

bits and bytes

- ❑ Ο υπολογιστής χρησιμοποιεί τη **κύρια μνήμη** για αποθήκευση δεδομένων
- ❑ Η μνήμη χωρίζεται σε **words** και κάθε word περιέχει τμήμα πληροφορίας
- ❑ Ο αριθμός των words σε μια κεντρική μνήμη εξαρτάται από τον Η/Υ (αρκετές χιλιάδες για ένα μικρο-processor σε εκατοντάδες εκατομμύρια για μεγάλους υπολογιστές)
- ❑ Κάθε word αποτελείται από μικρότερες μονάδες που ονομάζονται **bits**

Το bit μπορεί να έχει μόνο μια από δύο τιμές: **0 ή 1**

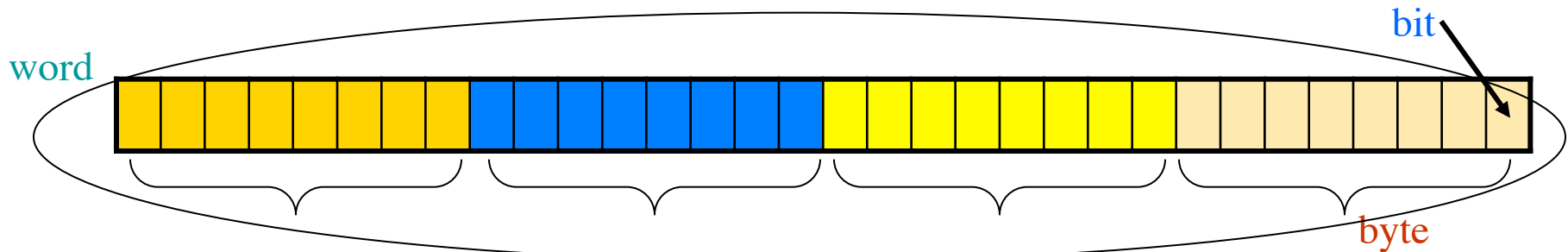
Αν το word περιέχει m bits τότε το word μπορεί να είναι σε οποιαδήποτε από 2^m διαφορετικές καταστάσεις

Η πραγματική σημασία που δίνεται σε κάποιο ιδιαίτερο συνδυασμό από 0 και 1 σε ένα word εξαρτάται από το μοντέλο του υπολογιστή

- ❑ Όλοι οι υπολογιστές (**σχεδόν**) αποθηκεύουν τους θετικούς INTEGERS με τέτοιο τρόπο ώστε κάθε bit που παίρνει τη τιμή 1 στο word αντιπροσωπεύει μια τιμή που είναι $2^{(n-1)}$ όπου n η σχετική θέση του bit στο word μετρώντας από δεξιά.

bits and bytes

- ❑ Οι περισσότεροι υπολογιστές έχουν μνήμη με 32 bits/word με ενδιάμεσο διαχωρισμό του word σε 4 ομάδες από 8 συνεχή bits
- ❑ Ένα τέτοιο γκρουπ αποτελούμενο από 8 bits ονομάζεται **byte**
- ❑ Ένα byte έχει 2^8 ή 256 διαφορετικές καταστάσεις.
Όλοι οι αριθμοί αντιπροσωπεύονται σε δυαδική μορφή



- ❑ Υπάρχει περιορισμένη ακρίβεια και επομένως προσεγγίσεις
- ❑ Το μήκος του word είναι συνήθως 32 bits ή 4 bytes
όπου κάθε byte αποτελείται από 8 bits

Μονάδες μέτρησης για την αποθήκευση αριθμών είναι:

$$1 \text{ byte} = 1\text{B} = 8 \text{ bits} = 8\text{b}$$

$$1 \text{ kbyte} = 2^{10} \text{ bytes} = 1024 \text{ bytes}$$

$$1 \text{ Mbyte} = 2^{10} \text{ kbytes} = 1024 \times 1024 \text{ bytes}$$

Γιατί υπάρχει περιορισμός στους αριθμούς

- ❑ Σχετίζεται με την αναπαράσταση αριθμών στον υπολογιστή
- ❑ Τρία συστήματα αριθμών:
 - Οι αριθμοί μέτρησης 0,1,2,...,32767
Συνδέεται με τον δείκτη των καταλόγων της μηχανής και άρα το εύρος των αριθμών είναι προσδιορισμένο.
 - Οι αριθμοί σταθερής υποδιαστολής
Οι αριθμοί αυτοί έχουν σταθερό μήκος = μήκος του word του Η/Υ ή κάποιο πολλαπλάσιο
Η διαφορά μεταξύ διαδοχικών αριθμών είναι η ίδια
Τέτοιοι είναι οι αριθμοί των υπολογισμών του χεριού
(Όλοι οι πίνακες είναι συνήθως σταθερής υποδιαστολής)
 - Οι αριθμοί κινητής υποδιαστολής (floating point numbers)
Σχετίζονται με τη λεγόμενη επιστημονική σήμανση
Το σύστημα αυτό είναι σχεδιασμένο για να περιγράφει και μικρούς αλλά και μεγάλους αριθμούς


