Προβλήματα στη Γλώσσα Προγραμματισμού C

Παναγιώτης Κόκκας

Αναπληρωτής Καθηγητής , Τμήμα Φυσικής

Πανεπιστήμιο Ιωαννίνων

Ιωάννινα 2009

Εισαγωγή:

Οι σημειώσεις αυτές απευθύνονται στους πρωτοετείς φοιτητές του Τμήματος Φυσικής του Πανεπιστημίου Ιωαννίνων στα πλαίσια του μαθήματος "Γλώσσες προγραμματισμού Η/Υ". Παρέχουν μια πληθώρα από προβλήματα λυμένα και άλυτα στην γλώσσα προγραμματισμού C. Πρέπει να συνδυαστούν με τις σημειώσεις που περιλαμβάνουν *"Εισαγωγή στη γλώσσα προγραμματισμού C*". Παρέχουν ένα σημαντικό βοήθημα κυρίως για το εργαστηριακό μέρος του μαθήματος. Σε καμιά περίπτωση δεν αντικαθιστούν το διδακτικό σύγγραμμα ούτε αντικαθιστούν τις παραδόσεις του μαθήματος.

Αναλυτικότερα το Κεφάλαιο Ι περιλαμβάνει εργαστηριακές ασκήσεις που αφορούν βασικές έννοιες και συναρτήσεις στη γλώσσα προγραμματισμού C. Ποιο συγκεκριμένα παρουσιάζονται ασκήσεις οι οποίες αναφέρονται σε τύπους δεδομένων, στις συναρτήσεις printf(), scanf(), getchar() και putchar(), σε μαθηματικές συναρτήσεις, σε απλά προβλήματα φυσικής καθώς και σε προβλήματα τα οποία αφορούν ανάλυση πειραματικών δεδομένων.

Το Κεφάλαιο ΙΙ περιλαμβάνει εργαστηριακές ασκήσεις που αφορούν τις εντολές της C οι οποίες ελέγχουν την ροή εκτέλεσης σε ένα πρόγραμμα όπως οι εντολές if-else, switch, while, for και do-while. Επί πλέον περιλαμβάνει εργαστηριακές ασκήσεις οι οποίες αναφέρονται σε διερεύνηση εξισώσεων, αναδρομικές σχέσεις, αριθμητικές σειρές, αναπτύγματα και υπολογισμούς ορισμένων ολοκληρωμάτων.

Το Κεφάλαιο ΙΙΙ περιλαμβάνει εργαστηριακές ασκήσεις που αφορούν τη χρήση δεικτών και πινάκων. Περιλαμβάνονται επίσης ασκήσεις οι οποίες αναφέρονται στη χρήση τυχαίων αριθμών.

Το Κεφάλαιο ΙV περιλαμβάνει εργαστηριακές ασκήσεις που αφορούν τη χρήση δομών και ενώσεων. Περιλαμβάνονται επίσης ασκήσεις οι οποίες αναφέρονται στη χρήση απλών βάσεων δεδομένων.

Το Κεφάλαιο V περιλαμβάνει εργαστηριακές ασκήσεις οι οποίες αφορούν σε προσπέλαση αρχείων. Τέλος στο κεφάλαιο VI περιλαμβάνονται γενικές επαναληπτικές ασκήσεις.

Η ύλη είναι οργανωμένη με τέτοιο τρόπο ώστε να καλύπτονται πλήρως οι απαιτήσεις ενός εξαμηνιαίου μαθήματος.

Κεφάλαιο Ι

Προβλήματα που αφορούν βασικές έννοιες και συναρτήσεις στη γλώσσα προγραμματισμού C.

Στο παρόν κεφάλαιο παρουσιάζονται προβλήματα τα οποία αφορούν βασικές έννοιες και συναρτήσεις στη γλώσσα προγραμματισμού C. Παρουσιάζονται ασκήσεις οι οποίες αναφέρονται σε:

- **Τύπους δεδομένων**
- **Στη συνάρτηση printf()**
- **Στη συνάρτηση scanf()**
- **Στις συναρτήσεις getchar() και putchar()**
- **Στις μαθηματικές συναρτήσεις που εμπεριέχονται στο math.h**
- **Υπολογισμούς σύνθετων μαθηματικών συναρτήσεων**
- **Απλά προβλήματα φυσικής**
- **Απλά προβλήματα που αφορούν ανάλυση πειραματικών δεδομένων**

1.1 Λυμένα Προβλήματα.

1.1.1 Γράψτε ένα πρόγραμμα το οποίο να εμφανίζει τις παρακάτω γραμμές:

C is a general-purpose programming language.
It has been closely associated with the UNIX system.
The Unix system has been developed using the C language.

Μια πιθανή λύση του προβλήματος είναι η ακόλουθη:

```
#include<stdio.h>

int main(void)
{
  printf("C is a general-purpose programming language.\n");
  printf("It has been closely assosiated with the UNIX system.\n");
  printf("The Unix system has been developed using the C language.\n");
  return 0;
}
-:--  example_2_1_1.c      (C Abbrev)--L9--All--------------------------------
```

Η εκτέλεση και τα αποτελέσματα του παραπάνω προγράμματος εικονίζονται στο παρακάτω σχήμα:

```
[student1@pc244 kef2]$
[student1@pc244 kef2]$ gcc example_2_1_1.c
[student1@pc244 kef2]$ a.out
C is a general-purpose programming language.
It has been closely assosiated with the UNIX system.
The Unix system has been developed using the C language.
[student1@pc244 kef2]$
```

Στην παραπάνω λύση χρησιμοποιήθηκε η συνάρτηση printf() τρεις φορές, μία για κάθε γραμμή. Θα μπορούσε να χρησιμοποιηθεί μόνο μια φορά και δώσει το ίδιο αποτέλεσμα. Αφήνεται στον φοιτητή να υλοποιήσει αυτή τη λύση.

1.1.2 Γράψτε ένα πρόγραμμα το οποίο να εμφανίζει τους παρακάτω αριθμούς στην μορφή που δίνονται:

```
    1    1
   12    12
  123    123
 1234    1234
12345    12345
```

Μια πιθανή λύση του προβλήματος είναι η ακόλουθη:

```
#include<stdio.h>

int main(void)
{
  int a,b,c,d,e;

  a=1;
  b=12;
  c=123;
  d=1234;
  e=12345;

  printf("%8d %-8d\n",a,a);
  printf("%8d %-8d\n",b,b);
  printf("%8d %-8d\n",c,c);
  printf("%8d %-8d\n",d,d);
  printf("%8d %-8d\n",e,e);

  return 0;
}
-:--  example_2_1_2.c      (C Abbrev)--L20--All----------------
```

Η εκτέλεση και τα αποτελέσματα του παραπάνω προγράμματος εικονίζονται στο παρακάτω σχήμα:

```
[student1@pc244 kef2]$
[student1@pc244 kef2]$ gcc example_2_1_2.c
[student1@pc244 kef2]$ a.out
       1 1
      12 12
     123 123
    1234 1234
   12345 12345
[student1@pc244 kef2]$
```

1.1.3 Γράψτε ένα πρόγραμμα το οποίο να υπολογίζει και να εκτυπώνει το συνημίτονο, το ημίτονο και την εφαπτομένη των 60˚.

Μια πιθανή λύση είναι η ακόλουθη:

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


int main(void)
{
  double x;

  x=60.0;                /* 60 degrees */
  x=x*3.14159/180.0;     /*60 degrees to rads */

  printf("The cosine of 60 degrees is : %f.\n",cos(x));
  printf("The sine of 60 degrees is   : %f.\n",sin(x));
  printf("The tangent of 60 degrees is: %f.\n",tan(x));

  return 0;
}
-:**  example_math.c      (C Abbrev)--L17--All----------------
```

Προσοχή! κατά την μεταγλώττιση πρέπει να γίνει σύνδεση με την βιβλιοθήκη μαθηματικών συναρτήσεων μέσω του **–lm**.
Η εκτέλεση του παραπάνω προγράμματος είναι η ακόλουθη:

```
[student1@pc244 kef2]$ gcc example_math.c -lm
[student1@pc244 kef2]$ a.out
The cosine of 60 degrees is : 0.500001.
The sine of 60 degrees is   : 0.866025.
The tangent of 60 degrees is: 1.732047.
[student1@pc244 kef2]$
```

1.1.4 Ένα σώμα αφήνεται να πέσει ελεύθερα από ηρεμία. Συντάξτε ένα πρόγραμμα το οποίο να υπολογίζει την απόσταση που διανύει το σώμα και την ταχύτητά του μετά από 1 sec και μετά από 5 sec. Διατηρείστε ακρίβεια 2 δεκαδικών ψηφίων στους υπολογισμούς σας. (Δίνονται $s=1/2*(g*t^2)$, $u=g*t$, $g=9.81\ m/sec^2$)

Μια πιθανή λύση είναι η ακόλουθη:

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

int main(void)
{
  double g,t,s,u;
  g=9.81;       /* epitaxinsi barititas */

  t=1.0;               /* xronos 1 sec */
  s=(g*pow(t,2.))/2.;  /* diastima */
  u=g*t;               /* taxitita */
  printf("H apostasi meta apo 1 sec einai: %8.2f m\n",s);
  printf("H taxitita meta apo 1 sec einai: %8.2f m/sec\n",u);

  t=5.0;               /* xronos 5 sec */
  s=(g*pow(t,2.))/2.;  /* diastima */
  u=g*t;               /* taxitita */
  printf("H apostasi meta apo 5 sec einai: %8.2f m\n",s);
  printf("H taxitita meta apo 5 sec einai: %8.2f m/sec\n",u);

  return 0;
}
--:--  example_2_1_5.c      (C Abbrev)--L23--All--------------------
```

Η εκτέλεση του παραπάνω προγράμματος παράγει το ακόλουθο αποτέλεσμα:

```
[student1@pc244 kef2]$
[student1@pc244 kef2]$ gcc example_2_1_5.c -lm
[student1@pc244 kef2]$ a.out
H apostasi meta apo 1 sec einai:     4.91 m
H taxitita meta apo 1 sec einai:     9.81 m/sec
H apostasi meta apo 5 sec einai:   122.63 m
H taxitita meta apo 5 sec einai:    49.05 m/sec
[student1@pc244 kef2]$
```

Προσοχή κατά την μεταγλώττιση πρέπει να γίνει σύνδεση με την βιβλιοθήκη μαθηματικών συναρτήσεων μέσω του –lm.

1.1.5 Γράψτε ένα πρόγραμμα στο οποίο να εισάγονται από το πληκτρολόγιο δύο αριθμοί τύπου int και ένας αριθμός τύπου float. Στη συνέχεια να εκτυπώσετε τους αριθμούς. Χρησιμοποιείστε τις συναρτήσεις scanf() και printf().

Μια πιθανή λύση είναι η ακόλουθη:

```
#include<stdio.h>

int main(void)
{
  int a,b;
  float c;

  printf("Please insert two integers:\n");
  scanf("%d %d",&a, &b);

  printf("Please insert one float number:\n");
  scanf("%f",&c);

  printf("The integers you have typed are: %d %d\n",a,b);
  printf("The float number you have typed is: %f\n",c);

  return 0;

}
-:**  example_scanf.c      (C Abbrev)--L19--All--------------
```

Η εκτέλεση του παραπάνω προγράμματος είναι η ακόλουθη:

```
[student1@pc244 kef2]$ gcc example_scanf.c
[student1@pc244 kef2]$ a.out
Please insert two integers:
12345 98765
Please insert one float number:
98.765
The integers you have typed are: 12345 98765
The float number you have typed is: 98.764999
[student1@pc244 kef2]$
```

1.1.6 Να γράψετε ένα πρόγραμμα το οποίο να υπολογίζει την αριθμητική τιμή της συνάρτησης $f(x)=\sqrt{\frac{4x^3-3x^2+2x-1}{5}}$ για διάφορες τιμές του x. Οι τιμές του x να εισάγονται στο πρόγραμμα με την χρήση της συνάρτησης scanf(). Να διατηρήσετε ακρίβεια τριών δεκαδικών ψηφίων στους υπολογισμούς σας. Εκτελέστε το πρόγραμμα δίνοντας τέσσερα δικά σας παραδείγματα..

Μια πιθανή λύση είναι η ακόλουθη:

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

int main(void)
{
  double x,result;
  printf("Dose thn timi tou x:\n");
  scanf("%lf",&x);

  result=sqrt((4.*pow(x,3)-3.*pow(x,2)+2.*x-1.)/(5.));

  printf("H timi tis parastasis gia x=%8.3f einai: %8.3f \n",x,result);
  return 0;
}
-:--  example_2_1_7.c      (C Abbrev)--L15--All----------------------------
```

Η εκτέλεση του παραπάνω προγράμματος με τέσσερα τυχαία παραδείγματα είναι η ακόλουθη:

```
[student1@pc244 kef2]$ gcc example_2_1_7.c -lm
[student1@pc244 kef2]$ a.out
Dose thn timi tou x:
7.23127
H timi tis parastasis gia x=   7.231 einai:   16.548
[student1@pc244 kef2]$
[student1@pc244 kef2]$ a.out
Dose thn timi tou x:
16.74258
H timi tis parastasis gia x=  16.743 einai:   59.940
[student1@pc244 kef2]$
[student1@pc244 kef2]$ a.out
Dose thn timi tou x:
24.78291
H timi tis parastasis gia x=  24.783 einai:  108.712
[student1@pc244 kef2]$
[student1@pc244 kef2]$ a.out
Dose thn timi tou x:
-11.2345
H timi tis parastasis gia x= -11.234 einai:      nan
[student1@pc244 kef2]$
```

Παρατηρείστε πως για την τιμή x=-11.2345 η τιμή της παράστασης δεν ορίζεται γιατί το υπόριζο είναι αρνητικό. Γιαυτό το λόγο εκτυπώνεται το nan.

1.1.7 Γράψτε ένα πρόγραμμα το οποίο να επαληθεύει την ισότητα:

sin(A+B) = sinA cosB + cosA sinB

Να υπολογίσετε ξεχωριστά το κάθε μέλος της ισότητας και να το εκτυπώσετε για διάφορες τιμές των μεταβλητών A και B. Δώστε μερικά παραδείγματα. Να διατηρήσετε ακρίβεια τεσσάρων δεκαδικών ψηφίων στους υπολογισμούς σας.

Μια πιθανή λύση είναι η ακόλουθη:

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

int main(void)
{
  float A, B, proto_melos, deftero_melos;
  printf("Doste tis times gia tis metablhtes A kai B se deg:\n");
  scanf("%f %f",&A,&B);

  A=A*3.14159/180.;  /* metatropi deg se rad */
  B=B*3.14159/180.;  /* metatropi deg se rad */

  proto_melos=sin(A+B);
  deftero_melos=sin(A)*cos(B)+cos(A)*sin(B);
  printf("To proto melos tis isothtas einai  : %8.4f\n",proto_melos);
  printf("To deutero melos tis isothtas einai: %8.4f\n",deftero_melos);

  return 0;
}
-:--  example_2_1_9.c      (C Abbrev)--L19--All----------------------------
```

Η εκτέλεση του παραπάνω προγράμματος με τρία τυχαία παραδείγματα είναι η ακόλουθη:

```
[student1@pc244 kef2]$ gcc example_2_1_9.c -lm
[student1@pc244 kef2]$ a.out
Doste tis times gia tis metablhtes A kai B se deg:
34. 59.
To proto melos tis isothtas einai  :   0.9986
To deutero melos tis isothtas einai:   0.9986
[student1@pc244 kef2]$
[student1@pc244 kef2]$ a.out
Doste tis times gia tis metablhtes A kai B se deg:
11.2 23.7
To proto melos tis isothtas einai  :   0.5721
To deutero melos tis isothtas einai:   0.5721
[student1@pc244 kef2]$
[student1@pc244 kef2]$ a.out
Doste tis times gia tis metablhtes A kai B se deg:
135.2 167.3
To proto melos tis isothtas einai  :  -0.8434
To deutero melos tis isothtas einai:  -0.8434
[student1@pc244 kef2]$
```

1.1.8 Να γράψετε ένα πρόγραμμα το οποίο να υπολογίζει την αριθμητική τιμή της συνάρτησης $f(x) = \sqrt{\ln(x)^2 + 4}$ για διάφορες τιμές του x. Να αναπτύξετε το πρόγραμμα γράφοντας πρώτα μία συνάρτηση στη C η οποία να υπολογίζει γενικά την τιμή της $f(x)$ και στη συνέχεια να την χρησιμοποιήσετε στη συνάρτηση main(). Οι τιμές του x να εισάγονται στο πρόγραμμα με την χρήση της συνάρτησης scanf(). Να διατηρήσετε ακρίβεια δύο δεκαδικών ψηφίων στους υπολογισμούς σας. Εκτελέστε το πρόγραμμα δίνοντας τέσσερα δικά σας παραδείγματα.

Μια πιθανή λύση είναι η ακόλουθη:

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

float fx(float x);  /* Dilosi prorotipo sinartisis */

int main(void)
{
  float x;
  printf("Dose thn timi tou x:\n");
  scanf("%f",&x);

  printf("H timi tis parastasis gia x=%8.2f einai: %8.2f \n",x,fx(x));
  return 0;
}

float fx(float x)
{
  float result;
  result=sqrt(pow(log(x),2)+4.);
  return result;
}
```

```
-:**  example_2_1_8.c      (C Abbrev)--L21--All----------------------------
```

Η εκτέλεση του παραπάνω προγράμματος με τρία τυχαία παραδείγματα είναι η ακόλουθη:

```
[student1@pc244 kef2]$ gcc example_2_1_8.c -lm
[student1@pc244 kef2]$ a.out
Dose thn timi tou x:
12.34567
H timi tis parastasis gia x=   12.35 einai:     3.21
[student1@pc244 kef2]$
[student1@pc244 kef2]$ a.out
Dose thn timi tou x:
532.6789
H timi tis parastasis gia x=  532.68 einai:     6.59
[student1@pc244 kef2]$
[student1@pc244 kef2]$ a.out
Dose thn timi tou x:
-67.2
H timi tis parastasis gia x=  -67.20 einai:      nan
[student1@pc244 kef2]$
```

Παρατηρείστε πως για την τιμή x=-67.2 η τιμή της παράστασης δεν ορίζεται γιατί ο λογάριθμος δεν ορίζεται για αρνητικούς αριθμούς. Γιαυτό το λόγο εκτυπώνεται το nan.

1.1.9 Γράψτε ένα πρόγραμμα με το οποίο να εισάγετε την λέξη “Hi” να την αναλύσετε στους δύο χαρακτήρες της, να τους τυπώσετε και να τους ανασυνθέσετε ώστε να τυπώσετε ξανά την λέξη “Hi”. Χρησιμοποιήσετε τις συναρτήσεις getchar() και putchar().

Μια πιθανή λύση είναι η ακόλουθη:

```
#include<stdio.h>

int main(void)
{
  int a1,a2;

  printf("Parakalo eisagete dyo xaraktires mazi.\n");

  a1=getchar();
  a2=getchar();
  printf("O protos xaraktiras pou eisagate einai o   :");
  putchar(a1);
  putchar(10);          /* O arithmos 10 antistoixei ston xaraktira "\n" */
  printf("O defteros xaraktiras pou eisagate einai o :");
  putchar(a2);
  putchar(10);          /* O arithmos 10 antistoixei ston xaraktira "\n" */
  printf("Kai oi dio xaraktires pou eisagate mas dinoun tin lexi :");
  putchar(a1);
  putchar(a2);
  putchar(10);          /* O arithmos 10 antistoixei ston xaraktira "\n" */

  return 0;
}
-:--  example_getchar_putchar.c      (C Abbrev)--L23--All----------------------
```

Η εκτέλεση του παραπάνω προγράμματος είναι η ακόλουθη:

```
[student1@pc244 kef2]$ gcc example_getchar_putchar.c
[student1@pc244 kef2]$ a.out
Parakalo eisagete dyo xaraktires mazi.
Hi
O protos xaraktiras pou eisagate einai o   :H
O defteros xaraktiras pou eisagate einai o :i
Kai oi dio xaraktires pou eisagate mas dinoun tin lexi :Hi
[student1@pc244 kef2]$
```

1.1.10 Γράψτε ένα πρόγραμμα με το οποίο να εισάγετε μια λέξη με πέντε χαρακτήρες. Στη συνέχεια να την αναλύσετε στους χαρακτήρες της και να τους τυπώσετε σε πίνακα με τους αντίστοιχους αριθμούς ASCII. Να χρησιμοποιήσετε τη συνάρτηση getchar().

Μια πιθανή λύση είναι η ακόλουθη:

```
#include<stdio.h>

int main(void)
{
  int a1,a2,a3,a4,a5;

  printf("Parakalo eisagete mia lexi me pente xaraktires.\n");

  a1=getchar();
  a2=getchar();
  a3=getchar();
  a4=getchar();
  a5=getchar();

  printf("Xaraktiras      ASCII\n");
  printf("   %c              %3d\n",a1,a1);
  printf("   %c              %3d\n",a2,a2);
  printf("   %c              %3d\n",a3,a3);
  printf("   %c              %3d\n",a4,a4);
  printf("   %c              %3d\n",a5,a5);

  return 0;
}
-:--  example_word.c      (C Abbrev)--L23--All--------------------
```

Η εκτέλεση του προγράμματος με ένα παράδειγμα είναι η ακόλουθη:

```
[student1@pc244 kef2]$ gcc example_word.c
[student1@pc244 kef2]$ a.out
Parakalo eisagete mia lexi me pente xaraktires.
Large
Xaraktiras      ASCII
   L              76
   a              97
   r             114
   g             103
   e             101
[student1@pc244 kef2]$
```

1.1.11 Να συντάξετε ένα πρόγραμμα το οποίο να τυπώνει τον πραγματικό αριθμό a=123.456789 χρησιμοποιώντας την εντολή **printf()** με τους χαρακτήρες μετατροπής %f, %12f, %12.4f, %12.2f, %e, %12.3e, %E, %12.4E.

Μια πιθανή λύση είναι η ακόλουθη:

```
#include<stdio.h>

int main(void)
{
  float a;

  a=123.456789;

  printf("Xrisimopoiontas to %%f o a typonete os     : %f\n",a);
  printf("Xrisimopoiontas to %%12f o a typonete os   : %12f\n",a);
  printf("Xrisimopoiontas to %%12.4f o a typonete os : %12.4f\n",a);
  printf("Xrisimopoiontas to %%12.2f o a typonete os : %12.2f\n",a);
  printf("Xrisimopoiontas to %%e o a typonete os     : %e\n",a);
  printf("Xrisimopoiontas to %%12.3e o a typonete os : %12.3e\n",a);
  printf("Xrisimopoiontas to %%E o a typonete os     : %E\n",a);
  printf("Xrisimopoiontas to %%12.4E o a typonete os : %12.4E\n",a);

  return 0;
}
--:--  example_2_1_4.c      (C Abbrev)--L19--All-------------------------
```

Η εκτέλεση του παραπάνω προγράμματος δίνει το αποτέλεσμα:

```
[student1@pc244 kef2]$ gcc example_2_1_4.c
[student1@pc244 kef2]$ a.out
Xrisimopoiontas to %f o a typonete os     : 123.456787
Xrisimopoiontas to %12f o a typonete os   :   123.456787
Xrisimopoiontas to %12.4f o a typonete os :     123.4568
Xrisimopoiontas to %12.2f o a typonete os :       123.46
Xrisimopoiontas to %e o a typonete os     : 1.234568e+02
Xrisimopoiontas to %12.3e o a typonete os :    1.235e+02
Xrisimopoiontas to %E o a typonete os     : 1.234568E+02
Xrisimopoiontas to %12.4E o a typonete os :   1.2346E+02
[student1@pc244 kef2]$
```

Εάν δηλώσουμε τον a ως double στο πρόγραμμα έχουμε το ακόλουθο αποτέλεσμα.

