Ελληνική Δημοκρατία
Τεχνολογικό Εκπαιδευτικό
Ίδρυμα Ηπείρου

Προγραμματισμός Ι

Ενότητα 2 : **Βασική σύνταξη προγράμματος pascal - Εντολές συνθήκης**

Τμήμα Μηχανικών Πληροφορικής Τ.Ε

Προγραμματισμός Ι

Ενότητα 2 : Βασική σύνταξη προγράμματος pascal - Εντολές συνθήκης

Αλέξανδρος Τζάλλας
Λέκτορας

Άρτα, 2015

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

Χρηματοδότηση

- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «**Εκπαίδευση και Δια Βίου Μάθηση**» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο ΤΕΙ Ηπείρου**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Σκοποί ενότητας

- Να αναλυθεί/περιγραφεί πλήρως η δομή του τμήματος δηλώσεων
- Να αναλυθεί πλήρως η συντακτική δομή και ο τρόπος χρήσης της εντολής ανάθεσης στην pascal
- Να αναλυθεί πλήρως η συντακτική δομή και ο τρόπος χρήσης των εντολών εισόδου/εξόδου Read/Readln και Write, Writeln
- Να περιγραφούν αναλυτικά οι δυνατότητες των εντολών συνθήκης.
- Να αναλυθεί με ακρίβεια η χρησιμότητα/σκοπός των εντολών συνθήκης.
- Να περιγραφούν οι συντακτικοί κανόνες των εντολών συνθήκης **if**, **φωλιασμένων εντολών if** και **case**.

Περιεχόμενα ενότητας

- Επικεφαλίδα Προγράμματος
- Τμήμα Δηλώσεων
- Τελεστές Ανάθεσης
- Τελεστές Εισόδου Read, Readln
- Τελεστές Ανάθεσης Write, Writeln
- Χρησιμότητα/Σκοπός Εντολών Συνθήκης
- Εντολή if
- Φωλιασμένες Εντολές if
- Εντολή Case

Δομή Προγράμματος Pascal

- Ένα Pascal πρόγραμμα αποτελείται:

1. Επικεφαλίδα του προγράμματος
2. Τμήμα των δηλώσεων (**declaration part**)
3. Τμήμα των προτάσεων (**statement part**)

Επικεφαλίδα Προγράμματος$_{1/2}$

- Η επικεφαλίδα του προγράμματος είναι μια πρόταση που αρχίζει με τη λέξη κλειδί **program ή PROGRAM**, η οποία ακολουθείται από ένα όνομα που χαρακτηρίζει όλο το πρόγραμμα και μια λίστα ονομάτων που σχετίζονται με τις συσκευές που θα χρησιμοποιηθούν από το πρόγραμμα για είσοδο και έξοδο δεδομένων
- Η επικεφαλίδα έχει την εξής **σύνταξη**:

PROGRAM Ονομα(Αρχείο 1, Αρχείο 2, …, Αρχείο N)

Επικεφαλίδα Προγράμματος$_{2/2}$

- **Εξηγήσεις:** "***Ονομα***" είναι το όνομα του προγράμματος και "***Αρχείο 1, Αρχείο 2, …, Αρχείο Ν***" είναι το όνομα των εξωτερικών αρχείων που θα χρησιμοποιήσει το πρόγραμμα
- Η σύνταξη των ονομάτων των αρχείων διέπεται από κανόνες του συγκεκριμένου υπολογιστικού συστήματος
- Τα δύο βασικά (**standard**) ονόματα αρχείων ***input*** και ***output*** στην λίστα των ονομάτων των εξωτερικών αρχείων αναφέρονται στις εξ' ορισμού περιφερειακές μονάδες εισόδου και εξόδου δεδομένων (όπως τερματικό, εκτυπωτής, αναγνώστης κλπ)
  - **program** Factorial (input, output);
  - **program** Second_Example (input, output, data, numbers);

Τμήμα Δηλώσεων

- Το τμήμα δηλώσεων περιέχει (δηλώνει) όλα τα στοιχεία που πρόκειται να χρησιμοποιηθούν στο πρόγραμμα, όπως μεταβλητές, σταθερές κλπ
- Το τμήμα αποτελείται από **πέντε περιοχές**:

- Ισχύουν δύο κανόνες για το τμήμα των δηλώσεων:
  - **Μοναδικότητα** (κάθε δήλωση είναι μοναδική)
  - **Ορισμός πριν την χρήση** (κάθε δήλωση πρέπει να ορισθεί πριν χρησιμοποιηθεί για πρώτη φορά)

Τμήμα Δηλώσεων-Περιοχή Επιγραφών

- Η περιοχή επιγραφών δηλώνεται με τη λέξη κλειδί **LABEL** και περιέχει τις επιγραφές (labels) που πρόκειται να χρησιμοποιηθούν από τις εντολές **GOTO** του προγράμματος

LABEL Επιγραφή 1, Επιγραφή 2, …, Επιγραφή N)

- Σαν επιγραφή μπορεί να χρησιμοποιηθεί ένας οποιοσδήποτε ακέραιος, χωρίς πρόσημο, έως **τέσσερα ψηφία.**
  - **label** 10, 9999, 80;

Τμήμα Δηλώσεων-Περιοχή Σταθερών$_{1/3}$

- Αν σε ένα πρόγραμμα χρησιμοποιήσουμε συχνά κάποια τιμή (π.χ. π=3,1415…) αντί να γράφουμε την τιμή αυτή κάθε φορά που χρειάζεται μπορούμε να της δώσουμε ένα συμβολικό όνομα (π.χ. pi) και να χρησιμοποιήσουμε αυτό το συμβολικό όνομα αντί της πραγματικής τιμής
- Αυτό προσφέρει:
  - Ευκολία στην ανάπτυξη αλλά και διόρθωση των προγραμμάτων
  - Αποφυγή λαθών, ειδικά στην περίπτωση που η τιμή που αντικαθιστούμε περιέχει πολλά ψηφία
  - Ευκολία αλλαγής της τιμής της, γιατί αρκεί τη τιμή του συμβολικού ονόματος και όχι κάθε εμφάνιση της τιμής στο πρόγραμμα

Τμήμα Δηλώσεων-Περιοχή Σταθερών$_{2/3}$

- Η περιοχή σταθερών δηλώνεται με τη λέξη **CONST** και περιέχει σταθερές που πρόκειται να χρησιμοποιηθούν στο πρόγραμμα
- Οι σταθερές δηλώνονται με το όνομά τους, το σύμβολο της ισότητας (=) και την τιμή τους

```
CONST Ονομα = Τιμή;
      :   =   :
      :   =   :
```

