BENETIA"
- ☞ Και στις δύο επόμενες περιπτώσεις η `func2()` καλείται με παράμετρο την διεύθυνση της θέσης μνήμης `a[5]`. Θα εμφανίσει τους χαρακτήρες από την πέμπτη θέση μέχρι το τέλος: "IA"
- ☞ Τη πρώτη φορά η `func3()` καλείται με πρώτη παράμετρο τη διεύθυνση του `a` και δεύτερη το 5. Θα εμφανίσει αντίστροφα τους πέντε πρώτους χαρακτήρες από τον `a[4]` μέχρι τον `a[0]`: "TENEB".
- ☞ Τη δεύτερη φορά η `func3()` καλείται με πρώτη παράμετρο τη διεύθυνση `a+3` και δεύτερη το 2. Θα εμφανίσει αντίστροφα τους δύο πρώτους χαρακτήρες ξεκινώντας όμως όχι από την αρχή αλλά από τη διεύθυνση `a+3` δηλαδή από τον `a[4]` μέχρι τον `a[3]`: "TE".

```
char func1(char p)
{
    return p+1;
}
```

Η συνάρτηση `func1()` επιστρέφει σαν τιμή τον επόμενο χαρακτήρα από τον χαρακτήρα της παραμέτρου της. Π.χ αν κληθεί `func1('A')` επιστρέφει το 'B'.

```
void func2(char *p)
{
    puts(p);
}
```

Η συνάρτηση `func2()` εμφανίζει ένα σύνολο χαρακτήρων ξεκινώντας από τον χαρακτήρα που δείχνει ο δείκτης `p` μέχρι να εντοπιστεί ο χαρακτήρας τερματισμού '\0'.

```
void func3(char *p, int num)
{
    int i;
    for(i=num-1; i>=0; i--) putchar(p[i]);
    putchar('\n');
}
```

Η συνάρτηση `func3()` εμφανίζει από το σύνολο χαρακτήρων που δείχνει ο δείκτης `p`, χαρακτήρες με αντίστροφη σειρά ξεκινώντας από τον χαρακτήρα τον `num` μέχρι τον πρώτο. Π.χ `func3("ΚΑΚΑΡΕΛΟΣ",5)` θα εμφανίσει "ΡΑΚΑΚ".

12.6

Εφαρμογή: Πίνακας που έχει το επόμενο πρόγραμμα. ***

```
int func5(char *p, char ch);
char *func6(char *p, char ch);
main()
{
    char a[10], *ptr;
    strcpy(a, "BENETIA");
    printf("RES1=%d\n", func5(a, 'E'));
    printf("RES2=%d\n", func5(a+5, 'E'));
    ptr=func6(a, 'I');
    if(ptr!=NULL) printf("To %c υπάρχει στον a\n", *ptr);
}
```

Πίνακας a

a[0]	B
a[1]	E
a[2]	N
a[3]	E
a[4]	T
a[5]	I
a[6]	A
a[7]	\0
a[8]	
a[9]	

```
int func5(char *p, char ch)
{
```

Η συνάρτηση `func5()` δέχεται σαν παραμέτρους έναν δείκτη σε ένα σύνολο χαρακτήρων και έναν χαρακτήρα. Μετράει και επιστρέφει σαν τιμή το πόσες φορές ο χαρακτήρας `ch` βρίσκεται μέσα στο σύνολο χαρακτήρων.

```

int i=0, cnt=0;
while(p[i] != '\0')
{
    if(p[i] == ch) cnt++;
    i++;
}
return cnt;
}

```

```

char *func6(char *p, char ch)
{
    int i=0;
    while (p[i] != '\0')
    {
        if (p[i] == ch) return &p[i];
        i++;
    }
    return NULL;
}

```

Η συνάρτηση **func6()** δέχεται σαν παραμέτρους έναν δείκτη σε ένα σύνολο χαρακτήρων και έναν χαρακτήρα. Αν ο χαρακτήρας **ch** βρίσκεται μέσα στο σύνολο χαρακτήρων επιστρέφει σαν τιμή έναν δείκτη στη θέση που εντόπισε για πρώτη φορά τον χαρακτήρα. Στη περίπτωση που ο χαρακτήρας δεν εντοπιστεί μέσα στο σύνολο χαρακτήρων επιστρέφει τιμή NULL.

👉 Η κλήση της **func5(a, 'E')** επιστρέφει το 2 δεδομένου ότι ο χαρακτήρας 'E' υπάρχει δύο φορές μέσα στον πίνακα **a[]**.

👉 Η κλήση της **func5(a+5, 'E')** επιστρέφει το 0 διότι αρχίζει να μετράει για τον χαρακτήρα 'E' ξεκινώντας από τη θέση **a[5]** του πίνακα **a[]**.

👉 Η κλήση της **func6(a, 'T')** επιστρέφει σαν τιμή τη διεύθυνση της θέσης στην οποία εντόπισε το 'T' (της **a[5]**). Η διεύθυνση αυτή καταχωρείται στον δείκτη **ptr** και δεν είναι NULL. Η πρόταση ***ptr** αναφέρεται στο περιεχόμενο της θέσης στην οποία δείχνει ο δείκτης **ptr**.

2
0
Το I υπάρχει στον a

12.7

Δεδομένου του κώδικα, όπως φαίνεται στο επόμενο σχήμα, τι αποτελέσματα θα έχει στον πίνακα η κλήση της επόμενης συνάρτησης;

```

func4(a, a+6);
.....
void func4(char *p1, char *p2)
{
    char ch;
    while (p1 < p2)
    {
        ch=*p1;
        *p1=*p2;
        *p2=ch;
        p1++;
        p2--;
    }
}

```

Πίνακας a
αρχικά

a[0]	B
a[1]	E
a[2]	N
a[3]	E
a[4]	T
a[5]	I
a[6]	A
a[7]	\0
a[8]	
a[9]	

Ο a μετά
την func4()

a[0]	A
a[1]	I
a[2]	T
a[3]	E
a[4]	N
a[5]	E
a[6]	B
a[7]	\0
a[8]	
a[9]	

👉 Οι δείκτες **p1** και **p2**, αρχικά 'δείχνουν' στις θέσεις **a[0]** και **a[6]** αντίστοιχα. Οι τρεις πρώτες προτάσεις του βρόχου **while**, αντιμεταθέτουν τα περιεχόμενα των θέσεων

μνήμης στις οποίες 'δείχνουν' οι δύο δείκτες. Αμέσως μετά ο **p1** αυξάνεται κατά 1 'δείχνοντας' έτσι στην επόμενη θέση του πίνακα **a**, και ο **p2** μειώνεται κατά 1 'δείχνοντας' στην προηγούμενη θέση του πίνακα. Γίνεται η αντιμετάθεση των νέων θέσεων κ.ο.κ. Αυτό επαναλαμβάνεται ενόσω το **p1 < p2**.

👉 Το αποτέλεσμα της όλης διαδικασίας είναι η αντιστροφή των χαρακτήρων στον πίνακα **a**.

12.8

👉 **Η κλήση της συνάρτησης**

```
int sum(int pin[][10])
```

👉 **Ορίσμος της συνάρτησης**

```
int sum(int pin[][10])
{
    int i,j;
    int ss=0;
    for (i=0;i<35;i++)
        for (j=0;j<10;j++)
            ss+=pin[i][j];
    return ss;
}
```

➤ Αν κάλεσέ την με τη συνάρτηση με όρισμα έναν πίνακα με όνομα **test** όπως **int test[35][10]**, πέραν και τη διαδικασία που έχει να κάνει ο **test** για να ληφθεί σωστά η συνάρτηση **sum()**.

➤ Πιο συγκεκριμένα, στη θέση **pin[30][5]**.

➤ Αν η δήλωση **int pin[][10]** ήταν **int pin[5][10]** θα δούλευε σωστά η συνάρτηση.

👉 Η συνάρτηση γεμίζει έναν πίνακα δύο διαστάσεων 35x10 με αριθμούς. Ο αριθμός που καταχωρείται σε μια θέση μνήμης του πίνακα, προκύπτει από το γινόμενο του αριθμού γραμμής και του αριθμού στήλης που ανήκει η θέση. Π.χ στη θέση **pin[30][5]** θα καταχωρηθεί ο αριθμός 150 (30*5).

👉 Η συνάρτηση πρέπει να κληθεί με όρισμα έναν πίνακα τύπου **int**, δύο διαστάσεων 35x10.

👉 Αν η δήλωση **int pin[][10]** ήταν **int pin[5][10]** η συνάρτηση θα δούλευε σωστά διότι αγνοεί πλήρως τη τιμή της πρώτης διάστασης της παραμέτρου.

12.9

👉 **Ο κώδικας της συνάρτησης** η οποία να πάρει ως παράμετρο έναν πίνακα **int** τριών διαστάσεων 10x20x5 και να επιστρέψει ως τιμή το άθροισμα όλων των θέσεων μνήμης του πίνακα. *

```
int sum(int pin[][20][5])
{
    int i,j,k,ss;
    ss=0;
    for (i=0;i<10;i++)
        for (j=0;j<20;j++)
            for (k=0;k<5;k++)
                ss=ss+pin[i][j][k];
    return ss;
}
```

Τα **i,j** και **k** μεταβάλουν την πρώτη, τη δεύτερη και τη τρίτη διάσταση αντίστοιχα.

Στη μεταβλητή **ss** αθροίζονται όλες οι θέσεις μνήμης του πίνακα **pin**.

12.10 Να γραφεί πρόγραμμα το οποίο να ζητάει 100 αριθμούς να τους καταχωρίσει σε έναν πίνακα της διάστασης 100x1 και να υπολογίσει το μέσο όρο τους.

```
main()
{
    int pin[100],i,sum=0;
    for(i=0;i<100;i++)
    {
        scanf("%d",&pin[i]);
        sum=sum+pin[i];
    }
    printf("Ο μέσος όρος είναι %f\n",sum/100.0);
}
```

Η **scanf()** κάθε φορά ζητάει έναν αριθμό και τον καταχωρίζει σε διαφορετική θέση του πίνακα, ανάλογα με την τιμή του **i**.

Στη **sum** προστίθενται όλοι οι αριθμοί που δίνουμε.

Ο μέσος όρος είναι το συνολικό άθροισμα (**sum**) δια 100.

12.11 Να γραφεί πρόγραμμα το οποίο να καταχωρεί σε έναν τετραγωνικό πίνακα με 6x6 αριθμούς γινόμενα και διηλεκτό τον τοπολογικό αριθμό της διαγωνίου του από το ΒΑ προς το ΝΑ.

```
main()
{
    int a[6][6];
    int i,j;
    for(i=0;i<6;i++)
        a[i][i]=8;
}
```

Το χαρακτηριστικό των κελιών της διαγωνίου ενός τετραγωνικού πίνακα, είναι ότι έχουν την ίδια τιμή και στις δύο διαστάσεις. Π.χ **a[1][1]**, **a[4][4]**, (**a[i][i]**) κ.ο.κ.

8					
	8				
		8			
			8		
				8	
					8

12.12 Να γραφεί πρόγραμμα το οποίο να καταχωρεί σε έναν τετραγωνικό πίνακα με 6x6 αριθμούς γινόμενα και διηλεκτό έναν αριθμό στο τετράτο της διαγωνίου του από ΒΑ προς ΝΑ, έναν αριθμό στο δεξί του τμήμα και έναν αριθμό στο αριστερό του.

```
main()
{
    int a[6][6];
    int i,j;
    for(i=0;i<6;i++)
    {
        for(j=0;j<6;j++)
        {
            if(i>j)
                a[i][j]=5;
            else if(j>i)
                a[i][j]=8;
            else
                a[i][j]=1;
        }
    }
}
```

Το χαρακτηριστικό των κελιών του ΝΔ τμήματος ενός τετραγωνικού πίνακα, είναι ότι η τιμή του αριθμού αναφοράς της πρώτης διάστασης είναι μεγαλύτερη από την τιμή του αριθμού αναφοράς της δεύτερης (για **a[i][j]** το **i>j**).

Το χαρακτηριστικό των κελιών του ΒΑ τμήματος ενός τετραγωνικού πίνακα, είναι ότι για **a[i][j]** το **j>i**.

Το χαρακτηριστικό των κελιών της διαγωνίου ενός τετραγωνικού πίνακα, είναι ότι για **a[i][j]** το **i==j**.

1	8	8	8	8	8
5	1	8	8	8	8
5	5	1	8	8	8
5	5	5	1	8	8
5	5	5	5	1	8
5	5	5	5	5	1

☞ Τα κελιά με κίτρινο φόντο αποτελούν το Νοτιοδυτικό (ΝΔ) τμήμα του πίνακα. Τα κελιά με γαλάζιο φόντο αποτελούν το Βορειοανατολικό (ΒΑ) τμήμα του πίνακα. Τα κελιά της διαγωνίου έχουν ροζ φόντο.

12.13 Το παρακάτω πρόγραμμα έχει σαν διατεταγμένο πίνακα `a` το επόμενο κομμάτι: *

```
main()
{
    int a[6][6];
    int i,j;
    for(i=0;i<6;i++)
    {
        for(j=0;j<6;j++)
        {
            if(j+i>5)
                a[i][j]=1;
            else if(j+i<5)
                a[i][j]=2;
            else
                a[i][j]=0;
        }
    }
}
```

2	2	2	2	2	0
2	2	2	2	0	1
2	2	2	0	1	1
2	2	0	1	1	1
2	0	1	1	1	1
0	1	1	1	1	1

Το χαρακτηριστικό των κελιών του ΝΑ τμήματος του πίνακα, είναι ότι για $a[i][j]$ το $j+i>5$.

Το χαρακτηριστικό των κελιών του ΒΑ τμήματος του πίνακα, είναι ότι για $a[i][j]$ το $j+i<5$.

Το χαρακτηριστικό των κελιών της διαγωνίου του πίνακα, είναι ότι για $a[i][j]$ το $j+i==5$.

☞ Τα κελιά με κίτρινο φόντο αποτελούν το Βορειοδυτικό (ΒΔ) τμήμα του πίνακα. Τα κελιά με γαλάζιο φόντο αποτελούν το Νοτιοανατολικό (ΝΑ) τμήμα του πίνακα. Τα κελιά της διαγωνίου έχουν ροζ φόντο.

12.14 Το παρακάτω πρόγραμμα έχει σαν διατεταγμένο πίνακα `a` το επόμενο κομμάτι: *

```
int sum(int a[][100])
{
    int i,s=0;
    for(i=0;i<100;i++)
    {
        s=s+a[i][i];
    }
    return s;
}
```

12.15 Να βρούμε το μέγιστο στοιχείο ενός πίνακα `int a[100][20]`. Να γραφεί έσθλιος κώδικας να εμφανίσει τον μέγιστο αριθμό υποκείμενο από κάθε σειρά του πίνακα (να εμφανίζεται ένα αποτέλεσμα). ★★

```
main()
{
    int a[100][20],max;
    int i,j;
    .....
    for(i=0;i<100;i++)
    {
        max=a[i][0];
        for(j=0;j<20;j++)
        {
            if(a[i][j]>max)
                max=a[i][j];
        }
        printf("Μεγιστος σειρας %d = %d\n",i,max);
    }
}
```

Η αρχική τιμή της `max` είναι κάθε φορά η τιμή της πρώτης θέσης κάθε γραμμής (`i`).

Στη `max` τελικά καταχωρείται η μεγαλύτερη από τις τιμές των θέσεων μνήμης της γραμμής `i`.

➡ Στο παραπάνω πρόγραμμα οι μέγιστες τιμές κάθε γραμμής απλά εμφανίζονται στην οθόνη. Αν θέλαμε να υπάρχουν και κάπου καταχωρημένες, θα έπρεπε να χρησιμοποιήσουμε έναν πίνακα 100 θέσεων π.χ `max[100]` όπου σε κάθε θέση μνήμης του πίνακα `max[]` θα καταχωρούσαμε τη μέγιστη τιμή της αντίστοιχης γραμμής του πίνακα `a[]`.

12.16 Ποια από τα επόμενα αληθεύουν: ★

- ☒ Ένας πίνακας ορίζει έμμεσα και ένα δείκτη με αρχική τιμή τη διεύθυνση της πρώτης θέσης μνήμης του πίνακα.
- ☐ Σε ένα πίνακα χαρακτήρων μιας διάστασης μπορούν να καταχωριστούν πολλές συμβολοσειρές.
- ☒ Για να έχει μια συνάρτηση πρόσβαση σε έναν πίνακα μιας διάστασης, πρέπει να της διαβιβάσουμε μόνο τη διεύθυνση του πίνακα.
- ☒ Για να έχει μια συνάρτηση, πρόσβαση σε έναν πίνακα περισσότερων από μιας διαστάσεων, πρέπει γνωρίζει εκτός από τη διεύθυνση του πίνακα και τις τιμές των διαστάσεων του εκτός από την πρώτη.
- ☐ Κατά τη δήλωση ενός πίνακα, η τιμές των διαστάσεων του μπορούν να είναι και μεταβλητές. Π.χ. η `int a[b]` δημιουργεί έναν πίνακα `a` με θέσεις μνήμης όσες και η τιμή της μεταβλητής `b`.

12.17 Να γραφεί πρόγραμμα το οποίο: ★★

- Να πάρει τα στοιχεία 10 αριθμών και θα τα καταχωρεί σε κάδο ή σε πίνακα χαρακτήρων.
- Να πάρει από τον χρήστη έναν χαρακτήρα και θα εμφανίζει μόνο το αντίστοιχο του πίνακα που εισάγαμε από τα χαρακτήρα που δόθηκε.

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

```

{
    char onomata[10][20],ch;
    int i;
    for (i=0;i<10;i++)
    {
        printf("Δώσε όνομα %d :",i+1);
        gets(onomata[i]);
    }
    printf("\nΔώσε αρχικό χαρακτήρα:");
    scanf("%c",&ch);
    printf("\nΤα ονόματα που αρχίζουν από %c είναι:\n",ch);
    for (i=0;i<10;i++)
    {
        if (onomata[i][0]==ch) puts(onomata[i]);
    }
}

```

Καταχώριση των 10 ονομάτων στον πίνακα **onomata**.

Ζητείται από το πληκτρολόγιο ο αρχικός χαρακτήρας.

Ελέγχεται ο πρώτος χαρακτήρας κάθε ονόματος.

12.18

Δεσμεύουν ένας πίνακα **a** και θέτουν μετρητή **i** σε 0. Ο πρώτος έλεγχος, με τον οποίο ο πρόγραμμα να μπορεί να συνεχίσει ή όχι, είναι το αν υπάρχει ολότελα τμήμα του πίνακα που να έχει **a[i]** που είναι μεγαλύτερο από έναν αριθμό **ar** σε έναν άλλο πίνακα **b** (μόλις 100 θέσεις χωρίς να αφήνει κενές θέσεις). Η συνάρτηση επιστρέφει μετρητή **i** που είναι ο πρώτος που αριθμούν τα μη αντέγραφα. Οι πίνακες **a** **b** είναι **copy** και **plithos**. Θα πρέπει να μεταβληθούν τιμές από ορισμένα στη συνάρτηση.

```

#include <stdio.h>
#include <stdlib.h>
main()
{
    int a[100],b[100],i,ar,plithos;
    for (i=0;i<100;i++) a[i]=rand();
    printf("Δώσε αριθμό:");
    scanf("%d",&ar);
    plithos=copy(a,b,ar);
    printf("\n");
    for (i=0;i<plithos;i++)
    {
        printf("%d\n",b[i]);
    }
}

int copy(int p1[],int p2[],int n)
{
    int k=0,i;
    for (i=0;i<100;i++)
        if (p1[i]>n) p2[k++]=p1[i];
    return k;
}

```

Γέμισμα του πίνακα **a** με τυχαίους αριθμούς.

