ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΔΥΤΙΚΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ Τ.Ε.

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ 1629

‘Χρήση της γλώσσας Pascal στο μαθηματικό λογισμό’

Καραχάλιος Γεώργιος

Γιαννόπουλος Σπυρίδων

ΕΙΣΗΓΗΤΗΣ : ΚΑΡΕΛΗΣ ΔΗΜΗΤΡΙΟΣ

ΠΑΤΡΑ 2017

Περιεχόμενα

Περίληψη

Στο πρώτο κεφάλαιο θα παρουσιαστούν οι εισαγωγικές έννοιες της γλώσσας **Pascal** καθώς επίσης και τα βασικά χαρακτηριστικά ενός τέτοιου προγράμματος. Θα αναλυθεί ποιες εντολές είναι εκτελέσιμες και ποιες όχι. Τι είναι ο **compiler** και γενικά το ρόλο που διαδραματίζει στην εκτέλεση του προγράμματος όπου για κάθε έκδοση της γλώσσας **Pascal** είναι εν γένη διαφορετικός. Θα ακολουθήσουν οι έννοιες του αλγορίθμου, του προγράμματος, και θα συνταχθούν τα πρώτα στοιχειώδη προγράμματα. Το κεφάλαιο αυτό θα κλείσει με τύπους δεδομένων, λογικές συναρτήσεις, σταθερές και με μερικά πιο σύνθετα προγράμματα.

Στο δεύτερο κεφάλαιο θα αναλυθούν οι επιλογικές δομές με τα αντίστοιχα διαγράμματα ροής. Π.χ. η εντολή **if** καθώς και η εμφωλευμένη δομή επιλογής **if**. Το κεφάλαιο θα κλείσει με προγράμματα εφαρμογής όπως επίλυση εξισώσεων.

Στο τρίτο κεφάλαιο θα έχουμε τις δομές επανάληψης όπως **for, while, repeat**. Δομές ιδιαίτερα χρήσιμες όταν θέλουμε να εκτελέσουμε ένα μεγάλο αριθμό εντολών του ιδίου τύπου.

Στο τέταρτο κεφάλαιο θα ασχοληθούμε με πίνακες δεδομένων, τον ορισμό των πινάκων καθώς και την σύνταξή του στην **Pascal** όταν αυτή είναι μονοδιάστατοι και δισδιάστατοι. Το κεφάλαιο θα κλείσει με παραδείγματα πράξεων επί των πινάκων.

Στο πέμπτο και τελευταίο κεφάλαιο θα παρουσιάσουμε τις διαδικασίες **(procedures)** και τις συναρτήσεις **(function)** της γλώσσας **pascal**. Τον τρόπο σύνταξής τους και τους τύπους δεδομένων που εμφανίζονται. Σαν εφαρμογή θα τις χρησιμοποιήσουμε στη μελέτη εύρεσης παραγώγου και ολοκληρώματος πραγματικής συνάρτησης, πραγματικής μεταβλητής

Θα ακολουθήσει στο τέλος της εργασίας ένα παράρτημα με τους κωδικούς χαρακτήρες **ASCII**.

Κεφάλαιο 1

Εισαγωγή

Η **Pascal** είναι μια γλώσσα προγραμματισμού υψηλού επιπέδου **(high-level programming language),** που αναπτύχθηκε από τον ***Niklaus Wirth*** στα τέλη της δεκαετίας του **1960.** Η γλώσσα πήρε το όνομά της από τον επιστήμονα ***Blaise Pascal*****,** έναν Γάλλο μαθηματικό του **17**ου αιώνα**,** ο οποίος κατασκεύασε μια από τις πρώτες μηχανές άθροισης και έβαλε έτσι**,** ίσως άθελά του**,** το δικό του λιθαράκι στην ιστορία των υπολογιστικών μηχανών και κατ' επέκταση των υπολογιστών**.**

Η **Pascal** είναι περισσότερο γνωστή ως μια από τις πιο κατάλληλες γλώσσες προγραμματισμού για τη διδασκαλία των τεχνικών του δομημένου προγραμματισμού **(structured programming techniques).** Είναι τέτοια η φύση της γλώσσας που αναγκάζει τους προγραμματιστές να σχεδιάσουν τα προγράμματά τους μεθοδικά και προσεκτικά**.** Για τον λόγο αυτόν**,** είναι ιδιαίτερα δημοφιλής για διδασκαλία**.** Η **Pascal** σχεδιάσθηκε αρχικά για διδασκαλία και είναι ακόμα κατάλληλη γι' αυτή τη δουλειά**.**

Αν μάθετε την **Pascal,** τότε οι περισσότερες από τις υπόλοιπες γλώσσες προγραμματισμού θα σας φανούν οικείες**.**
Η **Pascal** χρησιμοποιεί τέτοια πρότυπα **(standards),** που κάνει εύκολο το γράψιμο των προγραμμάτων**.** Η ***TurboPascal*** αποτελεί μια δημοφιλή παραλλαγή **(**έκδοση**, version)** της **Pascal,** που δημιουργήθηκε από την εταιρεία ***Borland/Inprise Inc*****.**

Η **Pascal** παραμένει πάντα μια πολύ χρήσιμη γλώσσα**,** ιδιαίτερα κατάλληλη για διδασκαλία**.** Οι **C** και **C++** είναι πολύ συμβολικές γλώσσες**,** δηλ**.** ενώ η **Pascal** χρησιμοποιεί κανονικές αγγλικές λέξεις**,** όπως για παράδειγμα τις **begin** και **end,** οι **C/C++** χρησιμοποιούν σύμβολα**,** όπως **{** και **}.** Επίσης**,** ενώ οι **C and C++** δεν είναι πολύ αυστηρές στους τύπους δεδομένων που χρησιμοποιούν**,**
στην **Pascal** η ανάμειξη των τύπων δεδομένων συχνά προκαλεί λάθος**.** Αντίθετα**,** στις **C/C++,** τίποτα δεν συμβαίνει**.**

Η **Pascal** θυμίζει πολύ την αγγλική γλώσσα στις δομές της**.** Αυτό περιλαμβάνει και τη χρήση λέξεων **(words),** που είναι σειρές **(strings)** χαρακτήρων που αναγνωρίζουν στοιχεία μέσα στο πρόγραμμα**.** Υπάρχουν πολλά διαφορετικά είδη λέξεων για την

αναγνώριση μεταβλητών **(variables),** προγραμμάτων **(programs)** και υποπρογραμμάτων **(subprograms),** που είναι περισσότερα γνωστά ως διαδικασίες **(procedures)** και συναρτήσεις **(functions).**

Υπάρχει βέβαια και μια λίστα από λέξεις που έχουν μια συγκεκριμένη χρήση και τις οποίες δεν μπορεί να αλλάξει ο προγραμματιστής**.** Αυτές είναι γνωστές ως *δεσμευμένες λέξεις **(reserved words)***.

Τα Βασικά Χαρακτηριστικά ενός Προγράμματος της Pascal

Το κάθε πρόγραμμα που γράφουμε σε **Pascal** αποτελείται από τα εξής τρία συστατικά **:**

- *Επικεφαλίδα προγράμματος **(Program heading)** **: program identifier(input, output);*** Είναι πιθανώς το μοναδικό αναγνωριστικό **(identifier)** που το γράφουμε μία μόνο φορά και με μήκος έως **64** χαρακτήρες**.** Άρα**,** μπορούμε να το κάνουμε να είναι αρκετά περιγραφικό**.**
- *Τμήμα δηλώσεων **(Declaration section) : var const*.** Πρόκειται για μεταβλητές και σταθερές που τις δηλώνουμε εδώ και που θα τις χρησιμοποιήσουμε αργότερα στο πρόγραμμά μας**.** Εδώ δεν γίνεται καμία ενέργεια**,** αλλά απλά ενημερώνουμε την **Turbo Pascal** για το τι μεταβλητές και σταθερές θα χρειασθούμε αργότερα μέσα στο κυρίως πρόγραμμα**.**
- *Εκτελέσιμο τμήμα **(Executable section) : begin ... end.*** Βλέπουμε ότι η λέξη **end** ακολουθείται από μια τελεία**,** που υποδηλώνει και το τέλος ενός προγράμματος της **Pascal.** Ανάμεσα στα **begin** και **end** βρίσκονται όλες οι εντολές του προγράμματος**,** οι οποίες μπορούν να κάνουν εργασίες όπως λήψη **(**διάβασμα**)** στοιχείων**,** τροποποίησή τους και εμφάνισή τους **(**εκτύπωση**)** στην οθόνη **(screen)** ή στον εκτυπωτή **(printer).**

Οι Εντολές προς Εκτέλεση (Εκτελέσιμες Εντολές)

Μια εκτελέσιμη εντολή **(executable statement)** αποτελείται από έγκυρα αναγνωριστικά **(identifiers),** δεσμευμένες λέξεις **(reserved words),** αριθμούς ή**/**και χαρακτήρες με την κατάλληλη στίξη**.** Κατά μια άποψη**,** μια εκτελέσιμη εντολή θυμίζει πολύ μια φράση μέσα σε μια πρόταση**.** Η κάθε εκτελέσιμη εντολή πρέπει να τερματίζει με τον χαρακτήρα **; (semicolon).** Υπάρχει μία μόνο εξαίρεση σ' αυτόν τον κανόνα όταν η εντολή ακολουθείται από τη δεσμευμένη λέξη **END.**

Η **Turbo Pascal** δεν απαιτεί από τις εκτελέσιμες εντολές να χωρίζονται με αλλαγές γραμμών αλλά μόνο η παρουσία του χαρακτήρα **;** είναι απαραίτητη για να μπορούν να ξεχωρίζουν οι εντολές**.** Κάτι άλλο πολύ βασικό που πρέπει να τηρούμε είναι οι εσοχές **(indentation)** στις δομές των εντολών**.** Οι εσοχές δεν σημαίνουν τίποτα για την **Pascal** αλλά είναι πολύ βασικές για να μπορούμε να καταλάβουμε τον τρόπο λειτουργίας ενός προγράμματος**.**

Ο Compiler της Pascal

Ο υπολογιστής δεν μπορεί να κατανοήσει την ομιλούμενη ή τη γραπτή γλώσσα που χρησιμοποιούν οι άνθρωποι στις καθημερινές τους συνομιλίες και από την άλλη μεριά οι άνθρωποι δεν είναι σε θέση να κατανοήσουν τη δυαδική γλώσσα που χρησιμοποιεί ο υπολογιστής για να κάνει τις εργασίες του**.** Είναι συνεπώς απαραίτητο να γράψουμε τις εντολές **(**οδηγίες**)** μας σε μια ειδικά ορισμένη γλώσσα**,** όπως είναι η **Pascal** για παράδειγμα**,** την οποία μπορούμε να κατανοήσουμε**,** και μετά να την μετατρέψουμε σ' έναν λιτό κώδικα **(**γλώσσα μηχανής**)** που μόνο ο υπολογιστής μπορεί να καταλάβει**.**

Ένας *μεταγλωττιστής **(compiler)*** της **Pascal** είναι και ο ίδιος ένα πρόγραμμα υπολογιστή που η μόνη δουλειά του είναι να μετατρέπει ένα πρόγραμμα της **Pascal** από τη μορφή που μπορεί να κατανοήσει ο άνθρωπος σε μια μορφή που μόνο ο υπολογιστής μπορεί να διαβάσει και να εκτελέσει**.** Ο υπολογιστής προτιμά τις σειρές από τα δυαδικά ψηφία **0** και **1,** τα οποία δεν σημαίνουν τίποτα για μας αλλά που μπορούν να τύχουν μιας πολύ γρήγορης επεξεργασίας από τον υπολογιστή**.**

Το αρχικό πρόγραμμα που γράφουμε σε **Pascal** αποκαλείται ο *πηγαίος κώδικας* ***(source code)*** και ο προκύπτον μεταγλωττισμένος κώδικας που δημιουργείται από τον μεταγλωττιστή **(compiler)** αποκαλείται συχνά ένα *αντικείμενο αρχείο* ***(object file)*****.**

Ένα ή περισσότερα αντικείμενα αρχεία **(object files)** μπορούν να συνδυαστούν με κάποιες προκαθορισμένες βιβλιοθήκες **(libraries)** από ένα άλλο πρόγραμμα που αποκαλείται ***linker*** ή και ***binder*****,** για να δημιουργηθεί έτσι το τελικό ολοκληρωμένο αρχείο **(**εκτελέσιμο αρχείο **– executable file),** το οποίο και μπορεί να εκτελεσθεί από τον υπολογιστή**.**

Μια *βιβλιοθήκη* ***(library)*** είναι μια συλλογή από μεταγλωττισμένα αντικείμενα αρχεία **(object code)** τα οποία παρέχουν κάποιες λειτουργίες **(**συναρτήσεις**)** που καλούνται επανειλημμένα από πολλά προγράμματα υπολογιστών**.**

Η έννοια του Αλγορίθμου

Για να λύσουμε ένα πρόβλημα χρησιμοποιούμε μία ακολουθία αυστηρώς καθορισμένων βημάτων **(**εντολών**)** καθένα από τα οποία είναι εκτελέσιμο σε πεπερασμένο χρόνο**.** Αυτή η ακολουθία λέγεται **αλγόριθμος.** Ένας αλγόριθμος μπορεί ναι περιγραφεί με **φυσική** γλώσσα**,** με **ελεύθερο κείμενο**, με **διάγραμμα ροής** και με **κωδικοποίηση**. Παράδειγμα αλγορίθμου**,** σε φυσική γλώσσα είναι όταν κάποιος βράζει ένα αβγό**.**

- Βάλε νερό στο μπρίκι**.**
- Άναψε το γκαζάκι για να βράσει το νερό**.**
- Βάλε το αβγό στο μπρίκι όταν βράζει το νερό**.**
- Βγάλε το αβγό από το μπρίκι μετά από μερικά λεπτά

Αυτή η ακολουθία των τεσσάρων εντολών είναι πράγματι ένας αλγόριθμος γιατί έχει πεπερασμένο πλήθος εντολών οι οποίες εκτελούνται σε πεπερασμένο χρόνο **(**μερικά λεπτά**).** Είναι φανερό ότι σε έναν αλγόριθμο μπορούμε να τροποποιήσουμε κάποιες εντολές ή να προσθέσουμε και καινούριες**.** Ένας πιο σύνθετος αλγόριθμος είναι αυτός με τον οποίο θέλουμε να εισάγουμε δυο αριθμούς και να υπολογίζεται το άθροισμά τους**.** Αυτός είναι**:**

- Δώσε δύο αριθμούς και τοποθέτησε τους σε δύο μεταβλητές **x,y**

- Βρες το άθροισμα **x+y** και τοποθέτησε το στη μεταβλητή **sum.**
- Εμφάνισε την τιμή της μεταβλητής **sum** δηλαδή του αθροίσματος **x+y.**

Ο παραπάνω αλγόριθμος μπορεί να τροποποιηθεί ώστε να κάνει κάποια άλλη πράξη. Από τα παραπάνω δυο παραδείγματα είναι φανερό ότι ένας αλγόριθμος έχει δύο βασικά χαρακτηριστικά. Την είσοδο **(input)** και την έξοδο **(output).** Η είσοδος είναι δεδομένα που τα εισάγει ο χρήστης και η έξοδος που είναι το αποτέλεσμα της λύσης του προβλήματος.

Η έννοια του Προγράμματος

Το σύνολο των εντολών **(**κώδικας**)** το οποίο πρέπει να δώσουμε στον υπολογιστή ώστε να υλοποιηθεί ο αλγόριθμος λέγεται **πρόγραμμα** η διαδικασία με την οποία το καταγράφουμε λέγεται **προγραμματισμός.** Πρέπει να τηρούνται αυστηροί συντακτικοί κανόνες όταν γράφουμε τις εντολές για έναν αλγόριθμο. Οι κανόνες αυτοί λέγονται **γλώσσες προγραμματισμού** και διακρίνονται σε υψηλού και χαμηλού επιπέδου.

Οι υψηλού επιπέδου γλώσσες είναι πιο κατανοητές στον άνθρωπο όχι όμως και στον υπολογιστή. Το αντίθετο συμβαίνει στις γλώσσες χαμηλού επιπέδου. Ο χρήστης εισάγει τις εντολές στον υπολογιστή από το πληκτρολόγιο και αυτές μεταφράζονται από την αντίστοιχη γλώσσα προγραμματισμού σε γλώσσα μηχανής από είναι πρόγραμμα που λέγεται **compiler.**

Η γλώσσα προγραμματισμού Pascal.

Η **Pascal** έχει αρκετά πλεονεκτήματα αφού εκτός από το ότι είναι πολύ καλή για την εκπαίδευση, υποστηρίζει την συστηματική, δομημένη και αλγοριθμική επίλυση προβλημάτων. Είναι ευανάγνωστη, έχει ξεκάθαρα δομικά στοιχεία που βοηθούν στη βαθμιαία ανάπτυξη προγραμμάτων, έχει απλό, σύντομο και αυστηρό συντακτικό με αποτέλεσμα να είναι εύκολη η επαλήθευση της ορθότητας του προγράμματος.

Το Πρώτο Πρόγραμμα σε Pascal.

