



Τμήμα Τεχνολογίας Πληροφορικής & Τηλεπικοινωνιών
Σχολή Διοίκησης και Οικονομίας - ΑΤΕΙ Ηπείρου

Προγραμματισμός Ι

Η Γλώσσα Προγραμματισμού PASCAL

Επιμέλεια: Κωλέτσου Ευτυχία

Ακαδημαϊκό Έτος 2009-2010



Περιεχόμενα

Η Γλώσσα Προγραμματισμού PASCAL Μέρος Α΄ Περιβάλλον προγραμματισμού, Βασικές εντολές, Εντολές επιλογής και αποφάσεων, Εντολές επανάληψης, Υποπρογράμματα	3 -
Εισαγωγή	4 -
TURBO PASCAL Version 7.0	5 -
Αλφάβητο της PASCAL	10 -
Τύποι Δεδομένων στην Pascal.....	11 -
Τελεστές της Pascal.....	12 -
Δομή ενός προγράμματος σε Pascal.....	12 -
Εντολές της Pascal.....	13 -
Εντολές εισόδου - εξόδου	13 -
Δομή Ακολουθίας	14 -
Η εντολή - IF.....	15 -
Η εντολή - CASE	17 -
Εντολές Επανάληψης (Η εντολή FOR).....	18 -
Εντολές Επανάληψης (Η εντολή WHILE - DO)	20 -
Εντολές Επανάληψης (Η εντολή REPEAT - UNTIL).....	20 -
Διαδικασίες Και Συναρτήσεις	21 -
Διαδικασίες - Procedures	22 -
Συναρτήσεις - Functions.....	24 -
Μαθηματικές συναρτήσεις βιβλιοθήκης της PASCAL	25 -
Ομοιότητες - Διαφορές Διαδικασιών και Συναρτήσεων.	26 -
Προχωρημένα θέματα στην PASCAL Μέρος Β΄ Αρχεία	27 -
Αρχεία	28 -
Αρχεία Κειμένου	28 -
Αρχεία Με Τύπο.....	34 -
Προχωρημένα θέματα στην PASCAL Μέρος Γ΄ Πίνακες, Εγγραφές	36 -
Πίνακες.....	37 -
Αναζήτηση Στοιχείων Σε Πίνακα: Σειριακή Αναζήτηση (Serial Search)	38 -
Ταξινόμηση Στοιχείων Σε Πίνακα: Ταξινόμηση Φυσαλίδας (Bubble Sort)	39 -
Εγγραφές - Records.....	42 -
Χρήσιμες αναφορές για την PASCAL Μέρος Δ΄ Βιβλιογραφία	45 -
Βιβλιογραφία.....	45 -



Η Γλώσσα Προγραμματισμού PASCAL

Μέρος Α΄

*Περιβάλλον προγραμματισμού,
Βασικές εντολές,
Εντολές επιλογής και αποφάσεων,
Εντολές επανάληψης,
Υποπρογράμματα*



Εισαγωγή

Η **Pascal** αναπτύχθηκε το **1971** από τον **Niklaus Wirth** και ανήκει στην κατηγορία των γλωσσών υψηλού επιπέδου, αφού είναι εύκολα κατανοητή από τον άνθρωπο, αλλά όχι και από τον ηλεκτρονικό υπολογιστή (**H/Y**).

Η **Pascal** είναι μια γλώσσα γενικής χρήσης, η οποία είναι κατάλληλη τόσο για την εκπαίδευση όσο και για τη δημιουργία ισχυρών προγραμμάτων κάθε τύπου. Χαρακτηριστικό της **Pascal** είναι η καταλληλότητα για τη δημιουργία δομημένων προγραμμάτων.

Οι γλώσσες προγραμματισμού διακρίνονται σε γλώσσες μηχανής, συμβολικές γλώσσες ή γλώσσες χαμηλού επιπέδου και σε γλώσσες υψηλού επιπέδου.

Αρχικά για να μπορέσει ο **H/Y** να εκτελέσει μια οποιαδήποτε λειτουργία, έπρεπε να δοθούν κατευθείαν οι κατάλληλες οδηγίες από **0** και **1**, δηλαδή εντολές σε μορφή κατανοητή από τον **H/Y** αλλά ακατανόητες για τον άνθρωπο. Ο τρόπος αυτός ήταν επίπονος και ελάχιστοι μπορούσαν να τον υλοποιήσουν, αφού απαιτούσε βαθιά γνώση του υλικού και της αρχιτεκτονικής του υπολογιστή.

Στη συνέχεια έχουμε τις συμβολικές γλώσσες ή γλώσσες χαμηλού επιπέδου. Οι εντολές θα μεταφραστούν από τον **H/Y** σε μια ακολουθία δυαδικών ψηφίων (**0** και **1**) και στη συνέχεια μπορούν να εκτελεστούν. Τη μετάφραση των εντολών την αναλαμβάνει ένα ειδικό πρόγραμμα, ο συμβολομεταφραστής (**assembler**).

Οι γλώσσες υψηλού επιπέδου είναι εύκολα κατανοητές από τον άνθρωπο αφού βασίζονται σε λέξεις της καθομιλούμενης γλώσσας. Τη μετάφραση των εντολών την αναλαμβάνει ο μεταγλωττιστής (**compiler**).

Μια γλώσσα προγραμματισμού προσδιορίζεται από το αλφάβητό της, το λεξιλόγιό της, τη γραμματική της και τη σημασιολογία της. Αλφάβητο καλείται το σύνολο των στοιχείων που χρησιμοποιείται από τη γλώσσα προγραμματισμού. (χαρακτήρες που επιτρέπονται στις εντολές του προγράμματος) Το λεξιλόγιο αποτελείται από ένα υποσύνολο όλων των ακολουθιών που δημιουργούνται από τα στοιχεία του αλφαβήτου. (λέξεις που χρησιμοποιούνται για την έκφραση των εντολών). Η γραμματική αποτελείται από το τυπικό ή τυπολογικό και το συντακτικό. Τυπικό είναι το σύνολο των κανόνων που ορίζει τις μορφές με τις οποίες μια λέξη είναι αποδεκτή. Συντακτικό είναι το σύνολο των κανόνων που καθορίζει τη νομιμότητα της διάταξης και της σύνδεσης των λέξεων μιας γλώσσας.

Η σημασιολογία είναι το σύνολο των κανόνων που καθορίζει το νόημα των λέξεων. Η διαδικασία προγραμματισμού των **H/Y** περιλαμβάνει τα εξής στάδια:

- è **Ανάλυση**: κατανόηση του δεδομένου προβλήματος και κατάστρωση της διαδικασίας επίλυσής του.
- è **Σχεδιασμός** του αλγορίθμου επίλυσης: μπορεί να εκφραστεί με φυσική γλώσσα με βήματα, διαγράμματα ροής, ψευδοκώδικα, κλπ.
- è **Συγγραφή** του προγράμματος: χρήση ειδικών προγραμμάτων (**editors**), που παρέχουν λειτουργίες αντιγραφής, μεταφοράς, αναζήτησης κειμένου, αποθήκευση ή εκτύπωση του προγράμματος. Το αποτέλεσμα είναι ο πηγαίος κώδικας (**source code**) του προγράμματος, το οποίο δεν είναι άμεσα κατανοητό από τον **H/Y**.
- è **Μεταγλώττιση (compilation)**: του πηγαίου κώδικα σε γλώσσα μηχανής. Αν ο πηγαίος κώδικας δε είναι συντακτικά σωστός, ο μεταγλωττιστής (**compiler**) εμφανίζει μηνύματα λάθους. Αν είναι συντακτικά σωστός παράγει τον εκτελέσιμο κώδικα.
- è **Διασύνδεση** μονάδων προγράμματος από διαφορετικά αρχεία.
- è **Εκτέλεση** του προγράμματος.
- è **Έλεγχος** του προγράμματος με διάφορα δεδομένα, ώστε να εξακριβωθεί η ορθή ή λανθασμένη λειτουργία του, και διόρθωση των λογικών λαθών του προγράμματος, δηλαδή λαθών που οδηγούν σε μη αναμενόμενα αποτελέσματα, με τη βοήθεια ειδικών προγραμμάτων που λέγονται εκσφαλματωτές (**debuggers**).



TURBO PASCAL Version 7.0

Για να ενεργοποιήσουμε τη γλώσσα προγραμματισμού **Pascal** θα πρέπει να πάμε στο κατάλογο που περιέχει το αρχείο με το όνομα **TURBO.EXE** και να πατήσουμε πάνω σε αυτό είτε διπλό κλικ είτε ένα κλικ και μετά το πλήκτρο **ENTER**. Θα δούμε να μας εμφανίζεται στην οθόνη μας το περιβάλλον της **Turbo Pascal 7**.