Η τιμή που επιστρέφει η **copy()** καταχωρίζεται στη μεταβλητή **plithos**.

Εμφάνιση των περιεχομένων του πίνακα **b**.

Αντιγράφονται οι τιμές του πίνακα **p1**, με τιμή μεγαλύτερη από τη παράμετρο **n**, στον **p2**.

Η συνάρτηση επιστρέφει ως τιμή, το πλήθος των τιμών που αντέγραψε.

12.19

Κάποιος καθηγητής θέλει να ελέγξει πόσο καλά οι μαθητές της βιολογίας του είναι σε όλα τα μαθήματα. Είναι υποχρεωμένος να δώσει βαθμολογία από 0 έως 5, ενώ θεωρείται ότι οι σταυροί σε 0 μαθητές ή 100 είναι μεγάλοι αριθμοί ή ίσως και 1000. Θα γράψει λοιπόν έναν πρόγραμμα που θα δίνει τις βαθμολογίες 50 μαθητών σε 3 μαθήματα και να τις καταχωρεί σε έναν πίνακα 3x50. Το πρόγραμμα θα πρέπει να υπολογίζει και το μέσο όρο ανά μάθημα, και το ποσοστό των μαθητών που απέτυχαν, καθώς και το πλήθος και το ποσοστό των μαθητών που αρίστευσαν.

```
#include <stdio.h>
#include <stdlib.h>
main()
{
    int b[50][3],i,j,pl_ap=0,pl_ar=0;
    float sum,mo;
    for (i=0;i<50;i++)
    {
        printf("Μαθητής %d\n",i+1);
        printf("-----\n");
        for (j=0;j<3;j++)
        {
            printf("Μάθημα %d:",j+1);
            scanf("%d",&b[i][j]);
        }
    }
    for (i=0;i<50;i++)
    {
        sum=0;
        for (j=0;j<3;j++) sum=sum+b[i][j];
        mo=sum/3;
        if (mo>=18.5) pl_ar++;
        if (mo<9.5) pl_ap++;
    }
    printf("Αποτυχόντες: %d  %5.2f%%\n",pl_ap,pl_ap*100/50.0);
    printf("Αριστούχοι: %d  %5.2f%%\n",pl_ar,pl_ar*100/50.0);
}
```

Οι βαθμολογίες θα καταχωριστούν σε έναν πίνακα 50 γραμμών και 3 στηλών.

Καταχώριση των βαθμών στον πίνακα b.

Υπολογισμός του μέσου όρου κάθε μαθητή.

Καταμέτρηση του πλήθους των αριστούχων και των αποτυχόντων.

12.20

Είστε πρόγραμματιστής και έχετε να λύσετε ένα πρόβλημα που αφορά τα αυτοκίνητα. Κάθε μήνα σε ένα σημείο ενός δρόμου, οι αυτοκίνητα που περνούν θα σταματούν και θα ελεγχονται για ταχύτητα. Κάθε 100 αυτοκίνητα που περνούν θα ελεγχθούν και θα ελεγχθεί αν είναι ο κανονικός. Για κάθε αυτοκίνητο που ελεγχθεί θα δοθεί ένας αριθμός που εκφράζει το πόσο μακριά είναι από το κανονικό. Ο αριθμός αυτός θα είναι από 0 έως 100. Ο αριθμός 0 σημαίνει ότι το αυτοκίνητο είναι κανονικό, ο αριθμός 100 σημαίνει ότι το αυτοκίνητο είναι πολύ μακριά από το κανονικό. Ο αριθμός 50 σημαίνει ότι το αυτοκίνητο είναι στο μέσο. Ο αριθμός 10 σημαίνει ότι το αυτοκίνητο είναι λίγο μακριά από το κανονικό. Ο αριθμός 20 σημαίνει ότι το αυτοκίνητο είναι λίγο μακριά από το κανονικό. Ο αριθμός 30 σημαίνει ότι το αυτοκίνητο είναι λίγο μακριά από το κανονικό. Ο αριθμός 40 σημαίνει ότι το αυτοκίνητο είναι λίγο μακριά από το κανονικό. Ο αριθμός 60 σημαίνει ότι το αυτοκίνητο είναι λίγο μακριά από το κανονικό. Ο αριθμός 70 σημαίνει ότι το αυτοκίνητο είναι λίγο μακριά από το κανονικό. Ο αριθμός 80 σημαίνει ότι το αυτοκίνητο είναι λίγο μακριά από το κανονικό. Ο αριθμός 90 σημαίνει ότι το αυτοκίνητο είναι λίγο μακριά από το κανονικό. Ο αριθμός 100 σημαίνει ότι το αυτοκίνητο είναι πολύ μακριά από το κανονικό.

- Για κάθε αυτοκίνητο, τον αριθμό του και τον αριθμό που είναι μακριά από το κανονικό, θα δοθεί ένας πίνακας μετρήσεων.
- Το αυτοκίνητο με τον πιο μακριά από το κανονικό, θα είναι το 10.
- Το αυτοκίνητο με τον πιο κοντά στο κανονικό, θα είναι το 50.

Κάθε μία από τις παραπάνω απαιτήσεις να υλοποιείται από κατάλληλη συνάρτηση.

```
#include <stdio.h>
#include <stdlib.h>
#define PLHTHOS 100
```

Με την οδηγία **define** ορίζουμε το **PLHTHOS** των αυτοκινήτων ώστε να είναι εύκολο να το αλλάξουμε αν χρειαστεί.

```
void display_synolo_paravaseon(char ak[][10], int pr[][12],int sp[]);
void display_10_top(char ak[][10], int sp[]);
void display_minimum(char ak[][10], int sp[]);
```

```
main()
{
```

```
    char ar_kykl[PLHTHOS][10];
    int par[PLHTHOS][12],syn_par[PLHTHOS],i,j;
    for (i=0;i<PLHTHOS;i++)
    {
        printf("Δώσε αριθ. κυκλοφορίας %d :",i+1);
        gets(ar_kykl[i]);
    }
```

Στον πίνακα χαρακτήρων **ar_kykl** θα καταχωρίζονται οι αριθμοί κυκλοφορίας των αυτοκινήτων (δεν ξεχναμε οτι περιέχουν και χαρακτήρες π.χ. ΥΥΑ3456). Στον πίνακα **par** οι παραβάσεις και στον πίνακα **syn_par** θα υπολογίζονται οι συνολικές παραβάσεις όλου του έτους.

```
    for (i=0;i<PLHTHOS;i++)
    {
        printf("\nΑυτοκίνητο %s\n",ar_kykl[i]);
        printf("-----\n");
        for (j=0;j<12;j++)
        {
            printf("Παραβάσεις μηνός %d:",j+1);
            scanf("%d",&par[i][j]);
        }
        display_synolo_paravaseon(ar_kykl,par,syn_par);
        display_10_top(ar_kykl,syn_par);
        display_minimum(ar_kykl,syn_par);
    }
```

Διάβασμα από το πληκτρολόγιο, των αριθμών κυκλοφορίας των αυτοκινή-

Διάβασμα, από το πληκτρολόγιο, των παραβάσεων όλων των αυτοκινήτων για κάθε μήνα.

```
void display_synolo_paravaseon(char ak[][10], int pr[][12], int sp[])
{
```

```
    int i,j,sum;
    printf("Αρ. Κυκλ    Παραβ\n");
    printf("-----  -----\n");
    for (i=0;i<PLHTHOS;i++)
    {
        sum=0;
        for (j=0;j<12;j++)
        {
            sum=sum+pr[i][j];
        }
    }
```

Υπολογισμός των συνολικών παραβάσεων του έτους για κάθε αυτοκίνητο.

```

        sp[i]=sum;
        printf("%10s  %5d\n",ak[i],sum);
    }
}

void display_10_top(char ak[][10], int sp[])
{
    int i;
    printf("\nTOP-10\n\n");
    printf("Αρ-Κυκλ      Παραβ\n");
    printf("-----  -----\n");
    for (i=0;i<PLHTHOS;i++)
    {
        if (sp[i]>10) printf("%10s  %5d\n",ak[i],sp[i]);
    }
}

void display_minimum(char ak[][10], int sp[])
{
    int i,min;
    printf("\nMinimum Παραβάσεις \n\n");
    printf("Αρ-Κυκλ.      Παραβ\n");
    printf("-----  -----\n");
    min=sp[0];
    for (i=0;i<PLHTHOS;i++)
        if (sp[i]<min) min=sp[i];
    for (i=0;i<PLHTHOS;i++)
        if (sp[i]==min) printf("%10s  %5d\n",ak[i],sp[i]);
}

```

Καταχώριση των παραβάσεων στον πίνακα **sp** και εμφάνισή τους στην οθόνη.

Στη περίπτωση που το σύνολο παραβάσεων του αυτοκινήτου είναι πάνω από 10, εμφανίζει τον αριθ. κυκλ και τον συνολικό αριθμό παραβάσεων.

Υπολογίζει τις λιγότερες παραβάσεις **min**.

Στην περίπτωση που το σύνολο παραβάσεων του αυτοκινήτου είναι ίσο με το **min**, εμφανίζει τον αριθ. κυκλ και το σύνολο παραβάσεων.

12.21

```

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

main()
{
    int xreosi[12][31],i,j,max;
    for (i=0;i<12;i++)
    {
        printf("\nΜήνας %d\n",i+1);
        printf("-----\n");
        for (j=0;j<31;j++)
        {
            printf("Χρέωση %d/%d:",j+1,i+1);

```

Στον πίνακα **xreosi** θα καταχωρίζονται οι χρεώσεις κάθε ημέρας (12 μήνες x 31 ημέρες το μέγιστο κάθε μήνα).

```

scanf("%d",&xreosi[i][j]);
}
}
max=xreosi[0][0];
for (i=0;i<12;i++)
    for (j=0;j<31;j++)
        if (xreosi[i][j]>max) max=xreosi[i][j];
printf("Ημέρες με τη μεγαλύτερη χρέωση:%d\n",max);
printf("-----\n");
for (i=0;i<12;i++)
    for (j=0;j<31;j++)
        if (xreosi[i][j]==max) printf("%d/%d\n",j+1,i+1);
}

```

Καταχώριση των ημερήσιων χρεώσεων στον πίνακα **xreosi**.

Υπολογισμός της μέγιστης χρέωσης

Εμφάνιση των ημερών με τη μέγιστη χρέωση.

12.22 Το πρόγραμμα καταχωρεί και ελέγχει επιβάτες που δεν πρέπει να επιβιβάζονται λόγω του ότι το όνομά τους υπάρχει ήδη στον πίνακα. Η συνάρτηση `is_on_board` που θα βρούμε το ονόματι των επιβατών ενός αεροπλανοφόρου και θα ελεγχθεί αν υπάρχει ή όχι στον πίνακα. Η συνάρτηση θα αναζητήσει στον πίνακα το ονομαστικό περιεχόμενο της στήλης `th` και θα επιστρέψει 1 αν υπάρχει και 0 αν δεν υπάρχει. Η καταχώριση των επιβατών στον πίνακα θα γίνει με εμφάνιση το ονόματι ενός από επιβατών της πλοιο.

```

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

int is_on_board(char th[][20],char onom[],int pl);

main()
{
    char theseis[100][20],onoma[20];
    int i,plithos=0;
    while (plithos<100)
    {
        printf("Επιβάτης θέσης %d:",plithos+1);
        gets(onoma);
        if (is_on_board(theseis,onoma,plithos))
        {
            printf("Ο %s υπάρχει ήδη\n",onoma);
            continue;
        }
        if (strcmp(onoma,"----")==0) break;
        strcpy(theseis[plithos],onoma);
        plithos++;
    }
    printf("\nΛίστα επιβατών\n");
    printf("-----\n");
    for (i=0;i<plithos;i++)
    {

```

Τα ονόματα των επιβατών θα καταχωρίζονται στον πίνακα χαρακτήρων **theseis**. Μέγιστος αριθμός επιβατών 100.

Διάβασμα του ονόματος ενός επιβάτη

Η συνάρτηση ελέγχει αν το όνομα είναι ήδη καταχωρημένο. Στην συνάρτηση μεταβιβάζεται ο πίνακας **theseis**, το όνομα του επιβάτη καθώς και το πλήθος των μέχρι στιγμής καταχωρήσεων.

Το όνομα καταχωρίζεται στον πίνακα **theseis** και το πλήθος των επιβατών αυξάνει κατά 1.

Εμφάνιση όλων των καταχωρημένων επιβατών του πίνακα.

```

        printf("Θέση %d:%s\n",i+1,theseis[i]);
    }
}

int is_on_board(char th[][20],char onom[],int pl)
{
    int i,found=0;
    for (i=0;i<pl;i++)
        if (strcmp(th[i],onom)==0)
        {
            found=1;
            break;
        }
    return found;
}

```

Η συνάρτηση προσπαθεί να εντοπίσει το όνομα **onom**, μέσα στον πίνακα **th** ο οποίος έχει ήδη καταχωρημένα **pl** ονόματα. Στη περίπτωση που το όνομα εντοπιστεί, επιστρέφει τιμή 1 διαφορετικά επιστρέφει τιμή 0.

12.23 Η συνάρτηση **common()** που θα δούμε παρακάτω, εμφανίζει τους κοινούς χαρακτήρες όσες φορές εμφανίζονται, ενώ η **common_once()** μόνο μια φορά τον καθένα.

```

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

void common(char *s1,char *s2);
void common_once(char *s1,char *s2);

main()
{
    char lex1[20],lex2[20];
    gets(lex1);
    gets(lex2);
    printf("Κοινοι χαρακτήρες των %s και %s:",lex1,lex2);
    common(lex1,lex2);
    printf("\n---\n");
    printf("Κοινοι χαρακτήρες των %s και %s:",lex1,lex2);
    common_once(lex1,lex2);
    printf("\n---\n");
}

```

Η συνάρτηση **common()** εμφανίζει τους κοινούς χαρακτήρες όσες φορές εμφανίζονται, ενώ η **common_once()** μόνο μια φορά τον καθένα.

```

void common(char *s1,char *s2)
{
    int i,j,ls1,ls2;
    ls1=strlen(s1);
    ls2=strlen(s2);
    for (i=0;i<ls1;i++)
        for (j=0;j<ls2;j++)
            if (s1[i]==s2[j])
            {
                putchar(s1[i]);
                break;
            }
}

```

Στις μεταβλητές **ls1** και **ls2** καταχωρίζεται το πλήθος χαρακτήρων των συμβολοσειρών **s1** και **s2** αντίστοιχα.

Στη περίπτωση που εντοπιστεί κοινός χαρακτήρας, εμφανίζεται στην οθόνη. Στη περίπτωση που το **s1** περιέχει το χαρακτήρα εκαί δεύτερη φορά τότε θα ξαναεμφανιστεί.

```

    }
}

void common_once(char *s1, char *s2)
{
    int i, j, k, ls1, ls2, found;
    ls1=strlen(s1);
    ls2=strlen(s2);
    for (i=0; i<ls1; i++)
    {
        found=0;
        for (k=0; k<i; k++) if (s1[k]==s1[i]) found=1;
        if (found) continue;
        for (j=0; j<ls2; j++)
            if (s1[i]==s2[j])
            {
                putchar(s1[i]);
                break;
            }
    }
}

```

Στη περίπτωση που ο χαρακτήρας **s[i]** υπάρχει στις προηγούμενες θέσεις της συμβολοσειράς (από τη 0 μέχρι την i), τότε η μεταβλητή **found** παίρνει την τιμή 1 και δεν γίνεται ξανά ο έλεγχος στο **s2** για αυτόν τον χαρακτήρα.

12.24

Σε έναν πίνακα χαρακτήρων **fr[10][40]** είναι καταχωρημένες 10 φράσεις. Να γραφεί συνάρτηση η οποία θα αναζητά με παράμετρος τον πίνακα **fr** και ένα λέξι **lexi** και να επιστρέφει τη συνάρτηση με τη φράση που περιέχει τον **lexi** στην οποία υπάρχει το **lexi**. Εάν η συμβολοσειρά δεν εντοπιστεί τότε η συνάρτηση θα επιστρέφει την **NULL**.

Επίσης να γραφεί πρόγραμμα το οποίο χρησιμεύει ως test. Η συγκεκριμένη συνάρτηση να ελεγχτεί μέσα στον πίνακα και να επιστρέφει τη φράση που περιέχει τον **lexi** ή ούτως ή άλλως να επιστρέφει με **NULL**.

```

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

char *find(char fr[][40], char lexi[])
{
    int i;
    char *ptr=NULL;
    for (i=0; i<20; i++)
        if (strstr(fr[i], lexi) != NULL)
        {
            ptr=fr[i];
            break;
        }
    return ptr;
}

```

Η συνάρτηση **find** εντοπίζει μέσα στον πίνακα **fr** τη συμβολοσειρά **lexi**.

Η συνάρτηση **strstr()** χρησιμοποιείται για να εντοπίζει τη συμβολοσειρά **lexi** στη γραμμή **fr[i]** του πίνακα. Στη περίπτωση που εντοπιστεί, η συνάρτηση **find()** επιστρέφει έναν δείκτη (**ptr**) στη γραμμή στην οποία εντόπισε τη συμβολοσειρά **lexi**.

```
main()
{
    char fraseis[20][40], *pt, lex[10];
    int i;
    for (i=0; i<20; i++)
        gets(fraseis[i]);
    printf("Δώσε λέξη για αναζήτηση:");
    gets(lex);
    pt=find(fraseis, lex);
    if (pt!=NULL)
        printf("Η λέξη βρέθηκε στη φράση:%s\n", pt);
    else
        printf("Η λέξη δεν βρέθηκε\n");
}
```

Πληκτρολογούνται 20 φράσεις οι οποίες καταχωρίζονται στον πίνακα **fraseis**.

Η συνάρτηση **find()** επιστρέφει ένα δείκτη στη γραμμή του πίνακα **fraseis** στην οποία εντόπισε τη λέξη **lex**.

Στη περίπτωση που η λέξη εντοπίστηκε, η **printf()** εμφανίζει όλη τη φράση που περιέχει τη συγκεκριμένη λέξη.

