Fortran και Αντικειμενοστραφής προγραμματισμός

www.corelab.ntua.gr/courses/fortran_naval/naval

Δδάσκοντες: Άρης Παγουρτζής (pagour@cs.ntua.gr)
(Επίκουρος Καθηγητής ΣΗΜΜΥ)
Δώρα Σούλιου (dsouliou@mail.ntua.gr)
(ΕΔΙΠ ΣΗΜΜΥ)

6η ενότητα

- ✓ Υπορουτίνες
- ✓ Συναρτήσεις
- ✓ Πέρασμα παραμέτρων
- ✓ Αναδρομικά υποπρογράμματα
- ✓ Hanoi, Mergesort, Quicksort

Συναρτήσεις και Υπορουτίνες (i)

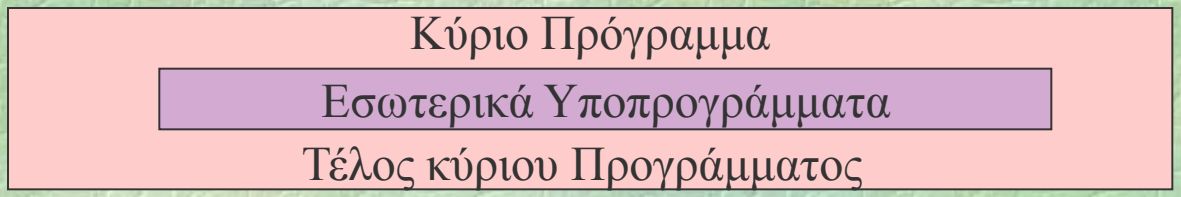
- Ομάδες εντολών που εκτελούν μια συγκεκριμένη εργασία (μια σειρά υπολογισμών) ή που επαναλαμβάνονται συχνά και αποτελούν ανεξάρτητες προγραμματιστικές οντότητες μπορούν να μετακινηθούν έξω από το κυρίως σώμα του προγράμματος σε υποπρογράμματα.
- Έτσι απλοποιείται το κυρίως πρόγραμμα και τα υποπρογράμματα μπορούν να διορθωθούν ή τροποποιηθούν πιο εύκολα.
- 2 Κατηγορίες υποπρογραμμάτων: υπορουτίνες και συναρτήσεις

Συναρτήσεις και Υπορουτίνες (i)

- Οι συναρτήσεις δηλώνονται με την εντολή **`FUNCTION`** και οι υπορουτίνες με την εντολή **`SUBROUTINE`**.
- Η βασική διαφορά τους είναι πως οι συναρτήσεις επιστρέφουν τουλάχιστον μια τιμή στο πρόγραμμα που τις καλεί ενώ οι υπορουτίνες μπορεί και να μην επιστρέφουν.
- Συντακτικά, οι συναρτήσεις έχουν ρόλο *παραστάσεων* (εκφράσεων), ενώ οι υπορουτίνες έχουν ρόλο *εντολών*.

Συναρτήσεις και Υπορουτίνες (ii)

- Η δομή ενός προγράμματος με υποπρογράμματα έχει ως εξής:



- Έχουμε ένα κυρίως πρόγραμμα που καλεί εσωτερικά τα υποπρογράμματά του (συναρτήσεις και υπορουτίνες).
- Τα εσωτερικά προγράμματα μπορούν να χρησιμοποιηθούν μόνο από το κυρίως πρόγραμμα που τα περιλαμβάνει.
- Τα υποπρογράμματα δεν μπορούν να περιλαμβάνουν άλλα προγράμματα αλλά μπορούν να καλούν το ένα το άλλο αν βρίσκονται εντός του ιδίου κυρίως προγράμματος.
- Υπάρχουν και τα εξωτερικά υποπρογράμματα που ουσιαστικά είναι ανεξάρτητα προγράμματα αποθηκευμένα σε διαφορετικά αρχεία και μπορούν να χρησιμοποιηθούν από οποιοδήποτε άλλο πρόγραμμα και μπορεί να περιέχουν εσωτερικά υποπρογράμματα αλλά και να καλούν άλλα εξωτερικά.
- Στη FORTRAN μπορούμε να συμπεριλάβουμε, σε οποιοδήποτε σημείο του κώδικα του κυρίως προγράμματος ή του υποπρογράμματος, κώδικα που είναι γραμμένος σε άλλο αρχείο χρησιμοποιώντας την εντολή INCLUDE όνομα αρχείου. Το νέο αρχείο μπορεί επίσης να έχει εντολές INCLUDE

Συναρτήσεις (i)