Στη σύντομη ιστορία του προγραμματισμού των υπολογιστών έχει δημιουργηθεί μια παράδοση που λέει ότι το πρώτο πρόγραμμα που γράφουμε σε μια καινούργια γλώσσα που μαθαίνουμε είναι πάντα το **"Hello, world".** Έτσι**,** δεν έχουμε παρά να αντιγράψουμε τον παρακάτω κώδικα σ' έναν κειμενογράφο **(text editor),** να τον αποθηκεύσουμε ως αρχείο με όνομα για παράδειγμα ***prog01.pas*****,** να τον μεταγλωττίσουμε **(compile)** και να τον εκτελέσουμε **(run) :**

```
program Hello;

    begin (* Κυρίως πρόγραμμα – αρχή *)

        write('Hello, world.')

end. (* Κυρίως πρόγραμμα – τέλος *)
```

Η έξοδος που θα δημιουργηθεί θα είναι ως εξής **:**

```
Hello, world.
```

Η πρώτη γραμμή του προγράμματος λέγεται επικεφαλίδα**.** Η λέξη **program** καθώς και όσες είναι γραμμένες έντονα λέγονται δεσμευμένες λέξεις και έχουν συγκεκριμένη σημασία για τον **compiler.** Η λέξη **Hello** δεν λαμβάνεται υπόψη από το πρόγραμμα**.** Με τις λέξεις **begin** και **end** έχω έναρξη και λήξη του προγράμματος**.** Στο συγκεκριμένο πρόγραμμα η μοναδική εντολή είναι η ***write('Hello, world.')*** η οποία λέει στον υπολογιστή να εμφανίσει στην οθόνη το μήνυμα **hello world.**

Να παρατηρήσουμε ότι μετά το ***program Hello;*** ακολουθεί ένα ελληνικό ερωτηματικό **; (semi –colon).** Αυτό το σύμβολο χρησιμοποιείται για να χωρίζουμε τις εντολές μεταξύ τους**.** Τοποθετείται στο τέλος κάθε εντολής εκτός αν αυτή τελειώνει με **end.** Τα εντός παρενθέσεως κείμενα και με αστερίσκο στην αρχή και στο τέλος αποτελούν σχόλια και αγνοούνται εντελώς από τον **compiler.** Η χρήση των σχολίων βοηθούν τον προγραμματιστή να γνωρίζει τη χρήση κάποιων εντολών ή κάποιου προγράμματος**.** Αυτά τα σχόλια είναι πολύ χρήσιμα κυρίως για τους αρχάριους και μπορούν να δώσουν μια εικόνα για το τι κάνει το πρόγραμμα**.** Από τα παραπάνω συμπεραίνουμε ότι η βασική δομή ενός προγράμματος σε γλώσσα **Pascal** είναι η εξής**:**

Τίτλος:	**program** όνομα
	uses wincrt
Δηλώσεις:	**var …**
	var…
	begin
Κυρίως	**begin**
Πρόγραμμα:	**end;**
	end.

Το Δεύτερο Πρόγραμμα

Ακολουθεί ένα δεύτερο πρόγραμμα πιο χρήσιμο από το προηγούμενο αλλά και πιο σύνθετο. **Δηλαδή να γίνει η πρόσθεση δύο ακεραίων αριθμών οι οποίοι θα εισάγονται από το πληκτρολόγιο και το αποτέλεσμα να εμφανίζεται στην οθόνη**. Τώρα είμαστε αναγκασμένοι να καταγράψουμε τον σχετικό αλγόριθμο του προβλήματος και μετά να δημιουργήσουμε το πρόγραμμα. Το προηγούμενο πρόγραμμα ήταν πολύ απλό και αρά δεν χρειαζόμασταν αλγόριθμο. Ο αλγόριθμος αυτός είναι:

- Να διαβάσει τον πρώτο ακέραιο και να τον αποθηκεύσει σε μια μεταβλητή.
- Να διαβάσει τον δεύτερο ακέραιο και να τον αποθηκεύσει σε μια άλλη μεταβλητή.
- Να προσθέσει τις δυο μεταβλητές και να τις αποθηκεύσει σε μια νέα μεταβλητή.
- Να εμφανίσει το αποτέλεσμα της πρόσθεσης στην οθόνη.

Συνολικά οι μεταβλητές που θα χρησιμοποιήσει το πρόγραμμα θα είναι τρείς. Τους ακεραίους **x,y** καθώς και το άθροισμα **sum** το οποίο και αυτός είναι ακέραιος. Ένα τέτοιο πρόγραμμα θα μπορούσε να ήταν το παρακάτω:

Program athroisma (input,output);

var x,y,sum: integer; (* αυτό το πρόγραμμα υπολογίζει το άθροισμα δυο ακεραίων και εμφανίζει το αποτέλεσμα στην οθόνη***)**

begin

writeln (‘δώσε τον πρωτο ακέραιο**:’);**

readln (x);

writeln (‘δώσε τον δεύτερο ακέραιο**:’);**

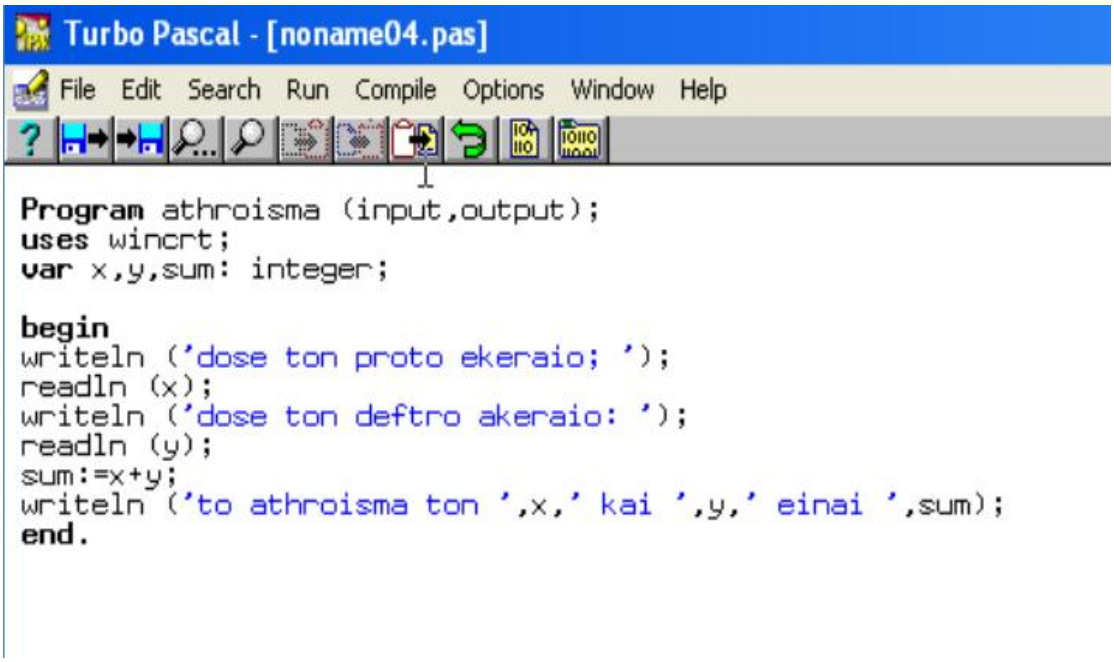
readln (y);

sum:=x+y;

writeln (‘το άθροισμα των **‘,x,’** και **‘,y,’** ειναι **‘,sum)**

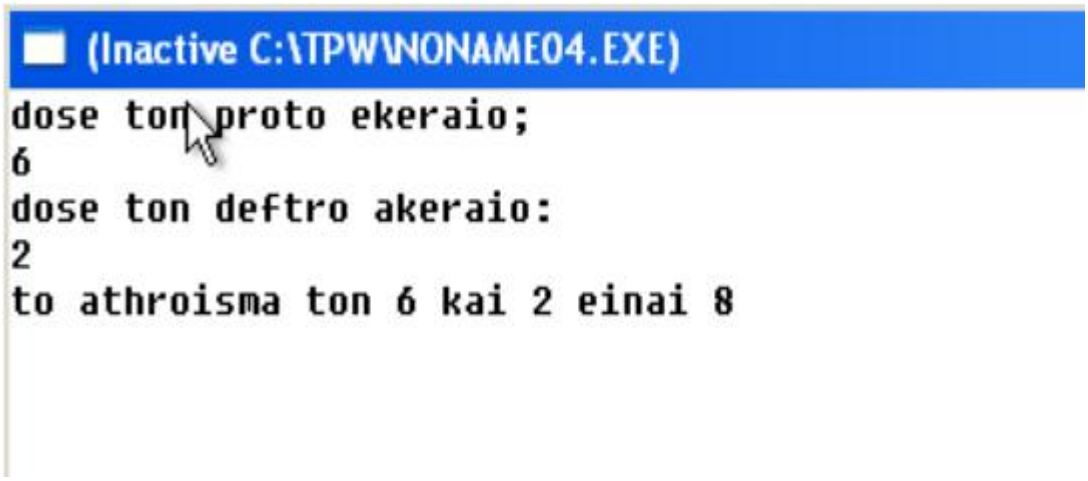
end.

Η εικόνα που ακολουθεί μας πληροφορεί πως ακριβός πληκτρολογήθηκε το πρόγραμμα σε **turbo Pascal.** Παρατηρούμε ότι το πρόγραμμα δεν δέχεται Ελληνικούς χαρακτήρες**.**



Μετά την εκτέλεση του ανωτέρου προγράμματος στην οθόνη θα εμφανιστεί ένα μήνυμα όπως αυτό καταχωρήθηκε ‘δώσε τον πρώτο ακέραιο**:’** πράγμα που σημαίνει ότι πρέπει να πληκτρολογήσουμε έναν ακέραιο π.χ. το έξι **(6).** Το οποίο και θα

αποθηκευτεί στην μεταβλητή **x** μέσω του **enter.** Θα ακολουθήσει η εμφάνιση δεύτερου μηνύματος ‘δώσε τον δεύτερο ακέραιο:’ π.χ. το δύο **(2).** Αφού δοθεί και αυτός θα αποθηκευτεί στη μεταβλητή **y** και το τελικό μήνυμα θα είναι: το άθροισμα των **6** και **2** είναι **8** όπως φαίνεται στην εικόνα που ακολουθεί.



Η πρώτη γραμμή του προγράμματος είναι η επικεφαλίδα**.** Η δεύτερη γραμμή του προγράμματος λέγεται γραμμή δήλωσης μεταβλητών **(var)** και ενημερώνει τον μεταγλωττιστή ότι τα **x** και **y** και **sum** είναι μεταβλητές και μάλιστα ακέραιες **(integer).** Έτσι λοιπόν ο μεταγλωττιστής θα δεσμεύσει τρείς αντίστοιχες θέσης μνήμης με τα ονόματα **x,y,sum.**

Το πρόγραμμα ξεκινάει με τη δεσμευμένη λέξη **begin** που δηλώνει την έναρξη του προγράμματος**.** Οι επόμενες δυο εντολές που ακολουθούν **readln (x)** και **readln (y)** εμφανίζουν δύο αντίστοιχα μηνύματα για τη εισαγωγή των ακεραίων και την αποθήκευσή τους**.** Η εντολή **sum:=x+y** λέει στον υπολογιστή να προσδιορίσει το άθροισμα **x+y** και να τον αποθηκεύσει στην μεταβλητή **sum.** η εντολή αυτή λέγεται **εντολή καταχώρησης** και το σύμβολο **:=** λέγεται **τελεστής καταχώρησης.** Το **x+y** λέγεται **αριθμητική έκφραση.**

Οι Δεσμευμένες Λέξεις της Γλώσσας Pascal.

Υπάρχουν λέξεις στη γλώσσα **Pascal** που προορίζονται για συγκεκριμένη χρήση, όπως **begin, end, program** οπου οι λέξεις αυτές δεν μπορούν να χρησιμοποιηθούν σαν ονόματα μεταβλητών. Οι λέξεις αυτές στο πρόγραμμα εμφανίζονται με έντονα γράμματα πράγμα που μας υπενθυμίζει ότι δεν πρέπει να τις χρησιμοποιήσουμε σαν μεταβλητές. Ακολουθεί ένας κατάλογος δεσμευμένων λέξεων.

and	**do**	**forward**	**mod**	**procedure**	**to**
array	**downto**	**function**	**nil**	**program**	**type**
begin	**else**	**goto**	**not**	**record**	**until**
case	**end**	**if**	**of**	**repeat**	**var**
const	**file**	**in**	**or**	**set**	**while**
div	**for**	**label**	**packed**	**then**	**with**

Τυποποιημένες Ταυτότητες της Pascal.

Οι τυποποιημένες ταυτότητες είναι λέξεις με προκαθορισμένη σημασία αλλά σε αντίθεση με τις δεσμευμένες λέξεις είναι δυνατόν ο προγραμματιστής να τις μεταβάλει και να τους δώσει κάποιο όνομα που να είναι τυποποιημένη ταυτότητα. Στην πράξη συνίσταται να χρησιμοποιούνται ακριβώς όπως οι δεσμευμένες λέξεις. Τέτοιες είναι:

Αρχεία: **input output**
Σταθερές: **false true maxint**
Τύποι: **boolean char integer real text**
Συναρτήσεις: **abs eof odd sin trunc arctan eoln** **ord sqr chr exp pred sqrt cos ln** **round succ**
Υποπρογράμματα: **dispose pack read rewrite** **writeln get page readln unpack** **new put reset write**

Τύποι Δεδομένων της Pascal

- Ακέραιος **(integer)**
- Πραγματικός **(real)**
- Χαρακτήρας **(char)**
- Λογικός **(boolean)**

Αυτοί είναι από τους πιο βασικούς τύπους προκαθορισμένων δεδομένων.

Οι ακέραιοι τύποι είναι οι γνωστοί από την άλγεβρα. Μπορεί να είναι θετικοί ή αρνητικοί. Κάθε μεταγλωττιστής της **Pascal** ορίζει ένα ανώτερο και κατώτερο όριο επιτρεπτών ακεραίων περίπου στις **32000** για το πάνω όριο **(maxint)** και **-32000** για το κάτω όριο **(minint).**Οι τελεστές των ακεραίων τύπων είναι δύο ειδών. Οι αριθμητικοί τελεστές και οι τελεστές συσχετισμού.

Αριθμητικοί τελεστές:

+ πρόσθεση

– αφαίρεση

***** πολλαπλασιασμός

div ακέραια διαίρεση **(**χωρίς δεκαδικά**)**

mod το υπόλοιπο της ακέραιας διαίρεσης

Τελεστές συσχετισμού:

> μεγαλύτερος

>= μεγαλύτερος ή ίσος

< μικρότερος

= ισος

<> άνισος

Εδώ να σημειώσουμε ότι οι τελεστές συσχετισμού μπορούν να χρησιμοποιηθούν και για τους τέσσερις τύπους δεδομένων. Υπάρχουν και αντίστοιχες τυποποιημένες συναρτήσεις των ακεραίων 'γνωστές' στον μεταγλωττιστή της **Pascal** οι οποίες και δίνουν ακέραιο αποτέλεσμα.

abs (x) Απόλυτη τιμή του **x.**

sqr(x) Το τετράγωνο του **x.**

tranc (x) Το ακέραιο μέρος του **x.**

round (x) Στρογγυλοποίηση του **x** στον κοντινότερο ακέραιο.

Συνεχίζοντας την παράθεση των τύπων δεδομένων θα αναφερθούμε στον πραγματικό τύπο **(real)** που δεν είναι τίποτε άλλο παρά η πραγματική **(**δεκαδικοί**)** αριθμοί. Δηλαδή όλοι εκείνοι οι

θετικοί και αρνητικοί αριθμοί που περιέχουν μια τελεία **(**υποδιαστολή**).**

Οι μεταγλωττιστές συνήθως αν δουν έναν ακέραιο αριθμό δηλωμένο ως **real** τον μετατρέπουν σε πραγματικό τοποθετώντας μια τελεία στο τέλος και αμέσως μετά ένα μηδενικό**.** Π**.**χ**.** ο αριθμός έξι **(6)** αν δηλωθεί ως **real** θα μετατραπεί αυτόματα σε **6,0.** Τυποποιημένες συναρτήσεις των πραγματικών στην **Pascal** είναι**:**

abs (x) Η απόλυτη τιμή του **x.**

sqr (x) Το τετράγωνο του **x.**

sin (x), cos (x), arctan (x) Είναι οι γνωστές τριγωνομετρικές συναρτήσεις όπου το **x** είναι σε **rad.**

ln (x), exp(x), sqrt (x) Με τους γνωστούς από τα μαθηματικά περιορισμούς για τις τιμές του **x.**

Ο τρίτος τύπος δεδομένων της **Pascal,** τύπος χαρακτήρα **char** περιλαμβάνει όλα τα διαθέσιμα σύμβολα**.** Οι μεταβλητές που δίνουμε στο πρόγραμμα χαρακτηρισμένες ως **char** πρέπει να είναι ένα μόνο σύμβολο και να τοποθετείται πριν και μετά το σύμβολο η απόστροφος**.** Π**.**χ**.** ο χαρακτήρας **3** πρέπει να δοθεί σαν **'3' , '@' , 'f' .**

Ο τελευταίος τύπος δεδομένων της **Pascal** δημιουργήθηκε από την ανάγκη στο να μπορούμε να ελέγχουμε αν ισχύει μια παράσταση ή όχι κατά τη διάρκεια του προγράμματος**.** Μια μεταβλητή **x** αν δηλώνεται να είναι τύπου **boolean** οι δυνατές τιμές που μπορεί να πάρει είναι δύο**. True (**αληθείς**)** ή **False (**ψευδείς**).** Συνδυασμός των ανωτέρων τύπων μεταβλητών δημιουργεί μια **λογική έκφραση (boolean expression).** Οι τελεστές συσχετισμού **(>, >=, <, <=, =, <>)** όπως επίσης και οι λογικοί τελεστές **and, or, not, xor** αποτελούν λογική έκφραση**.**

Τυποποιημένες Λογικές Συναρτήσεις

Οι συναρτήσεις αυτού του είδους επιστρέφουν κάποια λογική τιμή **true** ή **false** και είναι**:**

odd (x) Λαμβάνει την τιμή **true** αν ο ακέραιος **x** είναι περιτός και την τιμή **false** αν ο ακέραιος **x** είναι άρτιος**.**

eof: Χρησιμοποιείται για επεξεργασία κειμένου και λαμβάνει την τιμή **true** στο τέλος του κειμένου**,** εντός του κειμένου έχει την τιμή **false.**

eoln: Αυτή λαμβάνει την τιμή **true** στο τέλος της σειράς και την τιμή **false** οταν βρισκόμαστε στο μέσον της σειράς**.**

Οι Σταθερές (constants)

Οι σταθερές στον προγραμματισμό είναι κάτι ανάλογο με τις σταθερές στην άλγεβρα**.** Για κάθε σταθερά ορίζεται μια συγκεκριμένη τιμή εκ των προτέρων και δεν μπορεί να αλλάξει κατά τη διάρκεια του προγραμματισμού**.** Η χρήση της γίνεται με τη δεσμευμένη λέξη **const.** Π.χ**. const pi = 3.14 , const name = 'giorgos' .**

Ακολουθεί ένα πρόγραμμα χρήσης της const για τον υπολογισμό τού όγκου μίας σφαίρας ακτίνας r.

