ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ

http://www.corelab.ece.ntua.gr/courses/programming/

Διδάσκοντες: Στάθης Ζάχος (zachos@cs.ntua.gr)
Νίκος Παπασπύρου (nickie@softlab.ntua.gr)

Διαφάνειες παρουσίασης #7

- ✓Πίνακες (συνέχεια)
- ✓Αριθμητικοί υπολογισμοί
- ✓Αναδρομή

Πίνακες (iii)

- Διάβασμα ενός πίνακα
  - γνωστό μέγεθος

```
for i:=1 to 10 do read(a[i])
```

  - πρώτα διαβάζεται το μέγεθος

```
read(howmany);
for i:=1 to howmany do read(a[i])
```

  - στα παραπάνω πρέπει να προηγηθούν

```
var a : array [1..10] of real;
    i, howmany : 1..10;
```

Πίνακες (iv)

- Διάβασμα ενός πίνακα (συνέχεια)
  - τερματισμός με την τιμή 0

```
read(x); i:=0;
while x<>0 do
begin i:=i+1; a[i]:=x; read(x) end
```

  - στο παραπάνω πρέπει να προηγηθούν

```
var a : array [1..10] of real;
    i : 0..10;  x : real;
```

  - Προσοχή: δε γίνεται έλεγχος για το πλήθος των στοιχείων που δίνονται!

Πράξεις με πίνακες

- Απλές πράξεις, π.χ.

```
a[k] := a[k]+1;
a[k] := a[1]+a[n];
for i:=1 to 10 do writeln(a[i]);
if a[k] > a[k+1] then ...
```

- Αρχικοποίηση (με μηδενικά)

```
for i:=1 to 10 do a[i]:=0
for ch:='a' to 'z' do b[ch]:=0
```

- Εύρεση ελάχιστου στοιχείου

```
x := a[1];
for i:=2 to 10 do
    if a[i] < x then x := a[i]
```

Γραμμική αναζήτηση (i)

- Πρόβλημα (αναζήτησης): δίνεται ένας πίνακας ακεραίων **a** και ζητείται να βρεθεί αν υπάρχει ο ακέραιος **x** στα στοιχεία του

```
program search(input,output);
    var x : integer;
        a : array [1..10] of integer;
    άλλες δηλώσεις;
begin  τίτλος επικεφαλίδα;
    οδηγίες στο χρήστη;
    read(x);
    διάβασμα του πίνακα;
    ψάξιμο στον πίνακα για τον x;
    παρουσίαση αποτελεσμάτων
end.
```

Γραμμική αναζήτηση (ii)

- Μια δυνατή συγκεκριμενοποίηση

```
for i:=1 to 10 do read(a[i]);

i:=0;
repeat i:=i+1 until (a[i]=x) or (i=10);

if a[i]=x then
   writeln('Το βρήκα στη θέση ', i)
else
   writeln('Δεν το βρήκα')
```

  - Στη χειρότερη περίπτωση θα ελεγχθούν όλα τα στοιχεία του πίνακα
  - Απαιτούνται $a\,n + b$ βήματα $\Rightarrow$ γραμμική (a, b σταθερές, n το μέγεθος του πίνακα)

Γραμμική αναζήτηση (iii)

- Εναλλακτική συγκεκριμενοποίηση #1

```
i:=0;
repeat i:=i+1;
   if a[i]=x then found:=true
             else found:=false
until found or (i=10);

if found then
   writeln('Το βρήκα στη θέση ', i)
else
   writeln('Δεν το βρήκα')
```

Γραμμική αναζήτηση (iv)

- Εναλλακτική συγκεκριμενοποίηση #2

```
i:=0; found:=false;
repeat i:=i+1;
   if a[i]=x then found:=true
until found or (i=10);

if found then
   writeln('Το βρήκα στη θέση ', i)
else
   writeln('Δεν το βρήκα')
```

Γραμμική αναζήτηση (v)

- Εναλλακτική συγκεκριμενοποίηση #3

```
i:=0;
repeat i:=i+1; found := a[i]=x
until found or (i=10);

if found then
   writeln('Το βρήκα στη θέση ', i)
else
   writeln('Δεν το βρήκα')
```

Δυαδική αναζήτηση (i)

- Προϋπόθεση: ο πίνακας να είναι ταξινομημένος, π.χ. σε αύξουσα διάταξη
- Είναι πολύ πιο αποδοτική από τη γραμμική αναζήτηση
  - Στη χειρότερη περίπτωση απαιτούνται
    $a \log_2 n + b$ βήματα
    (a, b σταθερές, n το μέγεθος του πίνακα)

Δυαδική αναζήτηση (ii)

- Το πρόγραμμα

```
program binsearch(input,output);
    const n=100;
    var i,howmany,mid,first,last : 0..n;
        a : array [1..n] of integer;
        x : integer; found : boolean;
begin Μήνυμα επικεφαλίδα και οδηγίες χρήσης;
    read(howmany); (* κατά αύξουσα σειρά *)
    for i:=1 to howmany do read(a[i]);
    read(x);
    first:=1; last:=howmany;
    found:=(x=a[first]) or (x=a[last]);
```

Δυαδική αναζήτηση (iii)

- Το πρόγραμμα (συνέχεια)

```
while not found and (first<last) do
begin mid:=(first+last) div 2;
    if x<a[mid] then begin
        last:=mid-1; first:=first+1
    end
    else begin
        first:=mid; last:=last-1
    end;
    found:=(x=a[first]) or x=a[last])
end
```