```
[student1@pc244 kef2]$ a.out
Xrisimopoiontas to %f o a typonete os     : 123.456789
Xrisimopoiontas to %12f o a typonete os   :   123.456789
Xrisimopoiontas to %12.4f o a typonete os :     123.4568
Xrisimopoiontas to %12.2f o a typonete os :       123.46
Xrisimopoiontas to %e o a typonete os     : 1.234568e+02
Xrisimopoiontas to %12.3e o a typonete os :    1.235e+02
Xrisimopoiontas to %E o a typonete os     : 1.234568E+02
Xrisimopoiontas to %12.4E o a typonete os :   1.2346E+02
[student1@pc244 kef2]$
```

Προσέξτε τις διαφορές στα τελευταία ψηφία των δύο πρώτων γραμμών στις δύο εκτυπώσεις.

1.1.12 Να γράψετε ένα πρόγραμμα το οποίο να τυπώνει σε τρεις στήλες τους ακεραίους 0 έως 15 σε **δεκαεξαδική** μορφή, ξεχωριστά πεζά και κεφαλαία, και σε **δεκαδική** μορφή.

Μια πιθανή λύση είναι η ακόλουθη:

```
#include<stdio.h>

int main(void)
{
  printf("hexadecimal    Hexadecimal     Decimal\n");
  printf("   %x              %X              %d\n",0,0,0);
  printf("   %x              %X              %d\n",1,1,1);
  printf("   %x              %X              %d\n",2,2,2);
  printf("   %x              %X              %d\n",3,3,3);
  printf("   %x              %X              %d\n",4,4,4);
  printf("   %x              %X              %d\n",5,5,5);
  printf("   %x              %X              %d\n",6,6,6);
  printf("   %x              %X              %d\n",7,7,7);
  printf("   %x              %X              %d\n",8,8,8);
  printf("   %x              %X              %d\n",9,9,9);
  printf("   %x              %X              %d\n",10,10,10);
  printf("   %x              %X              %d\n",11,11,11);
  printf("   %x              %X              %d\n",12,12,12);
  printf("   %x              %X              %d\n",13,13,13);
  printf("   %x              %X              %d\n",14,14,14);
  printf("   %x              %X              %d\n",15,15,15);
  return 0;
}
--:--  example_2_1_3.c      (C Abbrev)--L23--All-----------------
```

Η εκτέλεση του παραπάνω προγράμματος παράγει το ακόλουθο αποτέλεσμα:

```
[student1@pc244 kef2]$ gcc example_2_1_3.c
[student1@pc244 kef2]$ a.out
hexadecimal    Hexadecimal     Decimal
   0              0              0
   1              1              1
   2              2              2
   3              3              3
   4              4              4
   5              5              5
   6              6              6
   7              7              7
   8              8              8
   9              9              9
   a              A              10
   b              B              11
   c              C              12
   d              D              13
   e              E              14
   f              F              15
[student1@pc244 kef2]$
```

1.1.13 Να γράψετε ένα πρόγραμμα το οποίο να τυπώνει σε πίνακα τους χαρακτήρες 1,2....9,0 και τους αντίστοιχους κωδικούς ASCII.

Μια πιθανή λύση είναι η ακόλουθη:

```
#include<stdio.h>

int main(void)
{
  printf("Xaraktiras    O kodikos ASCII\n");
  printf("   %c                %d\n",'0','0');
  printf("   %c                %d\n",'1','1');
  printf("   %c                %d\n",'2','2');
  printf("   %c                %d\n",'3','3');
  printf("   %c                %d\n",'4','4');
  printf("   %c                %d\n",'5','5');
  printf("   %c                %d\n",'6','6');
  printf("   %c                %d\n",'7','7');
  printf("   %c                %d\n",'8','8');
  printf("   %c                %d\n",'9','9');
  return 0;
}
--:--  example_2_1_6.c      (C Abbrev)--L17--All---------------
```

Η εκτέλεση του παραπάνω προγράμματος τυπώνει τα ακόλουθα:

```
[student1@pc244 kef2]$
[student1@pc244 kef2]$ gcc example_2_1_6.c
[student1@pc244 kef2]$ a.out
Xaraktiras    O kodikos ASCII
   0                48
   1                49
   2                50
   3                51
   4                52
   5                53
   6                54
   7                55
   8                56
   9                57
[student1@pc244 kef2]$
```

1.1.14 Γράψτε ένα γενικό πρόγραμμα το οποίο τυπώνει το μέγεθος σε **bytes** κάθε ενός από τους βασικούς τύπους δεδομένων που συναντάμε στη C.

Υπόδειξη: Να χρησιμοποιήσετε τον τελεστή **sizeof()** ο οποίος επιστρέφει τον αριθμό των bytes για την κάθε μία μεταβλητή.

Μια πιθανή λύση είναι η ακόλουθη:

```
#include<stdio.h>

int main(void)
{

  char a1;

  int b1;
  short int b2;
  long int b3;

  float c1;
  double c2;
  long double c3;

  printf("The size of char in bytes is %d.\n",sizeof(a1));

  printf("The size of int in bytes is %d.\n",sizeof(b1));
  printf("The size of short int in bytes is %d.\n",sizeof(b2));
  printf("The size of long int in bytes is %d.\n",sizeof(b3));

  printf("The size of float in bytes is %d.\n",sizeof(c1));
  printf("The size of double in bytes is %d.\n",sizeof(c2));
  printf("The size of long double in bytes is %d.\n",sizeof(c3));

  return 0;

}
--:--  type_size.c       (C Abbrev)--L28--All-------------------------
```

Η εκτέλεση του παραπάνω προγράμματος δίνει το αποτέλεσμα:

```
[student1@pc244 kef2]$ gcc type_size.c
[student1@pc244 kef2]$ a.out
The size of char in bytes is 1.
The size of int in bytes is 4.
The size of short int in bytes is 2.
The size of long int in bytes is 4.
The size of float in bytes is 4.
The size of double in bytes is 8.
The size of long double in bytes is 12.
[student1@pc244 kef2]$
```

Να σημειωθεί εδώ πως τα μεγέθη των διαφόρων τύπων ισχύουν για μηχανές του τύπου PC που χρησιμοποιούμε. Το αποτέλεσμα του συγκεκριμένου προγράμματος σε άλλη μηχανή μπορεί να είναι διαφορετικό.

1.2 Προβλήματα.

1.2.1 Να γράψετε ένα πρόγραμμα το οποίο να τυπώνει σε τρεις στήλες τους ακεραίους αριθμούς 6, 10, 15, 28, 31, 63, 100, 127, 240 και 255 σε δεκαδική, οκταδική και δεκαεξαδική μορφή.

1.2.2 Να γράψετε ένα πρόγραμμα το οποίο να εμφανίζει σε πίνακα τις τιμές των ημιτόνων και συνημιτόνων για γωνίες 0º, 30º, 45º, 60º και 90º.

1.2.3 Να αναπτύξετε προγράμματα τα οποία να επαληθεύουν κάθε μία από τις ακόλουθες τριγωνομετρικές ισότητες:

$\sin(A-B) = \sin A \cos B - \cos A \sin B$
$\cos(A+B) = \cos A \cos B - \sin A \sin B$
$\cos(A-B) = \cos A \cos B + \sin A \sin B$
$\sin(2A) = 2 \sin A \cos A$
$\cos(2A) = \cos^2 A - \sin^2 A$
$\tan(2A) = (2 \tan A)/(1 - \tan^2 A)$
$\sin(3A) = 3 \sin A - 4 \sin^3 A$
$\cos(3A) = 4 \cos^3 A - 3 \cos A$
$\tan(3A) = (3 \tan A - \tan^3 A)/(1 - 3 \tan^2 A)$

Σε κάθε πρόγραμμα να υπολογίσετε ξεχωριστά το κάθε μέλος των παραπάνω ισοτήτων και να το εκτυπώσετε για διάφορες τιμές των μεταβλητών A και B. Δώστε μερικά παραδείγματα. Να διατηρήσετε ακρίβεια τεσσάρων δεκαδικών ψηφίων στους υπολογισμούς σας.

1.2.4 Να αναπτύξετε προγράμματα τα οποία να επαληθεύουν κάθε μία από τις ακόλουθες ισότητες

$\log(AB) = \log A + \log B$
$\log(A/B) = \log A - \log B$
$\log(A^n) = n \log A$

Σε κάθε πρόγραμμα να υπολογίσετε ξεχωριστά το κάθε μέλος των παραπάνω ισοτήτων και να το εκτυπώσετε για διάφορες τιμές των μεταβλητών A, B και n. Δώστε μερικά παραδείγματα. Να διατηρήσετε ακρίβεια τεσσάρων δεκαδικών ψηφίων στους υπολογισμούς σας.

1.2.5 Να αναπτύξετε προγράμματα τα οποία να επαληθεύουν κάθε μία από τις ακόλουθες υπερβολικές συναρτήσεις:

$$\sinh(x) = \frac{e^x - e^{-x}}{2}$$

$$\cosh(x) = \frac{e^{x} + e^{-x}}{2}$$

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

Σε κάθε πρόγραμμα να υπολογίσετε ξεχωριστά το κάθε μέλος των παραπάνω συναρτήσεων και να το εκτυπώσετε για διάφορες τιμές της μεταβλητής x. Δώστε μερικά παραδείγματα. Να διατηρήσετε ακρίβεια τεσσάρων δεκαδικών ψηφίων στους υπολογισμούς σας.

1.2.6 Να αναπτύξετε προγράμματα τα οποία να υπολογίζουν την αριθμητική τιμή για κάθε μία από τις ακόλουθες συναρτήσεις:

$$f(x) = \sqrt{\frac{5x^{5} - 3x^{3} + x - 1}{4}}$$

$$f(x) = \ln(x) * \sqrt{\frac{6x^{2} - 4x + 4}{2\ln(x)}}$$

$$f(x) = \ln(2x) * \sqrt{\frac{e^{3x} - 4}{5}}$$

και για διάφορες τιμές του x. Οι τιμές του x να εισάγονται στο πρόγραμμα με την χρήση της συνάρτησης scanf(). Να διατηρήσετε ακρίβεια τριών δεκαδικών ψηφίων στους υπολογισμούς σας. Εκτελέστε το πρόγραμμα δίνοντας τέσσερα δικά σας παραδείγματα.

1.2.7 Να αναπτύξετε αντίστοιχα προγράμματα για τις συναρτήσεις της προηγούμενης άσκησης, γράφοντας πρώτα μία συνάρτηση στη C η οποία να υπολογίζει γενικά την τιμή των $f(x)$ και στη συνέχεια να την χρησιμοποιήσετε στη συνάρτηση main().

1.2.8 Ένα σώμα εκτελεί πλάγια βολή. Το σώμα εκτοξεύεται με αρχική ταχύτητα v_0=100m/sec. Να γράψετε ένα πρόγραμμα το οποίο να υπολογίζει για πόσο χρόνο το σώμα μένει στον αέρα και ποιό είναι το βεληνεκές του όταν εκτοξεύεται υπό γωνία θ ίση με 30º, 40º, 45º, 50º και 60º. Αγνοείστε την τριβή του αέρα. Να παρουσιάσετε τα αποτελέσματα στοιχισμένα σε πίνακα. Στα αποτελέσματά σας να αναγράψετε και μονάδες. Να διατηρήσετε ακρίβεια τριών δεκαδικών ψηφίων στους υπολογισμούς σας. Δίνονται: g=9.81m/sec^2, t=(2 v_0 sinθ)/(g) και R=(v_0 cosθ)t

1.2.9 Μια ομάδα φοιτητών στα εργαστήρια μηχανικής παίρνει τις παρακάτω επαναληπτικές μετρήσεις για το μήκος ενός αντικειμένου:

α/α	Μήκος (cm)
1	10.3
2	10.7
3	9.6
4	11.1
5	10.9
6	9.7
7	10.1
8	9.9

Να γράψετε ένα πρόγραμμα το οποίο να υπολογίζει το μέσο μήκος του αντικειμένου και το πειραματικό τυπικό σφάλμα του μέσου μήκους. Να εκτυπώσετε το αποτέλεσμα αναγράφοντας και μονάδες. Να διατηρήσετε στα αποτελέσματά σας την ακρίβεια που προβλέπεται.

1.2.10 Μια ομάδα φοιτητών στα εργαστήρια ηλεκτρισμού παίρνει τις παρακάτω επαναληπτικές μετρήσεις τάσης και ρεύματος που διαρρέει μία ωμική αντίσταση:

α/α	V(Volts)	I(mA)
1	10.39	1.01
2	10.19	1.06
3	10.23	1.02
4	10.27	1.05
5	10.32	0.99
6	10.37	1.00
7	10.25	1.04
8	10.29	0.98

Να γράψετε ένα πρόγραμμα το οποίο να υπολογίζει την μέση τιμή και το σφάλμα της μέσης τιμής της τάσης και του ρεύματος. Στη συνέχεια κάνοντας χρήση των τύπων της σύνθετης μέτρησης να υπολογίσετε την τιμή και το σφάλμα της αντίστασης σε ΚΩ. Να εκτυπώσετε το αποτέλεσμα αναγράφοντας και μονάδες. Να διατηρήσετε στα αποτελέσματά σας την ακρίβεια που προβλέπεται.

Κεφάλαιο ΙΙ

Προβλήματα που αφορούν εντολές ελέγχου της ροής ενός προγράμματος.

Στο παρόν κεφάλαιο παρουσιάζονται προβλήματα τα οποία αφορούν κυρίως τις εντολές της C οι οποίες ελέγχουν την ροή εκτέλεσης σε ένα πρόγραμμα. Οι εντολές αυτές είναι οι ακόλουθες:

- **if-else και else-if**
- **switch**
- **while**
- **for**
- **do-while**
- **break** και **continue**
- **goto** και **ετικέτες**

Στα προβλήματα που ακολουθούν, εκτός από απλά προγράμματα τα οποία αναφέρονται στη χρήση των παραπάνω εντολών, παρουσιάζονται ασκήσεις οι οποίες αναφέρονται σε:

- **Διερεύνηση εξισώσεων**
- **Αναδρομικές σχέσεις**
- **Αριθμητικές σειρές**
- **Αναπτύγματα σε σειρές**
- **Υπολογισμός ορισμένων ολοκληρωμάτων**

2.1 Λυμένα Προβλήματα.

2.1.1 Να γραφεί ένα πρόγραμμα το οποίο να υπολογίζει την ρίζα ενός πραγματικού αριθμού (θετικού, μηδέν ή αρνητικού). Ο πραγματικός αριθμός να εισάγεται από το πληκτρολόγιο με την χρήση της συνάρτησης scanf(). Να χρησιμοποιήσετε την εντολή **if**.

Η τετραγωνική ρίζα ενός θετικού ή ίσου με το μηδέν πραγματικού αριθμού είναι επίσης πραγματικός αριθμός. Η τετραγωνική ρίζα αρνητικού πραγματικού αριθμού είναι φανταστικός αριθμός. Κατά συνέπεια διακρίνουμε δύο περιπτώσεις. Μια πιθανή λύση με την χρήση της εντολής if είναι η ακόλουθη:

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

int main(void)
{
  float x;
  printf("Eisagete ton arithmo x: ");
  scanf("%f",&x);

  if(x>=0.){
    printf("H tetragoniki riza tou x=%8.3f einai: %8.3f\n",x,sqrt(x));
      }

  if(x<0.){
    printf("H tetragoniki riza tou x=%8.3f einai: (%8.3f)i\n",x,sqrt(-x));
      }
  return 0;
}
-:--  example_sqrt.c      (C Abbrev)--L18--All----------------------------------
```

Η εκτέλεση του παραπάνω προγράμματος εικονίζεται παρακάτω:

```
[student1@pc244 kef3]$ gcc example_sqrt.c -lm
[student1@pc244 kef3]$ a.out
Eisagete ton arithmo x: 9
H tetragoniki riza tou x=   9.000 einai:    3.000
[student1@pc244 kef3]$ a.out
Eisagete ton arithmo x: -9
H tetragoniki riza tou x=  -9.000 einai: (   3.000)i
[student1@pc244 kef3]$
```

2.1.2 Να γράψετε ένα πρόγραμμα το οποίο να τυπώνει τους κεφαλαίους χαρακτήρες (A έως Z) και τους μικρούς χαρακτήρες (a έως z) του λατινικού αλφαβήτου και τους αντίστοιχους κωδικούς ASCII. Να χρησιμοποιήσετε την εντολή **for**. Οι χαρακτήρες A έως Z αντιστοιχούν στους κωδικούς ASCII 65 έως 90 ενώ οι χαρακτήρες a έως z αντιστοιχούν στους κωδικούς ASCII 97 έως 122. Να γράψετε το ίδιο πρόγραμμα χρησιμοποιώντας τις εντολές **while** και **do-while**.

Μια πιθανή λύση με την χρήση της εντολής for είναι η ακόλουθη:

```
#include<stdio.h>

int main(void)
{
  int i;

  printf(" Kefalaia     ASCII    Mikra    ASCII\n");

  for(i=1;i<=26;i++)
    printf("    %c           %d       %c       %3d\n",i+64,i+64,i+96,i+96);

  return 0;
}
--:--  example_for.c      (C Abbrev)--L13--All----------------------------------
```

Μια πιθανή λύση με την χρήση της εντολής **while** είναι η ακόλουθη:

```
#include<stdio.h>

int main(void)
{
  int i;

  printf(" Kefalaia     ASCII    Mikra    ASCII\n");

  i=1;
  while(i<=26){
    printf("    %c           %d       %c       %3d\n",i+64,i+64,i+96,i+96);
    i++;
  }

  return 0;
}
--:--  example_while.c      (C Abbrev)--L16--All----------------------------------
```

Μια πιθανή λύση με την χρήση της εντολής **do-while** είναι η ακόλουθη:

```
#include<stdio.h>

int main(void)
{
  int i;

  printf(" Kefalaia     ASCII    Mikra    ASCII\n");

  i=1;
  do{
    printf("    %c           %d       %c       %3d\n",i+64,i+64,i+96,i+96);
    i++;
  }while(i<=26);

  return 0;
}
--:--  example_dowhile.c      (C Abbrev)--L16--All----------------------------------
```

Η εκτέλεση των παραπάνω προγραμμάτων δίνει το παρακάτω αποτέλεσμα:

```
[student1@pc244 kef3]$ gcc example_for.c
[student1@pc244 kef3]$ a.out
 Kefalaia     ASCII    Mikra    ASCII
    A           65       a        97
    B           66       b        98
    C           67       c        99
    D           68       d       100
    E           69       e       101
    F           70       f       102
    G           71       g       103
    H           72       h       104
    I           73       i       105
    J           74       j       106
    K           75       k       107
    L           76       l       108
    M           77       m       109
    N           78       n       110
    O           79       o       111
    P           80       p       112
    Q           81       q       113
    R           82       r       114
    S           83       s       115
    T           84       t       116
    U           85       u       117
    V           86       v       118
    W           87       w       119
    X           88       x       120
    Y           89       y       121
    Z           90       z       122
[student1@pc244 kef3]$
```

2.1.3 Να γραφεί ένα πρόγραμμα το οποίο να υπολογίζει το άθροισμα $\sum_{n=1}^{n=m} n$. Ο ακέραιος αριθμός m να εισάγεται από το πληκτρολόγιο με την χρήση της συνάρτησης scanf(). Εκτελέστε το πρόγραμμα και εκτυπώστε το αποτέλεσμα για m=100 και m=200.

Μια πιθανή λύση με την χρήση της εντολής for είναι η ακόλουθη:

```
#include<stdio.h>

int main(void){
  int m,i;
  double sum;

  printf("Dose ton arithmo m: ");
  scanf("%d",&m);

  sum=0;
  for(i=1; i<=m; i++){
    sum=sum+(double)i;
  }

  printf("To athrisma isoute me:%.2f \n",sum);
  return 0;
}
u:--  athrisma.c        (C Abbrev)--L17--All----------------------------
```

Η εκτέλεση του παραπάνω προγράμματος εικονίζεται παρακάτω:

```
[panos@pc-247 lymena_problimata]$
[panos@pc-247 lymena_problimata]$ gcc athrisma.c
[panos@pc-247 lymena_problimata]$ a.out
Dose ton arithmo m: 100
To athrisma isoute me:5050.00
[panos@pc-247 lymena_problimata]$ a.out
Dose ton arithmo m: 200
To athrisma isoute me:20100.00
[panos@pc-247 lymena_problimata]$
```

Προσοχή! Όταν υπολογίζουμε ένα άθροισμα πρέπει πάντοτε να αρχικοποιούμε στο μηδέν την μεταβλητή που χρησιμοποιούμε για τον υπολογισμό του.

2.1.4 Να γράψετε μια συνάρτηση η οποία να υπολογίζει το **παραγοντικό** ενός θετικού ακεραίου αριθμού. Στη συνέχεια να την χρησιμοποιήσετε για να υπολογίζετε το παραγοντικό οποιουδήποτε ακεραίου εισάγετε από το πληκτρολόγιο. Προστατέψτε το πρόγραμμά σας από εισαγωγή αρνητικών ακεραίων αριθμών. Δίνονται:

n! = 1*2*3*…*(n-1)*n

0! = 1

Το παραγοντικό είναι μία συνάρτηση η οποία επιστρέφει γενικά πολύ μεγάλους αριθμούς. Για τον λόγο αυτό θα χρησιμοποιήσουμε double αριθμούς στον υπολογισμό του. Μια πιθανή λύση του προβλήματος είναι η ακόλουθη:

```
#include<stdio.h>

double factorial(int a); /* Dilosi prototipo sinartisis */

int main(void)
{
  int n;

  printf("Doste ton akeraio n:\n");
  scanf("%d",&n);

                  /* Elegxos gia arnitikous akeraious */
  if(n<0){
  printf("To paragontiko toy arithmou n=%d den orizetai.\n",n);
  return 0;
  }

  printf("To paragontiko toy arithmou n=%d einai:%e\n",n,factorial(n));
  return 0;
}

double factorial(int a){
  int i;
  double result;

  result=1.;
  for (i=1;i<=a;i++)
    result*=i;

  return result;
}
-:--  example_factorial_big.c      (C Abbrev)--L31--All----------------------
```

Προσέξτε τον έλεγχο για αρνητικούς ακεραίους ο οποίος υλοποιείται με μια απλή εντολή if. Ακολουθούν παραδείγματα εκτέλεσης του προγράμματος.

```
[student1@pc244 kef3]$ gcc example_factorial_big.c
[student1@pc244 kef3]$ a.out
Doste ton akeraio n:
4
To paragontiko toy arithmou n=4 einai:2.400000e+01
[student1@pc244 kef3]$ a.out
Doste ton akeraio n:
20
To paragontiko toy arithmou n=20 einai:2.432902e+18
[student1@pc244 kef3]$ a.out
Doste ton akeraio n:
120
To paragontiko toy arithmou n=120 einai:6.689503e+198
[student1@pc244 kef3]$ a.out
Doste ton akeraio n:
0
To paragontiko toy arithmou n=0 einai:1.000000e+00
[student1@pc244 kef3]$ a.out
Doste ton akeraio n:
-15
To paragontiko toy arithmou n=-15 den orizetai.
```

2.1.5 Δίδεται η συνάρτηση:

$$f(n) = \frac{\sqrt{n^3 + n^2 + n + 1}}{n - 10}$$

όπου n ακέραιος. Να γραφεί ένα πρόγραμμα το οποίο να υπολογίζει την τιμή της για κάθε n το οποίο εισάγεται από το πληκτρολόγιο. Να γίνει διερεύνηση ώστε να αποκλεισθούν μη πραγματικές τιμές και τυχόν απειρισμοί της συνάρτησης.

Η παραπάνω συνάρτηση έχει πραγματικές τιμές όταν η υπόριζος ποσότητα είναι >=0. Επίσης η συνάρτηση απειρίζεται όταν n=10. Μια πιθανή λύση του προβλήματος είναι η ακόλουθη:

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

float fun(float a); /* Dilosi prototipo sinartisis */

int main(void)
{
  int n;

  printf("Doste tin timi gia to n:\n");
  scanf("%d",&n);

  if((pow(n,3)+pow(n,2)+n+1.)<0.){
    printf("H parastasi den einai pragmatiki.\n");
    return 0;
  }

  if(n==10){
    printf("H parastasi apeirizetai.\n");
    return 0;
  }

  printf("H timi tis parastasis gia n=%d einai:%f\n",n,fun(n));
  return 0;
}

float fun(float a)
{
  float result;

  result = (sqrt(pow(a,3)+pow(a,2)+a+1))/(a-10.);
  return result;
}
```

Στο συγκεκριμένο παράδειγμα χρησιμοποιούμε μεταβλητή ακέραια. **Προσοχή** πρέπει να δοθεί όταν συγκρίνουμε πραγματικούς αριθμούς (για παράδειγμα εάν ο n ήταν πραγματικός τότε θα είχαμε την εντολή if(n==10) με πραγματικές τιμές). Γενικά πρέπει να αποφεύγονται τέτοιες συγκρίσεις οι οποίοι πολλές φορές οδηγούν σε λάθη. Συγκρίσεις με ακεραίους αριθμούς είναι καθ' όλα αποδεκτό. Ακολουθεί η εκτέλεση του προγράμματος με μερικά παραδείγματα.

```
[student1@pc244 kef3]$ a.out
Doste tin timi gia to n:
35
H timi tis parastasis gia n=35 einai:8.403428
[student1@pc244 kef3]$
[student1@pc244 kef3]$
[student1@pc244 kef3]$ a.out
Doste tin timi gia to n:
10
H parastasi apeirizetai.
[student1@pc244 kef3]$
[student1@pc244 kef3]$
[student1@pc244 kef3]$ a.out
Doste tin timi gia to n:
-6
H parastasi den einai pragmatiki.
[student1@pc244 kef3]$
```

2.1.6 Δίνεται η αναδρομική σχέση

$$a_n = \frac{3a_{n-1}^2 - 1}{a_{n-1}}, \text{ με } a_0 = 1$$

Να γραφεί ένα πρόγραμμα για n=1,2,3,4… με την βοήθεια του οποίου να υπολογίζετε την τιμή του n-οστου όρου (όπου το n να εισάγεται από το πληκτρολόγιο). Εκτελέστε το πρόγραμμα και υπολογίστε τους όρους για n= 2, 5, 10, 20.

Μια πιθανή λύση είναι η ακόλουθη:

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

int main(void)
{
  int n,i;
  float a0, an, an1;

  printf("Doste ton oro n tis anadromikis sxesis, n:\n");
  scanf("%d",&n);

  if(n<0){        /* Elegxos gia arnitikes times tou n */
    printf("Eisagate arnitiki timi gia to n.\n");
    return 0;
  }

  a0=1.;          /* O arxikos oros a_0 */
  an1=a0;
  for(i=1;i<=n;++i){
    an = (3.*pow(an1,2)-1.)/an1;
    an1=an;
  }

  printf("O %d-tos oros tis anadromikis sxesis einai a_n=%e\n",n,an);
  return 0;
}
```

Η εκτέλεση του προγράμματος εικονίζεται παρακάτω:

```
[student1@pc244 kef3]$ a.out
Doste ton oro n tis anadromikis sxesis, n:
2
O 2-tos oros tis anadromikis sxesis einai a_n=5.500000e+00
[student1@pc244 kef3]$ a.out
Doste ton oro n tis anadromikis sxesis, n:
5
O 5-tos oros tis anadromikis sxesis einai a_n=1.466593e+02
[student1@pc244 kef3]$ a.out
Doste ton oro n tis anadromikis sxesis, n:
10
O 10-tos oros tis anadromikis sxesis einai a_n=3.563760e+04
[student1@pc244 kef3]$ a.out
Doste ton oro n tis anadromikis sxesis, n:
20
O 20-tos oros tis anadromikis sxesis einai a_n=2.104364e+09
[student1@pc244 kef3]$
```

2.1.7 Δίνεται η αναδρομική σχέση

$$a_n = 3a_{n-1} - a_{n-2}, \text{ με } a_0 = 1 \text{ και } a_1 = 2$$

Να γραφεί ένα πρόγραμμα για n=2,3,4… με την βοήθεια του οποίου να υπολογίζετε την τιμή του n-οστου όρου (όπου το n να εισάγεται από το πληκτρολόγιο). Εκτελέστε το πρόγραμμα και υπολογίστε τους όρους για n= 2, 3, 4, 5 και 6.

