Ροή Προγράμματος

λογικές αποφάσεις και βρόγχοι επανάληψης

Μάθημα 3

Εντολές και μπλοκ

Κάθε πρόταση στο πρόγραμμα που τελειώνει σε ελληνικό ερωτηματικό (;) αποτελεί μια **εντολή**.
Οι εντολές συμπεριλαμβάνουν δηλώσεις μεταβλητών, αναθέσεις τιμών, κάλεσμα συναρτήσεων κ.α.

Πολλές εντολές μπορούν να ομαδοποιούνται σε μια **σύνθετη εντολή** ή ένα **μπλοκ**. Η ομαδοποίηση αυτή γίνεται με τα άγκιστρα { }

```
main()
{
        double a,b,c,theta,phi;
        a=3.5; b=7.8; /* endeiktikes times */
        c=sqrt(a*a+b*b);
        phi=atan(b/a)*180/PI;
        theta=90-phi;
        printf("c=%f   theta=%5f   phi=%f \n\n\n",c,theta,phi);
        /* Meros ii */
        {
              c=10.0; phi=30;  /* endeiktikes times */
              phi=phi*PI/180;
              a=c*cos(phi); b=c*sin(phi);
        }
        printf("a=%f   b=%f \n",a,b);

        getchar();
}
```

Μπλοκ 1

Μπλοκ 2

Παρακάτω όταν αναφερόμαστε σε «εντολή» θα συμπεριλαμβάνουμε και τα «μπλοκ»

Μέρος 1ο

Λογικές Αποφάσεις

Λογικές Αποφάσεις
(if – else)

***συνθήκη* = παράσταση με τιμή bool**

```
.................
if(συνθήκη)  εντολή1
        else εντολή2
..................
```

Αν η συνθήκη είναι αληθής (1) εκτελείται η *εντολή1*, αν είναι ψευδής (0) εκτελείται η *εντολή2*

- **Αντί απλών εντολών μπορούμε να έχουμε σύνθετες εντολές (μπλοκ)**
- **Το τμήμα `else` είναι προαιρετικό.**

Παράδειγμα 1

```
if(x>=0) printf("The number x is possitive");
         else printf("The number x is negative");
```

Ένας λογικός έλεγχος

Παράδειγμα 2

```
if(x>0) printf("The number x is possitive");
if(x<0) printf("The number x is negative");
if(x==0) printf("The number is zero");
```

Τρεις λογικοί έλεγχοι

Παράδειγμα 3

```
if(x>0) printf("The number x is possitive");
        else if(x<0) printf("The number x is negative");
             else printf("The number is zero");
```

Δύο λογικοί έλεγχοι

Η δομή `else-if`

Λογικές Αποφάσεις (else-if)

Διακλαδωμένη απόφαση else-if

```
.................
if(συνθήκη1) εντολή1
        else if(συνθήκη2) εντολή2
        else if(συνθήκη3) εντολή3
        ..........................
         else if(συνθήκηN) εντολήN
         else εντολή(Τελική)
...................
```

- Οι συνθήκες υπολογίζονται με τη σειρά Όταν υπολογιστεί η πρώτη αληθής συνθήκη εκτελείται η αντίστοιχη εντολή και η αλυσίδα `else-if` τερματίζεται
- Η τελική εντολή (μετά το τελευταίο `else` εκτελείται όταν καμία συνθήκη δεν είναι αληθής.
- Το τελευταίο `else` είναι προαιρετικό

```
if(x>0) printf("The number x is possitive");
        else if(x<0) printf("The number x is negative");
             else printf("The number is zero");
```

Δες κώδικα prog31elseif.c

Λογικές Αποφάσεις (switch)

Η δομή switch

```
switch (k) {
        case τιμή1 : εντολή 1; break;
        case τιμή2 : εντολή 2; break;
        ..............................
         case τιμή n : εντολή n; break;
         default  : εντολή(Τελική)
}
```

- Το k αντιστοιχεί σε ακέραια τιμή
- Εκτελείται η περίπτωση (case) η τιμή της οποίας ισούται με k
- Αν η k δεν ισούται με καμία τιμή από τις δηλωμένες περιπτώσεις, τότε εκτελείτε η εντολή της περίπτωσης default.
- Η break σπάει τη δομή – μεταφερόμαστε εκτός της δομής switch.
- Αν δεν δοθεί εντολή break, συνεχίζει η εκτέλεση και των εντολών των επόμενων περιπτώσεων.

```
c=getchar();
switch(c) {
          case '0': printf("You pressed zero\n"); break;
          case '1': printf("You pressed one\n"); break;
          case '2': printf("You pressed two\n"); break;

          case '9': printf("You pressed nine\n"); break;
          default: printf("You didn't pressed a number\n");
}
```