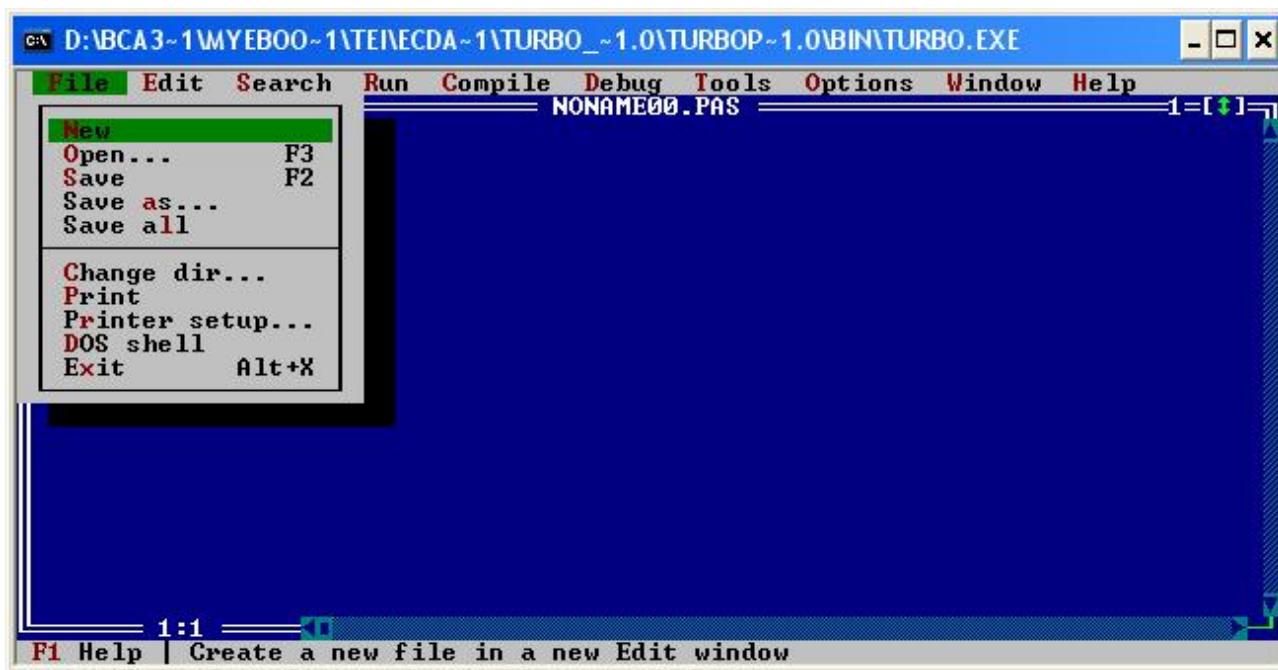


Στην κορυφή της οθόνης φαίνονται οι επιλογές **File, Edit, Search, Run, Compile, Debug, Tools, Options, Window** και **Help**. Πατώντας το πλήκτρο **F10** θα βρεθούμε στο μενού επιλογών που βρίσκεται στην κορυφή της οθόνης και με τα πλήκτρα του δρομέα αριστερό ή δεξί ή μπορούμε να μετακινηθούμε στην κάθε επιλογή και να την ενεργοποιήσουμε. Ένας άλλος τρόπος είναι να πατήσουμε το πλήκτρο **<Alt>** και το πρώτο γράμμα της επιλογής που θέλουμε να ενεργοποιήσουμε. Αν στον υπολογιστή που έχουμε υπάρχει ποντίκι μπορούμε να ενεργοποιήσουμε τις πιο πάνω επιλογές του μενού με το δρομέα του ποντικιού.

Θα παρατηρήσουμε ότι στην πάνω μεριά στο κέντρο υπάρχει το όνομα **NONAME00.PAS** επειδή δεν έχουμε δώσει ακόμα κάποιο όνομα στο αρχείο μας. Τώρα είμαστε έτοιμοι να γράψουμε το πρώτο μας πρόγραμμα σε γλώσσα **Pascal**. Γράψτε το πιο κάτω πρόγραμμα.

```
PROGRAM MY_FIRST_PROG;  
BEGIN  
    WRITELN('HELLO WORLD!!');  
END.
```

Για να σώσουμε το πρόγραμμα μας είτε στο σκληρό δίσκο (**C:**) είτε πηγαίνετε το βέλος του ποντικιού στη λέξη **File** και πατήστε ένα κλικ, είτε έχοντας πατημένο το πλήκτρο **ALT** πατήστε και το πλήκτρο **F**, ή πατήστε το πλήκτρο **F10** και πηγαίνετε στην επιλογή **File** και πατήστε το πλήκτρο **Enter**. Θα δείτε να εμφανίζεται η παρακάτω οθόνη.



το μενού **File** περιέχει τις πιο κάτω επιλογές:

New → σας επιτρέπει να δημιουργήσετε ένα καινούργιο πρόγραμμα.

Open → σας επιτρέπει να ανοίξετε κάποιο παλιό σας αποθηκευμένο πρόγραμμα

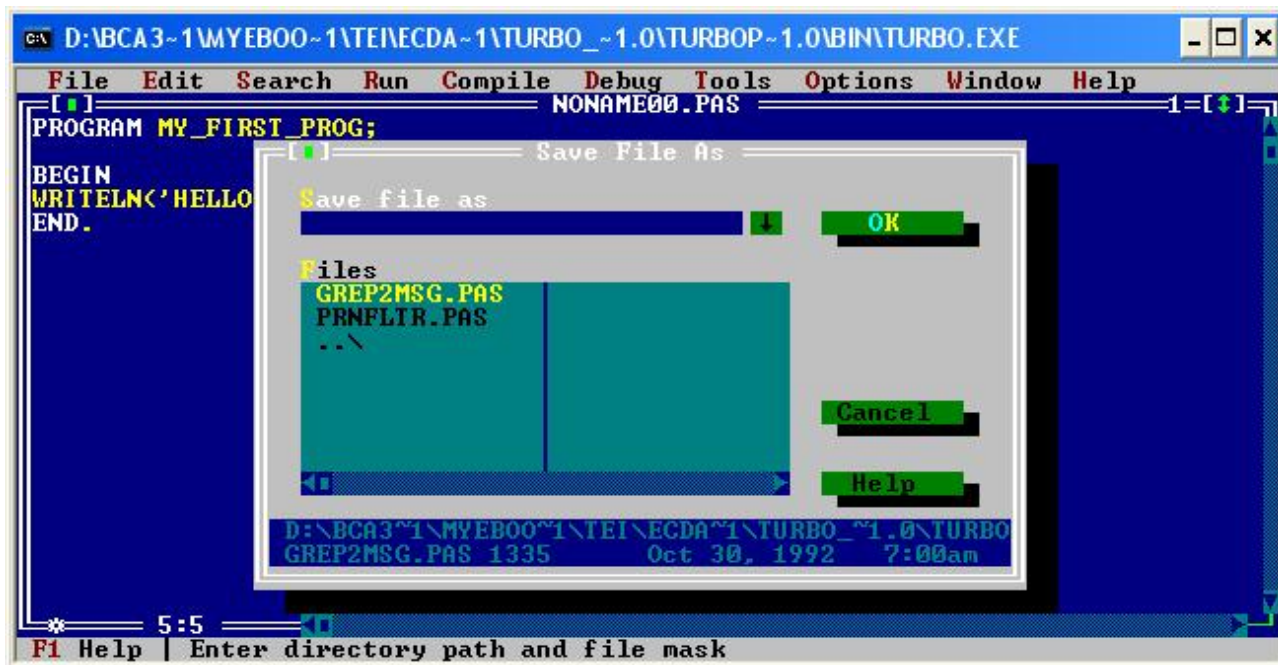
Save → σας επιτρέπει να σώσετε το τρέχον πρόγραμμά σας με ένα όνομα της επιλογή σας.

Save as → σας επιτρέπει να σώσετε το τρέχον πρόγραμμά σας και με κάποιο άλλο όνομα.

Print → σας επιτρέπει να εκτυπώσετε το τρέχον πρόγραμμα σας

Exit → σας επιτρέπει να φύγετε από το περιβάλλον της **Turbo Pascal 7**.

Μετακινηθείτε με το πάνω ή το κάτω βελάκι στην επιλογή **Save** και πατήστε **Enter** τότε θα εμφανιστεί στον Η/Υ σας η πιο κάτω οθόνη:



Στη περιοχή που λέει **Save file as** γράψτε **C:ASK1-1** και πατήστε με το ποντίκι σας το **OK**. Θα παρατηρήσετε ότι στην πάνω μεριά στο κέντρο το όνομα **NONAME00.PAS** έχει αλλάξει σε



C:ASK1-1.PAS όπου το **C:** είναι του σκληρού δίσκου και το **ASK1-1.PAS** είναι το όνομα του αρχείου **ASK1-1** που δώσαμε με την επέκταση **PAS** (πρόγραμμα σε γλώσσα **PASCAL**).

Θα παρατηρήσετε ότι στο πρώτο πρόγραμμα που φτιάξαμε ξεκινήσαμε γράφοντας τη λέξη **program** και ένα όνομα της αρεσκειάς μας. Η λέξη **program** είναι δεσμευμένη λέξη (δεν μπορούμε να την χρησιμοποιήσουμε αλλού) και θα είναι η πρώτη λέξη σε κάθε πρόγραμμα μας. Ακολουθείται πάντα από ένα όνομα της επιλογή μας το οποίο θα ξεκινά πάντα με ένα γράμμα και θα ακολουθείται από συνδυασμό γραμμάτων ή αριθμών χωρίς ενδιάμεσα κενά. Αν θέλετε να χωρίσετε το όνομα χρησιμοποιήστε την κάτω παύλα (_). Αμέσως μετά το όνομα βάλτε το ;