Αναπαράσταση αριθμών στον Η/Υ

- Οι αριθμοί κινητής υποδιαστολής (floating point numbers)

$$.31415927 \times 10^1$$

$$.12345678 \times 10^{-1}$$

$$-.12345678 \times 10^4$$

**Mantissa:** Το πρώτο τμήμα των 8 ψηφίων στους παραπάνω αριθμούς
Εκθέτης: Το τελευταίο ψηφίο (με εύρος τιμών [-38, +38])

Η mantissa και ο εκθέτης αποθηκεύονται στο ίδιο word της μηχανής

Σαν αποτέλεσμα η mantissa ενός αριθμού κινητής υποδιαστολής είναι μικρότερου μήκους του αντίστοιχου αριθμού σταθερής υποδιαστολής

Απεικόνιση αριθμών στους υπολογιστές

- Ένας ακέραιος αριθμός 2^N αντιπροσωπεύεται από N bits

Το 1^ο bit δίνει το πρόσημο

Τα υπόλοιπα $N-1$ bits χρησιμοποιούνται για τον αριθμό

Άρα 32 bits INTEGERS θα πέρνουν τιμές στο διάστημα:

$$-2147483648 < \text{Integer}_{32} < 2147483648 = 2^{31}$$

- Η αριθμητική με Integer αριθμούς είναι ακριβής εκτός από **υπερχείλιση** (overflow) και **υποχείλιση** (underflow)

- Οι αριθμοί **απλής ακρίβειας** (single precision ή floating - point F) χρειάζονται 4 bytes (32 bits) για την αναπαράστασή τους

Τα οποία χρησιμοποιούνται ως εξής:

{	23 bits για mantissa
	8 bits για εκθέτη
	1 bit για το πρόσημο

Η μικρότερη mantissa είναι $2^{-23} \sim 10^{-7}$

Εκθέτης:

$2^{\alpha - 127}$ όπου $\alpha \in [0, 255]$ (2^8 bits)

↖ "bias"

Απεικόνιση αριθμών

□ Για αριθμούς **διπλής ακρίβειας** χρειάζονται 8 bytes δηλαδή 64 bits

και χρησιμοποιούνται ως εξής:

- 52 bits για την mantissa
- 1 bit για το πρόσημο
- 11 bits για τον εκθέτη (**bias=1023**)

Η μικρότερη mantissa είναι $2^{-52} \sim 10^{-16}$

Ο εκθέτης είναι $2^{\alpha-1023}$ με $\alpha \in [0, 2048]$ (2^{11} bits)

Διάστημα

$$2.9 \times 10^{-39} \leq \text{float}_{32} \leq 3.4 \times 10^{38}$$

$$10^{-322} \leq \text{double}_{64} \leq 10^{308}$$

Ακρίβεια

float_{32} : 5-7 δεκαδικά ψηφία

double_{64} : 16 δεκαδικά ψηφία

Αριθμοί κινητής υποδιαστολής

□ Έχουν μια σειρά από ιδιαιτερότητες:

- Ας υποθέσουμε για απλότητα ότι έχουμε μια mantissa με 3 ψηφία και ο εκθέτης αποτελείται από 1 ψηφίο:

Το μηδέν, $0 = .000 \times 10^{-9}$ είναι απομονωμένο από τους υπόλοιπους αριθμούς

αφού οι αμέσως επόμενοι θετικοί αριθμοί είναι $.100 \times 10^{-9}$ και $.101 \times 10^{-9}$ βρίσκονται 10^{-12} μακριά!!

Δεν υπάρχει κανένας άλλος αριθμός με mantissa να ξεκινά από 0

- Κάθε δεκάδα έχει ακριβώς τον ίδιο πλήθος αριθμών: **συνολικά 900** πηγαίνοντας από $.100 \times 10^a$ μέχρι $.999 \times 10^a$ με $a = -9, -8, \dots, 0, \dots, 8, 9$

0.999x10 ⁻¹ 0.100x10 ⁰	1x10 ⁻⁴
0.999x10 ⁰ 0.100x10 ¹	1x10 ⁻³
0.999x10 ¹ 0.100x10 ²	1x10 ⁻²

- Σε κάθε δεκάδα οι 900 αριθμοί είναι χωρισμένοι σε ίσα διαστήματα

αλλά η διαφορά μεταξύ 2 δεκάδων είναι άνιση

Υπάρχει γεωμετρικό πήδημα

Αριθμοί κινητής υποδιαστολής

- ❑ Ο αριθμός 0 και -0 (δηλαδή -0.000×10^{-9}) είναι λογικά ίδιοι
Οι περισσότεροι υπολογιστές τους έχουν ίδιους αλλά μερικοί όχι
 - Συνήθως δεν υπάρχει άπειρο που να αντιστοιχεί στο μηδέν
- ❑ Στα μαθηματικά υπάρχει ένα και μοναδικό μηδέν
- ❑ Στο προγραμματισμό υπάρχουν 2 είδη:
 - Αυτό που εμφανίζεται σε εκφράσεις όπως $\alpha \cdot 0 = 0$ που συμπεριφέρεται σαν κανονικό 0
 - Αλλά και το 0 που εμφανίζεται σε σχέσεις της μορφής $\alpha - \alpha = 0$
Αυτό το 0 μπορεί να προέρχεται από $1 - (1 - \epsilon)(1 + \epsilon) = 0$ όταν το $\epsilon < \frac{1}{2} 10^{-3}$
όπου το 3 είναι ο αριθμός των δεκαδικών ψηφίων της mantissa
 - Αυτό το τελευταίο 0 εμφανίζεται κατά την αφαίρεση 2 σχεδόν ίσου μεγέθους αριθμών
 - Αποτελεί την πηγή πολλών σφαλμάτων στους υπολογισμούς με ένα ηλεκτρονικό υπολογιστή