- Για τα ονόματα των σταθερών ισχύει ότι ισχύει γενικά για τα ονόματα της Pascal, δηλαδή είναι μοναδικά, δεν πρέπει να είναι κάποια λέξη-κλειδί της Pascal, δε μπορεί να χρησιμοποιηθεί για άλλο σκοπό
- Η τιμή της σταθερά καθορίζει τον τύπο της

Τμήμα Δηλώσεων-Περιοχή Σταθερών$_{3/3}$

- Π.χ. **const** meres=5;
  pi=3.14159;
  light=300000.0;
  Flag=true;
  Aster=‘*’;

- Η πρώτη σταθερά είναι τύπου integer, η δεύτερη και η τρίτη τύπου real, η τέταρτη τύπου boolean, και η πέμπτη τύπου char
- Υπάρχουν 3 σταθερές οι οποίες είναι προδηλωμένες στην Pascal και μπορούν να χρησιμοποιούνται χωρίς να δηλώνονται (είναι οι δύο boolean τιμές false και true και η τιμή του μεγαλύτερου ακεραίου, maxint)

Τμήμα Δηλώσεων-Περιοχή Τύπων$_{1/3}$

- Εκτός από τους 4 στοιχειώδεις τύπους της Pascal (integer, real, boolean και char) που είναι προδηλωμένοι στην Pascal, ο προγραμματιστής μπορεί να ορίσει και **νέους τύπους δεδομένων**
- Η περιοχή τύπων δηλώνεται με τη λέξη κλειδί **Type** και περιέχει τους τύπους δεδομένων, που ορίζει ο προγραμματιστής
- Ένας τύπος δεδομένων ορίζεται με ένα όνομα, το σύμβολο της ισότητας (=) και τον προσδιορισμό του τύπου

Τμήμα Δηλώσεων-Περιοχή Τύπων$_{2/3}$

```
TYPE Ονομα = Τύπος Δεδομένων;
     :     =          :
     :     =          :
```

- Για τα ονόματα των τύπων ισχύει ότι ισχύει για τα ονόματα της Pascal
- Οι τύποι δεδομένων που ορίζει ο χρήστης μπορούν να χρησιμοποιηθούν όπως ακριβώς χρησιμοποιούνται και οι στοιχειώδεις τύποι δεδομένων της Pascal, και μπορεί να είναι απαριθμητή, υποπεριοχής, δομημένοι, ή δείκτη

Τμήμα Δηλώσεων-Περιοχή Τύπων$_{3/3}$

- Π.χ. **type** Day=(Mon,Tues,Wed,Thur,Fri,Sat,Sun);

  DayNum=1…31;

  pin = array[1…10] of integer;

  num = record

  x: integer; y:real;

Τμήμα Δηλώσεων-Περιοχή Μεταβλητών$_{1/3}$

- Η περιοχή μεταβλητών δηλώνεται με τη λέξη κλειδί **VAR** και περιέχει τις μεταβλητές που πρόκειται να χρησιμοποιηθούν στο πρόγραμμα
- Μια μεταβλητή δηλώνεται με το όνομά της, το σύμβολο (:) και δίπλα τον τύπο των τιμών της

```
VAR Ονομα = Τύπος Δεδομένων;
        :     =       :
        :     =       :
```

- Για τα ονόματα των μεταβλητών ισχύει ότι ισχύει γενικά για τα ονόματα της Pascal

Τμήμα Δηλώσεων-Περιοχή Μεταβλητών$_{2/3}$

- Θεωρώντας ότι ισχύουν οι τύποι δεδομένων που ορίστηκαν προηγουμένως μπορούμε να γράψουμε:

```
var  letter: char;
     ok: boolean;
     aktina: real;
     mikos: integer;
     Weekday: Day;
     Date: DayNum;
```

Τμήμα Δηλώσεων-Περιοχή Μεταβλητών$_{3/3}$

- **Σημείωση:** όταν περισσότερες από μια μεταβλητές είναι του ίδιου τύπου μπορούν να δηλωθούν μαζί:

```
var  x: integer;
     y: integer;
     z: integer;
```

- Μπορούμε να γράψουμε:

```
var  x,y,z: integer;
```

Τμήμα Δηλώσεων
Περιοχή Υποπρογραμματων

- Στην περιοχή αυτή ορίζονται 2 είδη υποπρογραμματων:
  - οι διαδικασίες (procedures)
  - οι συναρτήσεις (functions)

Τμήμα Δηλώσεων-Περιοχή Προτάσεων$_{1/2}$

- Το τμήμα προτάσεων περιέχει τις εκτελέσιμες εντολές του προγράμματος
- Αρχίζει με τη λέξη-κλειδί **begin** και τελειώνει με τη λέξη-κλειδί **end.**
- Οι εκτελέσιμες προτάσεις ορίζουν τις λειτουργίες που πρόκειται να εκτελεστούν πάνω στα στοιχεία, που δηλώνονται στο τμήμα των δηλώσεων και περιέχουν εντολές της Pascal, κλήσεις ενσωματωμένων συναρτήσεων και διαδικασιών της Pascal, κλήσεις συναρτήσεων και διαδικασιών του χρήστη, κλπ

Τμήμα Δηλώσεων-Περιοχή Προτάσεων$_{2/2}$

- Η Pascal είναι σειριακή γλώσσα προγραμματισμού
- Αυτό σημαίνει ότι οι προτάσεις του προγράμματος εκτελούνται σειριακά, η μία μετά την άλλη με τη σειρά που είναι δηλωμένες στο πρόγραμμα και όχι ταυτόχρονα
- Ισχύουν τα ακόλουθα:
  - Το τέλος των δηλώσεων και των προτάσεων δηλώνεται με το διαχωριστικό χαρακτήρα (;)
  - Ο διαχωριστικό χαρακτήρας (;) πριν από τη λέξη-κλειδί end είναι προαιρετικός
  - Στο τέλος του προγράμματος, δηλαδή μετά το end βάζουμε την τελεία (.)