επιστρεφόμενος τύπος **FUNCTION** *όνομα* (όρισμα Α, όρισμα Β, …)
τύπος ορίσματος Α :: όρισμα Α
τύπος ορίσματος Β :: όρισμα Β
...
τύπος :: τοπική μεταβλητή
...
κώδικας
...
END FUNCTION όνομα συνάρτησης

Παράδειγμα:

```
REAL FUNCTION emvado_trigwnou(a,b,c)
IMPLICIT NONE
REAL :: a,b,c,temp
temp=(a+b+c)/2.0
emvado_trigwnou=SQRT(temp*(temp-a)*(temp-b)*(temp-c))
END
```

Συναρτήσεις (ii)

- Ο επιστρεφόμενος τύπος μπορεί να είναι μια δήλωση τύπου όπως real, integer, κ.λπ. ή μία από τις εντολές recursive, pure, elemental.
- Το όνομα της συνάρτησης θα πρέπει να δηλωθεί και στο κυρίως πρόγραμμα στις δηλώσεις μεταβλητών και ο τύπος να είναι ο ίδιος.
- Το όνομα είναι υποχρεωτικό και με αυτό καλείται η συνάρτηση. Ακολουθεί τους κανόνες ονοματολογίας.
- Ορίσματα είναι η λίστα των παραμέτρων με τις οποίες καλείται η συνάρτηση και τα οποία θα πρέπει να δηλωθούν σαν μεταβλητές αντίστοιχου τύπου. Αν δεν υπάρχουν ορίσματα βάζουμε απλά τις παρανθέσεις.
- Ορίζονται οι τοπικές μεταβλητές της συνάρτησης
- Προαιρετικά μπορεί να υπάρχει η εντολή RESULT (αποτέλεσμα). Στην παρένθεση υπάρχει το όνομα της μεταβλητής που επιστρέφει η συνάρτηση. Αν δεν υπάρχει αυτή η εντολή το όνομα της συνάρτησης είναι αυτή η μεταβλητή.
- Αν υπάρχει εντολή RETURN διακόπτεται η εκτέλεση της συνάρτησης και συνεχίζεται η εκτέλεση του κυρίως προγράμματος.

Συναρτήσεις (iii)

```
PROGRAM ypologismos_emvadou
IMPLICIT NONE
REAL::x,y,z,emv,emvado_trigwnou
READ(*,*) x,y,z
emv = emvado_trigwnou(x,y,z)
WRITE(*,*) emv
END

REAL FUNCTION emvado_trigwnou(a,b,c)
IMPLICIT NONE
REAL :: a,b,c,temp
temp=(a+b+c)/2.0
emvado_trigwnou=SQRT(temp*(temp-a)*(temp-b)*(temp-c))
END
```

Συναρτήσεις (iv)

```
PROGRAM ypologismos_paragontikou_1
IMPLICIT NONE
INTEGER:: x, paragontiko
READ(*,*) x
WRITE(*,*) "To paragontiko einai", paragontiko(x)
END

INTEGER FUNCTION paragontiko(a)
IMPLICIT NONE
INTEGER :: a,i,p=1
DO i=1,a
   p=p*i
END DO
paragontiko = p
END
```

Συναρτήσεις (v)

2ος τρόπος δήλωσης με την εντολή **CONTAINS**
PROGRAM όνομα προγράμματος
Δηλώσεις μεταβλητών
Κλήση συναρτήσεων με ορίσματα
CONTAINS

FUNCTION όνομα 1ης συνάρτησης(όρισμα Α, όρισμα Β, …)
Δηλώσεις μεταβλητών καθώς και του τύπου επιστροφής της συνάρτησης
Υπολογισμός τιμής της συνάρτησης
END FUNCTION

FUNCTION όνομα 2ης συνάρτησης(όρισμα Α, όρισμα Β, …)
Δηλώσεις μεταβλητών καθώς και του τύπου επιστροφής της συνάρτησης
Υπολογισμός τιμής της συνάρτησης
END FUNCTION
END PROGRAM

Συναρτήσεις (vi)

2ος τρόπος δήλωσης με την εντολή **CONTAINS**

```
PROGRAM Synarthseis
IMPLICIT NONE
INTEGER:: x=12,y=10
WRITE(*,*) athroisma(x,y);  WRITE(*,*) diafora(x,y)
CONTAINS

FUNCTION athroisma(a,b)
IMPLICIT NONE
INTEGER:: athroisma, a,b
athroisma=a+b
END FUNCTION athroisma

FUNCTION diafora(c,d)
IMPLICIT NONE
INTEGER:: diafora,c,d
diafora=c-d
END FUNCTION diafora
END PROGRAM
```