Μεταβλητές (**variables**)

- ❑ Μεταβλητή είναι μια ποσότητα που μπορεί να αλλάζει τιμή κατά τη διάρκεια εκτέλεσης του προγράμματος
π.χ. η θερμοκρασία κατά τη διάρκεια μιας μέρας
- ❑ Η μεταβλητή πρέπει να δηλωθεί με ένα συμβολικό όνομα για να διαφοροποιείται από τις υπόλοιπες.
- ❑ Σύμφωνα με τους κανόνες της FORTRAN, το όνομα μιας μεταβλητής πρέπει να ξεκινά με ένα γράμμα και να ακολουθείτε από χαρακτήρες (γράμματα ή αριθμούς και **όχι σύμβολα !, #, \$, %, &, ^, *, (,), -, =, +, /, **) με μήκος μέχρι 32 χαρακτήρες χωρίς να παρεμβάλετε κενό.
- ❑ Υπάρχουν διάφορα είδη μεταβλητών στη FORTRAN:

Το είδος των μεταβλητών που θα χρησιμοποιηθούν σε ένα πρόγραμμα πρέπει να δηλωθεί στην αρχή του προγράμματος (μετά την εντολή PROGRAM) και πριν από οποιαδήποτε εντολή εκτέλεσης

Οι γραμμές σχολίων δεν εκτελούνται και επομένως μπορούν να εμφανίζονται οπουδήποτε στο πρόγραμμα

Μεταβλητές – προσοχή

- ❑ Αν ο ορισμός των μεταβλητών παραληφθεί τότε ο compiler (ο μεταφραστής της γλώσσας στη γλώσσα του υπολογιστή) θα κάνει κάποιες υποθέσεις:
 - Μεταβλητές το όνομα των οποίων αρχίζει από I,J,K,L,M,N θεωρούνται από τον compiler σαν ακέραια μεταβλητή (INTEGER)

π.χ. Έστω ότι έχουμε τη μεταβλητή INDEX

Αν δεν δηλωθεί ο τύπος της μεταβλητής τότε ο compiler θεωρεί ότι είναι INTEGER αφού το συμβολικό της όνομα αρχίζει από I

- Ο Η/Υ θεωρεί ότι η αριθμητική τιμή που μπορεί να πάρει η μεταβλητή αυτή είναι μόνο INTEGER αριθμός:

INDEX = 3.34

PRINT *, 'Index =', INDEX

Θα τυπώσει: Index =3

- ❑ Για αποφυγή σφαλμάτων και λαθών οι μεταβλητές πρέπει πάντοτε να ορίζονται και να λαμβάνεται υπ' όψη το είδος τους κατά την γραφή του προγράμματος
- ❑ Αριθμητικά δεδομένα πρέπει να διαχωρίζονται σε REAL και INTEGER
- ❑ Ο Η/Υ δεν κάνει αλγεβρικές πράξεις με τα συμβολικά ονόματα των μεταβλητών αλλά μόνο με τις εκάστοτε τιμές των μεταβλητών.

Είδη Μεταβλητών

❑ INTEGER (ακέραιες)

Είναι μεταβλητές στις οποίες αποθηκεύουμε ακέραιους αριθμούς

π.χ. 7, 43242, 213

➤ Η σύνταξη του ορισμού μιας μεταβλητής INTEGER είναι:

INTEGER variable_name1, variable_name2

```
PROGRAM TEST
```

```
INTEGER AVERAGE, SUM, IX
```

```
IX = 7
```

```
PRINT *, ' IX = ', IX
```

```
END
```

Είδη Μεταβλητών (συνέχεια)

❑ REAL (πραγματικές)

Είναι μεταβλητές στις οποίες αποθηκεύουμε πραγματικούς αριθμούς (αυτούς που περιέχουν υποδιαστολή)

π.χ. 3.14, 1.0, 8.23

➤ Η σύνταξή είναι:

REAL variable_name1, variable_name2

```
PROGRAM TEST
```

```
REAL MEAN, TEMP
```

```
TEMP = 32.5
```

```
PRINT *, 'TEMP = ', TEMP
```

```
END
```

Είδη Μεταβλητών (συνέχεια)

❑ CHARACTER (χαρακτήρων)

Μεταβλητές χαρακτήρων περιέχουν γραμματοσειρές με ένα ή περισσότερους χαρακτήρες

π.χ. G, KYPROS

➤ Η σύνταξη είναι:

CHARACTER name1, name2 name1, name2 περιέχουν 1 χαρακτήρα

CHARACTER*n name1, name2 name1, name2 περιέχουν n χαρακτήρες

CHARACTER name1*n1, name2*n2 name1 περιέχει n1 χαρακτήρες
name2 περιέχει n2 χαρακτήρες

```
PROGRAM TEST
```

```
CHARACTER*2 YESNO
```

```
CHARACTER CLASS*3, NUM*2
```

```
YESNO = 'Y'
```

```
CLASS = 'PHY'
```

```
PRINT *, ' YESNO = ', YESNO, ' CLASS = ', CLASS
```

```
END
```

Είδη μεταβλητών (συνέχεια)

❑ Μεταβλητές Χαρακτήρων (συνέχεια...)