Δομή Προγράμματος Pascal

```
program όνομα_προγράμματος(αρχείο 1,αρχείο2,…,αρχείοN;
label   επιγραφή1,επιγραφή2,…,επιγραφήN ;
const  δήλωση σταθεράς;
                              :          :
          δήλωση σταθεράς;
type    δήλωση τύπου;
                              :          :
          δήλωση τύπου;
var      δήλωση μεταβλητής;
                              :          :
           δήλωση μεταβλητής;
begin
εκτελέσιμη πρόταση;
                :          :
                :          :
εκτελέσιμη πρόταση;
end.
```

Παράδειγμα

- Να γραφεί πρόγραμμα το οποίο να διαβάζει δύο αριθμούς και να τυπώνει το άθροισμα και τη διαφορά τους

```
Program sum_diff(input,output);
var x,y,sum,dif: integer;
begin
      read(x,y);
      sum:=x+y;
      dif:=x-y;
      writeln('Athroisma=', sum);
      writeln('Diafora=', dif);
end.
```

Επικεφαλίδα

Δηλώσεις

Προτάσεις

Τελεστής Ανάθεσης
Εντολές Εισόδου/Εξόδου$_{1/2}$

- Σκοπός κάθε προγράμματος είναι να διαχειρίζεται δεδομένα που τα παρέχει ο χρήστης και να παράγει τα ζητούμενα αποτελέσματα
- Κάθε γλώσσα προγραμματισμού παρέχει τα μέσα επικοινωνίας των προγραμμάτων με το χρήστη
- Στην Pascal για την είσοδο και την έξοδο δεδομένων δε θεωρείται κάποια συγκεκριμένη συσκευή όπως πληκτρολόγιο, εκτυπωτής κ.α. αλλά το πρόγραμμα αντιμετωπίζεται σαν μια διαδικασία που παίρνει πληροφορίες (δεδομένα) από μια μονάδα εισόδου και μεταφέρει πληροφορίες (αποτελέσματα) σε μια μονάδα εξόδου

Τελεστής Ανάθεσης Εντολές Εισόδου/Εξόδου$_{2/2}$

- Στην επικεφαλίδα του προγράμματος γίνεται αναφορά στις μονάδες εισόδου/εξόδου με τη μορφή **program first(input, output)** που δηλώνει το πρόγραμμα που κάνει χρήση των βασικών μονάδων εισόδου και εξόδου, όπως αυτές καθορίζονται από το συγκεκριμένο υπολογιστικό σύστημα στο οποίο εκτελείται το Pascal πρόγραμμα

Τελεστής Ανάθεσης$_{1/3}$

- **Σκοπός:** Ανάθεσης της τιμής μια έκφρασης σε μια μεταβλητή

Ονομα Μεταβλητής := Έκφραση

- **Εξηγήσεις:** Υπολογίζεται η τιμή της έκφρασης και ανατίθεται (καταχωρείται) στη μεταβλητή

a:=5+2; *Έχει σαν αποτέλεσμα την ανάθεση της τιμής 7 στη μεταβλητή a*

b:=4.1*2; *Θέτει στη μεταβλητή b την τιμή 8.2;*

c:='F'; *Θέτει στη μεταβλητή c το χαρακτήρα 'F';*

x:=3>1; *Θέτει στη μεταβλητή x την τιμή true;*

Τελεστής Ανάθεσης$_{2/3}$

- **Κανόνες:**
  - Ο τύπος της τιμής της έκφρασης πρέπει να είναι ίδιος με τον τύπο της μεταβλητής, πχ **var** a : integer, b: real, c : char, x : boolean;
  - Εξαίρεση στον προηγούμενο κανόνα η δυνατότητα να θέτουμε ακέραιες τιμές (integer) σε μεταβλητές τύπου real, πχ **αν a : real τότε είναι αποδεκτές: a:=4; a:=5+4*2;**
  - Μπορούμε να χρησιμοποιήσουμε την ίδια μεταβλητή ταυτόχρονα στο αριστερό μέρος του τελεστή ανάθεσης και μέσα στην έκφραση, στο δεξιό μέρος του τελεστή, πχ **αν a είναι 5 τότε η εκτέλεση της πρότασης: a:= a +1; έχει σαν αποτέλεσμα την ανάθεση στην a της τιμής 6**

Τελεστής Ανάθεσης$_{3/3}$

- **Παράδειγμα:**

```
program example;
var     a,b: integer;
        flag: boolean;
        res:real;
begin
    a:=234;
    b:=a*5-a div 2;
    flag:=(a>100) and (b>1000);
    res:=a/b;
end.
```

Εντολή Read (Εντολή Εισόδου)$_{1/9}$

- **Σκοπός:** Εισαγωγή (ανάγνωση) τιμών από τη μονάδα εισόδου και ανάθεση σε μεταβλητές του προγράμματος

READ(INPUT,V1,V2,…Vn) ή READ(V1,V2,…Vn)

- **Εξηγήσεις:** Με την εντολή αυτή εισάγονται τιμές από τη μονάδα εισόδου και αποθηκεύονται στις μεταβλητές **V1…Vn** του προγράμματος με τη σειρά που είναι διατεταγμένες
  - Αν η μονάδα εισόδου είναι το πληκτρολόγιο, τότε μόλις εκτελεστεί η εντολή, η εκτέλεση του προγράμματος σταματά και περιμένει δεδομένα από το πληκτρολόγιο για να τα τοποθετήσει στις μεταβλητές και να συνεχίσει την εκτέλεση του υπόλοιπου προγράμματος

Εντολή Read (Εντολή Εισόδου)$_{2/9}$

Παράδειγμα:

- **read(x);** *Με δεδομένο 3, αναθέτει στη μεταβλητή x την τιμή 3*
- **read(a,b,c)***; Με δεδομένα από το πληκτρολόγιο: 5 7 83 έχει ως αποτέλεσμα την ανάθεση των τιμών 5 7 83 στις μεταβλητές a, b και c αντίστοιχα*

Παρατηρήσεις:

- Η εντολή read(v1, v2, …, vn) είναι ισοδύναμη με την ακολουθία εντολών: read(v1) ; read(v2); …;read(vn);
- Τα δεδομένα εισόδου ενός προγράμματος αντιμετωπίζονται σα μια συνεχόμενη ροη π.χ. οι εντολές: read(a) ; read(b,c,d); read(x); με δεδομένα 2 5 8 1 9 έχουν σαν αποτέλεσμα την ανάθεση των τιμών 2 5 8 1 και 9 στις μεταβλητές a, b, c, και x αντίστοιχα