Μια πιθανή λύση είναι η ακόλουθη:

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

int main(void)
{
  int n,i;
  float a0, a1, an, an1, an2;

  printf("Doste ton oro n tis anadromikis sxesis, n:\n");
  scanf("%d",&n);

  if(n<2){        /* Elegxos gia dynates times tou n */
    printf("Eisagate timi gia to n mikroteri tou 2.\n");
    return 0;
  }

  a0=1.;           /* Arxikopoihsi */
  a1=2.;
  an2=a0;
  an1=a1;

  for(i=2;i<=n;++i){
    an = 3.*an1 - an2;     /* Ypologismos tou orou n */
    an2 = an1;             /* Epistrofi tou orou n-2 gia to epomeno loop */
    an1 = an;              /* Epistrofi tou orou n-1 gia to epomeno loop */
  }

  printf("O %d-tos oros tis anadromikis sxesis einai a_n=%e\n",n,an);
  return 0;
}
```

Η εκτέλεση του προγράμματος εικονίζεται παρακάτω:

```
[student1@pc244 kef3]$ a.out
Doste ton oro n tis anadromikis sxesis, n:
2
O 2-tos oros tis anadromikis sxesis einai a_n=5.000000e+00
[student1@pc244 kef3]$ a.out
Doste ton oro n tis anadromikis sxesis, n:
3
O 3-tos oros tis anadromikis sxesis einai a_n=1.300000e+01
[student1@pc244 kef3]$ a.out
Doste ton oro n tis anadromikis sxesis, n:
4
O 4-tos oros tis anadromikis sxesis einai a_n=3.400000e+01
[student1@pc244 kef3]$ a.out
Doste ton oro n tis anadromikis sxesis, n:
5
O 5-tos oros tis anadromikis sxesis einai a_n=8.900000e+01
[student1@pc244 kef3]$ a.out
Doste ton oro n tis anadromikis sxesis, n:
6
O 6-tos oros tis anadromikis sxesis einai a_n=2.330000e+02
[student1@pc244 kef3]$
```

2.1.8 Δίνεται η ακόλουθη αριθμητική σειρά:

$$1^2 + 2^2 + 3^2 + ... + n^2 = \frac{n(n+1)(2n+1)}{6}$$

Να αναπτύξετε ένα πρόγραμμα το οποίο να υπολογίζει χωριστά τα δύο μέρη της αριθμητικής προόδου και να επαληθεύσετε τις σχέσεις με μερικά παραδείγματα. Το n να δίνεται από το πληκτρολόγιο

Μια πιθανή λύση είναι η ακόλουθη:

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

int main(void)
{
  int n,i;
  float left_side,right_side;

  printf("Eisagete ton akeraio n:\n");
  scanf("%d",&n);

  left_side=0;    /* arxikopoihsi */

  for(i=1;i<=n;i++)
    left_side+=pow(i,2);

  right_side=(n*(n+1)*(2*n+1))/6.;

  printf("To aristero meros ths proodou gia n=%d einai:%10.2f\n",n,left_side);
  printf("To dexi meros ths proodou gia n=%d einai    :%10.2f\n",n,right_side);
  return 0;
}
```

Η εκτέλεση του προγράμματος εικονίζεται παρακάτω:

```
[student1@pc244 kef3]$ a.out
Eisagete ton akeraio n:
8
To aristero meros ths proodou gia n=8 einai:    204.00
To dexi meros ths proodou gia n=8 einai    :    204.00
[student1@pc244 kef3]$
[student1@pc244 kef3]$
[student1@pc244 kef3]$ a.out
Eisagete ton akeraio n:
31
To aristero meros ths proodou gia n=31 einai:  10416.00
To dexi meros ths proodou gia n=31 einai    :  10416.00
[student1@pc244 kef3]$
```

2.1.9 Δίνεται η ακόλουθη σειρά:

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + ... \quad \text{για} \quad -\infty < x < \infty$$

Να αναπτύξετε ένα πρόγραμμα το οποίο να υπολογίζει χωριστά τα δύο μέρη της παράστασης και να επαληθεύσετε τη σχέση με μερικά παραδείγματα. Το x να δίνεται από το πληκτρολόγιο.

Μια πιθανή λύση είναι η ακόλουθη:

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

double factorial(int a); /* Dilosi prototipo sinartisis */

int main(void)
{
  int n,i;
  float x,left_side,right_side;

  printf("Eisagete ton arithmo x:\n");
  scanf("%f",&x);
  printf("Eisagete ton thetiko akeraio n:\n");
  scanf("%d",&n);

  right_side=0;         /* arxikopoihsi */
  for(i=0;i<=n;i++)
    right_side+=pow(x,i)/factorial(i);

  left_side=exp(x);

  printf("To aristero melos ths parastasis gia x=%f einai     : %10.3f\n",x,left_side);
  printf("To dexi melos ths parastasis gia x=%f kai n=%d einai : %10.3f\n",x,n,right_side);
  return 0;
}

double factorial(int a){
  int i;
  double result;

  result=1.;
  for (i=1;i<=a;i++)
    result*=i;

  return result;
}
--:**  example_3_1_9.c      (C Abbrev)--L36--All----------------------------------------------
```

Σημειώνουμε πως στο παραπάνω πρόγραμμα δεν έχουμε βάλει προστασία για τιμές του n<0. Το σωστό θα ήταν ένας έλεγχος στην τιμή του n και έξοδος από το πρόγραμμα εάν ο χρήστης εισάγει n<0. Για το πρόβλημά μας χρειαζόμαστε την συνάρτηση του παραγοντικού την οποία αναπτύξαμε σε προηγούμενο παράδειγμα.

Στη συνέχεια παρουσιάζουμε ένα παράδειγμα όπου x=3.78. Υπολογίζουμε το δεξί μέλος της παράστασης για n=4,8 και 15. Παρατηρείστε πως η τιμή του δεξιού μέλους της παράστασης πλησιάζει σιγά-σιγά αυτήν του αριστερού.

```
[student1@pc244 kef3]$ gcc example_3_1_9.c -lm
[student1@pc244 kef3]$ a.out
Eisagete ton arithmo x:
3.78
Eisagete ton thetiko akeraio n:
4
To aristero melos ths parastasis gia x=3.780000 einai     :     43.816
To dexi melos ths parastasis gia x=3.780000 kai n=4 einai :     29.432
[student1@pc244 kef3]$ a.out
Eisagete ton arithmo x:
3.78
Eisagete ton thetiko akeraio n:
8
To aristero melos ths parastasis gia x=3.780000 einai     :     43.816
To dexi melos ths parastasis gia x=3.780000 kai n=8 einai :     43.137
[student1@pc244 kef3]$ a.out
Eisagete ton arithmo x:
3.78
Eisagete ton thetiko akeraio n:
15
To aristero melos ths parastasis gia x=3.780000 einai     :     43.816
To dexi melos ths parastasis gia x=3.780000 kai n=15 einai :     43.816
[student1@pc244 kef3]$
```

2.1.10 Δίνεται η ακόλουθη σειρά:

$$\frac{1}{1^4}+\frac{1}{3^4}+\frac{1}{5^4}+\frac{1}{7^4}+....=\frac{\pi^4}{96}$$

Να γραφεί ένα πρόγραμμα το οποίο να υπολογίζει αρχικά την τιμή σύγκλισης της σειράς, Στη συνέχεια να κάνετε έναν πίνακα ο οποίος να περιέχει τα αθροίσματα με 1,2,3 έως 12 όρους. Διατηρείστε ακρίβεια 5 δεκαδικών ψηφίων. Παρατηρείστε την σύγκλιση.

Μια πιθανή λύση είναι η ακόλουθη:

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

float serie(int a);  /* Dilosi prototipo sinartisis */

int main(void)
{
  int i;
  float pi=3.14159;

  printf("H seira siglinei sto : %8.5f\n",pow(pi,4)/96.);
  printf("n oroi     Timi athrismatos\n");

  for(i=1;i<=12;i++)
    printf(" %2d          %8.5f\n",i,serie(i));

  return 0;
}

float serie(int a)
{
  int i, ii;
  float result;

  result=0;
  for(i=1;i<=a;i++){
    ii=2*i-1;
    result += 1./pow(ii,4);
  }
  return result;
}
-:--  example_3_1_10.c      (C Abbrev)--L31--All--------------------------
```

Η εκτέλεση του προγράμματος εικονίζεται παρακάτω:

```
[student1@pc244 kef3]$ gcc example_3_1_10.c -lm
[student1@pc244 kef3]$ a.out
H seira siglinei sto :  1.01467
n oroi     Timi athrismatos
 1          1.00000
 2          1.01235
 3          1.01395
 4          1.01436
 5          1.01451
 6          1.01458
 7          1.01462
 8          1.01464
 9          1.01465
10          1.01466
11          1.01466
12          1.01467
```

2.1.11 Να υπολογιστεί αριθμητικά με τον **κανόνα του τραπεζίου** το παρακάτω ορισμένο ολοκλήρωμα:

$$\int_a^b \frac{dx}{\sqrt{3x+2}} = \left.\frac{2\sqrt{3x+2}}{3}\right|_a^b$$

Να συγκρίνετε την τιμή που βρίσκετε ολοκληρώνοντας αριθμητικά με αυτή που βρίσκετε ολοκληρώνοντας αναλυτικά . Να χρησιμοποιήσετε 100000 βήματα κατά την αριθμητική ολοκλήρωση. Τα όρια της ολοκλήρωσης να δίνονται από το πληκτρολόγιο. Προστατέψτε το πρόγραμμα από λανθασμένα όρια ολοκλήρωσης. Διατηρείστε ακρίβεια 5 δεκαδικών ψηφίων στην εκτύπωση του αποτελέσματος. Εκτελέστε το πρόγραμμα για όρια ολοκλήρωσης a=0 έως b=10 και a=15 έως b=100.

Μια πιθανή λύση είναι η ακόλουθη:

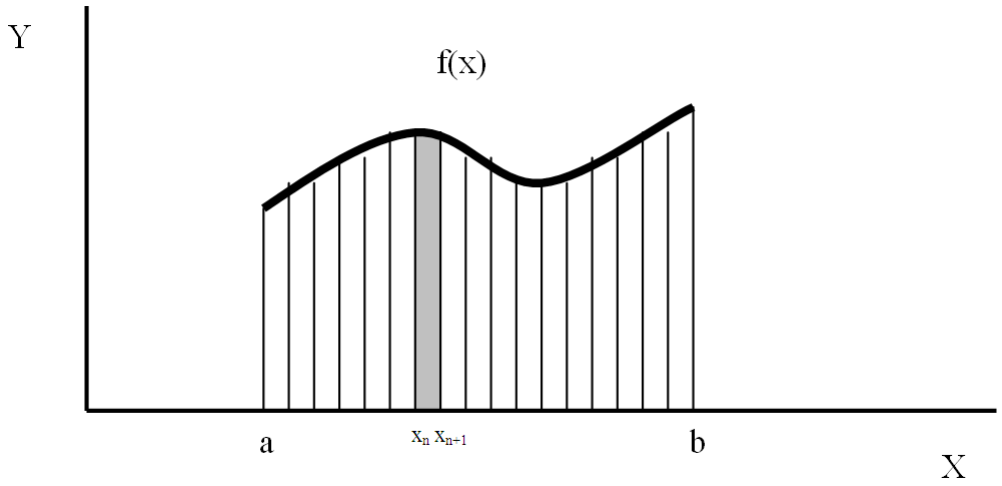
```
#include<stdio.h>
#include<math.h>

float f_x(float x);  /* Dilosi prototipo sinartisis */

int main(void)
{
  int number_of_steps,i;
  float a,b;
  float interval,step;
  float kato_orio;
  float area,analitiki_timi;

  printf("Eisagete ta oria oloklirosis a kai b:\n");
  scanf("%f %f",&a,&b);
  if((a<=-2./3)||(b<=-2./3.)||a>=b){   /* elegxos ton orion oloklirosis */
    printf("Problima me ta oria oloklirosis.\n");
    return 0;
  }

  number_of_steps=100000;              /* arithmos bimaton oloklirosis */
  interval=b-a;
  step=interval/number_of_steps;       /* bima oloklirosis */

  area=0.;
  kato_orio=a;
  for(i=1;i<=number_of_steps;i++){
    area+=0.5*step*(f_x(kato_orio)+f_x(kato_orio+step));
    kato_orio+=step;
  }
  analitiki_timi=(2.*sqrt(3.*b+2.)/3.)-(2.*sqrt(3.*a+2.)/3.);
  printf("Apotelesma me arithmitiki oloklirosi: %10.5f\n",area);
  printf("Apotelesma me analitiki oloklirosi  : %10.5f\n",analitiki_timi);
  return 0;
}

float f_x(float x){
  float result;
  result=1./(sqrt(3.*x+2.));
  return result;
}
-:**  example_3_1_11.c      (C Abbrev)--L41--All----------------------------------
```

Η εκτέλεση του προγράμματος εικονίζεται παρακάτω:

```
[student1@pc244 kef3]$
[student1@pc244 kef3]$ gcc example_3_1_11.c -lm
[student1@pc244 kef3]$ a.out
Eisagete ta oria oloklirosis a kai b:
0 10
Apotelesma me arithmitiki oloklirosi:    2.82833
Apotelesma me analitiki oloklirosi  :    2.82843
[student1@pc244 kef3]$
[student1@pc244 kef3]$ a.out
Eisagete ta oria oloklirosis a kai b:
15 100
Apotelesma me arithmitiki oloklirosi:    7.01444
Apotelesma me analitiki oloklirosi  :    7.01500
[student1@pc244 kef3]$
```

Σχόλια:

Ο **κανόνας του τραπεζίου** αποτελεί την πιο απλή αριθμητική μέθοδο για τον υπολογισμό ενός ορισμένου ολοκληρώματος. Βασίζεται στον χωρισμό του εμβαδού, που περιλαμβάνει το ολοκλήρωμα, σε πολλά τραπέζια το καθένα με "απειροστό" ύψος το οποίο ονομάζουμε **βήμα ολοκλήρωσης**.



Στο παραπάνω σχήμα βλέπουμε τον χωρισμό του εμβαδού, που περιλαμβάνει το ολοκλήρωμα, σε πολλά "στοιχειώδη" τραπέζια, ένα από τα οποία έχουμε σκιαγραφήσει. Το εμβαδόν αυτού του "στοιχειώδους" τραπεζίου ισούται με:

$$S = \frac{1}{2}(x_{n+1} - x_n)\left(f(x_{n+1}) + f(x_n)\right)$$

Αθροίζοντας τα εμβαδά όλων των "στοιχειωδών" τραπεζίων από το a έως το b υπολογίζουμε το ολοκλήρωμα. Η μέθοδος αυτή ονομάζεται **αριθμητική ολοκλήρωση** (**numerical integration**) με εφαρμογή του κανόνα του τραπεζίου. Καταλαβαίνουμε πως όσο πιο μικρό είναι το βήμα ολοκλήρωσης (ή αλλιώς όσο περισσότερα είναι τα βήματα), τόσο η τιμή που υπολογίζουμε βρίσκεται πιο κοντά στην πραγματική τιμή του ολοκληρώματος.

2.1.12 Να υπολογιστεί αριθμητικά με τον **κανόνα του Simpson** το ορισμένο ολοκλήρωμα του προηγούμενου παραδείγματος. Να συγκρίνετε την τιμή που βρίσκετε ολοκληρώνοντας αριθμητικά με αυτή που βρίσκετε ολοκληρώνοντας αναλυτικά . Να χρησιμοποιήσετε 100000 βήματα κατά την αριθμητική ολοκλήρωση. Τα όρια της ολοκλήρωσης να δίνονται από το πληκτρολόγιο. Προστατέψτε το πρόγραμμα από λανθασμένα όρια ολοκλήρωσης. Διατηρείστε ακρίβεια 5 δεκαδικών ψηφίων στην εκτύπωση του αποτελέσματος. Εκτελέστε το πρόγραμμα για όρια ολοκλήρωσης a=0 έως b=10 και a=15 έως b=100.

Μια πιθανή λύση είναι η ακόλουθη:

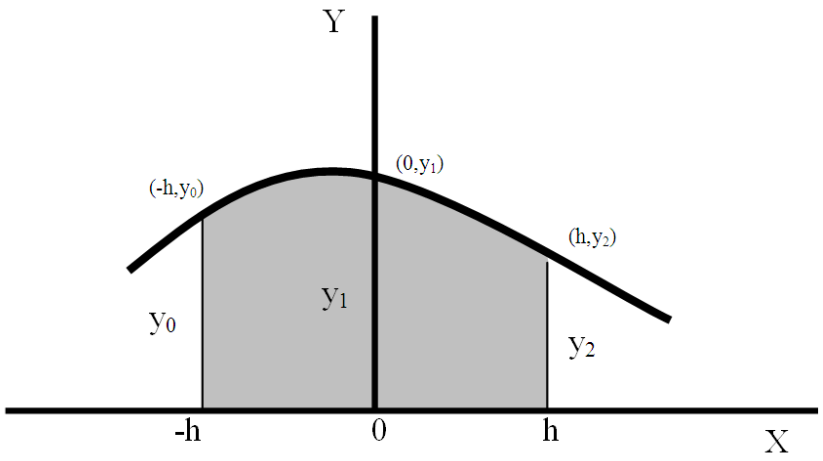
```
#include<stdio.h>
#include<math.h>

float f_x(float x);  /* Dilosi prototipo sinartisis */

int main(void)
{
  int number_of_steps,i;
  float a,b;
  float interval,step;
  float kato_orio;
  float area,analitiki_timi;

  printf("Eisagete ta oria oloklirosis a kai b:\n");
  scanf("%f %f",&a,&b);
  if((a<=-2./3)||(b<=-2./3.)||a>=b){   /* elegxos ton orion oloklirosis */
    printf("Problima me ta oria oloklirosis.\n");
    return 0;
  }

  number_of_steps=100000;              /* arithmos bimaton oloklirosis */
  interval=b-a;
  step=interval/number_of_steps;       /* bima oloklirosis */

  area=0.;
  kato_orio=a;
  for(i=1;i<=number_of_steps/2;i++){
    area+=step/3.*(f_x(kato_orio)+4.*f_x(kato_orio+step)+f_x(kato_orio+2.*step));
    kato_orio+=2.*step;
  }
  analitiki_timi=(2.*sqrt(3.*b+2.)/3.)-(2.*sqrt(3.*a+2.)/3.);
  printf("Apotelesma me arithmitiki oloklirosi: %10.5f\n",area);
  printf("Apotelesma me analitiki oloklirosi  : %10.5f\n",analitiki_timi);
  return 0;
}

float f_x(float x){
  float result;
  result=1./(sqrt(3.*x+2.));
  return result;
}
--:**  example_3_1_12.c      (C Abbrev)--L41--All----------------------------------------
```

Η εκτέλεση του προγράμματος εικονίζεται παρακάτω:

```
[student1@pc244 kef3]$ gcc example_3_1_12.c -lm
[student1@pc244 kef3]$ a.out
Eisagete ta oria oloklirosis a kai b:
0 10
Apotelesma me arithmitiki oloklirosi:    2.82855
Apotelesma me analitiki oloklirosi  :    2.82843
[student1@pc244 kef3]$
[student1@pc244 kef3]$ a.out
Eisagete ta oria oloklirosis a kai b:
15 100
Apotelesma me arithmitiki oloklirosi:    7.01442
Apotelesma me analitiki oloklirosi  :    7.01500
[student1@pc244 kef3]$
```

Σχόλια:

Ο **κανόνας του Simpson** αποτελεί μια απλή αριθμητική μέθοδο για τον υπολογισμό ενός ορισμένου ολοκληρώματος. Βασίζεται στον υπολογισμό του εμβαδού κάτω από την παραβολή με μορφή $y = Ax^2 + Bx + C$ η οποία εικονίζεται στο παρακάτω σχήμα.



Στο παραπάνω σχήμα το εμβαδόν της σκιαγραφημένης περιοχής ισούται με:

$$S = \frac{h}{3}\left(y_0 + 4y_1 + y_2\right)$$

Λαμβάνοντας υπ' όψη τα παραπάνω μπορούμε να χωρίσουμε το εμβαδόν, που περιλαμβάνει ένα ολοκλήρωμα, σε πολλά "στοιχειώδη" εμβαδά αλλά Simpson, το καθένα από τα οποία υπολογίζουμε με τον παραπάνω τύπο.

Αθροίζοντας όλα “στοιχειώδη” εμβαδά υπολογίζουμε το ολοκλήρωμα. Η μέθοδος αυτή ονομάζεται **αριθμητική ολοκλήρωση** (**numerical integration**) με εφαρμογή του κανόνα του Simpson. Ισχύουν και εδώ ότι αναφέραμε και στην εφαρμογή του κανόνα του τραπεζίου.

2.1.13 Να υπολογιστεί αναλυτικά το παρακάτω ορισμένο ολοκλήρωμα:

$$\int_0^{10} e^{3x} dx = \frac{e^{3x}}{3}\Big|_0^{10}$$

Μεταβάλλοντας τον αριθμό των βημάτων ολοκλήρωσης για 10, 20, 30,...100 βήματα, καταγράψτε σε πίνακα την τιμή του ολοκληρώματος με τον **κανόνα του τραπεζίου** και με τον **κανόνα του Simpson**. Διατηρείστε ακρίβεια 3 δεκαδικών ψηφίων στην εκτύπωση του αποτελέσματος.

Μια πιθανή λύση είναι η ακόλουθη:

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

float f_x(float x); /* Dilosi prototipo sinartisis */

int main(void)
{
  int number_of_steps,i,j;
  float analitiki_timi,interval,step,kato_orio,area_trapezoidal,area_simpson;

  analitiki_timi=exp(3.*10.)/3.-exp(3.*0.)/3.;
  printf("Apotelesma me analitiki oloklirosi  : %10.3e\n",analitiki_timi);
  printf("Bimata    Trapezio       Simpson\n");

  interval=10.-0.;                      /* diastima oloklirosis */

  for(j=1;j<=10;j++){
    number_of_steps=j*10;               /* arithmos bimaton oloklirosis */
    step=interval/number_of_steps;      /* bima oloklirosis */

    area_trapezoidal=0.;                /* kanonas tou trapeziou */
    kato_orio=0.;
    for(i=1;i<=number_of_steps;i++){
      area_trapezoidal+=0.5*step*(f_x(kato_orio)+f_x(kato_orio+step));
      kato_orio+=step;
    }

    area_simpson=0.;                    /* kanonas tou simpson */
    kato_orio=0.;
    for(i=1;i<=number_of_steps/2;i++){
      area_simpson+=step/3.*(f_x(kato_orio)+4.*f_x(kato_orio+step)+f_x(kato_orio+2.*step));
      kato_orio+=2.*step;
    }

  printf("%4d     %10.3e     %10.3e\n", number_of_steps,area_trapezoidal,area_simpson);
  }
  return 0;
}

float f_x(float x){
  return exp(3.*x);
}
--:--  example_3_1_13.c      (C Abbrev)--L42--Bot----------------------------------------------
```

Η εκτέλεση του παραπάνω προγράμματος εικονίζεται παρακάτω. Παρατηρούμε πως η μέθοδος Simpson δίνει την τιμή του ολοκληρώματος με λιγότερα βήματα ολοκλήρωσης από αυτήν του Τραπεζίου.

