



ΑΝΟΙΚΤΑ ακαδημαϊκά
μαθήματα ΠΠ



Εισαγωγή στον Προγραμματισμό Η/Υ για Χημικούς Μηχανικούς

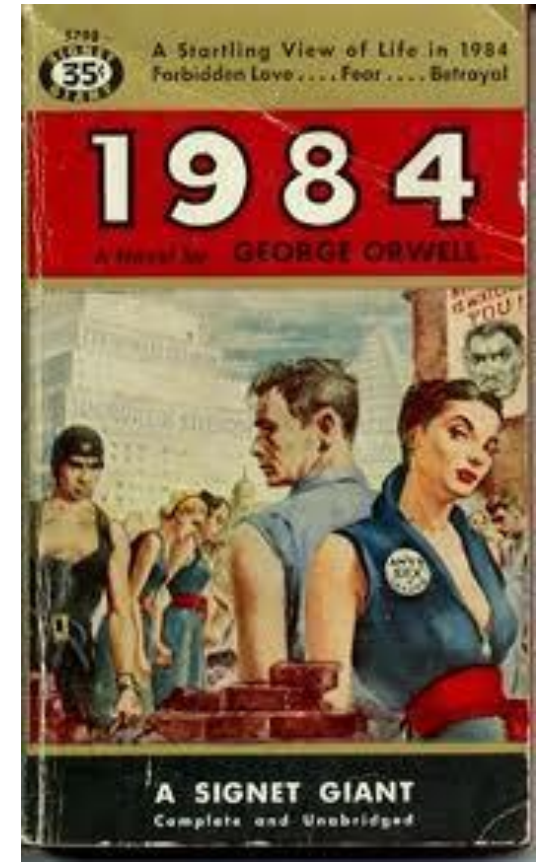
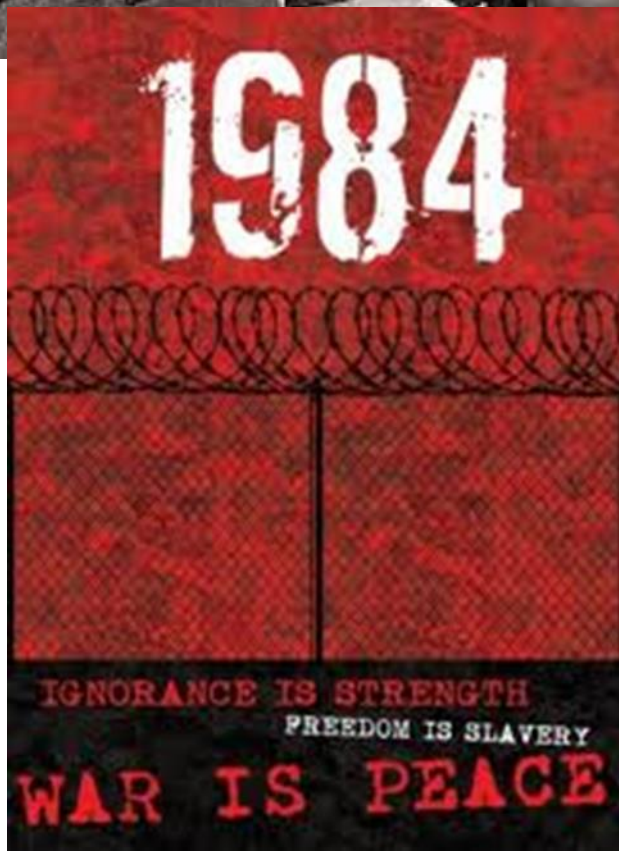
Παρουσίαση Διαλέξεων: 9. Δυναμικά Δεδομένα
Καθηγητής Δημήτρης Ματαράς



Copyright © 2014 by Prof. D. S. Mataras (mataras@upatras.gr). This work is made available under the terms of the Creative Commons Attribution-NonCommercial-NoDerivs 3.0 license, <http://creativecommons.org/licenses/by-nc-nd/3.0/>

‘Γιατί, στο κάτω κάτω, πως το ξέρουμε ότι δύο και δύο κάνουν τέσσερα, ή ότι ισχύει ο νόμος της βαρύτητας, ή ότι το παρελθόν είναι αμετάβλητο; Αν το παρελθόν και ο εξωτερικός κόσμος υπάρχουν μόνο στο νου, κι' αν ο νους μπορεί να ελέγχεται και να καθοδηγείται, τότε τι γίνεται;’

‘1984’, Τζωρτζ Όργουελ



Δυναμικά δεδομένα

- ▶ Δεδομένα που μπορούν να αλλάζουν χαρακτηριστικά και τιμή κατά τη διάρκεια της εκτέλεσης του προγράμματος:
 1. Δυναμικοί Πίνακες **allocatable**
 2. Δυναμικοί Πίνακες Διαδικασιών:
 - a. Εικονικοί Πίνακες Υποθετικής Μορφής
 - b. Εικονικοί Πίνακες Υποθετικού Μεγέθους
 - c. Αυτόματοι Πίνακες
 - d. Functions που επιστρέφουν Πίνακα
 - e. Εικονικοί **allocatable** Πίνακες **F₀₃**
 - f. Allocatable Functions **F₀₃**
 3. Δείκτες (pointers):
 - a. Πίνακες Δεικτών
 - b. Λίστες Δεικτών
 - c. Δυαδικά Δένδρα
 - d. Διαδικασίες Δείκτες **F₀₃**