Είναι γνωστό από τη γεωμετρία ότι ο όγκος μιας σφαίρας ακτίνας **r** εξαρτάται μόνο από την ακτίνα και είναι**:**

v=(4/3)π**r**^{3} αρά**:**

Πρόγραμμα **1.1**

```
program ogkos_sfairas (input, output);
    const pi=3.14;
    var v,r:real;
    begin
        writeln ('Δώσε την ακτίνα της σφαίρας:');
        readln (r):
```

```
        v:= 4*pi*sqr(r)*r/3;
        writeln ('Ο όγκος της σφαίρας ειναι ισο με ',v)
    end
```

Μετά το τέλος του προγράμματος στην οθόνη εμφανίζεται ένα μήνυμα όπως αυτό το καταγράψαμε που μας προτρέπει να δώσουμε την ακτίνα της σφαίρας**,** για να ακολουθήσει ο υπολογισμός του όγκου της σφαίρας**.**

Ακολουθούν μερικά προγράμματα που υπολογίζουν τις τιμές διαφόρων γνωστών συναρτήσεων από τα μαθηματικά**.**

Αφού δοθεί ο πραγματικός αριθμός x να υπολογιστεί η τιμή της συνάρτησης y= f(x)= $(2x^2-2x+1)/(x-1)$, με x≠1, με μέγιστο αριθμό ψηφίων = 5 και 3 δεκαδικά.

Πρόγραμμα **1.2**

```
program ypologismos (input, output);
    var x:integer; y:real;
    begin
        writeln ('Δώσε τιμή για το x:');
        readln (x);
        y:= (2*sqr(x)-2*x+1)/(x-1);
        writeln ('Η τιμή της συνάρτησης είναι ίση με ',y:5:3);
    end.
```

Αν δώσουμε τον αριθμό π.χ**. 8** στην οθόνη θα εμφανιστεί**:** Η τιμή της συνάρτησης είναι ίση με **16.142** δηλαδή θα κρατήσει **3** δεκαδικά μετά την υποδιαστολή**.**

Να γραφεί πρόγραμμα που να υπολογίζει το εμβαδό ενός τραπεζίου συνάρτηση των δυο βάσεων a, b και με αντίστοιχο ύψος h.

Από την γεωμετρία γνωρίζουμε ότι το εμβαδό του τραπεζίου είναι**:** Ε**=(1/2)**·**(a+b)**·**h.** Όπου **a,b** οι δύο βάσεις και **h** η μεταξύ τους απόσταση**.**

Πρόγραμμα **1.3**

```
program ypologismos (input, output)
    var a,b,h,e:real
    begin
        writeln ('Δώσε τις τιμές των a, b, h');
        readln (a, b, h);
        e:=0.5*(a+b)*h;
        writeln ('Για το τραπέζιο με στοιχεία a,b,h');
        writeln (' ',a:5:2, b:5:2, h:5:2);
        writeln ('e=', e:6:2);
    end.
```

Το πρόγραμμα θα εμφανίσει μήνυμα στην οθόνη για να δώσουμε τις τιμές **a, b, h** και αν δώσουμε τις τιμές **a=20, b=40, h=10** το πρόγραμμα θα απαντήσει **e=300.00**

Να γραφεί πρόγραμμα το οποίο να διαβάζει τρείς πραγματικούς αριθμούς a, b, c και να υπολογίζει τη μέση τιμή τους x καθώς και την μέση τετραγωνική τιμή y.

Είναι γνωστό από τα μαθηματικά οτι η μέση τιμή **x** για Ν το πλήθος αριθμών είναι ίσο με **x=(N1+N2+...)/N** ενώ για τη μέση τετραγωνική τιμή έχουμε $\mathbf{y=((1/N)\cdot(N1^2+N2^2+...))^{1/2}}$

Πρόγραμμα **1.4**

```
program ypologismos (input, output)
    var a,b,c,x,y:real
    begin
        writeln ('Δώσε τις τιμές των a, b, c');
        readln (a,b,c);
        x:=(a+b+c)/3;
        y:=sqrt((1/3)*(sqr(a)+sqr(b)+sqr(c))
        writeln (' ',a:4:2, b:4:2, c:4:2);
        writeln ('Μέση τιμή x=', x:5:2);
        writeln ('Μέση τετραγωνική τιμη y=', y:5:2);
    end.
```

Αν, μετά το σχετικό μήνυμα, δώσουμε τους αριθμούς **a=20.8 b=-5.2** και **c=7.3** τότε το πρόγραμμα θα επιστρέψει:

Μέση τιμή **x=7.63**

Μέση τετραγωνική τιμή **y=13.07**

Να γραφεί πρόγραμμα που να υπολογίζει τον όγκο ενός δωματίου (room volume), την παράπλευρη επιφάνεια του (lateral surface), το εμβαδόν του πατώματος (floorarea) καθώς και την περίμετρο (perimmeter) με γνωστές τις διαστάσεις, μήκος (length), πλάτος (width) και ύψος (hight).

Πρόγραμμα **1.5**

```
program room (input, output)
    var lenght, width, height, perimeter, floorarea:integer;
    begin
        write ('Δώσε το μήκος, πλάτος, υψος: ');
        read (lenght, width, height);
        perimeter:=2*(lenght+width);
        floorarea:=lenght*width;
        writeln ('Περίμετρος=', perimeter:6);
        writeln ('Εκταση=', floorarea:6);
        writeln ('Πλευρική επιφάνεια=', perimeter*height);
        writeln ('Ογκος δωματίου=', floorarea*height);
    end.
```

Το πρόγραμμα θα εμφανίσει μήνυμα στην οθόνη για να δώσουμε τις τιμές για το μήκος, πλάτος και το ύψος όπου θα είναι **2,3** και **2** αντίστοιχα. Και θα εμφανίσει: Η περίμετρος θα είναι **10** η έκταση **6,** η πλευρική επιφάνεια **20** και ο όγκος δωματίου **12.**

Σχόλια Στο Πρόγραμμα Pascal

Η γλώσσα **Pascal,** όπως άλλωστε και όλες οι γλώσσες υψηλού επιπέδου, παρέχει τη δυνατότητα δημιουργίας σχολίων **(comments).** Αυτά τα σχόλια τοποθετούνται μέσα σε παρενθέσεις και ανάμεσα σε άγκιστρα. Ένα παράδειγμα διευκρινίζει τα παραπάνω.

Ένας τυπικός λογαριασμός της ΔΕΗ να τυπωθεί όπως φαίνεται παρακάτω.

##############################

ΝΕΑ ΕΝΔΕΙΞΗ ΜΕΤΡΗΤΗ **105KW**

ΠΑΛΑΙΑ ΕΝΔΕΙΞΗ ΜΕΤΡΗΤΗ **102KW**

ΚΑΤΑΝΑΛΩΣΗ **3KW**

ΤΙΜΗ ΜΟΝΑΔΑΣ **0.5€**

ΠΑΓΙΟ **15€**

ΣΥΝΟΛΟ **16.5€**

##############################

Πρόγραμμα **1.6**

```
program logariasmos_deh (input, output);
    const pagio=15
    var nea,palaia:integer;
    begin
        write ('Δώσε νέα και παλαιά ένδειξη μετρητή: ');
        writeln ('##########################');
        writeln ('ΝΕΑ ΕΝΔΕΙΞΗ ΜΕΤΡΗΤΗ', nea:12, 'KW');
        writeln ( 'ΠΑΛΑΙΑ ΕΝΔΕΙΞΗ ΜΕΤΡΗΤΗ', palaia:9, 'KW');
        writeln ('ΚΑΤΑΝΑΛΩΣΗ', nea-palaia:21, 'KW');
        writeln ('ΤΙΜΗ ΜΟΝΑΔΑΣ          0.5€');
        writeln ('ΠΑΓΙΟ', pagio:34:1);
        writeln ('ΣΥΝΟΛΟ');
        write((nea-palaia)*0.5+pagio):31:1);
        writeln('##########################');
end.
```

Τα αποτελέσματα που θα επιστρέψει το πρόγραμμα είναι το παραπάνω ζητούμενο.

Χρήση των τελεστών DIV και MOD

Είναι γνωστό από την άλγεβρα ότι η διαίρεση δύο αριθμών δίνει ένα πηλίκο και ένα υπόλοιπο. Οι τελεστές **DIV** και **MOD** (αναφέρθηκαν σε προηγούμενη παράγραφο) δίνουν ο τελεστής **DIV** το ακέραιο μέρος της διαίρεσης και ο τελεστής **MOD** το υπόλοιπο της διαίρεσης. Π.Χ.

Να γίνει πρόγραμμα που να διαβάζει δευτερόλεπτα και να εμφανίζει ώρες, λεπτά και δευτερόλεπτα που να αντοιστιχούν στο σύνολο των δευτερολέπτων που δώθηκαν.

Αρα στο πρόγραμμα θα χρειαστώ τέσσερις μεταβλητές: **sec** που δηλώνει τα δευτερόλεπτα που πρέπει να δώσουμε, **o** που δηλώνει τις ώρες, **l** που δηλώνει τα λεπτά και **d** που δηλώνει τα δευτερόλεπτα.**(o,l,d** θα δοθούν από το πρόγραμμα σαν αποτελέσματα**)**

Πρόγραμμα **1.7**

```
program defterolopta (input, output);
    var sec,o,l,d:integer;
    begin
        write ('Δώσε δευτερόλεπτα:  ');
        readln (sec);
        o:=sec DIV 3600;
        d:=sec MOD 3600;
        l:=d DIV 60;
        d:=d MOD 60;
        write ('Τα ',sec, 'δευτερόλεπτα αντιστοιχούν σε ');
        writeln(o, 'Ωρες ',l, 'λεπτα και ',d, ' δευτερα');
```

end.

Το αποτέλεσμα της εκτέλεσης του προγράμματος είναι**:**

Δώσε δευτερόλεπτα**: 5100**

Τα **5100** δευτερόλεπτα αντιστοιχούν σε **1** ωρες **25** λεπτα και **0** δευτερόλεπτα**.**

Κεφάλαιο 2

Επιλογικές Δομές

Στο πρόγραμμα **1.2** όπου ζητούσαμε την τιμή της συνάρτησης **y** για διάφορες τιμές του **x** εκεί υποθέσαμε ότι το **x** δεν μπορεί να πάρει την τιμή ένα δηλαδή η συγκεκριμένη συνάρτηση για **x=1** γίνεται άπειρη**.** Στο σχετικό πρόγραμμα που συντάχθηκε για αυτήν την περίπτωση σιωπηρός εννοείται ότι ο χρήστης δεν θα δώσει την τιμή **x=1 .** Πώς πρέπει να διαμορφωθεί ένα τέτοιο πρόγραμμα που να περιλαμβάνει και αυτή την περίπτωση όπου το πρόγραμμα να δώσει μήνυμα περί του αδυνάτου του **x=1;**

Πλην του ανωτέρου παραδείγματος**,** στην καθημερινότητα έχουμε να επιλύσουμε διάφορα προβλήματα και να αποφασίσουμε εκ των προτέρων ανάλογα με κάποιες δεσμευτικές συνθήκες αν το πρόγραμμα εκτελεστεί προς τη μια ή προς την άλλη κατεύθυνση**.** Δηλαδή να έχουμε ροή του προγράμματος προς την επιθυμητή κατεύθυνση κάθε φορά ανάλογα με τις προϋποθέσεις που ισχύουν**.** Σε τέτοιες περιπτώσεις πρέπει να χρησιμοποιούμε τις **δομές επιλογής.** Μία κατάλληλη εντολή για δομές επιλογής είναι η εντολή **if.**

Η εντολή if.

Το συντακτικό της δομής επιλογής είναι**:**

if απαιτούμενη συνθήκη**=true**

then

begin

διάφορες εντολές

end;

else

begin

άλλες συνθήκες

ακολουθούν άλλες εντολές

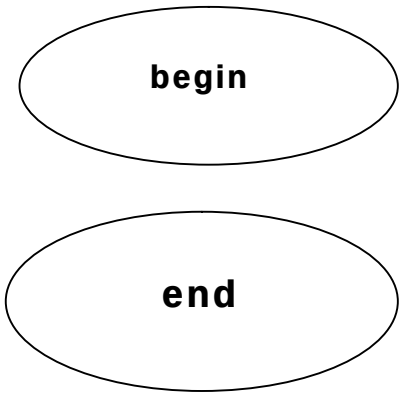
end.

Να εξηγήσουμε εδώ ότι οι απαιτούμενες συνθήκες που ακολουθούν το **if** θα περιέχουν διάφορους λογικούς τελεστές ή τελεστές συσχετισμού όπως π.χ. **or, and, not** και **<,>,=** κ.λπ. Οι μεταβλητές αυτές θα είναι σταθερές ή τύπου **boolean.**

- Η εντολή **if** θα εκτελεστεί αν η συνθήκη είναι αληθείς **(true)** διαφορετικά αν είναι ψευδής **(false)** το πρόγραμμα θα μεταφερθεί στη εκτέλεση της εντολής **else.**
- Αν μετά την εντολή **then** ή την εντολή **else** έχουμε μόνο μία συνθήκη τότε τα **begin** και **end** δεν χρειάζονται π.χ. **if x=5 then k:=k+2.** Η εντολή μετά το **then** είναι η **k:=k+2.**
- Είναι φανερό ότι ο όρος **else** είναι προαιρετικός.

Διαγράμματα ροής - Αλγόριθμοι

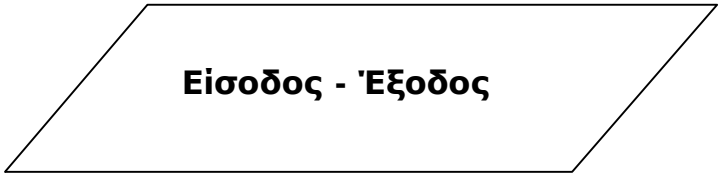
Είναι χρήσιμο και βοηθάει αρκετά στην τελική σύνταξη ενός προγράμματος η απεικόνιση ενός διαγράμματος μέσω γνωστών γεωμετρικών σχημάτων της ροής ενός προγράμματος. Ο προγραμματιστής σχεδιάζει ένα τέτοιο αρχικό διάγραμμα με γνωστά σχήματα και γραμμές ροής όπου αποτυπώνεται αρχικά η πιθανή επίλυση του προβλήματος. Μια τέτοια διαδικασία ονομάζεται **αλγόριθμος.** Τα γνωστά γεωμετρικά σχήματα που χρησιμοποιεί η **Pascal** καθώς και άλλε γλώσσες είναι:



Το σχήμα της έλλειψης δηλώνει την αρχή και το τέλος του αλγορίθμου.

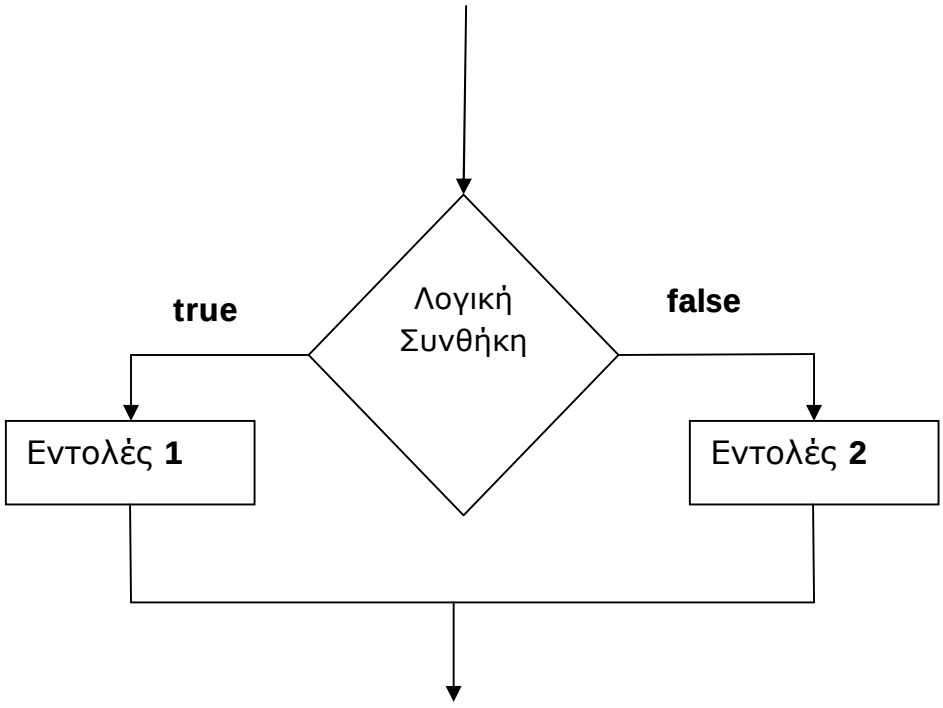
Το ορθογώνιο σχήμα δηλώνει επεξεργασία ή εκτέλεση πράξεων. Δηλαδή περιλαμβάνει το τμήμα των εντολών ανάθεσης.