12.25 Να γράψετε πρόγραμμα που θα δέχεται ως παράμετρο μία συμβολοσειρά και θα αντιστρέφει τα γράμματα της. Το πρόγραμμα να επιστρέφει ένα δείκτη στην ίδια συμβολοσειρά. *

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

char *anakatema(char *str);

main()
{
    char lex[20];
    gets(lex);
    printf(anakatema(lex));
    printf("\n---\n");
}

char *anakatema(char *str)
{
    int i, ta1, ta2, len;
    char temp;
    len=strlen(str);
    //Αρχικοποίηση της γεννήτριας τυχαίων αριθμών
    srand(time(NULL));
    for (i=0; i<100; i++)
    {
        ta1=rand()%len;
        ta2=rand()%len;
        temp=str[ta1];
        str[ta1]=str[ta2];
        str[ta2]=temp;
    }
    return str;
}
```

Η συνάρτηση επιστρέφει δείκτη σε συμβολοσειρά, οπότε μπορεί να χρησιμοποιηθεί απευθείας από την **printf()**.

Στη μεταβλητή **len** καταχωρίζεται το πλήθος των χαρακτήρων της συμβολοσειράς.

Γίνεται αρχικοποίηση της γεννήτριας χαρακτήρων με διαφορετικό κάθε φορά αριθμό. Η συνάρτηση **time()** επιστρέφει τον αριθμό των δευτερολέπτων από την 1/1/1970.

Στις μεταβλητές **ta1** και **ta2** καταχωρίζονται δύο τυχαίοι αριθμοί από το 0 μέχρι το **len-1**.

Γίνεται αντιμετάθεση των περιεχομένων των τυχαίων θέσεων **str[ta1]** και **str[ta2]** του πίνακα.

Η συνάρτηση επιστρέφει τη διεύθυνση του πίνακα χαρακτήρων.

Ασκήσεις Κεφαλαίου 13

13.1

Να γραφεί πρόγραμμα το οποίο να καταχωρεί σε μια μεταβλητή δομής τα στοιχεία ενός αυτοκινήτου (αριθμός κυκλοφορίας, χρώμα, κυβικά, ισχύς, τύπος και ιπποδύναμη). Να επεξεργαστείτε το πρόγραμμα σύμφωνα με τις οδηγίες που πρέπει να χρησιμοποιήσετε ***

```
struct cars
{
    char ar_kykl[8];
    char xroma[15];
    char marka[15];
    int kybika;
    int ipodynami;
};

main()
{
    struct cars mycar;
    printf("Αριθμός κυκλοφορίας:");
    gets(mycar.ar_kykl);
    printf("Χρώμα:");
    gets(mycar.xroma);
    printf("Μάρκα:");
    gets(mycar.marka);
    printf("Κυβικά:");
    scanf("%d",&mycar.kybika);
    printf("Ιπποδύναμη:");
    scanf("%d",&mycar.ipodynami);
}
```

Δήλωση της δομής **cars** με πέντε πεδία. Παρατηρούμε ότι σε αυτό το σημείο δεν δηλώνεται καμία μεταβλητή.

Δήλωση της μεταβλητής **mycar** τύπου δομής **cars**.

Καταχώριση στοιχείων στα πεδία της μεταβλητής **mycar**.

13.2

Να γραφεί πρόγραμμα το οποίο να καταχωρεί σε έναν πίνακα από δομές τα στοιχεία ενός ή περισσότερων αυτοκινήτων (αριθμός κυκλοφορίας, χρώμα, κυβικά, ισχύς, τύπος και ιπποδύναμη). Να επεξεργαστείτε το πρόγραμμα σύμφωνα με τις οδηγίες που πρέπει να χρησιμοποιήσετε. Η διαφορά στα σχήματα είναι να σημειωθεί ο-
 ταν δοθεί κανόνας για τον αριθμό των αυτοκινήτων ***

```
struct cars
{
    char ar_kykl[8];
    char xroma[15];
    char marka[15];
    int kybika;
    int ipodynami;
};
```



```
main()
{
    struct cars mycars[100];
    int i;
    for(i=0;i<100;i++)
    {
        printf("Αριθμός κυκλοφορίας:");
        gets(mycars[i].ar_kykl);
        if(strcmp(mycars[i].ar_kykl,"")==0)
            break;
        printf("Χρώμα:");
        gets(mycars[i].xroma);
        printf("Μάρκα:");
        gets(mycars[i].marka);
        printf("Κυβικά:");
        scanf("%d",&mycars[i].kybika);
        printf("Ιπποδύναμη:");
        scanf("%d",&mycars[i].ipodynami);
    }
}
```

Δήλωση του πίνακα δομών **mycars** με 100 θέσεις.

Στην περίπτωση καταχώρισης κενού αριθμού κυκλοφορίας, η επαναληπτική διαδικασία σταματάει.

13.3

Υποθέτουμε ότι έχουμε έναν πίνακα δομών με τη δομή που έχουμε προηγουμένως. Στον πίνακα αυτό είναι καταχωρημένοι οι στοιχεία των 100 μαθητών ενός σχολείου. Να γραφεί συνάρτηση η οποία να εμφανίζει τη ηλικία των μαθητών. Η κάθε δομή να περιέχει επώνυμο του μαθητή και το μέσο όρο των κοφτερότητας με ένα δεκαδικό ψηφίο. *

```
struct stoixeia
{
    char eponymo[30];
    char taxi[5];
    float mesos_oros;
    int ilikia;
} mathites[100];
```

```
void print_it(struct stoixeia pin[])
{
    int i;
    for(i=0;i<100;i++)
    {
        printf("%s %s %4.1f\n", pin[i].eponymo, pin[i].taxi,
            pin[i].mesos_oros);
    }
}
```

Στη συνάρτηση, δηλώνουμε σαν παράμετρο ένα πίνακα ίδιου τύπου με τον πίνακα **mathites[]**.

13.4

Να γραφεί συνάρτηση που θα δίνει τη μέγιστη ηλικία των μαθητών. Η οποία να επιστρέφει ως τιμή τη μέγιστη ηλικία των μαθητών. *

```
int max_ilikia(struct stoixeia pin[])
{
    int i,max;
    max=pin[0].ilikia;
    for(i=0;i<100;i++)
```

Στη συνάρτηση, δηλώνουμε σαν παράμετρο ένα πίνακα ίδιου τύπου με τον πίνακα **mathites[]**.

Καταχωρούμε σαν αρχική τιμή της **max**, την ηλικία του πρώτου μαθητή (θέση 0 του πίνακα).

```
{
    if (pin[i].ilikia>max)
        max=pin[i].ilikia;
}
return max;
}
```

Αν η ηλικία του μαθητή i είναι μεγαλύτερη από την μέχρι στιγμής μέγιστη ηλικία (\max), τότε στη θέση του \max καταχωρούμε αυτή την ηλικία.

13.5 Ενδεικτικά με λήξη στο επόμενο πρόγραμμα: *

```
enum days {mon,tue,wed,thu,fri,sat,sun} birth_day,today;
typedef int meres;

main()
{
    meres a,b;
    days my_date;
    a=1;
    my_date=mon;
    wed=5;
    meres=23;
    today=3;
    birth_day=fri;
}
```

Το **web** δεν είναι μεταβλητή.

Το **meres** δεν είναι μεταβλητή αλλά τύπος δεδομένων.

13.6 Ποια από τα επόμενα αληθεύουν: ★

- ☑ Μια δομή δεν μπορεί να περιέχει δύο πεδία με το ίδιο όνομα.
- ❑ Σε μια μεταβλητή τύπου **enum** δεν μπορούμε να καταχωρίσουμε άλλη τιμή πέρα από τις τιμές που ορίσαμε στη δήλωση του τύπου **enum** στον οποίο ανήκει.
- ☑ Για να έχουμε πρόσβαση στο πεδίο μιας δομής μέσω ενός δείκτη χρησιμοποιούμε τον τελεστή βέλους \rightarrow .
- ❑ Όταν μεταβιβάζουμε μια δομή ως παράμετρο σε μια συνάρτηση, η παράμετρος **πρέπει** να είναι του ίδιου τύπου δομής με το όρισμα που θα της μεταβιβάσουμε.
- ☑ Όταν έχουμε δύο μεταβλητές του ίδιου τύπου δομής, μπορούμε με τον τελεστή $=$ να καταχωρίσουμε όλα τα περιεχόμενα της μιας στην άλλη. Για παράδειγμα, η **filos=pelatis** καταχωρίζει όλα τα πεδία της μεταβλητής δομής **pelatis** στα αντίστοιχα πεδία της μεταβλητής δομής **filos**.

[illegible]

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

```
#include <stdlib.h>

#define AR 50

struct metoxi
{
    char onoma[10];
    int posotita;
    float timi;
    char eidos;
};

main()
{
    struct metoxi kiniseis[AR];
    float syn_axia;
    int i, syn_a, syn_p;
    for (i=0; i<AR; i++)
    {
        printf("Στοιχεία κίνησης No:%d\n", i+1);
        printf("=====\n");
        printf("Είδος κίνησης (A/P):");
        scanf("%c", &kiniseis[i].eidos);
        printf("Ονομασία μετοχής:");
        scanf("%s", &kiniseis[i].onoma);
        printf("Ποσότητα:");
        scanf("%d", &kiniseis[i].posotita);
        printf("Τιμή:");
        scanf("%f", &kiniseis[i].timi);
        getchar();
        putchar('\n');
    }
    syn_axia=0;
    syn_a=0;
    syn_p=0;
    for (i=0; i<AR; i++)
    {
        if (kiniseis[i].eidos=='A') syn_a=syn_a+kiniseis[i].posotita;
        if (kiniseis[i].eidos=='Π') syn_p=syn_p+kiniseis[i].posotita;
        syn_axia=syn_axia+kiniseis[i].timi*kiniseis[i].posotita;
    }
    printf("Πλήθος μετοχών που αγοράστηκαν = %d\n", syn_a);
    printf("Πλήθος μετοχών που πουλήθηκαν = %d\n", syn_p);
    printf("Συνολική αξία διακίνησης = %f\n", syn_axia);
}
```

Για μεγαλύτερη ευελιξία δηλώνουμε τον αριθμό των κινήσεων σαν σταθερά με όνομα **AR**.

Ορισμός της δομής **metoxi** με τα κατάλληλα πεδία.

Ορισμός ενός πίνακα **kiniseis**, δομών **metoxi**, με **AR (50)** θέσεις.

Στη μεταβλητή **syn_axia** θα υπολογιστεί η συνολική αξία των κινήσεων. Στις δε μεταβλητές **syn_a** και **syn_b** το πλήθος των μετοχών που αγοράστηκαν και πουλήθηκαν αντίστοιχα.

Διαβάζονται τα στοιχεία κάθε κίνησης και καταχωρίζονται στις θέσεις του πίνακα **kiniseis** (στα αντίστοιχα πεδία).

Δίδονται αρχικές τιμές στις μεταβλητές.

Υπολογισμός του συνόλου μετοχών που αγοράστηκαν και που πουλήθηκαν καθώς και της συνολικής αξίας των κινήσεων.



Η κλήση της **getchar()** μετά από ορισμένες **scanf()**, χρησιμοποιείται για το διάβασμα του παραμένοντα χαρακτήρα αλλαγής γραμμής στο ρεύμα εισόδου (βλέπε την παράγραφο "Η scanf() και τα μικρά της προβλήματα" στη σελίδα 104 του βιβλίου).

13.8

Μια εφαρμογή που θα υπολογίζει και θα εμφανίζει με διαφάνεια τον μέσο όρο πτήσεων. Για κάθε πτήση η η εταιρεία καταχωρεί τα ετήσια στοιχεία. Το μέσο όρο της πτήσης το πλήθος των θέσεων και ο αριθμός των επιβατών που επιβαίνουν. Τα γραφικά περιλαμβάνουν τα ακόλουθα:

- Θα ζητηθεί ο αριθμός από το πληκτρολόγιο και θα εμφανιστεί το αντίστοιχο πεδίο πτήσης.
- Θα υπολογιστούν οι επιβατικοί και επιβατικοί όλοι οι επιβατικοί.
- Θα υπολογιστεί ο μέσος όρος πτήσεων όλων των πτήσεων. Θα ληφθεί υπόψη ο αριθμός των θέσεων και ο αριθμός των επιβατικών.

```
#include <stdio.h>
#include <stdlib.h>
#define AR 10
```

Για μεγαλύτερη ευελιξία δηλώνουμε τον αριθμό των πτήσεων σαν σταθερά με όνομα AR.

```
struct ptisi
{
    int ar_ptisis;
    int theseis;
    int epibates;
```

Ορισμός της δομής **ptisi** με τα κατάλληλα πεδία.

```
};

main()
```

Ορισμός ενός πίνακα **ptiseis**, δομών **ptisi**, με AR (10) θέσεις.

```
{
    struct ptisi ptiseis[AR];
    int i,syn_epibaton,pliotita;
    for (i=0;i<AR;i++)
```

Στη μεταβλητή **syn_epibaton** θα υπολογιστεί το σύνολο των επιβατών όλων των πτήσεων. Στη μεταβλητή **pliotita** το πλήθος των πτήσεων με πληρότητα.

```
{
    printf("Στοιχεία πτήσης No:%d\n",i+1);
    printf("=====\n");
    printf("Αριθμός πτήσης:");
    scanf("%d",&ptiseis[i].ar_ptisis);
    printf("Πλήθος θέσεων:");
    scanf("%d",&ptiseis[i].theseis);
    printf("Πλήθος επιβατών:");
    scanf("%d",&ptiseis[i].epibates);
    putchar('\n');
```

Διαβάζονται τα στοιχεία κάθε πτήσης και καταχωρίζονται στις θέσεις του πίνακα **ptiseis** (στα αντίστοιχα πεδία).

```
}
syn_epibaton=0;
pliotita=0;
```

Δίδονται αρχικές τιμές στις μεταβλητές.

```
for (i=0;i<AR;i++)
{
    syn_epibaton=syn_epibaton+ptiseis[i].epibates;
    if (ptiseis[i].epibates>=ptiseis[i].theseis*80/100)
        pliotita++;
}
```

Υπολογισμός του συνόλου των επιβατών και του πλήθους των πτήσεων με πληρότητα.

```
printf("Συνολικό πλήθος επιβατών = %d\n",syn_epibaton);
printf("Πλήθος πιθήσεων με πληρότητα = %d\n",plirotita);
}
```

13.9

```

struct math
{
    char eponymo[30];
    char onoma[20];
    float bathmos;
    int ilikia;
};

struct tmima
{
    char titlos[3];
    int ar_mat;
    struct math mathites[25];
} tmimata[15];

```

```
#include <stdio.h>
#include <stdlib.h>
#define AR 15

struct math
{
    char eponymo[30];
    char onoma[20];
    float bathmos;
    int ilikia;
};

struct tmima
{
    char titlos[3];
    int ar_mat;
    struct math mathites[25];
} tmimata[15];

main()
{
    int i,j;
    for (i=0;i<AR;i++)
    {
        printf("Στοιχεία τμήματος\n");
        printf("=====
        printf("Τίτλος τμήματος\n");
```

Για μεγαλύτερη ευελιξία δηλώνουμε το πλήθος των τμημάτων σαν σταθερά με όνομα **AR**.

Στο πίνακα **tmimata** καταχωρίζονται τα στοιχεία των 15 τμημάτων του σχολείου.

```

gets(tmimata[i].titlos);
printf("Πλήθος μαθητών:");
scanf("%d",&tmimata[i].ar_mat);
getchar();
printf("\nΟι %d μαθητές του τμήματος\n",tmimata[i].ar_mat);
printf("=====\n");
for (j=0;j<tmimata[i].ar_mat;j++)
{
    printf("\nΣτοιχεία μαθητή No:%d\n",j+1);
    printf("=====\n");
    printf("Επώνυμο:");
    gets(tmimata[i].mathites[j].eponymo);
    printf("Όνομα:");
    gets(tmimata[i].mathites[j].onoma);
    printf("Βαθμός:");
    scanf("%f",&tmimata[i].mathites[j].bathmos);
    printf("Ηλικία:");
    scanf("%d",&tmimata[i].mathites[j].ilikia);
    getchar();
    printf("\n");
}
printf("\n\n");
}

```

Διαβάζονται τα στοιχεία κάθε τμήματος και καταχωρίζονται στις θέσεις του πίνακα **tmimata** (στα αντίστοιχα πεδία).

Διαβάζονται τα στοιχεία κάθε μαθητή του τμήματος και καταχωρίζονται στις θέσεις του πίνακα **mathites** (στα αντίστοιχα πεδία).

👉 Η κλήση της **getchar()** μετά από ορισμένες **scanf()**, χρησιμοποιείται για το διάβασμα του παραμένοντα χαρακτήρα αλλαγής γραμμής στο ρεύμα εισόδου (βλέπε την παράγραφο "Η scanf() και τα μικρά της προβλήματα" στη σελίδα 104 του βιβλίου).

13.10 Μετάδοση του πίνακα δομών της προηγούμενης ασκήσεως. Να γραφεί συνάρτηση στην οποία θα μεταβιβάζεται ο πίνακας **tmimata** και θα επιστρέφει το ακόλουθο: *

- Το σύνολο πλήθος των μαθητών του σχολείου.
- Το μέσο όρο της βαθμολογίας των μαθητών για κάθε τμήμα.
- Τα τμήματα με το μεγαλύτερο μέσο όρο βαθμολογίας.
- Το σύνολο των μαθητών με βαθμολογία πάνω από 18.5 εντάσσεται στο τμήμα στο οποίο ανήκουν.

```

void display(struct tmima tm[])
{
    int synolo_mathiton,i,j;
    float max_mo,mo[AR],sum;
    for (i=0;i<AR;i++)
    {
        synolo_mathiton=synolo_mathiton+tmimata[i].ar_mat;
    }
    max_mo=0;
}

```

Στη συνάρτηση μεταβιβάζεται ένας πίνακας δομών **tmima**.

Στον πίνακα **mo[]** θα καταχωρίζονται οι μέσοι όροι των βαθμολογιών των τμημάτων.

Στη μεταβλητή **synolo_mathiton** υπολογίζεται το σύνολο των μαθητών από όλα τα τμήματα.

```

for (i=0;i<AR;i++)
{
    sum=0;
    for (j=0;j<tmimata[i].ar_mat;j++)
    {
        sum=sum+tmimata[i].mathites[j].bathmos;
    }
    mo[i]=sum/tmimata[i].ar_mat;
    //εύρεση του μέγιστου μέσου όρου
    if (mo[i]>max_mo) max_mo=mo[i];
    printf("Ο μεσος όρος του τμήματος %s είναι
           %f\n",tmimata[i].titlos,mo[i]);
}
//Εμφάνιση των τμημάτων με μέσο όρο ίσο με τον μέγιστο μέσο όρο.
printf("\nΤα τμήματα με το μεγαλύτερο μέσο όρο %f\n",max_mo);
for (i=0;i<AR;i++)
{
    if (mo[i]==max_mo) puts(tmimata[i].titlos);
}
printf("\nΜαθητές με βαθμό πάνω από 18.5\n");
printf("\n===== \n");
for (i=0;i<AR;i++)
{
    for (j=0;j<tmimata[i].ar_mat;j++)
    {
        if (tmimata[i].mathites[j].bathmos>=18.5)
            puts(tmimata[i].mathites[j].eponymo);
    }
}
}

```

Στη μεταβλητή **sum** υπολογίζεται το σύνολο των βαθμών των μαθητών ενός τμήματος.

Στον πίνακα **mo[]** καταχωρίζονται οι μέσοι όροι των βαθμολογιών κάθε τμήματος.

Στην περίπτωση που ο μαθητής έχει βαθμό πάνω από 18.5, εμφανίζει το επώνυμό του στην οθόνη.

Ασκήσεις Κεφαλαίου 14

14.1 Να γράψετε πρόγραμμα το οποίο να δημιουργεί στο αρχείο `arithmoi` και να καταχωρίζει στο αρχείο τους αριθμούς από το 1 μέχρι το 100.

```
main()
{
    FILE *fp;
    int i;
    if((fp=fopen("arithmoi","w")) == NULL)
    {
        puts("Πρόβλημα στο άνοιγμα του αρχείου");
        exit(2);
    }
    for(i=1;i<=100;i++)
        fprintf(fp,"%d\n",i);
    fclose(fp);
}
```

Ανοίγει το αρχείο `arithmoi` και ταυτόχρονα ελέγχει αν υπήρξε πρόβλημα στο άνοιγμα του αρχείου (βλέπε σελ. 332 του βιβλίου).

Καταχωρεί στο αρχείο τους αριθμούς από το 1 μέχρι το 100. Οι αριθμοί χωρίζονται με αλλαγή γραμμής.

Κλείνει το αρχείο `arithmoi`.

14.2 Να γράψετε πρόγραμμα το οποίο να διαβάζει το σκελετωμένο αρχείο `input` και να μετράει τον αριθμό των αριθμών που είναι καταχωρημένοι σε αυτό. Να υπολογιστεί ο μέσος όρος των αριθμών.

