ΣΗΜΕΙΩΣΕΙΣ

ΕΙΣΑΓΩΓΗ ΣΤΗΝ

FORTRAN 77

Ν. ΣΤΕΡΓΙΟΥΛΑΣ
ΤΜΗΜΑ ΦΥΣΙΚΗΣ
Α.Π.Θ.

ΦΕΒΡΟΥΑΡΙΟΣ 2009

ΕΓΚΑΤΑΣΤΑΣΗ

Εγκαθιστούμε τον μεταγλωττιστή από το αρχείο http://www.astro.auth.gr/~simos/na/Force208.zip

Δημιουργούμε ένα φάκελο π.χ. c:\fortran\projects στον οποίο θα αποθηκεύουμε τα προγράμματα.

Τα ονόματα των προγραμμάτων καθώς και των φακέλων στα οποία βρίσκονται αυτά καλό είναι να μη ξεπερνούν τους 8 χαρακτήρες και να μη περιέχουν άλλους χαρακτήρες εκτός από γράμματα και αριθμούς (ώστε να υπάρχει συμβατότητα με το DOS).

Στο Options-> Environment Options -> Run -> Run with επιλέγουμε **Console.**

ΔΗΜΙΟΥΡΓΙΑ ΕΝΟΣ ΠΡΟΓΡΑΜΜΑΤΟΣ

Δημιουργούμε ένα νέο πρόγραμμα μέσω της εντολής File -> New (ή ανοίγουμε ένα υπάρχον μέσω της File -> Open. Τα προγράμματα πρέπει να έχουν extension .f .

Κάθε γραμμή εντολών πρέπει να αρχίζει από την 8η στήλη και μετά (στην αρχή κάθε γραμμής με το Tab μας πάει στην 9η).

Κάθε γραμμή μπορεί να έχει μέχρι 6+66=72 χαρακτήρες. Μπορούμε να συνεχίσουμε μια εντολή στην επόμενη γραμμή, βάζοντας το σύμβολο «.» στην 6η στήλη.

Μπορούμε να εισάγουμε σχόλια, βάζοντας έναν αστερίσκο «*» στην πρώτη στήλη.

ΜΕΤΑΓΛΩΤΤΙΣΗ ΚΑΙ ΕΚΤΕΛΕΣΗ ΕΝΟΣ ΠΡΟΓΡΑΜΜΑΤΟΣ

Αφού γράψουμε το πρόγραμμα, κάνουμε τη μεταγλώττιση με Ctrl-F9 και στη συνέχεια μπορούμε να το τρέξουμε με Shift-F9. Το πρόγραμμα τρέχει σε ένα νέο παράθυρο (console).

ΠΑΡΑΔΕΙΓΜΑ ΠΡΟΓΡΑΜΜΑΤΟΣ ΣΤΗ FORTRAN 77

Το ακόλουθο πρόγραμμα (tut1.f) υπολογίζει τον μέσο όρο δύο αριθμών και εκτυπώνει το αποτέλεσμα:

```
x=4.7
y=4.9
avg=(x+y)/2.0
print *, x, y, avg
pause
end
```

Οι **x**, **y** και **avg** είναι μεταβλητές, στις οποίες δίνουμε τιμές με χρήση του **"="**. Η εντολή **print *,** χρησιμοποιείται για να εκτυπώσουμε την τιμή των μεταβλητών στην οθόνη. Η εντολή **pause** κρατά το παράθυρο εκτέλεσης ανοικτό. Η εντολή **end** δηλώνει το τέλος του προγράμματος.

ΜΕΤΑΓΛΩΤΙΣΗ ΚΑΙ ΕΚΤΕΛΕΣΗ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ

Σώζουμε το πρόγραμμα που δημιουργήσαμε και προχωράμε στη μεταγλώτιση με Ctrl-F9.

Εάν εμφανισθούν λάθη, τα διορθώνουμε, έως ότου η μεταγλώτιση αναφέρει: "Done. Compiled. … Errors: 0".

Εκτελούμε το πρόγραμμα με Shift-F9.

Στην οθόνη βλέπουμε την εκτύπωση:

4.69999981 4.9000001 4.80000019

Παρατηρούμε ότι η ακρίβεια με την οποία αποθηκεύονται οι τιμές των μεταβλητών και με την οποία γίνονται όλες οι πράξεις είναι **6** δεκαδικά ψηφία (απλή ακρίβεια). Όταν γίνονται πολλές πράξεις, το σφάλμα στο 7ο ψηφίο συσσωρεύεται με αποτέλεσμα η ακρίβεια να περιορίζεται σε λιγότερα ψηφία, όσο αυξάνει ο αριθμός των πράξεων.

