ΔΕΥΤΕΡΗ ΔΙΑΛΕΞΗ

Επανάληψη

Δομή Προγράμματος

```
PROGRAM onoma
   IMPLICIT NONE
   ... Δηλώσεις
   ... Εντολές
END PROGRAM onoma
```

Τύποι

ακέραιος **INTEGER**

πραγματικός **DOUBLE PRECISION**

Δήλωση

τύπος :: όνομα_μεταβλητήςA, όνομα_μεταβλητήςB, ...

Αριθμητικοί Τελεστές

`+,-,*,/,**`

MOD`(,)`

Είσοδος/έξοδος δεδομένων

Ανάγνωση δεδομένων από το πληκτρολόγιο

```
READ *, a, b, c
```

ή, ισοδύναμα,

```
READ (*,*) a, b, c
```

όπου `a, b, c` μεταβλητές.

Είσοδος/έξοδος δεδομένων

Ανάγνωση δεδομένων από το πληκτρολόγιο

```
READ *, a, b, c
```

ή, ισοδύναμα,

```
READ (*,*) a, b, c
```

όπου `a, b, c` μεταβλητές.

Εκτύπωση δεδομένων στην οθόνη

```
PRINT *, "To apotelesma einai", a+2*b
```

ή, ισοδύναμα,

```
WRITE (*,*) "To apotelesma einai", a+2*b
```

Τυπώνουμε κείμενο, αποτέλεσμα πράξεων, κλπ.

Είσοδος/έξοδος δεδομένων

Ανάγνωση δεδομένων από το πληκτρολόγιο

```
READ *, a, b, c
```

ή, ισοδύναμα,

```
READ (*,*) a, b, c
```

όπου `a, b, c` μεταβλητές.

Εκτύπωση δεδομένων στην οθόνη

```
PRINT *, "To apotelesma einai", a+2*b
```

ή, ισοδύναμα,

```
WRITE (*,*) "To apotelesma einai", a+2*b
```

Τυπώνουμε κείμενο, αποτέλεσμα πράξεων, κλπ.

Παρατήρηση

Τι νόημα έχει ο κώδικας

```
READ *, a
a = 4
```

Παράδειγμα

Να επαληθεύσετε με πρόγραμμα ότι για δύο ακέραιους a, b, με $b > 0$, ισχύει ότι η έκφραση `(a/b) * b + MOD(a,b)` είναι ίση με a. Επομένως, διαβάστε από τον χρήστη δύο ακέραιους αριθμούς a, b, υπολογίστε την έκφραση και τυπώστε τη στην οθόνη μαζί με το a. Οι δύο αριθμοί που θα τυπώσετε πρέπει να είναι ίδιοι.

Παράδειγμα

Να επαληθεύσετε με πρόγραμμα ότι για δύο ακέραιους a, b, με b>0, ισχύει ότι η έκφραση `(a/b) * b + MOD(a,b)` είναι ίση με a. Επομένως, διαβάστε από τον χρήστη δύο ακέραιους αριθμούς a, b, υπολογίστε την έκφραση και τυπώστε τη στην οθόνη μαζί με το a. Οι δύο αριθμοί που θα τυπώσετε πρέπει να είναι ίδιοι.

```
PROGRAM check
    IMPLICIT NONE



END PROGRAM check
```

Παράδειγμα

Να επαληθεύσετε με πρόγραμμα ότι για δύο ακέραιους a, b, με b>0, ισχύει ότι η έκφραση `(a/b) * b + MOD(a,b)` είναι ίση με a. Επομένως, διαβάστε από τον χρήστη δύο ακέραιους αριθμούς a, b, υπολογίστε την έκφραση και τυπώστε τη στην οθόνη μαζί με το a. Οι δύο αριθμοί που θα τυπώσετε πρέπει να είναι ίδιοι.

```
PROGRAM check
    IMPLICIT NONE



    READ *, a,b

END PROGRAM check
```