1. Δυναμικοί Πίνακες `allocatable`

```
τύπος, ALLOCATABLE::όνομα πίνακα1 (: [: ::...:]) &  
      [,όνομα πίνακα2 (: [: ::...:]), ...]
```

!στην παρένθεση χρησιμοποιούμε ένα ':'

!για κάθε διάσταση του πίνακα

....

```
ALLOCATE (όνομα πίνακα1 (ενδείκτης1 [, ενδείκτης2, ...:]), &  
          [STAT = όνομα ελέγχου])
```

! δίνουμε τις ακριβείς διαστάσεις του πίνακα

....

```
DEALLOCATE (όνομα πίνακα1)
```

```
integer, allocatable::a (:)
```

```
allocate (a(3)); print '(3i8)', a
```

```
  a = [1,2,3]; print '(3i8)', a
```

```
deallocate (a)
```

```
allocate (a(5)); print '(5i8)', a
```

```
  a = [2,4,6,8,10]; print '(5i8)', a
```

```
end
```

3616360	3604672	0		
1	2	3		
3620048	3615952	0	0	0
2	4	6	8	10


Αυτόματη δέσμευση στην F₀₃

```
program autoallocation
  implicit none
  integer          :: i
  real, allocatable :: a(:)
  real             :: c(5) = [(i, i = 1,5)]
  real             :: d(2,3) = reshape([(i, i = 1,6)], [2,3])

!allocation F03
a = c; print '(5f5.2)', a
print '(a15,i2,a9)', ' the array has', size(a), 'elements'
a = d(1,:); print '(3f5.2)', a
print '(a15,i2,a9)', ' the array has', size(a), 'elements'

end program autoallocation
```

```
1.00 2.00 3.00 4.00 5.00
the array has 5 elements
1.00 3.00 5.00
the array has 3 elements
```

 Η fortran 2003 επιτρέπει την αυτόματη δέσμευση και επαναδέσμευση **allocatable** πινάκων με εκφράσεις ανάθεσης αρκεί να πρόκειται για πίνακες της ίδιας τάξης.

Γεννήτρια n τυχαίων στο διάστημα $[a, b]$...σωστά

1

```
program randomizer !n τυχαίοι αριθμοί στο διάστημα [a,b]
  implicit none
! δηλώσεις:
  integer          :: n
  real             :: a, b
  real,allocatable:: harvest(:)

! αρχή:
  print*, 'how many numbers'
  read *, n; allocate(harvest(n))!allocation (δέσμευση μνήμης)
  print*, 'give the range of numbers: [a,b]'
  read *, a, b

  call init_random_seed() !αρχικοποίηση της γεννήτριας
  call random_number(harvest)
  harvest = a + harvest * (b - a) !ανάπτυξη στο [a,b]
  print '(4g15.8)', harvest !τυπώνει 4 τιμές σε κάθε σειρά
```

contains

Γεννήτρια n τυχαίων στο διάστημα $[a, b]$...σωστά

2

```
subroutine init_random_seed !αρχικοποίηση γεννήτριας
  integer                :: i, n, clock
  integer, allocatable :: seed(:)

  call random_seed(size = n) !προσδιορισμός μεγέθους σπόρου

  allocate(seed(n)) !allocation (δέσμευση μνήμης)

  call system_clock(count = clock)

  seed = clock + 37 * [(i-1, i = 1, n)]

  call random_seed(put = seed) !χρήση τυχαίου σπόρου

  deallocate(seed) !αποδέσμευση μνήμης

end subroutine init_random_seed
```

```
end program randomizer
```

Πίνακες & Διαδικασίες

1. Εικονικοί Πίνακες Ρητής Μορφής
2. Δυναμικοί Πίνακες Διαδικασιών
 - a. Εικονικοί Πίνακες Υποθετικής Μορφής
 - b. Εικονικοί Πίνακες Υποθετικού Μεγέθους **F₇₇**
 - c. Αυτόματοι Πίνακες
 - d. Συναρτήσεις με αποτέλεσμα πίνακα
 - e. Εικονικοί **allocatable** πίνακες **F₀₃**
 - f. **allocatable** functions **F₀₃**

1. Εικονικοί Πίνακες Ρητής Μορφής

```
program explicit_shape_arrays
  implicit none
  integer :: i
  real    :: y(-4:5) = [(i, i = 1, 10)]
  print ' (5f5.2) ', f(y, size(y))
contains
  pure real function f(x,n)
    integer, intent(in) :: n
    real,    intent(in) :: x(n)
    integer          :: i

    f = 0.0
    do i = 1, size(x), 2
      f = f + x(i)
    end do
  end function f
end program explicit_shape_arrays
!τυπώνει: 25.00
```