Δυαδική αναζήτηση (iv)

- Το πρόγραμμα (συνέχεια)

```
    if found then
        if x=a[first] then writeln(first)
                      else writeln(last)
    else writeln('not found')
end.
```

Δυαδική αναζήτηση (v)

- Εναλλακτική μορφή κώδικα αναζήτησης

```
found := false;
while not found and (first<=last) do
begin mid := (first+last) div 2;
    found := x=a[mid];
    if x<a[mid] then last:=mid-1
                else first:=mid+1
end;
if found then writeln(mid)
         else writeln('not found')
```

Ταξινόμηση (i)

- Πρόβλημα: να αναδιαταχθούν τα στοιχεία ενός πίνακα ακεραίων σε αύξουσα σειρά
- Μια από τις σημαντικότερες εφαρμογές των ηλεκτρονικών υπολογιστών
- Βασική διαδικασία: εναλλαγή τιμών

```
procedure swap(var x, y : integer);
    var save : integer;
begin
    save:=x; x:=y; y:=save
end
```

Ταξινόμηση (ii)

- Μέθοδος της φυσαλίδας

```
for i:=1 to n-1 do
    for j:=n-1 downto i do
        if a[j] > a[j+1] then
            swap(a[j], a[j+1])
```

- Πλήθος συγκρίσεων

$$(n-1) + (n-2) + \ldots + 2 + 1 = n\,(n-1) / 2$$

της τάξης του n^2 $\Rightarrow O(n^2)$

Ταξινόμηση (iii)

- Παράδειγμα εκτέλεσης

input:	12 4 9 8 6 7 5
	12 4 9 8 6 **5 7**
	12 4 9 8 **5 6** 7
	12 4 9 **5 8** 6 7
	12 4 **5 9** 8 6 7
	12 **4 5** 9 8 6 7
$i = 1$:	**4 12** 5 9 8 6 7
	4 12 5 9 8 **6 7**
	4 12 5 9 **6 8** 7
	4 12 5 **6 9** 8 7
	4 12 **5 6** 9 8 7
$i = 2$:	4 **5 12** 6 9 8 7
	4 5 12 6 9 **7 8**
	4 5 12 6 **7 9** 8
	4 5 12 **6 7** 9 8
$i = 3$:	4 5 **6 12** 7 9 8
	4 5 6 12 7 **8 9**
	4 5 6 12 **7 8** 9
$i = 4$:	4 5 6 **7 12** 8 9
	4 5 6 7 12 **8 9**
$i = 5$:	4 5 6 7 **8 12** 9
$i = 6$:	4 5 6 7 8 **9 12**

Πολυδιάστατοι πίνακες

- Παράδειγμα

```
var a : array [1..10,5..16] of integer;
...
a[1,13] := 42;
for i:=1 to 10 do
    for j:=5 to 16 do read(a[i,j])
```

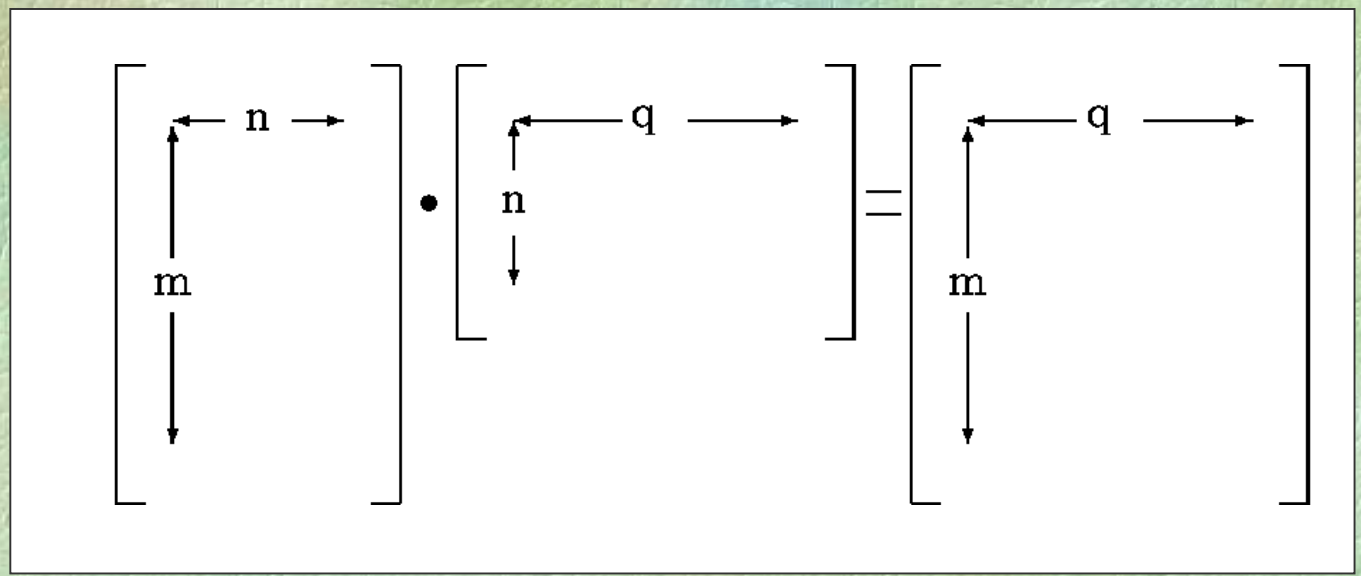
- Ισοδύναμος ορισμός και χρήση

```
var a : array [1..10] of
                array [5..16] of integer;
... a[i][j] ...
```

