ΟΓΔΟΗ ΔΙΑΛΕΞΗ

Υποπρόγραμμα

Ένα σύνολο από εντολές που

- κάνουν κάτι συγκεκριμένο,
- έχουν στενή σχέση/εξάρτηση μεταξύ τους,
- έχουν «χαλαρή» σύνδεση με τον υπόλοιπο κώδικα,

μπορεί να εξαχθεί από το πρόγραμμά μας σε ξεχωριστό υποπρόγραμμα.

Υποπρόγραμμα

Ένα σύνολο από εντολές που

- κάνουν κάτι συγκεκριμένο,
- έχουν στενή σχέση/εξάρτηση μεταξύ τους,
- έχουν «χαλαρή» σύνδεση με τον υπόλοιπο κώδικα,

μπορεί να εξαχθεί από το πρόγραμμά μας σε ξεχωριστό υποπρόγραμμα.

Στη Fortran τα υποπρογράμματα διακρίνονται σε
συναρτήσεις (functions), αν επιστρέφουν τιμή στον κώδικα, και σε
υπορουτίνες (subroutines), αν δεν επιστρέφουν τιμή στον κώδικα.

Συνάρτηση στα μαθηματικά (ως συμβολισμός)

Ο ορισμός $f(x) = 5x^2 + 2$ σημαίνει ότι η έκφραση $5x^2 + 2$

1. έχει αποκτήσει ένα όνομα, το f,
2. εξαρτάται από μία παράμετρο, το x, για την οποία γνωρίζουμε μόνο ότι είναι πραγματικός αριθμός,
3. δεν μπορεί να υπολογιστεί και να δώσει αποτέλεσμα παρά μόνο αφού προσδιορίσουμε τιμή για το x, και
4. όποτε θέλουμε να υπολογίσουμε την τιμή της γράφουμε π.χ. $f(3.5)$ και το χειριζόμαστε ως πραγματικό αριθμό.

Συνάρτηση στον προγραμματισμό

Σε αντιστοιχία με τη μαθηματική συνάρτηση, ένα σύνολο εντολών σε κάποιο πρόγραμμα μπορεί

1. να αποκτήσει ένα όνομα,
2. να εξαρτάται από καμία, μία ή περισσότερες παραμέτρους για τις οποίες γνωρίζουμε μόνο τον τύπο και το όνομα αλλά όχι συγκεκριμένη τιμή,
3. μπορεί να εκτελεστεί μόνο αφού προσδιορίσουμε τιμές για τις παραμέτρους,
4. όποτε θέλουμε να εκτελεστούν αρκεί να γράψουμε το όνομα που τις συμβολίζει μαζί με κατάλληλες τιμές για τις παραμέτρους (τα ορίσματα). Το όνομα τότε έχει συγκεκριμένο τύπο και αποκτά συγκεκριμένη τιμή.

Παράδειγμα (1/4)

Υπολογισμός του

$$e^x \approx \sum_{n=0}^{10} \frac{x^n}{n!} = \frac{1}{0!} + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots + \frac{x^{10}}{10!}$$

```
INTEGER :: n
DOUBLE PRECISION :: x,s

READ *, x
s = 0.0d0
DO n = 0, 10



   s = s + x**n / ...
END DO
```

Παράδειγμα (1/4)

Υπολογισμός του

$$e^x \approx \sum_{n=0}^{10} \frac{x^n}{n!} = \frac{1}{0!} + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots + \frac{x^{10}}{10!}$$

```
INTEGER :: n,k,p
DOUBLE PRECISION :: x,s

READ *, x
s = 0.0d0
DO n = 0, 10
   p = 1
   DO k = 1, n
      p = p * k
   END DO
   s = s + x**n / p
END DO
```

Παράδειγμα (2/4)

Συνάρτηση παραγοντικού: Ορισμός

Έξω από το πρόγραμμα (πριν το **PROGRAM** ... ή μετά το **END PROGRAM** ...) ή άλλο υποπρόγραμμα, γράφουμε:

```
FUNCTION par(n)
   IMPLICIT NONE
   INTEGER, INTENT (in) :: n
   INTEGER :: par
   INTEGER :: k,p

   p = 1
   DO k = 1, n
      p = p * k
   END DO
   par = p
END FUNCTION par
```