Δήλωση εικονικού πίνακα
ρητής μορφής

2.a. Εικονικοί Πίνακες Υποθετικής Μορφής

```
program assumed_shape_arrays
  implicit none
  integer :: i
  real    :: y(10) = [(i, i = 1, 10)]
  print ' (5f5.2) ', f(y)
contains
  pure real function f(x)
  real, intent(in) :: x(:)
  integer          :: i

  f = 0.0
  do i = 1, size(x), 2
    f = f + x(i)
  end do

end function f
```

Δήλωση εικονικού πίνακα
υποθετικής μορφής

```
end program assumed_shape_arrays
!τυπώνει: 25.00
```

2.σ. Τοπικοί Αυτόματοι Πίνακες

```
program automatic_arrays
  implicit none
  integer          :: i, n
  real, allocatable :: y(:)
  print*, 'how many'; read *, n
  allocate(y(n)); y = [(i, i = 1, n)]
  print ' (5f5.2) ', f(y)
```

contains

```
  pure real function f(x)
    real, intent(in) :: x(:)
    real              :: z(size(x)/2)
```

```
    z = x(::2)
```

```
    f = sum(z)
```

```
  end function f
```

```
end program automatic_arrays
```

```
!για n=10 τυπώνει: 25.00
```

Δήλωση τοπικού
αυτόματου πίνακα

2.d. Function με αποτέλεσμα Πίνακα

```
program array_valued_function
  implicit none
  integer, allocatable, dimension(:) :: x, y
  integer                               :: i, n

  call execute_command_line('chcp 1253')
  print *, 'πλήθος ακεραίων;'; read *, n

  x = [(i, i = 1, n)] !allocation F03
  y = f(x)           !allocation F03
  print '(a35)', 'Υπάρχουν οι εξής πρώτοι αριθμοί:'

  do i = 2, n
    if(y(i) /= 0) print *, y(i)
  end do

contains
```

2.d. Functions με αποτέλεσμα Πίνακα

```
pure function f(x)
  integer, intent(in) :: x(:)           !υποθετικός
  integer              :: f(size(x))    !αυτόματος
  integer              :: i

  f = x
  f(1) = 0
  do i = 2, size(f)
    if(f(i) /= 0) then                  ! Αν ο i είναι πρώτος...
      f(2*i:size(f):i) = 0             ! μηδενίζω τα πολλαπλάσιά του
    endif
  enddo
end function f

end program array_valued_f
```

```
πλήθος ακεραίων;
10
Υπάρχουν οι εξής πρώτοι αριθμοί:
2
3
5
7
```

2.e. Εικονικοί Allocatable πίνακες F₀₃

```
program Eratosthenes
  implicit none
  ! δηλώσεις:
  integer :: i, n
  integer, allocatable :: x(:), y(:)

  ! αρχή:
  call execute_command_line('chcp 1253')
  print *, 'πλήθος ακεραίων: '; read *, n

  allocate(x(n)); x = [(i, i = 1, n)]
  print '(a35)', 'Υπάρχουν οι εξής πρώτοι αριθμοί: '
  call primes(x, y); print '(5i7)', y

contains
```

2.e. Εικονικοί Allocatable πίνακες F₀₃

```
pure subroutine primes(x, y)
  integer, intent(in) :: x(:) !υποθετικός
  integer, allocatable, intent(out) :: y(:) !allocatable
  integer :: i, nr_primes

  y = x !first allocation F03
  y(1) = 0
  do i = 2, size(y)
    if(y(i) /= 0) y(2*i:size(y):i) = 0
  enddo

  nr_primes = count(y /= 0)
  y(1:nr_primes) = pack(y, y /= 0)
  y = y(1:nr_primes) !second allocation F03

end subroutine primes
end program Eratosthenes
```

πλήθος ακεραίων;
10
Υπάρχουν οι εξής πρώτοι αριθμοί:
2 3 5 7

2.f. Allocatable Functions **F₀₃**

```
program Eratosthenes
  implicit none
  integer          :: i, n
  integer, allocatable :: y(:)

  call execute_command_line('chcp 1253')
  print *, 'πλήθος ακεραίων;'; read *, n

  y = [(i, i = 1, n)]
  print '(a35)', 'Υπάρχουν οι εξής πρώτοι αριθμοί:'
  print '(5i7)', f(y)

contains
  pure function f(x)
    integer, intent(in) :: x(:) !υποθετικός
    integer, allocatable :: f(:) !allocatable
    integer               :: i, primes
```

2.f. Allocatable Functions F₀₃

```
f = x ! first allocation F03; f(1) = 0
do i = 2, size(f)
  if(f(i) /= 0) then !Αν ο i είναι πρώτος
    f(2*i:size(f):i) = 0
  end if
end do
primes = count(f /= 0);
f(1:primes) = pack(f, f /= 0)
f = f(1:primes) !second allocation F03
               !ή αλλιώς allocate(f(primes))
               !f = z(1:primes)

end function f
end program Eratosthenes
```