```
main()
{
    FILE *fp;
    float ar,mo,synolo=0.0;
    int cnt=0;
    if((fp=fopen("input","r")) == NULL)
    {
        puts("Πρόβλημα στο άνοιγμα του αρχείου");
        exit(2);
    }
    while(!feof(fp))
    {
        fscanf(fp,"%f",&ar);
        cnt++;
        synolo=synolo+ar;
    }
    mo=synolo/cnt;
    fclose(fp);
    printf("Σύνολο=%f\n",synolo);
    printf("Μέσος όρος=%f\n",mo);
}
```

Παίρνουμε τη γενική περίπτωση που το αρχείο μπορεί να περιέχει και δεκαδικούς αριθμούς.

Η μεταβλητή `cnt` χρησιμοποιείται για την καταμέτρηση του πλήθους των αριθμών στο αρχείο.

Εφόσον δεν γνωρίζουμε το πλήθος των αριθμών του αρχείου, διαβάζουμε έναν-έναν τους αριθμούς μέχρι να φτάσουμε στο τέλος του αρχείου.

Διαβάζει έναν αριθμό και τον καταχωρεί στη μεταβλητή `ar`.

Αυξάνει το πλήθος (`cnt`) κατά 1 και το σύνολο (`synolo`) κατά τον αριθμό `ar`.

Κλείνει το αρχείο `input`.

14.3

Να γράψετε συνάρτηση η οποία να προσέτασσε στο τέλος ενός αρχείου με όνομα **output** τον αριθμό που βρίσκεται μέσα σε έναν πίνακα **int a[100]**. Ο αριθμός που μεταβιβάζεται στη συνάρτηση ως παράμετρος *

```
int append(int pin[])
{
    FILE *fp;
    int i;
    if((fp=fopen("output","a")) == NULL)
        return 0;
    for(i=0;i<100;i++)
        fprintf(fp,"%d\n",pin[i]);
    fclose(fp);
    return 1;
}
```



Η παραπάνω συνάρτηση επιστρέφει τιμή 0 αν υπήρξε πρόβλημα στο άνοιγμα του αρχείου και τιμή 1 αν δεν υπήρξε κανένα πρόβλημα.

14.4

Να γράψετε πρόγραμμα το οποίο να εμφανίζει στον μεταβλητό και τον μικρότερο και τον μεγαλύτερο αριθμό που είναι αριθμοί που είναι κενά διαχωρισμένοι με ένα διάστημα με όνομα **input**.

```
main()
{
    FILE *fp;
    float min_ar,max_ar,ar;
    if((fp=fopen("input","r")) == NULL)
    {
        puts("Πρόβλημα στο άνοιγμα του αρχείου");
        exit(2);
    }
    fscanf(fp,"%f",&ar);
    min_ar=max_ar=ar;
    while(!feof(fp))
    {
        fscanf(fp,"%f",&ar);
        if (ar>max_ar) max_ar=ar;
        if (ar<min_ar) min_ar=ar;
    }
    fclose(fp);
    printf("Μικρότερος=%f\n",min_ar);
    printf("Μεγαλύτερος=%f\n",max_ar);
}
```

Παίρνουμε τη γενική περίπτωση που το αρχείο μπορεί να περιέχει και δεκαδικούς αριθμούς.

Ανοίγει το αρχείο **input** για διάβασμα. Ταυτόχρονα γίνεται έλεγχος για το σωστό άνοιγμα του αρχείου.

Διαβάζουμε τον πρώτο αριθμό από το αρχείο, και τον καταχωρούμε σαν αρχική τιμή των **min_ar** και **max_ar**.

Εφόσον δεν γνωρίζουμε το πλήθος των αριθμών του αρχείου, διαβάζουμε έναν-έναν τους υπόλοιπους αριθμούς μέχρι να φτάσουμε στο τέλος του αρχείου.

Διαβάζει έναν αριθμό από το αρχείο και τον καταχωρεί στη μεταβλητή **ar**. Αν ο αριθμός είναι μεγαλύτερος από τον **max_ar** τον καταχωρούμε στη θέση **max_ar**, και αν είναι μικρότερος από τον **min_ar** τον καταχωρούμε στη θέση **min_ar**.

Κλείνει το αρχείο **input**.

Τελικά η **max_ar** θα περιέχει τον μεγαλύτερο αριθμό από τους αριθμούς του αρχείου, και η **min_ar** τον μικρότερο.

14.5

Να γράψουμε πρόγραμμα το οποίο να διαβάζει τον αριθμό που βρίσκεται καταχωρημένος στο τριάνο - στα τριάνο **data** και να υπολογίζει το μέσο όρο τους. Δεν γινεται από τρεις αριθμούς, αλλά από τριάνο. *

data
10 19 20
11 19 14
8 12 11
19 18 15
.....

```
main()
{
    FILE *fp;
    float ar1,ar2,ar3,mo;
    if((fp=fopen("data","r")) == NULL)
    {
        puts("Πρόβλημα στο άνοιγμα του αρχείου");
        exit(2);
    }
    while(!feof(fp))
    {
        fscanf(fp,"%f %f %f",&ar1,&ar2,&ar3);
        mo=(ar1+ar2+ar3)/3;
        printf("mo=%f\n",mo);
    }
    fclose(fp);
}
```

Παίρνουμε τη γενική περίπτωση που το αρχείο μπορεί να περιέχει και δεκαδικούς αριθμούς.

Εφόσον δεν γνωρίζουμε το πλήθος των τριάδων του αρχείου, διαβάζουμε τρεις-τρεις τους αριθμούς μέχρι να φτάσουμε στο τέλος του αρχείου.

Διαβάζει μία τριάδα αριθμών και τους καταχωρεί στις μεταβλητές **ar1**, **ar2** και **ar3**.

14.6

Να γράψουμε πρόγραμμα το οποίο να διαβάζει από το αρχείο **sxoleio** τα στοιχεία που υπάρχουν στο παράδειγμα 14.1 και να υπολογίζει το μέσο όρο των μαθητών. Να γράφει το αποτέλεσμα στο αρχείο **stoxeia** και να εμφανίζει το στοιχείο του μαθητή με το μεγαλύτερο μέσο όρο. Να γράφει στο **stoxeia** το στοιχείο του μαθητή με το μεγαλύτερο μέσο όρο. *

```
struct stoxeia
{
    char eponymo[30];
    char taxi[5];
    float mesos_oros;
    int ilikia;
};

main()
{
    FILE *fp;
    struct stoxeia mathitis,kaliteros;
    float max_mo=0.0;
    fp=fopen("sxoleio","rb");
    while(!feof(fp))
    {
        fread(&mathitis,sizeof(struct stoxeia),1,fp);
        if(mathitis.mesos_oros>max_mo)
        {
```

Στη μεταβλητή **mathitis** θα καταχωρούνται τα στοιχεία του κάθε μαθητή που διαβάζουμε από το αρχείο. Στη μεταβλητή **kaliteros** θα καταχωρηθούν τα στοιχεία του μαθητή με τον μεγαλύτερο μέσο όρο. Στη μεταβλητή **max_mo** θα καταχωρείται ο μεγαλύτερος μέχρι στιγμής μέσος όρος. Θέτουμε αρχική τιμή 0.

Διαβάζει μία έγγραφη από το αρχείο και την καταχωρεί στη μεταβλητή **mathitis** (βλέπε παράδειγμα Π14.5 σελ. 352 του βιβλίου).

```

        max_mo=mathitis.mesos_oros;
        kaliteros=mathitis;
    }
}
fclose(fp);
printf("Στοιχεία καλύτερου μαθητή\n");
printf("Επώνυμο:%s\n", kaliteros.eponymo);
printf("Ταξη:%s\n", kaliteros.taxi);
printf("Μέσος όρος:%f\n", kaliteros.mesos_oros);
printf("Ηλικία:%d\n", kaliteros.ilikia);
}

```

Στη περίπτωση που ο μαθητής που διαβάστηκε έχει μεγαλύτερο μέσο όρο από τον μέχρι στιγμής μέγιστο (max_mo) τότε τα στοιχεία του καταχωρούνται στη μεταβλητή δομής **kaliteros**.

- 👉 Διαβάζουμε μια-μια εγγραφή και κάθε φορά τα στοιχεία της τα αποθηκεύουμε στη μεταβλητή **mathitis**. Η επαναλαμβανόμενη διαδικασία σταματάει όταν διαβάσουμε και την τελευταία εγγραφή του αρχείου όποτε η **feof(fp)** επιστρέφει τιμή αλήθεια.
- 👉 Μετά το τέλος της επαναληπτικής διαδικασίας, η μεταβλητή **kaliteros** θα περιέχει τα στοιχεία του μαθητή με τον μεγαλύτερο μέσο όρο.

14.7

Υποθέτουμε ότι έχουμε το αρχείο **sxoleio** που περιλαμβάνει τα παρακάτω στοιχεία (βλ. 14.3). Να γράψετε πρόγραμμα το οποίο να διαβεί και να εμφανίσει τα στοιχεία της 15ης εγγραφής. *

```

struct stoixeia
{
    char eponymo[30];
    char taxi[5];
    float mesos_oros;
    int ilikia;
};

main()
{
    FILE *fp;
    struct stoixeia mathitis;
    fp=fopen("sxoleio","rb");
    fseek(fp,14*sizeof(struct stoixeia),0);
    fread(&mathitis,sizeof(struct stoixeia),1,fp);
    fclose(fp);
    printf("Στοιχεία 15ου μαθητή");
    printf("Επώνυμο:%s\n",mathitis.eponymo);
    printf("Ταξη:%s\n", mathitis.taxi);
    printf("Μέσος όρος:%f\n", mathitis.mesos_oros);
    printf("Ηλικία:%d\n", mathitis.ilikia);
}

```

Η **fseek()** τοποθετεί τον δείκτη θέσης του αρχείου στην αρχή της 15ης εγγραφής (βλέπε σελίδα 343 του βιβλίου).

Διαβάζει τα στοιχεία της 15ης εγγραφής και τα καταχωρεί στη μεταβλητή **mathitis**.

14.8

Υποθέτουμε ότι έχουμε το αρχείο **sxoleio** που περιλαμβάνει τα παρακάτω στοιχεία (βλ. 14.3). Να γράψετε πρόγραμμα το οποίο να αναζητά το σφραγισμένο στοιχείο της 15ης εγγραφής. Έχουμε στο αρχείο 100 στοιχεία που χωρίζονται με τον χαρακτήρα '\n'. Να αναζητήσουμε το στοιχείο της 15ης εγγραφής. *

```

struct stoixeia

```

```

{
    char eponymo[30];
    char taxi[5];
    float mesos_oros;
    int ilikia;
};

main()
{
    FILE *fp;
    struct stoixeia mathitis;
    fp=fopen("sxoleio", "rb+");
    fseek(fp, 14*sizeof(struct stoixeia), 0);
    printf("Επώνυμο:");
    scanf("%s", mathitis.eponymo);
    printf("Τάξη:");
    scanf("%s", mathitis.taxi);

    printf("Μέσος όρος:");
    scanf("%f", &mathitis.mesos_oros);
    printf("Ηλικία:");
    scanf("%d", &mathitis.ilikia);
    fwrite(&mathitis, sizeof(struct stoixeia), 1, fp);
    fclose(fp);
}

```

Ανοίγει το αρχείο για ανάγνωση/εγγραφή

Τοποθετεί τον δείκτη θέσης του αρχείου στην αρχή της 15ης εγγραφής.

Ζητάει από τον χρήστη να πληκτρολογήσει τα στοιχεία του μαθητή και τα καταχωρεί στα αντίστοιχα πεδία της μεταβλητής **mathitis**.

Καταχωρεί τα στοιχεία της μεταβλητής **mathitis** στη 15η εγγραφή του αρχείου.

14.9

Υποθέτουμε ότι έχουμε το αρχείο **sxoleio** που δημιουργήθηκε στο παράδειγμα Π14.4. Σήμερα πρόκειται το σχολείο να κάνει μια εξέταση. Οι μαθητές που θα εξεταστούν είναι οι μαθητές που έχουν μέσο όρο μεγαλύτερο ή ίσο του 5. Η εξέταση θα γίνει με τη μέθοδο που θα ορίσει ο δάσκαλος. Η εξέταση θα γίνει με τη μέθοδο που θα ορίσει ο δάσκαλος.

```

struct stoixeia
{
    char eponymo[30];
    char taxi[5];
    float mesos_oros;
    int ilikia;
};

main()
{
    FILE *fp;
    struct stoixeia mathitis;
    char ch;
    fp=fopen("sxoleio", "rb");
    printf("Δώσε χαρακτήρα:");
    ch=getch();
    while(!feof(fp))
    {

```

Στη μεταβλητή **mathitis** θα καταχωρούνται τα στοιχεία του κάθε μαθητή που διαβάζουμε από το αρχείο.

Διαβάζει μία εγγραφή από το αρχείο και την καταχωρεί στη μεταβλητή **mathitis** (βλέπε παράδειγμα Π14.5 σελ. 352 του βιβλίου).

```

        fread(&mathitis,sizeof(struct stoixeia),1,fp);
        if (mathitis.eponymo[0]==ch)
        {
            printf("%s %d\n",mathitis.eponymo,mathitis.ilikia);
        }
    }
    fclose(fp);
}

```

Στη περίπτωση που ο πρώτος χαρακτήρας του επωνύμου είναι αυτός που δώσαμε (ch), τότε εμφανίζει το επώνυμο και την ηλικία του μαθητή.

👉 Διαβάζουμε μια-μια εγγραφή και κάθε φορά τα στοιχεία της τα αποθηκεύουμε στη μεταβλητή **mathitis**. Η επαναλαμβανόμενη διαδικασία σταματάει όταν διαβάσουμε και την τελευταία εγγραφή του αρχείου όποτε η **feof(fp)** επιστρέφει τιμή αλήθεια.

14.10 Ένα πρόγραμμα σε C ανοίγει ένα αρχείο και τονίζει τον πρώτο χαρακτήρα με τον εξής τρόπο. Στο αρχείο υπάρχει ο κώδικας εξομολογία χρωματισμένη από τον κανονικό. Δηλαδή αν για παράδειγμα η λέξη "HELLO" υπάρχει η λέξη "ΕΙΣΑΓΗΓΗ". Να γράψετε πρόγραμμα το οποίο θα διαβάζει το κείμενο ενός αρχείου, θα το επεξεργαστεί όπως φαίνεται παρακάτω, και θα εμφανίζει το αποτέλεσμα στην οθόνη. 🌟

```

main()
{
    FILE *fp;
    char ch,file_in[30];
    printf("Δώσε όνομα αρχείου:");
    gets(file_in);
    fp=fopen(file_in,"r");
    if (fp==NULL)
    {
        printf("Πρόβλημα στο άνοιγμα αρχείου");
        exit(2);
    }
    while(ch!=EOF)
    {
        ch=fgetc(fp);
        putchar(ch-1);
    }
    fclose(fp);
}

```

Άνοιγμα του αρχείου για ανάγνωση δεδομένων.

Διαβάζει έναν χαρακτήρα από το αρχείο και τον καταχωρεί στη μεταβλητή **ch**.

Εμφανίζει στην οθόνη τον προηγούμενο χαρακτήρα από αυτόν που διάβασε.

14.11 Ποια από τα επόμενα αληθεύουν: ★

- ☑ Ο χειρισμός ενός αρχείου στη C γίνεται μέσω ενός δείκτη τύπου FILE.
- ❑ Ένα αρχείο είναι μια σειρά από bit.
- ☑ Όταν ζητήσουμε να ανοίξουμε ένα αρχείο για εγγραφή και το αρχείο δεν υπάρχει, τότε δημιουργείται.
- ❑ Δεν μπορούμε να έχουμε περισσότερα από δύο αρχεία ταυτόχρονα ανοιχτά.
- ☑ Η **fseek()** είναι η συνάρτηση με την οποία επιτυγχάνουμε τυχαία προσπέλαση σε ένα αρχείο.

- ☑ Τα προκαθορισμένα ρεύματα είναι δείκτες τύπου FILE.
- ☑ Κάθε διαφορετικό ρεύμα χρησιμοποιεί διαφορετικό ενδιάμεσο αποθηκευτικό χώρο (buffer).
- ☐ Η C χειρίζεται τα προκαθορισμένα ρεύματα με διαφορετικό τρόπο από τα υπόλοιπα.

14.12

Έστω αρχείο `bathmoi.txt` με όνομα `bathmoi`, είναι καταχωρισμένο το αποτέλεσμα της βαθμολογίας των 100 φοιτητών ενός τμήματος σε τρία μαθήματα. Να γραφεί πρόγραμμα το οποίο το διαβάζει το στοιχείο από το αρχείο και να εμφανίζει το όνομα του φοιτητή με το μέσο όρο του μέσο όρο `mo` * *

bathmoi.txt

Ανδρέου Νίκος 5.5 7 9
Μάκρας Μένης 10 4 6.5
Τόπας Πάτροκλος 8 7 3
Καντσά Βενετία 9.5 8 8
.....

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

```
main()
```

```
{
```

```
    int i;
```

```
    char epon[30], onom[30], maxonom[30], maxepon[30];
```

```
    float b1, b2, b3, mo, maxmo;
```

```
    FILE *fp;
```

```
    //άνοιγμα του αρχείου
```

```
    fp=fopen("bathmoi.txt", "r");
```

```
    fscanf(fp, "%s %s %f %f %f", epon, onom, &b1, &b2, &b3);
```

```
    mo=(b1+b2+b3)/3;
```

```
    maxmo=mo;
```

```
    strcpy(maxepon, epon);
```

```
    strcpy(maxonom, onom);
```

```
    for(i=2; i<=100; i++)
```

```
    {
```

```
        fscanf(fp, "%s %s %f %f %f", epon, onom, &b1, &b2, &b3);
```

```
        mo=(b1+b2+b3)/3;
```

```
        if(mo>maxmo)
```

```
        {
```

```
            maxmo=mo;
```

```
            strcpy(maxepon, epon);
```

```
            strcpy(maxonom, onom);
```

```
        }
```

```
    }
```

```
    fclose(fp);
```

```
    printf("%s %s\n", maxepon, maxonom);
```

```
}
```

Διαβάζει την πρώτη γραμμή του αρχείου και καταχωρίζει το επώνυμο στον πίνακα **epon**, το όνομα στον πίνακα **onom** και τους βαθμούς στις μεταβλητές **b1, b2 & b3**.

Ανάθεση των στοιχείων του πρώτου φοιτητή, ως αρχικές τιμές στη μεταβλητή **maxmo** και στους πίνακες **maxepon** και **maxonom**.

Διαβάζει τις επόμενες 99 γραμμές του αρχείου.

Ελέγχει αν ο μέσος όρος του φοιτητή είναι μεγαλύτερος από τον μέχρις στιγμής μέσο όρο. Αν είναι καταχωρίζει τα στοιχεία του στη μεταβλητή **maxmo** και στους πίνακες **maxepon** και **maxonom**.

14.13

Δε ένα πρόγραμμα που είναι ικανό να ανοίγει τα αρχεία εισόδου και εξόδου και να δημιουργεί τα αρχεία αυτά. Να γράψει πρόγραμμα το οποίο: ***

- Θα πάρει το όνομα του αρχείου
- Θα δημιουργήσει στοιχεία από το αρχείο. Αν δημιουργήσει δύο νέα αρχεία. Τα νέα αρχεία θα έχουν το όνομα με το αρχικό αλλά με διαφορετικές προεκτάσεις. Ερ και αρ αντίστοιχα.
- Στο αρχείο εισόδου θα γράφει τα στοιχεία που εισάγει χωρίς να βαθμύ (>=5) και στο αρχείο με προεκτάση αρ το ανάμεσα στα στοιχεία του.
- Στο αρχείο με προεκτάση ερ το μεταξύ των στοιχείων καθώς και τα στοιχεία που γράφει το μέθριο.

Αναγνώστου Νίκος 9
Νικολάου Τάκης 6.5
Παπής Γιώργος 4
.....

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

main()
{
    int ar=0;
    char epon[30], onom[30], arxeio[30], arxeio_ep[30], arxeio_ap[30];
    float b;
    printf("Δώσε ένα όνομα αρχείου:");
    gets(arxeio);
    FILE *fp, *fp_ep, *fp_ap;
    fp=fopen(arxeio, "r");
    if (fp==NULL)
    {
        puts("Πρόβλημα στο ανοίγμα του αρχείου");
        exit(1);
    }
    strcpy(arxeio_ep, arxeio);
    strcpy(arxeio_ap, arxeio);
    fp_ep=fopen(strcat(arxeio_ep, ".ep"), "w");
    fp_ap=fopen(strcat(arxeio_ap, ".ap"), "w");
    while (!feof(fp))
    {
        fscanf(fp, "%s %s %f", epon, onom, &b);
        if (b>=5)
            fprintf(fp_ep, "%s %s %f\n", epon, onom, b);
        else
            fprintf(fp_ap, "%s %s %f\n", epon, onom, b);
        ar++;
    }
    fclose(fp);
    fclose(fp_ep);
    fclose(fp_ap);
    printf("%d\n", ar);
}
```

Ανοίγει το αρχείο με το όνομα που δώσαμε.