Εντολή Read (Εντολή Εισόδου)$_{3/9}$

- Τα δεδομένα που παρέχει μια μονάδα εισόδου στο πρόγραμμα είναι συνήθως πεπερασμένου μήκους
- Το τέλος των δεδομένων δηλώνεται με ένα ειδικό χαρακτήρα ελέγχου: **end-of-file character** και συμβολίζεται <eof>
- Για την ανίχνευση αυτού του χαρακτήρα από τα προγράμματα, η Pascal παρέχει μια λογική (τύπου boolean) συνάρτηση**:** **EOF(INPUT)** ή **EOF** της οποίας η τιμή είναι αληθής στο τέλος των δεδομένων και ψευδής σε οποιαδήποτε άλλη περίπτωση
- Τα δεδομένα εισόδου να είναι οργανωμένα σε σειρές [**end-of-line character, EOLN(INPUT) ή EOLN**]

Εντολή Read (Εντολή Εισόδου)$_{4/9}$

- **Κανόνες:**
  - Ο τύπος των δεδομένων πρέπει να συμφωνεί με τον τύπων των μεταβλητών και μπορεί να είναι μόνο τύπου integer, real ή char ή ενός τύπου που είναι υποπεριοχή των τύπων integer ή char
  - Αν τα δεδομένα στη τρέχουσα σειρά εισόδου είναι λιγότερα από τις μεταβλητές που αναφέρονται στην εντολή read τότε η ανάγνωση δεδομένων συνεχίζει στην επομένη σειρά
  - Λάθος εκδηλώνεται, όταν διαβαστεί το τελευταίο δεδομένο εισόδου και υπάρχουν μεταβλητές στην εντολή read που δεν έχουν πάρει τιμή

Εντολή Read (Εντολή Εισόδου)$_{5/9}$

- **Κανόνες:**
  - Ανάγνωση αριθμητικών δεδομένων:
    - Τα αριθμητικά δεδομένα πρέπει να χωρίζονται μεταξύ τους με έναν τουλάχιστον κενό χαρακτήρα
    - Κατά την ανάγνωση αριθμητικών δεδομένων (ακέραιων ή πραγματικών), οι κενοί χαρακτήρες που προηγούνται ή βρίσκονται μεταξύ των αριθμών αγνοούνται
    - Ο χαρακτήρας ελέγχου <eoln> κατά την ανάγνωση αριθμητικών αγνοείται και η ανάγνωση συνεχίζει στην επόμενη σειρά

Εντολή Read (Εντολή Εισόδου)$_{6/9}$

- **Κανόνες:**
  - Ανάγνωση χαρακτήρων:
    - Κατά την ανάγνωση χαρακτήρων λαμβάνονται υπόψη τόσο οι κενοί χαρακτήρες όσο και οι χαρακτήρες ελέγχου <eoln>
    - Ο χαρακτήρας ελέγχου <eoln> κατά την ανάγνωση χαρακτήρων, διαβάζεται, ανατίθεται στην αντίστοιχη μεταβλητή της εντολής read και η ανάγνωση συνεχίζει στην επόμενη σειρά

Εντολή Read (Εντολή Εισόδου)$_{6/9}$

- **Κανόνες:**
- Ανάγνωση αριθμών και χαρακτήρων
  - Ιδιαίτερη προσοχή απαιτείται όταν η σειρά εισόδου περιέχει αριθμητικά δεδομένα και χαρακτήρες

π.χ. **var** r:real;
a, b, c:char;
x:integer;
η εντολή read(r,a,b,c,x)

a) 34.1r@w89 r=34.1 a='f' b='@' c='w' x=89
b) 1E10EH 897K r=2E10 a='E' b='H' c=' ' x=897
c) +23E-3D5634 r=23E-3 a='D' b='5' c='6' x=34
d) 45F.1 r=45.0 a='F' b='.' c='1' x=62
62

Εντολή Read (Εντολή Εισόδου)$_{7/9}$

- **Σημείωση:**
  - σε πολλές υλοποιήσεις της Pascal το τέλος ενός αριθμητικού δεδομένου δηλώνεται με ένα κενό χαρακτήρα
  - Οποιοσδήποτε άλλος χαρακτήρας προκαλεί λάθος
  - Πρόβλημα δημιουργείται όταν υπάρχει εναλλαγή αριθμητικών και μη δεδομένων

Εντολή Read (Εντολή Εισόδου)$_{8/9}$

- **Σημείωση:**
  - Το πρόβλημα λύνεται γράφοντας ένα κενό χαρακτήρα μετά από κάθε αριθμητικό δεδομένο και διαβάζοντας τις μεταβλητές τύπο χαρακτήρα δύο φορές, μία για να διαβαστεί το κενό και μία για να διαβαστεί το πραγματικό δεδομένο, **π.χ.** 34 7 67 διαβάζοντας την εντολή read(x,a,a,y) με x,y,:integer και a:char
  - Αρχικά η μεταβλητή a παίρνει τιμή ‘ ’ και στην συνέχεια την τιμή ‘f’
  - Παρατηρούμε ότι για τα ίδια δεδομένα η εντολή read(x,a,y) προκαλεί λάθος γιατί προσπαθεί να διαβάσει την τιμή της ακεραίας μεταβλητής y αρχίζοντας από το χαρακτήρα ‘f’

Εντολή Read (Εντολή Εισόδου)$_{9/9}$

- **Παράδειγμα:**

```
program reading(input);
var     x,y: integer;
              a,b,c:real;
begin
      read(x);
      y:=x mod 2;
      read(a,b);
      c:=a/b;
end.
```

Εντολή Readln$_{1/2}$

- **Σκοπός:** Η εντολή readln (read until end of line) αποτελεί μια ειδική μορφή της εντολής read που επιτρέπει περισσότερο έλεγχο στα δεδομένα που είναι οργανωμένες σε σειρές

READLN(INPUT)	ή READLN
READLN(INPUT,V1,V2,…Vn)	ή READ(V1,V2,…Vn)

- **Εξηγήσεις:** η εντολή αυτή προκαλεί παράλειψη όλων των δεδομένων που έμειναν αχρησιμοποίητα στη τρέχουσα σειρά δεδομένων συμπεριλαμβανομένου και του χαρακτήρα <eoln>. Αυτό σημαίνει ότι η επόμενη εισαγωγή δεδομένου θα γίνει από την αρχή της επόμενης σειράς των δεδομένων

Εντολή Readln$_{2/2}$

- **Παράδειγμα:** Έστω τα δεδομένα:

  23 4 5

  10 8 79 6

  84 91 30

  και οι εντολές:

  readln(a,b);readln(c);readln(d,e) αποθηκεύουν a=23, b=4, c=10, d=84, e=91’