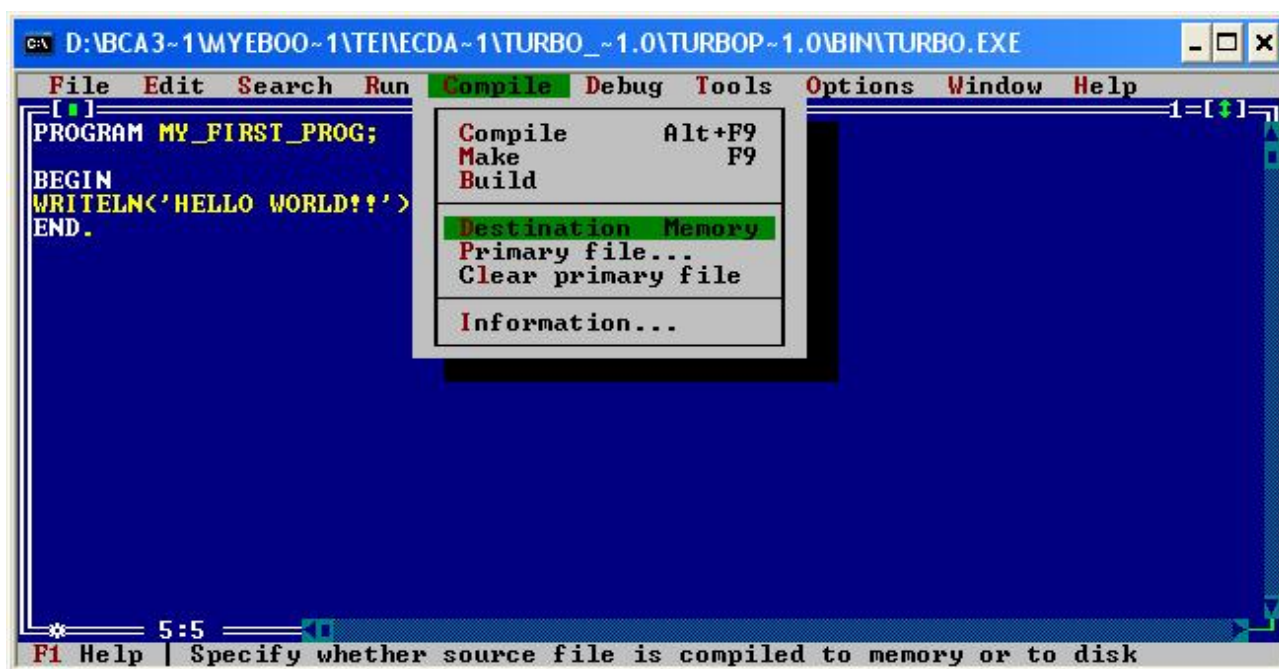
Το κυρίως μέρος του προγράμματός σας περιλαμβάνεται στις λέξεις **begin** και **end**.

Η **writeln** είναι μια ειδική λέξη που επιτρέπει να εμφανιστούν κάποια δεδομένα στην οθόνη του υπολογιστή μας. Τα δεδομένα μέσα στην παρένθεση είναι τα δεδομένα που θα εκτυπωθούν, και παρότι υπάρχουν πολλά ήδη δεδομένων που θα επιθυμούσαμε να εκτυπώσουμε, θα περιοριστούμε στο πιο απλό παράδειγμα για την ώρα. Οποιαδήποτε πληροφορία ανάμεσα στις αποστρόφους θα εκτυπωθεί ως απλό κείμενο. Η ειδική λέξη **writeln** δεν είναι δεσμευμένη λέξη αλλά είναι καθορισμένη από το σύστημα να κάνει μια ειδική δουλειά για σας, δηλαδή να εκτυπώνει μια γραμμή από δεδομένα στην οθόνη.

Παρατηρήστε το ερωτηματικό στο τέλος της γραμμής. Είναι ο διαχωριστής εκφράσεων και λείπει στον **compiler** της **Pascal** ότι η γραμμή είναι πλήρης, δηλαδή τίποτα από τα επόμενα δεν μπορεί να θεωρηθεί ως συνέχεια αυτής της έκφρασης. Η επόμενη έκφραση, είναι μια ξεχωριστή έκφραση που μπορεί να εκτελεστεί σειριακά ακολουθώντας την έκφραση της προηγούμενης γραμμής. Αυτό το πρόγραμμα θα εκτυπώσει τις δυο γραμμές κειμένου και θα τερματιστεί.

Τέλος, υπάρχει και κάποιο άλλο πολύ σημαντικό σημείο αυτού του προγράμματος: η τελεία που ακολουθεί το **end**. Η τελεία αυτή είναι σημάδι προς τον **compiler** ότι τελείωσε το πρόγραμμα του εκτελέσιμου μέρους και επίσης ότι τελείωσε η μεταγλώττιση. Κάθε πρόγραμμα **Pascal** έχει μια και μόνη τελεία η οποία θα βρίσκεται στο τέλος του προγράμματος. Σκεφτείτε ένα πρόγραμμα **Pascal** ως μια πρόταση με μια τελεία στο τέλος.

Αυτή ήταν η περιγραφή του πρώτου μας προγράμματος. Τώρα είναι καιρός να το μεταγλωττίσετε και να το τρέξετε. Πατήστε **<Alt>+C** ή με το ποντίκι σας κάντε ένα κλικ στη επιλογή **Compile**. Θα εμφανιστεί η πιο κάτω οθόνη:



Από το μενού αυτό μας ενδιαφέρουν οι επιλογές **[Compile Alt+F9]** και **[Destination.....]**.



Η επιλογή **Destination** αλλάζει αν πάμε πάνω σε αυτή και πατήσουμε **Enter**. Αλλάζει από **Destination Memory** σε **Destination Disk**. Αν αφήσουμε την επιλογή **Destination Memory**, τότε το αποτέλεσμα του **Compilation** θα πάει στην μνήμη του Η/Υ και το πρόγραμμα θα εκτελείται μέσα από το περιβάλλον της **Turbo Pascal** με την επιλογή **RUN** από το μενού που βρίσκεται στην κορυφή της οθόνης. Αν επιλέξουμε **Destination Disk** τότε το αποτέλεσμα του **Compilation** θα είναι ένα αρχείο το οποίο τώρα θα βρίσκεται στο **C:** εκεί που σώσαμε το αρχείο μας **ASK1-1.PAS** και θα έχει τον ίδιο όνομα αλλά με την επέκταση **.EXE** και θα μπορεί να εκτελείται κατευθείαν από το περιβάλλον του **DOS**.

Αφού ολοκληρωθεί η μεταγλώττιση (**Compilation**) και δεν έχει βρεθεί κανένα λάθος στο πρόγραμμά μας τότε θα πάρουμε στην οθόνη του υπολογιστή μας το πιο κάτω μήνυμα:

```

C:\ D:\BCA3~1\WYEBOO~1\TEINECDA~1\TURBO_~1.0\TURBOP~1.0\BIN\TURBO.EXE
File Edit Search Run Compile Debug Tools Options Window Help
[ ] NONAME00.PAS 1=[↑↓]
PROGRAM MY_FIRST_PROG;
BEGIN
WRITELN('HELLO WORLD!');
END.

Compiling

Main file: NONAME00.PAS
Done.

Destination: Memory    Line number:    0
Free memory:  310K    Total lines:    5

Compile successful: Press any key

* 5:5
F1 Help | Compile source file
```

ότι το **compilation** ήταν επιτυχές και να πατήσουμε ένα οποιοδήποτε πλήκτρο για να συνεχίσουμε. Τώρα είμαστε έτοιμοι να τρέξουμε το πρόγραμμά μας. Πηγαίνουμε στην επιλογή **RUN** και εμφανίζεται η πιο κάτω οθόνη:

```

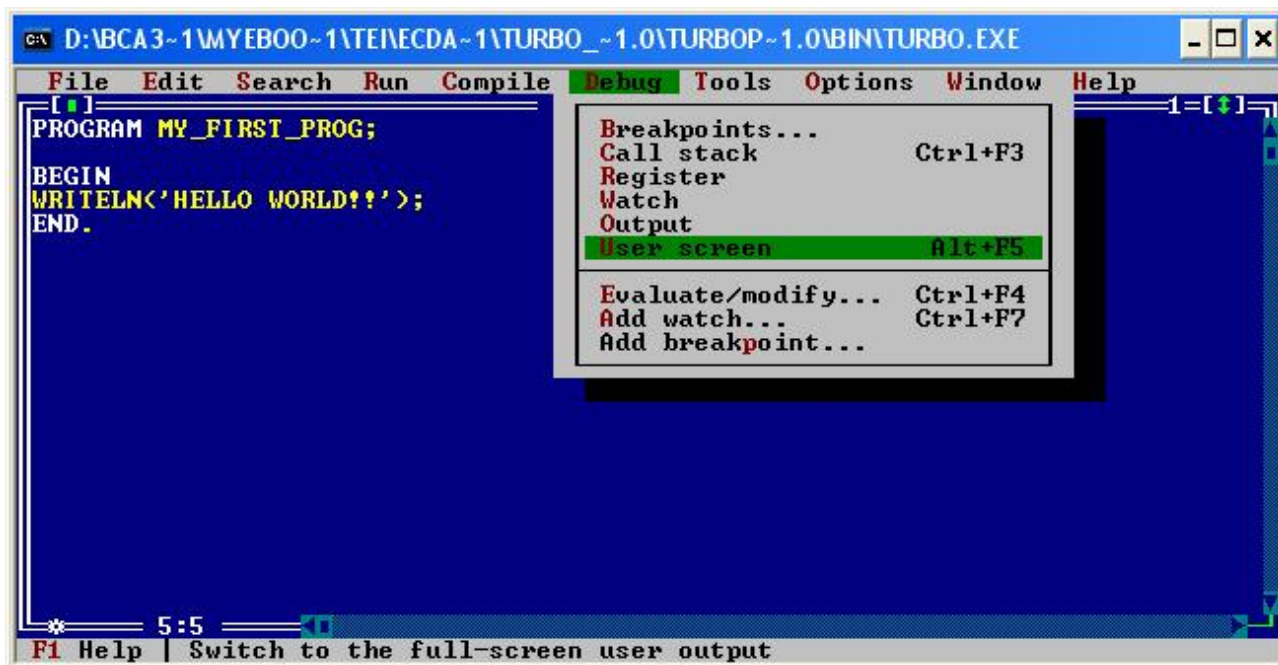
C:\ D:\BCA3~1\WYEBOO~1\TEINECDA~1\TURBO_~1.0\TURBOP~1.0\BIN\TURBO.EXE
File Edit Search Run Compile Debug Tools Options Window Help
[ ] NONAME00.PAS 1=[↑↓]
PROGRAM MY_FIRST_PR
BEGIN
WRITELN('HELLO WORL
END.

Run      Ctrl+F9
Step over  F8
Trace into F7
Go to cursor F4
Program reset Ctrl+F2
Parameters...

* 5:5
F1 Help | Run the current program
```