Πολλαπλασιασμός πινάκων (i)

- Δίνονται οι πίνακες: $a\ (m \times n)$, $b\ (n \times q)$
- Ζητείται ο πίνακας: $c = a\ b\ \ (m \times q)$ όπου:

$$c_{i,j} = \sum_{k=1}^{n} a_{i,k} b_{k,j}$$



Πολλαπλασιασμός πινάκων (ii)

- Το πρόγραμμα

```
var a : array [1..m,1..n] of real;
    b : array [1..n,1..q] of real;
    c : array [1..m,1..q] of real;

...

for i:=1 to m do
  for j:=1 to q do
  begin
    c[i,j] := 0;
    for k:=1 to n do
      c[i,j] := c[i,j] + a[i,k]*b[k,j]
  end
```

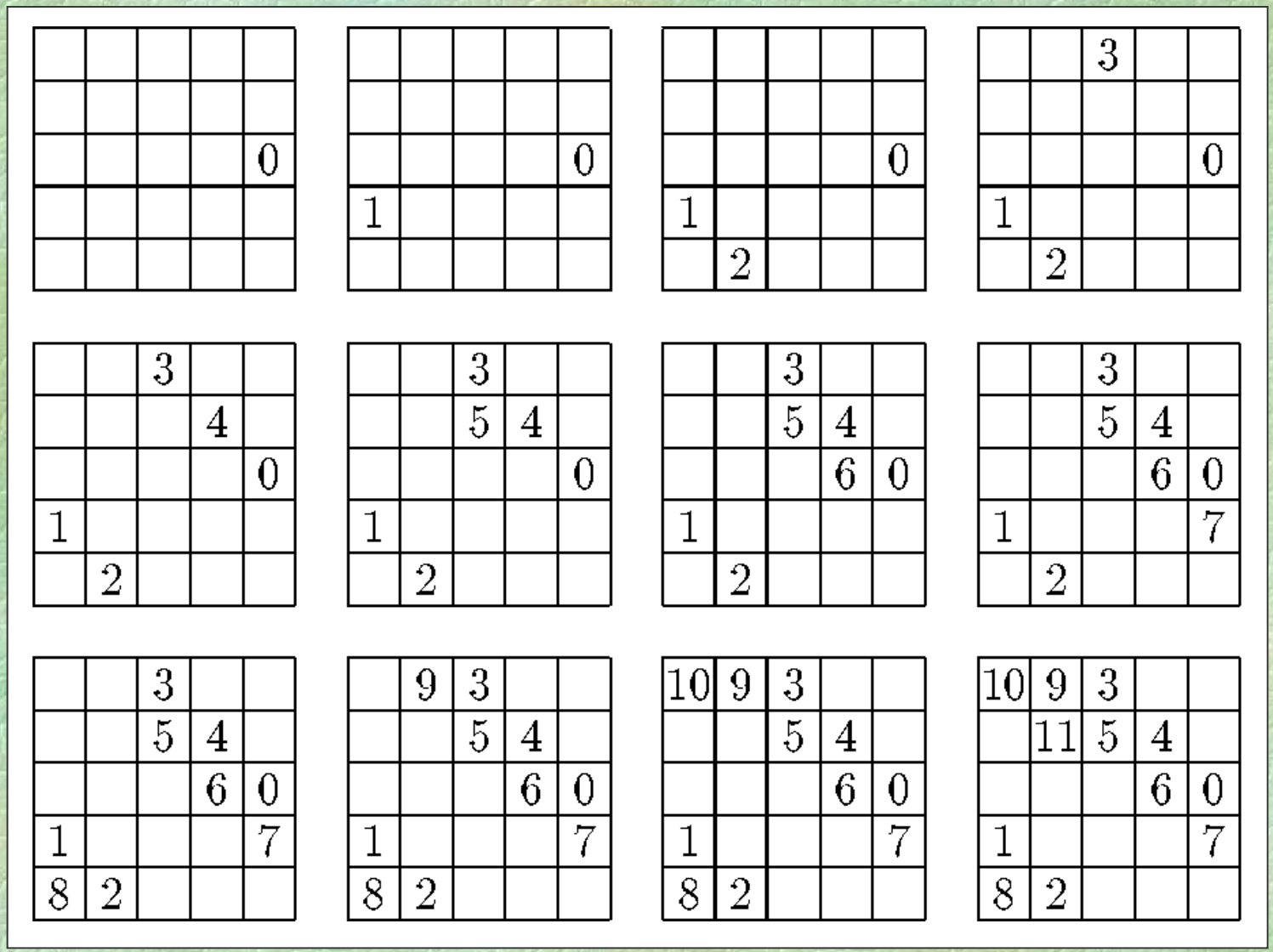
Μαγικά τετράγωνα (i)

- Διδιάστατοι πίνακες ($n \times n$) που περιέχουν όλους τους φυσικούς μεταξύ 0 και n^2-1
  - το άθροισμα των στοιχείων κάθε στήλης, γραμμής και διαγωνίου είναι σταθερό

10	9	3	22	16
17	11	5	4	23
24	18	12	6	0
1	20	19	13	7
8	2	21	15	14

- Πρόβλημα: κατασκευή μαγικού τετραγώνου ($n \times n$) για περιττό n

Μαγικά τετράγωνα (ii)



Μαγικά τετράγωνα (iii)