Συναρτήσεις: πέρασμα παραμέτρων

```
PROGRAM perasma_timi_anafora
IMPLICIT NONE
INTEGER:: x=5,y=10,k,l,syn1,syn2
WRITE(*,*) x,y
k=syn1(x)
l=syn2(y)
WRITE(*,*) x,y
END
INTEGER FUNCTION syn1(a)
IMPLICIT NONE
INTEGER,INTENT(in):: a          ! Πέρασμα κατά τιμή
a=a*10                          ! Compilation error
syn1=5
END FUNCTION syn1
INTEGER FUNCTION syn2(b)
IMPLICIT NONE
INTEGER,INTENT(inout):: b       ! Πέρασμα κατ'αναφορά
b=b*10
syn2=30
END FUNCTION syn2
```

Συναρτήσεις: πέρασμα παραμέτρων

```
PROGRAM perasma_anafora_synartisi
IMPLICIT NONE
INTEGER::x,y,w,allagi
READ(*,*) x,y
WRITE(*,*) “arxika x kai y: ", x, y
w=allagi(x,y)
WRITE(*,*) “telika x kai y: ", x, y
END

INTEGER FUNCTION allagi(a,b)
IMPLICIT NONE
INTEGER :: a,b,temp
temp=a
a=b              ! Συμπεραίνεται: a κατ’αναφορά
b=temp                ! Συμπεραίνεται: b κατ’αναφορά
allagi=temp
END
```

Πληκτρολογήστε το πρόγραμμα και δείτε ποιές είναι οι τιμές των x, y πρίν και μετά την κλήση της συνάρτησης

Υπορουτίνες

```
PROGRAM perasma_anafora_diadikasia
IMPLICIT NONE
INTEGER::x,y
READ(*,*) x,y
WRITE(*,*) “arxika x kai y: ", x, y
CALL allagi(x,y)
WRITE(*,*) “telika x kai y: ", x, y
END

SUBROUTINE allagi(a,b)
IMPLICIT NONE
INTEGER :: a,b,temp
temp=a
a=b              ! Συμπεραίνεται: a κατ’αναφορά
b=temp                ! Συμπεραίνεται: b κατ’αναφορά
END
```

Πληκτρολογήστε το πρόγραμμα και δείτε ποιές είναι οι τιμές των x, y πρίν και μετά την κλήση της συνάρτησης

Συναρτήσεις με αναδρομή

```
PROGRAM ypologismos_paragontikou_2
IMPLICIT NONE
INTEGER:: x, paragontiko
READ(*,*) x
WRITE(*,*) "To paragontiko einai", paragontiko(x)
END

INTEGER RECURSIVE FUNCTION paragontiko(a) result (parag_result)
IMPLICIT NONE
INTEGER :: a
IF (a==0) THEN
    parag_result=1
ELSE
    parag_result=a*paragontiko(a-1)
END IF
END
```

Υπορουτίνες με αναδρομή

```
PROGRAM hanoitowers
  IMPLICIT NONE
  INTEGER::n
  READ(*,*) n
  CALL hanoi(n,'a',‘b',‘c')
END

RECURSIVE SUBROUTINE hanoi(n,frompeg,auxpeg,topeg)
   IMPLICIT NONE
   INTEGER:: n
   CHARACTER:: frompeg,auxpeg,topeg
   IF (n == 1) THEN
       WRITE(*,101) 1, frompeg, topeg
   ELSE
       CALL hanoi(n-1,frompeg,topeg,auxpeg)
       WRITE(*,101) n, frompeg,topeg
       CALL hanoi(n-1,auxpeg,frompeg,topeg)
   ENDIF
   101 FORMAT(1x,"Move disk ",i1," from peg ", A1, " to peg ", A1)
 END
```

Υπορουτίνες με αναδρομή

```
PROGRAM hanoitowers
  IMPLICIT NONE
  INTEGER::n
  READ(*,*) n
  CALL hanoi(n,'a',‘b',‘c')
END

RECURSIVE SUBROUTINE hanoi(n,frompeg,
   IMPLICIT NONE
   INTEGER:: n
   CHARACTER:: frompeg,auxpeg,topeg
   IF (n == 1) THEN
      WRITE(*,101) 1, frompeg, topeg
   ELSE
      CALL hanoi(n-1,frompeg,topeg,auxpeg)
      WRITE(*,101) n, frompeg,topeg
      CALL hanoi(n-1,auxpeg,frompeg,topeg)
   ENDIF
   101 FORMAT(1x,"Move disk ",i1," from peg ", A1, " to peg ", A1)
 END
```