Παράδειγμα

Να επαληθεύσετε με πρόγραμμα ότι για δύο ακέραιους a, b, με b>0, ισχύει ότι η έκφραση `(a/b) * b + MOD(a,b)` είναι ίση με a. Επομένως, διαβάστε από τον χρήστη δύο ακέραιους αριθμούς a, b, υπολογίστε την έκφραση και τυπώστε τη στην οθόνη μαζί με το a. Οι δύο αριθμοί που θα τυπώσετε πρέπει να είναι ίδιοι.

```
PROGRAM check
    IMPLICIT NONE
    INTEGER :: a,b


    READ *, a,b

END PROGRAM check
```

Παράδειγμα

Να επαληθεύσετε με πρόγραμμα ότι για δύο ακέραιους a, b, με $b > 0$, ισχύει ότι η έκφραση `(a/b) * b + MOD(a,b)` είναι ίση με a. Επομένως, διαβάστε από τον χρήστη δύο ακέραιους αριθμούς a, b, υπολογίστε την έκφραση και τυπώστε τη στην οθόνη μαζί με το a. Οι δύο αριθμοί που θα τυπώσετε πρέπει να είναι ίδιοι.

```
PROGRAM check
    IMPLICIT NONE
    INTEGER :: a,b

    PRINT *, "Dose dyo akeraious, o deyteros na einai thetikos"
    READ *, a,b

END PROGRAM check
```

Παράδειγμα

Να επαληθεύσετε με πρόγραμμα ότι για δύο ακέραιους a, b, με $b > 0$, ισχύει ότι η έκφραση `(a/b) * b + MOD(a,b)` είναι ίση με a. Επομένως, διαβάστε από τον χρήστη δύο ακέραιους αριθμούς a, b, υπολογίστε την έκφραση και τυπώστε τη στην οθόνη μαζί με το a. Οι δύο αριθμοί που θα τυπώσετε πρέπει να είναι ίδιοι.

```
PROGRAM check
   IMPLICIT NONE
   INTEGER :: a,b

   PRINT *, "Dose dyo akeraious, o deyteros na einai thetikos"
   READ *, a,b
   PRINT *, (a/b) * b + MOD(a,b), a
END PROGRAM check
```

Σταθερή ποσότητα

Μια ποσότητα που επιθυμούμε να πάρει τιμή που να μην μπορεί να αλλάξει κατά τη διάρκεια εκτέλεσης του προγράμματος, δηλώνεται με τη χρήση της λέξης **PARAMETER**.
Είναι απαραίτητο να της δώσουμε αρχική (και μόνιμη) τιμή κατά τον ορισμό της.

Παράδειγμα

```
DOUBLE PRECISION, PARAMETER :: pi=3.14159265358d0
```

Ενσωματωμένες μαθηματικές συναρτήσεις (1/4)

Συνάρτηση	**Επιστρεφόμενη τιμή**
`ABS(x)`	Απόλυτη τιμή.
`SQRT(x)`	Η τετραγωνική ρίζα του ορίσματος.
`EXP(x)`	Εκθετικό (e^x).
`LOG(x)`	Φυσικός λογάριθμος ($\ln x$).
`LOG10(x)`	Δεκαδικός λογάριθμος ($\log x$).
`MAX(x1,x2,...)`	Ο μεγαλύτερος από τα δύο ή περισσότερα ορίσματα.
`MIN(x1,x2,...)`	Ο μικρότερος από τα δύο ή περισσότερα ορίσματα.