Ελέγχει αν το άνοιγμα έγινε χωρίς πρόβλημα

Δημιουργεί τα ονόματα για τα δύο αρχεία εξόδου και ανοίγει αυτά τα αρχεία.

Διαβάζει μια γραμμή δεδομένων από το αρχείο εισόδου.

Ανάλογα με τον βαθμό γράφει τα στοιχεία στο αντίστοιχο αρχείο εξόδου.

14.14

Να γράψετε πρόγραμμα το οποίο να διαβάζει το όνομα και τον αριθμό μητρώου για 100 φοιτητές και να φυλάσσει τα δεδομένα σε πίνακες. Το πρόγραμμα θα δημιουργεί ένα δεύτερο αρχείο το οποίο προσεγγίζει με όνομα ΜΗΤΡΟΟ, στο οποίο θα καταχωριστεί το περιεχόμενο των πινάκων. Κάθε εγγραφή του αρχείου θα κωδικοποιείται με το κέπια του αριθμού μητρώου και το όνομα του φοιτητή. ***

```
#include <stdio.h>
#include <stdlib.h>
#define foit 100

main()
{
    int ar=0,mitroa[foit],i;
    char onomata[foit][30];
    FILE *fp;
    for (i=0;i<foit;i++)
    {
        printf("Δώσε όνομα:");
        gets(onomata[i]);
        printf("Δώσε αριθμό μητρώου:");
        scanf("%d",&mitroa[i]);
        getchar();
    }
    fp=fopen("MHTR00","wb");
    if (fp==NULL)
    {
        puts("Πρόβλημα στο άνοιγμα του αρχείου");
        exit(1);
    }
    for (i=0;i<foit;i++)
    {
        fwrite(&mitroa[i],sizeof(int),1,fp);
        fwrite(onomata[i],30,1,fp);
    }
    fclose(fp);
}
```

Το πλήθος των φοιτητών ορίζεται ως η σταθερά **foit**.

Διάβασμα των στοιχείων και καταχώριση τους στους πίνακες **onomata** και **mitroa**.

Διαβάζει τον χαρακτήρα αλλαγής γραμμής που έχει μείνει στο ρεύμα εισόδου (βλέπε βιβλίο σελίδα 104).

Άνοιγμα του αρχείου **MHTR0A** για εγγραφή σε δυαδική μορφή (wb).

Εγγραφή των περιεχομένων των πινάκων στο αρχείο.

14.15

Με βάση το δυαδικό αρχείο που δημιουργήθηκε από την προηγούμενη άσκηση 14.14 να γράψετε πρόγραμμα το οποίο: ***

- Θα ανοίγει το αρχείο ΜΗΤΡΟΟ
- Θα εμφανίζει το στοιχείο της ποσότητας της 100ης εγγραφής
- Θα δώσει τον αριθμό μητρώου ενός φοιτητή και θα εμφανίσει το όνομα του. 2 η περίπτωση που δεν υπάρχει αριθμός μητρώου θα εμφανίσει το μήνυμα "Δεν υπάρχει".


```
#include <stdio.h>
#include <stdlib.h>
#define foit 100
main()
{
    int i, mitroo, ar, brika;
    char onoma[30];
    FILE *fp;
    fp=fopen("MHTROO", "rb");
    if (fp==NULL)
    {
        puts("Πρόβλημα στο άνοιγμα του αρχείου");
        exit(1);
    }
    fseek(fp, 0, SEEK_SET);
    fread(&mitroo, sizeof(int), 1, fp);
    fread(onoma, 30, 1, fp);
    printf("%s %d\n", onoma, mitroo);

    fseek(fp, -34, SEEK_END);
    fread(&mitroo, sizeof(int), 1, fp);
    fread(onoma, 30, 1, fp);
    printf("%s %d\n", onoma, mitroo);

    printf("Δώσε αριθμό μητρώου:");
    scanf("%d", &ar);

    brika=0;
    fseek(fp, 0, SEEK_SET);
    for (i=0; i<foit; i++)
    {
        fread(&mitroo, sizeof(int), 1, fp);
        fread(onoma, 30, 1, fp);
        if (mitroo==ar)
        {
            printf("%s %d\n", onoma, mitroo);
            brika=1;
            break;
        }
    }
    if (brika==0)
        printf("Δεν υπάρχει αυτό το μητρώο\n");
    fclose(fp);
}
```

Οι εγγραφές του αρχείου **MHTROO** έχουν μέγεθος 34 byte. 4 για τον αριθμό μητρώου (sizeof(int)) και 30 για το όνομα.

Άνοιγμα του αρχείου **MHTROO** για ανάγνωση σε δυαδική μορφή (rb).

Τοποθέτηση του δείκτη αρχείου στην αρχή.

Διάβασμα και εμφάνιση των στοιχείων της εγγραφής.

Τοποθέτηση του δείκτη αρχείου 34 bytes πριν από το τέλος. Δηλαδή στην αρχή της τελευταίας εγγραφής.

Διάβασμα και εμφάνιση των στοιχείων της εγγραφής.

Διαβάζει μια-μια τις εγγραφές του αρχείου.

Εντοπισμός της εγγραφής με αρ. μητρώου **ar**.

Μόλις εντοπιστεί η εγγραφή, διακόπτεται η διαδικασία αναζήτησης

Στην περίπτωση που δεν εντοπιστεί η εγγραφή ...

Ασκήσεις Κεφαλαίου 15

15.1

Νοητάει αναδρομική συνάρτηση, που να υπολογίζει το άθροισμα της σειράς $1/1 + 1/2 + 1/3 + 1/4 + \dots$ Η συνάρτηση θα ονομάζεται `par` με πρότυπο πηλίκο `n`. **

```
float par(int n)
{
    float p;
    if (n==1) return 1.0;
    p=1.0/n+par(n-1);
    return p;
}
```

Μη αναδρομική περίπτωση.

15.2

Νοητάει το εύρος στην επόμενη αναδρομική συνάρτηση. *

```
int par(int n)
{
    int p;
    p=n+par(n-1);
    return p;
}
```

Η συνάρτηση δεν έχει μια τουλάχιστον μη-αναδρομική περίπτωση (βλέπε σελίδα 363 του βιβλίου).

15.3

Η κάνει επόμενη συνάρτηση. *

```
int par(int n)
{
    int p;
    if (n==1) return 0;
    p=n+par(n/2);
    return p;
}
```

Τι τιμή θα επιστρέψει η `par(20)`;

☞ Η συνάρτηση επιστρέφει σαν τιμή το άθροισμα της σειράς $n + n/2 + n/4 + \dots$ μέχρι το πηλίκο να είναι διαφορετικό από το 1.

☞ Η κλήση της `par(20)` θα επιστρέψει το 37 ($20 + 10 + 5 + 2 + 0$).

15.4

Ο σκοπός της άσκησης είναι να προσεγγιστεί το π . Η πρόταση προτείνεται το οποίο να χρησιμοποιήσει τον τύπο και να υπολογιστεί την τιμή π . Εάν προηγουμένως είναι δεξιά μέγιστη ή χαμηλότερη από την αναδρομική συνάρτηση. Να προσεγγιστεί μέχρι να 1000, πάλι σε άρα. **

$$\pi = 2 \times \frac{2}{\sqrt{2}} \times \frac{2}{\sqrt{2+\sqrt{2}}} \times \frac{2}{\sqrt{2+\sqrt{2+\sqrt{2}}}}$$

```
double par(int n);
double all(int k);

main()
{
    printf("%lf\n", all(1000));
}
```

```
double par(int n)
{
    double p;
    if(n==0) return 0;
    p=sqrt(2+par(n-1));
    return p;
}
```

Μη αναδρομική περίπτωση.

Αναδρομική κλήση της συνάρτησης.

```
double all(int k)
{
    double p;
    if(k==0) return 2;
    p=2/par(k)*all(k-1);
    return p;
}
```

Μη αναδρομική περίπτωση

Αναδρομική κλήση της συνάρτησης.

- 👉 Η συνάρτηση **par()** χρησιμοποιεί αναδρομική διαδικασία για τον υπολογισμό του παρονομαστή κάθε μέλους της ακολουθίας.
- 👉 Η συνάρτηση **all()** χρησιμοποιεί αναδρομική διαδικασία για τον υπολογισμό του γινομένου όλων των μελών της ακολουθίας.

15.5 Ποια από τα επόμενα αληθεύουν: ★

- ☒ Κάθε αναδρομική συνάρτηση πρέπει να έχει μία τουλάχιστον μη αναδρομική περίπτωση.
- ☐ Όλες οι συναρτήσεις μπορούν να γραφούν με αναδρομική μορφή.
- ☒ Μια αναδρομική συνάρτηση μπορεί να επιφέρει εξάντληση της μνήμης του Η/Υ.
- ☐ Όλες οι γλώσσες προγραμματισμού υποστηρίζουν αναδρομικές συναρτήσεις.
- ☒ Μια αναδρομική συνάρτηση πρέπει να έχει τουλάχιστον μία παράμετρο.

15.6 Ο αλγόριθμος του βιβλίου για τον υπολογισμό του μέγιστου κοινού διαιρέτη δύο θετικών ακέραιων a και b περιγράφεται ως εξής:

Βήμα 1: Θέσε στο a τον μεγαλύτερο και στο b τον μικρότερο αριθμό.

Βήμα 2: Στην περίπτωση που a διαιρεί το b ο $\text{MCD}(a, b)$ είναι το a .

Βήμα 3: Διαβάσε το a με το r και σκέψου το a το επόμενο.

Βήμα 4: Αν $a=0$, ο επόμενος σταματά και το αποτέλεσμα είναι r .

Παράδειγμα 5: Γράψτε την κλήση και τον ορισμό της συνάρτησης που μετρά τον αριθμό των ψηφίων ενός αριθμού.

Η αναδρομική περίπτωση που θέλουμε να υλοποιήσουμε είναι η ακόλουθη: Η συνάρτηση θα πρέπει να υπολογιστεί **mkd()**. Αν δαθεί ο αριθμός n τότε θα υπολογιστεί και να επιστρέψει ως αποτέλεσμα το γινόμενο του αριθμού που δόθηκε με το 10 και να υπολογιστεί το υπόλοιπο της διαίρεσης του n με το 10. Έτσι, αν δαθεί ο αριθμός n τότε η συνάρτηση **mkd()** θα επιστρέψει:

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

main()
{
    int a,b;
    printf("Δώσε δύο αριθμούς:");
    scanf("%d %d",&a,&b);
    printf("Ο MKΔ του %d και του %d είναι %d\n",a,b,mkd(a,b));
}

int mkd(int m, int n)
{
    int temp;
    if (n>m)
    {
        temp=m;
        m=n;
        n=temp;
    }
    if (n==0)
        return m;
    else
        return mkd(n, m % n);
}
```

Αντιμετάθεση των **m** και **n** ώστε στη μεταβλητή **m** να είναι πάντα ο μεγαλύτερος αριθμός.

Μη αναδρομική περίπτωση

Αναδρομική κλήση της συνάρτησης.

15.7

Με δοθείσες δύο $x^y = x \cdot x \cdot x \cdot \dots$ υπολογιστέα αναδρομική συνάρτηση η οποία να υπολογίζει το x^y . Η συνάρτηση θα δέχεται δύο παραμέτρους x και y και θα επιστρέφει την τιμή της παραμέτρους x^y . Έτσι, να υλοποιηθεί η συνάρτηση **mypow** να χρησιμοποιείται συνάρτηση **mypow**.

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

double mypow(double x, double y);

main()
{
    double a,b;
    printf("Δώσε δύο αριθμούς:");
    scanf("%lf %lf",&a,&b);
    printf("%f εις την %f = %f\n",a,b,mypow(a,b));
}
```

```
double mypow(double x, double y)
{
    if (y==0)
        return 1;
    else
        return x*mypow(x,y-1);
}
```

Μη αναδρομική περίπτωση

Αναδρομική κλήση της συνάρτησης.

15.8

Να γράψετε πρόγραμμα το οποίο να διαβάζει τους χαρακτήρες ενός αρχείου και να τονίσει το σύνολο των χαρακτήρων θα βρεθεί ως παράμετρο, στη γραμμή εντολών. * * *

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

main(int argc, char *argv[])
{
    FILE *fin;
    int cnt=0;
    char ch;
    if (argc!=2)
    {
        printf("Λάθος πλήθος παραμέτρων\n");
        exit(2);
    }
    fin=fopen(argv[1], "r");
    if (fin==NULL)
    {
        printf("Πρόβλημα στο αρχείο εισόδου\n");
        exit(2);
    }
    while (!feof(fin))
    {
        ch=fgetc(fin);
        cnt++;
    }
    fclose(fin);
    printf("Το αρχείο %s έχει %d χαρακτήρες\n", argv[1], cnt);
}
```

Έλεγχος για το σωστό πλήθος παραμέτρων.

Άνοιγμα του αρχείου εισόδου, το όνομα του οποίου είναι η πρώτη παράμετρος της γραμμής εντολών.

Διαβάζεται ένας-ένας χαρακτήρας και η μεταβλητή **cnt** καταμετράει το πλήθος τους.

15.9

Να γράψετε πρόγραμμα το οποίο να διαβάζει τις γραμμές ενός αρχείου οι οποίες να έχουν ενά-ον κεφαλαίο σύμβολο χαρακτήρων. Το σύνολο των αρχείων που κερδίζει και το σύνολο των ηττημένων θα δοθούν τότε ως παράμετροι στη γραμμή εντολών. * * *

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

main(int argc, char *argv[])
```

```

{
    FILE *fin;
    int cnt=0;
    char grammi[100];
    if (argc!=3)
    {
        printf("Λάθος πλήθος παραμέτρων\n");
        exit(2);
    }
    fin=fopen(argv[1],"r");
    if (fin==NULL)
    {
        printf("Πρόβλημα στο αρχείο εισόδου\n");
        exit(2);
    }
    while (!feof(fin))
    {
        fgets(grammi,100,fin);
        if (strstr(grammi,argv[2])!=0)
        {
            cnt++;
            printf("%4d. %s",cnt,grammi);
        }
    }
    fclose(fin);
}

```

Άνοιγμα του αρχείου εισόδου, το όνομα του οποίου είναι η πρώτη παράμετρος της γραμμής εντολών.

Διαβάζει μια γραμμή μέχρι 100 χαρακτήρων από το αρχείο.

Ελέγχει αν η γραμμή περιέχει τους χαρακτήρες της δεύτερης παραμέτρου.

Εμφανίζει α/α και τη γραμμή στην οποία εντόπισε τους χαρακτήρες.

15.10 Ένα απλό πρόγραμμα που υπολογίζει το μέσο όρο δύο αριθμών *

```

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

double mo(double x, double y)
{
    return (x+y)/2;
}

double max(double x, double y)
{
    if (x>y) return x; else return y;
}

double min(double x, double y)
{
    if (x>y) return y; else return x;
}

```

```
main()
{
    double (*ptr[3]) (double x, double y);
    int i;
    ptr[0]=&mo;
    ptr[1]=&max;
    ptr[2]=&min;
    for (i=0;i<3;i++) printf("%f\n",ptr[i](10,23));
}
```

Ορίζεται ένας πίνακας δεικτών σε συναρτήσεις τύπου **double**, με δύο παραμέτρους τύπου **double** επίσης.

Στις θέσεις του πίνακα δεικτών **ptr** καταχωρίζονται οι διευθύνσεις των συναρτήσεων **mo()**, **max()** και **min()** αντίστοιχα.

Καλούνται οι συναρτήσεις **mo()**, **max()** και **min()**, με ορίσματα 10 & 23, μέσω των δεικτών του πίνακα **ptr**.

16.500000
23.000000
10.000000

15.11 Να φτιάξετε τις παρακάτω συναρτήσεις, να γραφεί η εφαρμογή που θα καλεί τις συγκεκριμένες συναρτήσεις, με ορίσματα 10 & 20, χρησιμοποιώντας δείκτες, προς τις συναρτήσεις και όχι τα ονόματά τους.