- Κατασκευή για περιττό *n*

```
i:=n div 2; j:=n; k:=0;
for h:=1 to n do
begin j:=j-1; a[i,j]:=k; k:=k+1;
  for m:=2 to n do
  begin j:=(j+1) mod n; i:=(i+1) mod n;
        a[i,j]:=k; k:=k+1
  end
end;

for i:=0 to n-1 do
begin
   for j:=0 to n-1 do write(a[i,j]:4);
   writeln
end
```

Αριθμητικοί υπολογισμοί (i)

- Τύπος **`real`**
  - προσεγγίσεις πραγματικών αριθμών
  - **`trunc`**: ακέραιο μέρος (αποκοπή)
  - **`round`**: στρογγυλοποίηση
- Παράσταση κινητής υποδιαστολής
  - mantissa και εκθέτης $\pm m \cdot 2^{x}$

    όπου $0.5 \le m < 1$ και $x \in \mathbf{Z}$ ή $m = x = 0$
  - το m είναι περιορισμένης ακρίβειας, π.χ. 8 σημαντικά ψηφία

Αριθμητικοί υπολογισμοί (ii)

- Αριθμητικά σφάλματα

  $1000000 + 0.000000001 = 1000000$ *γιατί;*

- Αναπαράσταση των αριθμών

$$1000000 \approx 0.95367432 \cdot 2^{20}$$

$$0.000000001 \approx 0.53687091 \cdot 2^{-29}$$

$$\approx 0.00000000 \cdot 2^{20}$$

$$\text{άθροισμα} \approx 0.95367432 \cdot 2^{20}$$

Εύρεση τετραγωνικής ρίζας (i)

- Χωρίς χρήση της συνάρτησης `sqrt`
- Μέθοδος Newton
  - Δίνεται ο αριθμός $x > 0$
  - Έστω προσέγγιση y της ρίζας, με $y \leq \sqrt{x}$
  - Έστω $z = x / y$
  - Το z είναι προσέγγιση της ρίζας, με $\sqrt{x} \leq z$
  - Για να βρω μια καλύτερη προσέγγιση, παίρνω το μέσο όρο των y και z
  - Επαναλαμβάνω όσες φορές θέλω

Εύρεση τετραγωνικής ρίζας (ii)

- Ακολουθία προσεγγίσεων

$$y_0 = 1 \qquad y_{i+1} = \frac{1}{2}\left(y_i + \frac{x}{y_i}\right)$$

- Παράδειγμα: $\sqrt{37}$ (6.08276253)

$y_0 = 1$

$y_1 = 19$

$y_2 = 10.473684$

$y_3 = 7.003174$

$y_4 = 6.143246$

$y_5 = 6.083060$

$y_6 = 6.082763$

...

Εύρεση τετραγωνικής ρίζας (iii)

```
function sqroot(x : real) : real;
    const eps = 0.00001;    (* 1E-5 *)
    var old, new : real;
begin
    new := 1;
    repeat
        old := new;
        new := (old + x/old) / 2
    until (* συνθήκη τερματισμού *) ;
    sqroot := new
end
```

Εύρεση τετραγωνικής ρίζας (iv)

- Εναλλακτικές συνθήκες τερματισμού
  - Σταθερός αριθμός επαναλήψεων

    `n = 20`
  - Επιτυχής εύρεση ρίζας

    `sqr(new) = x`

    λάθος!
  - Απόλυτη σύγκλιση

    `abs(sqr(new)-x) < eps`
  - Σχετική σύγκλιση

    `abs(sqr(new)-x) / new < eps`

Εύρεση τετραγωνικής ρίζας (v)

- Εναλλακτικές συνθήκες τερματισμού
  - Απόλυτη σύγκλιση κατά Cauchy

    `abs(new-old) < eps`
  - Σχετική σύγκλιση

    `abs(new-old) / new < eps`

Προκαθορισμένες συναρτήσεις

Συναρτήσεις	Τύπος ορίσματος	Τύπος αποτελέσματος
abs, sqr (απόλυτο) (τετράγωνο)	integer, real	ίδιος
sin, cos, exp, ln (ημ) (συν) (εκθ) (log) sqrt, arctan (ρίζα) (τοξεφ)	integer, real	real
odd (περιττός)	integer	boolean
eof, eoln	text	boolean
trunc, round	real	integer
succ (επόμενος) pred (προηγούμενος)	integer, boolean, char	ίδιος
ord (κωδικός ASCII) chr (αντίστροφη της ord)	char integer	integer char