- Αν μια μεταβλητή είναι «μακρύτερη» απ' ότι έχει δηλωθεί τότε οι επιπλέον χαρακτήρες στα δεξιά αποκόπτονται

```
PROGRAM TEST
```

```
CHARACTER*3 DAY
```

```
DAY = 'DEYTERA'
```

```
PRINT *, ' DAY = ', DAY
```

θα τυπώσει DAY='DEY'

```
END
```

- Αν έχει ορισθεί με περισσότερους χαρακτήρες τότε θα προστεθούν επιπλέον κενά στα δεξιά

```
PROGRAM TEST
```

```
CHARACTER*8 DAY
```

```
DAY = 'TRITH'
```

```
PRINT *, ' DAY = ', DAY
```

```
END
```

3 επιπλέον κενά

θα τυπώσει DAY='TRITHbbb'



DAY=' TRITH '

Είδη Μεταβλητών (συνέχεια)

❑ DOUBLE PRECISION (Διπλής ακρίβειας)

Για υψηλής ακρίβειας υπολογισμούς η FORTRAN παρέχει double precision μεταβλητές, που αποθηκεύονται σε 2 θέσεις μνήμης και παρέχουν διπλάσιο αριθμό σημαντικών δεκαδικών ψηφίων

➤ Η σύνταξη είναι

DOUBLE PRECISION variable_name1, variable_name2

REAL*8 variable_name1, variable_name2 (ισοδύναμος σύνταξη)

```
PROGRAM TEST
```

```
DOUBLE PRECISION TEMP
```

```
REAL*8 AVE
```

```
TEMP = 32.5D0
```

```
PRINT *, ' TEMP = ', TEMP
```

```
END
```

Αριθμητικές πράξεις και εκφράσεις

□ Οι 5 βασικές πράξεις για το προγραμματισμό στη FORTRAN είναι:

πρόσθεση	+	$\alpha + \beta$	
αφαίρεση	-	$\alpha - \beta$	
πολλαπλασιασμός	*	$\alpha * \beta$	$\alpha\beta$
διαίρεση	/	α / β	
ύψωση σε δύναμη	**	$\alpha ** 2$	(α στο τετράγωνο)

Με τα σύμβολα αυτά και με τη χρήση των σταθερών και μεταβλητών γράφουμε τις πράξεις που θέλουμε να κάνει ο Η/Υ και σχηματίζουμε εκφράσεις που αντιστοιχούν σε αριθμητικές ή αλγεβρικές παραστάσεις

Αριθμητικές πράξεις

□ Αριθμητικές πράξεις με REAL μεταβλητές και σταθερές

Αν όλες οι μεταβλητές και σταθερές σε μια αριθμητική έκφραση είναι REAL, η αριθμητική πράξη θα δώσει σαν αποτέλεσμα REAL

➤ Δε θα έχουμε αποκοπή δεκαδικών ψηφίων.

Για παράδειγμα: $5./2.$ δίνει σαν αποτέλεσμα 2.5

Αλλά $5/2$ δίνει σαν αποτέλεσμα **2 και όχι 2.5**

όπως επίσης $3/4$ δίνει σαν αποτέλεσμα **0 και όχι 0.75**

Διαίρεση ακεραίων ➡ Αποτέλεσμα θα είναι ακέραιος

□ Μικτές αριθμητικές πράξεις

Συμβαίνει όταν η αριθμητική έκφραση περιέχει REAL και INTEGER

Αν κάποιο από τα μέλη της έκφρασης είναι REAL

τότε το αποτέλεσμα είναι REAL



Θέλει μεγάλη προσοχή: Οι πράξεις γίνονται από αριστερά προς δεξιά:

$$5/2 * 3.0 = ? \quad 6.0 !!! \quad 5/2 = 2 \text{ και } 2*3 = 6$$

$$3.0 * 5/2 = ? \quad 7.5 \quad 3.0*5 = 15.0 \text{ και } 15.0/2 = 7.5$$

Αναθέσεις Μεταβλητών

❑ Μικτή Ανάθεση Μεταβλητών

Αν μια αριθμητική έκφραση αναθέτεται σε μια μεταβλητή η οποία έχει δηλωθεί σαν REAL τότε το αποτέλεσμα από τον υπολογισμό της έκφρασης θα διατηρήσει τα δεκαδικά ψηφία και θα είναι REAL

```
PROGRAM TEST
```

```
REAL PRODUCT
```

```
PRODUCT = 5*2.1
```

```
PRINT *, ' PRODUCT = ', PRODUCT
```

θα τυπώσει PRODUCT=10.5

```
END
```

➤ Αν όμως η μεταβλητή δηλωθεί σαν INTEGER τότε το αποτέλεσμα θα είναι INTEGER

```
PROGRAM TEST
```

```
INTEGER PRODUCT
```

```
PRODUCT = 5*2.1
```

```
PRINT *, ' PRODUCT = ', PRODUCT
```

θα τυπώσει PRODUCT=10

```
END
```

Κανόνες Προτεραιότητας

Οι αριθμητικές εκφράσεις υπολογίζονται σύμφωνα με τους κανόνες:

- Όλα τα εκθετικά υπολογίζονται πρώτα.
Διαδοχικά εκθετικά υπολογίζονται **από δεξιά προς τα αριστερά**
- Οι πολ/σμοι και διαιρέσεις υπολογίζονται μετά τα όποια εκθετικά, και σύμφωνα με τη σειρά που εμφανίζονται **από αριστερά στα δεξιά**
- Προσθέσεις και αφαιρέσεις πραγματοποιούνται τελευταίες και πάντα με την σειρά που εμφανίζονται **από αριστερά στα δεξιά**.