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

void print_mo(double x, double y)
{
    printf("MO=%5.2f\n", (x+y)/2);
}

int einai_isa(double x, double y)
{
    if(x==y) return 1; else return 0;
}
```

```
main()
{
    void (*ptr1)(double x, double y);
    int (*ptr2)(double x, double y);
    int i;
    ptr1=print_mo;
    ptr2=einai_isa;
    ptr1(10,20);
    if (ptr2(10,20))
        printf("Ισα\n");
    else
        printf("Ανισα\n");
}
```

Δήλωση δύο δεικτών (**ptr1** & **ptr2**) σε συναρτήσεις με συγκεκριμένα αποτυπώματα.

Στους δείκτες **ptr1** και **ptr2** ανατίθενται οι διευθύνσεις των συναρτήσεων **print_mo()** και **einai_isa()** αντίστοιχα.

Καλείται η συνάρτηση **print_mo()** μέσω του δείκτη **ptr1**.

Καλείται η συνάρτηση **einai_isa()** μέσω του δείκτη **ptr1**.

15.12 Η εφαρμογή που φτιάξατε στην άσκηση 15.11, να τροποποιηθεί ώστε να μην καλείται η συνάρτηση **print_mo()** με ορίσματα 10 & 20, αλλά να καλείται η συνάρτηση **einai_isa()** με ορίσματα 10 & 20, να τροποποιηθεί η συνάρτηση **einai_isa()** ώστε να επιστρέφει 1 αν τα ορίσματα είναι ίσα, διαφορετικά να επιστρέφει 0.

αποθήκευσε από το 1 μέχρι το 100. Το πρόγραμμα προγράμμιζε και η συνάρτηση το πρόγραμμα σε ξεχωριστά αρχεία. Θα αναφερθεί ένα η διαχωρισμό μεταγλώττισης και σύνδεσης, με χρήση του μεταγλωττιστή gcc.

Έστω ότι στο αρχείο **func.c** έχουμε γράψει τον κώδικα της συνάρτησης **sum()** και στο αρχείο **kyrio.c** τη συνάρτηση **main()** δηλαδή τον κώδικα του κυρίως προγράμματος. Επίσης στο αρχείο κεφαλίδας **myfunc.h** έχουμε γράψει την δήλωση της συνάρτησης **sum()**:

func.c	kyrio.c	myfunc.h
<pre>int sum(int n) { int p; if (n==0) return 0; p=n+sum(n-1); return p; }</pre>	<pre>#include <stdio.h> #include <stdlib.h> #include "myfunc.h" main() { printf("%d\n",sum(100)); }</pre>	<pre>int sum(int n);</pre>

Αρχικά μεταγλωττίσουμε τα δύο πηγαία αρχεία. Με αυτό τον τρόπο παράγονται τα αντίστοιχα αρχεία αντικειμενικού κώδικα:

```
c:\myfiles>gcc -Wall -c kyrio.c
c:\myfiles>gcc -Wall -c func.c
```

Συνδέουμε τα δύο αρχεία αντικειμενικού κώδικα και παράγουμε το τελικό εκτελέσιμο αρχείο **final.exe**:

```
c:\myfiles>gcc kyrio.o func.o -o final
```

Εκτελούμε το εκτελέσιμο αρχείο και βλέπουμε το αποτέλεσμα:

```
c:\myfiles>final
5050
```

15.13 Ποια από τα επόμενα αληθεύουν: ★

- ☒ Ένας δείκτης σε συνάρτηση μπορεί να περιέχει μόνο διευθύνσεις συναρτήσεων που έχουν το ίδιο αποτύπωμα.
- ☒ Η διεύθυνση μιας συνάρτησης αποδίδεται είτε με τον τελεστή & είτε απλά με το όνομά της.
- ☐ Δεν είναι δυνατόν να καλέσουμε μια συνάρτηση χρησιμοποιώντας έναν δείκτη που δείχνει σε αυτή τη συνάρτηση.
- ☒ Όταν αναπτύσσουμε προγράμματα σε ξεχωριστά αρχεία πηγαίου κώδικα, η μεταγλώττιση των αρχείων γίνεται χωριστά μέσω του μεταγλωττιστή, και η σύνδεσή τους σε ένα ενιαίο εκτελέσιμο αρχείο μέσω του συνδετή.
- ☐ Η τεχνική του διαχωρισμού ενός προγράμματος σε περισσότερα του ενός πηγαία αρχεία δεν ενδείκνυται σε μεγάλα προγράμματα διότι προσθέτει πολυπλοκότητα και καθυστερήσεις.

Ασκήσεις Κεφαλαίου 16

16.1

Νο. πρέπει προγράψουν το οποίο να ζητάει μια λέξη, να εμφανίζει τους χαρακτήρες της λέξης κατά αύξοντα αριθμό σειράς, και να την ταξινομήσει στην αλυσίδα. Για παράδειγμα, αν πληκτρολογήσει η λέξη ΑΝΑΝΑΣ, θα εμφανιστεί το λέξη ΑΑΑΑΝΝΤΣ ***

```
main()
{
    int i,k,n;
    char ch,lex[30];
    printf("Δώσε λέξη:");
    gets(lex);
    n=strlen(lex);
    for(i=1;i<n;i++)
    {
        for(k=n-1;k>=i;k--)
        {
            if(lex[k]<lex[k-1])
            {
                ch=lex[k];
                lex[k]=lex[k-1];
                lex[k-1]=ch;
            }
        }
    }
    puts(lex);
}
```

Στη μεταβλητή **n** καταχωρείται το πλήθος των χαρακτήρων του πίνακα **lex[]**.

Χρησιμοποιείται η μέθοδος bubble sort για την ταξινόμηση του πίνακα **lex[]** (βλέπε σελίδα 400 του βιβλίου).

16.2

Νο. πρέπει προγράψουν (ή ταξινομήσουν) πάλι να ταξινομήσει τα σύμβολα σύμφωνα με τη ταξινόμηση. Σε αυτή το δεύτερο γινόμενο τους, οι δείκτες χαρακτήρων ενός πίνακα χαρακτηρίζουν ένα σύμβολο με ένα γινόμενο και ένα σύμβολο ***

```
void order(char x[][40])
{
    int i,k;
    char temp[40];
    for(i=1;i<100;i++)
    {
        for(k=99;k>=i;k--)
        {
            if(x[k][1]<x[k-1][1])
            {
                strcpy(temp,x[k]);
                strcpy(x[k],x[k-1]);
                strcpy(x[k-1],temp);
            }
        }
    }
}
```

Χρησιμοποιείται η μέθοδος bubble sort για την ταξινόμηση του πίνακα συμβολοσειρών (βλέπε σελίδα 412 του βιβλίου).

Συγκρίνεται ο δεύτερος χαρακτήρας των συμβολοσειρών.

Αντιμετάθεση των συμβολοσειρών..

```

    }
}

```

16.3

Αν υποθέσουμε ότι στον πίνακα `lex` υπάρχει το ακόλουθο σύνολο χαρακτήρων: "ΧΣΡΟΥΜΕΝΟΑ". Καθώς είναι το τελευταίο στοιχείο του `lex` μετά από την εκτέλεση του παραπάνω κώδικα θα είναι *

```

int i=0,p1,p2;
char ch1,ch2,temp;
ch1=ch2=lex[0];
p1=p2=0;
while(lex[i]!='\0')
{
    if(ch1>lex[i])
    {
        ch1=lex[i];
        p1=i;
    }
    if(ch2<lex[i])
    {
        ch2=lex[i];
        p2=i;
    }
    i++;
}
temp=lex[0];
lex[0]=lex[p2];
lex[p2]=temp;
temp=lex[i-1];
lex[i-1]=lex[p1];
lex[p1]=temp;

```

Εντοπίζει τον μεγαλύτερο (σε κωδικό) χαρακτήρα του πίνακα και καταχωρεί τη θέση του στην μεταβλητή `p1`.

Εντοπίζει τον μικρότερο (σε κωδικό) χαρακτήρα του πίνακα και καταχωρεί τη θέση του στην μεταβλητή `p2`.

Αντιμεταθέτει τον μεγαλύτερο χαρακτήρα (στη θέση `p1`) με τον πρώτο χαρακτήρα (θέση 0).

Αντιμεταθέτει τον μικρότερο χαρακτήρα (στη θέση `p2`) με τον τελευταίο χαρακτήρα (θέση `i-1`).

👉 Ο παραπάνω κώδικας εντοπίζει τον μεγαλύτερο (σε κωδικό) χαρακτήρα του πίνακα `lex[]` και τον αντιμεταθέτει με τον χαρακτήρα της πρώτης θέσης. Επίσης εντοπίζει τον μικρότερο (σε κωδικό) χαρακτήρα του πίνακα και τον αντιμεταθέτει με τον χαρακτήρα της τελευταίας θέσης.

👉 Ο μεγαλύτερος χαρακτήρας του πίνακα είναι ο 'X' ο οποίος παραμένει στην πρώτη θέση, ενώ ο μικρότερος που είναι το 'A' αντιμετατίθεται με τον τελευταίο χαρακτήρα το 'Σ'. Επομένως ο πίνακας `lex[]` μετά από την εκτέλεση του παραπάνω κώδικα θα περιέχει τους χαρακτήρες "ΧΣΡΟΥΜΕΝΟΑ".

16.4

Να γραφεί συνάρτηση η οποία θα προσθέτει ένα χαρακτήρα, που βρίσκεται στον πίνακα χαρακτήρων που δίνεται, μετά σε έναν υπάρχοντα πίνακα χαρακτήρων. Για παράδειγμα, αν ο πίνακας περιέχει τους χαρακτήρες "ΑΓΓΔΚΜΧ", και ο χαρακτήρας που πρόκειται να προστεθεί είναι ο 'Ε', η συνάρτηση θα έχει ως αποτέλεσμα ο πίνακας των χαρακτήρων "ΑΓΓΔΕΚΜΧ". Η συνάρτηση να δέχεται δύο παραμέτρους: τον πίνακα των χαρακτήρων και το χαρακτήρα που πρόκειται να προστεθεί. *

```
int insert (lex, ch)
char lex[], ch;
{
    int i, n, pos;
    n = strlen (lex);
    pos = n;
    for (i = 0; i < n; i++)
    {
        if (ch <= lex[i])
        {
            pos = i;
            break;
        }
    }
    for (i = n; i >= pos; i--)
    {
        lex[i+1] = lex[i];
    }
    lex[pos] = ch;
    return pos;
}
```

Στη μεταβλητή **n** καταχωρείται το πλήθος των χαρακτήρων της λέξης.

Εντοπίζει τη θέση στην οποία πρέπει να παρεμβληθεί ο χαρακτήρας **ch** και την καταχωρεί στη μεταβλητή **pos**.

Μετακινεί όλους τους χαρακτήρες, από την θέση **pos** μέχρι τέλος, μία θέση δεξιότερα ώστε να γίνει χώρος για τον χαρακτήρα που θα παρεμβληθεί.

Καταχωρεί τον χαρακτήρα στη θέση **pos**.

👉 Παρατηρούμε ότι με την πρόταση **pos=n** η αρχική τιμή της **pos** τίθεται ίση με την τελευταία θέση του πίνακα (n). Αυτό γίνεται ώστε στη περίπτωση που το **ch** δεν είναι μικρότερο από κανέναν χαρακτήρα της λέξης (επόμενος βρόχος for) να τοποθετηθεί στο τέλος της.

👉 Η παραπάνω συνάρτηση επιστρέφει σαν τιμή, τη θέση στην οποία παρεμβλήθηκε ο νέος χαρακτήρας.

16.5 Ποια από τα επόμενα αληθεύουν: ★

- ☒ Η δυαδική αναζήτηση προϋποθέτει ταξινομημένα δεδομένα.
- ☒ Οι διαδικασίες ταξινόμησης είναι γενικά χρονοβόρες σε μεγάλους πίνακες.
- ☐ Σε ένα μεγάλο πίνακα με τυχαίους αριθμούς η καλύτερη μέθοδος ταξινόμησης είναι η μέθοδος της φυσαλίδας.
- ☐ Σε έναν μισοταξινομημένο πίνακα η καλύτερη μέθοδος ταξινόμησης είναι η quick sort.
- ☒ Σε έναν πίνακα με τυχαίους αριθμούς, η μόνη μέθοδος αναζήτησης είναι η σειριακή.
- ☒ Η μέθοδος ταξινόμησης φυσαλίδας βασίζεται στη σύγκριση και αντιμετάθεση γειτονικών θέσεων μνήμης.
- ☒ Η μέθοδος ταξινόμησης *quick sort* βασίζεται στο διαχωρισμό των θέσεων μνήμης του πίνακα με βάση μιας τιμής διαχωρισμού.

16.6

Θεωρούμε ότι έχουμε το επόμενο πρόγραμμα. Το πρόγραμμα αυτό καταχωρίζει σε έναν πίνακα τον αριθμό των γειτονικών που έχουμε τη δυνατότητα κοινόφροντος το ποσό. Οι ονομασίες είναι οι παρακάτω. Πόσοι από αυτούς θα ονομάζονται...

Επίσης, όπως και το πρόγραμμα, το οποίο είναι διαθέσιμο στον δικτυακό μας χώρο, αναφέρεται η σειρά: * * *

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

```
main()
```

```
{
```

```
    int i,k,n=0;
```

```
    char onomata[50][30],temp[30];
```

```
    for(i=0;i<50;i++)
```

```
    {
```

```
        gets(onomata[i]);
```

```
        if (strcmp(onomata[i],"")==0)
```

```
            break;
```

```
        else
```

```
            n++;
```

```
    }
```

```
    for(i=1;i<n;i++)
```

```
    {
```

```
        for(k=n-1;k>=i;k--)
```

```
        {
```

```
            if(strcmp(onomata[k],onomata[k-1])!=-1)
```

```
            {
```

```
                strcpy(temp,onomata[k]);
```

```
                strcpy(onomata[k],onomata[k-1]);
```

```
                strcpy(onomata[k-1],temp);
```

```
            }
```

```
        }
```

```
    }
```

```
    for(i=0;i<n;i++)
```

```
    {
```

```
        puts(onomata[i]);
```

```
    }
```

```
}
```

Διαβάζει ονόματα μέχρι να δοθεί ως όνομα το κενό "" ή μέχρι να συμπληρωθούν 50 ονόματα.

Ο πίνακας **onomata**, ταξινομείται με τη μέθοδο της ταξινόμησης φυσαλίδας.

Συγκρίνονται με απόλυτη αλφαβητική σειρά οι δύο γειτονικές συμβολοσειρές.

Στην περίπτωση που δεν είναι στη σωστή σειρά, γίνεται αντιμετάθεσή τους.

16.7

Δοθέντων τριών αριθμών, να τον τριπλό του αθροίσματος, δηλαδή το άθροισμα κάθε δύο από τους αριθμούς, να βρεθεί η μέγιστη τιμή που θα πάρει το άθροισμα και να βρεθεί ο αριθμός των αριθμών που θα πάρει το άθροισμα. Το πρόγραμμα να εμφανίζει τον τριπλό του αθροίσματος των αριθμών και την καλύτερη βολή τους με την πρώτη σειρά. Ανάλογα τριπλό του άθροισμα με τη μεγαλύτερη βολή και τελευταίο ο αριθμός με τη μικρότερη βολή.

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

```
main()
```

```

{
    int i,j,k;
    char onomata[10][20],temp[20];
    float voles[10][6],max[10],tt;
    for(i=0;i<10;i++)
    {
        printf("Δώσε όνομα %d ->",i+1);
        gets(onomata[i]); //Διαβάζει ένα όνομα και το καταχωρίζει στον πίνακα
        for(j=0;j<6;j++)
        {
            printf("Voli %d:",j+1);
            scanf("%f",&voles[i][j]);
        }
        getchar(); //Διόβαρμα του παραμένοντα χαρακτήρα αλλαγής γραμμής
        puts("-----\n");
    }
    for(i=0;i<10;i++)
    {
        max[i]=voles[i][0];
        for(j=0;j<6;j++)
        {
            if (voles[i][j]>max[i]) max[i]=voles[i][j];
        }
    }
    for(i=1;i<10;i++)
    {
        for(k=9;k>=i;k--)
        {
            if(max[k]>max[k-1])
            {
                tt=max[k];
                max[k]=max[k-1];
                max[k-1]=tt;
                strcpy(temp,onomata[k]);
                strcpy(onomata[k],onomata[k-1]);
                strcpy(onomata[k-1],temp);
            }
        }
    }
    for(i=0;i<10;i++)
    {
        printf("%s %5.2f\n",onomata[i],max[i]);
    }
}

```

Στον πίνακα **max** θα καταχωριστούν οι μέγιστες τιμές των γραμμών του πίνακα **voles**.

Διαβάζει τις τιμές για τις έξι βολές και τις καταχωρίζει στον πίνακα **voles**.

Εύρεση της μέγιστης τιμής για κάθε γραμμή του πίνακα και καταχώρισή της στην αντίστοιχη θέση του πίνακα **max[]**.

Φθίνουσα ταξινόμηση του πίνακα **max**.

Πέρα από την αντιμετάθεση των θέσεων του πίνακα **max**, γίνεται επίσης αντιμετάθεση των αντίστοιχων γραμμών του πίνακα **onomata**.

Εμφάνιση των ονομάτων των αθλητών και των αντίστοιχων μέγιστων βολών τους.

16.8

Νο προσπαθούμε να προσομοιάσουμε την προηγούμενη, σε οριστικό βαθμό, διαδικασία του ονόματος των όσων οδήγησαν στο αρχείο `onomata.txt` και τις βολές τους (όσοι άκυρες, υπό το όνομα `voles.txt`). Λεξιματικά, οι πιο εύκολοι να μην βολή, μετράται είναι 0, το πρόβλημα να εμφανιστεί είναι, το συνολικό μήκος. Για να γίνει βολή, ***

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

void read_files(char on[][20], float vol[][6]);
```

```
main()
{
    int i,j,k,akyres=0;
    char onomata[10][20],temp[20];
    float voles[10][6],max[10],tt;
    read_files(onomata,voles);
    for(i=0;i<10;i++)
    {
        max[i]=voles[i][0];
        for(j=0;j<6;j++)
        {
            if (voles[i][j]>max[i]) max[i]=voles[i][j];
            if (voles[i][j]==0) akyres++;
        }
    }
    for(i=1;i<10;i++)
    {
        for(k=9;k>=i;k--)
        {
            if(max[k]>max[k-1])
            {
                tt=max[k];
                max[k]=max[k-1];
                max[k-1]=tt;
                strcpy(temp,onomata[k]);
                strcpy(onomata[k],onomata[k-1]);
                strcpy(onomata[k-1],temp);
            }
        }
    }
    for(i=0;i<10;i++)
        printf("%s %5.2f\n",onomata[i],max[i]);
    printf("Ακυρες βολές -> %d\n",akyres);
}
```

Γέμισμα των πινάκων `onomata` και `voles` από τα στοιχεία που βρίσκονται στα αντίστοιχα αρχεία.

Η μεταβλητή `akyres` μετράει, τις άκυρες βολές (με μέτρηση 0).

Φθίνουσα ταξινόμηση του πίνακα `max`.

```
FILE *fp1,*fp2;
int i,j;
fp1=fopen("onomata.txt","r");
if (fp1==NULL)
{
    puts("Πρόβλημα στο άνοιγμα του αρχείου onomata.txt");
    exit(1);
}
for(i=0;i<10;i++)
    fscanf(fp1,"%s",on[i]);
fclose(fp1);
fp2=fopen("voles.txt","r");
if (fp2==NULL)
{
    puts("Πρόβλημα στο άνοιγμα του αρχείου voles.txt");
    exit(1);
}
for(i=0;i<10;i++)
{
    for(j=0;j<6;j++)
        fscanf(fp2,"%f",&vol[i][j]);
}
fclose(fp2);
}
```

Άνοιγμα του αρχείου **onomata.txt**.

Διάβασμα των ονομάτων από το αρχείο και καταχώρισή τους στον πίνακα.

Άνοιγμα του αρχείου **voles.txt**.

Διάβασμα των βολών από το αρχείο και καταχώρισή τους στον πίνακα.

16.9

[illegible]

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

int check_sort(int pin[],int n)
{
    int ayxousa=1,fthinousa=1,i;
    for (i=0;i<n-1;i++)
    {
        if (pin[i]>pin[i+1]) ayxousa=0;
        if (pin[i]<pin[i+1]) fthinousa=0;
    }
    if (ayxousa)
        return 1;
    else if (fthinousa)
        return -1;
    else
        return 0;
}
```

Αρχικά θεωρούμε ότι ο πίνακας είναι ταξινομημένος τόσο με αύξουσα όσο και φθίνουσα ταξινόμηση.

Ελέγχονται οι διαδοχικές θέσεις του πίνακα.

Αν βρεθούν τιμές με αντίθετη σειρά τότε μηδενίζονται οι αντίστοιχες μεταβλητές.

Στην περίπτωση που η μεταβλητή **ayxousa** παραμένει με τιμή 1 τότε ο πίνακας έχει αύξουσα ταξινόμηση. Στην περίπτωση που η μεταβλητή **fthinousa** παραμένει με τιμή 1 τότε ο πίνακας έχει φθίνουσα ταξινόμηση. Στην περίπτωση που και οι δύο μεταβλητές έχουν τιμή 0 τότε ο πίνακας δεν είναι ταξινομημένος. Στην περίπτωση που και οι δύο μεταβλητές έχουν τιμή 1 τότε ο πίνακας περιέχει το ίδιο αριθμό σε όλες τις θέσεις του.