Δες κώδικα prog32switch.c

Ασκήσεις

C3.10. Να γίνει ένα πρόγραμμα που να υπολογίζει τις ρίζες ενός τριωνύμου

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

Δήλωσε ως μεταβλητές τις a,b,c (συντελεστές τριωνύμου),
Δήλωσε την D (για διακρίνουσα)
Δήλωσε τις x1, x2 (για τις ρίζες)
Θέσε τιμές στα a,b,c
Υπολόγισε D=b^2-4ac
Αν D<0 τότε Εκτύπωσε μήνυμα «Δεν υπάρχουν ρίζες»
Αλλιώς {
υπολόγισε x1= ……
υπολόγισε x2= …….
Εκτύπωσε τα x1 και x2
}
Τέλος προγράμματος

C3.11. Να ξαναγίνει το πρόγραμμα της άσκησης 3.10 ώστε να εκτυπώνει και τις μιγαδικές ρίζες, όταν υπάρχουν.

Μέρος 2ο

Επαναληπτικές διαδικασίες - Βρόχοι

Επαναληπτικές Διαδικασίες - Βρόχος `while`

```
while(συνθήκη)
                εντολή
```

Η εντολή εκτελείται συνεχώς όσο η συνθήκη είναι αληθής

- Για να σταματήσει ο βρόχος θα πρέπει στην εντολή να μεταβάλλεται η συνθήκη, ώστε αυτή κάποια στιγμή να γίνει ψευδής και να έχουμε έξοδο από τον βρόχο.
- Για να εκτελεστεί τουλάχιστον μια φορά η εντολή θα πρέπει αρχικά η συνθήκη να είναι αληθής.

```c
/* prog35Bwhile.c */

#include <stdio.h>
#include <conio.h>

main()
{
      char a='5';
      char b;
      printf("Mantepse to <a> \n\n");
      printf("Dwse haraktira : "); b=getche();
      while(b!=a)
      {
          printf("\n patises to %c . Lathos!\n",b);
          printf("Dwse haraktira : "); b=getche();
      }
      printf("\n Bravo, brikes to swsto\n");
      getchar();
}
```

Αρχικοποίηση συνθήκης

Συνθήκη επανάληψης

Μεταβολή συνθήκης

Επαναληπτικές Διαδικασίες - Βρόχος `for`

`for(` *Α*; *Β*; *Γ* `)`
εντολή

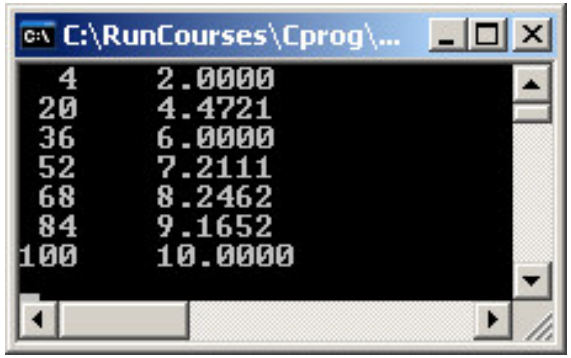
Α,Β,Γ : παραστάσεις
Α: Αρχικοποίηση συνθήκης επανάληψης
Β: Συνθήκη επανάληψης
Γ: Αλλαγή συνθήκης επανάληψης

Η επανάληψη for χρησιμοποιείται συνήθως με «*μετρητή*», *Π.χ.*

`for( i=i1; i<i2; i+=d )` εντολή

`i` : μετρητής (συνήθως ακέραιος)
`i1`: αρχική τιμή
`i2`: μέγιστη τιμή
`d` : βήμα μεταβολής τιμής

```
int i; double x;
for(i=4; i<=110; i+=16) {
     x=sqrt(i);
     printf("%3d     %6.4f\n",i,x);
}
```



Ισοδύναμος βρόχος με "while"

```
int i; double x;
i=4;
while(i<=110){
     x=sqrt(i);
     printf("%3d     %6.4f\n",i,x);
     i+=16;
}
```

Επαναληπτικές Διαδικασίες - Βρόχος do-while

```
do
    εντολή
while(συνθήκη);
```

Η εντολή εκτελείται συνεχώς όσο η συνθήκη είναι αληθής

- Για να σταματήσει ο βρόχος θα πρέπει στην εντολή να μεταβάλλεται η συνθήκη, ώστε αυτή κάποια στιγμή να γίνει ψευδής και να έχουμε έξοδο από τον βρόχο.
- η εντολή εκτελείται τουλάχιστον μια φορά, αφού η συνθήκη ελέγχεται στο τέλος.

```c
/* prog37dowhile.c */
#include <stdio.h>
#include <conio.h>

main()
{
      long unsigned int n=1;
      int k=0; char c;
      printf("Powers of 2 - press ESC to exit\n");
      do{
          n*=2;
          printf("2^%d  = %lu\n",++k,n);
          c=getch();
      } while(c!=27&&k<31);
      if(k>=31) printf("Number out of range\n");
      printf("press Enter");  getchar();
}
```

Ατέρμονοι βρόχοι

```
for(;;);
```

```
while(1){ };
```

```
do {} while(1);
```

Βρόχοι που δεν τερματίζουν ποτέ.