Πηγαίνετε στην επιλογή **[Compile Ctrl+f9]** και πατήστε **Enter**. Για να δείτε τα αποτελέσματα της εκτέλεσης του προγράμματος πηγαίνετε στην επιλογή **Debug** και πατήστε **ENTER**. Θα εμφανιστεί η πιο κάτω οθόνη:



Πηγαίνετε στη επιλογή **[User screen Alt+F5]** και πατήστε **Enter**. Θα δείτε τα αποτελέσματα της εκτέλεσης του πρώτου προγράμματός σας που είναι η πιο κάτω οθόνη:

```
Turbo Pascal Version 7.0 Copyright © 1983,92 Borland International  
HELLO WORLD!!
```

για να γυρίσετε στο περιβάλλον της **Turbo Pascal** πατήστε ένα οποιοδήποτε πλήκτρο στο πληκτρολόγιο σας.

Σημείωση: Μπορούμε να κάνουμε όλες τις πιο πάνω διαδικασίες πολύ πιο γρήγορα με τα πιο κάτω βήματα:

- Για να τρέξετε το πρόγραμμά σας: έχοντας πατημένο το πλήκτρο **Ctrl** πατήστε μια φορά το πλήκτρο **F9**
- Για να δείτε τα αποτελέσματα: έχοντας πατημένο το πλήκτρο **Alt** πατήστε μια φορά το πλήκτρο **F5**

Τώρα γράψτε τα πιο κάτω προγράμματα:

```
PROGRAM TEST1;  
BEGIN  
    WRITELN('MY NAME IS ..... ');  
    WRITELN('AND I AM A STUDENT ');  
    WRITELN('AT TEI OF EPIRUS. ');  
END.
```

```
PROGRAM TEST2;  
BEGIN  
    WRITE('MY NAME IS ..... ');  
    WRITE('AND I AM A STUDENT ');  
    WRITE('AT TEI OF EPIRUS. ');  
END.
```



Η **write** είναι μια άλλη ειδική λέξη η οποία υποχρεώνει το κείμενο να εκτυπωθεί όπως και με την **writeln**, αλλά ο κέρσορας δεν αλλάζει γραμμή στην έξοδο. Η **writeln** γράφει το προς εκτύπωση κείμενο και αλλάζει γραμμή, η **write** όχι. Το τελικό αποτέλεσμα είναι και οι τρεις γραμμές να εκτυπώνονται στην ίδια γραμμή όταν το πρόγραμμα εκτελεστεί. Προσέξτε ότι στο τέλος κάθε λέξης υπάρχει ο κενός χαρακτήρας, ώστε το τελικό αποτέλεσμα να είναι καλαίσθητο. Μεταγλωττίστε και εκτελέστε το πρόγραμμα.

Γράψτε το πιο κάτω πρόγραμμα:

```
PROGRAM TEST3;  
VAR A, B, C, D, SUM :INTEGER;  
MO : REAL;  
BEGIN  
    A:=10;  
    B:=20;  
    C:=30;  
    B:=40;  
    SUM:=(A+B+C+D);  
    MO:=SUM/4;  
    WRITELN('ΤΟ ΑΤΗΡΟΙΣΜΑ ΕΙΝΑΙ ', SUM);  
    WRITELN('Ο ΜΕΣΟΣ ΟΡΟΣ ΕΙΝΑΙ ', MO);  
END.
```

Αλφάβητο της PASCAL

Το αλφάβητο της **Pascal** περιλαμβάνει τους λατινικούς χαρακτήρες **a – z** και **A – Z**, τα ψηφία **0 – 9** και τα σύμβολα **+ - * / = < > [] . , _ () : ; ' ^ @ { } \$ #**

Η **Pascal** είναι **non case-sensitive** γλώσσα (μη ευαίσθητη σε πεζά -κεφαλαία), δηλαδή τα πεζά και τα κεφαλαία έχουν την ίδια ακριβώς σημασία. Για παράδειγμα, τα ονόματα **sum**, **Sum** και **SUM** αναφέρονται στην ίδια ακριβώς μεταβλητή.

Όπως σε κάθε γλώσσα προγραμματισμού, υπάρχουν και στην **Pascal** οι δεσμευμένες λέξεις (**reserved words**), που χρησιμοποιούνται στη σύνταξη των εντολών της **Pascal** και δεν μπορούν να χρησιμοποιηθούν για άλλο σκοπό (π.χ. για ονομασία μεταβλητών). Μερικές από αυτές είναι οι: **program, const, var, function, procedure, begin, end, if, then, else, for, to, downto, goto, do, while, repeat, until, case, of, array, record**, κ.α.

Για την επιλογή από τον προγραμματιστή ενός ονόματος στην **Pascal** (ως μεταβλητής, σταθεράς, προγράμματος, συνάρτησης, κλπ.) υπάρχουν οι εξής κανόνες:

- ο Δεν πρέπει το όνομα να είναι δεσμευμένη λέξη της **Pascal**.
- ο Επιτρέπονται μόνο αλφαριθμητικοί χαρακτήρες και ο χαρακτήρας **_**
- ο Επιτρέπονται μόνο λατινικά γράμματα.
- ο Ο πρώτος χαρακτήρας πρέπει να είναι λατινικό γράμμα ή **_** (όχι ψηφίο).

Οι σταθερές διακρίνονται σε αριθμητικές (π.χ. **12, 4.5, 2.5E-04**) και σε αλφαριθμητικές, οι οποίες περικλείονται σε αποστροφούς (π.χ. **'a', 'hello', 'kalimera', '12345', '\$@#42a'**). Για τις αλφαριθμητικές σταθερές δεν υπάρχουν περιορισμοί σχετικά με το περιεχόμενό τους (μπορεί να είναι οποιοδήποτε σύμβολο ή γράμμα σε οποιαδήποτε γλώσσα).

Καλό πάντως είναι να χρησιμοποιείτε μόνο το λατινικό πληκτρολόγιο γιατί σε περίπτωση που χρησιμοποιηθεί σε εντολή χαρακτήρας από το ελληνικό πληκτρολόγιο θα δυσκολευτείτε να εντοπίσετε που είναι το λάθος (π.χ. Το **Ι** το λατινικό είναι το ίδιο με το **I** το ελληνικό).

Σημαντικό στοιχείο για ένα πρόγραμμα είναι η ύπαρξη σχολίων, τα οποία βελτιώνουν την αναγνωσιμότητα του πηγαίου κώδικα και κάνουν τη συντήρησή του πιο εύκολη. Τα σχόλια περικλείονται σε άγκιστρα, π.χ. **{Αυτό είναι το πρώτο σχόλιο}**, ή σε παρενθέσεις με αστερίσκο,



π.χ. (*Αυτό είναι το δεύτερο σχόλιο *), και μπαίνουν οπουδήποτε στο πρόγραμμα, χωρίς περιορισμούς σχετικά με το περιεχόμενό τους.

Τύποι Δεδομένων στην Pascal

Ένας τύπος στην **Pascal**, και σε πολλές άλλες δημοφιλείς γλώσσες προγραμματισμού, καθορίζει μια μεταβλητή με τέτοιο τρόπο που κατ' αρχήν προσδιορίζει το εύρος των τιμών που είναι ικανή να αποθηκεύσει, και επίσης καθορίζει ένα σύνολο από λειτουργίες που είναι επιτρεπτό να εφαρμοσθούν πάνω στις μεταβλητές αυτού του τύπου.

Integer Όλοι οι αριθμοί στο διάστημα από **-32768** έως **32767**

Byte Όλοι οι ακέραιοι από **0** έως **255**

Real Όλοι οι πραγματικοί αριθμοί από **1E-38** έως **1E+38**

Boolean Μπορεί να έχει μόνο τιμές **TRUE** και **FALSE**

Char Όλοι οι χαρακτήρες του κώδικα **ASCII**

Shortint Όλοι οι ακέραιοι από **-128** έως **127**

Word Όλοι οι ακέραιοι από **0** έως **65535**

Longint Όλοι οι ακέραιοι από **-2147483648** έως **2147483647**

Single Όλοι οι πραγματικοί με **7** σημαντικά ψηφία

Double Όλοι οι πραγματικοί με **15** σημαντικά ψηφία

Extended Όλοι οι πραγματικοί με **19** σημαντικά ψηφία

Comp Όλοι οι ακέραιοι από **-10E18** έως **10E18**

Η **Pascal** διαθέτει ενσωματωμένους τρεις βασικούς τύπους δεδομένων, τα αριθμητικά δεδομένα, τα δεδομένα χαρακτήρα και τα λογικά δεδομένα.