Ενσωματωμένες μαθηματικές συναρτήσεις (2/4)

Συνάρτηση	**Επιστρεφόμενη τιμή**
`MOD(x,y)`	Το υπόλοιπο της διαίρεσης x/y, δηλαδή το `x-INT(x/y)*y`.
`INT(x)`	Ο μεγαλύτερος ακέραιος που δεν ξεπερνά το $\|x\|$, με το πρόσημο του x.
`NINT(x)`	Η πλησιέστερη ακέραια τιμή στο όρισμα.
`FLOOR(x)`	Η μεγαλύτερη ακέραια τιμή που είναι μικρότερη ή ίση από το όρισμα.
`CEILING(x)`	Η μικρότερη ακέραια τιμή που είναι μεγαλύτερη ή ίση από το όρισμα.
`REAL(x)`	Το όρισμα ως πραγματικός απλής ακρίβειας.
`DBLE(x)`	Το όρισμα ως πραγματικός διπλής ακρίβειας.

Συνάρτηση	Επιστρεφόμενη τιμή
`COS(x)`	Συνημίτονο.
`SIN(x)`	Ημίτονο.
`TAN(x)`	Εφαπτομένη.
`ACOS(x)`	Τόξο συνημιτόνου, $\cos^{-1}(x)$.
`ASIN(x)`	Τόξο ημιτόνου, $\sin^{-1}(x)$.
`ATAN(x)`	Τόξο εφαπτομένης, $\tan^{-1}(x)$.
`ATAN2(y, x)`	Τόξο εφαπτομένης του y/x, $\tan^{-1}(y/x)$.

Παρατήρηση

Τα `ATAN2(y,x)` και `ATAN(y/x)` είναι ισοδύναμα, εκτός αν x=0 (οπότε το πρώτο επιστρέφει σωστή τιμή ενώ το δεύτερο όχι).

Ενσωματωμένες μαθηματικές συναρτήσεις (4/4)

Οι τριγωνομετρικές συναρτήσεις (**SIN**, **COS**, **TAN**) δέχονται γωνία σε rad.
Οι αντίστροφες τριγωνομετρικές συναρτήσεις (**ASIN**, **ACOS**, **ATAN**, **ATAN2**) επιστρέφουν γωνία σε rad.

Ενσωματωμένες μαθηματικές συναρτήσεις (4/4)

Οι τριγωνομετρικές συναρτήσεις (**SIN**, **COS**, **TAN**) δέχονται γωνία σε rad.
Οι αντίστροφες τριγωνομετρικές συναρτήσεις (**ASIN**, **ACOS**, **ATAN**, **ATAN2**) επιστρέφουν γωνία σε rad.

Μετατροπή μοιρών σε rad

$$(\text{γωνία σε rad}) = (\text{γωνία σε μοίρες}) \times \frac{\pi}{180} \ .$$

Ενσωματωμένες μαθηματικές συναρτήσεις (4/4)

Οι τριγωνομετρικές συναρτήσεις (**SIN**, **COS**, **TAN**) δέχονται γωνία σε rad.
Οι αντίστροφες τριγωνομετρικές συναρτήσεις (**ASIN**, **ACOS**, **ATAN**, **ATAN2**) επιστρέφουν γωνία σε rad.

Μετατροπή μοιρών σε rad

$$(\text{γωνία σε rad}) = (\text{γωνία σε μοίρες}) \times \frac{\pi}{180} \ .$$

Υπολογισμός του π

$$\tan\left(\frac{\pi}{4}\right) = 1 \Rightarrow \pi = 4\tan^{-1}(1) \ .$$

Επομένως, `pi = 4d0 * ATAN(1d0)`.

Ενσωματωμένες μαθηματικές συναρτήσεις: Παράδειγμα Α'

Θέλω να τυπώσω την τετραγωνική ρίζα του πραγματικού που θα δίνει ο χρήστης:

```
PROGRAM riza
  IMPLICIT NONE
  DOUBLE PRECISION :: x

  PRINT *, "Δώσε μη αρνητικό πραγματικό"
  READ *, x
  PRINT *, "Η ρίζα είναι", SQRT(x)
END PROGRAM riza
```

Ενσωματωμένες μαθηματικές συναρτήσεις: Παράδειγμα Β'