Παράδειγμα 1: $2 ** 3 ** 2 = 2 ** 9 = 512$

Παράδειγμα 2: $5 * 11 - 5 ** 2 * 4 + 9 =$
 $5 * 11 - 25 * 4 + 9 =$
 $55 - 100 + 9 = -36$

- ❑ Αν υπάρχουν παρενθέσεις, τότε η έκφραση μέσα στην παρένθεση θα υπολογισθεί πρώτη χρησιμοποιώντας τους κανόνες προτεραιότητας.
- Είναι χρήσιμο να χρησιμοποιούνται παρενθέσεις ακόμα και όταν δεν υπάρχει πολύπλοκη έκφραση γιατί βοηθά στην εύκολη ανάγνωση της έκφρασης

Εντολή Ανάθεσης (**Assignment statement**)

Η εντολή ανάθεσης χρησιμοποιείται για να δίνει τιμές σε μεταβλητές

- Έχει την ακόλουθη σύνταξη

Μεταβλητή = Αριθμητική Έκφραση ή Σταθερά

```
PROGRAM TEST
```

```
REAL X, Y, RADIUS
```

```
X = 15.
```

```
Y = 25./5.
```

```
RADIUS = X**2+Y**2
```

```
PRINT *, ' RADIUS = ', RADIUS
```

```
END
```

- Στα αριστερό μέλος της εντολής ανάθεσης (αριστερά του =) πρέπει να υπάρχει **μια και μόνο μεταβλητή**:

$A+B = C$

ΛΑΘΟΣ

$A+10.3 = \text{length}$

ΛΑΘΟΣ

$5=N$

ΛΑΘΟΣ

$A=B=C$

ΛΑΘΟΣ

- Επίσης στην Fortran ισχύει

$N = N + 1$

άθροισε στην υπάρχουσα τιμή του N το 1 και βάλε το αποτέλεσμα ξανά στο N

Σωστό γιατί πρώτα θα γίνει η αριθμητική πράξη και μετά η ανάθεση

Συναρτήσεις

- Η FORTRAN παρέχει πολλές εσωτερικές συναρτήσεις για την πραγματοποίηση υπολογισμών.

Οι συναρτήσεις αυτές δέχονται σα όρισμα οποιαδήποτε αριθμό/μεταβλητή και επιστρέφουν το ανάλογο αποτέλεσμα

➤ Ο τρόπος που καλούμε μια συνάρτηση είναι:

variable name = function_name(name1, name2,...)

Ο παρακάτω πίνακας δίνει τις πιο συνηθισμένες συναρτήσεις.

Function	Περιγραφή	Type of Argument(s)*	Type of Value
ABS (x)	Absolute value of x	I, R, DP	Same as argument
COS (x)	Cosine of x radians	R, DP	Same as argument
DBLE(x)	Conversion of x to double precision form	I, R	DP
DPROD(x,y)	Double precision product of x and y	R	DP
EXP(x)	Exponential function	R, DP	Same as argument
INT(x)	Integer part of x	R, DP	I
LOG(x)	Natural logarithm of x	R, DP	Same as argument
MAX(x ₁ ,..., X _n)	Maximum of x ₁ ,...,x _n	I, R, DP	Same as argument
MIN(x ₁ ,..., x _n)	Minimum of x ₁ ,..., x _n	I, R, DP	Same as argument
MOD(x,y)	x (mod y); $x - \text{INT}(x/y) * y$	I, R, DP	Same as argument
NINT(x)	x rounded to nearest integer	R, DP	I
REAL(x)	Conversion of x to real type	I, DP	R
SIN(x)	Sine of x radians	R, DP	Same as argument
SQRT(x)	Square root of x	R, DP	Same as argument

Προσοχή: Δεν μπορείτε να χρησιμοποιήσετε μεταβλητή με όνομα ίδιο με αυτό μιας συνάρτησης

Input/Output

Μια εντολή ανάθεσης δεν μας δείχνει το αποτέλεσμα μιας πράξης και ούτε μας επιτρέπει να αλλάξουμε τη τιμή μιας μεταβλητής κατά την διάρκεια εκτέλεσης του προγράμματος

```
PROGRAM PROJEC
```

```
REAL HGHT0, HGHT, VELOC0, VELOC, ACCEL, TIME
```

```
ACCEL = -9.807
```

```
HGHT0 = 150.0
```

```
VELOC0 = 100.0
```

```
TIME = 5.0
```

```
HGHT = 0.5 * ACCEL * TIME ** 2 + VELOC0 * TIME + HGHT0
```

```
VELOC = ACCEL * TIME + VELOC0
```

```
END
```

Δίνουν το ύψος (HGHT) και ταχύτητα (VELOC)
σε κάθε χρονική στιγμή (TIME)