```
[student1@pc244 kef3]$
[student1@pc244 kef3]$ gcc example_3_1_13.c -lm
[student1@pc244 kef3]$ a.out
Apotelesma me analitiki oloklirosi  :  3.562e+12
Bimata    Trapezio       Simpson
  10      5.903e+12      4.291e+12
  20      4.206e+12      3.641e+12
  30      3.854e+12      3.580e+12
  40      3.728e+12      3.568e+12
  50      3.668e+12      3.565e+12
  60      3.636e+12      3.563e+12
  70      3.617e+12      3.563e+12
  80      3.604e+12      3.563e+12
  90      3.595e+12      3.562e+12
 100      3.589e+12      3.562e+12
[student1@pc244 kef3]$
```

2.2 Προβλήματα.

2.2.1 Δίνονται οι συναρτήσεις:

$$f(n) = \frac{\sqrt{5n^3 + 3n + 1}}{(n-5)(n-12)}$$

$$f(x) = \ln(x)\sqrt{x^5 + x^3 + x}$$

$$f(x) = \frac{e^{\sqrt{2x-10}} - 4}{\ln(x)}$$

όπου x πραγματικός αριθμός και n ακέραιος. Να γραφεί ένα πρόγραμμα για κάθε μία συνάρτηση το οποίο να υπολογίζει την τιμή της για κάθε n ή x το οποίο εισάγεται από το πληκτρολόγιο. Να γίνει διερεύνηση ώστε να αποκλεισθούν μη πραγματικές τιμές και τυχόν απειρισμοί της συνάρτησης.

2.2.2 Να γραφεί ένα πρόγραμμα το οποίο να υπολογίζει τις πραγματικές λύσεις του τριωνύμου:

$$ax^2 + bx + c = 0$$

όπου οι a, b, c είναι πραγματικοί αριθμοί. Να γίνει **πλήρης διερεύνηση** του προβλήματος. Δίνονται:

$$D = b^2 - 4ac\ ,\ x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

2.2.3 Να γραφεί ένα πρόγραμμα το οποίο να υπολογίζει τις πραγματικές λύσεις της τριτοβάθμιας εξίσωσης:

$$x^3 + a_1 x^2 + a_2 x + a_3 = 0$$

όπου οι a_1, a_2, a_3 είναι πραγματικοί αριθμοί. Να γίνει **πλήρης διερεύνηση** του προβλήματος.

Υπόδειξη: Δίνεται η γενική λύση του προβλήματος:

Έστω $Q = \frac{3a_2 - a_1^2}{9}$, $R = \frac{9a_1a_2 - 27a_3 - 2a_1^3}{54}$, $S = \sqrt[3]{R + \sqrt{Q^3 + R^2}}$ και $T = \sqrt[3]{R - \sqrt{Q^3 + R^2}}$

τότε

$$x_1 = S + T - \frac{1}{3}a_1$$

$$x_2 = -\frac{1}{2}(S+T) - \frac{1}{3}a_1 + \frac{1}{2}i\sqrt{3}(S-T)$$

$$x_3 = -\frac{1}{2}(S+T) - \frac{1}{3}a_1 - \frac{1}{2}i\sqrt{3}(S-T)$$

Ορίζουμε ως διακρίνουσα την $D = Q^3 + R^2$ τότε έχουμε τις ακόλουθες περιπτώσεις:
Εάν $D < 0$ τότε όλες οι ρίζες είναι πραγματικές και άνισες μεταξύ τους.
Εάν $D = 0$ τότε όλες οι ρίζες είναι πραγματικές και τουλάχιστον δύο ίσες μεταξύ τους.
Εάν $D > 0$ τότε μία ρίζα είναι πραγματική και οι άλλες δύο μιγαδικές.

2.2.4 Δίνονται οι ακόλουθες αναδρομικές σχέσεις:

$a_n = n! - a_{n-1}^2$, με $a_0 = 1$ η οποία ορίζεται για n>0

$a_n = \frac{a_{n-1}^2 - 2}{a_{n-1}}$, με $a_0 = 1$ η οποία ορίζεται για n>0

$a_n = \frac{4a_{n-1}^2 + 2}{3a_{n-1}}$, με $a_0 = 1$ η οποία ορίζεται για n>0

$a_n = \frac{2a_{n-1}^4 + a_{n-1}^2}{a_{n-1}^3 + 1}$, με $a_0 = 1$ η οποία ορίζεται για n>0

$a_n = 2a_{n-1}^2 - a_{n-2} + 4$, με $a_0 = 1$ και $a_1 = 2$ η οποία ορίζεται για n>1

$a_n = \frac{a_{n-1} + a_{n-3}}{a_{n-2}}$, με $a_0 = 1$, $a_1 = 2$ και $a_2 = 3$ η οποία ορίζεται για n>2

Να γραφούν προγράμματα για τον υπολογισμό του n-οστου όρου (όπου το n να εισάγεται από το πληκτρολόγιο). Εκτελέστε τα προγράμματα και υπολογίστε τους όρους για n= 3, 5, 10 και 15.

2.2.5 Δίνονται οι ακόλουθες αριθμητικές σειρές:

$$1 + 2 + 3 + ... + n = \frac{n(n+1)}{2}$$

$$1 + 3 + 5 + ... + (2n-1) = n^2$$

$$1^3 + 2^3 + 3^3 + ... + n^3 = \frac{n^2(n+1)^2}{4} = (1 + 2 + 3 + ... + n)^2$$

$$1^4 + 2^4 + 3^4 + ... + n^4 = \frac{n(n+1)(2n+1)(3n^2+3n-1)}{30}$$

$$1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \frac{1}{5} - .... = \ln 2$$

$$1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - .... = \frac{\pi}{4}$$

$$\frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + ... = \frac{\pi^2}{6}$$

$$\frac{1}{1 \bullet 3} + \frac{1}{3 \bullet 5} + \frac{1}{5 \bullet 7} + \frac{1}{7 \bullet 9} + ... = \frac{1}{2}$$

$$\frac{1}{1 \bullet 3} + \frac{1}{2 \bullet 4} + \frac{1}{3 \bullet 5} + \frac{1}{4 \bullet 6} + ... = \frac{3}{4}$$

$$\frac{1}{1^2 \bullet 3^2} + \frac{1}{3^2 \bullet 5^2} + \frac{1}{5^2 \bullet 7^2} + \frac{1}{7^2 \bullet 9^2} + ... = \frac{\pi^2 - 8}{16}$$

Για κάθε μια από αυτές να αναπτύξετε ένα πρόγραμμα το οποίο να υπολογίζει χωριστά τα δύο μέρη της αριθμητικής προόδου και να επαληθεύσετε τις σχέσεις με μερικά παραδείγματα. Το n να δίνεται από το πληκτρολόγιο.

2.2.6 Δίνονται οι ακόλουθες σειρές Taylor:

$$(1+x)^{-1} = 1 - x + x^2 - x^3 + x^4 - .... \qquad \text{για} \qquad -1 < x < 1$$

$$(1+x)^{-2} = 1 - 2x + 3x^2 - 4x^3 + 5x^4 - .... \qquad \text{για} \qquad -1 < x < 1$$

$$(1+x)^{-1/2} = 1 - \frac{1}{2}x + \frac{1 \bullet 3}{2 \bullet 4}x^2 - \frac{1 \bullet 3 \bullet 5}{2 \bullet 4 \bullet 6}x^3 + ... \qquad \text{για} \qquad -1 < x \le 1$$

$$(1+x)^{1/2} = 1 + \frac{1}{2}x - \frac{1}{2 \bullet 4}x^2 + \frac{1 \bullet 3}{2 \bullet 4 \bullet 6}x^3 - .... \qquad \text{για} \qquad -1 < x \le 1$$

$$a^x = e^{x \ln a} = 1 + x \ln a + \frac{(x \ln a)^2}{2!} + \frac{(x \ln a)^3}{3!} + ... \qquad \text{για} \qquad -\infty < x < \infty$$

$$\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + ... \qquad \text{για} \qquad -1 < x \le 1$$

$$\frac{1}{2}\ln\left(\frac{1+x}{1-x}\right) = x + \frac{x^3}{3} + \frac{x^5}{5} + \frac{x^7}{7} + ... \qquad \text{για} \qquad -1 < x < 1$$

$$\frac{\ln(1+x)}{1+x} = x - \left(1+\frac{1}{2}\right)x^2 + \left(1+\frac{1}{2}+\frac{1}{3}\right)x^3 - .... \qquad \text{για} \qquad |\, x\,| < 1$$

Για κάθε μια από αυτές να αναπτύξετε ένα πρόγραμμα το οποίο να υπολογίζει χωριστά τα δύο μέρη των παραστάσεων και να επαληθεύσετε τις σχέσεις με μερικά παραδείγματα. Το x να δίνεται από το πληκτρολόγιο.

2.2.7 Δίνονται οι ακόλουθες σειρές τριγωνομετρικών και υπερβολικών συναρτήσεων:

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + ... \qquad \text{για} \qquad -\infty < x < \infty$$

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + ... \qquad \text{για} \qquad -\infty < x < \infty$$

$$\sin^{-1} x = x + \frac{1}{2}\frac{x^3}{3} + \frac{1\bullet 3}{2\bullet 4}\frac{x^5}{5} + \frac{1\bullet 3\bullet 5}{2\bullet 4\bullet 6}\frac{x^7}{7} + ... \qquad \text{για} \qquad |\, x\,| < 1$$

$$\sinh x = x + \frac{x^3}{3!} + \frac{x^5}{5!} + \frac{x^7}{7!} + ... \qquad \text{για} \qquad -\infty < x < \infty$$

$$\cosh x = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \frac{x^6}{6!} + ... \qquad \text{για} \qquad -\infty < x < \infty$$

Για κάθε μια από αυτές να αναπτύξετε ένα πρόγραμμα το οποίο να υπολογίζει χωριστά τα δύο μέρη των παραστάσεων και να επαληθεύσετε τις σχέσεις με μερικά παραδείγματα. Το x να δίνεται από το πληκτρολόγιο.

2.2.8 Να υπολογιστούν αριθμητικά με τον **κανόνα του τραπεζίου** τα παρακάτω ορισμένα ολοκληρώματα:

$$\int_a^b \sin x dx = -\cos x\Big|_a^b$$

$$\int_a^b \cos(2x) dx = \frac{\sin(2x)}{2}\Big|_a^b$$

$$\int_a^b \frac{dx}{2x+1} = \frac{1}{2}\ln(2x+1)\Big|_a^b$$

$$\int_a^b \frac{xdx}{2x+1} = \frac{x}{2} - \frac{1}{4}\ln(2x+1)\Big|_a^b$$

$$\int_a^b \frac{xdx}{x^2-3^2} = \frac{1}{2}\ln(x^2-3^2)\Big|_a^b$$

$$\int_a^b \frac{dx}{\sqrt{x^2+2^2}} = \ln\left(x+\sqrt{x^2+2^2}\right)\Big|_a^b$$

$$\int_a^b xe^{3x}dx = \frac{e^{3x}}{3}\left(x-\frac{1}{3}\right)\Big|_a^b$$

$$\int_a^b x\ln xdx = \frac{x^2}{2}\left(\ln(x)-\frac{1}{2}\right)\Big|_a^b$$

Να συγκρίνετε την τιμή που βρίσκετε ολοκληρώνοντας αριθμητικά με αυτή που βρίσκετε ολοκληρώνοντας αναλυτικά . Να χρησιμοποιήσετε 10000 βήματα κατά την αριθμητική ολοκλήρωση. Τα όρια της ολοκλήρωσης να δίνονται από το πληκτρολόγιο. Προστατέψτε το πρόγραμμα από λανθασμένα όρια ολοκλήρωσης. Διατηρείστε ακρίβεια 5 δεκαδικών ψηφίων στην εκτύπωση του αποτελέσματος. Εκτελέστε το πρόγραμμα για διάφορα όρια ολοκλήρωσης a και b.

2.2.9 Επαναλάβετε την προηγούμενη άσκηση κάνοντας εφαρμογή του **κανόνα του Simpson** .

2.2.10 Να υπολογιστούν αριθμητικά με τον **κανόνα του τραπεζίου** και **με τον κανόνα του Simpson** τα παρακάτω ορισμένα ολοκλήρωμα:

$$\int_a^b \frac{dx}{\ln x} = \left(\ln(\ln x) + \ln x + \frac{\ln^2 x}{2\bullet 2!} + \frac{\ln^3 x}{3\bullet 3!} + ...\right)\Big|_a^b$$

$$\int_a^b \frac{e^{3x}}{x} = \left(\ln x + \frac{3x}{1\bullet 1!} + \frac{(3x)^2}{2\bullet 2!} + \frac{(3x)^3}{3\bullet 3!} + ...\right)\Big|_a^b$$

Να συγκρίνετε την τιμή που βρίσκετε ολοκληρώνοντας αριθμητικά με αυτή που βρίσκετε από το δεξί μέλος των παραστάσεων . Να χρησιμοποιήσετε 100000 βήματα κατά την αριθμητική ολοκλήρωση. Τα όρια της ολοκλήρωσης να δίνονται από το πληκτρολόγιο. Προστατέψτε το πρόγραμμα από λανθασμένα όρια ολοκλήρωσης. Διατηρείστε ακρίβεια 5 δεκαδικών ψηφίων στην εκτύπωση του αποτελέσματος. Εκτελέστε το πρόγραμμα για διάφορα όρια ολοκλήρωσης a και b.

Κεφάλαιο ΙΙΙ

Προβλήματα που αφορούν δείκτες και πίνακες.

Στο παρόν κεφάλαιο παρουσιάζονται προβλήματα τα οποία αφορούν δείκτες και πίνακες. Παρουσιάζονται ασκήσεις οι οποίες αναφέρονται σε:

- **Δείκτες**
- **Στον Τελεστή Διεύθυνσης &**
- **Δήλωση Δεικτών**
- **Πίνακες**
- **Αρχικοποίηση Πινάκων**
- **Πίνακες χαρακτήρων και Συμβολοσειρών**
- **Πίνακες πολλών διαστάσεων**
- **Προβλήματα με τυχαίους αριθμούς**

3.1 Λυμένα Προβλήματα.

3.1.1 Γράψτε ένα πρόγραμμα το οποίο να τυπώνει το περιεχόμενο και την διεύθυνση ενός χαρακτήρα , ενός ακεραίου και ενός float αριθμού.

Μια πιθανή λύση είναι:

```
#include<stdio.h>

int main(void)
{
  char a;
  int i;
  float x;

  a='W';
  i=15;
  x=3.14159;

  printf("a: periexomeno=%c  dieuthinsi=%p\n",a,&a);
  printf("i: periexomeno=%d  dieuthinsi=%p\n",i,&i);
  printf("x: periexomeno=%f  dieuthinsi=%p\n",x,&x);
  return 0;
}
```

Η εκτέλεση του παραπάνω προγράμματος δίνει:

```
[student1@pc244 kef4]$ a.out
a: periexomeno=W  dieuthinsi=0xbffff9e7
i: periexomeno=15  dieuthinsi=0xbffff9e0
x: periexomeno=3.141590  dieuthinsi=0xbffff9dc
[student1@pc244 kef4]$
```

Προσοχή !!! Οι τιμές των διευθύνσεων γενικά είναι διαφορετικές από υπολογιστή σε υπολογιστή και από τη χρονική στιγμή εκτέλεσης του προγράμματος. Αυτό συμβαίνει διότι ο τρόπος διαχειρισμού της μνήμης σε ένα υπολογιστή είναι τυχαίος (**Random Access Memory** - RAM).

Σχόλια: Προσέξτε τον τρόπο εκτύπωσης των διευθύνσεων με τον χαρακτήρα μετατροπής **%p**. Το αποτέλεσμα είναι πάντοτε σε δεκαεξαδική μορφή. Οι αριθμοί των διευθύνσεων έχουν μήκος 32 bit.

3.1.2 Γράψτε ένα πρόγραμμα στο οποίο να ορίσετε μια **ακέραια** μεταβλητή **a** και έναν **ακέραιο** δείκτη **point_a**. Δώστε στην a την τιμή 15 και αντιστοιχίστε στον δείκτη point_a την διεύθυνση της μεταβλητής a. Τυπώστε τη διεύθυνση και το περιεχόμενο τόσο της μεταβλητής a όσο και του δείκτη point_a. Τυπώστε την τιμή στην οποία δείχνει ο *point_a. Εκτελέστε το πρόγραμμα. Κάντε ένα απλό διάγραμμα της μνήμης του Η/Υ και σχεδιάστε την διασύνδεση των a και point_a.

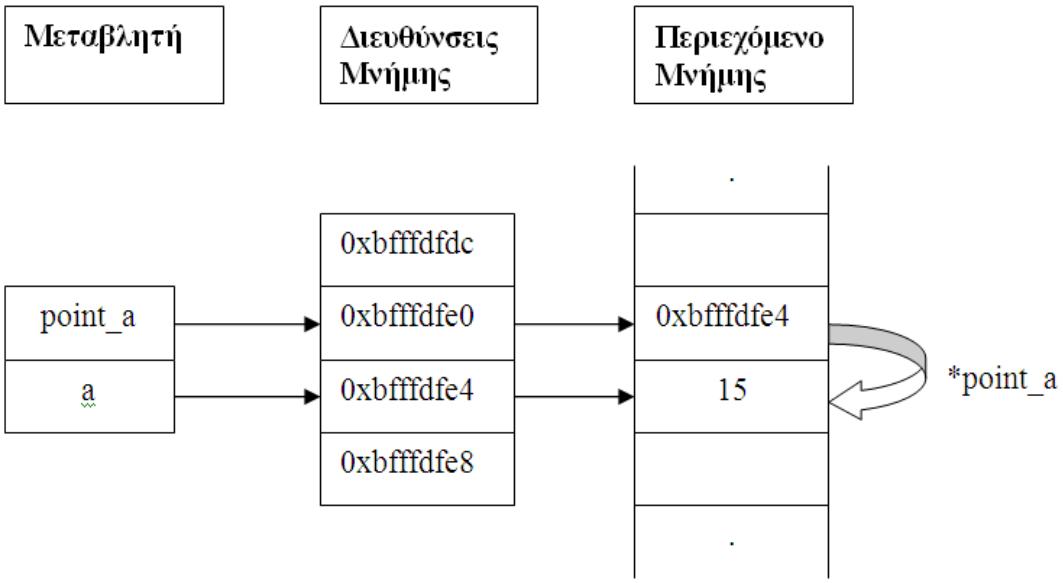
Μια πιθανή λύση είναι

```c
#include<stdio.h>

int main(void)
{
  int a, *point_a;

  a=15;          /* antistoixizoume ston akeraio a to 15 */
  point_a=&a;    /* antistoixizoume ston deikti point_a tin dieuthinsi tou a */

  printf("Dieuthinsi  tou akeraiou a : %p\n",&a);
  printf("Periexomeno tou akeraiou a : %d\n",a);
  printf("Dieuthinsi  tou deikti point_a : %p\n",&point_a);
  printf("Periexomeno tou deikti point_a : %p\n",point_a);
  printf("O deiktis *point_a deixnei sthn timi: %d\n",*point_a);
  return 0;
}
```

Η εκτέλεση του προγράμματος δίνει:

```
[student1@pc244 kef4]$ a.out
Dieuthinsi  tou akeraiou a : 0xbfffdfe4
Periexomeno tou akeraiou a : 15
Dieuthinsi  tou deikti point_a : 0xbfffdfe0
Periexomeno tou deikti point_a : 0xbfffdfe4
O deiktis *point_a deixnei sthn timi: 15
[student1@pc244 kef4]$
```

Διάγραμμα Μνήμης Η/Υ:



3.1.3 Γράψτε ένα πρόγραμμα στο οποίο να ορίσετε μια μεταβλητή **a** τύπου **char** και έναν δείκτη **point_a** επίσης τύπου **char**. Εισάγετε στην a τον χαρακτήρα 'W' και αντιστοιχίστε στον δείκτη point_a την διεύθυνση της μεταβλητής a. Τυπώστε τη διεύθυνση και το περιεχόμενο τόσο της μεταβλητής a όσο και του δείκτη point_a. Τυπώστε την τιμή στην οποία δείχνει ο *point_a. Στη συνέχεια χρησιμοποιώντας τον δείκτη point_a αλλάξτε την τιμή της μεταβλητής a στον χαρακτήρα 'R'. Τυπώστε εκ νέου τις μεταβλητές. Εκτελέστε το πρόγραμμα.

Μια πιθανή λύση είναι

```
#include<stdio.h>

int main(void)
{
  char a, *point_a;

  a='W';         /* antistoixizoume sto a to 'W' */
  point_a=&a;    /* antistoixizoume ston deikti point_a tin dieuthinsi tou a */

  printf("Dieuthinsi kai periexomeno tou a : %p %c\n",&a,a);
  printf("Dieuthinsi kai periexomeno tou deikti point_a : %p %p\n",&point_a,point_a);
  printf("O deiktis *point_a deixnei sthn timi : %c\n\n",*point_a);

  *point_a='R';  /* antistoixizoume ston *point_a to 'R' */

  printf("Dieuthinsi kai periexomeno tou a : %p %c\n",&a,a);
  printf("Dieuthinsi kai periexomeno tou deikti point_a : %p %p\n",&point_a,point_a);
  printf("O deiktis *point_a deixnei sthn timi : %c\n",*point_a);

  return 0;
}
```

Η εκτέλεση του προγράμματος δίνει:

```
[student1@pc244 kef4]$ a.out
Dieuthinsi kai periexomeno tou a : 0xbfffe967 W
Dieuthinsi kai periexomeno tou deikti point_a : 0xbfffe960 0xbfffe967
O deiktis *point_a deixnei sthn timi : W

Dieuthinsi kai periexomeno tou a : 0xbfffe967 R
Dieuthinsi kai periexomeno tou deikti point_a : 0xbfffe960 0xbfffe967
O deiktis *point_a deixnei sthn timi : R
[student1@pc244 kef4]$
```

3.1.4 Γράψτε ένα πρόγραμμα στο οποίο να ορίσετε έναν πίνακα με τους float αριθμούς 0.0, 1.0, 2.0,...9.0. Στη συνέχεια να υπολογίσετε τα τετράγωνα και τις τετραγωνικές ρίζες κάθε στοιχείου του πίνακα. Να αποθηκεύσετε τις τιμές σε δύο νέους πίνακες. Να τυπώσετε σε μορφή πίνακα τις τιμές των πινάκων που δημιουργήσατε.

Μια πιθανή λύση είναι η ακόλουθη:

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

int main(void)
{
  int i;
  float a[10], a_sqr[10], a_sqrt[10];

  for(i=0; i<10; i++)
    a[i]=(float)i;

  for(i=0;i<10; i++){
    a_sqr[i]=pow(a[i],2);
    a_sqrt[i]=sqrt(a[i]);
  }

  printf("Number    Square   Square Root\n");
  for(i=0; i<10; i++)
    printf("%4.1f   %8.2f   %8.2f\n",a[i],a_sqr[i],a_sqrt[i]);

  return 0;
}
```

Η εκτέλεση του παραπάνω προγράμματος είναι η ακόλουθη:

```
[student1@pc244 kef4]$ a.out
Number    Square   Square Root
 0.0       0.00       0.00
 1.0       1.00       1.00
 2.0       4.00       1.41
 3.0       9.00       1.73
 4.0      16.00       2.00
 5.0      25.00       2.24
 6.0      36.00       2.45
 7.0      49.00       2.65
 8.0      64.00       2.83
 9.0      81.00       3.00
[student1@pc244 kef4]$
```

3.1.5 Γράψτε ένα πρόγραμμα στο οποίο να ορίσετε τον πίνακα a[10] με τους αριθμούς 1,2,3,...10. Ορίστε έναν δείκτη *pa ο οποίος να δείχνει στον πίνακα a. Στη συνέχεια να τυπώσετε για κάθε στοιχείο του πίνακα a[i], την τιμή του, την διεύθυνση &a[i] καθώς και τις μεταβλητές pa+i και *(pa+i). Εκτελέστε το πρόγραμμα και παρατηρείστε προσεκτικά το αποτέλεσμα.

Μια πιθανή λύση είναι η ακόλουθη:

```
#include<stdio.h>

int main(void)
{
  int i, a[10]={1,2,3,4,5,6,7,8,9,10};
  int *pa;

  pa=a;
  printf(" i  a[i]     &a[i]         pa+i     *(pa+i) \n");
    for(i=0;i<10;i++)
      printf("%2d   %2d   %p    %p    %2d\n", i, a[i], &a[i], pa+i, *(pa+i));

  return 0;
}
```

Η εκτέλεση του προγράμματος δίνει:

```
[student1@pc244 kef4]$ a.out
 i  a[i]     &a[i]         pa+i     *(pa+i)
 0    1   0xbfffe620    0xbfffe620     1
 1    2   0xbfffe624    0xbfffe624     2
 2    3   0xbfffe628    0xbfffe628     3
 3    4   0xbfffe62c    0xbfffe62c     4
 4    5   0xbfffe630    0xbfffe630     5
 5    6   0xbfffe634    0xbfffe634     6
 6    7   0xbfffe638    0xbfffe638     7
 7    8   0xbfffe63c    0xbfffe63c     8
 8    9   0xbfffe640    0xbfffe640     9
 9   10   0xbfffe644    0xbfffe644    10
[student1@pc244 kef4]$
```

Σχόλιο: Από το παραπάνω πρόβλημα φαίνεται καθαρά η σχέση μεταξύ δεικτών και πινάκων. Η σχέση δεικτών και πινάκων όμως είναι ακόμη πιο στενή. Μια αναφορά στο i-στο στοιχείο ενός πίνακα a[i] μπορεί να γραφεί ως *(a+i). Στην πραγματικότητα στη C το a[i] μετατρέπεται αμέσως σε *(a+i). Οι δύο αυτές μορφές είναι **ισοδύναμες**. Επίσης ισοδύναμες είναι οι μορφές &a[i] και a+i. Αυτό προκύπτει εάν στα a[i] και *(a+i) εφαρμόσουμε τον τελεστή διεύθυνσης &. Ας δούμε το πρόβλημα χωρίς να ορίσουμε τον δείκτη *pa.

```c
#include<stdio.h>

int main(void)
{
  int i, a[10]={1,2,3,4,5,6,7,8,9,10};


  printf(" i  a[i]     &a[i]         a+i      *(a+i) \n");
    for(i=0;i<10;i++)
      printf("%2d   %2d   %p    %p    %2d\n", i, a[i], &a[i], a+i, *(a+i));

  return 0;
}
```

Αντί του δείκτη *pa χρησιμοποιούμε κατ' ευθείαν τον πίνακα a. Έτσι αντί των μεταβλητών pa+i και *(pa+i) τυπώνουμε κατ' ευθείαν τα a+i και *(a+i). Το αποτέλεσμα είναι το ίδιο. Η διαφορά στις διευθύνσεις που βλέπετε οφείλετε στην τυχαία πρόσβαση της μνήμης του υπολογιστή την οποία ήδη εξηγήσαμε.

```
[student1@pc244 kef4]$ a.out
 i  a[i]     &a[i]         a+i      *(a+i)
 0    1   0xbffff920    0xbffff920     1
 1    2   0xbffff924    0xbffff924     2
 2    3   0xbffff928    0xbffff928     3
 3    4   0xbffff92c    0xbffff92c     4
 4    5   0xbffff930    0xbffff930     5
 5    6   0xbffff934    0xbffff934     6
 6    7   0xbffff938    0xbffff938     7
 7    8   0xbffff93c    0xbffff93c     8
 8    9   0xbffff940    0xbffff940     9
 9   10   0xbffff944    0xbffff944    10
[student1@pc244 kef4]$
```

3.1.6 Γράψτε ένα πρόγραμμα στο οποίο να ορίσετε τον πίνακα χαρακτήρων str[7] με τους χαρακτήρες ‘H’, ‘e’, ‘l’, ‘l’, ‘o’, ‘!’. Τυπώστε ένα-ένα τα στοιχεία του. Στη συνέχεια τυπώστε την λέξη που σχηματίζουν τα στοιχεία του τυπώνοντάς τα ένα-ένα και όλα μαζί ως συμβολοσειρά.

Μια πιθανή λύση είναι η ακόλουθη:

```
#include<stdio.h>

int main(void)
{
  int i;
  char str[7]={'H','e','l','l','o','!','\0'};

  for(i=0;i<7;i++)
    printf("O pinakas str[%d] periexei to: %c\n",i,str[i]);

    printf("Ektiposi olon ton stoixeion mazi (Methodos I):\n");
    for(i=0;str[i]!='\0' && i<7;i++)
      printf("%c",str[i]);       /* ektiposi ana xaraktira */

    printf("\nEktiposi olon ton stoixeion mazi (Methodos II):\n");
    printf("%s\n",str);          /* ektiposi os string */

    return 0;
}
```

Η εκτέλεση του προγράμματος δίνει:

```
[student1@pc244 kef4]$ a.out
O pinakas str[0] periexei to: H
O pinakas str[1] periexei to: e
O pinakas str[2] periexei to: l
O pinakas str[3] periexei to: l
O pinakas str[4] periexei to: o
O pinakas str[5] periexei to: !
O pinakas str[6] periexei to:
Ektiposi olon ton stoixeion mazi (Methodos I):
Hello!
Ektiposi olon ton stoixeion mazi (Methodos II):
Hello!
[student1@pc244 kef4]$
```

3.1.7 Γράψτε ένα πρόγραμμα στο οποίο να ορίσετε τον δυσδιάστατο ακέραιο πίνακα

int array_two_dim[3][4]={1,2,3,4,5,6,7,8,9,10,11,12}

Εκτυπώστε τον πίνακα κατά γραμμές και κατά στήλες.