- Για τις εφαρμογές κονσόλας, η εφαρμογή τερματίζει με Ctrl+Break
- Ένας ατέρμονος βρόχος μπορεί να χρησιμοποιηθεί σε ένα πρόγραμμα σε συνδυασμό με την εντολή `Break`.
- Γενικά, ατέρμονοι βρόχοι δημιουργούνται **λόγω λάθους** στον σχεδιασμό ή στην υπολοποίηση του προγραμματος

```
short int n=1;
 while(n<40000) {
      printf("%d\n",n);
      n*=2;
 }
```

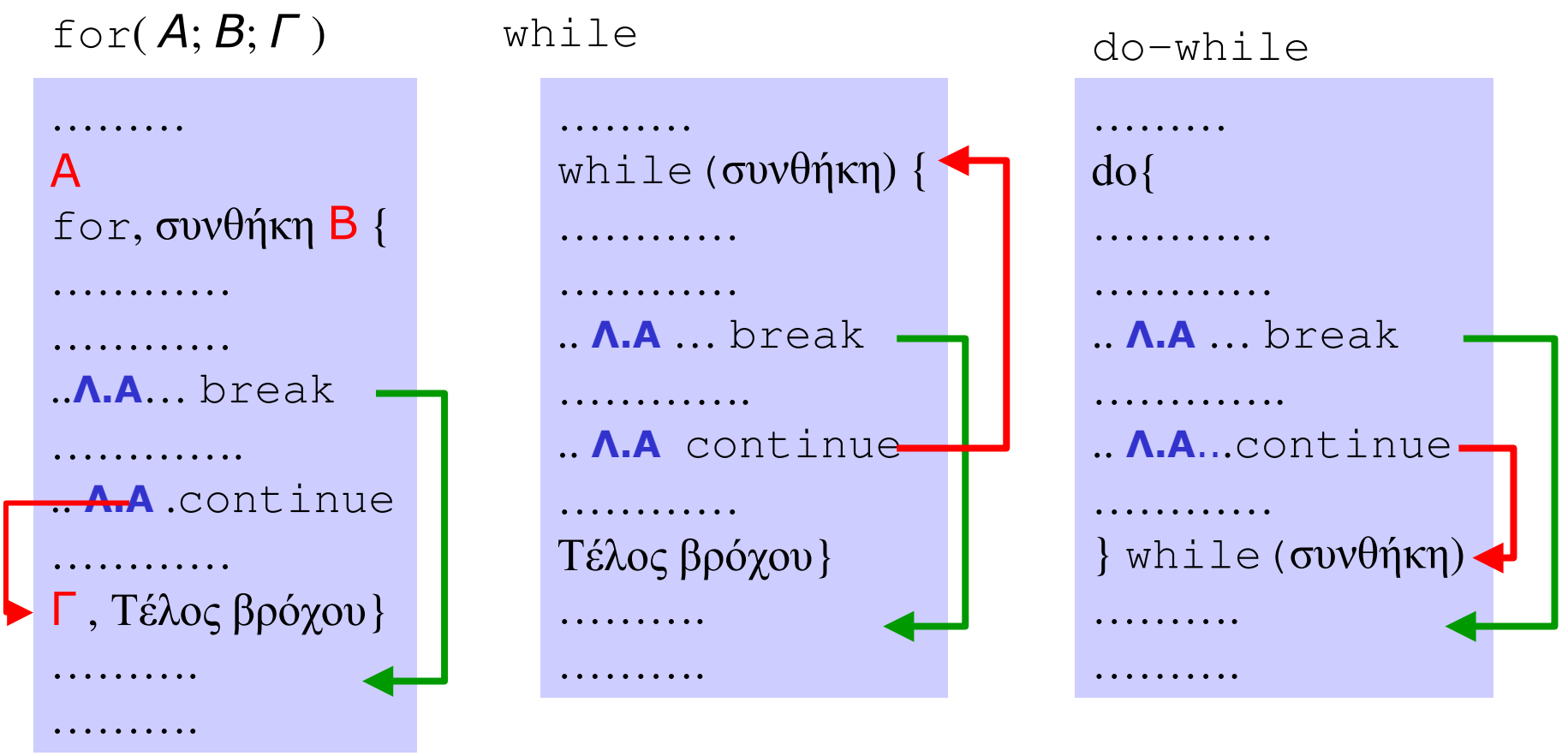
```
for(n=1;n<100;n++) printf("%d \n",n--);
```

```
char c;
do{
      c=getch();
}while(c!=27||c!='A');
```