Δεν μας επιστρέφει όμως οποιαδήποτε τιμή

Input/Output

- ❑ Αν οι υπολογισμοί γίνονταν για διαφορετικές τιμές αρχικού ύψους ή ταχύτητας και θέλαμε το αποτέλεσμα για διαφορετικούς χρόνους τότε θα έπρεπε να αλλάξουμε αρκετές εντολές και πολλές φορές.
 - ❑ Θα δούμε μια εντολή **output** (μεταφορά εσωτερικής πληροφορίας σε εξωτερική μονάδα- οθόνη)
 - ❑ Και μια εντολή **input** (μεταφορά από μια εξωτερική μονάδα – πληκτρολόγιο στο εσωτερικό του προγράμματος) κατά τη διάρκεια της εκτέλεσης του προγράμματος.
- Η FORTRAN δίνει 2 τύπους input/output εντολών.
- Στον πρώτο τύπο, ο προγραμματιστής πρέπει να δώσει ακριβώς τη μορφή που τα δεδομένα θα διαβαστούν από τον υπολογιστή ή θα αναγραφούν στην οθόνη
- Ο τρόπος αυτός λέγεται **FORMATTED input/output**
- θα τον δούμε μετά από μερικές διαλέξεις

Input/Output (συνέχεια)

- ❑ Στο δεύτερο τύπο υπάρχουν προκαθορισμένες μορφές που παρέχονται από τον compiler που συμφωνούν με τους τύπους των μεταβλητών που χρησιμοποιούνται στην εισαγωγή/έξοδο δεδομένων.

Αυτός ο τύπος ονομάζεται: `list-directed input/output`

- ❑ Οι απλούστερες μορφές έχουν την ακόλουθη σύνταξη:

`PRINT *, output-list`

όπου `output-list` είναι είτε μια μόνο μεταβλητή, ή σειρά μεταβλητών χωρισμένες με κόμμα.

Μπορεί να είναι μια σταθερά ή ακόμα και μια αριθμητική έκφραση

π.χ. `PRINT *, A, B, A*B`

- Μπορεί να χρησιμοποιηθεί επίσης χωρίς κάποια λίστα μεταβλητών:

`PRINT *`

οπότε και παράγει μια κενή σειρά σαν `output`

- Για κάθε εντολή `PRINT` μια καινούρια σειρά τυπώνεται στην εξωτερική μονάδα

Παράδειγμα

```
PROGRAM PROJEC
REAL HGHT0, HGHT, VELOC0, VELOC, ACCEL, TIME
ACCEL = -9.807
HGHT0 = 150.0
VELOC0 = 50.0
TIME = 5.0
HGHT = 0.5 * ACCEL * TIME ** 2 + VELOC0 * TIME + HGHT0
VELOC = ACCEL * TIME + VELOC0
PRINT *, 'AT TIME ', TIME, ' THE VERTICAL VELOCITY IS ', VELOC
PRINT *, 'AND THE HEIGHT IS ', HGHT
END
```

List-Directed Input

□ Η σύνταξη είναι: `READ *, input-list`

όπου `input-list` είναι μια μεταβλητή ή λίστα μεταβλητών χωρισμένες με κόμμα

- Μεταφέρει τιμές μεταβλητών από μια εξωτερική πηγή (πληκτρολόγιο ή αρχείο) και την ανάθεση (assignment) των τιμών αυτών στις μεταβλητές της λίστας.

`READ *, A, B`

Με την εκτέλεση της εντολής αυτής ο Η/Υ περιμένει να του δώσουμε τιμές για τις μεταβλητές αυτές από το πληκτρολόγιο

`25., 32.` (τιμές που δίνουμε από πληκτρολόγιο)

`PRINT *, 'A=', A, ' B=', B` Θα τυπώσει `A=25. , B=32.`

Οι μεταβλητές A και B έχουν αποκτήσει τις REAL τιμές 25. και 32.

List-Directed Input (συνέχεια)

❑ Ισχύουν οι ακόλουθοι κανόνες:

- Για κάθε εντολή READ διαβάζεται νέα σειρά δεδομένων
- Αν υπάρχουν λιγότερα δεδομένα από το πλήθος των μεταβλητών της λίστας, διαδοχικές input γραμμές διαβάζονται μέχρι όλες οι μεταβλητές της λίστας να αποκτήσουν τιμές
- Αν το πλήθος των τιμών που δίνεται είναι μεγαλύτερο από αυτό των μεταβλητών, οι επιπλέον τιμές αγνοούνται και μόνο οι πρώτες λαμβάνονται υπ' όψη
- Τα δεδομένα σε κάθε input γραμμή πρέπει να αποτελούνται από σταθερές και του ίδιου τύπου με αυτό της αντίστοιχης μεταβλητής της λίστας
- Ωστόσο ισχύει η σειρά ισχύος: Δηλαδή ένας ακέραιος μπορεί να δοθεί σε REAL ή DOUBLE PRECISION και ένας REAL σε DOUBLE PRECISION με την μετατροπή να συμβαίνει εσωτερικά
- Διαδοχικά δεδομένα σε μια input γραμμή πρέπει να διαχωρίζονται με κόμμα ή με ένα ή περισσότερα κενά (spaces)

Παράδειγμα

```
PROGRAM PROJEC
REAL HGHT0, HGHT, VELOC0, VELOC, ACCEL, TIME
ACCEL = -9.807
PRINT *, 'Enter the initial height and velocity'
READ *, HGHT0, VELOC0
PRINT *, 'Enter time at which to calculate height and velocity'
READ *, TIME
HGHT = 0.5 * ACCEL * TIME ** 2 + VELOC0 * TIME + HGHT0
VELOC = ACCEL * TIME + VELOC0
PRINT *, 'AT TIME ', TIME, ' THE VERTICAL VELOCITY IS ', VELOC
PRINT *, 'AND THE HEIGHT IS ', HGHT
END
```

- ❑ Σε κάθε READ εντολή η εκτέλεση του προγράμματος σταματά και ξανασυνεχίζει εφόσον ο σωστός αριθμός σταθερών έχει διαβαστεί.