Μια πιθανή λύση είναι η ακόλουθη:

```
#include<stdio.h>

int main(void)
{
  int array_two_dim[3][4]={1,2,3,4,5,6,7,8,9,10,11,12};
  int i,j;

  for(i=0;i<3;i++){
    printf("\n");
    for(j=0;j<4;j++){
      printf(" %2d",array_two_dim[i][j]);
    }
  }
  printf("\n");
  return 0;
}
```

Η εκτέλεση του προγράμματος δίνει:

```
[student1@pc244 kef4]$ a.out

 1  2  3  4
 5  6  7  8
 9 10 11 12
[student1@pc244 kef4]$
```

Σχόλιο: Για να τυπώσουμε ένα πολυδιάστατο πίνακα χρησιμοποιούμε δύο βρόγχους τον ένα μέσα στον άλλο. Ο κάθε βρόγχος τρέχει πάνω στους δείκτες του πολυδιάστατου πίνακα. Προσέξτε στο παράδειγμα τις εντολές for.

3.1.8 Δίνεται ο πίνακας

int num_int[10]={37,1,8,-14,52,-6,102,17,-23,27}

Να γράψετε ένα πρόγραμμα το οποίο να ανακατατάσσει τα στοιχεία του παραπάνω πίνακα κατά φθίνουσα σειρά. Να τυπωθεί ο πίνακας μετά την ανακατάταξη.

Μια πιθανή λύση είναι η ακόλουθη:

```
#include<stdio.h>

int main(void)
{
  int num_int[10]={37,1,8,-14,52,-6,102,17,-23,27};
  int i,j,temp;

  for(i=0;i<10;i++){
    /* find the largest of all remaining elements */
    for(j=i+1;j<10;j++){
      if(num_int[j]>num_int[i]){
        /* interchange the two elements */
        temp = num_int[i];
        num_int[i]=num_int[j];
        num_int[j]=temp;
      }
    }
  }

  for(i=0;i<10;i++)  /* display the reordered array */
    printf("num_int[%d] = %3d\n",i,num_int[i]);

  return 0;
}
```

Η εκτέλεση του προγράμματος δίνει:

```
[student1@pc244 kef4]$ a.out
num_int[0] = 102
num_int[1] =  52
num_int[2] =  37
num_int[3] =  27
num_int[4] =  17
num_int[5] =   8
num_int[6] =   1
num_int[7] =  -6
num_int[8] = -14
num_int[9] = -23
[student1@pc244 kef4]$
```

3.1.9 Δίνονται οι πίνακες

$$A=\begin{bmatrix} 32 & -26 & 43 & 14 \\ 21 & -6 & -7 & 12 \\ -9 & -31 & 42 & 11 \end{bmatrix} \text{ και } B=\begin{bmatrix} 10 & 7 & 12 & -5 \\ 24 & -18 & 33 & 4 \\ 67 & -8 & 51 & 44 \end{bmatrix}$$

Να γραφεί ένα πρόγραμμα το οποίο να υπολογίζει τον πίνακα $C_{ij} = A_{ij} + B_{ij}$. Να τυπωθεί σε μορφή γραμμών-στηλών ο πίνακας C.

Μια πιθανή λύση είναι η ακόλουθη:

```
#include<stdio.h>

int main(void)
{
  int a[3][4]={32,-26,43,14,21,-6,-7,12,-9,-31,42,11};
  int b[3][4]={10,7,12,-5,24,-18,33,4,67,-8,51,44};
  int c[3][4];
  int i,j;

  for(i=0;i<3;i++){     /* ypologismos tou c[3][4] */
    for(j=0;j<4;j++){
      c[i][j]=a[i][j]+b[i][j];
    }
  }

  printf("O pinakas c[3][4] einai:\n");
  for(i=0;i<3;i++){     /* ektyposi tou c[3][4] */
    printf("\n");
    for(j=0;j<4;j++){
      printf(" %3d ",c[i][j]);
    }
  }
  printf("\n");

  return 0;
}
```

Η εκτέλεση του παραπάνω προγράμματος είναι:

```
[student1@pc244 kef4]$ a.out
O pinakas c[3][4] einai:

  42  -19   55    9
  45  -24   26   16
  58  -39   93   55
[student1@pc244 kef4]$
```

3.1.10 Να γραφεί ένα πρόγραμμα το οποίο να παράγει δέκα **τυχαίους αριθμούς** τύπου float από το 0 έως το 1. Οι αριθμοί αυτοί να αποθηκευτούν σε ένα πίνακα με δέκα στοιχεία. Να εκτυπωθεί ο πίνακας.

Η ανάπτυξη του προγράμματος θα βασιστεί στη χρήση της συνάρτησης **rand()** η οποία διατίθεται στην έτοιμη βιβλιοθήκη **<stdlib.h>** της C. Στην βιβλιοθήκη **<stdlib.h>** ορίζονται τα ακόλουθα:

#define RAND_MAX 2147483647

int rand(void)

void srand(unsigned int seed)

Ας δούμε αναλυτικά τα παραπάνω:

- Η συνάρτηση **rand()** επιστρέφει έναν ψευδοτυχαίο ακέραιο αριθμό από το 0 έως το RAND_MAX. Σημειώνουμε εδώ πως η τιμή του RAND_MAX είναι η μεγαλύτερη θετική τιμή που παίρνει ένας ακέραιος αριθμός στο σύστημά μας.
- Η συνάρτηση **srand(seed)** χρησιμοποιεί το seed ως φύτρο για νέα ακολουθία ψευδοτυχαίων αριθμών. Εάν παραλειφθεί η συνάρτηση το αρχικό φύτρο είναι πάντοτε το 1. Η srand() εκτελείται πάντοτε μία φορά στην αρχή του προγράμματος και αρχικοποιεί τη γεννήτρια των τυχαίων αριθμών. Εάν θέλουμε να ξανατρέξουμε τον πρόγραμμά μας και να δημιουργήσουμε νέους τυχαίους αριθμούς πρέπει να αλλάξουμε τον ακέραιο seed.

Προσοχή πρέπει να δοθεί στο ότι η συνάρτηση rand() επιστρέφει ένα ακέραιο αριθμό από 0 έως 2147483647. Συνήθως στα προγράμματά μας χρειαζόμαστε αριθμούς τύπου float ή int σε ένα συγκεκριμένο διάστημα. Κατά συνέπεια πρέπει να επεξεργαστούμε την τιμή που μας επιστρέφει η rand(). Ο πιο απλός τρόπος είναι πρώτα από την rand() να δημιουργήσουμε έναν αριθμό τύπου float μεταξύ του 0 και του 1. Αυτόν τον αριθμό μπορούμε εύκολα στη συνέχεια να τον χρησιμοποιήσουμε για να κατασκευάσουμε αριθμούς float ή int τους οποίους χρειαζόμαστε σε ένα συγκεκριμένο διάστημα. Θα πρέπει εδώ να δώσετε προσοχή στους μετασχηματισμούς αριθμών float σε int με το πρόθεμα μετατροπής (float) και σε αριθμούς int σε float με το πρόθεμα μετατροπής (int).

Στη χρήση τυχαίων αριθμών βασίζονται πολλές σύγχρονες υπολογιστικές τεχνικές στη φυσική. Πολλά από τα προβλήματα που απαντώνται στη σύγχρονη φυσική είναι τόσο πολύπλοκα ώστε οι αναλυτικές τους λύσεις να είναι σχεδόν αδύνατες. Σε αυτές τις περιπτώσεις τα προβλήματα εξομοιώνονται στον υπολογιστή. Για παράδειγμα στην Πυρηνική Φυσική και ιδίως στην Φυσική Υψηλών Ενεργειών υπάρχει η ανάγκη εξομοίωσης στον υπολογιστή των διαφόρων πειραμάτων. Τα συγκεκριμένα προγράμματα εξομοίωσης χρησιμοποιούν τεχνικές Monte Carlo οι οποίες βασίζονται στη χρήση τυχαίων αριθμών.

Ας δούμε τώρα μια πιθανή λύση του προβλήματος

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

#define SEED 12345

int main(void)
{
  int i;
  float ran_float[10];

  /* Arxikopoihsi ths genitrias tixaion arithmon */
  srand(SEED);

  /* Dimiourgia kai apothikeusi deka tixaion arithmon */
  for(i=0;i<10;i++)
    ran_float[i]=(float)rand()/(float)RAND_MAX;

  /* Ektiposi deka tixaion arithmon */
  for(i=0;i<10;i++)
    printf("ran_float[%d] = %f\n",i,ran_float[i]);

  return 0;
}
```

Εάν εκτελέσουμε το πρόγραμμα έχουμε τα ακόλουθα.

```
[student1@pc244 kef4]$ a.out
ran_float[0] = 0.178395
ran_float[1] = 0.399677
ran_float[2] = 0.166599
ran_float[3] = 0.212122
ran_float[4] = 0.061936
ran_float[5] = 0.054150
ran_float[6] = 0.275665
ran_float[7] = 0.047757
ran_float[8] = 0.321033
ran_float[9] = 0.272734
[student1@pc244 kef4]$
```

3.1.11 Να γραφεί ένα πρόγραμμα το οποίο να παράγει δέκα **τυχαίους αριθμούς** τύπου int από το 100 έως το 200. Οι αριθμοί αυτοί να αποθηκευτούν σε ένα πίνακα με δέκα στοιχεία. Να εκτυπωθεί ο πίνακας.

Μια πιθανή λύση είναι η ακόλουθη:

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

#define SEED 9876

int main(void)
{
  int i,ran_int[10];
  float x;

  /* Arxikopoihsi ths genitrias tixaion arithmon */
  srand(SEED);

  /* Dimiourgia deka tixaion int arithmon apo to 100 eos to 200 */
  for(i=0;i<10;i++){
    /* Dimiourgia enos tixaiou float arithmou apo to 0 eos to 1 */
    x=(float)rand()/(float)RAND_MAX;
    ran_int[i]=100+(int)(100*x);
  }

  /* Ektiposi ton deka tixaion arithmon */
  for(i=0;i<10;i++)
    printf("ran_int[%d] = %d\n",i,ran_int[i]);

  return 0;
}
```

Η εκτέλεση του παραπάνω προγράμματος δίνει:

```
[student1@pc244 kef4]$ a.out
ran_int[0] = 183
ran_int[1] = 151
ran_int[2] = 143
ran_int[3] = 186
ran_int[4] = 194
ran_int[5] = 133
ran_int[6] = 141
ran_int[7] = 163
ran_int[8] = 155
ran_int[9] = 171
[student1@pc244 kef4]$
```

3.1.12 Να γραφεί ένα πρόγραμμα στο οποίο να εισάγονται έως και 100 αριθμοί τύπου float. Οι αριθμοί αυτοί να αποθηκευθούν σε έναν πίνακα. Στη συνέχεια να γράψετε μια συνάρτηση η οποία να υπολογίζει τον μέσο των στοιχείων του πίνακα. Να χρησιμοποιήσετε αυτή την συνάρτηση για να υπολογίσετε τον μέσο τον αριθμών που εισάγατε.

Μια πιθανή λύση είναι η ακόλουθη:

```
#include<stdio.h>

#define MAX_NUMBERS 100  /* Megistos arithmos stoixeion */

float average(int a, float list[]);

int main(void)
{
  float x[MAX_NUMBERS];
  int n,i;

  /* Eisagogi timis tou n */
  printf("Posous arithmous tha eisagete? ");
  scanf("%d",&n);
  printf("\n");

  /* Eisagogi arithmon */
  for(i=0;i<n;i++){
    printf("n = %d   x = ",i+1);
    scanf("%f",&x[i]);
  }

  printf("\nO mesos ton arithmon pou eisagate einai : %f\n",average(n,x));
  return 0;
}


float average(int a, float list[])
{
  int i;
  float sum=0.;

  /* Ypologismos tou athrismatos ton stoixeion tou pinaka list[] */
  for(i=0;i<a;i++)
    sum+=list[i];

  return sum/a;      /* Epistrofi toy mesou */
}
```

Η εκτέλεση του προγράμματος εικονίζεται παρακάτω:

```
[student1@pc244 kef4]$ a.out
Posous arithmous tha eisagete? 7

n = 1   x = 4.56
n = 2   x = 1.23
n = 3   x = 6.74
n = 4   x = 2.56
n = 5   x = 3.68
n = 6   x = 0.23
n = 7   x = 5.41

O mesos ton arithmon pou eisagate einai : 3.487143
[student1@pc244 kef4]$
```

3.1.13 Να γραφεί ένα πρόγραμμα στο οποίο να εισάγονται από το πληκτρολόγιο μια-μια οι λέξεις τις πρότασης “The capital of Greece is Athens.”. Οι λέξεις να αποθηκευθούν σε έναν πίνακα. Να χρησιμοποιήσετε τον πίνακα για να τυπώσετε την πρόταση.

Υπόδειξη: Ένας πίνακας από συμβολοσειρές είναι ένας πίνακας τύπου char δύο διαστάσεων. Η πρώτη διάσταση αντιστοιχεί στον αριθμό των συμβολοσειρών και η δεύτερη στον αριθμό των χαρακτήρων ανά συμβολοσειρά. Προσέξετε ώστε να ορίσετε έναν πίνακα δύο διαστάσεων ικανό να χωρέσει τις λέξεις σας.

Μια πιθανή λύση είναι η ακόλουθη:

```
#include<stdio.h>

int main(void)
{
  char x[6][12];
  int i;

  /* eisagogi ton lexeon tis protasis */
  printf("Eisagete mia mia tis lexeis thw protasis.\n\n");
  for(i=0;i<6;i++){
    printf("Lexi %d : ",i+1);
    scanf("%s",&x[i]);
  }

  printf("\nH protasi pou sximatizoun oi lexeis pou eisagate einai:\n");

  /* Ektiposi ton lexeon tis protasis */
  for(i=0;i<6;i++)
    printf("%s ",x[i]);

  printf("\n");
  return 0;
}
```

Η εκτέλεση του προγράμματος εικονίζεται παρακάτω:

```
[student1@pc244 kef4]$ a.out
Eisagete mia mia tis lexeis thw protasis.

Lexi 1 : The
Lexi 2 : capital
Lexi 3 : of
Lexi 4 : Greece
Lexi 5 : is
Lexi 6 : Athens.

H protasi pou sximatizoun oi lexeis pou eisagate einai:
The capital of Greece is Athens.
[student1@pc244 kef4]$
```

3.2 Προβλήματα.

3.2.1 Δίνεται ο πίνακας

int list_num[]={-7, 18, 5, 9, -11, 0, 23, 13, -9, -1}

Να γράψετε ένα πρόγραμμα το οποίο να υπολογίζει το μεγαλύτερο και το μικρότερο αλγεβρικά στοιχείο καθώς και τη θέση τους μέσα στον πίνακα. Τυπώστε τα αποτελέσματα.

3.2.2 Λύστε τις ασκήσεις 2.2.9 και 2.2.10 τοποθετώντας τις τιμές των πειραματικών μετρήσεων πρώτα σε κατάλληλους πίνακες.

3.2.3 Γράψτε ένα πρόγραμμα στο οποίο να εισάγετε από το πληκτρολόγιο έως και 20 αριθμούς τύπου float. Αποθηκεύστε τους σε έναν πίνακα. Υπολογίστε το άθροισμα των στοιχείων, τον μέσο, την απόκλιση κάθε στοιχείου από τον μέσο και την τυπική απόκλιση. Εκτελέστε το πρόγραμμα και τυπώστε τα αποτελέσματα για τα εξής δεδομένα: 27.5, 13.4, 53.8, 29.2, 74.2, 87.0, 39.9, 47.7, 8.1 και 63.2.

Δίνονται: Ο μέσος $\bar{x} = \frac{\sum_{i=1}^{m} x_i}{m}$, η απόκλιση από τον μέσο $d_i = x_i - \bar{x}$ και η τυπική απόκλιση $s = \sqrt{\frac{\left(d_1^2 + d_2^2 + \ldots + d_m^2\right)}{m(m-1)}}$

3.2.4 Γράψτε μια συνάρτηση η οποία να δέχεται ως ορίσματα έναν πίνακα και το μέγεθός του και να ανακατατάσσει τα στοιχεία του κατ' αύξουσα σειρά. Στη συνέχεια να γράψετε ένα πρόγραμμα στο οποίο να εισάγετε μια λίστα έως και 100 αριθμών από το πληκτρολόγιο. Οι αριθμοί αυτοί να αποθηκευθούν σε ένα πίνακα και στη συνέχεια να ανακαταταχθούν κατ' αύξουσα σειρά καλώντας την συνάρτηση που γράψατε αρχικά. Στο τέλος τυπώστε τους αριθμούς για να ελέγξετε το πρόγραμμά σας.

3.2.5 Γράψτε ένα πρόγραμμα το οποίο να παράγει δέκα τυχαίους αριθμούς τύπου float από το 30 έως το 40. Αποθηκεύστε τους σε έναν πίνακα. Στη συνέχεια υπολογίστε το άθροισμα των στοιχείων, τον μέσο, την απόκλιση κάθε στοιχείου από τον μέσο και την τυπική απόκλιση. Εκτελέστε το πρόγραμμα και τυπώστε αναλυτικά τα αποτελέσματα.

3.2.6 Γράψτε ένα πρόγραμμα το οποίο να παράγει 5 διαφορετικούς τυχαίους ακεραίους αριθμούς στο διάστημα 1 έως 45 και έναν τυχαίο ακέραιο αριθμό στο διάστημα 1 έως 20. Τυπώστε το αποτέλεσμα.

3.2.7 Δίνονται οι πίνακες

$$A=\begin{bmatrix}1 & 2 & 3 & 4\\ 5 & 6 & 7 & 8\\ 9 & 10 & 11 & 12\end{bmatrix} \text{ και } B=\begin{bmatrix}10 & 11 & 12 & 13\\ 14 & 15 & 16 & 17\\ 18 & 19 & 20 & 21\end{bmatrix}$$

Να γραφεί ένα πρόγραμμα το οποίο να υπολογίζει τον πίνακα $C_{ij} = A_{ij} - B_{ij}$. Να τυπωθεί σε μορφή γραμμών-στηλών ο πίνακας C.

3.2.8 Δίνεται ο πίνακας

$$A=\begin{bmatrix}1 & 2 & 3 & 4\\ 5 & 6 & 7 & 8\\ 9 & 10 & 11 & 12\end{bmatrix}$$

Να γραφεί ένα πρόγραμμα το οποίο να υπολογίζει τους πίνακες B και C όπου $B_{ij} = A_{ij}^2$ και $C_{ij} = \sqrt{A_{ij}}$. Να τυπωθούν σε μορφή γραμμών-στηλών οι πίνακες B και C. Διατηρείστε ακρίβεια δύο δεκαδικών ψηφίων.

3.2.9 Δίνονται οι πίνακες

$$A=\begin{bmatrix}2.7 & 7.3 & 42.1 & 35\\ 5.6 & 4.1 & 8.2 & 5.5\\ 9 & 23 & 67.1 & 3.1\end{bmatrix} \text{ και } B=\begin{bmatrix}2.4\\ 7.1\\ 3.4\\ 9.2\end{bmatrix}$$

Να γραφεί ένα πρόγραμμα το οποίο να υπολογίζει το γινόμενό τους, δηλαδή τον πίνακα $C_i = \sum_{j=1}^{4} A_{ij} * B_j$. Να τυπωθεί σε μορφή γραμμών-στηλών ο πίνακας C. Διατηρείστε ακρίβεια δύο δεκαδικών ψηφίων.

3.2.10 Να γραφεί ένα πρόγραμμα το οποίο να υπολογίζει το γινόμενό ενός πίνακα A διαστάσεων 2x3 και ενός πίνακα B μιας στήλης τριών στοιχείων. Τα στοιχεία των πινάκων να εισάγονται από το πληκτρολόγιο. Να τυπωθεί σε μορφή γραμμών-στηλών το γινόμενό τους. Διατηρείστε ακρίβεια δύο δεκαδικών ψηφίων. Ελέγξτε το πρόγραμμά σας με τους εξής πίνακες:

$$A = \begin{bmatrix} 4 & 8 & 2 \\ 9 & 5 & 1 \end{bmatrix} \text{ και } B = \begin{bmatrix} 7 \\ 3 \\ 6 \end{bmatrix}$$

3.2.11 Να γραφεί ένα πρόγραμμα στο οποίο να εισάγονται από το πληκτρολόγιο δέκα διαφορετικές λέξεις με μέγιστο αριθμό χαρακτήρων 12. Οι λέξεις να αποθηκευθούν σε έναν κατάλληλο πίνακα. Στη συνέχεια να ανακατατάξετε τις λέξεις στον πίνακα κατά αλφαβητική σειρά. Να τυπώσετε τις λέξεις κατά αλφαβητική σειρά χρησιμοποιώντας τον πίνακα.

Υπόδειξη: Να χρησιμοποιήσετε τις συναρτήσεις strcmp() και strcpy() της βιβλιοθήκης <string.h>.

3.2.12 Δίνονται τα ακόλουθα κράτη και πρωτεύουσες:

Canada	Ottawa
England	London
Greece	Athens
France	Paris
Italy	Rome
USA	Washington
Russia	Moscow
Spain	Madrid

Να γραφεί ένα πρόγραμμα στο οποίο να εισάγετε από το πληκτρολόγιο το όνομα του κράτους και να τυπώνετε το όνομα της πρωτεύουσας και το αντίθετο.

Υπόδειξη: Να χρησιμοποιήσετε τη συνάρτηση strcmp() της βιβλιοθήκης <string.h>.

Κεφάλαιο IV

Προβλήματα που αφορούν δομές και ενώσεις.

Στο παρόν κεφάλαιο παρουσιάζονται προβλήματα τα οποία αφορούν δομές και ενώσεις. Παρουσιάζονται ασκήσεις οι οποίες αναφέρονται σε:

- **Δομές**
- **Αρχικοποίηση Δομών**
- **Ένθετες Δομές**
- **Δομές και δείκτες**
- **Δομές και συναρτήσεις**
- **Πίνακες Δομών**
- **Βάσεις Δεδομένων**
- **Ενώσεις**

4.1 Λυμένα Προβλήματα.

4.1.1 Γράψτε ένα πρόγραμμα στο οποίο να εισάγετε τις συντεταγμένες ενός σημείου στο επίπεδο από το πληκτρολόγιο. Να τις αποθηκεύσετε σε μια κατάλληλη δομή. Χρησιμοποιώντας τη δομή να υπολογίσετε την απόσταση του σημείου από την αρχή των αξόνων.