Εντολές break και continue

- Η εντολή break προκαλεί έξοδο από το βρόχο
- Η εντολή continue μεταφέρει τη ροή στον έλεγχο της συνθήκης του βρόχου

** Οι εντολές πρέπει να βρίσκονται μέσα σε βρόχο και εν'γένει μετά από μια λογική απόφαση (**Λ.Α**.) if-else



Παραδείγματα

1. Δώστε έναν θετικό ακέραιο αριθμό *n* και υπολογίστε το παραγοντικό του *f=n*!

```c
main1()  /* code 1 */
{
    unsigned int k,n,f;
    n=10;    /*θα υπολογίσουμε το n! */
    f=1;     /*f: μεταβλητή για το αποτέλεσμα */
    for(k=1;k<=n;k++) f=f*k;    /* k μετρητής */
    printf("%d!=%lu",n,f);
    getchar();
}
```

```c
main()  /*code 2 */
{
      unsigned int n=0, f=1;
      printf("%d!=",n);
      while(n>1) f*=n--;
      printf("%lu",f);
      getchar();
}
```

2. Δώστε έναν θετικό ακέραιο αριθμό και υπολογίστε αν είναι πρώτος ή όχι και γιατί.

Δες κώδικα primes.c

Δώσε ακέραιο *N*
Επανέλαβε από *I*=2 έως *I*<=*N*/2 (I ένας μετρητής)
  Αν *N*%I==0 διέκοψε την επανάληψη γιατί το N δεν είναι πρώτος.
Τέλος Επανάληψης
Αν *N*%*I*==0 **Εκτύπωσε** Μήνυμα1 (δεν είναι πρώτος)
Αλλιώς Εκτύπωσε Μήνυμα2 (είναι πρώτος)

```c
/* Σύντομος κώδικας primes0.c*/
#include <stdio.h>
main()
{
     int i, N=525;
     for(i=2;i<=N/2;i++) if(!(N%i)) break;
     if(N%i) printf("The number %d is prime!\n",N);
     else printf("The number %d is not prime, is devided by %d\n",N,i);
     getchar();
}
```

Παραδείγματα

3. Υπολογίστε τη τιμή της συνάρτησης e^x από τους N πρώτους όρους της σειράς

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots + \frac{x^n}{n!}$$

Μετά τον υπολογισμό του κάθε όρου να εκτυπώνεται η διαφορά της τιμής που μας δίνει η σειρά από την πραγματική τιμή που μας δίνει η συνάρτηση sin της βιβλιοθήκης <math>.

Ορισμός μεταβλητών
x το όρισμα που θα υπολογιστεί, N ο μέγιστος αριθμός όρων που θα υπολογιστούν
S : άθροισμα των όρων της σειράς που προσεγγίζει το αποτέλεσμα
Βοηθητικές μεταβλητές, i μετρητής όρων, $y=x^i$ και $f=i!$
Θέτω αρχικές τιμές για τις μεταβλητές S=1, y=1, f=1;
Επανάληψη από i=1 έως N
Υπολόγισε παραγοντικό i!, f=f*i (στην i-οστή επανάληψη)
Υπολόγισε τη δύναμη x^i , y=y*x (στην i-οστή επανάληψη)
Πρόσθεσε στη σειρά τον i-οστό όρο, S=S+y/f
Εκτύπωσε i, S και τη διαφορά Δ=S-e^x,
Τέλος Επανάληψης **Κώδικας: taylorExp.c**

Παραδείγματα

4. Θέλουμε να υπολογίσουμε τη τετραγωνική ρίζα Y ενός αριθμού X σύμφωνα με τον παρακάτω κανόνα

Αν Y_0 είναι μια προσέγγιση του Y τότε η $Y_1=(Y_0+X/Y_0)/2$ είναι μια καλύτερη προσέγγιση, Δηλαδή η ακολουθία $Y_{n+1}=(Y_n+X/Y_n)/2$ συγκλίνει ομοιόμορφα στη τετραγωνική ρίζα του X.

* Ο υπολογισμός να γίνει μέχρι να έχουμε μια ακρίβεια της τάξης του 10^{-5}.