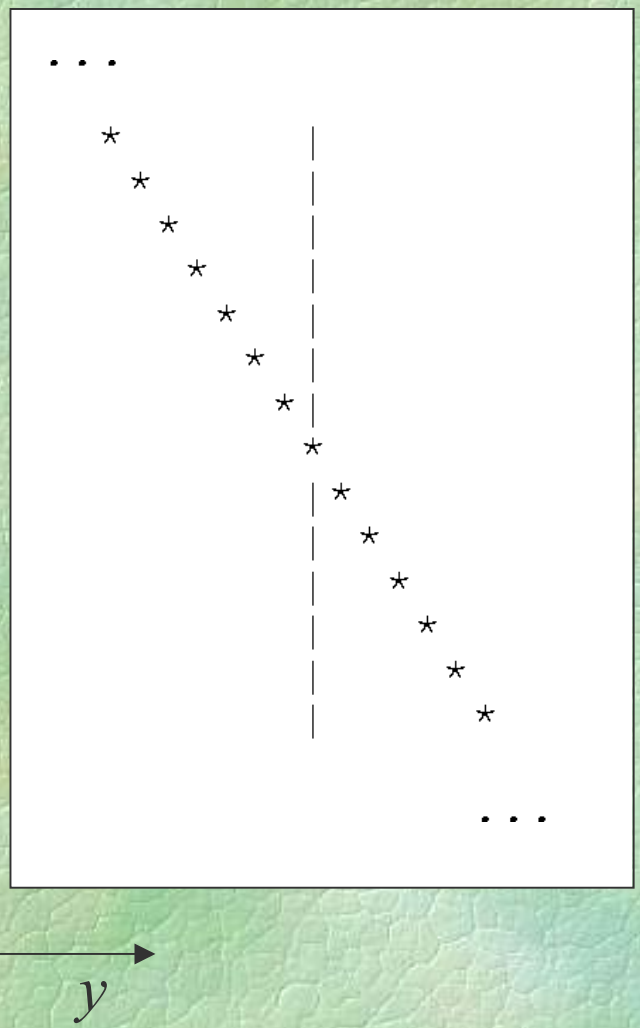
Γραφικές παραστάσεις (i)

- Μορφοποίηση εξόδου (επανάληψη)
  - `write(i:15)`
  - `write('|':40)`
  - `write(x:10:4)`
- Γραφικές παραστάσεις με χαρακτήρες
  - Συνάρτηση $y = f(x)$
  - Συνήθως μας βολεύει να έχουμε
    τον άξονα των x κατακόρυφο και
    τον άξονα των y οριζόντιο

Γραφικές παραστάσεις (ii)

- Παράδειγμα: $f(x) = -x$

```
for i:=1 to 39 do
  writeln('*':i, '|':40-i);
writeln('*':40);
for i:=1 to 39 do
  writeln('|':40, '*':i)
```



Γραφικές παραστάσεις (iii)

- Παράδειγμα: $f(x) = 18 \sin x + 15 \cos 2x$

```
program graph(output);

    var k,n : integer; pi : real;

    procedure axis;
        var i : integer;
    begin
        for i := 1 to 79 do write('-');
        writeln
    end;
```

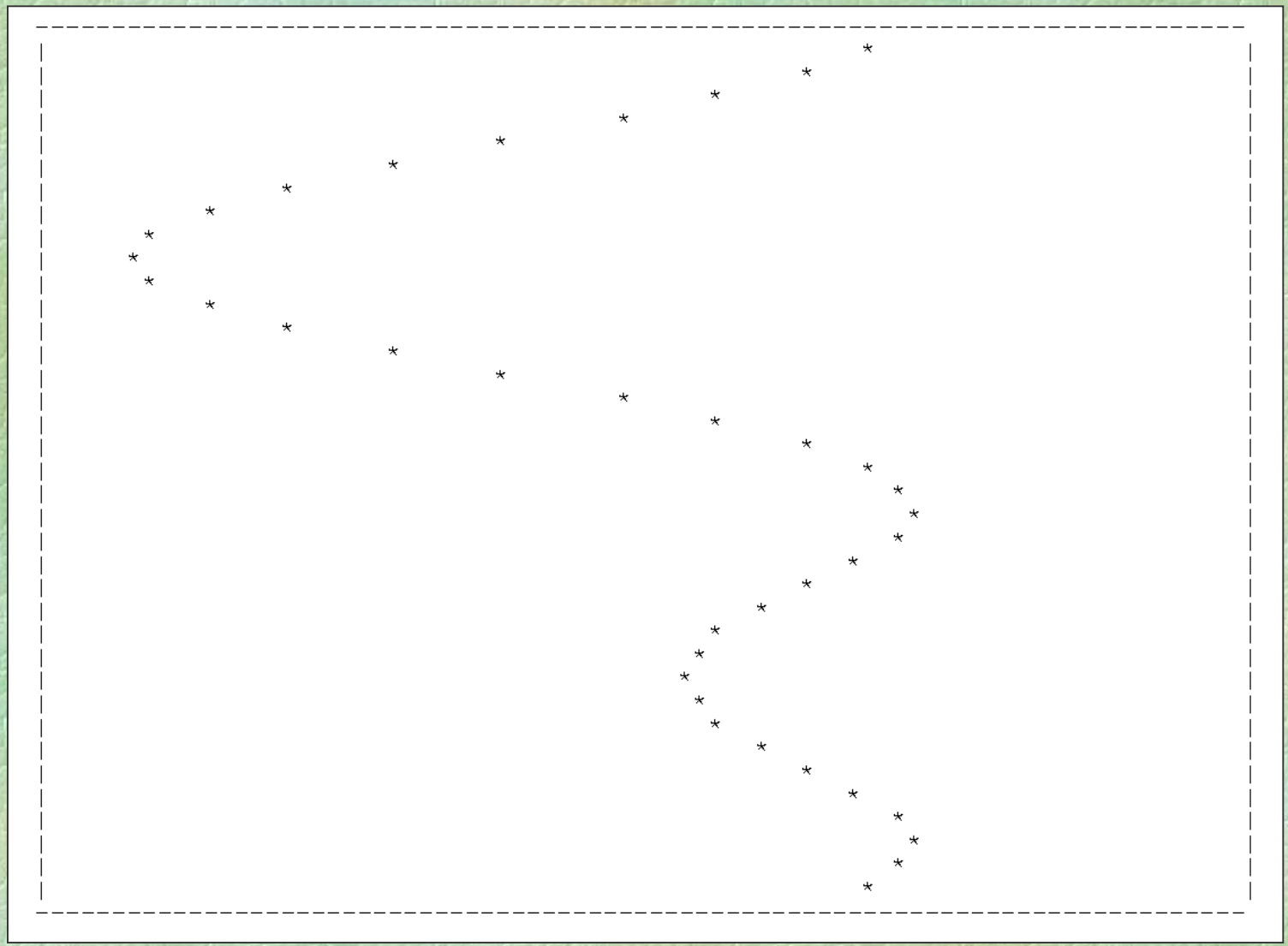
Γραφικές παραστάσεις (iv)

