Υπο-προγράμματα στη Fortran

- Μέχρι τώρα τα προβλήματα και τα προγράμματα που έχουμε δεί ήταν αρκετά απλά και επομένως ένα και μόνο πρόγραμμα ήταν αρκετό για να τα λύσουμε
- Όταν τα προγράμματα γίνονται μεγάλα είναι πολλές φορές δύσκολο να αποφασίσουμε από που θα ξεκινήσουμε τους υπολογισμούς.
- Ακόμα όταν τα προγράμματα γίνονται μεγάλα είναι αρκετά δύσκολο να τα έχουμε σε ένα ενιαίο μοναδικό πρόγραμμα.

  Και αυτό γιατί είναι δύσκολο να παρακολουθήσουμε πολλές γραμμές κώδικα
- Επομένως σε στύλ «από πάνω προς τα κάτω» (top-down) προσέγγισης ενός πολύπλοκου προβλήματος είναι να χρησιμοποιήσουμε μια στατηγική διαίρεσης και επίλυσης επιμέρους τμημάτων του προβλήματος

Υπο-προγράμματα στη Fortran

- Τα πλεονεκτήματα χρήσης μικρών υποπρογραμμάτων στην επίλυση ενός μεγαλύτερου προβλήματος είναι πολλά:
  - Τεμαχίζει το πρόγραμμα σε μικρές λογικές μονάδες οι οποίες είναι πολύ πιο εύκολο να λυθούν και να παρακολουθήσουμε τη λειτουργία τους
  - Κάνει πιό ευανάγνωστα και ευκολονόητα τα μεγάλα προγράμματα
  - Επιτρέπει την ανάπτυξη προγραμμάτων σε επίπεδο τμήματος του προγράμματος σε αντίθεση με την προσπάθεια ανάπτυξης ολόκληρου του προγράμματος σα ενιαία οντότητα
  - Είναι πολύ πιο εύκολο να βρεθούν λάθη τόσο λογικής αλλά και υπολογισμού μια και μπορούν να απομονωθούν σε τμήματα του προγράμματος
  - Τα υποπρογράμματα αυτά γράφονται μια φορά αλλά μπορούν να κληθούν πολλαπλές φορές μέσα από ένα πρόγραμμα ή άλλα προγράμματα.

    Δημιουργούμε έτσι μια βιβλιοθήκη υποπρογραμμάτων
  - Πολύ σημαντικό το γεγονός ότι αποφεύγεται η γραφή του ίδιου κώδικα πολλαπλές φορές.

Συναρτήσεις(Functions) & Υπορουτίνες (Subroutines)

- Δύο είδη υπογραμμάτων στη FORTRAN:
  - Οι συναρτήσεις (**functions**)
  - Οι υπορουτίνες (**subroutines**)

- Για να χρησιμοποιήσουμε Functions και Subroutines χρειάζεται να γράψουμε ένα κύριο πρόγραμμα στο οποίο υπάρχουν οι αναφορές σε όλα τα καλούμενα υποπρογράμματα σύμφωνα με την επιθυμητή σειρά κλίσης και κατόπιν προχωρούμε στη γραφή των επιμέρους υποπρογραμμάτων.

- Η γραφή αυτή είναι παρόμοια με το να γράψουμε το σκελετό μιας έκθεσης και κατόπιν αναπτύσσουμε το θέμα χρησιμοποιώντας τα θέματα του σκελετού σαν οδηγό.

Συναρτήσεις – Functions

- Ο λόγος για τον οποίο χρησιμοποιούμε τις συναρτήσεις είναι να δώσουμε μια σειρά από μεταβλητές (arguments) ή σταθερές, να εκτελέσουμε ένα υπολογισμό με αυτά που δώσαμε σαν arguments και να μας επιστραφεί το αποτέλεσμα και μόνο του υπολογισμού.
- Υπάρχει μια σειρά από συναρτήσεις που είναι γραμμένες ήδη στη βιβλιοθήκη της Fortran (library functions ή intrinsic functions) και μας βοηθούν σε διάφορους υπολογισμούς. Για παράδειγμα η συνάρτηση SQRT(x) υπολογίζει τη τετραγωνική ρίζα της μεταβλητής x. Κάθε φορά που καλούμε τη συνάρτηση SQRT με κάποιο argument παίρνουμε πίσω μια και μοναδική τιμή που δεν είναι άλλη από την τετραγωνική ρίζα του argument (μεταβλητής ή σταθεράς).
- Το πλεονέκτημα των συναρτήσεων είναι ότι εκτελούν κάθε φορά τον ίδιο υπολογισμό χωρίς να χρειάζεται κάθε φορά να επαναλαμβάνουμε τον ίδιο κώδικα. Απλά χρειάζεται να γράψουμε το κώδικα του υποπρογράμματος μια φορά και κατόπιν στο κύριο μέρος του προγράμματος αναφερόμαστε στη συνάρτηση.