Στους αριθμητικούς τύπους περιλαμβάνονται:

- § **Πραγματικοί αριθμοί:** Στην **Pascal** οι πραγματικοί αριθμοί δηλώνονται ως **real** (μια μεταβλητή αυτού του τύπου καταλαμβάνει **6 bytes** στη μνήμη), **single (4 bytes)**, **double (8 bytes)** και **extended (10 bytes)**. Όσο μεγαλύτερο χώρο καταλαμβάνει ένας πραγματικός αριθμός στη μνήμη, τόσο μεγαλύτερος θα μπορεί να είναι και μεγαλύτερη ακρίβεια να έχει.
- § **Ακέραιοι αριθμοί:** Στην **Pascal** οι ακέραιοι αριθμοί δηλώνονται ως **integer** (μια μεταβλητή αυτού του τύπου καταλαμβάνει **16 bits**, εκ των οποίων το ένα **bit** χρησιμοποιείται ως πρόσημο, άρα οι τιμές που μπορεί να πάρει είναι από **-215 = -32768** μέχρι **215 - 1 = 32767**), **shortint 98 bits** με πρόσημο και με τιμές από **-128** μέχρι **127**), **longint (32 bits** με πρόσημο και με τιμές από **-2.147.483.648** μέχρι **2.147.483.647**), **byte (8 bits** χωρίς πρόσημο και με τιμές από **0** μέχρι **255**) και **word (16 bits** χωρίς πρόσημο και με τιμές από **0** μέχρι **65535**).

Τα δεδομένα τύπου χαρακτήρα διακρίνονται σε δεδομένα ενός χαρακτήρα **char** και σε δεδομένα πολλών χαρακτήρων **string**, που μπορεί να είναι από **1** μέχρι **255** χαρακτήρες.

Τα λογικά δεδομένα δηλώνονται στην **Pascal** ως **boolean** και έχουν το χαρακτηριστικό ότι μπορούν να πάρουν δύο τιμές, τις **TRUE** (αληθείς) και **FALSE** (ψευδής).

Οι τύποι **integer**, **char** και **boolean** λέγονται και τακτικοί τύποι, γιατί υπάρχει διάταξη μεταξύ των μελών τους (καθένα από τα μέλη έχει προηγούμενο και επόμενο). Με βάση αυτούς τους τύπους, μπορούν να σχηματιστούν και σύνθετοι τύποι, δηλαδή ομάδες δεδομένων, όπως οι **πίνακες (array)** και οι **εγγραφές (record)**.



Επιπλέον, ο προγραμματιστής έχει τη δυνατότητα να ορίσει και δικούς του τύπους δεδομένων, χρησιμοποιώντας στο τμήμα δηλώσεων τη δεσμευμένη λέξη **type**. Για παράδειγμα, με τη δήλωση **type month_number = 1..12;** ορίζεται ένας νέος τύπος δεδομένων με όνομα **month_number**. Οι μεταβλητές του τύπου αυτού θα παίρνουν ακέραιες τιμές από **1** μέχρι **12**. Ο συμβολισμός **..** δηλώνει διάστημα από - έως.

Τελεστές της Pascal

Οι τελεστές της **Pascal** διακρίνονται σε τέσσερις κατηγορίες:

Αριθμητικοί τελεστές:

+ : Πρόσθεση
- : Αφαίρεση
***** : Πολλαπλασιασμός
/ : Διαίρεση
div : Πηλίκo ακέραιας διαίρεσης
mod : Υπόλοιπο ακέραιας διαίρεσης

Τελεστές σύγκρισης:

> : Μεγαλύτερο
>= : Μεγαλύτερο ή ίσο
< : Μικρότερο
<= : Μικρότερο ή ίσο
= : Ίσο
<> : Διάφορο

Λογικοί Τελεστές:

AND : ΚΑΙ
OR : Ή
NOT : ΟΧΙ

Τελεστής ανάθεσης τιμής:

Ο τελεστής αυτός είναι ο **:=** και χρησιμοποιείται για απόδοση τιμής σε μια μεταβλητή. Για παράδειγμα, με την εντολή **x:=5;** δίνεται στη μεταβλητή **x** η τιμή **5**. Ο τελεστής αυτός είναι δυαδικός, δηλαδή έχει δύο τελεστέους, έναν αριστερά του και έναν δεξιά του. Ο αριστερά τελεστέος είναι υποχρεωτικά το όνομα μιας μεταβλητής, ενώ δεξιά τελεστέος μπορεί να είναι το όνομα μιας άλλης μεταβλητής, μια σταθερά ή μια έκφραση. Οι δύο τελεστέοι πρέπει συνήθως να είναι του ίδιου τύπου (π.χ. και οι δύο ακέραιοι - **integer**).

Σημείωση: Στα μαθηματικά δεν ισχύει **x=x+1**, στον προγραμματισμό όμως ισχύει το **x:=x+1**. Αριστερά από το **:=** είναι πάντα το άγνωστο μέρος το οποίο θα πάρει τιμή ενώ δεξιά το γνωστό μέρος. Όταν βλέπουμε μια μεταβλητή να είναι και από τη δεξιά μεριά του **:=** και από την αριστερή αρχικοποιούμε (δίνουμε αρχική τιμή) πιο πριν σε αυτή την μεταβλητή.

Δομή ενός προγράμματος σε Pascal

Ένα πρόγραμμα σε γλώσσα **Pascal** αποτελείται από δύο τμήματα: το **τμήμα δηλώσεων (declaration part)** και το **σώμα του προγράμματος (program body)**.



Το τμήμα δηλώσεων περιλαμβάνει τη δήλωση του ονόματος του προγράμματος (π.χ. **PROGRAM MY_FIRST_PROGRAM;**), τη δήλωση των **εξωτερικών μονάδων** που θα χρησιμοποιηθούν (π.χ. **USES CRT;**), τις δηλώσεις των **τύπων** (π.χ. **TYPE MONTH = 1..12;**), τις δηλώσεις των **σταθερών** (π.χ. **CONST LIMIT = 50;**) και των **μεταβλητών** (π.χ. **VAR X: INTEGER;**), αλλά και τους ορισμούς των **συναρτήσεων (function)** και **διαδικασιών (procedure)**.

Το σώμα του προγράμματος αρχίζει με τη δεσμευμένη λέξη **BEGIN** και τελειώνει με τη δεσμευμένη λέξη **END.**, περικλείοντας ανάμεσά τους τις εκτελέσιμες εντολές του προγράμματος.

Η ολοκληρωμένη δομή ενός προγράμματος **Pascal** είναι:

```
PROGRAM <όνομα προγράμματος>;  
USES <εξωτερικές μονάδες που χρησιμοποιεί το πρόγραμμα>;  
CONST <τμήμα δήλωσης σταθερών του προγράμματος>;  
TYPE <τμήμα δήλωσης τύπων του προγράμματος>;  
VAR <τμήμα δήλωσης μεταβλητών του προγράμματος>;  
    <τμήμα δήλωσης συναρτήσεων και διαδικασιών>;  
BEGIN {έναρξη σώματος κυρίως προγράμματος}  
    <εντολές κυρίως προγράμματος>;  
END.
```

Εντολές της Pascal

Στην **Pascal** υπάρχουν τρεις βασικές προγραμματιστικές δομές:

- [1] Η **δομή της ακολουθίας** των εντολών, στην οποία οι εντολές εκτελούνται με τη σειρά που αναγράφονται και κατεύθυνση από πάνω προς τα κάτω.
- [2] Η **δομή της επιλογής ή απόφασης**, στην οποία ο έλεγχος της ροής του προγράμματος αποφασίζεται από την τιμή μιας λογικής έκφρασης. Αν η τιμή της λογικής έκφρασης είναι αληθής (**TRUE**) εκτελείται το τμήμα A του προγράμματος, αν είναι ψευδής (**FALSE**) εκτελείται το τμήμα B του προγράμματος.
- [3] Η **δομή της επανάληψης**, στην οποία ένα τμήμα του προγράμματος εκτελείται κατ' επανάληψη ανάλογα με την τιμή μιας λογικής έκφρασης.

Στην **Pascal** δεν υπάρχει ο περιορισμός μιας εντολής σε κάθε γραμμή. Μπορούμε να γράψουμε μια εντολή σε περισσότερες γραμμές ή και πολλές εντολές σε μια γραμμή. Για να διαχωρίζονται οι εντολές μεταξύ τους χρησιμοποιείται το σύμβολο του ελληνικού ερωτηματικού (**;**), το οποίο τοποθετούμε στο τέλος κάθε εντολής. Οι δεσμευμένες λέξεις **BEGIN** και **END** χρησιμοποιούνται επίσης ως διαχωριστές εντολών, γι' αυτό δεν είναι απαραίτητο το ερωτηματικό μετά από το **BEGIN** και πριν από το **END**.

Εντολές εισόδου - εξόδου

Για εισαγωγή δεδομένων στο πρόγραμμα από το χρήστη (μέσω του πληκτρολογίου) και εμφάνιση των αποτελεσμάτων στην οθόνη, χρησιμοποιούμε τις διαδικασίες εισόδου και εξόδου (**procedures**) **read** και **write** (ή **readln** και **writeln**, όταν θέλουμε να διαβάσει μαζί με την τιμή που θα δώσουμε και το **enter**, ή να αλλάξει γραμμή μετά την εκτέλεση της **writeln**, αντίστοιχα).

Η **writeln** είναι μια ειδική λέξη που επιτρέπει να εμφανιστούν κάποια δεδομένα στην οθόνη του υπολογιστή μας. Τα δεδομένα μέσα στην παρένθεση είναι τα δεδομένα που θα εκτυπωθούν. Οποιαδήποτε πληροφορία ανάμεσα στις αποστρόφους θα εκτυπωθεί ως απλό κείμενο. Η ειδική λέξη **writeln** δεν είναι δεσμευμένη λέξη αλλά είναι καθορισμένη από το σύστημα να κάνει μια ειδική δουλειά για σας, δηλαδή να εκτυπώνει μια γραμμή από δεδομένα στην οθόνη. Η εντολή **write** χρησιμοποιείται για εκτύπωση στην οθόνη, όπως και με η **writeln**, αλλά ο κέρσορας δεν αλλάζει γραμμή στην έξοδο. Η **writeln** γράφει το προς εκτύπωση κείμενο και αλλάζει γραμμή, η **write** όχι.