ΠΡΟΓΡΑΜΜΑ ΔΙΠΛΗΣ ΑΚΡΙΒΕΙΑΣ

Για να αυξήσουμε την ακρίβεια των αποτελεσμάτων, χρησιμοποιούμε μεταβλητές διπλής ακρίβειας **(double precision)**, όπως στο πρόγραμμα tut2.for:

```
real*8 x,y,avg

x=4.123456789012345678 90d0
y=4.9d0
avg=(x+y)/2.0d0
write(*, "(3F20.17)") x,y,avg
pause
end
```

Δηλώνουμε τις μεταβλητές στην αρχή του προγράμματος ως **real*8**. Επιπλέον, **κάθε πραγματικός αριθμός πρέπει να δηλώνεται π.χ. ως** 2.0**d0** (και όχι απλά ως 2.0). Τα ονόματα των μεταβλητών μπορεί να έχουν μέχρι και 8 χαρακτήρες.

Εδώ, αντί της απλής εντολής print, χρησιμοποιήσαμε την εντολή write, με την οποία μπορούμε να επιλέξουμε πόσες μεταβλητές θα εκτυπωθούν (3), πόσα συνολικά ψηφία (μαζί με τα κενά) (20) και πόσα ψηφία μετά την υποδιαστολή (17). Επιπλέον δυνατότητες διαμόρφωσης της εκτύπωσης μεταβλητών θα βρείτε εδώ:
http://www.cs.mtu.edu/~shene/COURSES/cs201/NOTES/chap05/format.html
Σημειώστε ότι, στο παραπάνω παράδειγμα ζητήσαμε να εκτυπωθούν 17 ψηφία μετά την υποδιαστολή. Όταν όμως τρέξουμε το πρόγραμμα, παρατηρούμε ότι στην εκτύπωση μόνο τα 15 ψηφία είναι σωστά, ενώ τα δύο τελευταία είναι τυχαία. Αυτό συμβαίνει διότι η «διπλή ακρίβεια» περιορίζεται στα 15 ψηφία μόνο.

ΕΙΣΑΓΩΓΗ ΔΕΔΟΜΕΝΩΝ ΣΤΗΝ ΟΘΟΝΗ

Οι τιμές των μεταβλητών μπορούν να εισαχθούν κατά την εκτέλεση όπως στο παρακάτω παράδειγμα (tut3.f) :

```
real*8 x, y, avg

read*, x, y
avg=(x+y)/2.0d0
print *, x, y, avg
pause
end
```

Μόλις εκτελέσουμε το πρόγραμμα, εισάγουμε την τιμή του **x** και πατάμε **Enter** κ.ο.κ.

ΕΠΑΝΑΛΗΠΤΙΚΗ ΔΙΑΔΙΚΑΣΙΑ (LOOP)

Εάν θέλουμε να επαναλάβουμε μια διαδικασία πολλές φορές, τότε χρησιμοποιούμε την εντολή **do**, όπως στο εξής παράδειγμα (tut4.f) :

```
real*8 x, y, avg
integer i

do i=1,3
  read*, x, y
  avg=(x+y)/2.0d0
  print *, x, y, avg
end do

pause
end
```

Η μεταβλητή **i** δηλώθηκε ως ακέραια μεταβλητή **(integer)** και χρησιμοποιείται για τη μέτρηση του αριθμού των επαναλήψεων. Η εντολή **do** πρέπει να τερματίζεται με την εντολή **end do**.

ΑΡΙΘΜΗΤΙΚΕΣ ΠΡΑΞΕΙΣ

Οι πράξεις μεταξύ αριθμών ή μεταβλητών γίνονται με τα συνήθη σύμβολα **+**, **-**, *****, **/** . Ένας αριθμός υψώνεται σε κάποια δύναμη με τη χρήση του ******, π.χ. το 4^2 γράφεται ως 4**2 και το 4^{-2} γράφεται ως 4**(-2). Ισχύουν οι συνήθεις προτεραιότητες στην εκτέλεση των διάφορων πράξεων και πρέπει να χρησιμοποιούνται **παρενθέσεις** όπου χρειάζεται.

ΓΝΩΣΤΕΣ ΣΥΝΑΡΤΗΣΕΙΣ

Μερικές χρήσιμες συναρτήσεις, που είναι γνωστές στη Fortran είναι οι εξής:

`sin(x)`	ημίτονο γωνίας x (σε rad)
`cos(x)`	συνημίτονο
`tan(x)`	εφαπτομένη
`asin(x)`	τόξο ημιτόνου. Το αποτέλεσμα είναι μεταξύ $-\pi/2$ και $+\pi/2$.
`acos(x)`	τόξο συνημιτόνου. Το αποτέλεσμα είναι μεταξύ 0 και $+\pi$.
`atan(x)`	τόξο εφαπτομένη. Το αποτέλεσμα είναι μεταξύ $-\pi/2$ και $+\pi/2$.
`sinh(x)`	υπερβολικό ημίτονο
`cosh(x)`	υπερβολικό συνημίτονο
`tanh(x)`	υπερβολική εφαπτομένη