Το σχήμα του ρόμβου δηλώνει κάποια λογική συνθήκη. Υπάρχουν δύο έξοδοι ανάλογα με την περίπτωση η συνθήκη να ειναι **true** ή **false**.



Το πλάγιο παραλληλόγραμμο σχήμα δηλώνει την είσοδο των δεδομένων στον αλγόριθμο με την εντολή **readln.** Επίσης δηλώνει την έξοδο των αποτελεσμάτων με την εντολή **writeln.**

Από τα προαναφερόμενα για το σχήμα του ρόμβου η εντολή **if** σχηματοποιείται ως εξής**:**



Δεν θα αποφύγουμε και εμείς τον πειρασμό να χρησιμοποιήσουμε σαν παράδειγμα τον υπολογισμό της απόλυτης τιμής ενός πραγματικού αριθμού που χρησιμοποιείται από το σύνολο των συγγραφέων ανώτερων γλωσσών προγραμματισμού**.**

Να γίνει πρόγραμμα που να εμφανίζει την απόλυτη τιμή ενός πραγματικού αριθμού (a) και να διευκρινίζει αν είναι θετικός η αρνητικός. Η εισαγωγή του αριθμού θα γίνεται από τον χρήστη.

Να υπενθυμίσουμε εδώ**,** πριν τη σύνταξη του προγράμματος τον ορισμό της απόλυτης τιμής**.** Η απόλυτη τιμή ενός αριθμού είναι ο

ίδιος ο αριθμός αν αυτός είναι θετικός**,** διαφορετικά **(**αν είναι αρνητικός**)** η απόλυτη τιμή του είναι ο αντίθετος του**.**

a, αν **a>=0**

|a|=

-a, αν **a<0**

Με βάση τα ανωτέρω το πρόγραμμα θα είναι**:**

Πρόγραμμα **2.1**

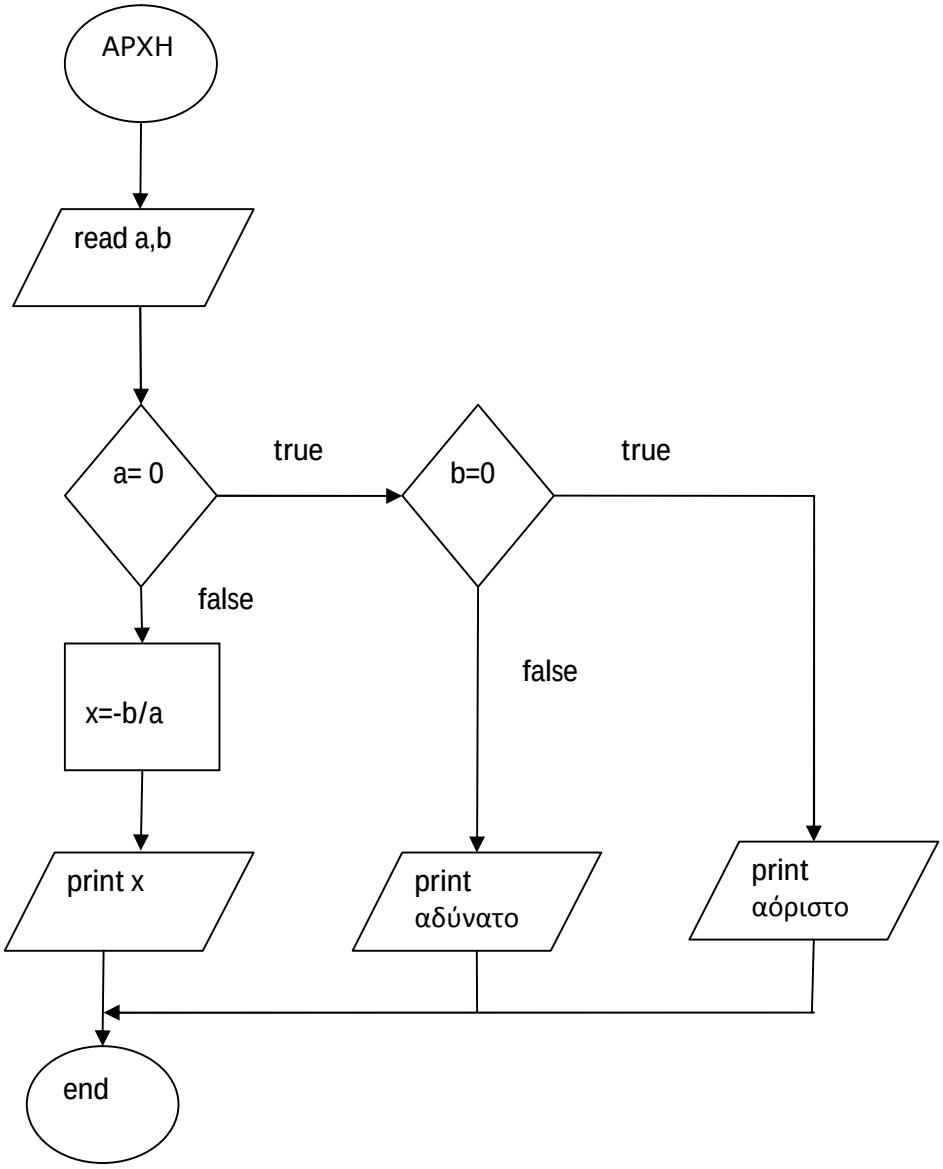
```
program apoliti_timi (input, output)
var a:real;
writeln (‘Δώσε αριθμο: ’);
readln (a);
if a >=0 then
begin
writeln (‘Η απόλυτη τιμή του ’,a,’ είναι: ’,a);
writeln (‘Ο αριθμός αυτός είναι θετικός ’);
end;
else
begin
writeln (‘Η απόλυτη τιμή του ’,a,’ είναι: ’,-a);
writeln (‘Ο αριθμός αυτός είναι αρνητικός ’);
end;
end.
```

Ένα επόμενο πιο προχωρημένο πρόγραμμα είναι είναι η επίλυση της γραμμικής εξίσωσης πρώτου βαθμού της μορφής ax+b=0. Τα a,b θα δίνονται από το χρήστη. Το πρόγραμμα να μας απαντάει αν έχει λύση κα ποιά είναι αυτή, καθώς και αν είναι αόριστη ή αδύνατη.

Είναι γνωστό ότι η λύση της ανωτέρω εξίσωσης είναι **x=-b/a** και ανάλογα με τις τιμές των **a,b** έχουμε τις εξής περιπτώσεις**.**

- Αν **a=0** και **b=0** τότε **x=**μη επιτρεπτό δηλαδή η εξίσωση είναι αόριστη**.**
- Αν **a=0** και **b≠0** τότε το **x=**άπειρο δηλαδή η εξίσωση είναι αδύνατη**.**

Να γίνει το λογικό διάγραμμα ροής καθώς και το σχετικό πρόγραμμα επίλυσης της ανωτέρω εξίσωσης που θα λαμβάνει υπ όψιν και θα δίνει μήνυμα των δύο ανωτέρω προϋποθέσεων**.**



Το πρόγραμμα σε γλώσσα **Pascal** είναι**:**

Πρόγραμμα **2.2**

```
program eksisosi;
var a:real;
b:real;
x:real;
begin
```

```
writeln (‘Επίλυση της εξίσωσης’);
writeln (‘Δωσε a ’);
read (a);
writeln (‘Δώσε b ’);
read (b);
if (a<>0) then
begin
x:-b/a;
writeln (‘Εχω λύση για x= ’,x:10:4);
end;
else
if (b=0) then
writeln (‘Αόριστη’);
esle
writeln (‘Αδύνατη’);
end.
```

Μετά την εκτέλεση του προγράμματος ακολουθούν μηνύματα για να δώσουμε τα **a** και **b** και ανάλογα θα μας απαντήσει αν υπάρχει λύση ή είναι αόριστη ή αδύνατη**.**

Συχνά συναντάμε σε διάφορες θετικές επιστήμες μια συνάρτηση **f(x)** να εκφράζεται με διαφορετικό νόμο ανάλογα με το πεδίο ορισμού δηλαδή ανάλογα με τις τιμές της μεταβλητής **x.** Η σύνταξη ενός προγράμματος διαχείρισης της **f(x)** πρέπει να περιλαμβάνει όλες τις εναλλακτικές τιμές της μεταβλητής **x.**

Να συνταχθεί πρόγραμμα πού να υπολογίζει τη συνάρτηση f(x) για τιμές του x που θα δίνονται από τον χρήστη.

$$f(x)=\begin{cases} 3x+4 & \text{για } x>5 \\ 1 & \text{για } x=5 \\ 4x+6\ln x+2 & \text{για } x<5 \end{cases}$$

Πρόγραμμα **2.3**

```
program eksisosi_f(x);
var x,f:real;
begin
writeln ('Δώσε τιμή στο x ');
read (x);
if (x>5) then
f:=3*x+4;
if (x=5) then
f:=1;
if (x<5) then
f:=4*x+6*ln(x)+2;
writeln ('f(x)= ',f:10:3);
end.
```

Μετά το τρέξιμο του προγράμματος θα εμφανιστεί μήνυμα να δοθεί ο αριθμός **x.** Αν δώσω **x=3** θα επιλεγεί η πρώτη μορφή της εξίσωσης **3x+4** και το πρόγραμμα θα απαντήσει **f(x)= 13.**

Επίλυση συστήματος δύο εξισώσεων με δύο μεταβλητές.

Ένα σύστημα της μορφής

$a_{11}x_1+a_{12}x_2=b_1$

$a_{21}x_1+a_{22}x_2=b_2$

Έχει λύση ως προς x_1 και x_2 που δίνεται απο τους τύπους

$x_1=(b_1 a_{22} - b_2 a_{12})/D$ και $x_2=(b_2 a_{11} - b_1 a_{21})/D$

Όπου $D=(a_{11} a_{22} - a_{12} a_{21})$ είναι η ορίζουσα του παρονομαστή. Το σύστημα θα έχει λύση μόνο όταν $D\neq 0$ διαφορετικά η λύση είναι αδύνατη. Μετά τα ανωτέρω έχουμε.

Να γραφεί πρόγραμμα που να βρίσκει τη λύση του παραπάνω συστήματος όταν υπάρχει διαφορετικά να δίνει μήνυμα ότι η λύση είναι αδύνατη.

Πρόγραμμα **2.4**

```
program systima;

var a11,a12,b1,a21,a22,b2,D,x1,x2:real;

begin

writeln (‘Λύση συστήματος’);

writeln (‘Δώσε a11’);

read (a11);

writeln (‘Δώσε a12’);

read (a12);

writeln (‘Δώσε b1’);

read (b1);

writeln (‘Δώσε a21’);

read (a21);

writeln (‘Δώσε a22’);
```

```
read (a22);
writeln (‘Δώσε b2’);
read (b2);
D:=(a11*a22-a12*a21);
if (D<>0) then
begin
x1:=(b1*a21-b2*a12)/D;
x2:=(b2*a11-b1*a21)/D;
writeln (‘x1= ’);
writeln (‘x2= ’);
end;
else
if (D=0) then
writeln (‘αδύνατη’);
end.
```

Για συστήματα περισσότερων των δύο εξισώσεων θα χρησιμοποιήσουμε τη μέθοδο των πινάκων αναλόγων διαστάσεων όπως θα δούμε σε επόμενο κεφάλαιο.

Λύση εξίσωσης δευτέρου βαθμού.

Μια τέτοια εξίσωση έχει τη μορφή $\mathbf{ax^2+bx+c=0}$ με $\mathbf{D=b^2-4ac}$ είναι η διακρίνουσα της εξίσωσης και έχουμε τρείς περιπτώσεις.

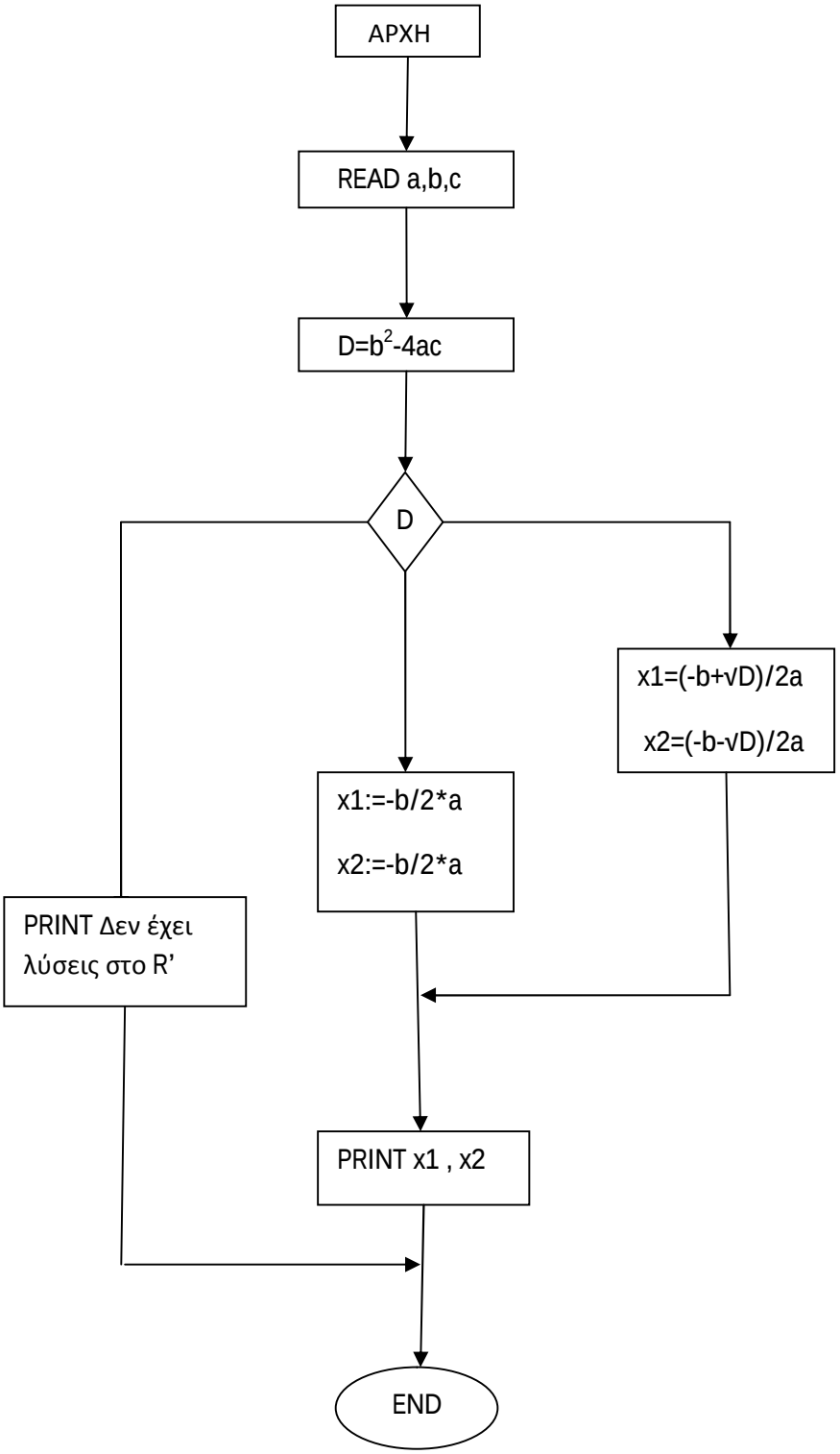
Αν **D<0** η εξίσωση δεν έχει πραγματικές λύσεις στο **R.**

Αν **D=0** η εξίσωση έχει μια ρίζα διπλή **x1=x2=-b/2a.**

Αν **D>0** η εξίσωση έχει δύο ρίζες πραγματικές και άνισες **x1=(-b+√D)/2a** και **x2=(-b-√D)/2a**

Μετά τα ανωτέρω έχουμε.

Πριν τη σύνταξη του προγράμματος θα παραθέσουμε την αλγοριθμική λύση χρησιμοποιώντας τα γνωστά διαγράμματα ροής.



Να γράφει πρόγραμμα που να διαβάζει τα a,b,c και να δίνει τα x1 και x2 σε κάθε περίπτωση να βγάζει αντίστοιχο μήνυμα.