Functions

- Ο γενικός τρόπος με τον οποίο μπορούμε να ενεργοποιήσουμε μια συνάρτηση είναι να χρησιμοποιήσουμε το όνομα της συνάρτησης σε μαθηματική έκφραση.

- Το όνομα της συνάρτησης ακολουθείται από μια σειρά από μεταβλητές εισόδου (input arguments) που περικλείονται σε παρενθέσεις.

solution = function_name(arg1, arg2, arg3,….)

- Υποπρόγραμμα συνάρτησης

<type> FUNCTION function_name(arg1, arg2, arg3,…)

δηλώσεις μεταβλητών

εντολές του υποπρογράμματος

RETURN

END

Παραδείγματα χρήσης συναρτήσεων

PRINT *, SQRT(X) Ο compiler θα τυπώσει τη τετραγωνική ρίζα του X

→ Το αποτέλεσμα της κλίσης συναρτήσεως πρέπει να χρησιμοποιείται σε εντολή ανάθεσης (assignment statement) ή σε εντολή PRINT

Y = SIN(X)+45 Ο compiler θα υπολογίσει το ημίτονο x και κατόπιν θα προσθέσει τη σταθερά 45. Το τελικό αποτέλεσμα θα. δοθεί στη μεταβλητή Y

M = Max(A,B,C) Ο compiler υπολογίζει το μέγιστο των A,B,C και το δίνει στη μεταβλητή M

→ Όπως φαίνεται από το παράδειγμα με τη συνάρτηση MAX, μια συνάρτηση μπορεί να έχει περισσότερα από ένα arguments σαν input αλλά επιστρέφει πάντα ένα και μόνο αποτέλεσμα.

C = SQRT(A**2+B**2) Ο compiler υπολογίζει πρώτα το άθροισμα τετραγώνων των A και B κατόπιν καλεί τη συνάρτηση της τετραγωνικής ρίζας και το αποτέλεσμα μεταφέρεται στη μεταβλητή C

→ Όπως φαίνεται από το παράδειγμα της SQRT το input argument μπορεί να είναι μια αριθμητική έκφραση, μια σταθερά ή μια μεταβλητή. Δεν είναι απαραίτητο δηλαδή να είναι μεταβλητή και μόνο.

Functions

- Το αποτέλεσμα της συνάρτησης πρέπει να δίνεται σε μια μεταβλητή όπως συμβαίνει και με τις μαθηματικές συναρτήσεις

Εξωτερικές Συναρτήσεις – External Functions

Οι συναρτήσεις βιβλιοθήκης είναι πολύ χρήσιμες αλλά κάποια στιγμή δεν θα υπάρχει η συνάρτηση που θα εκτελέσει τον ακριβή υπολογισμό που θέλετε

- Όταν συμβεί αυτό θα χρειαστεί να γράψετε τη δική σας function

- Πριν χρησιμοποιήσετε μια συνάρτησή σας στο κύριο πρόγραμμα πρέπει να τη δηλώσετε, όπως ακριβώς κάνετε και για όλες τις μεταβλητές.

  Πρέπει να δηλώσετε το είδος της συνάρτησης ανάλογα με το είδος του αποτελέσματος που επιστρέφει: INTEGER REAL DOUBLE PRECISION

- Τα ονόματα των συναρτήσεων ακολουθούν ακριβώς τους ίδιους κανόνες που διέπουν τα ονόματα των μεταβλητών. Δηλαδή λιγότεροι από 32 χαρακτήρες (γράμματα ή νούμερα) από τους οποίους ο πρώτος είναι πάντοτε γράμμα

  - Εξαιτίας αυτής της ομοιότητας τα ονόματα των συναρτήσεων δεν πρέπει να χρησιμοποιούνται σα μεταβλητές

- Από τη στιγμή που έχετε γράψει τη συνάρτηση και την έχετε δηλώσει μπορείτε να τη χρησιμοποιήσετε ακριβώς όπως και τις συναρτήσεις βιβλιοθήκης