Μια πιθανή λύση είναι:

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

int main(void)
{
  /* orismos tis domis */
  struct point{
    float x;
    float y;
  }pt1;

  float distance;

  /* eisagogh ton sintetagmenon x kai y */
  printf("Doste ti sintetagmeni x : ");
  scanf("%f",&pt1.x);
  printf("Doste ti sintetagmeni y : ");
  scanf("%f",&pt1.y);

  /* ypologismos kai ektiposi tis apostasis apo to shmeio (0,0) */
  distance=sqrt(pow(pt1.x,2)+pow(pt1.y,2));
  printf("H apostatasi tou shmeiou apo to (0,0) einai : %f\n",distance);
  return 0;
}
```

Η εκτέλεση του προγράμματος εικονίζεται παρακάτω:

```
[student1@pc244 kef5]$ a.out
Doste ti sintetagmeni x : 2.34
Doste ti sintetagmeni y : 7.89
H apostatasi tou shmeiou apo to (0,0) einai : 8.229684
[student1@pc244 kef5]$
```

4.1.2 Να γράψετε ένα πρόγραμμα στο οποίο να εισάγετε από το πληκτρολόγιο τον τύπο και την ταχύτητα της CPU, το έτος κατασκευής και το κόστος του Η/Υ σας. Να αποθηκεύσετε τα δεδομένα σε κατάλληλη δομή. Να χρησιμοποιήσετε τη δομή για να τυπώσετε τα δεδομένα.

Μια πιθανή λύση είναι:

```
#include<stdio.h>

int main(void)
{
  /* Orismos domis */
  struct computer{
    char cpu_type[12];
    float cpu_speed;
    int year;
    float cost;
  }my_computer;

  /* Eisagogi dedomenon domis */
  printf("Eisagete ton typo tis CPU : ");
  scanf("%s",&my_computer.cpu_type);
  printf("Eisagete tin taxitita tis CPU : ");
  scanf("%f",&my_computer.cpu_speed);
  printf("Eisagete to etos kataskeyis tou H/Y : ");
  scanf("%d",&my_computer.year);
  printf("Eisagete to kostos tou H/Y : ");
  scanf("%f",&my_computer.cost);

  /* ektiposi dedomenon eisagogis */
  printf("\nDedomena Eisagogis:\n");
  printf("Typos CPU         : %s\n",my_computer.cpu_type);
  printf("Taxitita CPU      : %2.1f GHz\n",my_computer.cpu_speed);
  printf("Etos kataskeyis   : %d\n",my_computer.year);
  printf("Kostos agoras H/Y : %6.1f Euro\n",my_computer.cost);

  return 0;
}
```

Η εκτέλεση του παραπάνω προγράμματος είναι:

```
[student1@pc244 kef5]$ a.out
Eisagete ton typo tis CPU : Pentium
Eisagete tin taxitita tis CPU : 1.7
Eisagete to etos kataskeyis tou H/Y : 2001
Eisagete to kostos tou H/Y : 1200

Dedomena Eisagogis:
Typos CPU         : Pentium
Taxitita CPU      : 1.7 GHz
Etos kataskeyis   : 2001
Kostos agoras H/Y : 1200.0 Euro
[student1@pc244 kef5]$
```

4.1.3 Γράψτε ένα πρόγραμμα στο οποίο να εισάγετε από το πληκτρολόγιο τις συντεταγμένες των δύο σημείων που αντιπροσωπεύουν τις διαγώνια απέναντι κορυφές ενός ορθογωνίου. Να τις αποθηκεύσετε σε μια κατάλληλη δομή. Χρησιμοποιώντας τη δομή να υπολογίσετε το εμβαδόν του ορθογωνίου. Λύστε το πρόβλημα μόνο στην περίπτωση όπου και τα δύο σημεία βρίσκονται στο πρώτο τεταρτημόριο που καθορίζουν οι άξονες x και y.

Μια πιθανή λύση είναι:

```
#include<stdio.h>

int main(void)
{
  /* orismos tis domis */
  struct point{
    float x;
    float y;
  };

  struct rect{
    struct point pt1;
    struct point pt2;
  }screen;

  float emvado;

  /* eisagogh ton sintetagmenon x kai y */
  printf("Doste ti sintetagmeni x kai y tou protou shmeiou   : ");
  scanf("%f %f",&screen.pt1.x,&screen.pt1.y);
  printf("Doste ti sintetagmeni x kai y tou deyterou shmeiou : ");
  scanf("%f %f",&screen.pt2.x,&screen.pt2.y);

  /* ypologismos kai ektiposi embadou orthogoniou */
  emvado=abs(screen.pt2.x-screen.pt1.x)*abs(screen.pt2.y-screen.pt1.y);
  printf("To emvado tou orthogoniou einai : %f\n",emvado);
  return 0;
}
```

Η εκτέλεση του παραπάνω προγράμματος είναι:

```
[student1@pc244 kef5]$ a.out
Doste ti sintetagmeni x kai y tou protou shmeiou   : 8 9
Doste ti sintetagmeni x kai y tou deyterou shmeiou : 3 4
To emvado tou orthogoniou einai : 25.000000
[student1@pc244 kef5]$
```

4.1.4 Γράψτε ένα πρόγραμμα στο οποίο να δημιουργήσετε μια κατάλληλη δομή η οποία να περιέχει ως μέλη τα ακόλουθα στοιχεία για ένα αυτοκίνητο:

Κατασκευαστής
Μοντέλο
Κυβισμός Μηχανής σε λίτρα
Έτος Κατασκευής.
Κόστος σε Euro.

Αρχικοποιήσετε την με τα ακόλουθα δεδομένα . {“OPEL”, “VECTRA”, 1.8, 1996, 25000}. Στη συνέχεια γράψτε μια συνάρτηση η οποία να τυπώνει αυτά τα δεδομένα. Χρησιμοποιείστε την συνάρτηση από την main().

Μια πιθανή λύση είναι η παρακάτω:

```
#include<stdio.h>

struct automobile{
  char manufacturer[12];
  char model[12];
  float capacity;
  int year;
  float cost;
};

void print_struct(struct automobile car_p);

int main(void)
{
  struct automobile car={"OPEL","Vectra",1.8,1996,25000};

  print_struct(car);

  return 0;
}

void print_struct(struct automobile car_p)
{
  printf("Manufactuter      : %s\n",car_p.manufacturer);
  printf("Model             : %s\n",car_p.model);
  printf("Engine capacity   : %3.1f liters\n",car_p.capacity);
  printf("Construction year : %d\n",car_p.year);
  printf("Cost              : %7.1f Euro\n",car_p.cost);
}
```

Η εκτέλεση του παραπάνω προγράμματος εικονίζεται παρακάτω:

```
[student1@pc244 kef5]$ a.out
Manufactuter      : OPEL
Model             : Vectra
Engine capacity   : 1.8 liters
Construction year : 1996
Cost              : 25000.0 Euro
[student1@pc244 kef5]$
```

Μια δομή περνάει σε μια συνάρτηση μέσου ενός δείκτη ο οποίος δείχνει σε αυτήν. Μια πιθανή λύση με χρήση δεικτών σε συνάρτηση εικονίζεται παρακάτω:

```
#include<stdio.h>

struct automobile{
  char manufacturer[12];
  char model[12];
  float capacity;
  int year;
  float cost;
};

void print_struct(struct automobile *pt_car); /* Dilosi tis print_struct */

int main(void)
{
  struct automobile *point_car, car={"OPEL","Vectra",1.8,1996,25000};
  point_car=&car;
  print_struct(point_car);  /* Diktis os orisma sti synartisi */

  return 0;
}

void print_struct(struct automobile *pt_car)
{
  printf("Manufactuter      : %s\n",pt_car->manufacturer);
  printf("Model             : %s\n",pt_car->model);
  printf("Engine capacity   : %3.1f liters\n",pt_car->capacity);
  printf("Construction year : %d\n",pt_car->year);
  printf("Cost              : %7.1f Euro\n",pt_car->cost);
}
```

Η εκτέλεση του παραπάνω προγράμματος εικονίζεται παρακάτω και είναι ακριβώς η ίδια με το προηγούμενο παράδειγμα:

```
[student1@pc244 kef5]$ a.out
Manufactuter      : OPEL
Model             : Vectra
Engine capacity   : 1.8 liters
Construction year : 1996
Cost              : 25000.0 Euro
[student1@pc244 kef5]$
```

4.1.5 Γράψτε ένα πρόγραμμα στο οποίο να δημιουργήσετε μια κατάλληλη δομή η οποία να περιέχει ως μέλη τα ακόλουθα στοιχεία για τέσσερα διαφορετικά αυτοκίνητα:

Κατασκευαστής
Μοντέλο
Κυβισμός Μηχανής σε λίτρα
Έτος Κατασκευής.
Κόστος σε Euro.

Αρχικοποιήσετε την με τα ακόλουθα δεδομένα:

{“OPEL”, “VECTRA”, 1.8, 1996, 25000}
{“FORD”, “Monteo”, 1.8, 2001, 25000}
{“FIAT”, “Punto”, 1.4, 2002, 12000}
{“HUNDAY”, “Atos”, 1.0, 2002, 9000}

Στη συνέχεια γράψτε μια συνάρτηση η οποία να τυπώνει αυτά τα δεδομένα για κάθε αυτοκίνητο. Χρησιμοποιείστε την συνάρτηση από την main() για να εκτυπώσετε τον πλήρη κατάλογο των αυτοκινήτων με τα στοιχεία τους.

Μια πιθανή λύση είναι η ακόλουθη:

```
#include<stdio.h>

struct automobile{
  char manufacturer[12];
  char model[12];
  float capacity;
  int year;
  float cost;
};

void print_struct(struct automobile car_p);

int main(void)
{
  struct automobile cars[4]={{"OPEL","Vectra",1.8,1996,25000},
                             {"FORD","Monteo",1.8,2001,24000},
                             {"FIAT","Punto",1.4,2002,12000},
                             {"HUNDAY","Atos",1.0,2002,9000}};
  int i;

  for(i=0;i<4;i++){
    print_struct(cars[i]);
    printf("\n");
  }

  return 0;
}

void print_struct(struct automobile car_p)
{
  printf("Manufactuter      : %s\n",car_p.manufacturer);
  printf("Model             : %s\n",car_p.model);
  printf("Engine capacity   : %3.1f liters\n",car_p.capacity);
  printf("Construction year : %d\n",car_p.year);
  printf("Cost              : %7.1f Euros\n",car_p.cost);
}
```

Προσέξτε τον τρόπο αρχικοποίησης του πίνακα δομών cars[4]. Η συνάρτηση print_struct() είναι η ίδια με αυτή που χρησιμοποιήσαμε στην πρώτη λύση του προηγουμένου παραδείγματος.

Η εκτέλεση του παραπάνω προγράμματος είναι η ακόλουθη:

```
[student1@pc244 kef5]$ a.out
Manufactuter      : OPEL
Model             : Vectra
Engine capacity   : 1.8 liters
Construction year : 1996
Cost              : 25000.0 Euros

Manufactuter      : FORD
Model             : Monteo
Engine capacity   : 1.8 liters
Construction year : 2001
Cost              : 24000.0 Euros

Manufactuter      : FIAT
Model             : Punto
Engine capacity   : 1.4 liters
Construction year : 2002
Cost              : 12000.0 Euros

Manufactuter      : HUNDAY
Model             : Atos
Engine capacity   : 1.0 liters
Construction year : 2002
Cost              :  9000.0 Euros

[student1@pc244 kef5]$
```

Οι πίνακες δομών όπως φαίνεται και στο παραπάνω παράδειγμα μπορούν εύκολα να χρησιμοποιηθούν για την κατασκευή απλών **Βάσεων Δεδομένων**. Οι βάσεις δεδομένων σήμερα αποτελούν χρήσιμα εργαλεία για την μηχανογράφηση προϊόντων, εργασιών, δεδομένων κτλ. και χρησιμοποιούνται ευρέως σε μεγάλη κλίμακα. Οι βάσεις δεδομένων επιτρέπουν στην εύκολη και ταχύτατη προσπέλαση σε πληροφορίες.

4.1.6 Γράψτε ένα πρόγραμμα στο οποίο να δημιουργήσετε την ακόλουθη ένωση:

```
union u_tag{
        int ival;
        float fval;
        char sval;
        };
```

Δώστε στα μέλη της ένωσης τις τιμές 15, 3,14159 και 'W'. Γράψτε μια συνάρτηση η οποία να τυπώνει τα μέλη της ένωσης. Χρησιμοποιήστε την συνάρτηση για να τυπώνετε μετά από κάθε αρχικοποίηση μέλους της ένωσης όλα τα μέλη αυτής.

Μια πιθανή λύση είναι η ακόλουθη:

```
#include<stdio.h>

union u_tag{                    /* orismos enosis */
  int ival;
  float fval;
  char sval;
};

void print_x(union u_tag x);    /* dilosi sinartisis ektiposis tis enosis */

int main(void)
{
  union u_tag u;

  u.ival=15;                /* Akeraia timi stin u */
  print_x(u);

  u.fval=3.14159;           /* Pragmatiki timi stin u */
  print_x(u);

  u.sval='W';               /* Xaraktiras stin u */
  print_x(u);

  return 0;
}

void print_x(union u_tag x)
{
    printf("H enosi exei tin akeraia timi     : %d\n",x.ival);
    printf("H enosi exei tin pragmatiki timi  : %f\n",x.fval);
    printf("H enosi periexei tin symboloseira : %c\n\n",x.sval);
}
```

Προσέξτε πως στο παραπάνω πρόγραμμα κάθε φορά αρχικοποιούμε ένα μέλος της ένωσης και στη συνέχεια τυπώνουμε όλα τα μέλη αυτής. Σε κάθε εκτύπωση μόνο ένα από τα μέλη της ένωσης έχει την σωστή τιμή. Τα άλλα δύο μέλη αυτής παίρνουν τυχαίες τιμές οι οποίες εξαρτώνται από το είδος του υπολογιστή μας και την κατάσταση στην οποία ευρίσκεται την στιγμή της εκτέλεσης του προγράμματος.

Η εκτέλεση του προγράμματός μας εικονίζεται στο αμέσως επόμενο σχήμα.

```
[student1@pc244 kef5]$ a.out
H enosi exei tin akeraia timi     : 15
H enosi exei tin pragmatiki timi  : 0.000000
H enosi periexei tin symboloseira :

H enosi exei tin akeraia timi     : 1078530000
H enosi exei tin pragmatiki timi  : 3.141590
H enosi periexei tin symboloseira : Đ

H enosi exei tin akeraia timi     : 1078529879
H enosi exei tin pragmatiki timi  : 3.141561
H enosi periexei tin symboloseira : W

[student1@pc244 kef5]$
```

4.2 Προβλήματα.

4.2.1 Να γράψετε ένα πρόγραμμα στο οποίο να δίνετε από το πληκτρολόγιο το πραγματικό μέρος και το φανταστικό μέρος ενός μιγαδικού αριθμού. Να αποθηκεύσετε τα δεδομένα σε κατάλληλη δομή. Χρησιμοποιείστε τη δομή για να υπολογίσετε το μέτρο του μιγαδικού αριθμού. Τυπώστε το αποτέλεσμα.

4.2.2 Να γράψετε ένα πρόγραμμα στο οποίο να δίνετε από το πληκτρολόγιο το πραγματικό μέρος και το φανταστικό μέρος δύο μιγαδικών αριθμών **a=real(a)+im(a)i** και **b=real(b)+im(b)i**. Να αποθηκεύσετε τα δεδομένα σε κατάλληλες δομές. Να κατασκευάσετε νέες κατάλληλες δομές οι οποίες να περιέχουν το αποτέλεσμα των παρακάτω πράξεων μιγαδικών αριθμών: **c=a+b**, **d=a-b** και **e=a*b**. Χρησιμοποιήστε τις νέες δομές για να τυπώσετε τους μιγαδικούς αριθμούς c, d και e.

4.2.3 Να γράψετε ένα πρόγραμμα στο οποίο να δίνετε από το πληκτρολόγιο το πραγματικό μέρος και το φανταστικό μέρος δύο μιγαδικών αριθμών **x=a+bi** και **y=c+di**. Να αποθηκεύσετε τα δεδομένα σε κατάλληλες δομές. Να κατασκευάσετε νέα κατάλληλη δομή η οποία να περιέχει το αποτέλεσμα της πράξης **w=x/y**. Χρησιμοποιήστε τη νέα δομή για να τυπώσετε τον μιγαδικό αριθμό c.

Δίνεται: $\frac{a+bi}{c+di} = \frac{ac+bd}{c^2+d^2} + \left(\frac{bc-ad}{c^2+d^2}\right)i$

4.2.4 Να γράψετε ένα πρόγραμμα στο οποίο να κατασκευάσετε μια βάση δεδομένων για πέντε φοιτητές η οποία να περιέχει για κάθε φοιτητή τα ακόλουθα:

Επώνυμο
Όνομα
Αριθμό Μητρώου

Εισάγετε από το πληκτρολόγιο τα δεδομένα για τον εαυτό σας και για άλλους τέσσερις συμφοιτητές σας. Να αποθηκεύσετε τα δεδομένα σε κατάλληλο πίνακα δομών. Χρησιμοποιείστε τη δομή για να τυπώσετε τα δεδομένα.

4.2.5 Να γράψετε ένα πρόγραμμα στο οποίο να κατασκευάσετε μια βάση δεδομένων για έξι φοιτητές η οποία να περιέχει για κάθε φοιτητή τα ακόλουθα:

Επώνυμο
Όνομα
Αριθμό Μητρώου
Βαθμό στο μάθημα υπολογιστών

Να αποθηκεύσετε τα δεδομένα σε κατάλληλο πίνακα δομών. Αρχικοποιείστε τον πίνακα με τα ακόλουθα δεδομένα:

{“Anastasiou”, “Dimitris”, 5, 5.5}
{“Georgiou”, “Apostolos”, 2, 8}
{“Dimitriou”, “Periklis”, 6, 7.5}
{“Karatasos”, “Panagiotis”, 4, 10}
{“Konstantopoulou”, “Maria”, 1, 7}
{“Ladopoulos”, “Kostas”, 3, 6.5}

Γράψτε ένα πρόγραμμα στο οποίο να εισάγετε από το πληκτρολόγιο τον αριθμό μητρώου του φοιτητή και να τυπώνει το επώνυμο το όνομα και τον βαθμό του.

4.2.6 Γράψτε ένα πρόγραμμα, με την βάση δεδομένων του προηγούμενου προβλήματος, στο οποίο να εισάγετε από το πληκτρολόγιο το επώνυμο του φοιτητή και να τυπώνει το όνομα τον αριθμό μητρώου και τον βαθμό του.

Υπόδειξη: Να χρησιμοποιήσετε τη συνάρτηση strcmp() της βιβλιοθήκης <string.h>.

Κεφάλαιο V

Προβλήματα που αφορούν την προσπέλαση αρχείων.

Στο παρόν κεφάλαιο παρουσιάζονται προβλήματα τα οποία αφορούν στην προσπέλαση αρχείων. Παρουσιάζονται ασκήσεις οι οποίες αναφέρονται σε:

- **Δημιουργία νέων αρχείων**
- **Επεξεργασία υπαρχόντων αρχείων**
- **Συναρτήσεις fopen() και fclose()**
- **Συναρτήσεις fprintf() και fscanf()**
- **Συναρτήσεις putc() και getc()**

5.1 Λυμένα Προβλήματα.

5.1.1 Γράψτε ένα πρόγραμμα το οποίο να δημιουργεί το αρχείο integer.dat και να γράφει σε αυτό τους ακεραίους 100 έως και 109 στη σειρά. Στη συνέχεια αφού ελέγξετε τα περιεχόμενα του αρχείου να γράψετε ένα δεύτερο πρόγραμμα το οποίο να διαβάζει και να τυπώνει τους ακεραίους που περιέχονται σε αυτό.

Μια πιθανή λύση είναι:

```
#include<stdio.h>

int main(void)
{
  int i;

  /* Orismos Deikti gia to arxeio dedomenon */
  FILE *file_pointer;

  /* Anoigma tou arxeiou integer.dat gia grapsimo se afto */
  file_pointer=fopen("integer.dat","w");

  /* Egrafh akeraion sto arxeio integer.dat */
  for(i=0;i<10;i++)
      fprintf(file_pointer,"%5d",i+100);

  /* Kleisimo tou arxeiou dedomenon integer.dat */
  fclose(file_pointer);
  return 0;
}
```

Η εκτέλεση του προγράμματος εικονίζεται παρακάτω:

```
[student1@pc244 kef6]$ a.out
[student1@pc244 kef6]$ more integer.dat
  100   101   102   103   104   105   106   107   108   109
[student1@pc244 kef6]$
```

Κατά την εκτέλεση δημιουργείται το αρχείο integer.dat. Για να δούμε τα περιεχόμενα του αρχείου integer.dat χρησιμοποιούμε την εντολή more του Linux, όπως φαίνεται παραπάνω. Ένας άλλος τρόπος να δούμε τα περιεχόμενα του αρχείου είναι να το ανοίξουμε με έναν διορθωτή κειμένου όπως πχ. τον emacs.

Το πρόγραμμα το οποίο να διαβάζει και να τυπώνει τους ακεραίους που περιέχονται στο αρχείο integer.dat το οποίο δημιουργήσαμε εικονίζεται στο επόμενο σχήμα.

```
#include<stdio.h>

int main(void)
{
  int i,integ;

  /* Orismos Deikti gia to arxeio dedomenon */
  FILE *file_pointer;

  /* Anoigma tou arxeiou integer.dat gia diabasma apo afto */
  file_pointer=fopen("integer.dat","r");

  /* Diabasma akeraion apo to arxeio integer.dat */
  for(i=0;i<10;i++){
      fscanf(file_pointer,"%d",&integ);
      printf("%d\n",integ);
  }

  /* Kleisimo tou arxeiou dedomenon integer.dat */
  fclose(file_pointer);
  return 0;
}
```

Η εκτέλεση του προγράμματος εικονίζεται παρακάτω:

```
[student1@pc244 kef6]$ a.out
100
101
102
103
104
105
106
107
108
109
[student1@pc244 kef6]$
```

5.1.2 Γράψτε ένα πρόγραμμα στο οποίο να εισάγετε από το πληκτρολόγιο την πρόταση: "C is a general-purpose programming language.". Στη συνέχεια να δημιουργήσετε το αρχείο text.dat και να γράψετε την πρόταση σε αυτό. Ελέγξετε τα περιεχόμενα του αρχείου που γράψατε.

Μια πιθανή λύση είναι:

```
#include<stdio.h>

int main(void)
{
  char c;

  /* Dimiourgia neou arxeiou */
  FILE *f_pnt;
  f_pnt=fopen("text1.dat","w");

  printf("Eisagete tin protasi : ");
  while((c=getc(stdin)) != '\n')     /* sinartiseis getc() kai putc() */
    putc(c,f_pnt);

  fclose(f_pnt);
  return 0;
}
```

Η εκτέλεση του προγράμματος εικονίζεται παρακάτω:

```
[student1@pc244 kef6]$ a.out
Eisagete tin protasi : C is a general-purpose programming language.
[student1@pc244 kef6]$
[student1@pc244 kef6]$ more text1.dat
C is a general-purpose programming language.
[student1@pc244 kef6]$
```

Προσέξτε την σύνταξη της εντολής while. Η πρόταση εισάγετε από το πληκτρολόγιο. Όταν τελειώσουμε την εισαγωγή της πατάμε το **Enter** το οποίο αντιστοιχεί στον χαρακτήρα νέας γραμμής **'\n'**. Με αυτόν τον τρόπο εξερχόμαστε από την εντολή while και τερματίζεται το πρόγραμμα . Η πρόταση γράφεται στο αρχείο text1.dat το οποίο και ελέγχουμε χρησιμοποιώντας την εντολή more του Linux.

5.1.3 Χρησιμοποιώντας τον emacs δημιουργήστε το αρχείο text2.dat και γράψτε σε αυτό το παρακάτω κείμενο:

C is a general-purpose programming language.
Dennis Ritchie of Bell Laboratories originally designed it.
It was first used as the system language for the UNIX operating system.

Γράψτε ένα πρόγραμμα το οποίο να διαβάζει το παραπάνω κείμενο από το αρχείο text2.dat και να τυπώνει στην οθόνη του Η/Υ.

Στον emacs γράφουμε τα παρακάτω:

```
C is a general-purpose programming language.
Dennis Ritchie of Bell Laboratories originally designed it.
It was first used as the system language for the UNIX operating system.

-:--  text2.dat        (Fundamental)--L3--All----------------------------
```

Μια πιθανή λύση είναι:

```
#include<stdio.h>

int main(void)
{
  char c;

  /* Anoigma iparxontos arxeiou gia diabasma*/
  FILE *f_pnt;
  f_pnt=fopen("text2.dat","r");

  printf("Periexomena arxeiou:\n");
  while((c=getc(f_pnt)) != EOF)   /* xrisi sinartiseon getc() kai putc() */
    putc(c,stdout);

  fclose(f_pnt);
  return 0;
}
```

Η εντολή while ελέγχει εάν φτάνουμε στο τέλος του αρχείου μέσω της τιμής EOF (End Of File) η οποία είναι διαφορετική από κάθε άλλον χαρακτήρα. Η εκτέλεση του προγράμματος εικονίζεται παρακάτω:

```
[student1@pc244 kef6]$ a.out
Periexomena arxeiou:
C is a general-purpose programming language.
Dennis Ritchie of Bell Laboratories originally designed it.
It was first used as the system language for the UNIX operating system.
[student1@pc244 kef6]$
```

5.1.4 Χρησιμοποιώντας το αρχείο text2.dat το οποίο δημιουργήσατε στο προηγούμενο πρόβλημα γράψτε ένα πρόγραμμα το οποίο να διαβάζει το κείμενο και να υπολογίζει τον αριθμό των χαρακτήρων ‘a’, ‘e’ και ‘o’ που εμπεριέχονται σ’ αυτό.

Μια πιθανή λύση είναι:

```
#include<stdio.h>

int main(void)
{
  char c;
  int count_a,count_e,count_o;

  /* Anoigma iparxontos arxeiou gia diavasma */
  FILE *f_pnt;
  f_pnt=fopen("text2.dat","r");

  count_a=0;
  count_e=0;
  count_o=0;
  while((c=getc(f_pnt)) != EOF){   /* xrisi sinartisis getc() */
    if(c=='a')
      ++count_a;

    if(c=='e')
      ++count_e;

    if(c=='o')
      ++count_o;
  }

  printf("O xaraktiras 'a' emperiexetai : %d fores.\n",count_a);
  printf("O xaraktiras 'e' emperiexetai : %d fores.\n",count_e);
  printf("O xaraktiras 'o' emperiexetai : %d fores.\n",count_o);

  fclose(f_pnt);
  return 0;
}
```

Η εκτέλεση του προγράμματος εικονίζεται παρακάτω:

```
[student1@pc244 kef6]$ a.out
O xaraktiras 'a' emperiexetai : 13 fores.
O xaraktiras 'e' emperiexetai : 17 fores.
O xaraktiras 'o' emperiexetai : 8 fores.
[student1@pc244 kef6]$
```

5.1.5 Γράψτε ένα πρόγραμμα στο οποίο να εισάγετε από το πληκτρολόγιο την πρόταση: “C is a general-purpose programming language.”. Να δημιουργήσετε το αρχείο text3.dat και να γράψετε την πρόταση σε αυτό με κεφαλαίους χαρακτήρες. Ελέγξετε τα περιεχόμενα του αρχείου που γράψατε.

Για να μετατρέψουμε ένα κεφαλαίο χαρακτήρα σε πεζό χρησιμοποιούμε την συνάρτηση toupper() η οποία εμπεριέχεται στην βιβλιοθήκη <ctype.h>. Οι συναρτήσεις για τη μετατροπή των πεζών χαρακτήρων σε κεφαλαίους και αντίστροφα είναι οι ακόλουθες:

int tolower(char c)	μετατρέπει τον c σε πεζό
int toupper(char c)	μετατρέπει τον c σε κεφαλαίο

Μια πιθανή λύση είναι η ακόλουθη:

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

int main(void)
{
  char c;

  /* Dimiourgia neou arxeiou */
  FILE *f_pnt;
  f_pnt=fopen("text3.dat","w");

  printf("Eisagete tin protasi : ");
  while((c=getc(stdin)) != EOF)     /* sinartiseis getc() kai putc() */
    putc(toupper(c),f_pnt);         /* sinartisi toupper() */

  fclose(f_pnt);
  return 0;
}
```

Προσέξτε πως η εντολή while() ελέγχει το τέλος του κειμένου με τον χαρακτήρα EOF. Ο χαρακτήρας EOF δίνεται από το πληκτρολόγιο με Ctrl-D.

Η εκτέλεση του προγράμματος εικονίζεται παρακάτω:

```
[student1@pc244 kef6]$ a.out
Eisagete tin protasi : C is a general-purpose programming language.
[student1@pc244 kef6]$ more text3.dat
C IS A GENERAL-PURPOSE PROGRAMMING LANGUAGE.
[student1@pc244 kef6]$
```

5.1.6 Η Γραμματεία του Τμήματος Φυσικής μας έδωσε την παρακάτω λίστα από δέκα επιτυχόντες φοιτητές στο μάθημα των Η/Υ.

```
Anastasiou Dimitrios 5 5.5
Georgiou Apostolos 2 8.
Dimitriou Periklis 6 7.5
Karatasos Panagiotis 4 10.
Konstantopoulou Maria 1 7.
Ladopoulos Kostas 3 6.5
Mpalafas Ioannis 10 8.5
Papadopoulos Grigorios 9 9.
Stasinou Alexandra 6 5.
Xanthopoulos Dimitrios 7 6.
-:--  fitites.dat       (Fundamental)--L10--All-----------------------------
```

Η παραπάνω λίστα εμπεριέχεται στο αρχείο fitites.dat και για κάθε φοιτητή περιλαμβάνει το Επώνυμο, Όνομα, Αριθμό Μητρώου και βαθμό στο μάθημα. Να γράψετε ένα πρόγραμμα το οποίο να διαβάζει τα δεδομένα από το παραπάνω αρχείο και να τα αποθηκεύει σε κατάλληλη δομή. Στη συνέχεια χρησιμοποιώντας την δομή να εκτυπώσετε τα αποτελέσματα σε μορφή πίνακα.