Παράδειγμα (3/4)

Συνάρτηση παραγοντικού: Δήλωση

Στο τμήμα των δηλώσεων, μέσα στο πρόγραμμα ή στο υποπρόγραμμα στο οποίο θα χρησιμοποιήσουμε τη συνάρτηση, γράφουμε:

```
INTERFACE
   FUNCTION par(n)
      IMPLICIT NONE
      INTEGER, INTENT (in) :: n
      INTEGER :: par
   END FUNCTION par
END INTERFACE
```

Από το **INTERFACE** ο compiler πληροφορείται το όνομα, το πλήθος και τον τύπο των παραμέτρων, και τον τύπο της επιστρεφόμενης τιμής.

Παράδειγμα (4/4)

Συνάρτηση παραγοντικού: Κλήση

Η κλήση της συνάρτησης στο πρόγραμμα/υποπρόγραμμα στο οποίο δηλώθηκε, γίνεται γράφοντας το όνομα και, σε παρενθέσεις, τα ορίσματα (εδώ, μία ακέραια τιμή):

```
INTEGER :: n
DOUBLE PRECISION :: x,s

READ *, x
s = 0.0d0
DO n = 0, 10
   s = s + x**n / par(n)
END DO
```

Γενική μορφή ορισμού συνάρτησης

```
FUNCTION όνομα(παράμετροςΑ, παράμετροςΒ,...)
    IMPLICIT NONE
    τύπος_παραμέτρου_Α, INTENT (xxx) :: παράμετροςΑ
    τύπος_παραμέτρου_Β, INTENT (yyy) :: παράμετροςΒ
    τύπος_επιστρεφόμενης_ποσότητας :: όνομα

    τύπος_Α :: τοπική_μεταβλητή_Α, ...
    τύπος_Β :: τοπική_μεταβλητή_Β, ...

     ...! κώδικας
    όνομα = ...
END FUNCTION όνομα
```

Γενική μορφή ορισμού υπορουτίνας

```
SUBROUTINE όνομα(παράμετροςΑ, παράμετροςΒ,...)
    IMPLICIT NONE
    τύπος_παραμέτρου_Α, INTENT (xxx) :: παράμετροςΑ
    τύπος_παραμέτρου_Β, INTENT (yyy) :: παράμετροςΒ

    τύπος_Α :: τοπική_μεταβλητή_Α, ...
    τύπος_Β :: τοπική_μεταβλητή_Β, ...

     ...! κώδικας
END SUBROUTINE όνομα
```

Παρατηρήσεις στον ορισμό υποπρογράμματος

- Κάθε υποπρόγραμμα γράφεται έξω από οποιοδήποτε πρόγραμμα ή άλλο υποπρόγραμμα.

Παρατηρήσεις στον ορισμό υποπρογράμματος

- Κάθε υποπρόγραμμα γράφεται έξω από οποιοδήποτε πρόγραμμα ή άλλο υποπρόγραμμα.
- Οι ποσότητες που ορίζονται στο εσωτερικό ενός υποπρογράμματος είναι τοπικές και δεν μπορούν να χρησιμοποιηθούν από το πρόγραμμα ή άλλο υποπρόγραμμα.

Παρατηρήσεις στον ορισμό υποπρογράμματος

- Κάθε υποπρόγραμμα γράφεται έξω από οποιοδήποτε πρόγραμμα ή άλλο υποπρόγραμμα.
- Οι ποσότητες που ορίζονται στο εσωτερικό ενός υποπρογράμματος είναι τοπικές και δεν μπορούν να χρησιμοποιηθούν από το πρόγραμμα ή άλλο υποπρόγραμμα.
- Το όνομα της συνάρτησης, στο σώμα της συνάρτησης, συμπεριφέρεται ως μεταβλητή, πρέπει να δηλωθεί και πρέπει να αποκτήσει τιμή πριν ολοκληρωθεί η συνάρτηση.