Δώσε X
Δώσε Y_0 (αρχική προσέγγιση στη ρίζα)
Εκτέλεσε (αρχή επανάληψης)
  Υπολόγισε $Y_1=0.5*(Y_0+X/Y_0)$;
  Υπολόγισε τη διαφορά $\Delta=|Y_1-Y_0|$
  Θέσε $Y_0=Y_1$ (νέα προσέγγιση)
Επανέλαβε εφόσον $\Delta>10^{-5}$
Εκτύπωσε Αποτέλεσμα

Δες επίσης κώδικα FindSqrRoot.c

```
main()
{
        double X,Y0,Y1,D;
        X=3; Y0=2;
        do{
             Y1=0.5*(Y0+X/Y0);
             D=fabs(Y1-Y0);
             Y0=Y1;
        }while(D>0.00001);
        printf("Sqrt(%f) = %f \n",X,Y1);
        getchar();
}
```

Ασκήσεις

C3.20. Υλοποιείστε ένα πρόγραμμα που να υπολογίζει τη τιμή της παρακάτω σειράς για δοθέν x μέχρι τον N-οστό όρο (το N να ορίζεται ως σταθερά)

$$y = \sum_{n=0}^{N} \frac{\cos(nx)}{(n+1)^2} = 1 + \frac{\cos(x)}{4} + \frac{\cos(2x)}{9} + \frac{\cos(3x)}{16} + \ldots\ldots + \frac{\cos(Nx)}{(N+1)^2}$$

for

Όρισε το N
Δώσε τιμή στο x
Θέσε y=0 (αρχική τιμή)
Για n από 0 έως N επανέλαβε
    υπολόγισε το $z=\cos(n*x)/(n+1)^2$
    αύξησε το y κατά z
Τέλος Επανάληψης
Εκτύπωσε το y

C3.21. Υλοποιείστε ένα πρόγραμμα που να υπολογίζει τη τιμή της παραπάνω σειράς (άσκηση 3.1) για δοθέν x και μέχρι ακρίβεια 10^{-8} (δηλαδή ο υπολογισμός να σταματάει όταν ο τελευταίος όρος που προστίθεται στο y είναι μικρότερος του 10^{-8} κατά απόλυτη τιμή). Να εκτυπώνονται τα αποτελέσματα σε κάθε βήμα.

do-while

Όρισε το N
Δώσε τιμή στο x
Θέσε y=0 (αρχική τιμή)
Θέσε n=0
Εκτέλεσε (Αρχή επανάληψης)
    υπολόγισε το $z=\cos(n*x)/(n+1)^2$
    αύξησε το y κατά z
    εκτύπωσε n, z, y
    αύξησε το n κατά 1
Επανέλαβε εφόσον $|z|>10^{-8}$
Εκτύπωσε το y

Ασκήσεις

C3.23. Υπολογίστε τη τιμή της συνάρτησης sin(x) από τους *N* πρώτους όρους της σειράς

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots + \frac{(-1)^{n-1} x^{2n-1}}{(2n-1)!}$$

Μετά τον υπολογισμό του κάθε όρου να εκτυπώνεται η διαφορά της τιμής που μας δίνει η σειρά από την τιμή που μας δίνει η συνάρτηση sin της βιβλιοθήκης <math>.

C3.25. Η ακολουθία Fibonacci δίνεται από τον γενικό όρο

$$u_n = (a^n - b^n)/\sqrt{5}, \quad \dot{o}\pi o\upsilon \quad a = \tfrac{1}{2}(1+\sqrt{5}), \quad a = \tfrac{1}{2}(1-\sqrt{5})$$

α) Υπολογίστε τους δέκα πρώτους όρους της ακολουθίας με βάση τον παραπάνω τύπο

β) Υπολογίστε τους δέκα πρώτους όρους της ακολουθίας χρησιμοποιώντας τον ισοδύναμο αναδρομικό τύπο $u_{n+2}=u_{n+1}+u_n$ με $u_1=1$ και $u_2=2$

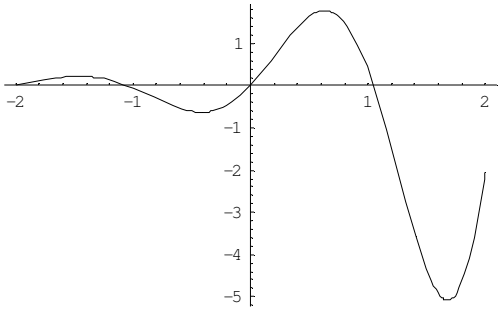
Ασκήσεις

Άσκηση C3.30

Δίνεται η συνάρτηση $y=y(x)$, $y = e^x \sin(3x)$

Σε ένα διάστημα τιμών (x_{min}, x_{max}), .οπου το x μεταβάλλεται με βήμα dx, να εκτυπωθούν στην οθόνη οι τιμές x, y και τετραγωνική ρίζα του y (εφόσον υπάρχει). Το πρόγραμμα να σταματάει αν το |y| γίνει για πρώτη φορά μεγαλύτερο του y_{max}

(Δοκιμή για x_{min}=-2, x_{max}=2, dx=0.1, y_{max}=4)