Κανόνες γραφής των External Functions

- Τα υποπρογράμματα τύπου Function (όπως και όλα τα υποπρογράμματα) γράφονται μετά την εντολή END του κυρίου προγράμματος.
- Ξεκινούν με μια γραμμή η οποία δηλώνει το είδος του αποτελέσματος που θα επιστρέψει η συνάρτηση, το όνομα της συνάρτησης και τη λίστα των arguments (ορίσματα) που η συνάρτηση δέχεται σαν input.
- Οποιαδήποτε arguments (συμπεριλαμβανομένων και των input arguments - ορίσματα) χρησιμοποιούνται από τη συνάρτηση θα πρέπει να δηλωθούν στην αρχή πριν από οποιαδήποτε εντολή εκτέλεσης όπως και στα κανονικά προγράμματα.
  - Το όνομα της συνάρτησης δεν δηλώνεται μέσα στο κώδικα της συνάρτησης (έχει ήδη δηλωθεί στην πρώτη γραμμή).

☠ Η συνάρτηση θα πρέπει να πάρει οπωσδήποτε κάποια τιμή μέσα στο υποπρόγραμμά που την προσδιορίζει .

- Πρέπει να χρησιμοποιήσετε το όνομα της συνάρτησης σε μια εντολή ανάθεσης μέσα στο υποπρόγραμμα δίνοντας έτσι τιμή στη συνάρτηση.
- Αυτός είναι ο μόνος τρόπος για να ξέρει ο compiler τι τιμή να επιστρέψει στο κύριο πρόγραμμα που καλεί τη συνάρτηση.

- Ένα υποπρόγραμμα τύπου Function πρέπει να τελειώνει πάντοτε με τις ακόλουθες δύο εντολές:

```
RETURN
END
```

Παράδειγμα γραφής συναρτήσεως

```
      PROGRAM DEMO
C…Δηλώσεις μεταβλητών για main program
      REAL A, B, C
      REAL AV, AVSQ1, AVSQ2
      REAL AVRAGE
C…Είσοδος δεδομένων
      DATA  A, B, C / 5.0, 2.0, 3.0/
C…Υπολογισμός μέσου όρου αριθμών
      AV = AVRAGE(A,B,C)
      AVSQ1 = AVRAGE(A,B,C) **2
      AVSQ2 = AVRAGE(A**2,B**2,C**2)
      PRINT *,'Statistical Analysis '
      PRINT *,'The average of the numbers is:',AV
      PRINT *,'The average squared of the numbers: ',AVSQl
      PRINT *,'The average of the squares is: ', AVSQ2
      END

      REAL FUNCTION AVRAGE(X,Y,Z)
      REAL X, Y, Z, SUM
      SUM = X + Y + Z
      AVRAGE = SUM /3.0
      RETURN
      END
```

Δήλωση της συνάρτησης

Χρησιμοποίηση της συνάρτησης μέσω assignment statement

Παρατηρήστε ότι τα ονόματα των μεταβλητών εισόδου δεν είναι απαραίτητα ίδια. Πρέπει όμως να 'ναι του ίδιου τύπου και ίδιου πλήθους.

Αρχή του υποπρογράμματος της Function AVRAGE

Η μεταβλητή SUM υπάρχει μόνο στο κώδικα της function Δεν χρειάζεται να δηλωθεί στο MAIN πρόγραμμα

Ανάθεση της τιμής της συνάρτησης

Παράδειγμα συνάρτησης

- Ο υπολογισμός του παραγοντικού ενός αριθμού N

```
      FUNCTION FACTRL (N)
C… This function computes n!, i.e. n*(n-1)*(n-2)*...*1
      INTEGER FACTRL, N, I
      FACTRL = 1
      IF ( N .GT. 1 ) THEN
         DO 10 I = 2, N
            FACTRL = FACTRL * I
10       CONTINUE
      END IF
      RETURN
      END
```

Η συνάρτηση αυτή μπορεί να καλείται από ένα πρόγραμμα με τη μορφή

```
PROGRAM FACTS
INTEGER   J, N, FACTRL
PRINT *,' DWSTE TO  EYROS TWN ARITHMWN'
READ *, N
DO  J = 1, N
    TERMA = FACTRL(J)   ! Υπολογισμός του παραγοντικού κάθε INTEGER  J από 1 έως N
     PRINT *,' To paragontiko tou ', j, ' einai ', TERMA
ENDDO
END
```