Πρόγραμμα **2.4**

```
program rizes_eksisosis;
var a,b,c,D:real;
begin
writeln (‘Δώσε a’);
read (a);
writeln (‘Δώσε b’);
read (b);
writeln (‘Δώσε c’);
read (c);
D:=sqr(b)-4*a*c;
if (D<0) then
begin
writeln (‘Δεν έχει λύσεις στο R’);
end
if (D=0) then
begin
x1:=-b/2*a;
x2:=-b/2*a;
writeln (‘H εξίσωση έχει μια ρίζα διπλή x1= ’);
end;
if (D>0) then
begin
```

```
x1:=(-b+sqrt(D))/2*a;

x2:=(-b-sqrt(D))/2*a;

writeln ('Η πρώτη ρίζα είναι x1= ');

writeln ('Η δεύτερη ρίζα είναι x2= ');

end;

end.
```

Στο πρόγραμμα **1.6** το ζητούμενο ήταν να τυπωθούν οι ενδείξεις του μετρητή καθώς και το κόστος αν **KWH** με συγκεκριμένο τρόπο**.** Ένα άλλο πρόγραμμα που περιλαμβάνει διαβαθμισμένη τη σχέση κόστους ανά **KWH** είναι το εξής**.**

Μια εταιρία παροχής ηλεκτρικού ρεύματος χρεώνει τι **KWH** κλιμακωτά ως εξής**:**

- **10** ευρώ πάγιο**.**
- Οι πρώτες **100 KWH** με **0,75** ευρώ ανά **KWH.**
- Τις επόμενες **200** K**WH** με **0,9** ευρώ ανά **KWH.**
- Πάνω από **300 KWH** με **1,2** ευρώ ανά **KWH.**

Να γίνει πρόγραμμα όπου ο χρήστης θα δίνει τον αριθμό των KWH που καταναλώνει και να εμφανίζεται το κόστος που πρέπει να πληρώσει στη εταιρία.

Οι μεταβλητές που έχουμε σε ένα τέτοιο πρόγραμμα είναι έστω **x** το σύνολο των **KWH** και **y** το αντίστοιχο κόστος**.** Άρα για το **x** έχουμε τις εξής περιπτώσεις**.**

- $x \leq 100$ άρα $y=10+0.75x$
- $100 < x \leq 300$ αρα $y=10+100\cdot 0.75+(x-100)\cdot 0.9$
- $x>300$ αρα $y=10+100\cdot 0.75+200\cdot 0.9+(x-300)\cdot 1.2$

Το πρόγραμμα θα είναι το εξής:

Πρόγραμμα **2.5**

```
program logariasmos;
var x,y:real;
begin
writeln (‘Δώσε αριθμό KWH: ’);
readln (x);
if x<0 then
writeln (‘Μή αναμενόμενα δεδομένα’);
else
begin
if x<=100 then
y:= 10+0.75*x;
else
if x<=300 then
y:= 10+100*0.75+(x-100)*0.9;
else
y:= 10+100*0.75+200*0.9+(x-300)*1.2;
writeln (‘Ο λογαριασμός είναι: ’,y, ‘ευρώ’);
end;
end.
```

Κεφάλαιο 3

Δομές Επανάληψης

Στο προηγούμενο κεφάλαιο είδαμε προγράμματα της **Pascal** με τη δυνατότητα να λαμβάνονται αποφάσεις με συγκεκριμένο έλεγχο πάνω σε διάφορες μεταβλητές. Είναι αναγκαίο σε ένα πρόγραμμα να επαναλάβουμε την εκτέλεση της ίδιας εντολής πολλές φορές. Έτσι λοιπόν οι ακολουθιακές δομές **(**εκτέλεση εντολής η μία μετά την άλλη**)** καθώς και οι δομές επιλογής δεν είναι πρακτικές**,** αφού οδηγούν σε προγράμματα αρκετά μεγάλα. Η σκέψη στη λύση αυτού του προβλήματος είναι η δημιουργία των επαναληπτικών δομών.

Επαναληπτικός βρόχος **(loop)** χαρακτηρίζεται η δυνατότητα του υπολογιστή να εκτελεί με επανάληψη μια ομάδα εντολών. Σαν τέτοιους επαναληπτικού βρόχους η **Pascal** διαθέτει τις εντολές **for, while** και **repeat.** Τα σταθερά χαρακτηριστικά ενός βρόχου είναι**:**

- **Σώμα του βρόχου (body):** Είναι οι εντολές στο εσωτερικό του βρόχου που θα επαναλαμβάνονται.
- **Μετρητης (counter):** Πρόκειται για μεταβλητή που έχει τη δυνατότητα να αυξάνεται ή να μειώνεται και ορίζει τον αριθμό των επαναλήψεων.
- **Είσοδος του βρόχου (loop entry):** Πρόκειται για το σημείο έναρξης για πρώτη φορά της ροης μέσα στο βρόχο.
- **Επανάληψη (iteration):** Η διαδικασία επανάληψης των εντολών των βρόχων.
- **Λογικός έλεγχος (contitional test):** Είναι το σημείο που αποφασίζεται αν θα επαναληφθούν οι εντολές του βρόχου**,** για μια λογική έκφραση.
- **Έξοδος του βρόχου (loop exit):** Είναι το σημείο στο οποίο τελειώνει και ο τελευταίος βρόχος και το πρόγραμμα εξέρχεται στην επόμενη εντολή.
- **Συνθήκη τερματισμού (terimination control).** Όταν η λογική συνθήκη γίνει **false** σταματάει η εκτέλεση του βρόχου.

Η εντολή for

Παράδειγμα χρήσης επαναληπτικού βρόχου έχουμε όταν θέλουμε να βρούμε το άθροισμα ή το γινόμενο των πενήντα πρώτων θετικών ακεραίων, αφού θα πρέπει να γράψουμε πενήντα φορές την εντολή **sum:=1+2+3+4+5+...+50.**

Σύνταξη της εντολής **for:**

for μετρητής **:=**αρχική τιμή **to** τελική τιμή **do**

begin

εντολές

end;

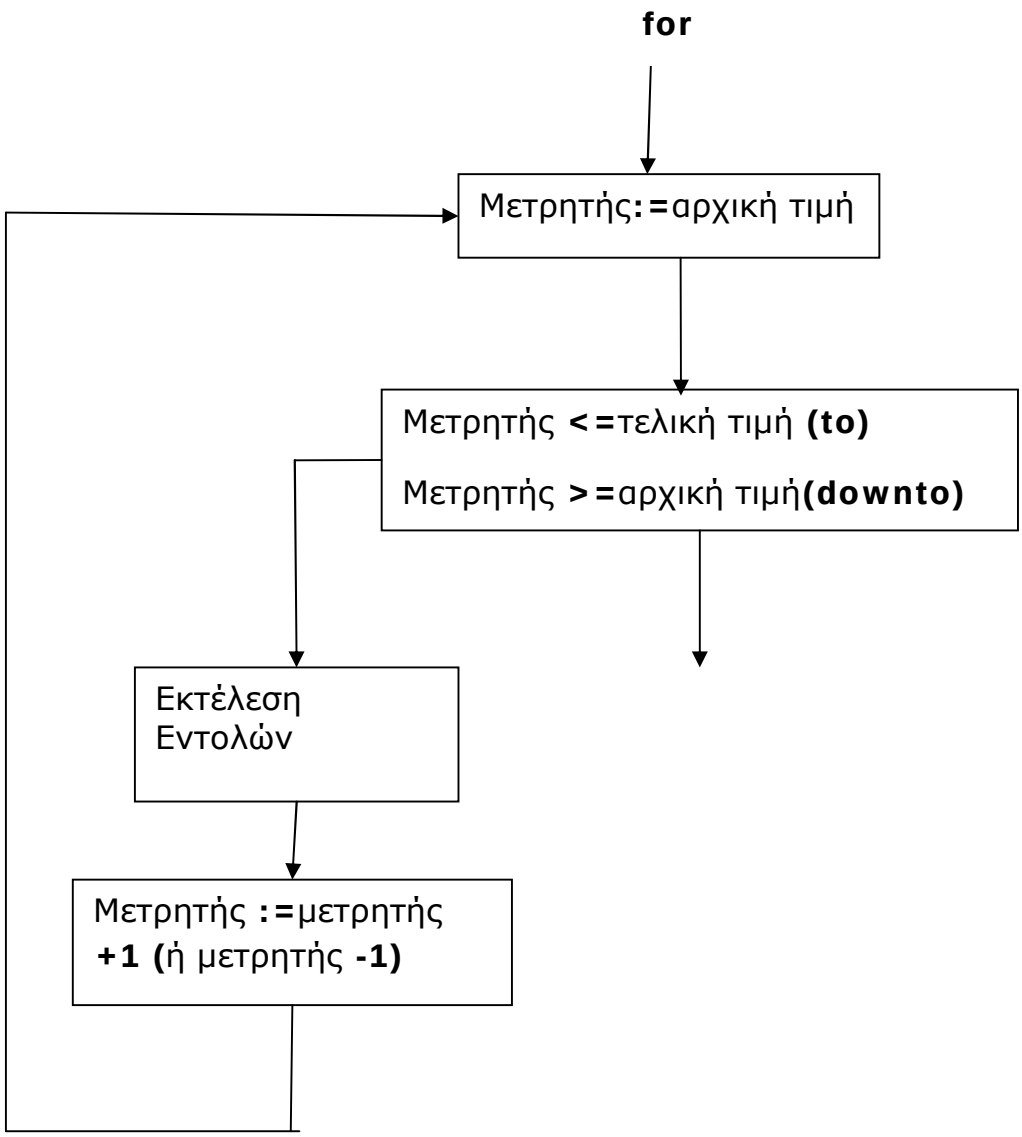
Μέσω της αρχικής και της τελικής τιμής του μετρητή, που είναι ακέραιοι αριθμοί ορίζουμε πόσες φορές θα εκτελεστεί ο βρόχος. Αυτή η εντολή είναι πολύ χρήσιμη γιατί αποφεύγονται οι άσκοπές επαναλήψεις των ίδιων εντολών αρκεί να γνωρίζουμε την αρχική και την τελική τιμή του μετρητή.

Σχόλια:

1)Αν η εντολή **for** χρησιμοποιεί μια μόνο περίπτωση τότε δεν χρειάζονται τα **begin** και **end** διαφορετικά χρειάζονται, όπως εξάλλου συμβαίνει και με την εντολή **if.**

2) Όταν η αρχική τιμή είναι μικρότερη της τελικής τότε χρησιμοποιούμε το **to** διαφορετικά χρησιμοποιούμε το **downto.**

Το συντακτικό της εντολής **for** σε διάγραμμα ροής είναι το παρακάτω.



Σαν παράδειγμα θα συντάξουμε ένα πρόγραμμα που θα διαβάζει έναν ακέραιο αριθμό n και θα υπολογίζει το γινόμενο 1·2·3·…·n δηλαδή θέλουμε το n! (παραγοντικό) καθώς και το άθροισμα 1!+2!+3!+…+n!.

Θα χρειαστούμε τέσσερις μεταβλητές**.**

i=ακέραιος με τιμές από **i=1** έως **i=n**

n=το σύνολο των ακεραίων

p=το γινόμενο **1·2·3·...·n**

sum=το άθροισμα των επιμέρους παραγοντικών **1!+2!+3!+...+n!**

Πρόγραμμα **3.1**

```
program paragontiko;
var i,n,p,sum:integer
begin
p:=1;
sum:=0;
write (‘Δώσε τιμή’);
readln (n);
for i:=1 to n do
begin
p:=p*i;
sum:=sum+p;
end
writeln (‘To ’ ,n, ‘παραγοντικό ειναι ’ ,p);
writeln (‘To άθροισμα των παραγοντικών εινα ’ ,sum);
end.
```

Μετά το τρέξιμο του προγράμματος και αν δόσουμε π.χ. **= 6** τοτε θα δώσει **6!=720** και **sum= 873.**

Ένα πιο πρακτικό και χρήσιμο πρόγραμμα είναι αυτό που μετατρέπει τη θερμοκρασία απο βαθμούς κελσίου σε βαθμούς Fahrenheit. Θέλουμε τα αποτελέσματα να εμφανίζονται σε μορφή στηλών. Οι τιμές της θερμοκρασίας θα είναι από -30 εως 120 βαθμούς κελσίου.

Η σχέση μεταξύ των βαθμών κελσίου **(c)** και των βαθμών **Fahrenheit (f)** είναι**: (f-32)/9=c/5** και λύνοντας αυτή τη σχέση ως προς **f** έχουμε**: f=(9·c/5)+32.**Θα χρειαστούμε δυο μεταβλητές την θερμοκρασία **c** που έστω ότι είναι ακέραιος και τη θερμοκρασία **f** που θα είναι **real** λόγω της διαίρεσης που περιέχει ο τύπος**.**

Το σύνολο των ακεραίων τιμών της μεταβλητής **c** θα είναι **120-(-30)=150** τιμές άρα θα χρησιμοποιήσουμε τον επαναληπτικό βρόχο **for** και τον ρόλο του μετρητή θα τον αναλάβει η μεταβλητή **c** με αρχική τιμή **-30** και τελική τιμή **120.**

Πρόγραμμα **3.2**

```
program celcius_fahrenheit;

var c:integer;

f:real;

begin

writeln('c | f');

for c:=-30 to 120 do

begin

f:=(9*c/5)+32;

writeln (c,' |  ',f:5:1);

end;

end.
```

Το αποτέλεσμα θα είναι αριστερά η στήλη των βαθμών κελσίου και δεξιά η αντίστοιχοι βαθμοί **Fahrenheit.**

Να συνταχθεί πρόγραμμα που να εντοπίζει όλους του διψήφιους φυσικούς αριθμούς που να διαιρούνται ακριβώς με το άθροισμα των ψηφίων τους. Να υπολογίζει και να εμφανίζει το μέσο όρο αυτών των αριθμών (που διαιρούνται επακριβώς).

Οι φυσικοί διψήφιοι αριθμοί είναι προφανώς οι ακέραιοι από το **10** έως το **99** άρα θέλουμε ένα **for** από **10** έως **99.** Έστω **x** αυτή η μεταβλητή που θα είναι **integer.** Θέλω άλλες δύο μεταβλητές έστω **k** και **l** για το πρώτο και δεύτερο ψηφίο και αυτοί θα είναι **integer.** Ομοίως μια μεταβλητή **sum** για το άθροισμα **(integer).** Έναν μετρητή **j (integer)** και μια μεταβλητή **y=sum/j (real)** για τον μέσο όρο**.**

Πρόγραμμα **3.3**

```
programm fysikoi_dipsifioi;

var x,j,k,l,sum:integer; y:real;

begin

writeln ('Οι αριθμοί που διαιρούνται ακριβώς με το άθροισμά τους είναι: ');

j:=0;

sum:=0;

for x:=10 to 99 do

begin

k:=x div 10;

l:= x mod 10;

if x mod (k+l)=0 then

begin

j:= j+1;

write (x:3);

sum:=sum+x;

end;
```

end;

y:= sum/j;

writeln (‘Ο μέσος όρος είναι**: ’ ,y);**

end.

Η εντολή while

Η εντολή **while** ανήκει στην ίδια κατηγορία των επαναληπτικών βρόχων **(**όπως και η **for)** αλλά αυτή μπορεί να εκφράσει οποιαδήποτε επαναληπτική διαδικασία αντίθετα από την **for** που δεν μπορεί**.** Κάθε φορά που η λογική συνθήκη είναι **true** εκτελείτε το σύνολο των εντολών της **while.**

Η σύνταξη στον προγραμματισμό της **while** είναι**:**

while λογική έκφραση **do**

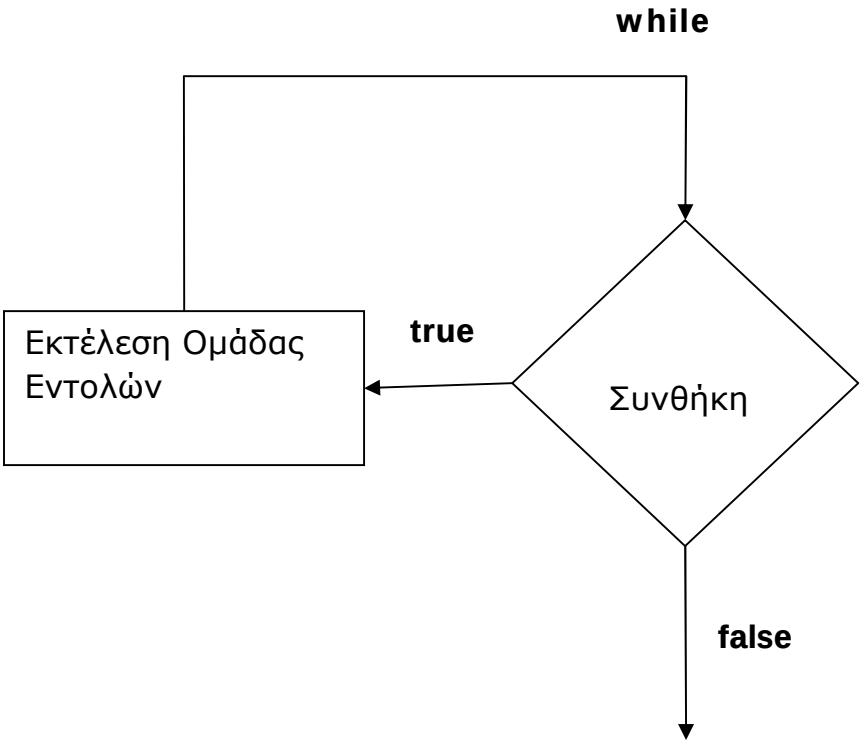
begin

εντολές

end;

Όσο η λογική έκφραση είναι **true** οι εντολές εκτελούνται και όταν γίνει **false** ο βρόχος διακόπτεται**.** Ένα πλεονέκτημα της **while** είναι το ότι δεν απαιτείτε να γνωρίζουμε από την αρχή τον αριθμό των επαναλήψεων του βρόχου αντίθετα με την εντολή **for** που πρέπει να είναι γνωστός**.** Στη **while** απαιτούνται εσωτερικά **begin...end** όταν έχω πάνω από δύο εντολές**,** όπως συμβαίνει εξάλλου και στο **for** και στο **if.**

Η σύνταξη της **while** σε μορφή διαγράμματος ροής είναι**:**



Σαν εφαρμογή της εντολής **while** έχουμε**:**

Να βρεθεί το άθροισμα των θετικών ακεραίων απο 1 εως n. Δηλαδή s=1+2+3+...+n. Τον αριθμό n θα τον δίνει ο χρήστης με n≤50.