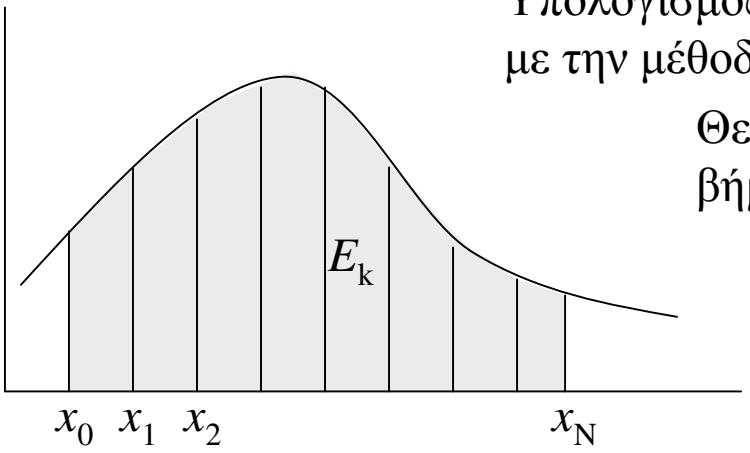
Θέσε x=x_{min}
(*αρχή βρόχου*)
Ενώ x<=x_{max} εκτέλεσε τα παρακάτω
υπολόγισε το y
Αν |y|>y_{max} έξοδος από το βρόχο
Εκτύπωσε x,y
Αν y>0 υπολόγισε και εκτύπωσε το z=$y^{1/2}$
αλλιώς εκτύπωσε «-»
Αύξησε το x κατά dx
Επανέλαβε (*τέλος βρόχου*)
Τέλος προγράμματος

Δες κώδικα exersizeC330.c

Άσκηση C3.31

Θεωρείστε την άσκηση C3.30. Χρησιμοποιείστε την επανάληψη `for` για να εκτυπώσετε μόνο τις τιμές x και $y^{1/2}$. Να γίνει και χρήση της continue.

Εφαρμογές – Υπολογισμός ολοκληρώματος



Υπολογισμός ολοκληρώματος της y=f(x) στο διάστημα [a,b] με την μέθοδο του τραπεζίου

Θεωρούμε N διαμερίσεις του διαστήματος [*a*,*b*] με βήμα $\Delta x=h=(b-a)/N$. Θα είναι

$$I=\int_a^b f(x)dx \approx \sum_{k=1}^{N} E_k, \quad \text{οπου} \quad E_k=\frac{f(x_{k-1})+f(x_k)}{2}h$$

Όρισε N,a,b
Υπολόγισε h=(b-a)/N
Θέσε x0=a και **υπολόγισε** f0=f(x0)
Θέσε I=0 (*αρχική τιμή ολοκληρώματος*)
Επανέλαβε N φορές
    x=x0+h;
    υπολόγισε f1=f(x);
    υπολόγισε E=(f0+f1)*h/2
    αύξησε το I κατά E (I=I+E)
    Θέσε x0=x, f0=f1 (*νέες τιμές για το επόμενο βήμα*)
Τέλος Επανάληψης
Εκτύπωσε αποτέλεσμα

Παράδειγμα f(x)=x*cos(x)
Διάστημα [0,π/2] με N=20

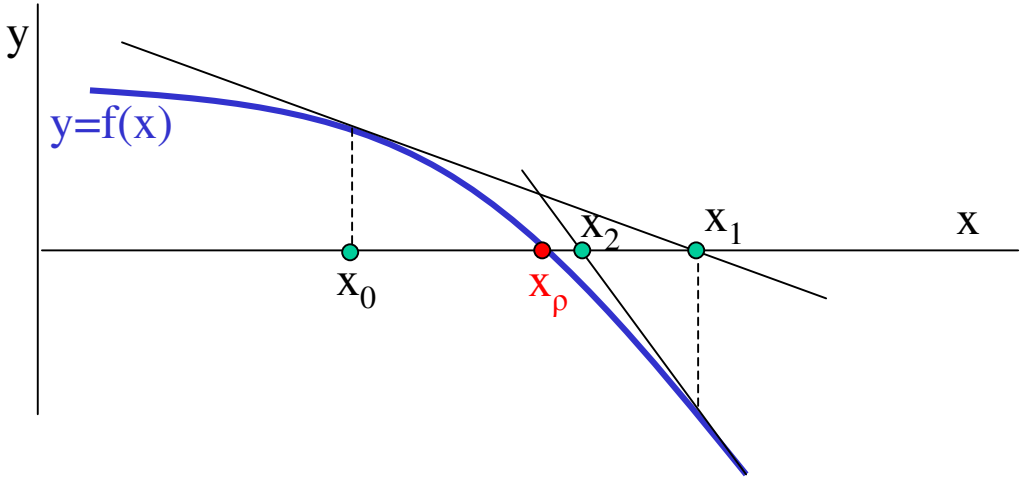
```
/* trapezio1.c */
#define pi 3.141592653589793

main()
{
        double a=0, b=pi/2; int i, N=20;
        double h,x0,x,f0,f1,E,I;

        h=(b-a)/N;
        x0=a;   f0=x0*cos(x0);
        I=0;
        for(i=0;i<N;i++){
                x=x0+h;    f1=x*cos(x);
                E=0.5*(f0+f1)*h;
                I+=E;
                x0=x; f0=f1;
        }
        printf("Integral=%f\n",I);
        getchar();
}
```