Παρατηρήσεις στον ορισμό υποπρογράμματος

- Κάθε υποπρόγραμμα γράφεται έξω από οποιοδήποτε πρόγραμμα ή άλλο υποπρόγραμμα.
- Οι ποσότητες που ορίζονται στο εσωτερικό ενός υποπρογράμματος είναι τοπικές και δεν μπορούν να χρησιμοποιηθούν από το πρόγραμμα ή άλλο υποπρόγραμμα.
- Το όνομα της συνάρτησης, στο σώμα της συνάρτησης, συμπεριφέρεται ως μεταβλητή, πρέπει να δηλωθεί και πρέπει να αποκτήσει τιμή πριν ολοκληρωθεί η συνάρτηση.
- Το όνομα της υπορουτίνας δεν είναι μεταβλητή, δεν δηλώνεται και δεν μπορεί να πάρει τιμή.

Παρατηρήσεις στον ορισμό υποπρογράμματος

- Κάθε υποπρόγραμμα γράφεται έξω από οποιοδήποτε πρόγραμμα ή άλλο υποπρόγραμμα.
- Οι ποσότητες που ορίζονται στο εσωτερικό ενός υποπρογράμματος είναι τοπικές και δεν μπορούν να χρησιμοποιηθούν από το πρόγραμμα ή άλλο υποπρόγραμμα.
- Το όνομα της συνάρτησης, στο σώμα της συνάρτησης, συμπεριφέρεται ως μεταβλητή, πρέπει να δηλωθεί και πρέπει να αποκτήσει τιμή πριν ολοκληρωθεί η συνάρτηση.
- Το όνομα της υπορουτίνας δεν είναι μεταβλητή, δεν δηλώνεται και δεν μπορεί να πάρει τιμή.
- Ένα υποπρόγραμμα ολοκληρώνεται όταν τελειώσει ή όταν εκτελεστεί κάποια εντολή **RETURN** στο σώμα του.

Δήλωση της πρόθεσης (`INTENT`)

Στις δηλώσεις των παραμέτρων ενός υποπρογράμματος καλό είναι να ενημερώνουμε το μεταγλωττιστή για το πώς πρόκειται να χρησιμοποιήσουμε κάθε όρισμα. Αν η δήλωση συμπληρωθεί με

INTENT (in) η τιμή του ορίσματος δεν πρόκειται να αλλάξει στο υποπρόγραμμα, μόνο θα χρησιμοποιηθεί.

Δήλωση της πρόθεσης (`INTENT`)

Στις δηλώσεις των παραμέτρων ενός υποπρογράμματος καλό είναι να ενημερώνουμε το μεταγλωττιστή για το πώς πρόκειται να χρησιμοποιήσουμε κάθε όρισμα. Αν η δήλωση συμπληρωθεί με

INTENT (in) η τιμή του ορίσματος δεν πρόκειται να αλλάξει στο υποπρόγραμμα, μόνο θα χρησιμοποιηθεί.

INTENT (out) η αρχική τιμή του ορίσματος δεν πρόκειται να χρησιμοποιηθεί στο υποπρόγραμμα. Το όρισμα θα αποκτήσει τιμή κατά την εκτέλεση του υποπρογράμματος.

Δήλωση της πρόθεσης (`INTENT`)

Στις δηλώσεις των παραμέτρων ενός υποπρογράμματος καλό είναι να ενημερώνουμε το μεταγλωττιστή για το πώς πρόκειται να χρησιμοποιήσουμε κάθε όρισμα. Αν η δήλωση συμπληρωθεί με

INTENT (in) η τιμή του ορίσματος δεν πρόκειται να αλλάξει στο υποπρόγραμμα, μόνο θα χρησιμοποιηθεί.

INTENT (out) η αρχική τιμή του ορίσματος δεν πρόκειται να χρησιμοποιηθεί στο υποπρόγραμμα. Το όρισμα θα αποκτήσει τιμή κατά την εκτέλεση του υποπρογράμματος.

INTENT (inout) η αρχική τιμή του ορίσματος θα χρησιμοποιηθεί στο υποπρόγραμμα και το όρισμα θα πάρει τιμή κατά την εκτέλεση.

Δήλωση της πρόθεσης (`INTENT`)

Στις δηλώσεις των παραμέτρων ενός υποπρογράμματος καλό είναι να ενημερώνουμε το μεταγλωττιστή για το πώς πρόκειται να χρησιμοποιήσουμε κάθε όρισμα. Αν η δήλωση συμπληρωθεί με

INTENT (in) η τιμή του ορίσματος δεν πρόκειται να αλλάξει στο υποπρόγραμμα, μόνο θα χρησιμοποιηθεί.