Παραδείγματα



read(a); Διαβάζει τη μεταβλητή **a**. Ο χρήστης πρέπει να δώσει από το πληκτρολόγιο του Η/Υ μια τιμή για τη μεταβλητή **a**. Αν η μεταβλητή έχει δηλωθεί **integer** θα πρέπει ο χρήστης να δώσει ακέραια τιμή για το **a**.

read(a,b); Ο χρήστης θα πρέπει να δώσει από το πληκτρολόγιο του Η/Υ δυο τιμές χωρισμένες με ένα κενό.

write('KALHMERA'); Εμφανίζει στην οθόνη τη λέξη **KALHMERA** και ο δρομέας βρίσκεται ακριβώς δίπλα από τη λέξη.

write('KALHMERA '); Εμφανίζει στην οθόνη τη λέξη **KALHMERA** και ο δρομέας βρίσκεται με ένα κενό δίπλα από τη λέξη.

writeln('KALHMERA'); Εμφανίζει στην οθόνη τη λέξη **KALHMERA** και ο δρομέας πηγαίνει στην ακριβώς από κάτω γραμμή από τη λέξη.

write('A = ',a); Εμφανίζει στην οθόνη τη λέξη **A =** και αμέσως μετά την τιμή της μεταβλητής **a**. Το σχόλιο από τη μεταβλητή χωρίζεται με ένα κόμμα.

write(a:5:2); Εμφανίζει στην οθόνη τη τιμή του πραγματικού αριθμού **a** σε πέντε θέσεις, από τις οποίες οι δύο είναι τα δεκαδικά ψηφία και η μία η υποδιαστολή. Αν οι θέσεις που μένουν δεν επαρκούν για το ακέραιο μέρος, θα χρησιμοποιηθούν περισσότερες χωρίς καμία παρέμβαση από τον χρήστη.

writeln(1,3*3,4+2,6-1); Εμφανίζει στην οθόνη τις τιμές **1 9 6** και **5** χωρίς κενά μεταξύ τους και ο δρομέας πηγαίνει στην από κάτω γραμμή.

writeln(1,' ',2,' '3*2); Εμφανίζει στην οθόνη τις τιμές **1 2** και **5** με ένα κενό μεταξύ τους και ο δρομέας πηγαίνει στην από κάτω γραμμή.

writeln(a+b); Εμφανίζει στην οθόνη την τιμή του αθροίσματος **a+b** και ο δρομέας πηγαίνει στην από κάτω γραμμή.

writeln('a+b'); Εμφανίζει στην οθόνη τη λέξη (σχόλιο) **a+b** και ο δρομέας πηγαίνει στην από κάτω γραμμή.

Δομή Ακολουθίας

Στη δομή ακολουθίας οι εντολές εκτελούνται η μία μετά την άλλη με τη σειρά που αναγράφονται και κατεύθυνση από πάνω προς τα κάτω.

```
εντολή 1;  
εντολή 2;  
...  
εντολή n;
```

Παράδειγμα

Να γραφεί πρόγραμμα σε γλώσσα **Pascal** το οποίο υπολογίζει και εμφανίζει το εμβαδόν τραπεζίου, όταν δίνονται οι βάσεις και το ύψος του. Το εμβαδόν τραπεζίου δίνεται αν πολλαπλασιάσουμε το ημιάθροισμα των βάσεων επί το ύψος.

```
PROGRAM PARADEIGMA_1;  
VAR B1,B2,Y:INTEGER;  
E,HA:REAL;  
BEGIN  
    WRITE('DOSE BASH TRAPEZIOY MEGALH ');  
    READLN(B1);  
    WRITE('DOSE BASH TRAPEZIOY MIKRH ');  
    READLN(B2);
```



```
WRITE( 'DOSE YPSOS TRAPEZIOY ' );  
READLN(Y);  
HA := (B1+B2)/2;  
E := HA * Y;  
WRITELN('TO EMBADON TOY TRAPEZIOY EINAI : ',E:5:2);  
END.
```

Η εντολή - IF

ΑΠΛΗ ΕΠΙΛΟΓΗ

Η εντολή χρησιμοποιεί τρεις δεσμευμένες (**IF**, **THEN**, **ELSE**). Μετά το **IF** ακολουθεί η συνθήκη, η οποία τελειώνει πριν από τη δεσμευμένη λέξη **THEN**. Μετά το **THEN** και πριν από τη λέξη **ELSE**, γράφεται η εντολή1 ή οι εντολές της ομάδας 1. Η εντολή2 ή οι εντολές της ομάδας 2 γράφονται αμέσως μετά τη βοηθητική λέξη **ELSE**.

```
IF <συνθήκη> THEN  
    εντολή1;  
ELSE  
    εντολή2;
```

Σημείωση: Αν υπάρχουν περισσότερες από μια εντολές πρέπει να τις βάλουμε ανάμεσα σε **BEGIN ... END**

```
IF <συνθήκη> THEN  
    BEGIN  
        ομάδα εντολών 1;  
    END;  
ELSE  
    BEGIN  
        ομάδα εντολών 2;  
    END;
```

Η δομή επιλογής εκτελείται ως εξής: Αρχικά ελέγχεται η συνθήκη. Στη συνέχεια υπάρχουν δύο δυνατές περιπτώσεις:

- δ Αν η συνθήκη ικανοποιείται (η απάντηση είναι ναι), εκτελείται μόνο η εντολή1 ή η ομάδα εντολών 1. Η εντολή2 ή ομάδα εντολών 2 αγνοείται και ο έλεγχος του αλγόριθμου βγαίνει από τη δομή επιλογής.
- δ Σε αντίθετη περίπτωση, αν δηλαδή η συνθήκη δεν ικανοποιείται (η απάντηση είναι όχι), δε λαμβάνεται υπόψη η εντολή1 ή η ομάδα εντολών 1. Εκτελείται μόνο η εντολή2 ή η ομάδα εντολών 2 και η εκτέλεση της δομής επιλογής τελειώνει.

Παράδειγμα

Να γραφεί πρόγραμμα σε γλώσσα **Pascal** το οποίο να διαβάσει δύο αριθμούς **A** και **B** να τους συγκρίνει και να εμφανίζει ποιος είναι μεγαλύτερος.

```
{ΠΡΩΤΟΣ ΤΡΟΠΟΣ ΛΥΣΗΣ}  
  
PROGRAM PARADEIGMA_2A;  
VAR A,B:INTEGER; C:STRING;  
BEGIN
```



```
WRITE( 'DOSE ARITHMO A ');
READLN(A);
WRITE( 'DOSE ARITHMO B ');
READLN(B);
IF A > B THEN
    BEGIN
        C := 'O ARITHMOS A EINAI MEGALYTEROS TOY B ';
        WRITELN(C);
    END;
ELSE
    BEGIN
        C := 'O ARITHMOS B EINAI MEGALYTEROS TOY A ';
        WRITELN(C);
    END;
END.
```

{ΜΠΟΡΟΥΜΕ ΝΑ ΒΑΛΟΥΜΕ ΤΗΝ ΕΝΤΟΛΗ WRITELN() ΜΟΝΟ ΜΙΑ ΦΟΡΑ ΜΕ ΤΟ ΤΕΛΟΣ ΤΗΣ ΕΝΤΟΛΗΣ ΑΠΟΦΑΣΗΣ IF}
{ΔΕΥΤΕΡΟΣ ΤΡΟΠΟΣ ΛΥΣΗΣ}

```
PROGRAM PARADEIGMA_2B;
VAR A,B:INTEGER;
C:STRING;
BEGIN
    WRITE( 'DOSE ARITHMO A ');
    READLN(A);
    WRITE( 'DOSE ARITHMO B ');
    READLN(B);
    IF A > B THEN
        C := 'O ARITHMOS A EINAI MEGALYTEROS TOY B '
    ELSE
        C := 'O ARITHMOS B EINAI MEGALYTEROS TOY A ';
    WRITELN(C);
END.
```

ΠΕΡΙΟΡΙΣΜΕΝΗ ΕΠΙΛΟΓΗ

Υπάρχει περίπτωση σε μια εντολή επιλογής, στη περίπτωση που ισχύει η συνθήκη, να εκτελείται μια εντολή ή μια σειρά εντολών, ενώ στην περίπτωση που δεν ισχύει η συνθήκη να μη θέλουμε να εκτελεστεί κάποια εντολή ή εντολές. Στη περίπτωση αυτή χρησιμοποιούμε τη δομή περιορισμένης επιλογής.

```
IF <συνθήκη> THEN
    εντολή 1;
IF <συνθήκη> THEN
    BEGIN
        ομάδα εντολών 1;
    END;
```

Στη δομή περιορισμένης επιλογής, η δεσμευμένη λέξη **ELSE**, καθώς και η εντολή**2** ή η ομάδα εντολών **2** παραλείπονται.