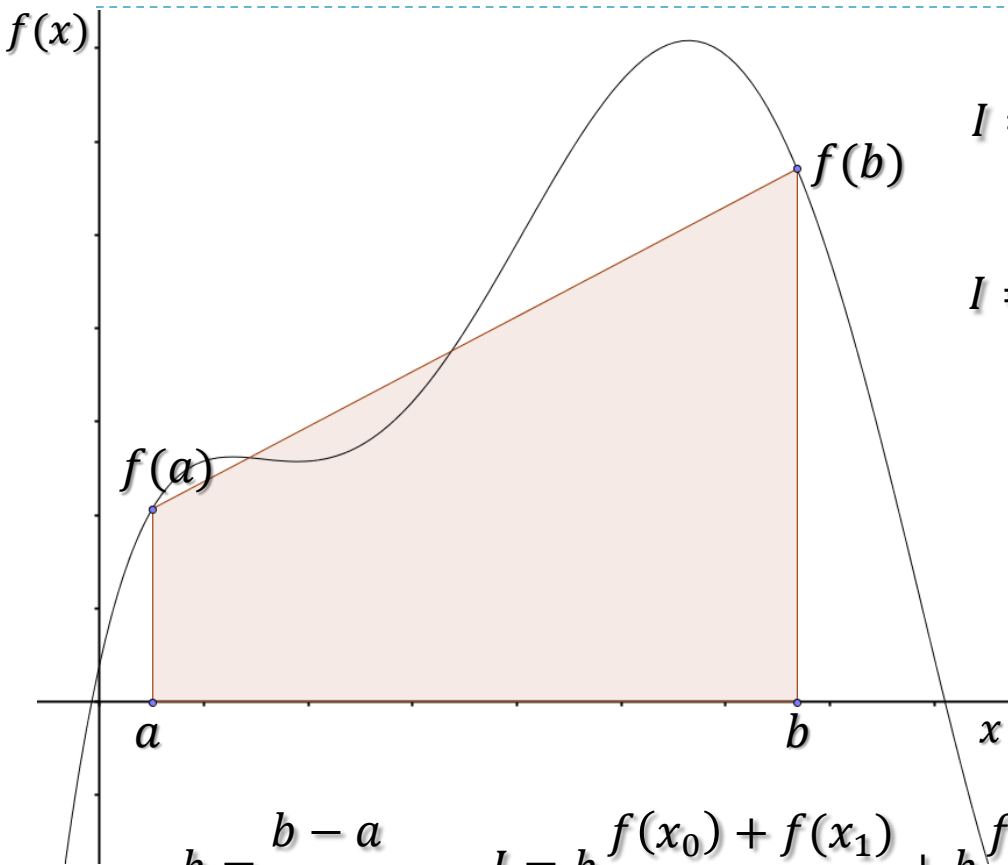
πλήθος ακεραίων;

100

Υπάρχουν οι εξής πρώτοι αριθμοί:

2	3	5	7	11
13	17	19	23	29
31	37	41	43	47
53	59	61	67	71
73	79	83	89	97

Η μέθοδος του τραπεζίου



$$I = \int_a^b \left[f(a) + \frac{f(b) - f(a)}{b - a} (x - a) \right] dx$$

$$I = (b - a) \frac{f(a) + f(b)}{2}$$

$$h = \frac{b - a}{n}$$

$$I = h \frac{f(x_0) + f(x_1)}{2} + h \frac{f(x_1) + f(x_2)}{2} + \dots + h \frac{f(x_{n-1}) + f(x_n)}{2}$$

$$I = \frac{h}{2} \left[f(x_0) + 2 \sum_{i=1}^{n-1} f(x_i) + f(x_n) \right]$$

Η μέθοδος του τραπεζίου

```
program integrator
  implicit none
```

```
interface
  real(kind=8) function f(x)
    real(kind=8), intent(in) :: x
  end function f
end interface
```

Ρητή Διεπιφάνεια
Εξωτερικής Διαδικασίας

```
real(kind=8) :: area
```

Ο όρος του ορίσματος είναι διαδικασία!!

```
area = integrate(f, a = 1.0_8, b = 5.0_8, n = 10000_8)
print '(a35,f10.6)', 'the integral of the function is:', area
```

Keyword Argument

contains

```
real(kind=8) function integrate(f, a, b, n)
```

```
interface
  real(kind=8) function f(x)
    real(kind=8), intent(in) :: x
  end function f
end interface
```

Ρητή Διεπιφάνεια
Εξωτερικής Διαδικασίας

Η μέθοδος του τραπεζίου

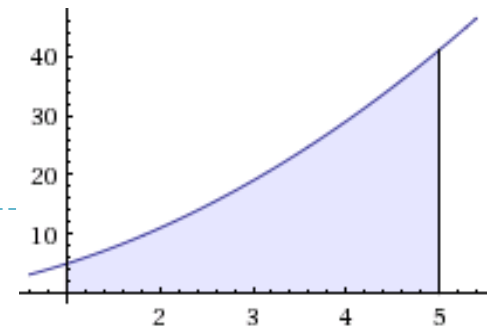
```
real(kind=8), intent(in) :: a, b
integer(kind=8), intent(in) :: n
real(kind=8) :: h, s
integer(kind=8) :: i
h = (b - a)/real(n, kind=8)
s = (f(a) + f(b)) / 2
do i = 1, n - 1
    s = s + f(a + i * h)
end do
integrate = h * s
end function integrate
end program integrator
```

```
real(kind=8) function f(x)
real(kind=8), intent(in) :: x
!f=exp(-x)
f = x**2 + 3 * x + 1
!f=cos(x)/x
end function f
```