Θέλω να βρω το ημίτονο, το συνημίτονο και την εφαπτομένη της γωνίας (σε μοίρες) που θα δίνει ο χρήστης:

```
PROGRAM trig
  IMPLICIT NONE
  DOUBLE PRECISION :: theta, phi, pi

  PRINT *, "Δώσε γωνία σε μοίρες"
  READ *, theta
  pi = 4d0 * ATAN(1d0)
  phi = theta * pi / 180d0
  PRINT *, "Το ημίτονο, το συνημίτονο και η εφαπτομένη είναι"
  PRINT *, SIN(phi), COS(phi), TAN(phi)
END PROGRAM trig
```

Ρητή μετατροπή αριθμητικής τιμής σε άλλο τύπο

Τι θα τυπωθεί στον παρακάτω κώδικα;

```
PROGRAM example
   IMPLICIT NONE
   INTEGER :: a, b
   a = 9
   b = 2
   PRINT *, a/b
END PROGRAM example
```

Ρητή μετατροπή αριθμητικής τιμής σε άλλο τύπο

Τι θα τυπωθεί στον παρακάτω κώδικα;

```
PROGRAM example
   IMPLICIT NONE
   INTEGER :: a, b
   a = 9
   b = 2
   PRINT *, a/b
END PROGRAM example
```

Χωρίς να αλλάξουμε τον τύπο των a,b, πώς θα υπολογίσουμε το λόγο τους (και όχι το πηλίκο);

Ρητή μετατροπή αριθμητικής τιμής σε άλλο τύπο

Τι θα τυπωθεί στον παρακάτω κώδικα;

```
PROGRAM example
   IMPLICIT NONE
   INTEGER :: a, b
   a = 9
   b = 2
   PRINT *, a/b
END PROGRAM example
```

Χωρίς να αλλάξουμε τον τύπο των a,b, πώς θα υπολογίσουμε το λόγο τους (και όχι το πηλίκο);

`DBLE(a)/DBLE(b)` για διπλή ακρίβεια,
`REAL(a)/REAL(b)` για απλή ακρίβεια.

Τύπος χαρακτήρα (**CHARACTER**)

Ο τύπος **CHARACTER** είναι κατάλληλος για την αναπαράσταση ποσοτήτων που οι δυνατές τιμές τους είναι χαρακτήρες ή σειρές χαρακτήρων.

Δήλωση

```
CHARACTER :: c
CHARACTER (20) :: s
```

Τιμές

Απλοί χαρακτήρες ή σειρές χαρακτήρων εντός απλών (') ή διπλών (") εισαγωγικών:

```
c = 'a'
s = "This is a message"
```

Λογικός τύπος (LOGICAL)

Ο τύπος **LOGICAL** είναι κατάλληλος για την αναπαράσταση ποσοτήτων που μπορούν να πάρουν δύο μόνο τιμές (π.χ. ναι/όχι, αληθές/ψευδές,...).

Δήλωση

```
LOGICAL :: a
```

Τιμές

.TRUE. ή **.FALSE.**

Παράδειγμα

```
a = .TRUE.
```

Απλή λογική έκφραση

Σύγκριση δύο ποσοτήτων με τους τελεστές:

Τελεστής	**Σύγκριση**	**Τελεστής**	**Σύγκριση**
==	ίσο	/=	άνισο
>	μεγαλύτερο	>=	μεγαλύτερο ή ίσο
<	μικρότερο	<=	μικρότερο ή ίσο

Παραδείγματα

```
x > 1D5     i /= 3     k == 5
```

Απλή λογική έκφραση

Σύγκριση δύο ποσοτήτων με τους τελεστές:

Τελεστής	**Σύγκριση**	**Τελεστής**	**Σύγκριση**
==	ίσο	/=	άνισο
>	μεγαλύτερο	>=	μεγαλύτερο ή ίσο
<	μικρότερο	<=	μικρότερο ή ίσο

Παραδείγματα

```
x > 1D5      i /= 3      k == 5
```

Παρατηρήσεις

- Οι τελεστές σύγκρισης έχουν χαμηλότερη προτεραιότητα από τους αριθμητικούς τελεστές (επομένως, το `k>3+2` κάνει αυτό που αναμένουμε, τη σύγκριση του `k` με το 5).
- Η σύγκριση λογικών ποσοτήτων δε γίνεται με τους τελεστές ==, /= αλλά με τους `.EQV.`, `.NEQV.` αντίστοιχα.