Το πρόβλημα αυτό θα μπορούσε να συνταχθεί και με την εντολή **for.** Εδώ όμως θα το συντάξουμε με την εντολή **while.**

Οι μεταβλητές που θα χρησιμοποιήσουμε θα είναι τρείς**.** Η μεταβλητή **x** που θα μετράει τους προσθετέους με αρχική τιμή ίση με ένα**.** Η μεταβλητή **n** που θα δίνεται από το χρήστη και η μεταβλητή **s** που θα είναι κάθε φορά το ενδιάμεσο άθροισμα**.**

Όλες οι μεταβλητές θα είναι **integer** μιας και το άθροισμα ακεραίων είναι πάλι ακέραιος**.**

Πρόγραμμα **3.4**

```
program athroisma_akeraiwn;
var x,n,s:integer;
begin
writeln (‘Δώσε το n: ’);
readln (n);
if (n<1) or (n>50) then
writeln (‘Λάθος δεδομένα’);
else
begin
s:=0;
x:=1;
while x<=n do
begin
s:=s+x;
x:=x+1;
end;
writeln (‘Το άθροισμα είναι ισο: ’,s);
end
end.
```

Η εντολή repeat

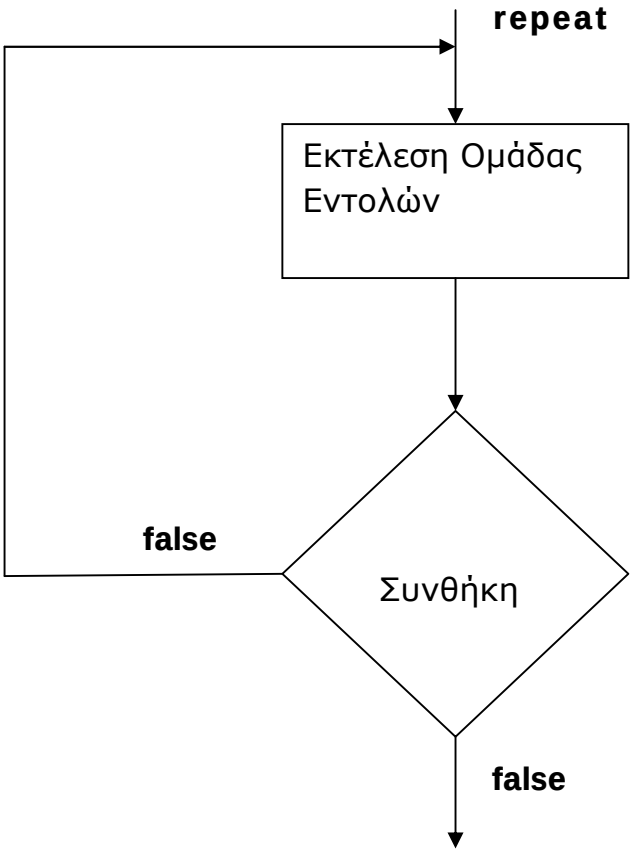
Και αυτή η εντολή ανήκει στην κατηγορία των επαναληπτικών βρόχων **(**δομές επανάληψης**).** Ο βρόχος εκτελείται όσο η λογική έκφραση είναι **true** και διακόπτεται όταν γίνει **false.** Ο βρόχος **repeat** εν αντιθέσει με τον **while** θα εκτελείται τουλάχιστον μια φορά και μετά θα κάνει έλεγχο της λογικής συνθήκης**.**

Η εντολή **repeat** είναι χρήσιμη στο να εντοπίζει μη αποδεκτά δεδομένα από το σύνολο των δεδομένων που εισάγει ο χρήστης δίνοντας του την ευκαιρία να τα διορθώσει**.**

Σύνταξη της εντολής**:**

repeat εντολές **until** δίνω λογική έκφραση

Το σχηματικό διάγραμμα της εντολής είναι το παρακάτω**:**



Να συνταχθεί πρόγραμμα που να βρίσκει το άθροισμα διάφορων ακεραίων αριθμών και να επισημαίνει αν κάποιος είναι μηδέν.

Θέλουμε μία μεταβλητή για τους ακέραιους έστω οτι είναι το **n** και μία μεταβλητή για το άθροισμά τους**,** έστω ότι είναι το **s.** Και οι δύο θα είναι **integer.**

Πρόγραμμα **3.5**

```
program prosthesi;
var n,s:integer;
begin
s:=0;
repeat
readln (n);
s:=s+n;
until
n:=0;
writeln ('s= ',s:5);
end.
```

Να συνταχθεί πρόγραμμα που να υπολογίζει το άθροισμα $s=e\cdot\cos1+e^2\cdot\cos2+\ldots+e^x\cdot\cos x=\Sigma e^x\cdot\cos x$.

Όπου **x** ακέραιος που δίνεται από τον χρήστη και εκφράζει το σύνολο των αθροισμάτων.

Πρόγραμμα **3.6**

```
program athroisma;
var x:integer;
s:real;
begin
writeln ('Δώσε τον αριθμό αθροισμάτων ') ;
readln (x);
writeln ('Για ', x:3, 'έχουμε αθροίσματα οτι ');
s:=0;
```

```
repeat

s:=s+exp(x)*cos(x);

n:=n-1;

until (x=0);

writeln (‘Το άθροισμα είναι s= ’,s:10:3);

end.
```

Εάν το πρόγραμμα τρέξει με μήνυμα θα μας ζητηθεί να δώσουμε τον αριθμό των αθροισμάτων και αν δώσουμε π.χ. **x=5** θα μας δώσει ότι το άθροισμα είναι **s=-1523,915**

Να κατασκευαστεί πρόγραμμα σε γλώσσα Pascal το οποίο να διαβάζει έναν δεκαδικό ακέραιο αριθμό από το 0 έως το 255 και να εμφανίζει τον αντίστοιχο αριθμό στο δυαδικό σύστημα αρίθμησης.

Μόνο **2** μεταβλητές θα χρειαστούμε. Την **NumberInDecimal** η οποία είναι **integer** και την **sNumberInBinary** η οποία είναι **string.**

Πρόγραμμα **3.7**

```
program Decimal2Binary;

var NumberInDecimal: Integer;

sNumberInBinary: string[8];

begin

Write('Δώστε έναν δεκαδικό αριθμό απ'' το 0 έως το 255: ');

Readln(NumberInDecimal);

Write('Στο δυαδικό σύστημα ο δεκαδικός αριθμός ',
NumberInDecimal, ' είναι ο: ');

if NumberInDecimal = 0 then

sNumberInBinary := '0'
```

```
else

begin

sNumberInBinary := '';

while NumberInDecimal > 0 do

begin

sNumberInBinary := chr(48 + NumberInDecimal mod 2) +
sNumberInBinary;

NumberInDecimal := NumberInDecimal div 2;

end;

end;

Writeln(sNumberInBinary:8);

end.
```

Κεφάλαιο 4

Πίνακες

Προκειμένου να κατασκευάσουμε ένα πρόγραμμα που περιέχει τον ίδιο τύπο δεδομένων πολλές φορές πρέπει να δημιουργήσουμε διάφορες δομές δεδομένων **(data structures).** Χρησιμοποιώντας λοιπόν δομές δεδομένων το πρόγραμμα γίνεται πιο ευέλικτο, πιο λειτουργικό και καθιστούν πιο εύκολη την αποθήκευση, την πρόσβαση, την αναζήτηση και την ομαδοποίηση δεδομένων. Οι απλούστερες δομές δεδομένων είναι οι **πίνακες.**

Ορισμός:

Μια δομή δεδομένων των οποίων όλα τα στοιχεία είναι ομοειδή **(**του ιδίου τύπου**)** θα την ονομάζουμε πίνακα. Οι πίνακες χωρίζονται σε μονοδιάστατους, δισδιάστατους και πολυδιάστατους. Σε αυτή την εργασία θα αναφερθούμε στους μονοδιάστατους και δισδιάστατους.

Έτσι λοιπόν όταν έχουμε π.χ. ένα σύνολο από πενήντα διαφορετικές μετρήσεις μιας τάσης ή ενός ρεύματος **(**μεταβλητές του ιδίου τύπου είναι πιο εύκολο αντί να ορίσουμε πενήντα διαφορετικές μεταβλητές στο πρόγραμμα να ορίσουμε μια μόνο μεταβλητή με το όνομα του πίνακα που θέλουμε. Μπορούμε λοιπόν να επεξεργαστούμε πιο εύκολα τα στοιχεία του πίνακα και να βρούμε τα μέγιστα, τα ελάχιστα, το άθροισμα τους, το γινόμενο τους κ.λπ.

Σε προηγούμενες εντολές είδαμε ότι όταν ο χρήστης εισάγει μια νέα τιμή η προηγούμενη σβήνεται ενώ τώρα με τις εντολές πινάκων, όλες οι τιμές **(**στοιχεία του πίνακα**)** παραμένουν και είναι διαθέσιμα στον χρήστη οποτεδήποτε χρειαστούν.

Μονοδιάστατος Πίνακας.

Ένας μονοδιάστατος πίνακας αποτελεί μια δομή όπου μπορούμε να καταχωρίσουμε σε μία μόνο μεταβλητή **(**όνομα πίνακα**)** απεριόριστα στοιχεία του ιδίου τύπου. Η θέση του κάθε στοιχείου στον πίνακα είναι καθορισμένη και μπορεί να προσδιοριστεί μονοσήμαντα με έναν αριθμό.

Π.χ. έστω ότι έχουμε ορίσει έναν πίνακα με το όνομα της μεταβλητής **a** ο οποίος περιέχει έξι στοιχεία τα οποία είναι ακέραιοι

αριθμοί. Οι ακέραιοι αυτοί είναι π.χ. **4,5,7,10,15,22** τότε στη μνήμη του υπολογιστή θα δημιουργηθεί η παρακάτω εποπτική κατάσταση.

a[1]	a[2]	a[3]	a[4]	a[5]	a[6]
4	5	7	10	15	22

Δηλαδή το στοιχείο **4** είναι το πρώτο στοιχείο του πίνακα **a,** το στοιχείο **5** είναι το δεύτερο στοιχείο του πίνακα **a** κ.λπ.

Στην **Pascal** ένας μονοδιάστατος πίνακας ορίζεται ως εξής:

var όνομα πίνακα:**array [a..b] of** τύπος – στοιχείων;

Το όνομα του πίνακα μπορεί να είναι οποιαδήποτε μεταβλητή **a,b,c** ή και σύνολο χαρακτήρων. Το **[a..b]** είναι το πρώτο και το τελευταίο στοιχείο του πίνακα. Ο τύπος των στοιχείων μπορεί να είναι **real, integer, char** ή **boolean** Η αρχικοποίηση για το **i-**στοιχείου του πίνακα γίνεται ως εξής: όνομα_πίνακα **[i]:=** τιμή.

Σχόλια:

1) Δεν είναι υποχρεωτικό στη δήλωση του πίνακα να αρχικοποιήσουμε όλα τα στοιχεία του. Μπορούμε να αρχικοποιήσουμε μόνο το τρίτο και το πέμπτο δηλαδή στο κυρίως πρόγραμμα να δώσουμε τις εντολές **a[3]:=7** και **a[5]:=15.**

2)Η επεξεργασία ενός πίνακα γίνεται ξεχωριστά ανά στοιχείο του πίνακα παράδειγμα αν θέλουμε όλα τα στοιχεία του προηγούμενου πίνακα να είναι ίσο με πέντε δεν επιτρέπεται η εντολή **a:=5** αλλά θα πρέπει να γράψουμε: **a[1]=5, a[2]:=5...a[6]:=5.** Αν όμως τα στοιχεία τού πίνακα είναι πάρα πολλά πρέπει να χρησιμοποιήσουμε μία **for.**

for i:=1 to 6 do a[i]:=5;

Να γραφούν οι εντολές για τη δημιουργία ενός πίνακα με το όνομα a που περιέχει τα στοιχεία 7,6,5,4,3,1.

Ο πίνακας αυτός είναι μονοδιάστατος και περιέχει **6** στοιχεία που είναι ακέραιοι αριθμοί. Θα έχουμε λοιπόν:

Πρόγραμμα **4.1**

```
var a:array [1..6] of integer
begin
a[1]:=7;
a[2]:=6;
a[3]:=5;
a[4]:=4;
a[5]:=3;
a[6]:=1;
end.
```

Πιο σύντομα:

```
for i:=1 to 6 do
a[i]:=7-i;
```

Να γραφεί πρόγραμμα σε γλώσσα Pascal που να διαβάζει τα ακέραια στοιχεία ενός πίνακα και να υπολογίζει το άθροισμα τους.

Έστω Ν το πλήθος των ακεραίων αριθμών και **i=1,2,...,N** για κάθε ένα στοιχείο, **S** θα είναι το άθροισμα. Και **a** για τον πίνακα. Όλα θα είναι **integer.**

Πρόγραμμα **4.2**

```
program Monodiastatos_Pinakas;
var i,s,N:integer;
a:array [1..100] of integer;
begin
writeln ('Δώσε το πλήθος των αριθμών : ');
readln(N);
s := 0;
for i:=1 to N do
begin
writeln('Δώσε αριθμό : ');
readln(a[i]);
end;
for i:=1 to N do
s:= s+a[i];
writeln('Το άθροισμα των αριθμών είναι : ', s);
end.
```

Δίνεται μονοδιάστατος πίνακας με δέκα στοιχεία και με όνομα a=(1,2,3,4,5,6,7,8,9,10). Να γραφεί πρόγραμμα που να υπολογίζει τη μέση τιμή των τιμών του πίνακα a.

Θα χρειαστούμε μια εντολή **read (a[i])** που θα διαβάζεται ο πίνακας **a** με τιμές από **i=1** έως **n=10.**

Θα χρειαστούμε έναν αθροιστή έστω **s** που θα αθροίζει όλες τις τιμές του πίνακα **a** με αρχικοποίηση **s=0.** Και μία μεταβλητή **y** για τον μέσο όρο**.** Όπου **y=s/n.** Άρα έχουμε**:**

Πρόγραμμα **4.3**

```
program athroisma_pinaka;
const n=10;
var i:integer;
s,y:real;
a:array [1..n] of integer;
begin
s:=0;
for i:= 1 to n do
begin
writeln (‘Δώσε το a[’, i:2, ‘]’ );
readln (a[i]);
s:s+a[i];
end;
y:=s/n;
writeln (‘Ο μέσος όρος είναι = ’,y:10:3);
end.
```

Τα αποτελέσματα θα είναι:

Δώσε το **a[1]** **1**

Δώσε το **a[2]** **2**

.

.

.

Δώσε το **a[10]** **10**

Ο μέσος όρος είναι **= 5.500**

Να συνταχθεί πρόγραμμα σε γλώσσα Pascal που να διαβάζει το πλήθος των στοιχείων του, να βρίσκει ποιό είναι το μεγαλύτερο στοιχείο καθώς και τη θέση του στοιχείου στον πίνακα.

Οι μεταβλητές που θα χρειαστούν είναι N για το πλήθος**, i** για το κάθε ένα**, T** για την θέση του κάθε ενός**, s** για το άθροισμα**, m** για το μεγαλύτερο και **a** το όνομα του πίνακα**.** Όλες οι μεταβλητές θα είναι **integer.**

Πρόγραμμα **4.4**

```
program max_pinaka;

var i,N,T,s,m:integer;

a : Array[1..100] of integer;

begin

writeln ('Δώσε το πλήθος των αριθμών : ');

 readln(N);

for i:=1 to N do

begin

writeln ('Δώσε έναν αριθμό : ');

readln (a[i]);

end;

m:=a[1];

T:=1;

for i:=2 to N do

if a[i] > m then

begin

m:=a[i];

T:=i;
```

end;

writeln ('το μεγαλύτερο στοιχείο του πίνακα είναι το **: ', m,'** και βρίσκεται στη θέση **: ', T);**

end.

Να γραφεί πρόγραμμα σε γλώσσα Pascal το οποίο να διαβάζει ένα μονοδιάστατο πίνακα ακέραιων αριθμών Α με 10 θέσεις και υπολογίζει και να τυπώνει το μέσο όρο, το άθροισμα και το γινόμενο των στοιχείων του πίνακα Α.