```
//Το ακόλουθο πρόγραμμα επιδεικνύει τη χρήση της συνάρτησης
main()
{
    int test[10]={5,67,8,3,56,87,12,14,9,27},tt,i,k;
    printf("Πριν την ταξινόμηση=%d\n",check_sort(test,10));
    //Αυξουσα ταξινόμηση πίνακα test[]
    for(i=1;i<10;i++)
    {
        for(k=9;k>=i;k--)
        {
            if(test[k]<test[k-1])
            {
                tt=test[k];
                test[k]=test[k-1];
                test[k-1]=tt;
            }
        }
    }
    printf("Μετά την πρώτη ταξινόμηση =%d\n",check_sort(test,10));

    //Φθίνουσα ταξινόμηση πίνακα test[]
    for(i=1;i<10;i++)
    {
        for(k=9;k>=i;k--)
        {
            if(test[k]>test[k-1])
            {
                tt=test[k];
                test[k]=test[k-1];
                test[k-1]=tt;
            }
        }
    }
    printf("Μετά την δεύτερη ταξινόμηση =%d\n",check_sort(test,10));
}
```

Αρχικές τιμές του πίνακα test.

16.10 *Το πρόγραμμα δείχνει πως γίνεται να ταξινομηθεί ένας πίνακας με τον αριθμό των στοιχείων που περιέχει (από την θέση 0 μέχρι την τιμή 9). Τα περιεχόμενα του πίνακα είναι ταξινομημένα με αύξουσα σειρά. Εξαιτίας αυτής της αύξουσας διαβάσεως στον πίνακα, ο αλγόριθμος περιεγράφοι να στη βασική δομή ώστε ο πίνακας να διατηρείται πάντοτε ταξινομημένος. * * *

➤ Η συνάρτηση θα δώσει ένα ταξινομημένο και θα επιστρέφει την τιμή int της θέσης που η τιμή 0 είναι. Στην πρώτη περίπτωση θα μεταβεί στο τέλος του πίνακα στη δεύτερη το π και στην τρίτη ο αριθμός (α) που θα είναι το αποτέλεσμα.

➤ Η συνάρτηση θα επιστρέφει τη νέα τιμή του πλήθους στοιχείων του πίνακα, ο οποίος έγινε επιπλέον, όταν ο αριθμός `ar` εισήχθη. Όταν ο πίνακας είναι γεμάτος, επιστρέφει το `n` χωρίς να καταχωρίσει τον αριθμό `ar`.

```
#include <stdio.h>
#include <stdlib.h>
int insert(int pin[],int n, int ar)
{
    int i,thesi=0;
    if (n>=100) return n;
    for (i=0;i<n;i++)
        if (ar>pin[i]) thesi++;
    for (i=n;i>thesi;i--)
        pin[i]=pin[i-1];
    pin[thesi]=ar;
    return n+1;
}
```

Στην περίπτωση που ο πίνακας είναι γεμάτος, επιστρέφει το `n` χωρίς να καταχωρίσει τον αριθμό `ar`.

Εντοπίζεται η θέση στην οποία πρέπει να παρεμβληθεί ο αριθμός `ar`.

Όλοι οι αριθμοί του πίνακα από τη `thesi` και κάτω, μετακινούνται μια θέση χαμηλότερα.

Στη θέση `thesi` καταχωρίζεται ο αριθμός `ar`. Η συνάρτηση επιστρέφει το πλήθος των αριθμών που περιέχει τώρα ο πίνακας.

```
main()
{
    int test[20]={5,67,8,3,56,87,12,14,9,27},tt,i,k,n=10;
    //Αυξουσα ταξινόμηση πίνακα test[]
    for(i=1;i<n;i++)
    {
        for(k=n-1;k>=i;k--)
        {
            if(test[k]<test[k-1])
            {
                tt=test[k];
                test[k]=test[k-1];
                test[k-1]=tt;
            }
        }
    }
    //Εμφάνιση ταξινομημένου πίνακα
    for (i=0;i<n;i++)
        printf("%d\n",test[i]);
    //Παρεμβολή του αριθμού 45 στον πίνακα
    //Η μεταβλητή n παίρνει τη νέα τιμή που επιστρέφει η συνάρτηση
    n=insert(test,n,45);
    //Εμφάνιση πίνακα
    for (i=0;i<n;i++)
        printf("%d\n",test[i]);
}
```

Αρχικές τιμές του πίνακα `test`.

Ταξινόμηση πίνακα `test`.

Ασκήσεις Κεφαλαίου 17

17.1

Να γράψετε πρόγραμμα το οποίο να διαβάζει το πλήθος των αριθμών που περιέχεται υπό καθορισμένη σε έναν πίνακα κενά, και να δηλώνει έναν πίνακα με τόσες θέσεις. Πληκτρολογήστε να χωρηθεί ο πίνακας τα πλήθος των δεδομένων και μετὰ να διαβάσει και να εγγραφεί τα στοιχεία που θα δώσει με το στον πίνακα. *

```
main()
{
    int i,ar,*pin;
    printf("Δώσε πλήθος:");
    scanf("%d",&ar);
    pin=(int *)calloc(ar,sizeof(int));
    for(i=0;i<ar;i++)
    {
        scanf("%d",&pin[i]);
    }
}
```

Δεσμεύει τόσες θέσεις μνήμης τύπου `int` όσες ο αριθμός `ar` που δώσαμε.

Καταχωρεί στις θέσεις μνήμης αριθμούς που ζητάει από το πληκτρολόγιο.

- ☞ Στον δείκτη `pin` καταχωρείται η διεύθυνση που επιστρέφει η `calloc()`. Ο δείκτης `pin` επομένως "δείχνει" στην αρχή του block μνήμης που δέσμευσε η `calloc()` (βλέπε σελίδα 424 του βιβλίου).
- ☞ ΠΡΟΣΟΧΗ στη μετατροπή τύπου (`int *`) του δείκτη που επιστρέφει η `calloc()`. Η μετατροπή είναι απαραίτητη διότι ο δείκτης που επιστρέφει η `calloc()` είναι αρχικά τύπου `void` (βλέπε σελίδα 424 του βιβλίου).
- ☞ Ο `pin` χρησιμοποιείται κανονικά σαν πίνακας με βάση τη σχέση δεικτών και πινάκων (βλέπε σελίδα 248 του βιβλίου).

17.2

Να συμπληρωθεί το πρόγραμμα της προηγούμενης άσκησης, ώστε μετά από την εκκίνηση του να μπορεί να αυξηθεί το μέγεθος του πίνακα στο οποίο δίνει τον αριθμό του μεγέθους *

Θα πρέπει να προστεθεί στον κώδικα της προηγούμενης άσκησης η παύση ή το πρόβλημα.

```
size = (size + 1) * sizeof(int);
```

- ☞ Η `realloc()` αυξάνει το μέγεθος του block μνήμης που "δείχνει" ο δείκτης `ptr` και ο δείκτης που επιστρέφει (και δείχνει στο νέο block) καταχωρείται στην ίδια μεταβλητή δείκτη `ptr`.
- ☞ Είναι πιθανόν το νέο block μνήμης να είναι σε διαφορετική θέση από το αρχικό (βλέπε σελίδα 425 του βιβλίου).

17.3

Θα πρέπει πρώτα να ορίσουμε το δομικό τύπο που θα χρησιμοποιήσουμε για την τιμή των στοιχείων ενός πίνακα και να δεσμεύσουμε χώρο μνήμης που θα χρειαστεί για την καταχώριση των στοιχείων του πίνακα. Όταν ορίσουμε στη δομή **struct** **stoxeia**, όπως με-
τό να ορίσει κάποιος, πρέπει να ορίσει και το πόσο χώρο θα
πρέπει να δώσει. * * *

```
struct stoxeia
{
    char onoma[15];
    char address[20];
    char thl[13];
    int ilikia;
};
```

```
main()
{
    int ar;
    struct stoxeia *pin;
    printf("Δώσε πλήθος μαθητών:");
    scanf("%d",&ar);
    pin=(struct stoxeia *)calloc(ar,sizeof(struct stoxeia));
    printf("Δώσε όνομα:");
    scanf("%s",pin[9].onoma);
    printf("Δώσε διεύθυνση:");
    scanf("%s",pin[9].address);
    printf("Δώσε τηλέφωνο:");
    scanf("%s",pin[9].thl);
    printf("Δώσε ηλικία:");
    scanf("%d",&pin[9].ilikia);
}
```

Δεσμεύει τόσες θέσεις μνήμης τύπου **stoxeia** όσες ο αριθμός **ar** που δώσαμε.

Ζητάει και καταχωρεί τα στοιχεία του μαθητή στη δέκατη θέση του πίνακα (η δέκατη θέση είναι η **pin[9]**).

👉 ΠΡΟΣΟΧΗ στη μετατροπή τύπου (**struct stoxeia ***) του δείκτη που επιστρέφει η **calloc()**. Η μετατροπή είναι απαραίτητη διότι ο δείκτης που επιστρέφει η **calloc()** είναι αρχικά τύπου **void** (βλέπε σελίδα 424 του βιβλίου).

17.4

Θα πρέπει πρώτα να ορίσουμε τον δομικό τύπο που θα χρησιμοποιήσουμε για την τιμή των στοιχείων ενός πίνακα και να δεσμεύσουμε χώρο μνήμης που θα χρειαστεί για την καταχώριση των στοιχείων του πίνακα. Όταν ορίσουμε στη δομή **struct** **stoxeia**, όπως με-
τό να ορίσει κάποιος, πρέπει να ορίσει και το πόσο χώρο θα
πρέπει να δώσει. * * *

Θα πρέπει να προσεθιστεί και ο κώδικας του παραδείγματος 17.2 για να γίνει η πρόταση:

```
k=realloc(k,sizeof(float)*50);
```

👉 Η **realloc()** μειώνει το μέγεθος του block μνήμης που "δείχνει" ο δείκτης **k** (από 100 θέσεις μεγέθους float σε 50) και ο δείκτης που επιστρέφει (ο οποίος δείχνει στο νέο block) καταχωρείται στην ίδια μεταβλητή δείκτη **k**.

👉 Είναι πιθανόν το νέο block μνήμης να είναι σε διαφορετική θέση από το αρχικό (βλέπε σελίδα 425 του βιβλίου).

17.5 Ποια από τα επόμενα αληθεύουν: ★

- ☐ Ο πίνακας αποτελεί μια δυναμική κατανομή μνήμης.
- ☒ Στη δυναμική κατανομή μνήμης υπάρχει περίπτωση να μην είναι δυνατή η δέσμευση της ποσότητας μνήμης που ζητάμε.
- ☒ Η **malloc(5,100)** δεσμεύει 500 byte μνήμης.

- ❑ Η **free(100)** αποδεσμεύει 100 byte από τη μνήμη.
- ☑ Η **calloc()** και η **malloc()** επιτελούν σχεδόν την ίδια λειτουργία.

17.6

Μέσω σε ένα αρχείο με όνομα times.txt είναι καταχωρημένες οι τιμές των τιμών. Ο πρώτος αριθμός του αρχείου υποδεικνύει το πλήθος των τιμών που ακολουθούν. Να γραφεί πρόγραμμα το οποίο να διαβάσει από το αρχείο τις τιμές και να τις καταχωρήσει σε έναν πίνακα τύπου float. Μετά να υπολογίσει το μέσο όρο των τιμών και να ταξινομήσει τον πίνακα με μέθοδο bubble sort.

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

main()
{
    FILE *fp;
    float *pin,sum,mo,temp;
    int i,k,plithos;
    fp=fopen("times.txt","r");
    if (fp==NULL)
    {
        printf("Πρόβλημα στο άνοιγμα του αρχείου");
        exit(2);
    }
    fscanf(fp,"%d",&plithos);
    pin=(float *)calloc(plithos,sizeof(float));
    for(i=0;i<plithos;i++)
        fscanf(fp,"%f",&pin[i]);
    fclose(fp);
    for(i=0;i<plithos;i++)
        sum=sum+pin[i];
    mo=sum/plithos;
    for(i=1;i<plithos;i++)
    {
        for(k=plithos-1;k>=i;k--)
        {
            if(pin[k]>pin[k-1])
            {
                temp=pin[k];
                pin[k]=pin[k-1];
                pin[k-1]=temp;
            }
        }
    }
    printf("Ο μέσος όρος των τιμών είναι %6.3f\n",mo);
    for(i=0;i<plithos;i++)
        printf("%5.2f\n",pin[i]);
}
```

Άνοιγμα του αρχείου

Διάβασμα από το αρχείο του πλήθους (plithos) των δεδομένων που ακολουθούν.

Δυναμική κατανομή μνήμης για το απαιτούμενο πλήθος δεδομένων. Διάβασμα των αριθμών και καταχώρισή τους στον πίνακα pin[].

Υπολογισμός αθροίσματος και μέσου όρου.

Φθίνουσα ταξινόμηση του πίνακα pin[] με τη μέθοδο bubble

Εμφάνιση του μέσου όρου και των τιμών του πίνακα.

Ασκήσεις Κεφαλαίου 18

18.1

Υποθέτουμε ότι έχουμε ένα πίνακα **a** με 100 τυχαιούς αριθμούς. Να γραφεί κώδικας ο οποίος να δημιουργεί μια απλά συνδεδεμένη λίστα με δεδομένα τους αριθμούς του πίνακα **a**. *

```
struct node
{
    int data;
    struct node *next;
} *list_head, *neos;
```

Βλέπε παράδειγμα δημιουργίας μιας απλά συνδεδεμένης λίστας στη σελίδα 437 του βιβλίου.

```
void add_node_to_list();
main()
```

```
{
    int i,a[100];
    for(i=0;i<100;i++) a[i]=rand();
    list_head=NULL;
    for(i=0;i<100;i++)
    {
        add_node_to_list(a[i]);
    }
}
```

Γέμισμα του πίνακα **a[]** με τυχαιούς αριθμούς.

Η αρχική τιμή του χειριστή της λίστας **list_head** τίθεται ίση με NULL.

Προσθήκη των αριθμών του πίνακα στη συνδεδεμένη λίστα.

```
void add_node_to_list(ar)
int ar;
{
    neos = (struct node *)malloc(sizeof(struct node));
    neos->data=ar;
    neos->next = list_head;
    list_head=neos;
}
```

Δέσμευση ενός τμήματος μνήμης, τόσων byte όσο το μέγεθος του τύπου **struct node**.

Καταχώριση του αριθμού στο πεδίο **data** του νέου κόμβου.

Στο πεδίο **next** καταχωρίζεται η διεύθυνση του μέχρι στιγμής κόμβου κεφαλής της λίστας, ενώ στο χειριστή της λίστας **list_head** καταχωρίζεται η διεύθυνση του νέου κόμβου, ο οποίος είναι τώρα στην κορυφή της λίστας.

18.2

Υποθέτουμε ότι έχουμε ένα πίνακα **a** με 100 τυχαιούς αριθμούς. Να γραφεί κώδικας ο οποίος να δημιουργεί μια διπλά συνδεδεμένη λίστα με δεδομένα τους αριθμούς του πίνακα **a**. *

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <malloc.h>
struct node
{
    int data;
    struct node *next;
    struct node *previous;
} *list_head, *neos;
```

Το τμήμα δεδομένων αποτελείται από το πεδίο **data**, στο οποίο καταχωρίζεται ο αριθμός. Για την υλοποίηση της ταξινομημένης λίστας, χρησιμοποιείται μια διπλά συνδεδεμένη λίστα (βλέπε σελίδα 434 του βιβλίου).

```

struct node *find_place();
void add_node_to_list();
main()
{
    int a[100],i;
    list_head=NULL;
    for(i=0;i<100;i++) a[i]=rand();
    for(i=0;i<100;i++) add_node_to_list(a[i]);
    display_all();
}

struct node *find_place(int ar)
{
    struct node *p,*tp;
    p=list_head;
    tp=NULL;
    while (p!=NULL)
    {
        if(ar>=p->data) tp=p;
        p=p->next;
    }
    return tp;
}

void add_node_to_list(ar)
int ar;
{
    struct node *temp_next,*thesi;
    thesi=find_place(ar);
    neos = (struct node *)malloc(sizeof(struct node));
    neos->data=ar;
    if(thesi==NULL)
    {
        if(list_head!=NULL)
        {
            list_head->previous=neos;
            neos->next=list_head;
            neos->previous=NULL;
            list_head=neos;
        }
        else
        {
            neos->next=NULL;
            neos->previous=NULL;
            list_head=neos;
        }
    }
    else

```

Γεμίζει τον πίνακα **a[]** με τυχαίους αριθμούς και τους προσθέτει στη συνδεδεμένη λίστα.

Εμφανίζει όλα τα περιεχόμενα της λίστας

Ελέγχονται τα δεδομένα ενός-ενός κόμβου με τη σειρά. Μόλις εντοπιστεί η θέση στην οποία πρέπει να παρεμβληθεί ο νέος κόμβος, επιστρέφει ως τιμή τη διεύθυνση του κόμβου μετά από τον οποίο πρέπει να γίνει η παρεμβολή. Επιστρέφει τιμή NULL όταν ο κόμβος πρέπει να παρεμβληθεί στην αρχή, πριν από τον πρώτο κόμβο της λίστας.

Μετάβαση στον επόμενο κόμβο.

Προσθέτει έναν κόμβο με τιμή **ar** στη λίστα, παρεμβάλλοντας τον στη σωστή θέση

Αν ο δείκτης **thesi** είναι NULL ο κόμβος θα παρεμβληθεί στην αρχή της λίστας

Περίπτωση η λίστα να είναι άδεια.

Η λίστα δεν είναι άδεια.

Ο δείκτης **thesi** δεν είναι NULL οπότε ο κόμβος θα παρεμβληθεί στο ενδιάμεσο (η στο τέλος) της λίστας.

```

    {
        temp_next=thesi->next;
        thesi->next=neos;
        neos->previous=thesi;
        neos->next=temp_next;
    }
}

display_all()
{
    struct node *p;
    p=list_head;
    while (p!=NULL)
    {
        printf("%d\n",p->data);
        p=p->next;
    }
}

```

Εμφανίζει όλα τα δεδομένα της ταξινομημένης λίστας.

Εμφάνιση των δεδομένων του κόμβου.

Μετάβαση στον επόμενο κόμβο.



Για την υλοποίηση της ταξινομημένης λίστας χρησιμοποιείται μια διπλά συνδεδεμένη λίστα.

18.3

Επανεξετάστε ότι έχουμε έναν πίνακα `a` με 100 τυχαίους αριθμούς. Να γραφεί κώδικας C στο οποίο να δημιουργείται ένα δένδρο αριστερό-δεξιό με δεδομένα που ελήφθησαν από τον πίνακα `a`.

```

#include <stdio.h>
#include <stdlib.h>
#include <stdlib.h>
#include <malloc.h>

struct node
{
    int data;
    struct node *left;
    struct node *right;
};

struct node *root;
struct node *newnode();
struct node *insert();
void display();

main()
{
    int a[100],i;

    for(i=0;i<100;i++) a[i]=rand();
}

```

Για την υλοποίηση του δυαδικού δένδρου χρησιμοποιούνται οι τεχνικές και ο κώδικας που αναφέρονται στις σελίδες 452-460 του βιβλίου.

Το τμήμα δεδομένων αποτελείται από το πεδίο `data`, στο οποίο καταχωρίζεται ο αριθμός.

Στον πίνακα `a[]` καταχωρούνται 100 τυχαίοι αριθμοί.

```

root=NULL;
for(i=0;i<100;i++) insert(a[i]);
display(root);
}

struct node *newnode(int num)
{
    struct node *new;
    new=malloc(sizeof(struct node));
    if(new==NULL)
    {
        puts("No memory");
        return NULL;
    }
    new->data = num;
    new->left = NULL;
    new->right = NULL;
    return new;
}

struct node *insert(int num)
{
    struct node *next,*current,*ptr;
    int isleft;
    next=current=root;
    ptr=newnode(num);
    if (root == NULL)
    {
        root=ptr;
        return ptr;
    }
    while(1)
    {
        if(num < current->data)
        {
            next = current->left;
            isleft=1;
        }
        else
        {
            next = current->right;
            isleft=0;
        }
        if(next == NULL)
        {
            if(isleft)
                current->left=ptr;
            else

```

Οι αριθμοί του πίνακα `a[]`, προστίθενται στο δυαδικό δένδρο.

Εμφανίζει όλα τα δεδομένα του δυαδικού δένδρου.

Δημιουργεί έναν νέο κόμβο με κλειδί `num`.

Προσθέτει έναν νέο κόμβο με κλειδί `num` στο δυαδικό δένδρο. Επιστρέφει έναν δείκτη στο νέο κόμβο.