Σύνθετη λογική έκφραση

Προκύπτει από ένωση απλών λογικών εκφράσεων με τους τελεστές `.AND.`, `.OR.`, ή αλλαγή της τιμής μίας λογικής έκφρασης με τον τελεστή `.NOT.`

Παραδείγματα

- $10 \leq x \leq 20 \Rightarrow$ `10d0 <= x .AND. x <= 20d0`
- k είναι 3 ή 5 $\Rightarrow$ `k == 3 .OR. k == 5`
- `.NOT. j==3`

Σύνθετη λογική έκφραση

Προκύπτει από ένωση απλών λογικών εκφράσεων με τους τελεστές `.AND.`, `.OR.`, ή αλλαγή της τιμής μίας λογικής έκφρασης με τον τελεστή `.NOT.`

Παραδείγματα

- $10 \leq x \leq 20 \Rightarrow$ `10d0 <= x .AND. x <= 20d0`
- k είναι 3 ή 5 $\Rightarrow$ `k == 3 .OR. k == 5`
- `.NOT. j==3`

Προτεραιότητες

- Οι λογικοί τελεστές έχουν χαμηλότερη προτεραιότητα από τους τελεστές σύγκρισης.
- Το `.NOT.` έχει υψηλότερη προτεραιότητα από το `.AND.`, το οποίο έχει υψηλότερη προτεραιότητα από το `.OR.`

Εκχώρηση λογικής έκφρασης σε λογική μεταβλητή

Μια απλή ή σύνθετη λογική έκφραση έχει τιμή **.TRUE.** ή **.FALSE.** και μπορεί να εκχωρηθεί σε μεταβλητή τύπου **LOGICAL**:

```
LOGICAL :: b,c

b = k /= 5
c = 10d0 <= x .AND. x <= 20d0
```

Σύγκριση πραγματικών αριθμών

Τι θα τυπωθεί με τον παρακάτω κώδικα;

```
PRINT *, 0.1d0 + 0.2d0 - 0.3d0
PRINT *, 0.1d0 - 0.3d0 + 0.2d0
```

Σύγκριση πραγματικών αριθμών

Τι θα τυπωθεί με τον παρακάτω κώδικα;

```
PRINT *, 0.1d0 + 0.2d0 - 0.3d0
PRINT *, 0.1d0 - 0.3d0 + 0.2d0
```

Απάντηση

`5.5511151231257827E-017` δηλαδή $\approx 6 \times 10^{-17}$

`2.7755575615628914E-017` δηλαδή $\approx 3 \times 10^{-17}$

Όχι 0, ούτε καν ο ίδιος αριθμός.

Σύγκριση πραγματικών αριθμών

Τι θα τυπωθεί με τον παρακάτω κώδικα;

```
PRINT *, 0.1d0 + 0.2d0 - 0.3d0
PRINT *, 0.1d0 - 0.3d0 + 0.2d0
```

Απάντηση

`5.5511151231257827E-017` δηλαδή $\approx 6 \times 10^{-17}$

`2.7755575615628914E-017` δηλαδή $\approx 3 \times 10^{-17}$

Όχι 0, ούτε καν ο ίδιος αριθμός.

Κανόνας

Οι πράξεις μεταξύ πραγματικών αριθμών δεν είναι απόλυτα ακριβείς. Επομένως, δεν συγκρίνουμε ποτέ για ισότητα πραγματικές ποσότητες που τουλάχιστον η μία προέκυψε από πράξεις.