Υπορουτίνες - Subroutines

- Χρησιμοποιούμε μια συνάρτηση αν θέλουμε να κάνουμε ένα πολύπλοκο υπολογισμό ο οποίος θα επέστρεφε ένα και μόνο αποτέλεσμα το οποίο και θα μπορούσαμε να χρησιμοποιήσουμε ή όχι αργότερα σε μια μαθηματική έκφραση. Όπως στο προηγούμενο παράδειγμα της συνάρτησης όπου καλέσαμε τη Function AVRAGE και κατόπιν την υψώσαμε στο τετράγωνο με μια και μόνο εντολή.
- Οι υπορουτίνες (**subroutines**) χρησιμοποιούνται όταν θέλουμε να κάνουμε μια σειρά υπολογισμών και να επιστρέψουμε στο κύριο πρόγραμμα περισσότερα από ένα αποτελέσματα. **Κλήσεις υπορουτινών δεν μπορούν να τεθούν σε μαθηματική έκφραση.**
- Στο κύριο πρόγραμμα, μια υπορουτίνα καλείται χρησιμοποιώντας την εντολή CALL ακολουθούμενη από το όνομα της υπορουτίνας και μια λίστα παραμέτρων περικλειόμενες σε παρένθεση.

  Οι παράμετροι αυτοί μπορεί να είναι **παράμετροι εισόδου** (δηλαδή δίνουν κάποιες τιμές στο υποπρόγραμμα της subroutine) και **παράμετροι εξόδου**, δηλαδή παράμετροι που περιέχουν τα αποτελέσματα διαφόρων υπολογισμών που περιέχονται στην υπορουτίνα.

Όλες αυτές οι παράμετροι ονομάζονται arguments της subroutine.

Subroutines – Κανόνες γραφής

- Τα ονόματα των subroutine ακολουθούν τους ίδιους κανόνες με τα ονόματα των functions και των υπόλοιπων μεταβλητών. Μπορεί να περιέχουν μέχρι 32 χαρακτήρες (αριθμητικούς ή όχι) αλλά ο πρώτος χαρακτήρας δεν μπορεί να είναι αριθμητικός.

  Τα ονόματα των subroutine δεν πρέπει να είναι ίδια με τα ονόματα μεταβλητών ή functions.

- Όπως και με τις functions υπάρχουν ορισμένοι κανόνες όταν γράφουμε subroutines. Οι κανόνες είναι οι ακόλουθοι:
  - Σε αντίθεση με τις functions, οι subroutines δεν χρειάζεται να δηλωθούν στο κύριο πρόγραμμα.
  - Η πρώτη γραμμή του κώδικα μιας subroutine ξεκινά με τη γραμμή: SUBROUTINE, το όνομα της subroutine, ακολουθούμενο από τη λίστα όλων των παραμέτρων (**arguments**) της subroutine.
  - Σε περίπτωση που το μήκος των arguments ξεπερνά τις 72 στήλες μπορούμε να συνεχίσουμε στην επόμενη γραμμή με την προϋπόθεση ότι υπάρχει η κατάλληλη ένδειξη συνέχειας.
  - Δεν είναι ωστόσο απαραίτητο να υπάρχουν παράμετροι !!!

Subroutines – Κανόνες γραφής

- Ένας τρόπος για να ξεχωρίσουμε ποια από τα arguments είναι εισόδου και ποια εξόδου είναι να θέσουμε τις παραμέτρους εισόδου στη πρώτη γραμμή και τις παραμέτρους εξόδου στην επόμενη γραμμή.
- Όλες οι παράμετροι και μεταβλητές που χρησιμοποιούνται σε μια subroutine πρέπει να δηλωθούν στο τμήμα του κώδικα της subroutine.
  - Το όνομα της subroutine δεν δηλώνεται πουθενά στο πρόγραμμα
- Όλες οι παράμετροι της subroutine πρέπει να δηλωθούν επίσης στο πρόγραμμα το οποίο καλεί την subroutine. Όπως και στην περίπτωση των functions, τα ονόματα των παραμέτρων μπορεί να είναι διαφορετικά στο πρόγραμμα που καλεί την subroutine και στο κώδικα της subroutine.
  - Ο αριθμός των παραμέτρων και η σειρά με την οποία δίνονται πρέπει να είναι η ίδια και στο κύριο πρόγραμμα και στο τμήμα της subroutine.
- Μια subroutine τελειώνει πάντοτε με δύο εντολές:

```
RETURN
END
```

- Σε μεγάλα προγράμματα είναι πάντοτε σωστός τρόπος προγραμματισμού να περιέχονται σχόλια μετά τις εντολές FUNCTION και SUBROUTINE που να περιγράφουν τι κάνουν τα υποπρογράμματα καθώς επίσης και τη σημασία και λειτουργία των παραμέτρων.