- Παράδειγμα (συνέχεια)

```
function f(j : integer) : integer;
        var x,y : real;
    begin x := pi * j / 18;
        y := 18 * sin(x) + 15 * cos(2*x);
        f := round(y)
    end;

begin pi := 4 * arctan(1); axis;
    for n := -18  to 18 do
    begin k := f(n) + 40;
        writeln('|', '*':k, '|':79-k)
    end;
    axis
end.
```

Γραφικές παραστάσεις (v)



Γραφικές παραστάσεις (vi)

- Παράδειγμα: $f(x) = (3 \cos x) / x$

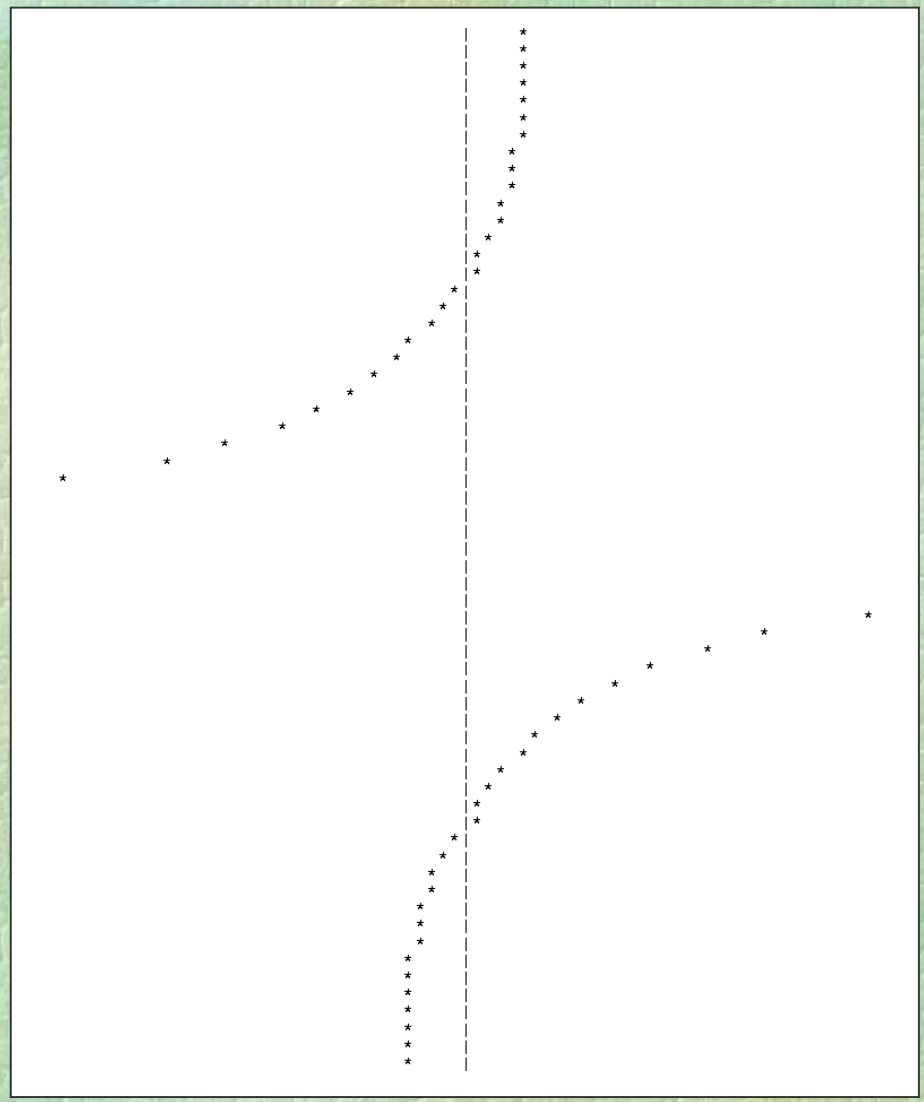
```
program printplot(output);
    const deltax = 0.1;
          bound = 30; wid = 39;
          scale = 5; shift0 = 40;
    var i : -bound..bound;
        n : integer; x : real;
    function f(x : real) : real;
       const eps = 1E-5;
             huge = (* μεγάλος αριθμός *);
    begin
       if abs(x)<eps then f := huge
          else f := 3 * cos(x) / x
    end;
```

Γραφικές παραστάσεις (vii)

- Παράδειγμα (συνέχεια)

```
begin
    x := - bound * deltax;
    for i := - bound to bound do
    begin
        n := round(scale*f(x));
        if abs(n)>wid then
            writeln ('|':shift0)
        else if n<0 then
            writeln('*':n+shift0,'|':-n)
        else
            writeln('|':shift0,'*':n);
        x := x + deltax
    end
end.
```

Γραφικές παραστάσεις (viii)



Γραφικές παραστάσεις (ix)