```

        current->right=ptr;
        return ptr;
    }
    current=next;
}

```

```

void display(struct node *ptr)
{
    if (ptr == NULL) return;
    display(ptr->left);
    printf("%d ", ptr->data);
    display(ptr->right);
}

```

Εμφανίζει τα δεδομένα όλων των κόμβων του δυαδικού δένδρου (με ρίζα ptr) σε διατεταγμένη σειρά. Χρησιμοποιούνται αναδρομικές κλήσεις της συνάρτησης.

18.4

Υποθέτουμε ότι έχουμε μια απλή συνδεδεμένη λίστα με την επόμενη δομή κόμβων: *

```

struct node
{
    float varos;
    struct node *next;
};
list_head;

```

Ο δείκτης της μεταβ **list_head** δείχνει στη κεφαλή της λίστας. Οι κόμβοι που περιέχει η λίστα είναι τετακτοί κατά έναν δείκτη με το δείκτη του. Να γράψετε κώδικας ο οποίος να υπολογίζει το μέσο όρο του αριθμητικού βαρὸς των κόμβων.

```

float calculate_mo()
{
    struct node *p;
    float sum=0,mo;
    int plithos=0;
    p=list_head;
    while (p!=NULL)
    {
        sum=sum+p->varos;
        p=p->next;
        plithos++;
    }
    mo=sum/plithos;
    return mo;
}

```

Η μεταβλητή **sum** θα χρησιμοποιηθεί για την αποθήκευση του αθροίσματος των βαρών.

Η μεταβλητή **plithos** θα χρησιμοποιηθεί για το "μέτρημα" του πλήθους των κόμβων της λίστας.

Επισκεπτόμαστε έναν-έναν τους κόμβους της λίστας και προσθέτουμε στη **sum** το εκάστοτε βάρος (p->varos). Κάθε φορά η μεταβλητή **plithos** αυξάνει κατά 1.

Η συνάρτηση επιστρέφει σαν τιμή τον μέσο όρο (sum/plithos) των βαρών.

18.5

Υποθέτουμε ότι έχουμε ένα δυαδικό δένδρο με την επόμενη δομή κόμβου, το οποίο περιέχει τις ηλικίες ενός δέντρου. ***

```
struct node
{
    float ilikia;
    struct node *left;
    struct node *right;
};
```

Το πεδίο **ilikia** του κόμβου **node** είναι το πεδίο-κλειδί, ο οποίος δείχνει στο κόμβο προς το δεξιό. Τα γράμματα της συνάρτησης **max_data()** δείχνουν ότι θα πάρουμε το δείκτη **max_node** και να επιστρέψουν τον δείκτη.

Τη μέγιστη ηλικία

```
int max_data(struct node *rt)
```

```
{
    struct node *max_node;
    max_node= find_right_most(rt);
    if(max_node==NULL)
    {
        printf("Το δένδρο είναι άδειο\n");
        exit(1);
    }
    else
        return max_node->ilikia;
}
```

Στο δείκτη **max_node** καταχωρείται η διεύθυνση του τελευταίου δεξιό κόμβου του δ.δ.

Στη περίπτωση που το δ.δ είναι άδειο, τερματίζει με μήνυμα λάθους.

Επιστρέφει σαν τιμή το πεδίο **ilikia** του τελευταίου δεξιό κόμβου.

```
struct node *find_right_most(struct node *rt)
```

```
{
    struct node *current;
    if(rt==NULL) return NULL;
    while(rt->right!=NULL)
    {
        rt=rt->right;
    }
    return rt;
}
```

Εντοπίζει τον τελευταίο δεξιό κόμβο του δυαδικού δένδρου. Ο κόμβος αυτός περιέχει τη μεγαλύτερη τιμή κλειδιού. Επιστρέφει έναν δείκτη σε αυτόν τον κόμβο. Στη περίπτωση που το δ.δ είναι άδειο, επιστρέφει τιμή NULL.



Εφόσον το πεδίο της ηλικίας είναι το πεδίο-κλειδί, ο κόμβος με τη μέγιστη ηλικία είναι ο τελευταίος δεξιό κόμβος του δυαδικού δένδρου. Η συνάρτηση **max_data()** χρησιμοποιεί τη **find_right_most()** για να εντοπίσει τον τελευταίο δεξιό κόμβο (βλέπε "Υλοποίηση της δομής δυαδικού δένδρου" σελίδα 461 του βιβλίου). Επιστρέφει σαν τιμή τη τιμή του πεδίου **ilikia** του κόμβου αυτού. Στη περίπτωση που το Δ.Δ είναι άδειο, τερματίζει με ένα μήνυμα λάθους.

Την ελάχιστη ηλικία

```
int min_data(struct node *rt)
{
    struct node *min_node;
    min_node= find_left_most(rt);
    if(min_node==NULL)
    {
        printf("Το δένδρο είναι άδειο\n");
        exit(1);
    }
    else
        return min_node->ilikia;
}

struct node *find_left_most(struct node *rt)
{
    struct node *current;
    if(rt==NULL) return NULL;
    while (rt->left!=NULL)
    {
        rt=rt->left;
    }
    return rt;
}
```

Στο δείκτη **min_node** καταχωρείται η διεύθυνση του τελευταίου αριστερά κόμβου του δ.δ.

Στη περίπτωση που το δ.δ είναι άδειο, τερματίζει με μήνυμα λάθους.

Επιστρέφει σαν τιμή το πεδίο **ilikia** του τελευταίου αριστερά κόμβου.

Εντοπίζει τον τελευταίο αριστερά κόμβο του δυαδικού δένδρου. Ο κόμβος αυτός περιέχει τη μικρότερη τιμή κλειδιού. Επιστρέφει έναν δείκτη σε αυτόν τον κόμβο. Στη περίπτωση που το δ.δ είναι άδειο, επιστρέφει τιμή NULL.

👉 Εφόσον το πεδίο της ηλικίας είναι το πεδίο-κλειδί, ο κόμβος με την ελάχιστη ηλικία είναι ο τελευταίος αριστερά κόμβος του δυαδικού δένδρου. Η συνάρτηση **min_data()** χρησιμοποιεί τη **find_left_most()** για να εντοπίσει τον τελευταίο αριστερά κόμβο (βλέπε "Υλοποίηση της δομής δυαδικού δένδρου" σελίδα 461 του βιβλίου). Επιστρέφει σαν τιμή τη τιμή του πεδίου **ilikia** του κόμβου αυτού. Στη περίπτωση που το Δ.Δ είναι άδειο, τερματίζει με ένα μήνυμα λάθους.

Το μέσο όρο των ηλικιών

```
float mo(struct node *rt)
{
    if(count(rt)!=0)
        return sum(rt)/count(rt);
    else
    {
        printf("Το δένδρο είναι άδειο\n");
        exit(1);
    }
}

int count(struct node *ptr)
{
    if (ptr == NULL) return 0;
    return 1+count(ptr->left)+count(ptr->right);
}
```

Στη περίπτωση που το δ.δ έχει τουλάχιστον ένα κόμβο, υπολογίζει και επιστρέφει τον μέσο όρο. $\text{Σύνολο_ηλικιών} / \text{Πλήθος_κόμβων}$.

Στη περίπτωση που το δ.δ είναι άδειο, τερματίζει με μήνυμα λάθους.

Η συνάρτηση χρησιμοποιεί αναδρομική διαδικασία για να υπολογίσει το πλήθος των κόμβων του δ.δ.

```

}

float sum(struct node *ptr)
{
    if (ptr == NULL) return 0;
    return ptr->ilikia+sum(ptr->left)+sum(ptr->right);
}
    
```

Η συνάρτηση **sum()** χρησιμοποιεί αναδρομική διαδικασία για να υπολογίσει το συνολικό άθροισμα του πεδίου **ilikia** όλων των κόμβων του δ.δ.

👉 Η συνάρτηση **mo()** καλεί την **sum()** για να υπολογίσει το συνολικό άθροισμα των ηλικιών των κόμβων και την **count()** για να υπολογίσει το πλήθος των κόμβων. Ο μέσος όρος των ηλικιών υπολογίζεται από τον τύπο Συνολικό_άθροισμα/πλήθος.

18.6

Παραδείγουμε ένα πρόγραμμα το οποίο δέχεται ονόματα και τα διατάσσει σε ένα δυαδικό δένδρο στο οποίο γίνεται καταχώριση. Το πρόγραμμα να σταματάει όταν δοθεί ένα κενό όνομα.

```

#include <stdio.h>
#include <stdlib.h>
#include <stdlib.h>
#include <malloc.h>

struct node
{
    char data[30];
    struct node *left;
    struct node *right;
};

struct node *root;
struct node *newnode();
struct node *insert();
void display();

main()
{
    char onoma[30];
    root=NULL;
    while(1)
    {
        printf("Δώσε όνομα:");
        gets(onoma);
        if(strcmp(onoma,"")==0) break;
        insert(onoma);
    }
    display(root);
}
    
```

Για την υλοποίηση του δυαδικού δένδρου χρησιμοποιούνται οι τεχνικές και ο κώδικας που αναφέρονται στις σελίδες 452~460 του βιβλίου.

Το τμήμα δεδομένων αποτελείται από το πεδίο **data**, στο οποίο καταχωρίζεται το όνομα.

Ζητάει να πληκτρολογηθεί ένα όνομα και το καταχωρεί στον πίνακα **onoma[]**. Η επαναληπτική διαδικασία σταματάει όταν δοθεί κενό.

Το όνομα, προστίθεται στο δυαδικό δένδρο.

Εμφανίζει όλα τα δεδομένα του δυαδικού δένδρου.

struct node *newnode(char lex[]) Δημιουργεί έναν νέο κόμβο με κλειδί **lex**.

```

{
    struct node *new;
    new=(struct node *)malloc(sizeof(struct node));
    if(new==NULL)
    {
        puts("No memory");
        return NULL;
    }
    strcpy(new->data,lex);
    new->left = NULL;
    new->right = NULL;
    return(new);
}

```

```

struct node *insert(char lex[])
{

```

Προσθέτει έναν νέο κόμβο με κλειδί **lex** στο δυαδικό δένδρο. Επιστρέφει έναν δείκτη στο νέο κόμβο.

```

{
    struct node *next,*current,*ptr;
    int isleft;
    next=current=root;
    ptr=newnode(lex);
    if (root == NULL)
    {
        root=ptr;
        return ptr;
    }
    while(1)
    {
        if(strcmp(lex,current->data)==-1)
        {
            next = current->left;
            isleft=1;
        }
        else
        {
            next = current->right;
            isleft=0;
        }
        if(next == NULL)
        {
            if(isleft)
                current->left=ptr;
            else
                current->right=ptr;
            return ptr;
        }
        current=next;
    }
}

```

```
void display(struct node *ptr)
{
    if (ptr == NULL) return;
    display(ptr->left);
    printf("%s\n", ptr->data);
    display(ptr->right);
}
```

Εμφανίζει τα δεδομένα όλων των κόμβων του δυαδικού δένδρου (με ρίζα ptr) σε διατεταγμένη σειρά. Χρησιμοποιούνται αναδρομικές κλήσεις της συνάρτησης.

18.7

Υποθέτουμε ότι έχουμε μια αλυσίδα συνδεδεμένη λίστα με την επόμενη δομή κόμβου:

```
struct tnode
{
    char onoma[50];
    struct tnode *next;
};
list_head
```

Ο χαρακτήρας **list_head** δείχνει στην κεφαλή της λίστας. Η λίστα περιέχει 5 κόμβους. Μπορούμε να πάρουμε οποιονδήποτε κόμβο από την λίστα σε ένα αρχείο κειμένου που ονομάζεται ONOMATA.

```
void list_to_file()
{
    struct node *p;
    FILE *fp;
    if ((fp=fopen("ONOMATA","w")) == NULL)
    {
        printf("Προβλημα στο άνοιγμα του αρχείου");
        return;
    }
    p=list_head;
    while (p!=NULL)
    {
        fputs(p->onoma, fp);
        p=p->next;
    }
    fclose(fp);
}
```

Ανοίγει το αρχείο **ONOMATA** για εγγραφή. Ελέγχει αν η διαδικασία ανοίγματος ήταν επιτυχής.

Επισκέπτεται έναν-έναν τους κόμβους της λίστας και καταχωρεί στο αρχείο, το πεδίο του ονόματος του κάθε κόμβου.

18.8

Two important considerations about the use of the *in vitro* model are that the model was developed for the study of the effects of chemical agents on the growth of *S. aureus* and that the model is not intended to be used for the study of the effects of chemical agents on the growth of other bacterial species.

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

```
struct node
{
    char data[30];
    struct node *next;
}*list head,*neos;
```

```
struct node *find_place(char lex[]);  
void add_node_to_list(char lex[]);
```

```
main()
```

```
{
    list_head=NULL;
    read_all();
    display_all();
}
```

Το τμήμα δεδομένων αποτελείται από το πεδίο **data**, στο οποίο καταχωρίζεται ο αριθμός.
Για την υλοποίηση της ταξινομημένης λίστας, χρησιμοποιείται μια διπλά συνδεδεμένη λίστα (βλέπε σελίδα 441 του βιβλίου).

Διαβάζει από το αρχείο τα ονόματα.

Εμφανίζει όλα τα περιεχόμενα της λίστας

```
struct node *find place(char lex[])
```

```

{
    struct node *p,*tp;
    p=list_head;
    tp=NULL;
    while (p!=NULL)
    {
        if(strcmp(lex,p->data)==1) tp=p;
        p=p->next;
    }
    return tp;
}

```

Ελέγχονται
Μόλις εντο-
βληθεί ο
του κόμβου
βολή.
Επιστρέφ-
βληθεί στη

Ελέγχονται τα δεδομένα ενός-ενός κόμβου με τη σειρά. Μόλις εντοπιστεί η θέση στην οποία πρέπει να παρεμβληθεί ο νέος κόμβος, επιστρέφει ως τιμή τη διεύθυνση του κόμβου πριν από τον οποίο πρέπει να γίνει η παρεμβολή.

Επιστρέφει τιμή NULL όταν ο κόμβος πρέπει να παρεμβληθεί στην αρχή, πριν από τον πρώτο κόμβο της λίστας.

Μετάβαση στον επόμενο κόμβο.

Προσθέτει τον κόμβο με τιμή `lex` στη λίστα, παρεμβάλλοντας τον στη σωστή θέση

```
void add node to list(char lex[])
```

```

{
    struct node *temp_next,*thesi;
    thesi=find_place(lex);
    neos = (struct node *)malloc(sizeof(struct node));
    strcpy(neos->data,lex);
    if(thesi==NULL)
    {
        if(list_head!=NULL)
        {

```

Αν ο δείκτης **thesi** είναι NULL ο κόμβος θα παρεμβληθεί στην αρχή της λίστας

Περίπτωση η) λίστα να είναι άδεια.

```

        neos->next=list_head;
        list_head=neos;
    }
    else
    {
        neos->next=NULL;
        list_head=neos;
    }
}
else
{
    temp_next=thesi->next;
    thesi->next=neos;
    neos->next=temp_next;
}
}

display_all()
{
    struct node *p;
    p=list_head;
    while (p!=NULL)
    {
        printf("%s\n",p->data);
        p=p->next;
    }
}

read_all()
{
    FILE *fp;
    char lex[30];
    fp=fopen("onomata.txt","r");
    while (!feof(fp))
    {
        fscanf(fp,"%s",lex);
        add_node_to_list(lex);
    }
}

```

Η λίστα δεν είναι άδεια.


Ο δείκτης **thesi** δεν είναι NULL οπότε ο κόμβος θα παρεμβληθεί στο ενδιάμεσο (η στο τέλος) της λίστας.

Εμφανίζει όλα τα δεδομένα της ταξινομημένης λίστας.

Εμφάνιση των δεδομένων του κόμβου.

Μετάβαση στον επόμενο κόμβο.

Προσθέτει έναν κόμβο με τιμή **lex[]** στη λίστα, παρεμβάλλοντάς τον στη σωστή θέση

 Για την υλοποίηση της ταξινομημένης λίστας χρησιμοποιήθηκε μια απλά συνδεδεμένη λίστα.

18.9 Ποια από τα επόμενα αληθεύουν: ★

- ☐ Οι λίστες και τα δυαδικά δένδρα δεσμεύουν συγκεκριμένο μέγεθος μνήμης.
- ☒ Σε μια δομή ουράς, το πρώτο στοιχείο που προστίθεται στην ουρά είναι το πρώτο που φεύγει από την ουρά.
- ☐ Σε μια δομή στοίβας, το πρώτο στοιχείο που προστίθεται στη στοίβα είναι το πρώτο που φεύγει από τη στοίβα.
- ☒ Σε μια απλά συνδεδεμένη λίστα δεν μπορούμε να εμφανίσουμε τα στοιχεία της λίστας με τη σειρά από το τελευταίο προς το πρώτο.
- ☒ Ένα δυαδικό δένδρο διατηρεί τα δεδομένα διατεταγμένα ως προς το πεδίο-κλειδί του δυαδικού δένδρου.

Ασκήσεις Κεφαλαίου 19

19.1 Εξοργιστείτε τη λειτουργία της if, αφού έχετε προγράψει τους * *

```

#include <stdio.h>

int
main()
{
    int a, b;
    printf("a: ");
    scanf("%d", &a);
    printf("b: ");
    scanf("%d", &b);
    if (a > b)
        printf("a > b\n");
    else
        printf("a <= b\n");
}

```

☞ Η **cin** περιμένει να πληκτρολογηθούν δύο αριθμοί από το πληκτρολόγιο και τους καταχωρίζει στις μεταβλητές **a**, και **b** αντίστοιχα.

☞ Η **if** ελέγχει αν η τιμή της μεταβλητής **a** είναι μεγαλύτερη από την τιμή της μεταβλητής **b**. Αν είναι εμφανίζει την τιμή της **a** διαφορετικά την τιμή της **b**. Σε κάθε περίπτωση δηλαδή εμφανίζει τον μεγαλύτερο από τους δύο αριθμούς που δώσαμε.

19.2 Ποια από τα παρακάτω αληθεύουν: ★

- ☒ Η C++ είναι μια επέκταση της C, προσθέτοντας αντικειμενοστρεφή χαρακτηριστικά.
- ☒ Όλα τα αντικείμενα μιας κλάσης έχουν τα ίδια χαρακτηριστικά και λειτουργίες.
- ☒ Η κληρονομικότητα είναι ένα χαρακτηριστικό του πολυμορφισμού.
- ☐ Με τη C++ μπορούμε να φτιάχνουμε μόνο αντικειμενοστρεφή προγράμματα.
- ☒ Ένα πρόγραμμα σε C μπορεί να μεταγλωττιστεί από οποιονδήποτε μεταγλωττιστή της C++, χωρίς ή με ελάχιστες αλλαγές.
- ☒ Στη C++ χρησιμοποιείται το αντικείμενο **cout** για την έξοδο πληροφοριών στην οθόνη.
- ☐ Στη C++ η μεταβίβαση παραμέτρων γίνεται όπως και στη C: μόνο με τιμή.
- ☒ Στη C++ μπορούμε να χρησιμοποιήσουμε δηλωτικές προτάσεις οπουδήποτε στον κώδικα ακόμα και μετά από εκτελέσιμες.
- ☐ Στη C++ δεν μπορούμε να χρησιμοποιήσουμε τις συναρτήσεις **printf()** και **scanf()**.
- ☒ Η C++ διαθέτει λογικό τύπο δεδομένων.