Εντολές Ελέγχου και Λογικής

□ Τα assignment statements είναι αρκετά απλά.

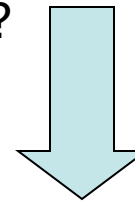
- Μεταφορά πληροφορίας από μια πράξη/σταθερά σε μεταβλητή που αντιστοιχεί σε θέση μνήμης του υπολογιστή
- Απλή δομή και η σειρά εκτέλεσης των εντολών ήταν προκαθορισμένη, μοναδική και ανεξάρτητη από τα μερικά αποτελέσματα

program trapezoid

```
c*****
c Calculate the height and area of a trapezoid
c Input the sides of the trapezoid A,B,C,D
c Output the height H and area A
c*****
  REAL A, B, C, D
  REAL HEIGHT, AREA
  REAL S, X
  READ *, A, B, C, D
c Calculate the height
  S = (A + B + C + D)/2.
  X = S*(S - A) * (S - B) * (S - C)
  HEIGHT = (2./((A+B)*S)*SQRT(X)
c Calculate the Area
  AREA = ((A+B)/2.) * HEIGHT
  PRINT *, HEIGHT, AREA
  END
```

□ Στην πράξη όμως ακόμη και τα πιο απλά προβλήματα απαιτούν μεγαλύτερη ευελιξία

□ Πως όμως αντιμετωπίζουμε προβλήματα όπως ο υπολογισμός της τετραγωνικής ρίζας ενός αρνητικού αριθμού χωρίς να σταματά η ροή εκτέλεσης του προγράμματος ?



Εντολές ελέγχου και λογικής
control statements

Εντολές ελέγχου – Control Statements

- Επιτρέπουν την παράλειψη ενός πλήθους εντολών (πηγαίνουμε από μια εντολή E1 σε μια άλλη E2 όχι επόμενη της πρώτης)
- Επιτρέπουν συγκρίσεις
Ανάλογα με το αποτέλεσμα, εκτελείται μια σειρά πράξεων ή κάποια άλλη πράξη
Ο Η/Υ αποκτά την δομή ανθρώπινης σκέψης (μέσω ενός προγράμματος)

- ❑ Οι κυριότερες εντολές ελέγχου και λογικής στην FORTRAN είναι:

Βασικό IF

IF ... THEN ... ELSEIF ... ELSE ...

DO

GO TO

CONTINUE, PAUSE, STOP, END

- ❑ Η μεταφορά του ελέγχου γίνεται με μια εντολή E1 στην οποία καθορίζουμε την εντολή E2 που πρέπει να εκτελέσει αμέσως μετά ο Η/Υ

➤ Η Εντολή E2 πρέπει να είναι εκτελέσιμη

Αν υπάρχουν υποπρογράμματα (επόμενο μάθημα) μέσα στο κύριο πρόγραμμα, οι E1 και E2 πρέπει να ανήκουν στο ίδιο πρόγραμμα ή υποπρόγραμμα

Βασικό IF

- Δίνει τη δυνατότητα σ' ένα πρόγραμμα να αποφασίζει ποια εντολή θα εκτελέσει στο επόμενο βήμα ανάλογα με την τιμή κάποιας μεταβλητής
- Η εντολή IF αυξάνει τις δυνατότητες ενός προγράμματος

PROGRAM IFIF

```

c=====
c THE BASIC IF STATEMENT
c =====
    INTEGER N
    PRINT *, 'Give me an integer'
    READ *, N
    IF (N.GT.0) THEN
        PRINT *, 'The number is positive'
    ENDIF
    IF (N.LT.0) THEN
        PRINT *, 'The number is negative'
    ENDIF
    STOP
    END
  
```

Η σύνταξη της εντολής είναι:

```

IF (logical expression) THEN
    block
ENDIF
  
```

Η λογική έκφραση στην παρένθεση είναι κάτι που μπορεί να ναι αληθές ή ψευδές

- Αν είναι **ψευδής** το πρόγραμμα συνεχίζει μετά την εντολή END IF
- Αν είναι **αληθής** τότε εκτελείται το group των εντολών μετά το IF.