Έξοδος στην οθόνη

```
Move disk 1 from peg a to peg b
Move disk 2 from peg a to peg c
Move disk 1 from peg b to peg c
Move disk 3 from peg a to peg b
Move disk 1 from peg c to peg a
Move disk 2 from peg c to peg b
Move disk 1 from peg a to peg b
Move disk 4 from peg a to peg c
Move disk 1 from peg b to peg c
Move disk 2 from peg b to peg a
Move disk 1 from peg c to peg a
Move disk 3 from peg b to peg c
Move disk 1 from peg a to peg b
Move disk 2 from peg a to peg c
Move disk 1 from peg b to peg c
```

Merge Sort (i)

- Ταξινόμηση με συγχώνευση (Ταξινόμηση με συγχώνευση (MergeSort)
  - Διαιρώ την ακολουθία των αριθμών σε δύο μέρη
  - Με αναδρομικές κλήσεις, ταξινομώ τα δύο μέρη ανεξάρτητα
  - Συγχωνεύω τα δύο ταξινομημένα μέρη
- Στη χειρότερη περίπτωση απαιτούνται $O(n \log n)$ συγκρίσεις)

Merge Sort (ii)

```
program MergeSortAlgorithm

 integer, parameter :: N = 20
 integer, dimension(N) ::   &
   A=(/85,46,39,81,72,24,105,210,133,22,27,78,28,98,12,34,48,64,23,45/)
 integer, dimension ((N+1)/2) :: T

 write(*,'(A,/,10I4,/,10I4)') 'ΜΗ Ταξινομημένος Πίνακας :',A
 call MergeSort(A,N)
 write(*,'(A,/,10I4,/,10I4)') 'Ταξινομημένος Πίνακας :',A

end
```

Merge Sort: η βασική υπορουτίνα (iii)

```
recursive subroutine MergeSort(A,N)
  integer, intent(in) :: N
  integer, dimension(N), intent(in out) :: A
  integer, dimension((N+1)/2):: T1,T2
  integer :: mid,rest,S
    if (N < 2) return
    if (N == 2) then
      if (A(1) > A(2)) then
         S = A(1)
         A(1) = A(2)
         A(2) = S
      endif
      return
   endif
    .
    .
    .
```

```
    .
    .
    .
  mid=(N+1)/2
  rest=N-mid
  call MergeSort(A,mid)
  call MergeSort(A(mid+1),rest)
  if (A(mid) > A(mid+1)) then
     T1(1:mid)=A(1:mid)
     T2(1:rest)=A(mid+1:N)
     call Merge(T1,mid,T2,rest,A,N)
  endif
  return
end subroutine MergeSort
```

Merge Sort: υπορουτίνα Merge (iv)

```
subroutine Merge (A,NA,B,NB,C,NC)
  integer :: NA,NB,NC
  integer :: A(NA) ,B(NB),C(NC)
  integer :: I,J,K

  I = 1; J = 1; K = 1;

  do while(I <= NA .and. J <= NB)
    if (A(I) <= B(J)) then
      C(K) = A(I);          I = I+1
    else
      C(K) = B(J);          J = J+1
    endif
    K = K + 1
  enddo
    .
    .
    .
```

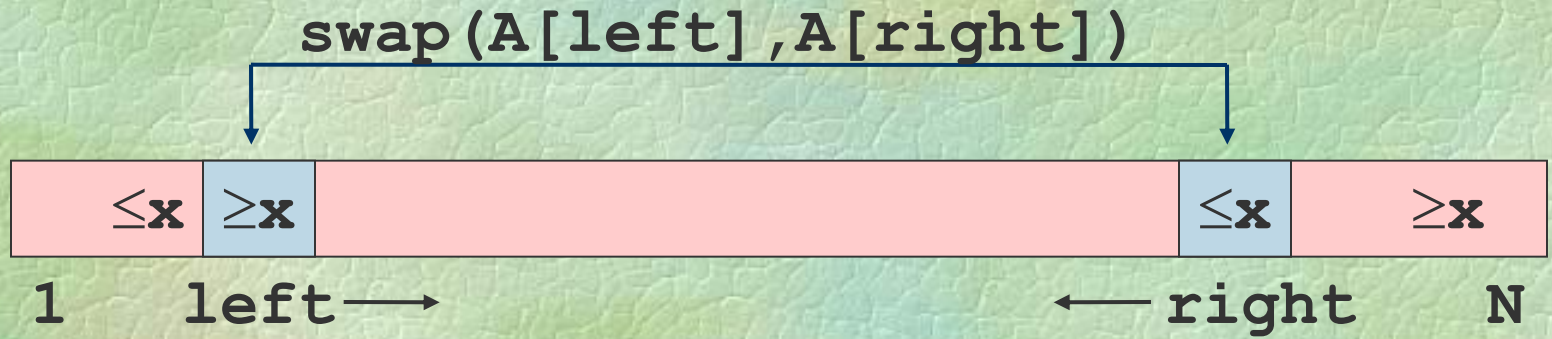
```
    .
    .
    .
  do while (I <= NA)
    C(K) = A(I)
    I = I + 1
    K = K + 1
  enddo

  do while (J <= NB)
    C(K) = B(J)
    J = J + 1
    K = K + 1
  enddo

end subroutine merge
```