Οι απαιτούμενες μεταβλητές είναι **i** για κάθε ένα στοιχείο από **1** έως **10, s** το άθροισμα τους**, g** για το γινόμενο**,** οι οποίες θα είναι **integer.** και **mo** ο μέσος όρος που θα είναι **real.**

Πρόγραμμα **4.5**

program ginomeno;

var i,s,g:integer ;

mo:real;

a:array [1..10] of integer;

begin

for i:=1 to 10 do

begin

write ('Δώσε **a[' ,**ι**, ']= ');**

read (a[i]);

end;

s:=0;

g:=1;

for i:-1 to 10 do

begin

s:=s+a[i];

```
g:=g*a[i];
end;
mo:= s/10;
writeln (‘Το άθροισμα είναι : ’,s);
writeln (‘Το γινόμενο είναι : ’,g);
writeln (‘Ο μέσος όρος είναι : ’,mo:5:2);
end.
```

Δίνεται πίνακας με το όνομα a που περιέχει 20 πραγματικούς αριθμούς. Πίνακας με το όνομα b που περιέχει 30 χαρακτήρες και πίνακας c που περιέχει 15 λογικές μεταβλητές. Δώστε πρόγραμμα που να διαβάζει τα στοιχεία του πίνακα a και να τα τυπώνει ανάποδα. Να διαβάζει τα στοιχεία του πίνακα b και να τυπώνει το πλήθος των εμφανίσεων του χαρακτήρα e. Να διαβάζει τα στοιχεία του πίνακα c και να τυπώνει το πλήθος των true και το πλήθος των false.

Πρόγραμμα **4.6**

```
program pinakes;
var a:array [1..20] of real;
b:array [1..30] of char;
c:array [1..15] of boolean;
i,j,k: integer;
begin
i:=0; (*αρχικοποίηση του μετρητή στον πίνακα a*)
j:=0; (*αρχικοποίηση του μετρητή στον πίνακα b*)
k:=0; (*αρχικοποίηση του μετρητή των true στον πίνακα c*)
```

```
writeln (‘Δηλωσε 20 πραγματικούς αριθμούς: ’);
for i:=1 to 20 do
readln (a[i]);
for i:=1 to 20 do
writeln (a[21-i]); (*εμφάνιση 20 αριθμών με ανάπωδη σειρά*)
writeln (‘Δώσε 30 χαρακτήρες: ’);
for i:=1 to 30 do
if b[i]= ‘e’ then
j:=j+1;
writeln (‘Το πλήθος των a στον πίνακα b είναι: ’ ,j);
writeln (‘Δώσε 15 λογικές μεταβλητές: ’);
for i:=1 to 15 do
readln (c[i]);
for i:=1 to 15 do
if c[i]
k:=k+1;
writeln (‘Το πλήθος των true είναι: ’,k,’ και το πλήθος των false είναι : ‘,15-k);
end.
```

Δισδιάστατος Πίνακας.

Δισδιάστατο πίνακα ορίζουμε μια δομή δεδομένων η οποία μπορεί να καταχωρηθεί σε μια μόνο μεταβλητή που περιέχει απεριόριστα στοιχεία του ιδίου τύπου. Να φανταστούμε έναν πίνακα μιας διάστασης (μιας γραμμής) όπου ακολουθεί δεύτερη, τρίτη κ.λπ. γραμμές. Με γενικό τύπο **a=i,j.** Με **i=1,2,...,n** οι γραμμές του πίνακα και **j=1,2,...,m** οι στήλες του πίνακα. Με **n≠m.** Αν **n=m** ο πίνακας λέγεται τετραγωνικός.

Η γλώσσα **Pascal** μπορεί να πραγματοποιήσει και πράξεις επί των πινάκων**,** όπως πρόσθεση και αφαίρεση καθώς και πολλαπλασιασμό δύο πινάκων**.** Ο τρόπος εισαγωγής δισδιάστατων πινάκων στη γλώσσα **Pascal** εισάγεται σε γραμμική παράταξη όπου η κάθε γραμμή διαχωρίζεται από την άλλη με ένα κόμμα **(,).**

var a:array [1..i,1..j];

Ας δούμε ένα παράδειγμα πως γίνεται η πρόσθεση και η αφαίρεση**.**

Η άλγεβρα των πινάκων μας λέει ότι η πρόσθεση δύο πινάκων ορίζεται μόνο όταν οι δύο πίνακες έχουν ίσες διαστάσεις και το τελικό αποτέλεσμα που προκύπτει στο νέο πίνακα **(**ιδίων διαστάσεων με τους προσθετέους**)** είναι το αποτέλεσμα του αθροίσματος των στοιχείων του κάθε πίνακα**.**

Άν ορίσουμε πίνακα $\mathbf{a=a_{ij}}$ και $\mathbf{b=b_{ij}}$ τότε το άθροισμά τους είναι ένας νέος πίνακας $\mathbf{c_{ij} = a_{ij} + b_{ij}}$**.** Όπου τ**o i = 1,2,....,n** και **j = 1,2,....,m** είναι οι γραμμές και οι στήλες αντίστοιχα**.**

Στην **Pascal** πρέπει να δώσουμε **c[i,j]:=a[i,j]+b[i,j];**

Δίνονται δύο πίνακες με διαστάσεις k x l. Να γίνει πρόγραμμα που να υπολογίζει το άθροισμα και την διαφορά των δύο πινάκων.

Πρόγραμμα **4.6**

```
program prosthesi_pinakon;

var

i,j,k,l: integer;

a,b,c,d: array [1..10, 1..10] of real;

begin

write (‘Δώσε τις στήλες’);

readln (k);

write (‘Δώσε τις γραμμές’);

readln (l);

for i:=10 to k do
```

```
begin
for j:=1 to l do
begin
write (‘Δώσε τις τιμές a[I,J], b[I,J] για I= ’, I:2, ‘J= ’, I:2, ‘:’);
readln (a[i, j], b[i, j]);
c[i, j] := a[i, j] + b[i, j];
d[i, j] := a[i, j] - b[i, j];
end;
end;
writeln (‘Πίνακας αθροίσματος’);
writeln (‘----------------’)
for i:=1 to k do
begin
for j:=1 to l do
begin
write (’ ’ , c[i, j]:8:3);
end;
writeln;
writeln;
end;
writeln (‘Πίνακας Διαφοράς’);
writeln (‘----------------’)
for i:=1 to k do
begin
for j:=1 to l do
begin
```

```
write (' ' , d[i, j]:8:3);

end;

writeln;

end;

end.
```

Αν **a = 1 3 5 9** και **b = 2 4 1 0**
2 3 7 0 **5 1 0 7**

Τότε μετά το τρέξιμο του προγράμματος θα μας ζητηθεί να δώσουμε τις στήλες **(j)** και τις γραμμές **(i).** Εμείς θα δώσουμε **j=4** και **i=2.** Ακολούθως θα μας ζητηθεί να δώσουμε τις τιμές των πινάκων **a** και **b** και δίνοντας για τον πίνακα **a** τις τιμές **1 3 5 9 , 2 3 7 0** και για τον **b** τις τιμές **2 4 1 0 , 5 1 0 7** θα πάρουμε

Πίνακας Αθροίσματος

3.000 7.000 6.000 9.000

7.000 4.000 7.000 7.000

Πίνακας Διαφοράς

-1.000 -1.000 4.000 9.000

-3.000 2.000 7.000 -7.000

Έστω οτι έχουμε δύο διανύσματα **a** και **b** με **a=a1i + a2j +...** και **b=b1i + b2j +...** όπου **a1, a2,...** και **b1, b2,...** είναι οι συντεταγμένες τους**.** Τα **i, j,...** είναι τα μοναδιαία διανύσματα και θέλουμε να υπολογίσουμε το εσωτερικό τους γινόμενο **a*b.**

Από την άλγεβρα των διανυσμάτων ξέρουμε ότι το εσωτερικό γινόμενο ορίζεται σαν το άθροισμα των γινομένων των αντιστοίχων συντεταγμένων**,** δηλαδή **a*b=a1*b1+a2*b2+...a_n*b_n.**

Άρα έχουμε:

Να συνταχθεί πρόγραμμα που να υπολογίζει το εσωτερικό γινόμενο δύο διανυσμάτων a,b με τις συντεταγμένες που θα εισάγονται από τον χρήστη με τιμές από 0 έως 10.

Για την σύνταξη αυτού του προγράμματος θα χρειαστούμε έναν μετρητή **i** που θα παίρνει τιμές μέχρι το **n,** μια μεταβλητή **s** για το αντίστοιχο άθροισμα και έναν βρόχο **for** για την διαδοχική πρόσθεση της μεταβλητής **s.**

Πρόγραμμα **4.7**

```
program ginomeno_dianismaton;

const max = 10;

var a,b:array [1..max] of real;

i,n:integer;

s:real;

begin

s:=0;

writeln (‘Δώσε τον αριθμό των στοιχείων των διανυσμάτων’);

readln (n);

while (n<0) or (n>max) do

begin

writeln (‘Οχι σωστά δεδομένα’);

readln (n);

end;

writeln (‘Δώσε στοιχεία πρώτου διανύσματος: ’);

for i:=1 to n do

readln (a[i]);

writeln (‘Δώσε στοιχεία δεύτερου διανύσματος’);
```

```
for i:=1 to n do
readln (b[i]);
for i:=1 to n do
 s:=s+a[i]*b[i];
writeln (‘Το ζητούμενο γινόμενο είναι: ’,s);
end.
```

Να γραφεί πρόγραμμα σε γλώσσα Pascal το οποίο να διαβάζει ένα πίνακα ακέραιων αριθμών a, δύο διαστάσεων 3 x 3 και να υπολογίζει και να τυπώνει το μέσο όρο, το άθροισμα και το γινόμενο των στοιχείων του πίνακα a, καθώς και τα στοιχεία του πίνακα κατά σειρές και στήλες.

Οι μεταβλητές που θα χρησιμοποιήσουμε για την σύνταξη του προγράμματος είναι**: i,j** για τις γραμμές και τις στήλες αντίστοιχα του πίνακα**, s** για το άθροισμα**, g** για το γινόμενο οι οποίες θα είναι **integer** και **mo** για τον μέσο όρο που θα είναι **real.**

Πρόγραμμα **4.8**

```
program pinakas_3x3;
var i,j,s,g:integer;
mo:real;
a:array [1..3,1..3] of integer;
begin
for i:=1 to 3 do
for j:=1 to 3 do
begin
write (‘Δώσε α[‘ ,i, ’ ‘ ,j, ’]= ’);
read (a[i,j]);
end;
```

```
s:=0;
g:=1;
for i:=1 to 3 do
for j:=1 to 3 do
begin
s:=s+a[i,j];
g:=g*a[i,j];
end;
mo:=s/9;
for i:=1 to 3 do
begin
for j:=1 to 3 do
write (a[i,j],' ');
writeln;
end;
writeln (‘Το άθροισμα είναι : ’,s);
writeln (‘Το γινόμενο είναι : ’,g);
writeln (‘Ο μέσος όρος είναι : ’,mo:5:2);
end.
```

Κεφάλαιο 5

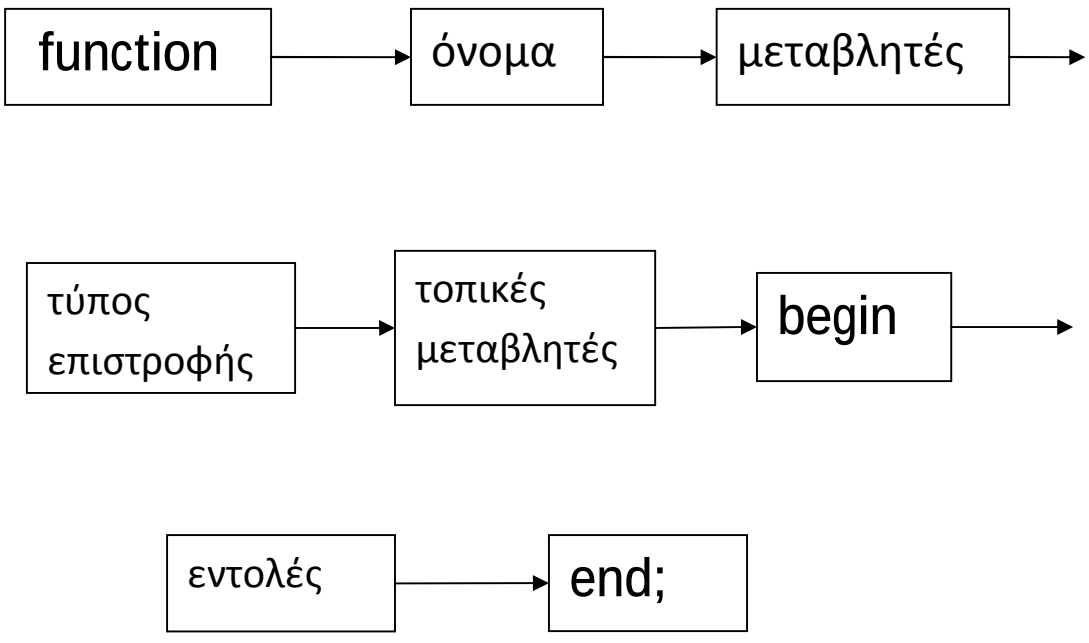
Διαδικασίες (procedures) και συναρτήσεις (functions).

Όταν έχουμε ένα πρόγραμμα όπου μία ομάδα εντολών πρέπει να επαναλαμβάνεται σε διάφορα σημεία του προγράμματος τότε θα ήταν καλύτερο να γραφεί αυτή η ομάδα εντολών μια φορά και να την καλούμε οπότε θέλουμε με μία κατάλληλη εντολή από το κυρίως πρόγραμμα. Αυτές οι ομάδες εντολών λέγονται **υποπρογράμματα (subprograms).** Τέτοιες ομάδες έχουμε δύο ειδών. Τις διαδικασίες και τις συναρτήσεις.

Συναρτήσεις **(functions)**

Πλην των καθορισμένων συναρτήσεων που διαθέτει η **pascal** όπως **exp, sin, ln,** κ.λπ. μας παρέχεται η δυνατότητα να δημιουργήσουμε και ένα πλήθος νέων συναρτήσεων που κάθε φορά θα χρειαζόμαστε μέσω της διαθέσιμης εντολής της **pascal function.**

Οι συναρτήσεις **(**αντίθετα με τις διαδικασίες**)** επιστρέφουν μια μόνο τιμή στο κυρίως πρόγραμμα μέσω του ονόματος τους. Για να ορίσουμε μια συνάρτηση ακολουθούμε την εξής συντακτική πορεία.



Η γενική μορφή της συνάρτησης είναι η εξής**:**

function όνομα συνάρτησης **(**λίστα παραμέτρων**):** τύπος της συνάρτησης

begin

εντολές της συνάρτησης

end;

begin

κυρίως πρόγραμμα

end.

Πιο αναλυτικά έχουμε**:**

function yp(a1: b1:, a2: b2,...,an: bn): b;

όπου **yp:** το όνομα της συνάρτησης

a1, a2, an: είναι οι παράμετροι της συνάρτησης **(**δηλαδή θα μπορούσαν να ήταν οι ανεξάρτητες μεταβλητές της συνάρτησης**).**

b1, b2, bn: είναι οι αντίστοιχοι τύποι των παραμέτρων **(real, integer** κ.πλ**.)**

b: ο τύπος της συνάρτησης που μεταφέρετε στο κυρίως πρόγραμμα **(real, integer** κ.πλ**.)**

Για αποσαφήνιση των ανωτέρω θα παραθέσουμε ένα απλό παράδειγμα**.**

Να γραφεί πρόγραμμα που να υπολογίζει την δύναμη ενός αριθμού.

Έστω ότι έχουμε την $\mathbf{y=a^x}$ **.** Στο τμήμα των δηλώσεων θα έχω τις εξής μεταβλητές**: a** για την βάση**, x** για τον εκθέτη και **y** για τη δύναμη**.**

Για την **function** θα της δώσω το όνομα **y,** τις παραμέτρους **base (**βάση**)** και την **exponent (**εκθέτης**)** με τύπο παραμέτρων να είναι

π.χ. **real** και το επιστρεφόμενο αποτέλεσμα της συνάρτησης να ειναι **real.** Άρα έχουμε**:**

Πρόγραμμα **5.1**

```
program dinami;
var a,x,yp:real;
function y (base, exponent:real):real;
begin
if base >0 then
y:= exp (exponent*ln(base));
else
y:=-1;
end;
begin
writeln (‘δώσε αριθμό της βάσης a: ’);
readln (a);
writeln (‘δώσε τον εκθέτη x: ’);
readln (x);
yp:=y(a,x);
writeln (a, ‘U’, x,’=’ ,yp);
end.
```

Το πρόγραμμα θα μας ζητήσει να δώσουμε τη βάση **a** καθώς και τον εκθέτη **b** για να υπολογίσει το αποτέλεσμα της δύναμης**.**

Θα ακολουθήσει ένα πρόγραμμα που θα υπολογίζει την παράγωγο της συνάρτησης.