Εντολή Write (Εντολή Εξόδου)$_{1/7}$

- **Σκοπός:** Μεταφορά (εγγραφή) δεδομένων στη μονάδα εξόδου

WRITE(OUTPUT,V1,V2,…Vn) ή WRITE(V1,V2,…Vn)

- **Εξηγήσεις:** η εντολή αυτή μεταφέρει τις τιμές των παραμέτρων της, στη βασική μονάδα εξόδου (π.χ. οθόνη) με την ίδια διάταξη που αναφέρονται στην εντολή

Εντολή Write (Εντολή Εξόδου)$_{2/7}$

- **Παράδειγμα:** **var** c:char;
  a, b:integer;
  k:boolean;

με τιμές c:='M';a:=4;,b:=15;k:=false;

Τότε:

write(a,b,c,k); → 4 15 M FALSE

write(a,b,a+b,2*b-a,a<b); → 4 15 19 26 TRUE

write('a=', a, 'b=',b, 'Sum=',a+b,'Letter=c'); → a=4 b=15, Sum=19
Letter=M

Εντολή Write (Εντολή Εξόδου)$_{3/7}$

- **Παράδειγμα:**

```
program print2(input,output);
var   x,y: integer;
       a,b,c,d,e:real;
begin
      read(x);
      y:=x mod 2;
      write(y)
      read(a,b);
      c:=a+b;
      d:=a/b;
      e:=a*b;
      write(a,b,c,d,e);
end.
```

Εντολή Write (Εντολή Εξόδου)$_{4/7}$

- **Παράδειγμα:**

```
program print1(input,output);
var  x: integer;
     a,b:real;
begin
     read(x);
     write(x mod 2); -> or write('apotelesma=', x mod 2);
     read(a,b);
     write(a,b,a+b,a/b,a*b);-> or write(a,b,'sum=',a+b,'div=',a/b,'mul=',a*b)
end.
```

Εντολή Write (Εντολή Εξόδου)$_{5/7}$

- **Παρατηρήσεις:**
  - Μπροστά από τους αριθμούς (στη μονάδα εξόδου) το (-) αν είναι αρνητικοί ή ο κενός χαρακτήρας αν είναι θετικοί. Τα δεδομένα τύπου real τυπώνονται σαν αριθμοί κινητής υποδιαστολής (δηλ. x.xx Exx)

αν r=-324.586 τότε το αποτέλεσμα της εντολής write(r) είναι - 3.2158600000E+02

  - Μπορούμε να δηλώσουμε το μήκος (σε χαρακτήρες) της τιμής που τυπώνεται βάζοντας δίπλα από κάθε παράμετρο δύο τελείς (:) και μια ακέραια τιμή που δηλώνει το επιθυμητό μήκος

αν a=-5, b=true και r=-324.586 τότε:

write(a,b) ; → **-5 TRUE**

write(a:3,b:6); → **-5 TRUE**

write(r: 10); → **-3.246E+02**

Εντολή Write (Εντολή Εξόδου)$_{6/7}$

- **Παρατηρήσεις:**
  - Επίσης μπορούμε να δηλώσουμε και τον αριθμό των δεκαδικών ψηφίων που θέλουμε να τυπωθούν μετά από το δεκαδικό σημείο, θέτοντα πάλι (:) και μια δεύτερη ακέραια τιμή που δηλώνει τον επιθυμητό αριθμό των δεκαδικών

    π.χ. αν r=-324.586, τότε: write(r:6:1); -> -324.6

Εντολή Write (Εντολή Εξόδου)$_{7/7}$

- **Παράδειγμα:**

- Να γραφεί πρόγραμμα το οποίο να διαβάζει την ακτίνα ενός κύκλου (πραγματικός αριθμός) και να τυπώνει την ακτίνα, την περιφέρεια και το εμβαδόν του κύκλου
  - Για δεδομένο 3 τότε:
    3.0000000000E+00
    1.884955590oE+01
    2.8274333850E+01
  - write(r:8,perif:10,emb:10) τότε:
    3.0E+00
    1.885E+01
    2.827E+01

```
program  circle(input,output); ->circle;
const  pi=3.1459265;
var      r,per,emb:real;
begin
      read(r);
      per:=2*pi*r;
      emb:=pi*r*r;  → or emb:=pi*sqr(r);
      writeln(r,per,emb);
end.
```

Εντολή Writeln$_{1/2}$

- **Σκοπός:** writeln (write until line) αποτελεί μια ειδική μορφή της εντολής write που επιτρέπει την οργάνωση των δεδομένων εξόδου (αποτελεσμάτων) σε σειρές

WRITELN(OUTPUT)	ή	WRITELN
WRITELN(INPUT,V1,V2,…Vn)	ή	WRITE(V1,V2,…Vn)

- **Εξηγήσεις:** Η εντολή αυτή μεταφέρει ένα χαρακτήρα ελέγχου <eoln> στη συσκευή εξόδου προκαλώντας έτσι τη δημιουργία μιας νέας (κενής) σειράς

- **Παράδειγμα:** **write(5+2,5-2,5*2)** → **7 3 10**

  write(5 DIV 2,5 MOD 2); → **2 1**

Εντολή Writeln$_{2/2}$

- **Παράδειγμα:**

Ποια είναι η εκτύπωση του ακόλουθου προγράμματος, όταν τα δεδομένα έχουν την μορφή:

9 7 6

4 5

```
program  ektiposi(input,output);
var      x,y,d,m:integer;
begin
      readln(x);
      readln(y);
      d:=x div y;
      writeln('Div=',d);
      writeln;
      z:=x mod y;
      write('Mod=',m);
end.
```

Ασκήσεις$_{1/5}$

Ποιες αντικαταστάσεις είναι σωστές και ποιες όχι;

```
program  askisi1(input,output);
Const mark='@'
          price=456.89;
var       x,y,z:integer;
          a,b:real;
          k,m:char;
          flag:boolean;
begin
          x:=245;                                  Σωστό
          y:=x+45+56*6;                            Σωστό
          z:=x div y+5*(x mod y);                  Σωστό
          z:=z+a div b;                            Λάθος
          a:=x+y+z;                                Σωστό
          b:=x/y;                                  Σωστό
          z:=x/y;                                  Λάθος
          y:=23*x*180;                             Σωστό
          b:=23*x*180;                             Σωστό
          k:='1';                                  Σωστό
          x:=x+k;                                  Λάθος
          m:='ab';                                 Λάθος
          flag:=(x>y) and (a<b) or (k<>m);         Σωστό
end.
```