`sqrt(x)`	τετραγωνική ρίζα
`log(x)`	Νεπέριος λογάριθμος (με βάση το e = 2.718281828...)
`exp(x)`	e^x
`log10(x)`	λογάριθμος με βάση το 10
`abs(x)`	απόλυτη τιμή του x
`sign(1.0,x)`	πρόσημο πραγματικού αριθμού
`sign(1,x)`	πρόσημο ακέραιου αριθμού

Το παρακάτω πρόγραμμα (tut5.f) διαβάζει έναν αριθμό και εκτυπώνει το λογάριθμό του:

```
real*8 x, res

read*, x
res=log10(x)
print *, x, res
pause
end
```

ΔΙΑΔΙΚΑΣΙΑ ΕΠΙΛΟΓΗΣ IF-THEN-ELSE

Σε κάποιο σημείο του προγράμματος, ίσως να θέλουμε να επιλέξουμε ποια εντολή θα εκτελέσουμε, ανάλογα με την τιμή μιας μεταβλητής. Δύο μεταβλητές (ή μία μεταβλητή και κάποιος αριθμός) μπορεί να συγκριθούν με τις εξής σχέσεις:

`.eq.`	ίση	equal
`.ge.`	μεγαλύτερη ή ίση	greater or equal
`.gt.`	μεγαλύτερη	Greater than
`.le.`	μικρότερη ή ίση	less or equal
`.lt.`	μικρότερη	less than
`.ne.`	διάφορη	not equal

Στο παρακάτω παράδειγμα (tut6.f) υπολογίζουμε την τετραγωνική ρίζα ενός αριθμού μόνο αφού πρώτα ελέγξουμε ότι είναι θετικός αριθμός.

```
real*8 x, sr

read*, x

if (x .ge. 0.0) then
  sr=sqrt(x)
  print *, x, sr
else
  print *, 'No real root exists for this number.'
  stop
end if

pause
 end
```

Η εντολή **if (x. ge. 0.0)** ελέγχει εάν η πραγματική μεταβλητή **x** έχει θετική ή μηδενική τιμή. Εάν αυτό αληθεύει, τότε **(then)** υπολογίζει και εκτυπώνει την τετραγωνική ρίζα. Αλλιώς, **(else)** εκτυπώνει μία φράση, η οποία πρέπει να περικλείεται ανάμεσα σε ‘ ‘. Η εντολή **stop** χρησιμεύει για να τερματίσει το πρόγραμμα, μετά από τον έλεγχο κάποιας συνθήκης. Η εντολή **if** πρέπει να τελειώνει με την εντολή **end if**.

Εάν δεν έχουμε εναλλακτική εντολή, το **else** μπορεί να παραληφθεί, π.χ.

…

```
if (x .ge. 0.0) then
  sr=sqrt(x)
  print *, x, sr
end if
…
```

Επίσης, εάν παραλείψουμε το **else** και θέλουμε να εκτελέσουμε μόνο μια εντολή μετά το **if**, τότε μπορούμε να παραλείψουμε και το **then**, π.χ.

```
if (x .ge. 0.0) print *, sqrt(x)
```

Μπορούμε να έχουμε συνεχόμενες διαδικασίες επιλογής, π.χ.

```
if ... then
  ...
  else if ... then
  ...
  else if ... then
  ...
  else
  ...
end if
```

οι οποίες τερματίζονται από ένα μόνο **end if** στο τέλος.

Η σύγκριση δύο μεταβλητών, π.χ. **(x. gt. y)** μπορεί να είναι αληθής **(true)** ή ψευδής **(false)**. Χρησιμοποιώντας τους λογικούς τελεστές **.or.**, **.and.** και **.not.** μπορούμε να σχηματίσουμε πιο περίπλοκες συνθήκες, οι οποίες θέλουμε να ικανοποιούνται πριν εκτελέσουμε μια εντολή, π.χ.

```
if ( ((x .gt. y) .and. (y .lt. z)) .or. (.not. (z .ge. m))) then ...
```

ΕΞΟΔΟΣ ΑΠΟ IF-THEN-ELSE

Εάν μετά τον έλεγχο κάποιας συνθήκης πρέπει να συνεχίσει το πρόγραμμα, μπορούμε να βγούμε από ένα βρόχο if-then-else, με την εντολή **go to**, όπως στο εξής παράδειγμα:

```
if ... then
  ...
  else if ... then
    go to 10
  else if ... then
  ...
  else
  ...
end if

10   continue
```

Στο παραπάνω παράδειγμα, το «**10**» είναι απλά μια **ετικέτα (label)** η οποία δείχνει σε ποιο σημείο συνεχίζει **(continue)** το πρόγραμμα. Οι ετικέτες είναι