Είναι γνωστό από τα μαθηματικά οτι η παράγωγος μιας συνάρτησης **y** σε ένα σημείο **x** ορίζεται σαν το όριο του παρακάτω λόγου**:**

y'(x)=lim(y(x+h)-y(x))/h όταν το **h** τίνει στο **0.**

Υποθέτουμε πραγματική συνάρτηση **y** πραγματικής μεταβλητής **x.**

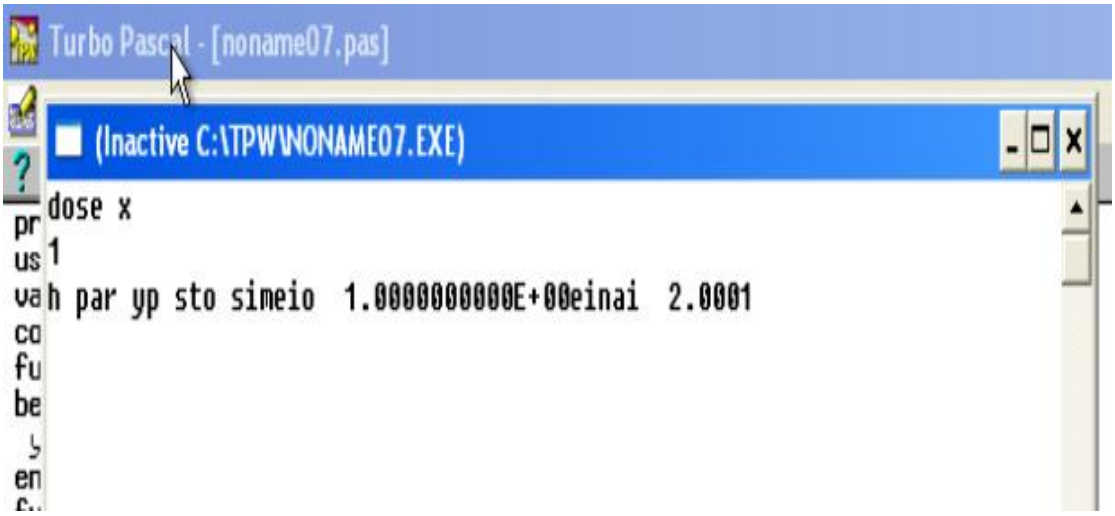
Στο τμήμα των δηλώσεων θα λάβουμε μια μεταβλητή με το όνομα **x** που θα είναι **real (**θα αντιπροσωπεύει την ανεξάρτητη μεταβλητή**)** και μια σταθερά **h** με πολύ μικρή τιμή **h=0.0001.** Όσο πιο μικρή είναι η τιμή της σταθεράς **h** τόση μεγαλύτερη ακρίβεια θα έχει ο υπολογισμός της παραγώγου**.** Με **yp** θα ονομάσουμε την παράγωγο της συνάρτησης **y.**

Το πρόγραμμα θα είναι το εξής**.**

Πρόγραμμα **5.2**

```
program paragogos;
var x:=real;
const h=0.0001;
function y(x:real):real;
begin
y:=sqr(x)-9;
end;
function yp(x:real):real;
begin
yp:=(y(x+h)-y(x))/h;
end;
begin
writeln ('δώσε x');
readln (x);
writeln('η παράγωγος yp στο σημείο',x , 'είναι ', yp (x):6:4);
end.
```

Το αποτέλεσμα της παραγώγου στο σημείο **x=1** θα είναι **2** όπως φένεται παρακάτω**.**



Να τονίσουμε εδώ ότι για κάθε συνάρτηση **y** θα πρέπει να λαμβάνουμε υπ όψιν και τους σχετικούς περιορισμούς όσον αφορά τις τιμές του **x.** Για τριγωνομετρικές συναρτήσεις το **x** πρέπει να δοθεί σε **rad.**

Θα αναπτύξουμε ένα πρόγραμμα πιο σύνθετο από το προηγούμενο**.**

Να γραφεί πρόγραμμα που να υπολογίζει τη ρίζα μιας συνάρτησης με τη βοήθεια του τύπου Newton- Raphson που είναι:

$$x_{n+1} = x_n - \frac{f(xn)}{f'(xn)}$$

Όπου είναι προφανές ότι το πρόγραμμα πλην των άλλων θα πρέπει να υπολογίζει και την παράγωγο της συνάρτησης **f(x).**

Σχόλιο**:** Εντολή **type.**

Όπως είδαμε στα προηγούμενα κεφάλαια**,** όλες οι μεταβλητές που χρησιμοποιούνται σε ένα πρόγραμμα δηλώνονται για το είδος τους **(real, integer** κ**.**λπ**.)** στο τμήμα των δηλώσεων του προγράμματος με την εντολή **var.** Όταν έχουμε όμως πιο σύνθετες μεταβλητές μπορούμε να χρησιμοποιήσουμε την εντολή **type.**

Σχετικά με την σύνταξη του προγράμματος έχουμε**.**

- Ορίζουμε με την εντολή **type** ένα νέο τύπο μεταβλητής και την ονομάζω **F** και είναι πραγματική συνάρτηση**.**
- Μέσω μιας **function** θα ορίσουμε την συνάρτηση της παραγώγου της **f** και θα την ονομάσουμε **fpar.**
- Τον τύπο της παραγώγου **(f(x+h)-f(x))/h** θα τον αναθέσουμε στην προαναφερόμενη μεταβλητή **fpar.**
- Για το **h** του τύπου της παραγώγου θα του δώσουμε μα πολύ μικρή τιμή ώστε να τείνει στο μηδέν**.**
- Θα ορίσουμε την επόμενη **function** με το όνομα **newrap** με παραμέτρους τα **f** και **x1** που θα είναι **real** και θα εκφράζει τον τύπο των **Newton- Raphson.**
- Η τελευταία **function** θα την ονομάσουμε **gx** και θα εκφράζει την συνάρτηση της οποίας θέλουμε να βρούμε τη ρίζα**.**

Με τις ανωτέρω παρατηρήσεις **–** διευκρινίσεις το πρόγραμμα θα είναι το εξής**.**

Πρόγραμμα **5.3**

```
program Newrap1;

uses wincrt;

{$F+}

const h=1e-8;

type

F = function (x:real) :real;

function fpar(f:F, x:real):real;

begin

fpar:= (f(x+h)-f(x))/h;

end;

function newrap (f:F, x1:real):real;

var x0:real;

begin

repeat
```

```
x0:=x1;
x1:=x0-f(x0)/fpar(f,x0);
until abs(x1-x0)<h;
newrap:=x1;
end;
function gx(x:real):real;
begin
gx:=sqr(x)-3*x+2;
end;
begin
writeln ('i riza tis eksisosis einai: ',newrap(gx,6):4:4);
end.
```

Σημείωση

Στην **Turbo Pascal 7.0** πρέπει στην αρχή του προγράμματος να δοθεί το σχόλιο **{$F+}** για να υποστηριχτεί η λειτουργία αυτή**.**

Να γραφεί πρόγραμμα που να υπολογίζει το ορισμένο ολοκλήρωμα μιας συνάρτησης με τη μέθοδο του τραπεζίου.

Θα λάβουμε υπ όψιν τα εξής**:**

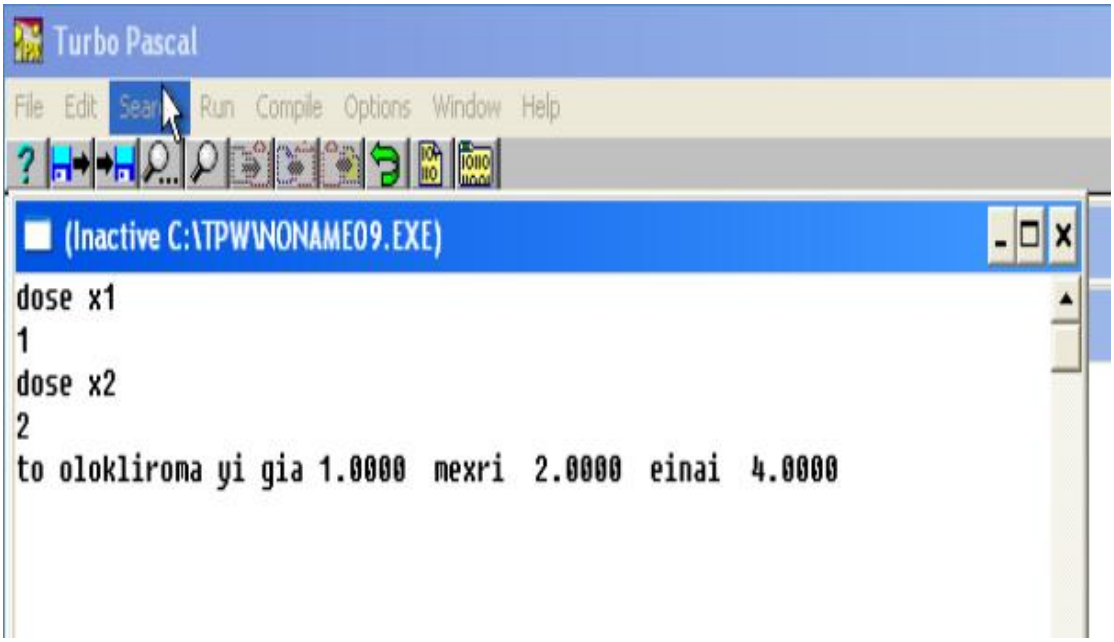
- Για ένα τραπέζιο βάσεων **a, b** που απέχουν απόσταση **c** το εμβαδό είναι**:** Ε**=0.5(a+b)c**
- Το ορισμένο ολοκλήρωμα μιας συνάρτησης **y(x)** από **x1** έως **x2 (x1<x2)** εκφράζει το εμβαδό που περικλείει η καμπύλη **y(x)** με τον άξονα των **x.**
- Ορίζουμε μεταβλητές **x1, x2, x** να είναι **real.**
- Καλούμε την πρώτη **function** με όνομα **y** και παράμετρο **x** να είναι και οι δύο **real** και θα τις αναθέσουμε την συνάρτηση της οποίας θέλουμε να βρούμε το ολοκλήρωμα **(**εδώ θα είναι η **exp(x)).**

- Καλούμε τη δεύτερη **function** με το όνομα **yi (integrated)** που θα της ανατεθεί ο τύπος του ολοκληρώματος **(**εμβαδό τραπεζίου**).**

Πρόγραμμα **5.4**

```
program olokliroma_1;

uses wincrt;

var x1,x2,x:real;

function y(x:real):real;

 begin

 y:=2*x+1;

 end;

function yi(x:real):real;

 begin

 yi:=(0.5)*(y(x1)+y(x2))*(x2-x1);

 end;

 begin

 writeln('dose x1');

 readln(x1);

 writeln('dose x2');

 readln(x2);

 writeln('to olokliroma yi gia ',x1:4:4,  '  mexri  ' , x2:4:4, '  einai  ', yi(x):6:4);

 end.
```

Τα αποτελέσματα από την εκτέλεση του προγράμματος φαίνονται παρακάτω**.**



Σχόλιο: Το παραπάνω πρόγραμμα παρουσιάζει σοβαρά μειονεκτήματα αφού ουσιαστικά μπορεί να ολοκληρώσει μόνο γραμμικές συναρτήσεις. Στην ουσία δηλαδή υπολογίζεται το εμβαδό ενός και μοναδικού τραπεζίου. Τι γένεται όμως στις περιπτώσεις που έχουμε συναρτήσεις με τυχαία, μη γραμμική καμπύλη; Το επόμενο πρόγραμμα ασχολείται ακριβός με αυτήν την περίπτωση.

Να γραφει πρόγραμμα που να υπολογίζει το ορισμένο ολοκλήρωμα της συνάρτησης $f(x)=e^{-x}/x$. Με όρια ολοκλήρωσης απο a=1 έως b=2.

Για την σύνταξη του ανωτέρω προγράμματος θα λάβουμε υπ ‘όψιν τα εξής:

- Θα χωρίσουμε το διάστημα από **a** έως **b** σε **n** τμήματα, με $n=2^k$ όπου **k=0,1,2,3,4,5,6,7.** Άρα το **maxn** θα είναι ισο με $2^7=256$.
- Στο τμήμα των δηλώσεων θα έχω τις σταθερές **a,b,maxn** καθώς και τις μεταβλητές **h** όπου $h=1/2^k$, **ivalue** που θα είναι η τιμή του αποτελέσματος της ολοκλήρωσης. Οι μεταβλητές αυτές θα είναι **real.** Τελευταία είναι η μεταβλητή **n** που θα είναι **integer.**

Πρόγραμμα **5.5**

```
program oloklíroma_2 ;

const a=1; b=2;
```

```
maxn=256;
var h, ivalue :real;
n:integer;
function f(x:real):real;
begin
f:=exp(-x)/x;
end;
procedure trapint;
begin
writeln(‘ n     h     invalue  ‘);
n:=1
repeat
trapint (n,ivalue);
h:=(b-a)/n;
writeln(h:10, n:6, ivalue:18:12);
n:=2*n
until
n<maxn
end.
```

Ενδεικτικά μετα το τρέξιμο του προγράμματος θα έχουμε τα εξής αποτελέσματα:

n	h	ivalue
1	1.00E+00	0.217773541395
2	5.00E-0.1	0.183263490747
4	2.50E-0.1	0.173757553810
8	1.25E-0.1	0.171307407474
16	6.25E-0.2	0.170689769993
32	3.12E-0.2	0.170535032321

64	**1.56E-0.2**	**0.170496327227**
128	**7.81E-0.3**	**0.170486649659**
256	**3.91E-0.3**	**0.170484230186**

Όπως παρατηρούμε καθώς αυξάνεται ο αριθμός **n** δηλαδή ο αριθμός των διαστημάτων ολοκλήρωσης εντός της περιοχής **b-a** τόσο αυξάνεται και η ακρίβεια του αποτελέσματος της ολοκλήρωσης**.** Δηλαδή η τελευταία τιμή του πίνακα μας δίνει το καλύτερο **ivalue.**

Παράρτημα

Οι κώδικες χαρακτήρων

Όταν εισάγονται ή αποθηκεύονται χαρακτήρες στον υπολογιστή, χρησιμοποιούνται ειδικοί κώδικες για την παράστασή τους.

Η κωδικοποίηση γίνεται με τη χρήση συνδυασμών του **0** και του **1**, οπότε κάθε χαρακτήρας αντιστοιχίζεται με μια μοναδική διαφορετική ακολουθία δυαδικών ψηφίων. Παράλληλα η ακολουθία αυτή στο δυαδικό σύστημα παριστάνει μια αριθμητική τιμή. **Η** τιμή αυτή, που μπορεί να εκφραστεί εκτός από το δυαδικό και στο οκταδικό, στο δεκαδικό ή στο δεκαεξαδικό σύστημα αρίθμησης, λέγεται **τιμή του κωδικού (code value)**. Οι κυριότεροι κώδικες χαρακτήρων που χρησιμοποιούνται είναι:

Κώδικας ASCII

Ο **ASCII (American Standard Code for Information Interchange)** δημιουργήθηκε χάρη στην υποστήριξη του Εθνικού Αμερικανικού Ινστιτούτου Προτύπων **ANSI (American National Standard Institute)**, σε μια προσπάθεια να υπάρξει ένας κοινός κώδικας για την ανταλλαγή των δεδομένων μεταξύ υπολογιστών καθώς και για την αποθήκευσή τους. Γι' αυτό το λόγο έχει υιοθετηθεί σχεδόν από όλους τους κατασκευαστές μικροϋπολογιστών και χρησιμοποιείται ευρύτατα.

Αρχικά στον κώδικα **ASCII** χρησιμοποιούνταν **7 bit** για την παράσταση των χαρακτήρων και ένα **bit**, το **8**ο, για έλεγχο ορθότητας κατά τη μεταφορά στοιχείων. Το **bit** αυτό ονομάστηκε **ψηφίο ισοτιμίας (parity bit)**. Με την κωδικοποίηση αυτή δίνεται η δυνατότητα να παραστήσουμε **128 (=2^7)** διαφορετικούς χαρακτήρες, ως αποτέλεσμα των **128** διαφορετικών συνδυασμών από **0** και **1**.

Δεκαδική τιμή	Χαρακτήρας	Δεκαδική τιμή	Χαρακτήρας	Δεκαδική τιμή	Χαρακτήρας
32		64	@	96	'
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	P
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	|
62	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_	127	_

Βιβλιογραφία

1)Εισαγωγή στην **Pascal.** Σπύρος Γερούλης. Έκδόσεις **SPIN.** Αθήνα **2004.**

2)Standard Pascal. Απ. Ατματζίδης, Μ. Γλαμπεδάκης. Εκδόσεις ΄ΙΩΝ΄ **2000.**

3)Παραδείγματα απλών προγραμμάτων σε **Turbo Pascal.** Σημειώσεις εργαστηρίων. Σχολή Ηλεκτρολογίας ΑΤΕΙ Πατρών.

4)Turbo Pascal 6.7 και εφαρμογές. Π. Παπαζόγλου. Εκδόσεις ΙΩΝ.

5)Η γλώσσα προγραμματισμού **Pascal.** Άρης Ν. Μπακάλης. Σχολή διοίκησης και οικονομίας. ΑΤΕΙ Πάτρας **2009.**

6)Προγραμματισμός Η/Υ με **Pascal,** σημειώσεις Γ. Δασκαλίτσας – Μ. Κωνσταντινίδου.**2014** Δημόσιο ΙΕΚ Έδεσσας.

7) http://dide.flo.sch.gr/Plinet/Tutorials/Tutorials-Pascal.html

8) http://ebooks.edu.gr/modules/ebook/show.php/DSGL-C127/577/3739,16396/

9) https://el.wikipedia.org/wiki/ASCII

10) https://jhnginis.wordpress.com/

11) Προγραμματισμός Ι Ασκήσεις. Τζάλλας Αλέξανδρος, Καθηγητής Εφαρμογών Τμ. Μηχανικών Πληροφορικής Τ.Ε., Άρτα, Μάιος **2015.**

12) Introduction to Numerical Computation in Pascal
Από τον/την **DEW/JAMES**