Ασκήσεις$_{2/5}$

Να γραφεί ένα πρόγραμμα το οποίο να διαβάζει το κόστος ενός είδους, το ποσό που έδωσε ο πελάτη και να τυπώνει το κόστος, το ποσό και τα ρέστα με την εξής μορφή:

cost $xxx.xx

amount $xxx.xx

change $xxx.xx

```
program  pelatis(input,output);
var cost,amount,change:real;
begin
      write('cost='); readln(cost);
      write('amount='); readln(amount);
      writeln('cost          $',cost:6:2);
      writeln('amount      $',amount:6:2);
      writeln('change     $',(amount-cost):6:2);
end.
```

Ασκήσεις$_{3/5}$

Να γραφεί πρόγραμμα που να διαβάζει τις ώρες, τα λεπτά και τα δευτερόλεπτα και να τυπώνει το συνολικό αριθμό δευτερολέπτων

```
program  seconds(input,output);
var      h,m,s:integer;
          total:real;
begin
      write('hours='); readln(h);
      write('minutes='); readln(m);
      write('seconds='); readln(s);
      total:=h*3600+m*60+s;
      writeln('total=',total:8:2);
end.
```

Ασκήσεις$_{4/5}$

Να γραφεί πρόγραμμα το οποίο διαβάζει 5 αριθμούς και να τυπώνει το άθροισμα και τον μέσο όρο

```
program  askisi4;
var      a,b,c,d,e,sum:integer;
          mo:real;
begin
       write('dwse arithmo 1='); readln(a);
       write('dwse arithmo 2 ='); readln(b);
       write('dwse arithmo 3 ='); readln(c);
       write('dwse arithmo 4 ='); readln(d);
       write('dwse arithmo 5 ='); readln(e);
       sum:=(a+b+c+d+e);
       mo:=sum/5;
       writeln('to athroisma einai',sum);
       writeln('o mesos oros einai',mo);
end.
```

Ασκήσεις$_{5/5}$

Να γραφεί πρόγραμμα που να υπολογίζει το εμβαδόν ενός τριγώνου με πλευρές a,b,c. Το εμβαδόν δίνεται από τον τύπο:

Εμβαδόν= $\sqrt{S(S-A)(S-B)(S-C)}$

όπου: $2S = A+B+C$

```
program  embado;
var a,b,c,s,e:real;
begin
      write('dwse tis pleyres:');
      readln(a,b,c);
      s:=(a+b+c)/2;
      e:=sqrt(s*(s-a)*(s-b)*(s-c));
      writeln('embadon=',e);
end.
```

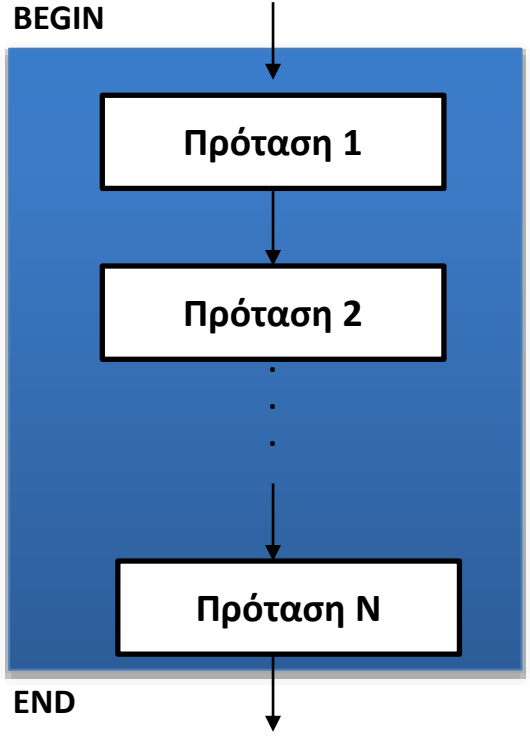
Εντολές συνθήκης

- Οι προτάσεις ενός προγράμματος εκτελούνται σειριακά, αρχίζοντας από την πρόταση του κύριου προγράμματος και συνεχίζοντας με τις επόμενες προτάσεις έως ότου εκτελεστεί και η τελευταία
- Υπάρχουν περιπτώσεις όπου απαιτείται η εκτέλεση μια πρότασης ανάλογα με την ισχύ ή όχι κάποιας συνθήκης ή η επιλογή και η εκτέλεση μιας πρότασης ανάμεσα από πολλές άλλες προτάσεις ανάλογα με την τιμή μιας έκφρασης
- Η Pascal υποστηρίζει για το σκοπό αυτό δύο εντολές συνθήκης, την **if** και την **case**

Εντολή Συνθήκης = Εντολή **if**
Εντολή **case**

Σύνθετες Προτάσεις

- Υπάρχουν περιπτώσεις που απαιτείται η ισχύς μια εντολής να μην αναφέρεται σε μια μόνο πρόταση αλλά σε μια ακολουθία προτάσεων που πρέπει να εκτελεστούν
- Σύνθετη πρόταση ονομάζεται μια ακολουθία προτάσεων που περικλείεται μεταξύ των λέξεων-κλειδιών **begin, end** και χωρίζονται με το διαχωριστικό (**;**)
- Εκτέλεση μιας σύνθετης πρότασης σημαίνει εκτέλεση των περιεχομένων προτάσεων με τη σειρά που είναι γραμμένες



Εντολή if$_{1/3}$

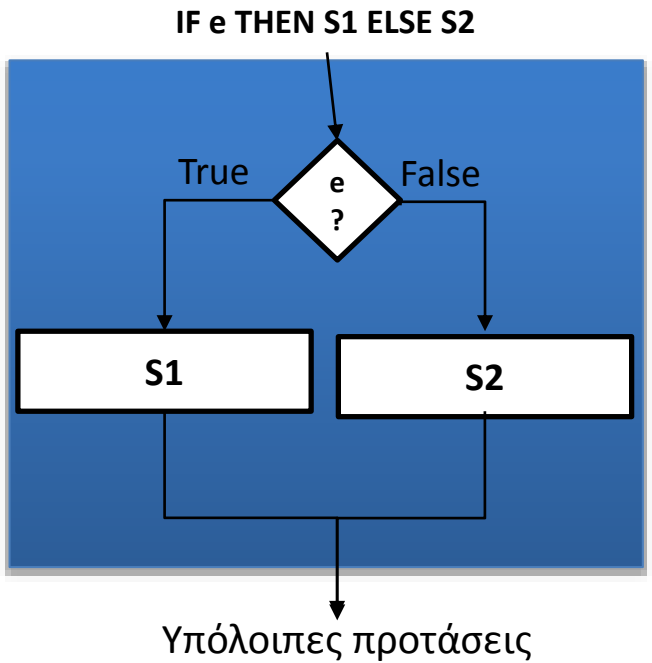
- **Σκοπός:** Εκτέλεση μια πρότασης αν ισχύει κάποια συνθήκη ή επιλογή ανάμεσα από δύο προτάσεις