απαραίτητες, διότι σ'ένα πρόγραμμα μπορεί να υπάρχουν πολλά go to – continue. **Η ετικέτα μπροστά από την εντολή continue πρέπει να αρχίζει από τη 2η στήλη!**

ΧΡΗΣΗ ΔΙΑΝΥΣΜΑΤΩΝ ΚΑΙ ΠΙΝΑΚΩΝ

Ένα διάνυσμα αποτελούμενο π.χ. από 10 στοιχεία ορίζεται όπως στο εξής παράδειγμα (tut7.f), στο οποίο δίνουμε τιμές σε δυο διανύσματα και υπολογίζουμε το άθροισμά τους.

```
real*8 A(10), B(10), C(10)
integer i

do i=1,10
  A(i)=10.0d0*i
  B(i)=20.0d0*i
end do

do i=1,10
  C(i)=A(i)+B(i)
  print *, A(i), B(i), C(i)
end do

pause
end
```

Ένας διδιάστατος πίνακας με 20 γραμμές και 30 στήλες ορίζεται π.χ. ως M(10,20). Στο παρακάτω παράδειγμα (tut8.f) ορίζουμε έναν πίνακα, δίνουμε τιμές στα στοιχεία του με βάση κάποια εξίσωση και τον εκτυπώνουμε στην οθόνη.

```
real*8 M(6,3)
integer i,j

do i=1,6
  do j=1,3
    M(i,j)=2.0d0*i+3.0d0*j
  end do
end do

do i=1,6
  print *, M(i,1), M(i,2), M(i,3)
end do

pause
end
```

ΣΤΑΘΕΡΕΣ

Κάποιοι αριθμοί που χρησιμοποιούνται συχνά, μπορεί να ορισθούν ως σταθερές, με την εντολή parameter, στην αρχή του προγράμματος. Π.χ. η αρχή του προγράμματος tut7.f μπορεί να γίνει

```
      parameter (n=10)
      real*8 A(n), B(n), C(n)
```

ΣΥΝΑΡΤΗΣΕΙΣ

Κάποιοι υπολογισμοί είναι χρήσιμο να γίνονται εκτός του κυρίως προγράμματος. Μπορούμε να ορίσουμε δικές μας συναρτήσεις, πρίν το κυρίως πρόγραμμα, με την εντολή **function** (η οποία τελειώνει με **end**). Η function παίρνει κάποια (μία ή περισσότερες) μεταβλητή, εκτελεί ορισμένες πράξεις και επιστρέφει το αποτέλεσμα. Το όνομα της function χρησιμοποιείται εσωτερικά ως μεταβλητή για να αποθηκευθεί το αποτέλεσμα. Στο παρακάτω παράδειγμα (tut9.f) ορίζουμε μια συνάρτηση, η οποία υπολογίζει το παραγοντικό ενός ακεραίου αριθμού. Στο κυρίως πρόγραμμα, το οποίο αρχίζει με την εντολή **program** (και κάποιο όνομα) για να ξεχωρίζει από τη συνάρτηση, διαβάζουμε από την οθόνη έναν ακέραιο αριθμό, υπογίζουμε το παραγοντικό του και το εκτυπώνουμε.

```
function factor(n)
  integer n, i

  factor=1
  do i=2,n
    factor=factor*i
  end do
  return

end

program factorial
integer i, nf

read *, i
nf=factor(i)
print *, nf

pause
end
```

ΥΠΟΡΟΥΤΙΝΕΣ

Όταν κάποιο τμήμα ενός προγράμματος επαναλαμβάνεται συχνά, μπορούμε να το ορίσουμε ως ξεχωριστή υπορουτίνα **(subroutine)** την οποία καλούμε όποτε χρειάζεται. Η υπορουτίνα μπορεί να δέχεται κάποιες μεταβλητές, τις οποίες μπορεί να αλλάξει. Στο παρακάτω πρόγραμμα (tut10.f), ορίζουμε μια υπορουτίνα, η οποία αλλάζει τις τιμές δύο μεταβλητών μεταξύ τους. Αυτή η υπορουτίνα χρησιμοποιείται στη συνέχεια στο κυρίως πρόγραμμα, το οποίο βρίσκει ποια είναι η αύξουσα σειρά τριών μεταβλητών.

```
subroutine swap(x,y)
  real*8 x,y,temp

  temp=x
  x=y
  y=temp
  return

end

program ascending
real*8 a, b, c

read *, a, b, c

if(a .gt. b) call swap(a,b)

if(b .gt. c) then
   call swap(b,c)
   if(a .gt. b) call swap(a,b)
end if

print *, a, b, c

pause
end
```

Η υπορουτίνα πρέπει να έχει (τουλάχιστον) ένα **return** και να τελειώνει με **end.**