- Παράδειγμα: $f(x)$ και $g(x)$

```
program twocurves(output);

    const ...

    var ...
        line: array[-wid..wid] of char;

    function one(...) ...
    function two(...) ...

begin
    for j:=-wid to wid do line[j]:=' ';
    line[0]:='|';
    ...
```

Γραφικές παραστάσεις (x)

- Παράδειγμα (συνέχεια)

```
    for i := ...
    begin
        n := ...one...; m := ...two...;
        line[n]:='*'; line[m]:='.';
        for j:=-wid to wid do
            write(line[j]);
        writeln;
        line[n] := ' '; line[m] := ' ';
        line[0] := '|';
        x := x + delta
    end
end.
```

Τριγωνομετρικές συναρτήσεις (i)

- Συνημίτονο με ανάπτυγμα Taylor

$$\cos(x) = \sum_{i=0}^{\infty} (-1)^i \frac{x^{2i}}{(2i)!}$$

- για τον όρο με δείκτη i+1 έχουμε:

$$(-1)^{i+1} \frac{x^{2i+2}}{(2i+2)!} = -\left[(-1)^i \frac{x^{2i}}{(2i)!} \right] \frac{x^2}{(2i+1)(2i+2)}$$

- οπότε αν $n = 2i+1$ έχουμε:

$$newterm = -oldterm \frac{x^2}{n(n+1)}$$

Τριγωνομετρικές συναρτήσεις (ii)

```
function mycos(x : real) : real;

    const eps = 1E-5;

    var sqx, term, sum : real;
        n : integer;

begin n := -1; sqx := sqr(x);
    term := 1; sum := 1;
    repeat n := n + 2;
        term := -term * sqx / (n*(n+1));
        sum := sum + term
    until abs(term/sum) < eps;
    mycos := sum
end
```

Αναδρομή (i)

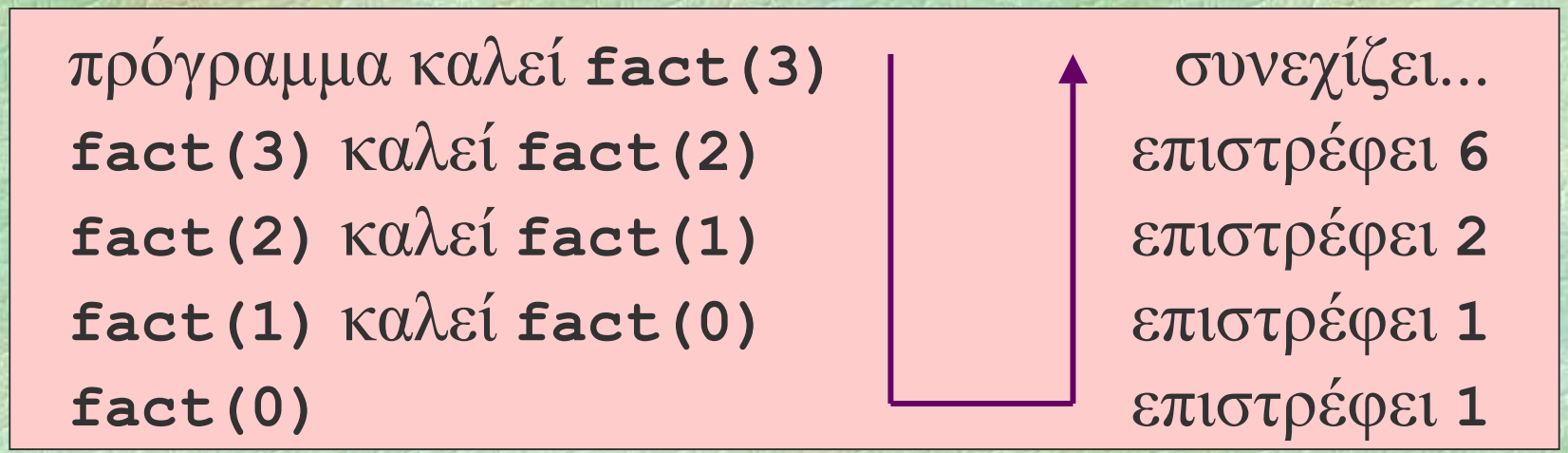
- Αναδρομικές διαδικασίες ή συναρτήσεις: αυτές που καλούν τον εαυτό τους
- Το αρχικό πρόβλημα ανάγεται στην επίλυση ενός ή περισσότερων μικρότερων προβλημάτων του ίδιου τύπου
- Παράδειγμα: παραγοντικό
  - $n! = n * (n–1) * (n–2) * ... * 2 * 1$
  - Αναδρομικός ορισμός

    $0! = 1$ $\qquad\qquad$ $(n+1)! = (n+1) * n!$

Αναδρομή (ii)

- Παράδειγμα: παραγοντικό (συνέχεια)

```
function fact(n : integer) : integer;
begin
    if n=0 then fact := 1
           else fact := fact(n-1) * n
end
```