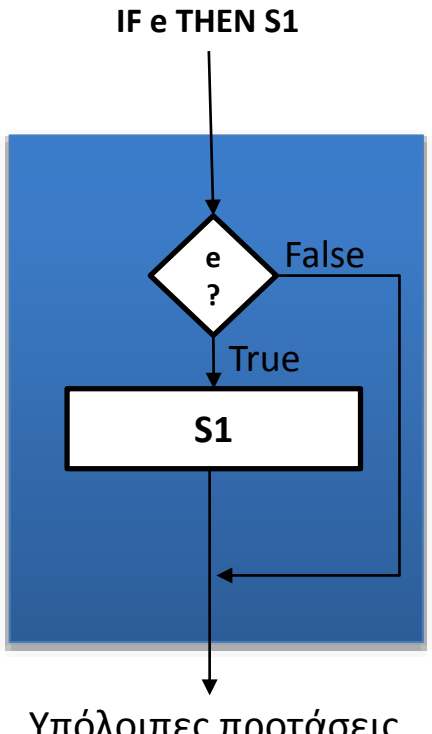
IF Λογική Έκφραση**THEN** Πρόταση 1
[**ELSE** Πρόταση 2]

- **Εξηγήσεις:** Στην πλήρη μορφή της εντολής (δηλαδή όταν υπάρχει τμήμα **then** και τμήμα **else**) αρχικά υπολογίζεται η τιμή της λογικής έκφρασης και αν είναι αληθής (**true**) τότε (**then**) εκτελείται η **Πρόταση 1**, αλλιώς (**else**) εκτελείται η **Πρόταση 2**
- Αν δεν υπάρχει **else** στην εντολή, τότε αν η τιμή της λογικής έκφρασης είναι **true** εκτελείται η πρόταση του τμήματος then ενώ αν είναι **false** η εκτέλεση συνεχίζεται με τις προτάσεις που υπάρχουν μετά την εντολή **if**

Εντολή $if_{2/3}$

Αν **e** μια Λογική Έκφραση και **S, S1, S2** είναι εκτελέσιμες προτάσεις τότε η ροή εκτέλεσης των προτάσεων: **if e then s1 else s2**





Εντολή if$_{3/3}$

- **Παράδειγμα:**

Να γραφεί πρόγραμμα το οποίο να διαβάζει δύο αριθμούς και να τυπώνει το μεγαλύτερο και το μικρότερο

```
program  min_max1;
var      k,m:real;
begin
        write('give 2 numbers:');
        readln(k,m);
        If k>m then
        writeln('max=',k,'min=',m)
                  else
        writeln('max=',m,'min=',k)
end.
```

Φωλιασμένες Εντολές if 1/5

- Οι προτάσεις που υπάρχουν στο τμήμα **then** ή στο τμήμα **else** μιας εντολής **if** μπορούν να περιέχουν επίσης μια άλλη εντολή **if**, κοκ. Για παράδειγμα:

```
if x>0 then writeln('positive')
        else if x<0 then writeln('negative')
                else writeln('zero')
```

- Ο γενικός κανόνας που ισχύει είναι ότι κάθε else αντιστοιχεί στο πιο πρόσφατο if. Για παράδειγμα:

```
if (1) x>0 then (1) writeln('positive')
        else  (1) if (2) x<0 then (2) writeln('negative')
                else  (2) writeln('zero')
```

Φωλιασμένες Εντολές if$_{2/5}$

- Σε περιπτώσεις όπου λείπουν κάποια τμήματα else ή υπάρχουν πολλές φωλιασμένες εντολές if ή γενικότερα όταν υπάρχει ασάφεια αντιστοίχισης, σχηματίζουμε σύνθετες προτάσεις χρησιμοποιώντας begin, end. Για παράδειγμα:

```
if e1 then if e2 then S1 else S2
```

- Σύμφωνα με το γενικό κανόνα το else S2 αντιστοιχεί στο if με τη λογική έκφραση e2. Η πρόταση γράφεται ισοδύναμα:

```
if e1 then begin

            if e2 then S1 else S2

end
```

Φωλιασμένες Εντολές if$_{3/5}$

- Αν θέλαμε το else S2 να αντιστοιχεί στο πρώτο if με τη λογική έκφραση e1 θα έπρεπε να γράψουμε την πρόταση ως εξής:

```
if e1 then begin
        if e2 then S1
        end
        else S2
```

Φωλιασμένες Εντολές if$_{4/5}$

- **Παράδειγμα:**

Να γραφεί πρόγραμμα το οποίο να διαβάζει δύο αριθμούς και να τυπώνει το μεγαλύτερο και το μικρότερο ή ότι είναι ίσοι

```
program  min_max2;
var      k,m:real;
begin
        write('give 2 numbers:');
        readln(k,m);
        If k>m then
        writeln('max=',k,'min=',m)
                  else if k<m then
        writeln('max=',m,'min=',k)
                  else writeln(k,m,'equal')
end.
```

Φωλιασμένες Εντολές if$_{5/5}$

- **Παράδειγμα:**

Λύση (με διερεύνηση) της πρωτοβάθμιας εξίσωσης: ax+b=0

```
program  exisosi;
var      a,b,x:real;
begin
        write('a='); readln(a);
        write('b='); readln(b);
        If a=o then if b=0 then writeln('AORISTH')
                                else writeln('ADYNATH')
                                 else begin
                                        x:=-b/a;
                                       writeln('x=',x)
                                        end
end.
```

Εντολή Case 1/4

- **Σκοπός:** Η επιλογή μιας πρότασης, ανάμεσα από μια ομάδα προτάσεων, σύμφωνα με την τιμή μιας έκφρασης

```
CASE  Εκφραση OF
Τιμή1: Πρόταση1;
Τιμή2: Πρόταση2;
      :            :
ΤιμήΝ: Πρόταση2;
END
```