Παράδειγμα Subroutine

Χωρίς arguments

```
SUBROUTINE MESSAG
PRINT*,'THIS MESSAGE'
RETURN
END
```

Με input/output arguments

```
SUBROUTINE CONVER (R, TH, X, Y)
REAL R, TH, X, Y
X = R * COS(TH)
Y = R * SIN(TH)
RETURN
END
```

- Η subroutine στα αριστερά δεν έχει κανένα argument και απλά τυπώνει ένα μήνυμα όταν καλείται.

- Η subroutine στα δεξιά χρησιμοποιεί τις μεταβλητές R, TH σε πολικές συντεταγμένες και τις μετατρέπει σε καρτεσιανές. Οι μεταβλητές αυτές επιστρέφονται στο καλών την υπορουτίνα πρόγραμμα ή άλλο υποπρόγραμμα χρησιμοποιώντας τις output μεταβλητές X και Y.

Subroutines

- Τα arguments τα οποία περνούν σε subroutines μπορεί να είναι:
  - Απλές μεταβλητές (Μπορείτε να περάσετε μια μαθηματική έκφραση σαν argument. Πρώτα θα υπολογισθεί η έκφραση και το αποτέλεσμα θα περάσει σαν argument. Το αποτέλεσμα πρέπει ωστόσο να ταυτίζεται με το δηλωμένο τύπο του argument
  - Πίνακες. Περνούν χρησιμοποιώντας το όνομα του πίνακα.
    - Ο πίνακας θα πρέπει να δηλωθεί στην subroutine με μέγεθος όχι μεγαλύτερο από αυτό που έχει δηλωθεί για το πίνακα στο κύριο πρόγραμμα.
    - Ένα στοιχείο του πίνακα χρησιμοποιείται ακριβώς σα να είναι μεταβλητή, π.χ. A(5) αναφέρεται σε μια μεταβλητή.
    - Οι πίνακες μπορεί να έχουν μεταβαλλόμενο μέγεθος, δηλαδή μπορούμε να ορίσουμε το μέγεθος ενός πίνακα σε μια subroutine βασιζόμενοι στην τιμή ενός argument της subroutine π.χ.

```
:
SUBROUTINE STUFF(AX, BX, N)
REAL   AX(N), BX(N)
 :
```

Subroutine – Array μεταβλητού μεγέθους

```
      PROGRAM COMPUTE
      REAL X(100), Y(100)
C… read in some data.
C….Check to make sure N is less than or equal to 100.
      OPEN (1,file= ‘MY.DATA ‘)
      READ(1,*) N
      IF ( N  .GT.  100 )  THEN
         WRITE(6,*)  ' N too large. Make dimensions bigger and try again ‘
         STOP
      ELSE
         DO 100 I=1, N
            READ(1,*) X(I), Y(I)
100      CONTINUE
       END IF
C
C     Call  subroutine  STUFF  to do some "stuff":
      CALL STUFF(X,Y,N)
       . . .
      END

      SUBROUTINE STUFF(AX,BX,N)
      REAL  AX(N),  BX(N)
      INTEGER N
       :
      RETURN
      END
```

Παράδειγμα χρήσης SUBROUTINE

```
PROGRAM  DEMO_SUBROUTINE
REAL A,B,C,SUM,SUMSQ
CALL INPUT( A,B,C)
CALL CALC(A,B,C,SUM,SUMSQ)
CALL OUTPUT(SUM,SUMSQ)
END

SUBROUTINE INPUT(X, Y, Z)
REAL X,Y,Z
PRINT *,'ENTER THREE NUMBERS => '
READ *,X,Y,Z
RETURN
END

SUBROUTINE CALC(X,Y,Z, ASUM,SQSUM)
REAL X,Y,Z,ASUM,SQSUM
ASUM = X + Y + Z
SQSUM = ASUM **2
RETURN
END

SUBROUTINE OUTPUT(SUM,SUMSQ)
REAL SUM, SUMSQ
PRINT *,'The sum of the numbers you entered are: ',SUM
PRINT *,'And the square of the sum is:',SUMSQ
RETURN
END
```