Εξωτερική Διαδικασία

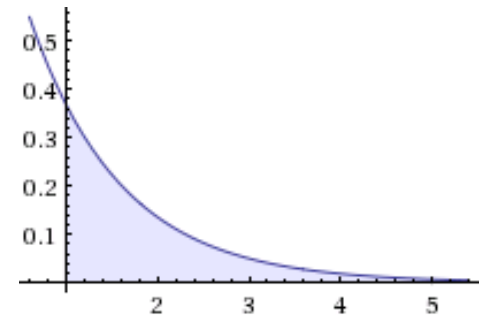
Η μέθοδος του τραπεζίου

the integral of the function is: 81.333333



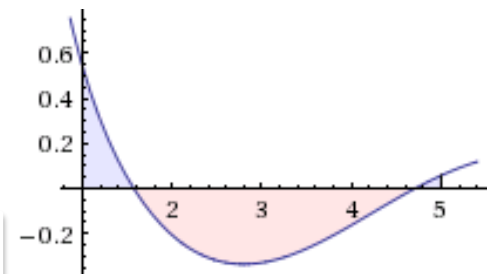
$$\int_1^5 (x^2 + 3x + 1) dx = \frac{244}{3}$$

the integral of the function is: 0.361141



$$\int_1^5 e^{-x} dx = \frac{e^4 - 1}{e^5} \approx 0.361141$$

the integral of the function is: -0.527434



$$\int_1^5 \frac{\cos(x)}{x} dx = \text{Ci}(5) - \text{Ci}(1) \approx -0.527434$$

Συνοψίζουμε τα είδη πινάκων



```
integer, parameter :: n = 5; integer :: i
real                :: a(n) !στατικός πίνακας
real, allocatable   :: b(:) !δυναμικός πίνακας (allocatable)
```

```
a = [(i * i, i = 1, n)]; b = a ! allocation F03
a = f(a, b, n); print '(5f8.2)', a
```

contains

```
function f(x, y, m) ! allocatable array valued function
integer          :: m          ! εικονική μεταβλητή
real             :: x(m)       ! ρητός εικονικός
real             :: y(:)       ! υποθετικός εικονικός
real             :: z(m)       ! αυτόματος
real, allocatable :: f(:)      ! allocatable
```

```
y = sqrt(x); z = x * y
f = z      ! allocation F03
```

```
end function f
```

```
end
```


Αναδρομικές Διαδικασίες

α) το παραγοντικό

```
program factorial
  implicit none

! δηλώσεις:
  integer(8) :: f, n, i

! αρχή:

! επικεφαλίδα
print '(t3, a1, t32, a2)', 'n', 'n!'
write (*, '(t2, 32a1)') ('=', i = 1, 32)

do n = 1, 20
  f = fact(n)      ! με function
  call fn(n,f)     ! ή με subroutine
  print '(x, i2, i30)', n, f
enddo
```

$$n! = \begin{cases} n(n-1)! & \forall n \geq 1 \\ 1 & n = 0 \end{cases}$$

contains

Αναδρομικές Διαδικασίες

α₁) το παραγοντικό με function

```
recursive function fact(n) result(factor)
```

```
integer(8), intent(in) :: n
```

```
integer(8)
```

```
:: factor
```

υποχρεωτικό στις recursive functions

```
if(n >= 1) then
```

```
factor = n * fact(n-1)
```

```
else
```

```
factor = 1
```

```
end if
```

αναδρομική κλήση

τελική έκφραση

```
end function fact
```

Αναδρομικές Διαδικασίες

```
recursive function fact(n
  integer(8), intent(in) :
  integer(8)           :

  if(n >= 1) then
    factor = n * fact(n-
  else
    factor = 1
  end if

end function fact
```

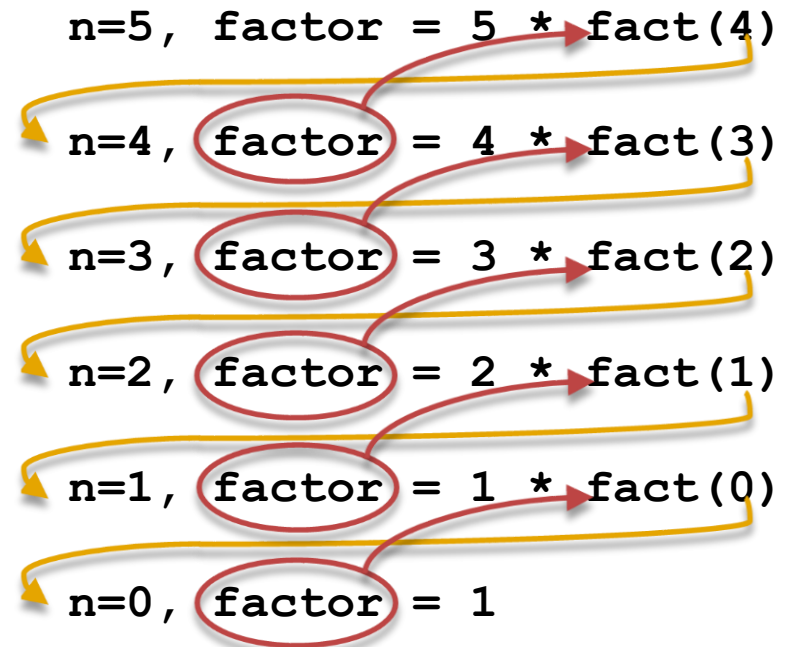
n	n!
1	1
2	2
3	6
4	24
5	120
6	720
7	5040
8	40320
9	362880
10	3628800
11	39916800
12	479001600
13	6227020800
14	87178291200
15	1307674368000
16	20922789888000
17	355687428096000
18	6402373705728000
19	121645100408832000
20	2432902008176640000