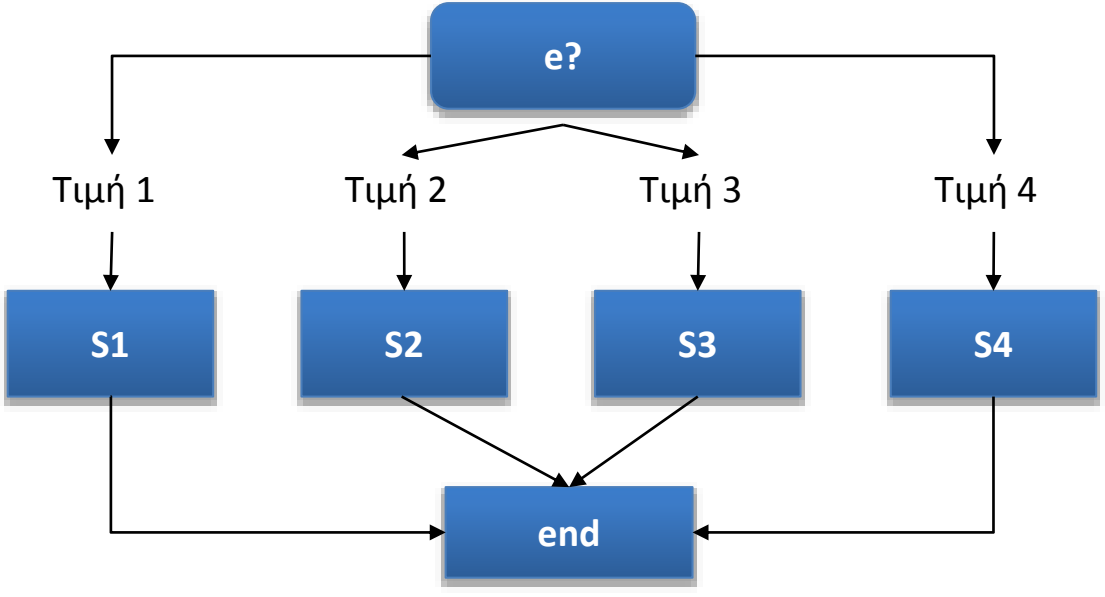
- **Εξηγήσεις:** Αρχικά υπολογίζεται η τιμή της έκφρασης (ονομάζεται και επιλογέας) και στην συνέχεια η τιμή αυτή συγκρίνεται με τις τιμές του case

Εντολή Case$_{2/4}$

- Όταν βρεθεί μια τιμή ίση με την τιμή της έκφρασης, τότε η αντίστοιχη πρόταση εκτελείται
- Η τιμή της έκφρασης μπορεί να είναι οποιουδήποτε διατεταγμένου τύπου, δηλαδή στοιχειώδη τύπου εκτός από real ή απαριθμητού τύπου ή τύπου υποπεριοχής

Εντολή Case$_{3/4}$

- Αν e είναι μια έκφραση και Si είναι εκτελέσιμες προτάσεις τότε η ροή εκτέλεσης της εντολή case. Δίνεται:



Εντολή Case 4/4

- **Παράδειγμα:**

Να γραφεί πρόγραμμα το οποίο να διαβάζει ένα χαρακτήρα A..F και να τυπώνει τα μηνύματα ¨Excellent", "Very good", "Good", "Fair", "Poor" και "Bad" αντίστοιχα για τις τιμές A,B,C,D,E και F

```
program  alex;
var      ch:char;
begin
        read(ch);
                case ch of
                'A': writeln('Excellent');
                'B': writeln('Very good');
                'C': writeln('Good');
                'D': writeln('Fair');
                'E': writeln('Poor');
                'F': writeln('End');
                 end
end.
```

Ασκήση

- **Παράδειγμα:**

Να γραφεί πρόγραμμα για την επίλυση της δευτεροβάθμιας εξίσωσης **ax^2+bx+c** όπου a, b και c πραγματικοί αριθμοί

```
program  exisosi;
var a,b,c,D,x,x1,x2:real;
begin
        write( 'Dose ta a,b,c:')
        readln(a,b,c);
        If a=0 then if b=o then if c=0 then writeln('Aoristh')
                                       else writeln('Adynath')
                            else begin {b=0}
                                   x:=-c/b;
                                   writeln('1 lysh x=',x);
                                  end
                    else begin
                          D:=sqr(b)-4*a*c;
                          if D<0 then writeln('2 rizes migadikes')
                          else if D=0 then writeln('diplh riza x=', -b/(2*a))
                          else begin
                                 x1=(-b+sqr(D))/(2*a);
                                 x2=(-b-sqr(D))/(2*a);
                                 writeln('2 rizes:',x1,x2)
                                  end
                           end
end.
```

Βιβλιογραφία

Βλαχάβας Ι. (1994). Η γλώσσα προγραμματισμού Pascal. Εκδόσεις Γαρταγάνης Διονύσιος.

Κάβουρας Ι.Κ. (1999). Δομημένος Προγραμματισμός με Pascal. Εκδόσεις Κλειδάριθμος.

Αλεβίζου Θ., & Καμπουρέλης Α. (1995). Μαθήματα Προγραμματισμού: Εισαγωγή με τη Γλώσσα Pascal. Εκδόσεις Παπασωτηρίου.

Cooper D. (1993). Oh! Pascal!, An Introduction to Computing, του. Εκδόσεις Norton.

Larry R.N. (1998). Advanced Programming in Pascal with Data Structures. Εκδόσεις Macmillan USA.

Τσελίκης Γ.Σ., Τσελίκας Ν.Δ. (2012). C: από τη Θεωρία στην Εφαρμογή (Β' Έκδοση). Εκδόσεις Παπασωτηρίου.

Aho A.V., Hopcroft J.E., & Ullman J.D. (1974). The design and analysis of computer algorithms. Εκδόσεις Addison Wesley.

Abelson H., Sussman G.J., Sussman J. (1985). Structure and Interpretation of Computer Programs, MIT Press, McGraw Hill Book Company.

Σημείωμα Αναφοράς

Σημείωμα Αδειοδότησης

[1] http://creativecommons.org/licenses/by-nc-nd/4.0/deed.el

Τέλος Ενότητας

Επεξεργασία: Ευάγγελος Καρβούνης

Άρτα, 2015

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Τέλος Ενότητας

Βασική σύνταξη προγράμματος pascal - Εντολές συνθήκης