Μια πιθανή λύση είναι η ακόλουθη:

```
#include<stdio.h>

int main(void)
{
  int i;
  struct fitites{            /* Orismos Domis */
    char eponimo[18];
    char onoma[18];
    int AM;
    float bathmos;
  }lista[10];

  /* Prosbasi sto arxeio fitites.dat */
  FILE *f_pnt;
  f_pnt=fopen("fitites.dat","r");

  /* Diavasma dedomenon */
  for(i=0;i<10;i++)
    fscanf(f_pnt,"%s %s %d %f", &lista[i].eponimo, &lista[i].onoma,
                                &lista[i].AM, &lista[i].bathmos);


  /* Ektiposi Domis */
  printf("\nBathmologia sto mathima H/Y\n");
  printf("Eponimo            Onoma              AM   Bathmos\n");
  for(i=0;i<10;i++){
    printf("%-18s %-18s %2d %6.1f\n",lista[i].eponimo, lista[i].onoma,
                              lista[i].AM, lista[i].bathmos);
  }

  fclose(f_pnt);
  return 0;
}
```

Η εκτέλεση του παραπάνω προγράμματος εικονίζεται παρακάτω:

```
[student1@pc244 kef6]$ a.out

Bathmologia sto mathima H/Y
Eponimo            Onoma              AM   Bathmos
Anastasiou         Dimitrios           5    5.5
Georgiou           Apostolos           2    8.0
Dimitriou          Periklis            6    7.5
Karatasos          Panagiotis          4   10.0
Konstantopoulou    Maria               1    7.0
Ladopoulos         Kostas              3    6.5
Mpalafas           Ioannis            10    8.5
Papadopoulos       Grigorios           9    9.0
Stasinou           Alexandra           6    5.0
Xanthopoulos       Dimitrios           7    6.0
[student1@pc244 kef6]$
```

5.2 Προβλήματα.

5.2.1 Γράψτε ένα πρόγραμμα το οποίο να δημιουργεί ένα αρχείο με το όνομα table.dat. Να γράψετε σε αυτό τους ακεραίους αριθμούς από το 0 έως το 99 στοιχισμένους σε 10 γραμμές με 10 αριθμούς ανά γραμμή. Να ελέγξετε τα περιεχόμενα του αρχείου table.dat χρησιμοποιώντας πχ. τον emacs.

5.2.2 Γράψτε ένα πρόγραμμα το οποίο να διαβάζει τα περιεχόμενα του αρχείου table.dat το οποίο δημιουργήσατε στο προηγούμενο πρόβλημα. Να υπολογίσετε την τετραγωνική ρίζα για κάθε αριθμό από αυτούς που εμπεριέχονται σε αυτό. Να γράψτε τα αποτελέσματά σας, αφού τα στοιχίσετε σε 10 γραμμές με 10 αριθμούς ανά γραμμή, σε ένα νέο αρχείο με το όνομα sqrt.dat. Να ελέγξετε τα περιεχόμενα του νέου αρχείου χρησιμοποιώντας την εντολή more του Linux. Διατηρείστε ακρίβεια 3 δεκαδικών ψηφίων στους υπολογισμούς σας.

5.2.3 Γράψτε ένα πρόγραμμα το οποίο να δημιουργεί ένα αρχείο με το όνομα numbers.dat. Να γράψετε σε αυτό 100 τυχαίους αριθμούς τύπου float από το 500 έως το 600 στοιχισμένους σε 10 γραμμές με 10 αριθμούς ανά γραμμή. Κρατείστε ακρίβεια 3 δεκαδικών ψηφίων. Να ελέγξετε τα περιεχόμενα του αρχείου numbers.dat. Στη συνέχεια να γράψετε ένα δεύτερο πρόγραμμα το οποίο να διαβάζει τους τυχαίους αριθμούς οι οποίοι εμπεριέχονται στο αρχείο που δημιουργήσατε και να υπολογίσετε το άθροισμά, τον μέσο και τυπική απόκλιση $s = \sqrt{\frac{\left(d_1^2 + d_2^2 + \ldots + d_m^2\right)}{m(m-1)}}$ τους, όπου $m = 100$ και $d_i = x_i - \bar{x}$.

5.2.4 Γράψτε ένα πρόγραμμα το οποίο να δημιουργεί ένα αρχείο με το όνομα keimeno.dat. Να γράψετε σε αυτό το παρακάτω κείμενο δακτυλογραφώντας το από το πληκτρολόγιο:

"The heart and soul of C programming is the function.
A function represents a piece of code that is a building
block in the problem solving process. All functions are
on the same external level; they cannot be nested one
inside another."

Να ελέγξετε τα περιεχόμενα του αρχείου keimeno.dat το οποίο δημιουργήσατε.

Υπόδειξη: Ο χαρακτήρας EOF δίνεται από το πληκτρολόγιο με Ctrl-D για να τερματίσετε την δακτυλογράφηση του κειμένου.

5.2.5 Να γράψτε ένα πρόγραμμα το οποίο να διαβάζει το αρχείο keimeno.dat του προηγούμενου προβλήματος και να υπολογίζει τον αριθμό των χαρακτήρων, των λέξεων και των γραμμών που εμπεριέχονται σε αυτό.

5.2.6 Να γράψτε ένα πρόγραμμα το οποίο να διαβάζει το αρχείο keimeno.dat του προβλήματος 5.2.4. Να δημιουργήσετε ένα νέο αρχείο με το όνομα keimeno1.dat το οποίο να περιέχει το ίδιο κείμενο γραμμένο με κεφαλαία γράμματα. Να ελέγξετε τα περιεχόμενα του νέου αρχείου keimeno1.dat το οποίο δημιουργήσατε.

5.2.7 Να γράψτε ένα πρόγραμμα το οποίο να διαβάζει το αρχείο fitites.dat του προβλήματος 5.1.6 και να αποθηκεύει τα αποτελέσματα σε κατάλληλη δομή. Στη συνέχεια να δημιουργήσετε ένα νέο αρχείο με το όνομα fitites_1.dat στο οποίο να γράψετε, χρησιμοποιώντας την δομή, την ίδια λίστα των φοιτητών αυτή τη φορά κατ' αύξοντα αριθμό μητρώου.

Κεφάλαιο VI

Προβλήματα Επανάληψης.

Στο παρόν κεφάλαιο παρουσιάζονται γενικές επαναληπτικές ασκήσεις.

6.1 Λυμένα Προβλήματα.

6.1.1 Περιγράψτε σε συντομία την λειτουργία του παρακάτω προγράμματος. Τι ακριβώς τυπώνει στην οθόνη το πρόγραμμα όταν εκτελείται;

```
#include<stdio.h>

int main(void)
{
  int i, x;
  i=0;
  x=0;

  for(i=1; i<10; i*=2) {
    x++;
    printf("%d ", x);
  }
  printf("\nx = %d\n", x);
}
```

Λύση:

Το πρόγραμμα αυτό εκτελεί ένα βρόχο μεσω της εντολής for.
Ο βρόχος εκτελείται τέσσερις φορές: Την πρώτη είναι i=1, τη δεύτερη i=2, την τρίτη i=4 και την τέταρτη i=8.
Κάθε φορά που εκτελείται ο βρόχος, η τιμή του x (η οποία είναι αρχικά μηδέν) αυξάνεται κατά ένα και τυπώνεται στην οθόνη (χωρίς αλλαγή γραμμής και βάζοντας ένα κενό μετά τον αριθμό).
Μετά το τέλος της εκτέλεσης της εντολής for, τυπώνεται στην οθόνη ο αριθμός x, βάσει της μορφοποίησης που δηλώνεται στην τελευταία εντολή printf..
Η έξοδος του προγράμματος στην οθόνη είναι λοιπόν:

```
1 2 3 4
x = 4
```

6.1.2 Περιγράψτε σε συντομία τη λειτουργία του παρακάτω προγράμματος. Τι ακριβώς τυπώνει στην οθόνη το πρόγραμμα όταν εκτελείται;

```
#include<stdio.h>

int main(void)
{
  int i, x;
  x=0;

  for(i=0;i<=12;i+=3){
    x += 2;
    printf("%d ",x);
    x++;
  }
  printf("\n x = %d\n",x);
  return 0;
}
```

Λύση:

Το πρόγραμμα αυτό εκτελεί ένα βρόχο μεσω της εντολής for.
Ο βρόχος εκτελείται πέντε φορές: Την πρώτη για i=0, τη δεύτερη για i=3, την τρίτη για i=6, την τέταρτη για i=9 και την πέμπτη για i=12.
Κάθε φορά που εκτελείται ο βρόχος, η τιμή του x (η οποία είναι αρχικά μηδέν) αυξάνεται κατά δύο, τυπώνεται στην οθόνη (χωρίς αλλαγή γραμμής και βάζοντας ένα κενό μετά τον αριθμό) και επιπλέον αυξάνεται κατά ένα.
Μετά το τέλος της εκτέλεσης της εντολής for, τυπώνεται στην οθόνη η τελική τιμή του x, βάσει της μορφοποίησης που δηλώνεται στην τελευταία εντολή printf.
Η έξοδος του προγράμματος στην οθόνη είναι:
2 5 8 11 14
x = 15

6.1.3 Εξηγήστε τη λειτουργία του παρακάτω προγράμματος. Τι ιδιαιτερότητα παρουσιάζουν οι χρησιμοποιούμενες εντολές for; Τι ακριβώς τυπώνει στην οθόνη όταν εκτελείται;

```
1     //--------------------------------------
2     #include <stdio.h>
3     int main(void) {
4       int i, j;
5       for ( i=0 ; ; i++ ) {
6         for ( j=i ; ; j-- ) {
7           printf("i=%d j=%d\n",i,j);
8           if ( j <= i/2 ) break;
9         }
10        if ( j == 1 ) return 0;
11      }
12      printf("Telos tou programmatos.\n");
13      return 0;
14    }
15    //--------------------------------------
```

Λύση:

Το πρόγραμμα χρησιμοποιεί δύο εντολές for (γραμμές 5 και 6), τη μία μέσα στην άλλη, οι οποίες δεν περιλαμβάνουν συνθήκη τερματισμού (ατέρμονες βρόχοι).

Εσωτερική for:

Ο ακέραιος j ξεκινά με την τιμή i, και μετά από κάθε βρόχο μειώνεται κατά 1.
Τυπώνονται οι τιμές των i και j (γραμμή 7)
Εφόσον το j είναι μικρότερο ή ίσο της ακέραιας διαίρεσης του i δια του 2,
η ροή του προγράμματος επιστρέφει στην εξωτερική for (μετά τη γραμμή 9)
με την εντολή break.

Εξωτερική for:

Ο ακέραιος i ξεκινά με την τιμή 0, και μετά από κάθε βρόχο αυξάνεται κατά 1.
Όταν η ροή του προγράμματος φτάνει στη γραμμή 10, γίνεται έλεγχος της τιμής του j
και αν αυτή είναι ίση με 1, το πρόγραμμα τερματίζεται με κωδικό εξόδου 0,
μέσω της εντολής «return 0».

Οι εντολές των γραμμών 13 και 14 δεν εκτελούνται ποτέ, αφού ο μόνος τρόπος εξόδου από την εξωτερική for είναι μέσω της εντολής «return 0», η οποία τερματίζει το πρόγραμμα.

Όταν εκτελείται, το πρόγραμμα τυπώνει στην οθόνη:

i=0 j=0
i=1 j=1
i=1 j=0
i=2 j=2
i=2 j=1

6.1.4 Δίνεται η αλγεβρική συνάρτηση

$$f(x) = \frac{\sqrt{x(x+2)}}{(x-1)(x-3)}$$

α. Να γίνει διερεύνηση της παραπάνω συνάρτησης (για ποιες τιμές του *x* ορίζεται και παίρνει πραγματικές τιμές). Περιγράψτε την διερεύνηση με συντομία.

β. Με βάση την παραπάνω διερεύνηση να γραφεί ένα πρόγραμμα το οποίο να τυπώνει σε δύο στήλες τις τιμές *x* και *f(x)*, για *x* = -5, -4.5, -4, -3.5, ..., 4, 4.5, 5, (δηλαδή από -5.0 έως 5.0 με βήμα 0.5). Σε περίπτωση που για κάποιο *x* η *f(x)* δεν ορίζεται, στη στήλη *f(x)* του πίνακα να τυπώνεται η λέξη **undefined.**

Λύση:

α. Η πραγματική συνάρτηση *f(x)* ορίζεται όταν η υπόριζη ποσότητα στον αριθμητή δεν είναι αρνητική και όταν ο παρανομαστής δεν μηδενίζεται.
Πρέπει λοιπόν $x \neq 1$ και $x \neq 3$ και $x \notin (-2,0)$.

β.

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

float f(float x);

int main (void) {

  float x;
  for (x=-5.0 ; x<=5.0 ; x+=0.5){
    if ( (x<=-2.0 || x>=0.0) && x!=1.0 && x!=3.0) {
      printf("%f %f\n",x,f(x));
    }
    else {
      printf("%f %s\n",x,"undefined");
    }
  }
}

float f(float x) {

  return sqrt(x*(x+2))/((x-1)*(x-3));

}
```

ΠΑΡΑΤΗΡΗΣΗ: Η συνθήκη στην εντολή `if` θα μπορούσε ισοδύναμα να είναι

```
if ( (x*(x+2.0)>=0.0) && (x-1.0)*(x-3.0)!=0.0) {
```

ή αντίστοιχη.

Όταν αυτό το πρόγραμμα τρέχει τυπώνει:

```
-5.000000 0.080687
-4.500000 0.081312
-4.000000 0.080812
-3.500000 0.078335
-3.000000 0.072169
-2.500000 0.058080
-2.000000 -0.000000
-1.500000 undefined
-1.000000 undefined
-0.500000 undefined
0.000000 0.000000
0.500000 0.894427
1.000000 undefined
1.500000 -3.055050
2.000000 -2.828427
2.500000 -4.472136
3.000000 undefined
3.500000 3.509986
4.000000 1.632993
4.500000 1.030158
5.000000 0.739510
```

6.1.5 Δίνεται η γνησίως φθίνουσα συνάρτηση f(x)

```
float f(float x) {
  return 2.0-x;
}
```

η οποία δύναται να έχει μία ρίζα *ρ* στο διάστημα [*α*, *β*].
Να γραφεί ένα πρόγραμμα το οποίο

α. να λαμβάνει από το πληκτρολόγιο τα *α* και *β*

β. να ελέγχει αν όντως *α*<*β* και να τερματίζει σε περίπτωση που αυτό δεν ισχύει, τυπώνοντας στην οθόνη αντίστοιχο μήνυμα

γ. να ελέγχει αν υπάρχει ρίζα στο διάστημα [*α, β*] και να τερματίζει σε περίπτωση που δεν υπάρχει, τυπώνοντας στην οθόνη αντίστοιχο μήνυμα

δ. να υπολογίζει τη ρίζα *ρ* με ακρίβεια δύο δεκαδικών ψηφίων

ε. να τυπώνει στην οθόνη τη ρίζα *ρ* και την τιμή της συνάρτησης για x=*ρ* με ακρίβεια δύο δεκαδικών ψηφίων.

ΣΗΜΕΙΩΣΗ: Η συνάρτηση f, ως γνησίως φθίνουσα, είναι θετική για $x < \rho$ και αρνητική για $x > \rho$. Στην περίπτωση που στο διάστημα [*α, β*] δεν υπάρχει ρίζα, το γινόμενο f(α) επί f(β) είναι θετικό. Υπολογίστε τη ρίζα *ρ*, αυξάνοντας τη μεταβλητή *x* από *α* έως *β* με βήμα 0.01, έως ότου η συνάρτηση γίνει αρνητική ή μηδέν (αλλαγή προσήμου ή μηδενισμός). Η τιμή αυτή του *x* είναι η ρίζα *ρ* με ακρίβεια 0.01 .

Μια πιθανή λύση είναι:

```
#include <stdio.h>

float f(float x) {
  return 2.0-x;
}

int main(void) {
  float a, b, x, r;

  printf("Dwse ta a kai b: ");
  scanf("%f",&a);
  scanf("%f",&b);
  if (a>=b) {
    printf("Sfalma: a>=b. Eksodos apo to programma.\n");
    return 0;
  }
  if ( f(a)*f(b)>0 ) {
    printf("Den uparxei riza sto diasthma auto. Eksodos apo to programma.\n");
    return 0;
  }
  x=a;
  while ( x<=b ) {
    if ( f(x)<=0 ) {
      r=x;
      break;
    }
    x+=0.01;
  }

  printf("r    = %.2f\n",r);
  printf("f(r) = %.2f\n",f(r));
  return 0;
}
```

6.1.6 Να γράψετε δύο συναρτήσεις:

- η πρώτη να δέχεται ως όρισμα την θερμοκρασία σε βαθμούς Celsius και να επιστρέφει την θερμοκρασία σε βαθμούς Fahrenheit.
- η δεύτερη να δέχεται ως όρισμα την θερμοκρασία σε βαθμούς Fahrenheit και να επιστρέφει την θερμοκρασία σε βαθμούς Celsius.

Δίνεται $\Theta_F=1.8\Theta_C+32$.

Μια πιθανή λύση είναι:

```
float cel_to_fahr(float celsius)
{
  float fahrenheit;
  fahrenheit = 1.8*celsius+32;
  return fahrenheit;
}


float fahr_to_cel(float fahrenheit)
{
  float celsius;
  celsius = (fahrenheit-32.)/1.8;
  return celsius;
}
```

6.1.7 Ένα σώμα εκτελεί πλάγια βολή υπό γωνία θ και με αρχική ταχύτητα v0. Ο χρόνος t κατά τον οποίο το σώμα μένει στον αέρα ισούται με $t=(2\ v_0\ \sin\theta)/(g)$ ενώ το βεληνεκές R της βολής ισούται με $R=(v_0\ \cos\theta)t$ (όπου $g=9.81 m/sec^2$ η επιτάχυνση της βαρύτητας).

α) Να γράψετε μία συνάρτηση

float hronos(float v0, float theta)

η οποία να δέχεται ως ορίσματα την αρχική ταχύτητα v0 (σε m/sec) και την γωνία θ (σε μοίρες), και να επιστρέφει τον χρόνο t κατά τον οποίο το σώμα μένει στον αέρα.

β) Να γράψετε μία συνάρτηση

float velinekes(float v0, float theta, float t)

η οποία να δέχεται ως ορίσματα την αρχική ταχύτητα v0 (σε m/sec), την γωνία θ (σε μοίρες) και τον χρόνο t (σε sec) κατά τον οποίο το σώμα μένει στον αέρα, και να επιστρέφει το βεληνεκές R της βολής.

γ) Να γράψετε ένα πρόγραμμα στο οποίο να εισάγετε από το πληκτρολόγιο την γωνία θ (σε μοίρες) και την αρχική ταχύτητα v0 (σε m/sec) και στη συνέχεια να τυπώνετε τον χρόνο t και το βεληνεκές R της βολής χρησιμοποιώντας τις δύο συναρτήσεις που γράψατε. Η εκτύπωση να γίνει με ακρίβεια δύο δεκαδικών ψηφίων. Στην εκτύπωση να αναγραφούν και μονάδες.

Μια πιθανή λύση είναι:

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

float xronos(float v0, float theta);
float velinekes(float v0, float theta, float t);

int main(void)
{
  float v0,theta,t,R;

  printf("Eisagete tin gwnia Theta kai tin arxiki taxitita v0 : ");
  scanf("%f %f", &theta, &v0);

  t=xronos(v0,theta);
  R=velinekes(v0,theta,t);
  printf("O xronos diarkeias tis bolis einai t=%8.2f sec\n",t);
  printf("To belinekes tis bolis einai       R=%8.2f m\n",R);

  return 0;
}

float xronos(float v0, float theta)
{
  float result, g;
  theta=theta*3.14159/180.;
  g=9.81;
  result=2.*v0*sin(theta)/g;
  return result;
}

float velinekes(float v0, float theta, float t)
{
  float result;
  theta=theta*3.14159/180.;
  result=v0*cos(theta)*t;
  return result;
}
```

6.1.8 Σε ένα πείραμα λαμβάνονται είκοσι (20) μετρήσεις ενός μεγέθους Y συναρτήσει ενός μεγέθους X, και παρατηρείται γραμμική σχέση μεταξύ τους.

Για κάθε πειραματικό σημείο καταγράφονται τα μεγέθη X, Y και σ_Y.

Να γραφεί ένα πρόγραμμα εφαρμογής της μεθόδου ελαχίστων τετραγώνων, το οποίο

α) να δέχεται από το πληκτρολόγιο τις τιμές των X_i, Y_i και σ_{Yi}

β) να υπολογίζει και να τυπώνει στην οθόνη του υπολογιστή τις παραμέτρους α και β της ευθείας ελαχίστων τετραγώνων Y=α+βX, καθώς και τα σφάλματά τους σ_α και σ_β.

Δίνονται οι παρακάτω γενικές σχέσεις. Στην προκειμένη περίπτωση N=20.

$$a=\frac{1}{\Delta}\left(\sum_{i=1}^{N}\frac{X_i^{\,2}}{\sigma_{Yi}^{\,2}}\sum_{i=1}^{N}\frac{Y_i}{\sigma_{Yi}^{\,2}}-\sum_{i=1}^{N}\frac{X_i}{\sigma_{Yi}^{\,2}}\sum_{i=1}^{N}\frac{X_iY_i}{\sigma_{Yi}^{\,2}}\right),\ \sigma_a^{\,2}=\frac{1}{\Delta}\sum_{i=1}^{N}\frac{X_i^{\,2}}{\sigma_{Yi}^{\,2}},$$

$$\beta=\frac{1}{\Delta}\left(\sum_{i=1}^{N}\frac{1}{\sigma_{Yi}^{\,2}}\sum_{i=1}^{N}\frac{X_iY_i}{\sigma_{Yi}^{\,2}}-\sum_{i=1}^{N}\frac{X_i}{\sigma_{Yi}^{\,2}}\sum_{i=1}^{N}\frac{Y_i}{\sigma_{Yi}^{\,2}}\right),\ \sigma_\beta^{\,2}=\frac{1}{\Delta}\sum_{i=1}^{N}\frac{1}{\sigma_{Yi}^{\,2}}\ \text{και}$$

$$\Delta=\sum_{i=1}^{N}\frac{1}{\sigma_{Yi}^{\,2}}\sum_{i=1}^{N}\frac{X_i^{\,2}}{\sigma_{Yi}^{\,2}}-\left(\sum_{i=1}^{N}\frac{X_i}{\sigma_{Yi}^{\,2}}\right)^2$$

Μια πιθανή λύση είναι:

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

int main(void) {
  float x[20], y[20], ey[20],
        S1=0, Sx=0, Sx2=0, Sy=0, Sxy=0,
        D, a, sa, b, sb, sigmaFactor;
  int i;
  // ανάγνωση των δεδομένων από το πληκτρολόγιο
  for (i=0 ; i<20 ; i++) {
    printf("Δώσε τα X, Y και σY του %2dου σημείου: ",i+1);
    scanf("%f %f %f",&x[i],&y[i],&ey[i]);
  }

  // υπολογισμός των αθροισμάτων S1, Sx, Sx2, Sy, Sxy
  for (i=0; i<20; i++) {
    sigmaFactor = 1/ey[i]/ey[i];
    S1  += sigmaFactor;
    Sx  += x[i]*sigmaFactor;
    Sx2 += x[i]*x[i]*sigmaFactor;
    Sy  += y[i]*sigmaFactor;
    Sxy += x[i]*y[i]*sigmaFactor;
  }
  D=S1*Sx2-Sx*Sx;             // παράγοντας D
  b=(S1*Sxy-Sx*Sy)/D;   // κλίση b
  sb=sqrt(S1/D);
  a=(Sx2*Sy-Sx*Sxy)/D;  // τεταγμένη επί την αρχή a
  sa=sqrt(Sx2/D);

  // εκτύπωση αποτελεσμάτων
  printf("Ευθεία ελαχίστων τετραγώνων: y = a + b x\n");
  printf("   τεταγμένη επί την αρχή a = %10f +- %10f\n",a,sa);
  printf("                    κλίση b = %10f +- %10f\n",b,sb);
  return 0;
}
```

6.1.9 Να γραφεί ένα πρόγραμμα στο οποίο

α. να ορίζεται ένας πίνακας 3×3,

β. τα στοιχεία του πίνακα να εισάγονται από το πληκτρολόγιο κατά την εκτέλεση του προγράμματος και

γ. να υπολογίζεται η ορίζουσά του.

Η ορίζουσα ενός πίνακα 3×3 δίνεται από τη σχέση:

$$\begin{vmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{vmatrix} = a_1(b_2c_3 - b_3c_2) - a_2(b_1c_3 - b_3c_1) + a_3(b_1c_2 - b_2c_1)$$

Μια πιθανή λύση είναι:

```
#include <stdio.h>

int main(void) {
  float A[3][3], det;
  int i,j;

  for (i=0 ; i<3 ; i++)
    for (j=0 ; j<3 ; j++) {
      printf("Dose to stoixeio A[%d,%d]: " , i+1 , j+1);
      scanf("%f",&A[i][j]);
    }

  det =   A[0][0] * ( A[1][1]*A[2][2] - A[1][2]*A[2][1] )
        - A[0][1] * ( A[1][0]*A[2][2] - A[1][2]*A[2][0] )
        + A[0][2] * ( A[1][0]*A[2][1] - A[1][1]*A[2][0] ) ;

  printf("Orizousa = %f\n" , det);

  return 0;

}
```

6.1.10 α) Θεωρήστε μια ωμική αντίσταση R=1500 Ω στην οποία εφαρμόζεται συνεχής τάση V. Να γράψετε ένα πρόγραμμα με το οποίο να υπολογίζεται το ρεύμα I (σε Amperes) για τιμές της τάσης V=100, 200, ..., 1000 Volts. Διατηρήστε ακρίβεια τριών δεκαδικών ψηφίων στους υπολογισμούς σας. Δημιουργήστε ένα αρχείο με το όνομα ohmslaw.data και αποθηκεύστε στο εν λόγω αρχείο τις τιμές των ζευγών της τάσης και του αντίστοιχου ρεύματος καθώς και τον αύξοντα αριθμό του κάθε ζεύγους.

β) Γράψτε ένα πρόγραμμα το οποίο θα διαβάζει τις τιμές από το αρχείο ohmslaw.data θεωρώντας ότι δεν γνωρίζετε το πλήθος των ζευγών, και θα υπολογίζει τη καταναλισκόμενη ισχύ στην αντίσταση P(Watts)=V (Volts) I(Amperes) την οποία και θα εκτυπώνετε, στην οθόνη του Η/Υ σας, με τα αναγκαία δεκαδικά ψηφία.

Υπόδειξη: Η συνάρτηση fscanf() επιστρέφει EOF στο τέλος του αρχείου.

Μια πιθανή λύση είναι:

α)