INTENT (out) η αρχική τιμή του ορίσματος δεν πρόκειται να χρησιμοποιηθεί στο υποπρόγραμμα. Το όρισμα θα αποκτήσει τιμή κατά την εκτέλεση του υποπρογράμματος.

INTENT (inout) η αρχική τιμή του ορίσματος θα χρησιμοποιηθεί στο υποπρόγραμμα και το όρισμα θα πάρει τιμή κατά την εκτέλεση.

Στις δύο τελευταίες περιπτώσεις, προφανώς, το όρισμα πρέπει να είναι μεταβλητή και όχι σταθερή τιμή.

Γενική μορφή δήλωσης του υποπρογράμματος

Πριν χρησιμοποιηθεί ένα υποπρόγραμμα πρέπει να δηλωθεί. Στο πρόγραμμα ή άλλο υποπρόγραμμα στο οποίο θα καλέσουμε το υποπρόγραμμά μας γράφουμε, στην περιοχή των δηλώσεων, εντός **INTERFACE/END INTERFACE**, όλο το υποπρόγραμμα χωρίς τις εντολές και τις δηλώσεις τοπικών ποσοτήτων.

Γενική μορφή δήλωσης του υποπρογράμματος

Πριν χρησιμοποιηθεί ένα υποπρόγραμμα πρέπει να δηλωθεί. Στο πρόγραμμα ή άλλο υποπρόγραμμα στο οποίο θα καλέσουμε το υποπρόγραμμά μας γράφουμε, στην περιοχή των δηλώσεων, εντός **INTERFACE/END INTERFACE**, όλο το υποπρόγραμμα χωρίς τις εντολές και τις δηλώσεις τοπικών ποσοτήτων.
Ειδικότερα, πρέπει να δηλώσουμε

- το όνομα του υποπρογράμματος,
- το πλήθος, τη σειρά και τους τύπους των παραμέτρων,
- στην περίπτωση συνάρτησης, τον τύπο του ονόματος.

Γενική μορφή δήλωσης συνάρτησης

```
INTERFACE
   FUNCTION όνομα(παράμετροςA, παράμετροςB,...)
     IMPLICIT NONE
     τύπος_παραμέτρου_A, INTENT (xxx) :: παράμετροςA
     τύπος_παραμέτρου_B, INTENT (yyy) :: παράμετροςB
     τύπος_επιστρεφόμενης_ποσότητας :: όνομα
   END FUNCTION όνομα
END INTERFACE
```

Γενική μορφή δήλωσης υπορουτίνας

```
INTERFACE
   SUBROUTINE όνομα(παράμετροςA, παράμετροςB,...)
     IMPLICIT NONE
     τύπος_παραμέτρου_A, INTENT (xxx) :: παράμετροςA
     τύπος_παραμέτρου_B, INTENT (yyy) :: παράμετροςB
   END SUBROUTINE όνομα
END INTERFACE
```

- Ένα **INTERFACE/END INTERFACE** μπορεί να περιλαμβάνει δηλώσεις περισσότερων του ενός υποπρογραμμάτων.

Γενική μορφή δήλωσης του υποπρογράμματος

Παρατηρήσεις

- Ένα **INTERFACE/END INTERFACE** μπορεί να περιλαμβάνει δηλώσεις περισσότερων του ενός υποπρογραμμάτων.
- Οι δηλώσεις των παραμέτρων του υποπρογράμματος σε ένα **INTERFACE** δεν είναι δηλώσεις μεταβλητών του προγράμματος ή υποπρογράμματος που περιέχει το **INTERFACE**.

Κλήση συνάρτησης

- Στο σημείο του κώδικα που επιθυμούμε να εκτελεστούν οι εντολές της συνάρτησης γράφουμε το όνομα και, εντός παρενθέσεων, παραθέτουμε ποσότητες με ίδια σειρά, ίδιο πλήθος και ίδιο τύπο με τις παραμέτρους. Αυτές λέγονται ορίσματα.