Παράδειγμα



Να γραφεί πρόγραμμα σε γλώσσα **Pascal** το οποίο να διαβάζει το βαθμό μαθήματος ενός σπουδαστή. Αν ο βαθμός είναι μεγαλύτερος ή ίσος του **5** να εμφανίζει το μήνυμα ότι πέρασε το μάθημα.

```
PROGRAM PARADEIGMA_3;  
VAR N:INTEGER;  
L:STRING;  
BEGIN  
    WRITE('DOSE BATHMO SPOYDASTH ');  
    READLN(N);  
    IF N >= 5 THEN  
        BEGIN  
            L := 'O SPOYDASTHS PERASE TO MATHIMA';  
            WRITELN(L);  
        END;  
END.
```

Η εντολή - CASE

Σε πολλές περιπτώσεις, υπάρχουν περισσότερες εναλλακτικές επιλογές, που εξαρτώνται από την τιμή μιας μεταβλητής. Στις περιπτώσεις αυτές, χρησιμοποιείται η εντολή **case**, που έχει την παρακάτω μορφή:

```
CASE <έκφραση> OF  
    <τιμή 1>: <εντολή 1>;  
    <τιμή 2>: <εντολή 2>;  
    ...  
    <τιμή n>: <εντολή n>;  
END;
```

Υπολογίζεται η έκφραση και συγκρίνεται με κάθε τιμή της λίστας των πιθανών τιμών της έκφρασης. Εκτελείται εκείνη η εντολή, στην οποία η τιμή συμπίπτει με την τιμή της έκφρασης, και ο έλεγχος του προγράμματος μεταφέρεται στην εντολή που ακολουθεί το **end**. Η έκφραση και οι πιθανές τιμές της πρέπει απαραίτητα να είναι τακτικού τύπου.

Στην εντολή **case** μπορεί να υπάρχει πριν το **end** μια εναλλακτική εντολή που θα εκτελεστεί όταν καμία τιμή δεν συμπίπτει με την τιμή της έκφρασης. Η εντολή αυτή συντάσσεται με **else**:

```
CASE <έκφραση> OF  
    <τιμή 1>: <εντολή 1>;  
    <τιμή 2>: <εντολή 2>;  
    ...  
    <τιμή k>: <εντολή k>  
ELSE <εντολή n>;  
END;
```

Σημείωση: Αν υπάρχουν περισσότερες από μια εντολές πρέπει να τις βάλουμε ανάμεσα σε **BEGIN ... END**.

Παράδειγμα

Να γραφεί πρόγραμμα σε γλώσσα **Pascal** το οποίο να διαβάζει ένα ακέραιο αριθμό N από το **1** μέχρι το **12** και να εμφανίζει τον αντίστοιχο μήνα του έτους (**1** για Ιανουάριο, **2** για Φεβρουάριο, ..., **12** για Δεκέμβριο).

```
PROGRAM PARADEIGMA_4;  
USES CRT;
```



```
VAR I:INTEGER;
BEGIN
    CLRSCR;
    WRITE('DOSE ARITHMO 1- 12 GIA NA EMFANISTEI O KATALLHLOS MHNAS ');
    READLN(I);
    CASE I OF
        1: WRITELN('IANΟΥΑΡΙΟΣ ');
        2: WRITELN('FEBΡΟΥΑΡΙΟΣ ');
        3: WRITELN('ΜΑΡΤΙΟΣ ');
        4: WRITELN('ΑΠΡΙΛΙΟΣ ');
        5: WRITELN('ΜΑΙΟΣ ');
        6: WRITELN('ΙΟΥΝΙΟΣ ');
        7: WRITELN('ΙΟΥΛΙΟΣ ');
        8: WRITELN('ΑΥΓΟΥΣΤΟΣ ');
        9: WRITELN('ΣΕΠΤΕΜΒΡΙΟΣ ');
        10: WRITELN('ΟΚΤΟΒΡΙΟΣ ');
        11: WRITELN('ΝΟΕΜΒΡΙΟΣ ');
        12: WRITELN('ΔΕΚΕΜΒΡΙΟΣ ');
    ELSE
        WRITELN('ΜΗ ΑΠΟΔΕΚΤΟΣ ΑΡΙΘΜΟΣ ΓΙΑ ΜΗΝΑ ');
    END;
END.
```

Εντολές Επανάληψης (Η εντολή FOR)

Όταν είναι γνωστός εκ των προτέρων ο αριθμός των επαναλήψεων, μπορεί να χρησιμοποιηθεί η εντολή επανάληψης **FOR**, η οποία μπορεί να έχει μια από τις ακόλουθες μορφές:

```
FOR <mm>:=<aa> TO <bb> DO <εντολή1>;
FOR <mm>:=<aa> DOWNTO <bb> TO <εντολή1>;
FOR <mm>:=<aa> TO <bb> DO
    Begin
        ομάδα εντολών;
    End;
```

Εδώ χρησιμοποιούνται οι δεσμευμένες λέξεις **FOR**, **TO (DOWNTO)**, **DO** και οι μεταβλητές-σταθερές **mm (μετρητής)**, **aa (αρχική τιμή)**, **bb (τελική τιμή)**. Οποιαδήποτε απλή μεταβλητή με τύπο **integer**, **byte** ή **char** μπορεί να χρησιμοποιηθεί ως δείκτης επανάληψης με την προϋπόθεση ότι πρέπει να υπάρχει μεταβλητή που να είναι δηλωμένη ως **var**.

Αυτή η μορφή της επαναληπτικής δομής έχει το χαρακτηριστικό ότι δεν απαιτείται στην ομάδα εντολών να υπάρχουν εντολές μεταβολής της τιμής του μετρητή, η οποία μεταβάλλεται από την ίδια την εντολή σε κάθε επανάληψη κατά **1** στην περίπτωση του **TO** ή κατά **-1** όταν υπάρχει το **DOWNTO**.

Παράδειγμα

Να γραφεί πρόγραμμα σε γλώσσα **Pascal** το οποίο να υπολογίζει και να τυπώνει το μέσο όρο, το άθροισμα και το γινόμενο των αριθμών από το **1** έως το **N**.

```
PROGRAM PARADEIGMA_5;
VAR I,N,SUM:INTEGER; GIN:LONGINT; MO:REAL;
BEGIN
    WRITE('DOSE ARITHMO N EOS TO 10 : ');
    READLN(N);
```



```
SUM := 0;  
GIN := 1;  
FOR I:=1 TO N DO  
    BEGIN  
        SUM := SUM + I;  
        GIN := GIN * I;  
    END;  
MO := SUM / N;  
WRITELN('ΤΟ ΑΘΡΟΙΣΜΑ ΕΙΝΑΙ : ',SUM);  
WRITELN('ΤΟ ΓΙΝΟΜΕΝΟ ΕΙΝΑΙ : ',GIN);  
WRITELN('Ο ΜΕΣΟΣ ΟΡΟΣ ΕΙΝΑΙ : ',MO:5:3);  
END.
```

Περιπτώσεις στις οποίες δεν μπορεί να χρησιμοποιηθεί η εντολή επανάληψης for

Παράδειγμα

Να γραφεί πρόγραμμα σε γλώσσα **Pascal** το οποίο να διαβάζει έως **10** ακέραιους αριθμούς, να υπολογίζει και να τυπώνει (α) το μέσο όρο τους και (β) πόσοι από αυτούς είναι θετικοί, πόσοι αρνητικοί και πόσοι μηδέν. Αν διαβάσει τον αριθμό **100** να σταματάει η εκτέλεσή του.

```
PROGRAM PARADEIGMA_6;  
VAR N,AR,POS,NEG,ZER,I,SUM:INTEGER;  
MO:REAL;  
BEGIN  
    SUM := 0;  
    POS := 0;  
    NEG := 0;  
    ZER := 0;  
    I := 0;  
    WRITE('ΔΟΣΕ ΑΡΙΘΜΟ ');  
    READLN(AR);  
    WHILE (I < 10) AND (AR <> 100) DO  
        BEGIN  
            SUM := SUM + AR;  
            IF AR > 0 THEN  
                POS := POS + 1  
            ELSE IF AR < 0 THEN  
                NEG := NEG + 1  
            ELSE  
                ZER := ZER + 1;  
                I := I + 1;  
            WRITE('ΔΟΣΕ ΑΡΙΘΜΟ ');  
            READLN(AR);  
        END;  
    IF I > 0 THEN  
        BEGIN  
            MO := SUM / I;  
            WRITELN('Ο ΜΕΣΟΣ ΟΡΟΣ ΕΙΝΑΙ ',MO:5:2);  
        END;  
    WRITELN('ΟΙ ΘΕΤΙΚΟΙ ΕΙΝΑΙ ',POS);  
    WRITELN('ΟΙ ΑΡΝΗΤΙΚΟΙ ΕΙΝΑΙ ',NEG);  
    WRITELN('ΜΗΔΕΝ ΕΙΝΑΙ ',ZER);  
END.
```



Εντολές Επανάληψης (Η εντολή WHILE - DO)

Η σύνταξη της εντολής μπορεί να έχει μια από τις ακόλουθες μορφές:

```
WHILE <συνθήκη> DO  
    <εντολή1>;  
WHILE <συνθήκη> DO  
    Begin  
        ομάδα εντολών;  
    End;
```

Εδώ χρησιμοποιούνται οι δεσμευμένες λέξεις **WHILE**, **DO**. Μετά τη λέξη **WHILE** ακολουθεί η συνθήκη συνέχειας. Αμέσως μετά την λέξη **DO** γράφεται η εντολή ή η ομάδα εντολών. Οι συνθήκες που χρησιμοποιούνται εδώ είναι ίδιες με τις συνθήκες που χρησιμοποιούνται στην εντολή επιλογής.