Εντολή `IF` (1/5)

```
IF (λογική_έκφραση) THEN
        ... ! τμήμα A
ELSE
        ... ! τμήμα B
END IF
```

Αν η λογική_έκφραση είναι αληθής, εκτελούνται οι εντολές στο τμήμα Α αλλιώς εκτελούνται οι εντολές στο τμήμα Β.

Εντολή IF (1/5)

IF *(λογική_έκφραση)* ***THEN***
... ! τμήμα A
ELSE
... ! τμήμα B
END IF

Αν η λογική_έκφραση είναι αληθής, εκτελούνται οι εντολές στο τμήμα A αλλιώς εκτελούνται οι εντολές στο τμήμα B.

Παράδειγμα

Αν η πραγματική ποσότητα x είναι μεγαλύτερη από 0 να τυπώσουμε «θετική» αλλιώς να τυπώσουμε «αρνητική ή 0»:

```
IF (x > 0.0d0) THEN
   PRINT *, "Θετική"
ELSE
   PRINT *, "Αρνητική ή 0"
END IF
```

Εντολή IF (2/5)

Παρατήρηση

Στην εντολή **IF** μπορούμε να παραλείψουμε το τμήμα από το **ELSE** μέχρι το **END IF**, αν είναι κενό το τμήμα B:

```
IF (λογική_έκφραση) THEN
        ... ! τμήμα A
END IF
```

Εντολή IF (2/5)

Παρατήρηση

Στην εντολή **IF** μπορούμε να παραλείψουμε το τμήμα από το **ELSE** μέχρι το **END IF**, αν είναι κενό το τμήμα B:

IF *(λογική_έκφραση)* ***THEN***
... ! τμήμα A
END IF

Παράδειγμα

Η μεταβλητή y να αλλάξει πρόσημο αν η πραγματική ποσότητα x είναι μεταξύ -1 και 1:

```
IF (-1.0d0 <= x .AND. x <= 1.0d0) THEN
   y = -y
END IF
```

Εντολή `IF` (3/5)

Παρατηρήσεις

- Στην περίπτωση που έχουμε μία εντολή στο τμήμα Α και καμία στο τμήμα Β, δηλαδή όταν το `IF` είναι

  IF *(λογική_έκφραση)* ***THEN***
  εντολή
  END IF

  μπορεί να γραφεί ισοδύναμα ως

  IF *(λογική_έκφραση) εντολή*

Εντολή `IF` (3/5)

Παρατηρήσεις

- Στην περίπτωση που έχουμε μία εντολή στο τμήμα Α και καμία στο τμήμα Β, δηλαδή όταν το `IF` είναι

  IF *(λογική_έκφραση)* ***THEN***
  εντολή
  END IF

  μπορεί να γραφεί ισοδύναμα ως

  IF *(λογική_έκφραση) εντολή*

- Ως λογική έκφραση μπορούμε να έχουμε ποσότητα τύπου `LOGICAL`.

Εντολή IF (3/5)

Παρατηρήσεις

- Στην περίπτωση που έχουμε μία εντολή στο τμήμα Α και καμία στο τμήμα Β, δηλαδή όταν το **IF** είναι

  IF *(λογική_έκφραση)* ***THEN***
  εντολή
  END IF

  μπορεί να γραφεί ισοδύναμα ως

  IF *(λογική_έκφραση) εντολή*

- Ως λογική έκφραση μπορούμε να έχουμε ποσότητα τύπου **LOGICAL**.