Αναδρομικές Διαδικασίες

α₁) το παραγοντικό με function

- ▶ Π.χ. Το παραγοντικό του 5:

```
if(n >= 1) then
  factor = n * fact(n-1)
else
  factor = 1
end if
```



Κλήση: →

Επιστροφή: ←

Αναδρομικές Διαδικασίες

α₂) το παραγοντικό με subroutine

```
recursive subroutine fn(n, factor)
  integer(8), intent(in) :: n
  integer(8), intent(out) :: factor
  integer(8) :: temp
```

```
if(n >= 1) then
```

```
  ↗ call fn(n-1, temp)
```

```
  factor = temp * n
```

αναδρομική κλήση

```
else
```

```
  factor = 1
```

```
end if
```

τελική έκφραση

```
end subroutine fn
```

```
end program factorial
```

Αναδρομικές Διαδικασίες

β) Η ακολουθία Fibonacci

```
program fibonacci
  implicit none
!  δηλώσεις:
  integer :: f, n
  real    :: ratio
```

```
! αρχή:
```

```
! επικεφαλίδα
```

```
print '(t3,a1,t8,a1,t13,a5)', 'n', 'f', 'ratio'
```

```
print '(a19)', ' ===== '
```

```
open (1,file='fibs.txt')
```

```
do n = 1, 20
```

```
  call fibs(n, f, ratio)
```

```
  print '(x,i2,i5,2x,f9.7)', n, f, ratio
```

```
  write (1, *) n, f, ratio
```

```
end do
```

```
close (1)
```

```
contains
```

$$F(n) = \begin{cases} F(n-1) + F(n-2) & \forall n > 2 \\ 1 & n = 1 \\ 1 & n = 2 \end{cases}$$