Κλήση συνάρτησης

- Στο σημείο του κώδικα που επιθυμούμε να εκτελεστούν οι εντολές της συνάρτησης γράφουμε το όνομα και, εντός παρενθέσεων, παραθέτουμε ποσότητες με ίδια σειρά, ίδιο πλήθος και ίδιο τύπο με τις παραμέτρους. Αυτές λέγονται ορίσματα.
- Η παραπάνω έκφραση έχει τιμή που πρέπει να χρησιμοποιηθεί (να αποθηκευτεί σε μεταβλητή, να τυπωθεί, να συμμετάσχει σε έκφραση, κλπ.).

Κλήση συνάρτησης

- Στο σημείο του κώδικα που επιθυμούμε να εκτελεστούν οι εντολές της συνάρτησης γράφουμε το όνομα και, εντός παρενθέσεων, παραθέτουμε ποσότητες με ίδια σειρά, ίδιο πλήθος και ίδιο τύπο με τις παραμέτρους. Αυτές λέγονται ορίσματα.
- Η παραπάνω έκφραση έχει τιμή που πρέπει να χρησιμοποιηθεί (να αποθηκευτεί σε μεταβλητή, να τυπωθεί, να συμμετάσχει σε έκφραση, κλπ.).
- Όταν κληθεί η συνάρτηση, εκτελούνται οι εντολές στο σώμα της. Κατόπιν, επιστρέφει η ροή εκτέλεσης εντολών στο σημείο της κλήσης. Το όνομα της συνάρτησης έχει αποκτήσει τιμή.

Κλήση υπορουτίνας

- Στο σημείο του κώδικα που επιθυμούμε να εκτελεστούν οι εντολές της υπορουτίνας γράφουμε **CALL**, το όνομα και, εντός παρενθέσεων, παραθέτουμε τιμές με ίδια σειρά, ίδιο πλήθος και ίδιο τύπο για τις παραμέτρους της υπορουτίνας.

Κλήση υπορουτίνας

- Στο σημείο του κώδικα που επιθυμούμε να εκτελεστούν οι εντολές της υπορουτίνας γράφουμε **CALL**, το όνομα και, εντός παρενθέσεων, παραθέτουμε τιμές με ίδια σειρά, ίδιο πλήθος και ίδιο τύπο για τις παραμέτρους της υπορουτίνας.
- Όταν κληθεί η υπορουτίνα εκτελούνται οι εντολές στο σώμα της. Κατόπιν, επιστρέφει η ροή εκτέλεσης εντολών στο σημείο της κλήσης.

Άλλα παραδείγματα (1/2)

Έστω ότι χρειαζόμαστε να επαναλάβουμε σε κώδικα τον υπολογισμό $x^2 + 5x + 3$ για διάφορα πραγματικά x.

Άλλα παραδείγματα (1/2)

Έστω ότι χρειαζόμαστε να επαναλάβουμε σε κώδικα τον υπολογισμό $x^2 + 5x + 3$ για διάφορα πραγματικά x.

Ορισμός

```
FUNCTION g(x)
  IMPLICIT NONE
  DOUBLE PRECISION, INTENT (in)::x
  DOUBLE PRECISION :: g

  g = x**2 + 5d0*x + 3d0
END FUNCTION g
```

Άλλα παραδείγματα (1/2)

Έστω ότι χρειαζόμαστε να επαναλάβουμε σε κώδικα τον υπολογισμό $x^2 + 5x + 3$ για διάφορα πραγματικά x.

Ορισμός

```
FUNCTION g(x)
  IMPLICIT NONE
  DOUBLE PRECISION, INTENT (in)::x
  DOUBLE PRECISION :: g

  g = x**2 + 5d0*x + 3d0
END FUNCTION g
```

Δήλωση

```
INTERFACE
  FUNCTION g(x)
    IMPLICIT NONE
    DOUBLE PRECISION, INTENT (in)::x
    DOUBLE PRECISION :: g
  END FUNCTION g
END INTERFACE
```

Άλλα παραδείγματα (1/2)

Έστω ότι χρειαζόμαστε να επαναλάβουμε σε κώδικα τον υπολογισμό $x^2 + 5x + 3$ για διάφορα πραγματικά x.