Quicksort (i)

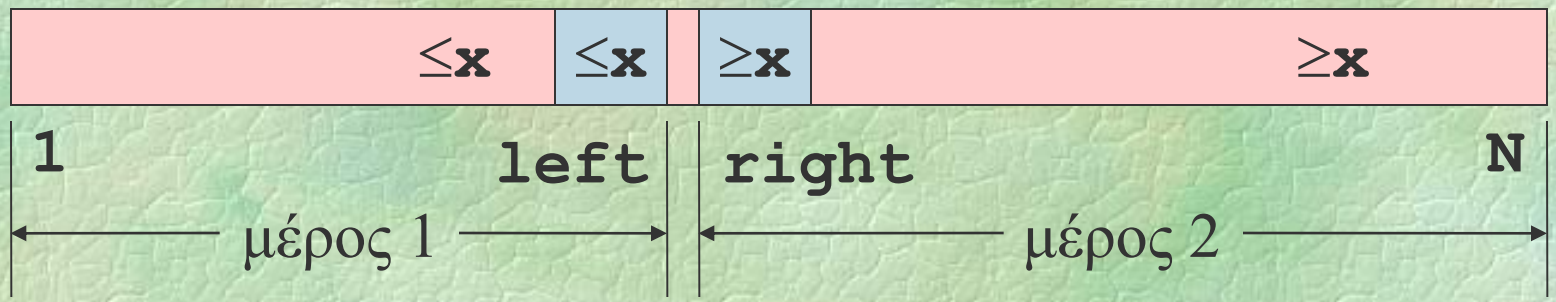
```
PROGRAM Quicksort
  USE Qsort_Module
  IMPLICIT NONE
  INTEGER, PARAMETER :: N = 20
  INTEGER :: A(n)
  INTEGER :: I
  write(*,*)"dwse times ston pinaka"
  DO I = 1,N
     read(*,*) A(I)
  END DO
  WRITE (*, "(A)") "O πίνακας πριν την ταξινόμηση"
  WRITE (*, "(10I5)") A
  CALL Qsort(A)
  WRITE (*,*)
  WRITE (*, "(A)") "O πίνακας μετά την ταξινόμηση"
  WRITE (*,"(10I5)") A
END PROGRAM Quicksort
```

Quicksort: ιδέα (ii)

- Σε κάθε βήμα της διαμέρισης



- Μετά τη διαμέριση



Quicksort: βασική υπορουτίνα (iii)

```
MODULE Qsort_Module
IMPLICIT NONE
CONTAINS
RECURSIVE SUBROUTINE Qsort(A)
  INTEGER, INTENT(IN OUT) :: A(:)
  INTEGER :: split
  IF(size(A) > 1) THEN
     CALL Partition(A, split)
     CALL Qsort(A(:split-1))
     CALL Qsort(A(split:))
  END IF
END SUBROUTINE Qsort
```

Quicksort: υπορουτίνα Partition (iv)

```
SUBROUTINE Partition(A, marker)
  INTEGER, INTENT(IN OUT) :: A(:)
  INTEGER, INTENT(OUT) :: marker
  INTEGER :: left, right, pivot, temp
  pivot = (A(1) + A(size(A))) / 2
  left = 0
  right = size(A) + 1
  DO WHILE (left < right)
     right = right - 1
     DO WHILE (A(right) > pivot)
        right = right-1
     END DO
     left = left + 1
     DO WHILE (A(left) < pivot)
        left = left + 1
     END DO
     IF (left < right) THEN
        temp = A(left)
        A(left) = A(right)
        A(right) = temp
     END IF
  END DO
```

```
IF (left == right) THEN
     marker = left + 1
  ELSE
     marker = left
  END IF
END SUBROUTINE Partition
END MODULE Qsort_Module
```