$$\frac{a}{b}$$

$$\frac{a+b}{a} = \frac{a}{b} = \varphi$$

$$\varphi = \frac{1 + \sqrt{5}}{2}$$

Αναδρομικές Διαδικασίες

β) Η ακολουθία Fibonacci

```
recursive subroutine fibs(n,f,r)
  integer,intent(in)      :: n
  integer,intent(out)    :: f
  real,intent(out)       :: r
  integer                :: t1, t2

  if(n > 2) then
    call fibs(n-1, t1, r)    !αναδρομική κλήση
    call fibs(n-2, t2, r)    !αναδρομική κλήση
    f = t1 + t2
    r = real(t1) / real(f)
  else
    f = 1
    r = 1.
  end if
end subroutine fibs

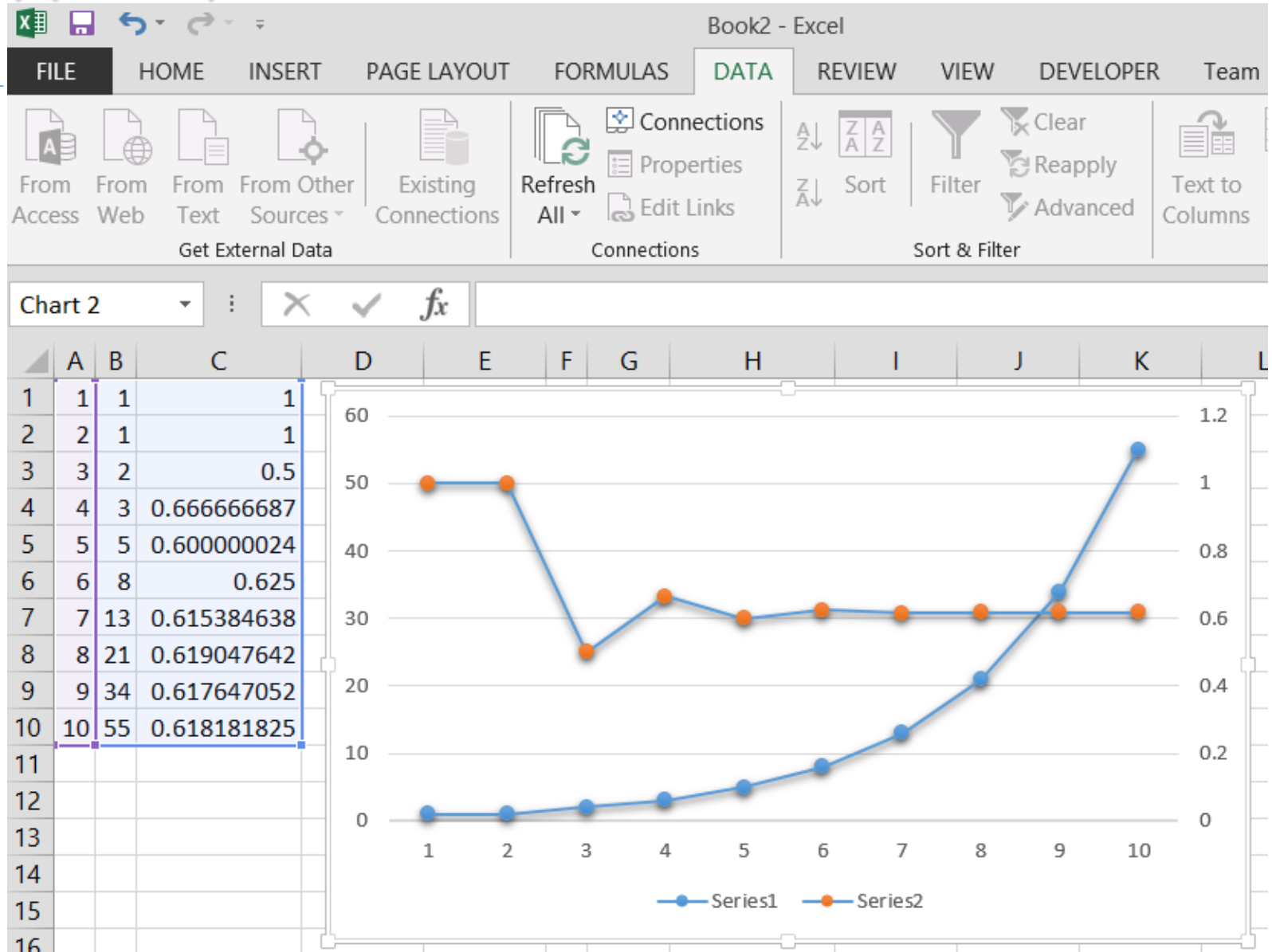
end program fibonacci
```


Αναδρομικές Διαδικασίες

γ) Η ακολουθία Fibonacci

n	f	ratio
1	1	1.0000000
2	1	1.0000000
3	2	0.5000000
4	3	0.6666667
5	5	0.6000000
6	8	0.6250000
7	13	0.6153846
8	21	0.6190476
9	34	0.6176471
10	55	0.6181818
11	89	0.6179775
12	144	0.6180556
13	233	0.6180258
14	377	0.6180372
15	610	0.6180328
16	987	0.6180344
17	1597	0.6180338
18	2584	0.6180341
19	4181	0.6180339
20	6765	0.6180340

Συνεργασία με το excel



Αναδρομικές Διαδικασίες

γ) αναδρομικός Ευκλείδης

```
program recursive_gcd
  implicit none
  integer :: a, b, c
  call execute_command_line('chcp 1253')
  do
    print *, 'Δώσε δύο ακέραιους'; read *, a , b
    if(a == 0 .or. b == 0) exit
    print '(a18,i4)', 'Ο ΜΚΔ τους είναι=', gcd(abs(a), abs(b))
  end do
contains
  recursive function gcd(x, y) result(z)
    integer:: x, y, z
    if(y == 0) then
      z = x
    else
      z = gcd(y, mod(x,y))
    end if
  end function gcd
end program recursive_gcd
```

Αναδρομικές Διαδικασίες

δ) αναδρομική ολοκλήρωση

```
program integrator
  implicit none
  interface
    real(8) function f(x)
      real(8), intent(in) :: x
    end function f
  end interface
  real(8) :: area
  area = integrate(f, a = 1.0_8, b = 5.0_8, tol = 1e-7_8)
  print '(*(g0.6, :, " "))', &
    &'the integral of the function is:', area
contains
  the integral of the function is: 81.3333
```

το format που
τυπώνει τα πάντα!!!

```
recursive function integrate(f, a, b, tol) result(area)
```

```
  interface
    real(8) function f(x)
      real(8), intent(in) :: x
    end function f
  end interface
```

Αναδρομικές Διαδικασίες

δ) αναδρομική ολοκλήρωση

```
real(8), intent(in)::a, b, tol
real(8):: area, h, mid, one_trap, two_traps, &
           left_area, right_area
h = b - a; mid = (a + b) / 2
one_trap = h * (f(a) + f(b)) / 2
two_traps = h * ((f(a) + f(mid)) + (f(mid) + f(b))) / 4
if (abs(one_trap - two_traps) < 3 * tol) then
    area = two_traps
else
    left_area = integrate(f, a, mid, tol / 2)
    right_area = integrate(f, mid, b, tol / 2)
    area = left_area + right_area
end if
end function integrate
end program integrator
real(8) function f(x)
    real(8), intent(in)::x
    f = x**2 + 3 * x + 1
end function f
```

```
program matrix_timer
  implicit none
  integer, parameter :: n = 1000
  real, dimension(n,n) :: a, b, c
  character(len=8) :: cdate, ctime
  real :: start_time, stop_time
  integer :: i, j, k
  character(len=*), parameter :: form = "(t2, a, f0.3, a)"
```