```
#include<stdio.h>

int main(void)
{
  int i;
  float V,R,I;
  FILE *file_pointer;
  file_pointer=fopen("ohmslaw.data","w");

  R=1500.;
  for(i=1;i<=10;i++){
    V=100.*(float)i;
    I=V/R;
    fprintf(file_pointer,"%2d %8.3f %8.3f\n",i,V,I);
  }

  fclose(file_pointer);
  return 0;
}
```

β)

```
#include<stdio.h>

int main(void)
{
  int i;
  float V,I;
  FILE *file_pointer;
  file_pointer=fopen("ohmslaw.data","r");

  while((fscanf(file_pointer,"%d %f %f",&i,&V,&I))!=EOF){
    printf("%2d  %8.3f\n",i,V*I);
  }

  fclose(file_pointer);
  return 0;
}
```

6.1.11 Να γραφεί ένα πρόγραμμα, το οποίο να ελέγχει τη θέση ενός σημείου Σ με συντεταγμένες (x_Σ, y_Σ, z_Σ) ως προς μία σφαίρα κέντρου C με συντεταγμένες (x_c, y_c, z_c) και ακτίνας R. Το πρόγραμμα θα πρέπει να δέχεται από το πληκτρολόγιο

α. τις συντεταγμένες x_Σ, y_Σ και z_Σ του σημείου Σ.
β. τις συντεταγμένες x_c, y_c και z_c του κέντρου C της σφαίρας.
γ. την ακτίνα R της σφαίρας.

Μετά από κατάλληλους υπολογισμούς, το πρόγραμμα θα πρέπει να τυπώνει αν το σημείο Σ βρίσκεται μέσα, έξω ή επάνω στην επιφάνειά της σφαίρας.
Δίνεται η εξίσωση της σφαίρας: $(x-x_c)^2+(y-y_c)^2+(z-z_c)^2=R^2$

Μια πιθανή λύση είναι:

```
#include <stdio.h>

int main(void) {
  float xs,ys,zs, xc,yc,zc, R, left, right;

  printf("Eisagete tis suvtetagmeves tou snmeiou S\n");
  printf("xs ys zs: ");
  scanf("%f %f %f",&xs,&ys,&zs);

  printf("Eisagete tis suvtetagmeves tou kevtrou C tns sfairas\n");
  printf("xc yc zc: ");
  scanf("%f %f %f",&xc,&yc,&zc);

  printf("Eisagete tnv aktiva R tns sfairas: ");
  scanf("%f",&R);

  left = (xs-xc)*(xs-xc) + (ys-yc)*(ys-yc) + (zs-zc)*(zs-zc);
  right=R*R;

  if (left==right)
    printf("To snmeio S brisketai pavw stn sfaira.\n");
  else if (left<right)
    printf("To snmeio S brisketai mesa stn sfaira.\n");
  else
    printf("To snmeio S brisketai eksw apo tn sfaira.\n");

  return 0;
}
```

6.1.12 Μία γωνία στο εξηκονταδικό σύστημα εκφράζεται σε μοίρες(°), λεπτά(′) και δευτερόλεπτα(″) (πχ. 10° 20′ 35″, όπου ° : μοίρες, ′ : λεπτά, ″ : δευτερόλεπτα), όπου 1°=60′ και 1′=60″. Γράψτε ένα πρόγραμμα στο οποίο:

- Να αναπτύξετε την κατάλληλη δομή που περιγράφει γωνίες στο εξηκονταδικό σύστημα.
- Με βάση αυτή να δημιουργήσετε τις δομές για δύο γωνίες, τα δεδομένα των οποίων θα εισάγονται από το πληκτρολόγιο.
- Να αθροίσετε τα δεδομένα των δύο γωνιών και να τα αποθηκεύσετε σε νέα κατάλληλη δομή.

Αφού βρεθεί το άθροισμα να το μετατρέψετε σε γωνία που εκφράζεται στο δεκαδικό σύστημα (πχ. 10.451°). Το άθροισμα να τυπώνεται τόσο στο εξηκονταδικό σύστημα όσο και στο δεκαδικό.

Μια πιθανή λύση είναι:

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

int main(void)
{
struct ang{
      int moi;
      int lep;
      int deu;
}a,b,sum;
float ssum;
      printf("moi1?");
      scanf("%d",&a.moi);
      printf("lep1?");
      scanf("%d",&a.lep);
      printf("deu1?");
      scanf("%d",&a.deu);
      printf("moi2?");
      scanf("%d",&b.moi);
      printf("lep2?");
      scanf("%d",&b.lep);
      printf("deu2?");
      scanf("%d",&b.deu);
sum.moi=a.moi+b.moi;
sum.lep=a.lep+b.lep;
if(sum.lep>=60){
      sum.moi+=1;
      sum.lep-=60;
}
sum.deu=a.deu+b.deu;
if(sum.deu>=60){
      sum.lep+=1;
      sum.deu-=60;
      if(sum.lep>=60){
            sum.moi+=1;
```

```
            sum.lep-=60;
      }
}
ssum=(float)sum.moi+(float)sum.lep/60.+(float)sum.deu/3600.;
printf ("athroisma sto eksikontadiko\t%d %d %d\n", sum.moi,sum.lep,sum.deu);
printf ("athroisma sto dekadiko\t%f\n",ssum);

return 0;
}
```

6.1.13 Μία ομάδα φοιτητών εκτελεί το πείραμα ηλεκτρισμού για την επαλήθευση του νόμου του Ohm μεταβάλλοντας και μετρώντας την τάση και το ρεύμα στα άκρα μιας αντίστασης R. Οι μετρήσεις που προκύπτουν είναι οι ακόλουθες: V=1.1, 1.9, 3.0, 4.1, 4.8, 6.2, 7.0, 8.0, 8.9, 10.0 σε [Volts] για I=0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0 σε [A] αντίστοιχα. Η γραφική παράσταση της τάσης V ως συνάρτηση του ρεύματος I είναι ευθεία της μορφής: y=α+βx. Να γραφεί ένα πρόγραμμα σε γλώσσα C το οποίο να αποθηκεύει τις παραπάνω μετρήσεις σε κατάλληλους πίνακες. Στη συνέχεια να υπολογίζει με την μέθοδο ελαχίστων τετραγώνων και να τυπώνει τους συντελεστές α και β της ανωτέρω ευθείας. Εκτυπώστε την τιμή της R σε [Ohms].
Κρατείστε ακρίβεια ενός δεκαδικού. Δίνονται:

$\alpha = \frac{\Sigma y \Sigma x^2 - \Sigma x \Sigma xy}{n\Sigma x^2 - (\Sigma x)^2}$ και $\beta = \frac{n\Sigma xy - (\Sigma x)(\Sigma y)}{n\Sigma x^2 - (\Sigma x)^2}$, με n τον αριθμό των μετρήσεων.

Μια πιθανή λύση είναι:

```
#include<stdio.h>

int main(void)
{
  float V[10]={1.1,1.9,3.0,4.1,4.8,6.2,7.0,8.0,8.9,10.0};
  float I[10]={0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0};
  float sx, sy, sxy, sx2, a, b;
  int i;

  sx=0;
  sy=0;
  sxy=0;
  sx2=0;
  for(i=0;i<10;++i){
    sx=sx + I[i];
    sy=sy + V[i];
    sxy=sxy + I[i]*V[i];
    sx2=sx2 + I[i]*I[i] ;
  }

  a=(sy*sx2-sx*sxy)/(10.*sx2-sx*sx);
  b=(10.*sxy-sx*sy)/(10.*sx2-sx*sx);

  printf("a=%.1f\n",a);
  printf("b=%.1f\n",b);
  printf("R=b=%.1f Ohm\n",b);

}
```

6.1.14 Να γράψετε ένα πρόγραμμα που να διαβάζει και να ελέγχει τα στοιχεία ενός αρχείου με όνομα organa.dat στο οποίο είναι καταχωρημένα τα όργανα ενός εργαστηρίου. Κάθε γραμμή καταχώρησης αντιστοιχεί σε ένα όργανο και περιέχει με την σειρά: τον αύξοντα αριθμό (τύπου int), την οικογένεια οργάνων στην οποία ανήκει (τύπου char[20]), το όνομα του οργάνου (τύπου char[20]) και την θέση που βρίσκετε το όργανο στο εργαστήριο (τύπου char[10]).
Συγκεκριμένα όταν το πρόγραμμα λειτουργεί θα δέχεται από το πληκτρολόγιο έναν ακέραιο αριθμό που μπορεί να είναι:

- -999 με τον οποίο τερματίζεται η λειτουργία του.
- Ένας θετικός αριθμός ο οποίος αντιστοιχεί στον αύξοντα αριθμό κάποιου οργάνου και σε αυτή την περίπτωση διαβάζει την αντίστοιχη γραμμή από το αρχείο καταχώρησης των οργάνων, εκτυπώνει την οικογένεια, το όνομα καθώς και την θέση του οργάνου στο εργαστήριο και να επανέρχεται στην αρχική κατάστασή του περιμένοντας τον επόμενο αριθμό από το πληκτρολόγιο. Εάν το όργανο με τον αντίστοιχο αύξοντα αριθμό δεν είναι καταχωρημένο στο αρχείο το πρόγραμμα εκτυπώνει αντίστοιχο μήνυμα και επανέρχεται στην αρχική κατάστασή του περιμένοντας τον επόμενο αριθμό από το πληκτρολόγιο.

Διευκρινίζεται ότι δεν είναι γνωστός ο αριθμός των οργάνων που είναι καταχωρημένα στο αρχείο καταχώρησης, Υπενθυμίζεται ότι η συνάρτηση fscanf() επιστρέφει EOF στο τέλος του αρχείου.

Μια πιθανή λύση είναι:

```
#include<stdio.h>

int main(void)
{
  struct organa_list{         /* Perigrafi Katallilis Domis */
    int aa;
    char family[20];
    char name[20];
    char pos[20];
  };
  struct organa_list organo;  /* Dimiourgia tis Domis */
  int iaa;
  FILE *f_p;

  for(;;){                    /* Dimiourgia atermona brogxou */
    printf("Doste aa eite -999 gia termatismo:");
    scanf("%d",&iaa);
    if(iaa==-999){
      return 0;
    }

    f_p=fopen("organa.dat","r");
    while(fscanf(f_p,"%d%s%s%s",&organo.aa, &organo.family, &organo.name,
                 &organo.pos)!=EOF)
      {
        if(organo.aa==iaa){
          printf("Oikogeneia Organou :%s\n",organo.family);
          printf("Onoma Organou      :%s\n",organo.name);
          printf("Thesi Organou      :%s\n",organo.pos);
          break;
        }
        else {
          printf("Den yparxei kataxwrisi organou.\n");
          break;
        }
      }
    fclose(f_p);
  }
  return 0;
}
```

6.1.15 Να γραφεί ένα πρόγραμμα που θα παράγει τυχαία 2500 αριθμούς με ακέραιες τιμές 0 ή 1 και θα γεμίζει με αυτούς έναν πίνακα 50x50 στοιχείων.
Στη συνέχεια να βρίσκει και να τυπώνει στην οθόνη τη μέση τιμή των στοιχείων της διαγωνίου του πίνακα.
Ακολούθως στις περιττές στήλες του πίνακα να ανταλλάσσει τις τιμές των 0 με 1 και τις τιμές των 1 με 0 και να βρίσκει και να εκτυπώνει στην οθόνη την μέση τιμή των στοιχείων της διαγωνίου του.
Τι τιμή αναμένεται να έχουν οι παραπάνω δύο μέσες τιμές.
Υπενθυμίζεται ότι το αρχείο stdlib.h παρέχει τις συναρτήσεις srand() και rand() και επιπλέον το rand() επιστρέφει τιμές από το 0 έως το RAND_MAX.

Μια πιθανή λύση είναι:

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

int main(void){
  int a[50][50];
  int i,j;
  float x, sum;
  srand(123);
  for(i=0;i<50;i++){
    for(j=0;j<50;j++){
      x=(float)rand()/(float)RAND_MAX;
      if(x>=0.5)
         a[i][j]=1;
      else
         a[i][j]=0;
    }
  }

  sum=0.;
  for(i=0;i<50;i++){
    sum=sum+(float)a[i][i];
  }
  printf("mesi timi=%f\n", sum/50.);

  for(i=0;i<50;i++){
    for(j=0;j<50;j=j+2){
      if(a[i][j]==0)
         a[i][j]=1;
      else
         a[i][j]=0;
    }
  }

  sum=0.;
  for(i=0;i<50;i++){
    sum=sum+(float)a[i][i];
  }
  printf("nea mesi timi=%f\n", sum/50.);

  return 0;
}
```

6.2 Προβλήματα.

6.2.1 Περιγράψτε σε συντομία τη λειτουργία του παρακάτω προγράμματος. Τι ακριβώς τυπώνει στην οθόνη το πρόγραμμα όταν εκτελείται;

```
#include<stdio.h>

int main(void){
  int i;

  i=12;
  while (i>0){
    if(i%4!=1)
      printf(" %d",i);
    --i;
  }
  printf("\n i=%d \n",i);

  return 0;
}
```

6.2.2 Περιγράψτε σε συντομία τη λειτουργία του παρακάτω προγράμματος. Τι ακριβώς τυπώνει στην οθόνη το πρόγραμμα όταν εκτελείται;

```
#include<stdio.h>
int main(void){
        int i, a, b;
        char c;
        char s[]=”University of Ioannina”;
        a=0;
        b=0;
        i=0;
        while(  (c=s[i] ) != ‘\0’ ){
                if(c==’i’) a++;
                if(c==’a’) b++;
                i++;
        }
        printf(“a=%d\n”,a);
        printf(“b=%d\n”,b);
        printf(“i=%d\n”,i);
        return 0;
}
```

6.2.3 Να γραφεί ένα πρόγραμμα το οποίο να υπολογίζει τις όποιες πραγματικές ρίζες της διτετράγωνης εξίσωσης $\alpha\, x^4 + \beta\, x^2 + \gamma = 0$. Στο πρόγραμμα θα πρέπει

α. οι παράμετροι α, β και γ να εισάγονται από το πληκτρολόγιο κατά την εκτέλεση του προγράμματος.

β. να γίνεται διερεύνηση του αριθμού των μιγαδικών και του αριθμού των πραγματικών ριζών και να τυπώνονται κατάλληλα μηνύματα στην οθόνη.

γ. σε περίπτωση που υπάρχουν πραγματικές ρίζες, να υπολογίζονται και να τυπώνονται στην οθόνη.

Δίνεται: $\kappa\, \rho^2 + \lambda\, \rho + \mu = 0 \quad \Rightarrow \quad \rho_{1,2} = \dfrac{-\lambda \pm \sqrt{\Delta}}{2\,\kappa}, \quad \Delta = \lambda^2 - 4\,\kappa\,\mu$

6.2.4 α. Να γραφεί μία συνάρτηση σε C που να ονομάζεται MyFunction και που να περιγράφει την πραγματική συνάρτηση

$$f(x) = \begin{cases} \dfrac{\sqrt{x(x+4)}}{(x-2)(x+3)}, & x < -3 \\ \dfrac{\ln x}{(x-2)(x+3)}, & -3 \le x < 2 \\ \dfrac{x+1}{(x-2)(x+3)}, & x \ge 2 \end{cases}$$

όπου x πραγματικός αριθμός.

β. Να βρεθεί για ποιες τιμές του *x* η f(x) παίρνει πραγματικές τιμές.

γ. Να γραφεί πρόγραμμα το οποίο να χρησιμοποιεί τη συνάρτηση MyFunction και να τυπώνει σε δύο στήλες τις τιμές *x* και *f(x)*, για *x* = -15, -14.5, -14, -13.5, ..., 14, 14.5, 15, (δηλαδή από -15.0 έως 15.0 με βήμα 0.5).

- Οι στήλες του πίνακα να στοιχίζονται ακριβώς όπως στο παράδειγμα που ακολουθεί.
- Οι τιμές των *x* να τυπώνονται με ακρίβεια δύο δεκαδικών ψηφίων.
- Οι τιμές των *f(x)* να τυπώνονται με ακρίβεια τριών δεκαδικών ψηφίων.
- Σε περίπτωση που για κάποιο *x* η *f(x)* δεν ορίζεται (δεν έχει πραγματική τιμή ή απειρίζεται), στη στήλη *f(x)* του πίνακα να τυπώνεται η λέξη undefined.

```
12345678901234567890123456789012345678901234567890   ← βοηθητική γραμμή
       x          f(x)
   -1.00         3.123
   -0.50         1.345
    0.00     undefined
    1.00        10.234
   11.50         3.900
12345678901234567890123456789012345678901234567890   ← βοηθητική γραμμή
```

6.2.5 Δίνεται η ακόλουθη αναδρομική σχέση:

$$a_n = \frac{a_{n-1} * n!}{(a_{n-1}+1)^2}, \qquad \text{με } a_0 = 1, \text{ η οποία ορίζεται για n>0}$$

Να γραφεί ένα πρόγραμμα το οποίο να υπολογίζει τον n-οστο όρο (όπου το n να εισάγεται από το πληκτρολόγιο). Προστατέψτε το πρόγραμμα από λανθασμένες τιμές του n. Για το παραγοντικό ισχύει: n! = 1*2*3*…*(n-1)*n με 0! = 1

6.2.6 α) Οι αριθμοί του Fibonacci είναι μέλη μιας ακολουθίας αριθμών στην οποία ο κάθε αριθμός είναι ίσος με το άθροισμα των δύο προηγουμένων. Με λίγα λόγια

$F_n = F_{n-1} + F_{n-2}$

όπου το F_n αναφέρεται στον n-οστό αριθμό του Fibonacci. Εξ ορισμού οι δύο πρώτοι αριθμοί του Fibonacci ισούνται με το 1, είναι δηλ: $F_1=F_2=1$

Γράψτε ένα πρόγραμμα σε C το οποίο να υπολογίζει και να εκτυπώνει τους n πρώτους αριθμούς του Fibonacci. Το n είναι ένας ακέραιος αριθμός μεγαλύτερος του δύο ο οποίος να εισάγεται από το πληκτρολόγιο. Προστατέψτε το πρόγραμμά σας από λανθασμένη τιμή του n. Η εκτύπωση να γίνεται όπως παρακάτω:

n	Fn
3	2
4	3
5	5
6	8

β) Με βάση το παραπάνω πρόγραμμα να γράψετε μια συνάρτηση int fibonacci(int n)
η οποία να δέχεται ως όρισμα έναν ακέραιο n>=1 και να επιστρέφει τον n-οστό αριθμό του Fibonacci για n=1,2,3,4....

6.2.7 Χρησιμοποιώντας την συνάρτηση rand() η οποία εμπεριέχεται στο αρχείο επικεφαλίδας <stdlib.h> να γράψετε ένα πρόγραμμα στο οποίο να παράγονται εκατό τυχαίοι αριθμοί τύπου float από το 30 έως το 40. Αποθηκεύστε τους σε έναν κατάλληλο πίνακα. Στη συνέχεια υπολογίστε τον μέσο των αριθμών. Τυπώστε το αποτέλεσμα.
Δίνονται:
RAND_MAX : ο μέγιστος ακέραιος
int rand(void) : η συνάρτηση επιστρέφει έναν τυχαίο αριθμό από 0 έως RAND_MAX.
void srand(unsigned int seed) : η συνάρτηση αυτή εκτελείται μια φορά στην αρχή του προγράμματος και αρχικοποιεί την γεννήτρια των τυχαίων αριθμών σύμφωνα με τον ακέραιο seed.

6.2.8 Τα στοιχεία για το «επώνυμο», «όνομα», «βαθμός» μιας τάξης 30 μαθητών είναι αποθηκευμένα στο αρχείο με το όνομα «taxis.data» σε μια σειρά για κάθε μαθητή με ενδιάμεσα κενά. Το «επώνυμο» και το «όνομα» έχουν 20 το πολύ χαρακτήρες ενώ ο «βαθμός» είναι πραγματικός αριθμός. Να γραφεί πρόγραμμα το οποίο να διαβάζει τα στοιχεία για όλους τους μαθητές και να τα αποθηκεύει σε κατάλληλη δομή. Στη συνέχεια με τη βοήθεια της δομής να εντοπίζονται και να τυπώνονται στην οθόνη τα στοιχεία του μαθητή με τον μεγαλύτερο βαθμό.
Σημείωση: Οι βαθμοί είναι διαφορετικοί για κάθε μαθητή.

6.2.9 Η τάση $V_C(t)$ στα άκρα ενός πυκνωτή C, ο οποίος εκφορτίζεται μέσω μίας ωμικής αντίστασης R, δίνεται από τη σχέση $V_C(t)=V_0\, e^{-t/(RC)}$, όπου V_0 τάση στα άκρα του για t=0. Χρησιμοποιώντας μεταβλητές διπλής ακρίβειας για τις φυσικές ποσότητες, να γραφεί ένα πρόγραμμα το οποίο:

α. Να δέχεται από το πληκτρολόγιο το V_0 σε V, το R σε MΩ και το C σε μF. Πριν από την εισαγωγή της κάθε τιμής, να τυπώνεται στην οθόνη κατάλληλο μήνυμα, π.χ. «Dwse to Vo se Volts: ».

β. Να τυπώνει ανά μία γραμμή το χρόνο t σε ms, την τάση V_C σε mV και το ρεύμα I_C σε nA, για χρόνους από t=0 έως και t=1000 ms, με βήμα 5 ms. Η μορφοποίηση της γραμμής να έχει τη μορφή των ακόλουθων τεσσάρων γραμμών:

```
   0ms   10000.00mV    1000.00nA
  10ms    9999.00mV     999.90nA
 100ms    9990.00mV     999.00nA
1000ms    9900.50mV     990.05nA
nnnn^^^nnnnnnnnnn^^^nnnnnnnnnn^^      ← βοηθητική γραμμή
```

(ΠΡΟΣΟΧΗ στις μετατροπές μονάδων!)

6.2.10 Να γράψετε ένα πρόγραμμα στο οποίο να κατασκευάζετε μια βάση δεδομένων του μητρώου γεννήσεων ενός Δήμου. Στην βάση δεδομένων θα πρέπει να περιέχονται: Το Επώνυμο, Όνομα, Όνομα Πατρός, Έτος γέννησης, Μήνας γέννησης και Ημέρα γέννησης όλων των δημοτών και να αποθηκεύονται σε κατάλληλη δομή. Αρχικοποιήστε την δομή σας για 5 τυχαίους δημότες με τα πλήρη στοιχεία τους. Στη συνέχεια το πρόγραμμα να ρωτά για την εισαγωγή από το πληκτρολόγιο του έτους γέννησης και με την είσοδό του να τυπώνει τα πλήρη στοιχεία των δημοτών που γεννήθηκαν αυτό το έτος. Προστατεύστε το πρόγραμμά σας από την εισαγωγή ετών μεγαλύτερων του 2005 και μικρότερων του 1900.

6.2.11 **α.** Να γραφεί μία συνάρτηση σε C που να ονομάζεται MyFunction και που να περιγράφει την πραγματική συνάρτηση

$$f(x)=\begin{cases}\dfrac{\sqrt{x(x+4)}}{(x-2)(x+3)}, & x<-3\\ \dfrac{\ln x}{(x-2)(x+3)}, & -3\le x<2\\ \dfrac{x+1}{(x-2)(x+3)}, & x\ge 2\end{cases}$$, όπου x πραγματικός αριθμός.

β. Να βρεθεί για ποιες τιμές του *x* η f(x) παίρνει πραγματικές τιμές.

γ. Να γραφεί πρόγραμμα το οποίο να χρησιμοποιεί τη συνάρτηση MyFunction και να τυπώνει σε δύο στήλες τις τιμές *x* και *f(x)*, για *x* = -15, -14.5, -14, -13.5, ..., 14, 14.5, 15, (δηλαδή από -15.0 έως 15.0 με βήμα 0.5).

- Οι στήλες του πίνακα **<u>να στοιχίζονται ακριβώς</u>** όπως στο παράδειγμα που ακολουθεί.
- Οι τιμές των *x* να τυπώνονται με ακρίβεια δύο δεκαδικών ψηφίων.
- Οι τιμές των *f(x)* να τυπώνονται με ακρίβεια τριών δεκαδικών ψηφίων.
- Σε περίπτωση που για κάποιο *x* η *f(x)* δεν ορίζεται (δεν έχει πραγματική τιμή ή απειρίζεται), στη στήλη *f(x)* του πίνακα να τυπώνεται η λέξη **undefined**.

```
12345678901234567890123456789012345678901234567890   ← βοηθητική γραμμή
       x          f(x)
   -15.00         3.123
   -14.50         1.345
   -14.00     undefined
   -13.50        10.234
   -13.00         3.900
12345678901234567890123456789012345678901234567890   ← βοηθητική γραμμή
```

6.2.12 Να γραφεί ένα πρόγραμμα το οποίο να παράγει τυχαίες συντεταγμένες μεταξύ 0.0 και 1.0 για δέκα διανύσματα **a** (με συντεταγμένες a_x, a_y, a_z) και δέκα διανύσματα **b** (με συντεταγμένες b_x, b_y, b_z). Στη συνέχεια να αποθηκεύει τα διανύσματα σε κατάλληλες δομές. Με την βοήθεια των δομών να υπολογίζει για κάθε ζεύγος διανυσμάτων **a** και **b** τις συντεταγμένες του διανυσματικού αθροίσματος **c=a+b**, το οποίο επίσης να αποθηκεύει σε ανάλογες δομές. Τέλος να τυπώνει σε μία σειρά τις συντεταγμένες των διανυσμάτων **a** και **b** ανά ζεύγος και το **μέτρο** του αντίστοιχου διανύσματος **c**.