Δες επίσης trapezio2.c

Εφαρμογές – Υπολογισμός ρίζας συνάρτησης



Η μέθοδος Newton-Raphson

Έστω x_ρ μια ζητούμενη ρίζα της συνάρτησης y=f(x), η οποία έχει παράγωγο την f_d=df/dx.
Αν x_0 είναι μια αρχική εκτίμηση της ρίζας τότε μια δεύτερη εκτίμηση δίνεται από την
$x_1=x_0-f(x_0)/f_d(x_0)$, μια τρίτη εκτίμηση θα είναι η
$x_2=x_1-f(x_1)/f_d(x_1)$, κ.ο. κ.
Αν η μέθοδος συγκλίνει τότε οι εκτιμήσεις μας x_i, i=1,2,3… θα συγκλίνουν προς την ρίζα x_ρ.

Δες επίσης kepler.c

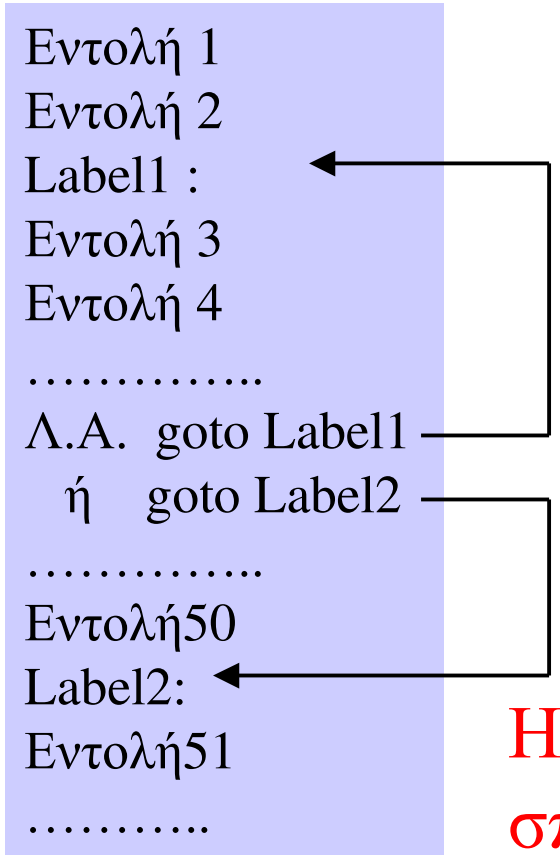
$\pi.\chi.$ Ρίζα της $e^{-x}-x=0, \quad x_0 \approx 0$

```
/* riza_synartisis.c */
#include <stdio.h>
#include <math.h>

#define KMAX 20
#define acc 1.0e-5

main()
{
        int k=1;
        double y,yd,dx;
        double x=0;
        do{
                y=exp(-x)-x;
                yd=-exp(-x)-1;
                dx=y/yd;
                x-=dx;
                printf("%2d  x=%10.9f
        }while(fabs(y)>acc&&k<KMAX);
        getchar();
}
```

Ετικέτες και goto



- Οι ετικέτες (labels) εισάγονται στο πρόγραμμα όπως και οι εντολές αλλά τερματίζουν με « **:** »
- Η εντολή `goto` "*label*" μεταφέρει τη ροή του προγράμματος στην επόμενη εντολή μετά την ετικέτα.
- Η ετικέτα και η `goto` πρέπει να βρίσκονται στην ίδια συνάρτηση.

Δες κώδικα prog38goto.c (σχετικός με την άσκηση 3.30)

Η εντολή goto πρέπει να χρησιμοποιείται σπάνια αν όχι καθόλου!!!

Ένα πρόγραμμα με goto είναι δυσνόητο, δύσκολο στη συντήρησή του και δυσκολεύει τον μεταγλωττιστή στην δημιουργία ενός βέλτιστου κώδικα.