Συνηθισμένη μορφή λογικής σχέσης είναι:

(Αριθ.σχέση) **Λογικός Τελεστής** (Αριθ. Σχέση)

π.χ. (A .GT. 0)

Βασικό IF (συνέχεια)

□ Οι τελεστές σχέσης που χρησιμοποιούνται είναι:

.GT. Greater Than ($>$)

.LT. Less Than ($<$)

.EQ. Equal ($=$)

.GE. Greater Equal (\geq)

.LE. Less Equal (\leq)

.NE. Not Equal (\neq)

Προσέξτε τις τελείες (.) που υπάρχουν πριν και μετά κάθε τελεστή σχέσης

Προσοχή: Τα σύμβολα δεν αναπαράστουν τους λογικούς τελεστές αλλά οι compilers τα θεωρούν σωστά

□ Οι λογικές σχέσεις συνδέονται ακόμα και με τους λογικούς τελεστές:

.AND. Λογικό AND

.EQV. (equivalent)

.OR. Λογικό OR

.NEQV. (not equivalent)

➤ Όταν χρησιμοποιούνται οι λογικοί τελεστές θα πρέπει κάθε λογική σχέση να βρίσκεται σε παρένθεση και το ίδιο η ολική λογική σχέση:

π.χ. IF ((A.GT.B) .AND. (B.LT.C)) THEN
 .
 .
 .
 ENDIF

Βασικό IF – Παραδείγματα (συνέχεια)

```
IF ( (A.GT.B) .OR. (B.EQ.C) ) THEN
```

```
  ⋮
  ⋮ } Block εντολών
  ⋮
```

Η ροή συνεχίζει στην εντολή μέσα στο
IF block αν είτε το $A > B$ ή το $B = C$

```
ENDIF
```

- ❑ Λογικές εκφράσεις μπορούν να αποκτήσουν αντίθετη σημασία αν προτάξουμε το **.NOT.**

(**.NOT.** (A .GT. B)) Η σχέση είναι ισοδύναμη με: $A < B$

- ❑ Δεν είναι απαραίτητο να υπάρχει κάποιο block εντολών μετά την εντολή IF
- ❑ **Καλή πρακτική:** Οι εντολές που περιέχονται μέσα σε ένα IF να γράφονται σε μετατοπισμένη στήλη προς τα δεξιά σε σχέση με την στήλη της αρχής του IF για εύκολη ανάγνωση του κώδικα δείχνοντας ότι το τμήμα αυτό του προγράμματος είναι ξεχωριστό.

```
IF (A .GT. B) THEN
```

```
    K = K + 1
```

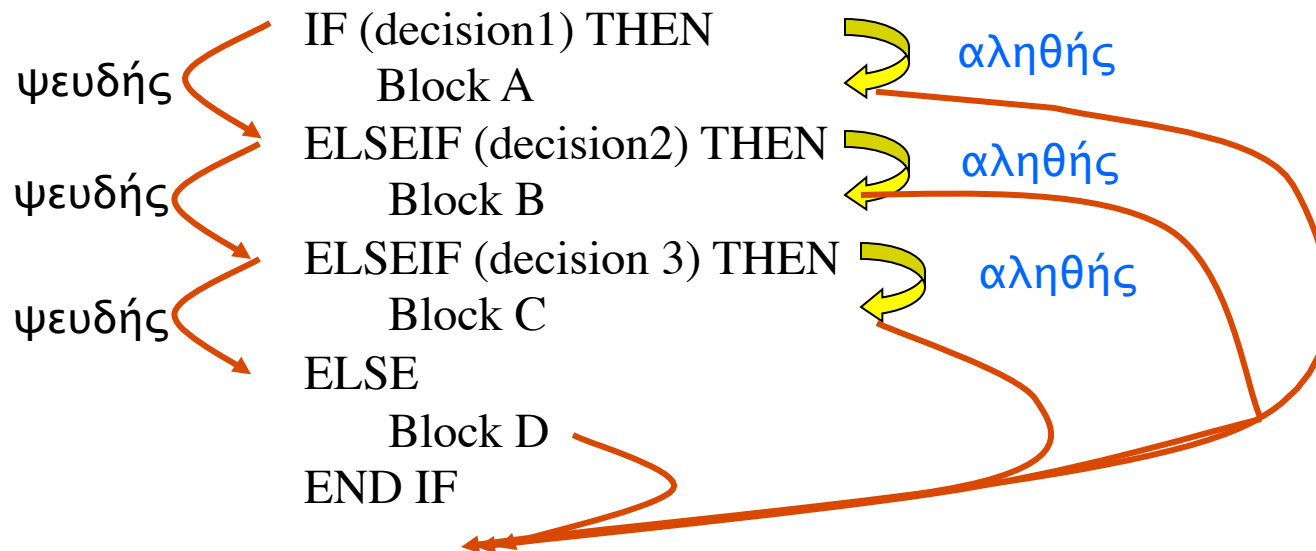
```
    L = L * K
```

```
ENDIF
```

IF...THEN ... ELSEIF ... THEN ... ELSE ...

□ Αυτή η μορφή χρησιμοποιείται για πολλαπλές αποφάσεις.

Αν καμιά από τις αποφάσεις δεν ικανοποιείται, τότε εκτελείται το block D των εντολών (μετά την εντολή ELSE).



IF χωρίς THEN και END IF

- Αν το block των εντολών που θέλουμε να εκτελέσουμε μετά ένα επιτυχημένο IF, αποτελείται μόνο από μια εντολή και δεν υπάρχουν άλλες αποφάσεις να παρθούν τότε μπορούμε να παραλείψουμε το THEN όπως επίσης και το ENDIF

```
IF (A .GT. B) PRINT *, 'is greater than B'
```

```
IF (NUM .LT. MIN) MIN= NUM
```

- Δεν μπορούμε να κάνουμε κάτι τέτοιο με THEN και ELSEIF δομές αποφάσεων.