! Αρχή:

```
call execute_command_line('chcp 1253')
call date_and_time(date = cdate, time = ctime)

print *, "Ημερομηνία: " // &
& cdate(7:) // "-" // cdate(5:6) // "-" // cdate(:4), &
& ", Ωρα: " // &
& ctime(:2) // ":" // ctime(3:4) // ":" // ctime(5:)
```

```
call random_seed()  
call random_number(a)  
call random_number(b)
```

```
call cpu_time(start_time)  
  c = 0  
  do k = 1, n  
    do j = 1, n  
      do i = 1, n  
        c(i, j) = c(i, j) + a(i, k) * b(k, j)  
      end do; end do; end do  
call cpu_time(stop_time)
```

```
print *  
print form, "Time of 1st DO loop version is: ", &  
  &stop_time - start_time, " seconds."
```



```
call cpu_time(start_time)
  c = 0
  do i=1,size(c,1)
    do j=1,size(c,2)
      c(i,j)=sum(a(i,:)*b(:,j))
    enddo; enddo
call cpu_time(stop_time)

print *
print form, "Time of 2nd DO loop version is: ", &
  &stop_time - start_time, " seconds."
```

```
call cpu_time(start_time)
  c = 0
  forall (i = 1:size(c,1), j = 1:size(c,2))
    c(i,j) = sum(a(i,:) * b(:,j))
  end forall
call cpu_time(stop_time)
print *
print form, "Time of 3rd DO loop version is: ", &
  & stop_time - start_time, " seconds."
```

```
call cpu_time(start_time)
  c = matmul(a, b)
call cpu_time(stop_time)

print *
print form, "      Time of matmul version is: ", &
  & stop_time - start_time, " seconds."
```

```
end program matrix_timer
```

DEBUG

```
Ημερομηνία: 10-12-2013, Ώρα: 19:10:26.8  
  
Time of 1st D0 loop version is: 3.000 seconds.  
  
Time of 2nd D0 loop version is: 6.406 seconds.  
  
Time of 3rd D0 loop version is: 6.469 seconds.  
  
Time of matmul version is: .609 seconds.
```

RELEASE

```
Ημερομηνία: 10-12-2013, Ώρα: 19:10:26.8  
  
Time of 1st D0 loop version is: .234 seconds.  
  
Time of 2nd D0 loop version is: 6.578 seconds.  
  
Time of 3rd D0 loop version is: .812 seconds.  
  
Time of matmul version is: .641 seconds.
```

Σημείωμα Αναφοράς

Copyright Πανεπιστήμιο Πατρών, Όνομα μέλους ή μελών ΔΕΠ 2014:
Δημήτριος Ματαράς. «Εισαγωγή στον Προγραμματισμό Η/Υ». Έκδοση: 1.0.
Πάτρα 2014. Διαθέσιμο από τη δικτυακή διεύθυνση:
<https://eclass.upatras.gr/courses/CMNG2178>.

Χρηματοδότηση

- ▶ Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στο πλαίσιο του εκπαιδευτικού έργου του διδάσκοντα.
- ▶ Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Αθηνών**» έχει χρηματοδοτήσει μόνο την αναδιαμόρφωση του εκπαιδευτικού υλικού.
- ▶ Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.



Σημείωμα Αδειοδότησης

Το παρόν υλικό διατίθεται με τους όρους της άδειας χρήσης Creative Commons Αναφορά, Μη Εμπορική Χρήση Παρόμοια Διανομή 4.0 [1] ή μεταγενέστερη, Διεθνής Έκδοση. Εξαιρούνται τα αυτοτελή έργα τρίτων π.χ. φωτογραφίες, διαγράμματα κ.λ.π., τα οποία εμπεριέχονται σε αυτό και τα οποία αναφέρονται μαζί με τους όρους χρήσης τους στο «Σημείωμα Χρήσης Έργων Τρίτων».



[1] <http://creativecommons.org/licenses/by-nc-sa/4.0/>

Ως **Μη Εμπορική** ορίζεται η χρήση:

- που δεν περιλαμβάνει άμεσο ή έμμεσο οικονομικό όφελος από την χρήση του έργου, για το διανομέα του έργου και αδειοδόχο
- που δεν περιλαμβάνει οικονομική συναλλαγή ως προϋπόθεση για τη χρήση ή πρόσβαση στο έργο
- που δεν προσπορίζει στο διανομέα του έργου και αδειοδόχο έμμεσο οικονομικό όφελος (π.χ. διαφημίσεις) από την προβολή του έργου σε διαδικτυακό τόπο

Ο δικαιούχος μπορεί να παρέχει στον αδειοδόχο ξεχωριστή άδεια να χρησιμοποιεί το έργο για εμπορική χρήση, εφόσον αυτό του ζητηθεί.