Παράδειγμα

Η μεταβλητή y να αλλάξει πρόσημο αν η πραγματική ποσότητα x είναι μεταξύ -1 και 1:

```
LOGICAL :: d
d = -1.0d0 <= x .AND. x <= 1.0d0
IF (d) y = -y
```

Εντολή IF (4/5)

Για πολλές αμοιβαία αποκλειόμενες λογικές εκφράσεις:

```
IF (λογική_έκφραση1) THEN
        ... ! τμήμα Α
ELSE IF (λογική_έκφραση2) THEN
        ... ! τμήμα Β
ELSE IF (λογική_έκφραση3) THEN
        ... ! τμήμα Γ
ELSE
        ... ! τμήμα Δ
END IF
```

Πριν την εκτέλεση της εντολής ελέγχονται οι λογικές εκφράσεις. Αν μία είναι αληθής, εκτελούνται οι εντολές που «συνδέονται» με αυτή. Αν όλες είναι ψευδείς, εκτελούνται οι εντολές μετά το **ELSE** (αν υπάρχει).

Εντολή IF (5/5)

Παράδειγμα IF/ELSE IF

Αν το πραγματικό x είναι στο διάστημα $[0, 1)$ το ακέραιο k να γίνει 0. Αν το x είναι στο διάστημα $[1, 2)$ το k να γίνει 1. Αν το x είναι στο διάστημα $[2, 3)$ το k να γίνει 2. Αλλιώς, να γραφτεί μήνυμα λάθους:

```
IF (0d0 <= x .AND. x < 1.0d0) THEN
   k = 0
ELSE IF (1d0 <= x .AND. x < 2.0d0) THEN
   k = 1
ELSE IF (2d0 <= x .AND. x < 3.0d0) THEN
   k = 2
ELSE
   PRINT *, "λάθος"
END IF
```

Εντολή SELECT CASE (1/4)

```
SELECT CASE (i)
  CASE (v1)
  ...
  CASE (v2)
  ...
  CASE (v3)
  ...
  CASE default
  ...
END SELECT
```

Ελέγχεται το i.
Αν είναι ίσο με v1 εκτελούνται οι εντολές μετά το **case** (v1).
Αν είναι ίσο με v2 εκτελούνται οι εντολές μετά το **case** (v2), κλπ.
Αν δεν είναι ίσο με κανένα, εκτελούνται οι εντολές μετά το **case default**.

Παρατήρηση

Το i μπορεί να είναι ακέραιος ή χαρακτήρας ή ποσότητα λογικού τύπου αλλά όχι πραγματικός.
Οι τιμές v1, v2, ... πρέπει να είναι σταθερές τιμές.

Εντολή SELECT CASE (2/4)

Α΄ Παράδειγμα

Αν ο ακέραιος k είναι 1,2,3,4 να τυπώσουμε “up”, “down”, “left”, “right” αντίστοιχα. Για άλλη τιμή να γράψουμε “error”:

```
SELECT CASE (k)
  CASE (1)
  PRINT *, "up"
  CASE (2)
  PRINT *, "down"
  CASE (3)
  PRINT *, "left"
  CASE (4)
  PRINT *, "right"
  CASE default
  PRINT *, "error"
END SELECT
```

Εντολή **SELECT CASE** (3/4)

Ομαδοποίηση τιμών

```
SELECT CASE (i)
  CASE (v1,v2,v3)
  ...
  CASE (v4:v5)  ! από v4 έως και v5
  ...
  CASE (v6:)   ! μεγαλύτερο ή ίσο με v6
   ...
  CASE (:v7)  !μικρότερο ή ίσο με v7
  ...
  CASE default
  ...
END SELECT
```

Εντολή SELECT CASE (4/4)

Β΄ Παράδειγμα

Αν ο ακέραιος k είναι 2, 4, 6, 8 να τυπώσουμε “even”. Αν είναι 1, 3, 5, 7, 9 να τυπώσουμε “odd”. Αν είναι αρνητικός να τυπώσουμε “error”:

```
SELECT CASE (k)
   CASE (2,4,6,8)
      PRINT *, "even"
   CASE (1,3,5,7,9)
      PRINT *, "odd"
   CASE (:-1)
      PRINT *, "error"
END SELECT
```