Ορισμός

```
FUNCTION g(x)
  IMPLICIT NONE
  DOUBLE PRECISION, INTENT (in)::x
  DOUBLE PRECISION :: g

  g = x**2 + 5d0*x + 3d0
END FUNCTION g
```

Δήλωση

```
INTERFACE
  FUNCTION g(x)
    IMPLICIT NONE
    DOUBLE PRECISION, INTENT (in)::x
    DOUBLE PRECISION :: g
  END FUNCTION g
END INTERFACE
```

Κλήση

```
DOUBLE PRECISION :: y
y = g(2.45d0)
```

Άλλα παραδείγματα (2/2)

Ανάγνωση δύο πραγματικών τιμών, εξασφαλίζοντας ότι είναι θετικές. Αν δεν είναι, επαναλαμβάνουμε την ανάγνωση.

Άλλα παραδείγματα (2/2)

Ανάγνωση δύο πραγματικών τιμών, εξασφαλίζοντας ότι είναι θετικές. Αν δεν είναι, επαναλαμβάνουμε την ανάγνωση.

Ορισμός

```
SUBROUTINE readpositive(x,y)
  IMPLICIT NONE
  DOUBLE PRECISION, INTENT(out):: x
  DOUBLE PRECISION, INTENT(out):: y
  DO
    PRINT *,"Δώσε δύο θετικούς πραγματικούς"
    READ *, x, y
    IF (x > 0.0d0 .AND. y > 0.0d0) EXIT
  END DO
END SUBROUTINE readpositive
```

Άλλα παραδείγματα (2/2)

Ανάγνωση δύο πραγματικών τιμών, εξασφαλίζοντας ότι είναι θετικές. Αν δεν είναι, επαναλαμβάνουμε την ανάγνωση.

Ορισμός

```
SUBROUTINE readpositive(x,y)
  IMPLICIT NONE
  DOUBLE PRECISION, INTENT(out):: x
  DOUBLE PRECISION, INTENT(out):: y
  DO
    PRINT *,"Δώσε δύο θετικούς πραγματικούς"
    READ *, x, y
    IF (x > 0.0d0 .AND. y > 0.0d0) EXIT
  END DO
END SUBROUTINE readpositive
```

Δήλωση

```
INTERFACE
  SUBROUTINE readpositive(x,y)
    IMPLICIT NONE
    DOUBLE PRECISION, INTENT(out):: x
    DOUBLE PRECISION, INTENT(out):: y
  END SUBROUTINE readpositive
END INTERFACE
```

Άλλα παραδείγματα (2/2)

Ανάγνωση δύο πραγματικών τιμών, εξασφαλίζοντας ότι είναι θετικές. Αν δεν είναι, επαναλαμβάνουμε την ανάγνωση.

Ορισμός

```
SUBROUTINE readpositive(x,y)
  IMPLICIT NONE
  DOUBLE PRECISION, INTENT(out):: x
  DOUBLE PRECISION, INTENT(out):: y
  DO
    PRINT *,"Δώσε δύο θετικούς πραγματικούς"
    READ *, x, y
    IF (x > 0.0d0 .AND. y > 0.0d0) EXIT
  END DO
END SUBROUTINE readpositive
```

Δήλωση

```
INTERFACE
  SUBROUTINE readpositive(x,y)
    IMPLICIT NONE
    DOUBLE PRECISION, INTENT(out):: x
    DOUBLE PRECISION, INTENT(out):: y
  END SUBROUTINE readpositive
END INTERFACE
```

Κλήση

```
DOUBLE PRECISION :: a,b
CALL readpositive(a,b)
```

Πότε χρησιμοποιούμε **function** και πότε **subroutine**;

Χρησιμοποιούμε **function** όταν

έχουμε να "επιστρέψουμε" μία τιμή στο υπόλοιπο πρόγραμμα, που θα χρησιμοποιηθεί σε επόμενες εντολές μετά την κλήση.

Χρησιμοποιούμε **subroutine** όταν

- δεν έχουμε να "επιστρέψουμε" τιμή στο υπόλοιπο πρόγραμμα (αλλά θέλουμε π.χ. να γράψουμε σε αρχείο ή στην οθόνη), ή
- θέλουμε να επιστρέψουμε περισσότερες από μία τιμές στο υπόλοιπο πρόγραμμα μέσω ορισμάτων της υπορουτίνας. Τα ορίσματα αυτά προφανώς πρέπει να έχουν δηλωθεί με **INTENT out** ή **inout** (ώστε να μπορούν να αλλάξουν τιμή).