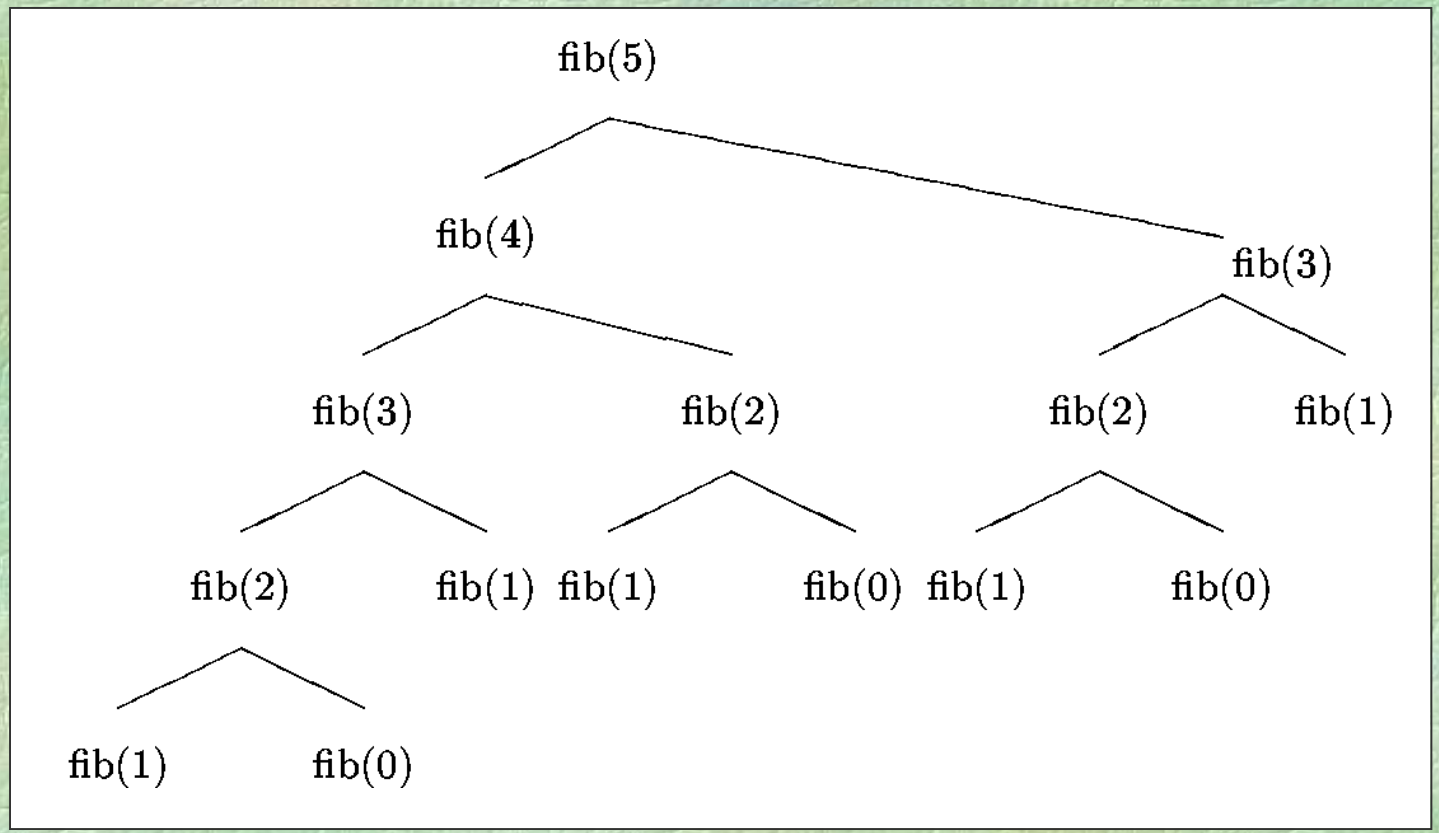
Αναδρομή (iii)

- Αριθμοί Fibonacci
  - $F_0 = 1 \;,\; F_1 = 1$
  - $F_{n+2} = F_n + F_{n+1} \quad , \forall n \in \mathbf{N}$
- Αναδρομική συνάρτηση υπολογισμού

```
function fib(n : integer) : integer;
begin
    if (n=0) or (n=1) then
        fib := 1
    else
        fib := fib(n-1) + fib(n-2)
end
```

Αναδρομή (iv)

- Αυτός ο αναδρομικός υπολογισμός των αριθμών Fibonacci δεν είναι αποδοτικός



Αναδρομή (v)

- Μέγιστος κοινός διαιρέτης
  - Αναδρομική υλοποίηση του αλγορίθμου του Ευκλείδη

```
function gcd(i,j : integer) : integer;
begin
    if (i=0) or (j=0) then
        gcd := i+j
    else if i > j then
        gcd := gcd(i mod j, j)
    else
        gcd := gcd(i, j mod i)
end
```

Αναδρομή (vi)

- Συνάρτηση παρόμοια με του Ackermann

$$z(i, j, 0) = j+1 \qquad z(i, 0, 1) = i$$

$$z(i, 0, 2) = 0 \qquad z(i, 0, n+3) = 1$$

$$z(i, j+1, n+1) = z(i, z(i, j, n+1), n) \quad , \forall i, j, n \in \mathbf{N}$$

```
function z(i,j,n : integer) : integer;
begin
   if n=0 then z:=j+1
   else if j=0 then
      if n=1 then z:=i
             else if n=2 then z:=0
                         else z:=1
   else z:=z(i,z(i,j-1,n),n-1)
end
```

Αμοιβαία αναδρομή

```
function f2(n:integer):integer; forward;

function f1(n:integer):integer;
begin
   if n=0 then f1 := 5
          else f1 := f1(n-1) * f2(n-1)
end

function f2(n:integer):integer;
begin
   if n=0 then f2 := 3
          else f2 := f1(n-1) + 2*f2(n-1)
end
```