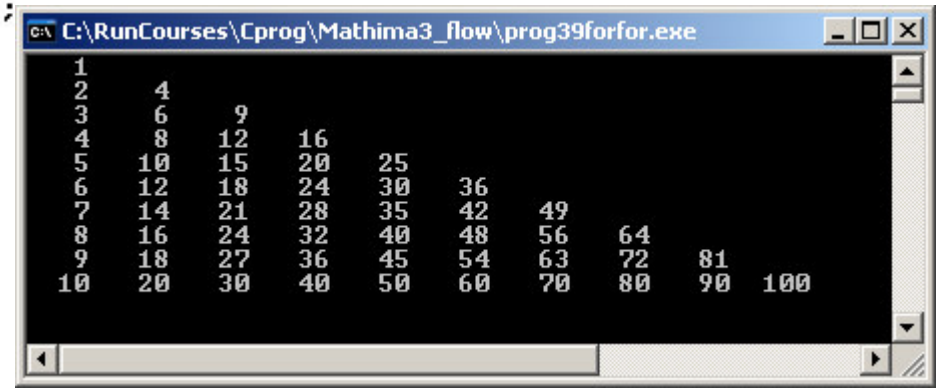
Ένθετοι Βρόχοι

```
do{
      ……..
     for( a; b; c) {
          …….
          while(Λ.Σ) {
               ……..
               ……..
           }
      }
      ……..
      for( a; b; c) {
          ……..
      }
      ……..
}while(Λ.Σ)
```

Οι εντολές ενός βρόχου μπορεί να περιλαμβάνουν και «δευτερεύοντες» βρόχους.
Για να προχωρήσει η ροή σε έναν βρόχο πρέπει να τερματίσει ο κάθε δευτερεύων βρόχος που θα συναντηθεί.

```c
/*prog39forfor.c*/
#include <stdio.h>

main()
{
        int i,j;
        for(i=1;i<=10;i++){
            for(j=1;j<=i;j++) printf(" %3d ",i*j);
            printf("\n");
        }
        getchar();
}
```

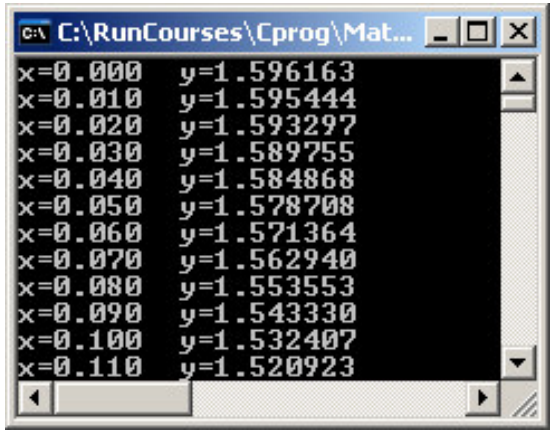


Ασκήσεις

Άσκηση C3.50

Υπολογίστε της τιμές της συνάρτησης y=y(x) που δίνεται από την διπλανή σειρά

$$y=\sum_{n=0}^{N}\frac{\cos(nx)}{(n+1)^2}=1+\frac{\cos(x)}{4}+\frac{\cos(2x)}{9}+\frac{\cos(3x)}{16}+.....+\frac{\cos(Nx)}{(N+1)^2}$$



Για ένα δεδομένο *N* και σε ένα διάστημα (*xmin*,*xmax*) με βήμα *dx*
(πχ, *N*=20, *xmin*=0, *xmax*=2, *dx*=0.01)

Άσκηση C3.51

Αν *i* και *j* ακέραιοι με τιμές στο διάστημα [0,10], βρείτε τη μέγιστη τιμή της συνάρτησης

$$z=(i+j)^4 e^{-\frac{i^2+j^2}{10}}$$

καθώς και το σημείο (i_0,j_0) του μεγίστου
(*Απάντηση max=214.23,* $(i_0,j_0)=(3,3)$)

Ασκήσεις

Άσκηση C3.52

Κάντε ένα πρόγραμμα που να υπολογίζει την τιμή της σειράς για δεδομένα N και M

$$y=\sum_{m=1}^{M}\sum_{n=0}^{N}\frac{\cos(\sqrt{n}\,\pi)}{(m+n)^2}=1+\frac{\cos(\pi)}{4}+\frac{\cos(\sqrt{2}\pi)}{9}+.....+\frac{\cos(\sqrt{N}\pi)}{(1+N)^2}+$$

$$\frac{1}{4}+\frac{\cos(\pi)}{9}+\frac{\cos(\sqrt{2}\pi)}{16}+.....+\frac{\cos(\sqrt{N}\pi)}{(2+N)^2}+$$

$$.....$$

$$\frac{1}{(M+0)^2}+\frac{\cos(\pi)}{(M+1)^2}+\frac{\cos(\sqrt{2}\pi)}{(M+2)^2}+.....+\frac{\cos(\sqrt{N}\pi)}{(M+N)^2}+$$

(*Για M=N=10 βρίσκουμε 1.118370*)

Άσκηση C3.53

Κάντε ένα πρόγραμμα που να υπολογίζει και να εκτυπώνει όλες τις τιμές της παραπάνω σειράς (της άσκηση 3.52) για N=100 και για M από 1 έως 10.