Στην αρχή της εκτέλεσης ελέγχεται η συνθήκη. Αν αυτή ικανοποιείται, τότε εκτελείται η εντολή ή η ομάδα εντολών. Στη συνέχεια, ελέγχεται ξανά η συνθήκη. Δεν είναι σίγουρο ότι αυτή εξακολουθεί να είναι αληθής διότι το αποτέλεσμα της μπορεί να έχει επηρεαστεί από την εκτέλεση της εντολής ή της ομάδας εντολών. Αν η συνθήκη ικανοποιείται ξανά, εκτελείται πάλι η εντολή ή η ομάδα εντολών. Η διαδικασία συνεχίζεται, όσο η συνθήκη παραμένει αληθής. Όταν η συνθήκη συνέχειας γίνει ψευδής, η εντολή ή η ομάδα εντολών δεν εκτελείται και τότε λήγει η εκτέλεση της δομής.

Παράδειγμα

Να γραφεί πρόγραμμα σε γλώσσα **Pascal** το οποίο να υπολογίζει και να τυπώνει το μέσο όρο, το άθροισμα και το γινόμενο των αριθμών από το 1 έως το N.

```
PROGRAM PARADEIGMA_7;  
VAR I,N,SUM:INTEGER; GIN:LONGINT; MO:REAL;  
BEGIN  
    WRITE('DOSE ARITHMO N EOS TO 10 : ');  
    READLN(N);  
    SUM := 0; GIN := 1; I := 1;  
    WHILE I <= N DO  
        BEGIN  
            SUM := SUM + I;  
            GIN := GIN * I;  
            I := I + 1;  
        END;  
    MO := SUM / N;  
    WRITELN('TO ATHROISMA EINAI : ',SUM);  
    WRITELN('TO GINOMENO EINAI : ',GIN);  
    WRITELN('O MESOS OROS EINAI : ',MO:5:3);  
END.
```

Εντολές Επανάληψης (Η εντολή REPEAT - UNTIL)

Η σύνταξη της εντολής μπορεί να έχει μια από τις ακόλουθες μορφές:

```
REPEAT  
    <εντολή1>;  
UNTIL <συνθήκη>;  
REPEAT  
    ομάδα εντολών;
```



UNTIL <συνθήκη>;

Εδώ χρησιμοποιούνται οι δεσμευμένες λέξεις **REPEAT**, **UNTIL**. Μετά τη λέξη **REPEAT** ακολουθούν οι εντολές οι οποίες τελειώνουν πριν από τη λέξη **UNTIL**. Και στη περίπτωση αυτή, οι συνθήκες που χρησιμοποιούνται είναι ίδιες με τις συνθήκες που χρησιμοποιούνται στην εντολή επιλογής. Η εντολή τελειώνει με το **UNTIL <συνθήκη>**.

Σε αυτή τη μορφή της εντολής επανάληψης ο έλεγχος γίνεται μετά την εκτέλεση της ομάδας εντολών. **Αυτό σημαίνει ουσιαστικά ότι η ομάδα εντολών εκτελείται τουλάχιστον μία φορά.** Μετά την εκτέλεση της ομάδας εντολών ελέγχεται η συνθήκη. Αν αυτή δεν ικανοποιείται, τότε εκτελείται ξανά η ομάδα εντολών, όσο η συνθήκη παραμένει ψευδής. Όταν η συνθήκη γίνει αληθής, λήγει η εκτέλεση της επαναληπτικής δομής.

Παράδειγμα

Να γραφεί πρόγραμμα σε γλώσσα **Pascal** το οποίο να υπολογίζει και να τυπώνει το μέσο όρο, το άθροισμα και το γινόμενο των αριθμών από το **1** έως το **N**.

```
PROGRAM PARADEIGMA_8;  
VAR I,N,SUM:INTEGER; GIN:LONGINT; MO:REAL;  
BEGIN  
    WRITE('DOSE ARITHMO N EOS TO 10 : ');  
    READLN(N);  
    SUM := 0; GIN := 1; I := 1;  
    REPEAT  
        SUM := SUM + I;  
        GIN := GIN * I;  
        I := I + 1;  
    UNTIL I > N;  
    MO := SUM / N;  
    WRITELN('TO ATHROISMA EINAI : ',SUM);  
    WRITELN('TO GINOMENO EINAI : ',GIN);  
    WRITELN('O MESOS OROS EINAI : ',MO:5:3);  
END.
```

Διαδικασίες Και Συναρτήσεις

Τις περισσότερες φορές ένα μεγάλο πρόγραμμα μπορεί να γίνει με τη δημιουργία πολλών μικρότερων υποπρογραμμάτων. Αυτό μας προσφέρει αρκετά πλεονεκτήματα όπως:

- Η συντήρηση των προγραμμάτων γίνεται ευκολότερη.
- Η δυνατότητα χρήσης των υποπρογραμμάτων που εκτελούν συνηθισμένες εργασίες μπορούν να μεταφερθούν και σε άλλα προγράμματα - εφαρμογές.
- Η δομή του προγράμματος γίνεται εύκολα κατανοητή.
- Η δυνατότητα ανάπτυξης μικρότερων προγραμμάτων από ανεξάρτητους προγραμματιστές, με σκοπό τη βελτίωση του χρόνου ανάπτυξης του λογισμικού.

Ένα **υποπρόγραμμα** είναι ένα σύνολο εντολών της **Pascal** που εκτελεί συγκεκριμένο έργο και γράφεται με τους ίδιους κανόνες που γράφεται ένα πρόγραμμα. Η **Pascal** μας δίνει τη δυνατότητα να έχουμε δύο είδη υποπρογραμμάτων: τις **διαδικασίες (procedures)** και τις **συναρτήσεις (functions)**. Η δομή τους ελάχιστα διαφέρει, αλλά χρησιμοποιούνται σε διαφορετικές περιπτώσεις. Τα υποπρογράμματα δεν είναι αυτοτελή προγράμματα γι' αυτό και δεν μπορούν να εκτελεστούν από μόνα τους.

Δηλώνονται μετά το τμήμα δηλώσεων του προγράμματος μετά το **VAR** και πριν από το **BEGIN** του κυρίως προγράμματος. (Έχουμε τη δυνατότητα να γράψουμε το υποπρόγραμμα σε



ξεχωριστό αρχείο και να το ενσωματώσουμε στην ίδια περιοχή χρησιμοποιώντας την διαδικασία **\$I (Include)**).

Τα υποπρογράμματα επικοινωνούν με το κυρίως πρόγραμμα μέσω παραμέτρων και περιλαμβάνουν ένα σύνολο εντολών οι οποίες εκτελούν μια συγκεκριμένη λειτουργία. Η διαφορά μεταξύ τους είναι ότι οι συναρτήσεις υπολογίζουν και επιστρέφουν στο σημείο το οποίο καλούνται μια τιμή, ενώ οι διαδικασίες εκτελούν κάποιες λειτουργίες μεταβάλλοντας κάποιες μεταβλητές χωρίς να επιστρέφουν τιμή στο σημείο στο οποίο καλούνται.

Διαδικασίες - Procedures

Η δήλωση μιας διαδικασίας έχει την πιο κάτω μορφή:

```
PROCEDURE <όνομα_διαδικασίας> (<λίστα παραμέτρων>:<τύπος>);  
<Τμήμα δηλώσεων τοπικών μεταβλητών>  
BEGIN  
    <Εντολές διαδικασίας>  
END;
```

Η λέξη **PROCEDURE** είναι μια δεσμευμένη λέξη η οποία ακολουθείται από το όνομα της διαδικασίας και μετά εμφανίζονται όλες οι παράμετροι και ο τύπος τους. Οι παράμετροι ονομάζονται **τυπικές (formal) παράμετροι**. Οι τυπικές παράμετροι δίνουν την δυνατότητα του ορισμού των υπολογισμών μιας διαδικασίας με ένα γενικό τρόπο. Υπάρχει περίπτωση να μην υπάρχουν τυπικές παράμετροι, γι αυτό η παρένθεση με τη λίστα των τυπικών παραμέτρων είναι προαιρετική. Οι μεταβλητές που εμφανίζονται στο τμήμα δηλώσεων της διαδικασίας είναι τοπικές δηλαδή ορίζονται μόνο κατά τη διάρκεια εκτέλεσής της.

Όταν μια διαδικασία είναι ορισμένη στο τμήμα δηλώσεων του κυρίως προγράμματος τότε μπορεί να χρησιμοποιηθεί οπουδήποτε εντός του κυρίως προγράμματος με μια απλή κλήση της που έχει τη γενική μορφή:

όνομα_διαδικασίας (λίστα πραγματικών παραμέτρων); όπου σαν όνομα της διαδικασίας αναφέρουμε το όνομα της διαδικασίας που επιθυμούμε να εκτελεστεί και αντικαθιστώντας τη λίστα των **τυπικών (formal) παραμέτρων** με εκείνη των **πραγματικών (actual)**. Οι πραγματικές παράμετροι είναι τιμές ή μεταβλητές που αντιπροσωπεύουν τιμές. Η πρώτη πραγματική παράμετρος αντιστοιχεί με την πρώτη τυπική παράμετρο, η δεύτερη πραγματική με τη δεύτερη τυπική κ.ο.κ. Ο αριθμός των πραγματικών και τυπικών παραμέτρων θα πρέπει να είναι ακριβώς ο ίδιος και ο τύπος των τιμών κάθε πραγματικής παραμέτρου θα πρέπει να είναι συμβατός με την αντίστοιχη τυπική παράμετρο.

Οι σταθερές, οι τύποι, οι μεταβλητές και οι διαδικασίες που δηλώνονται στο τμήμα δηλώσεων ονομάζονται **γενικές (global)** γιατί μπορούν να χρησιμοποιηθούν οπουδήποτε μέσα στο κυρίως πρόγραμμα, σε μια διαδικασία ή και σε μια διαδικασία που έχει κληθεί μέσα από άλλη διαδικασία. Οι σταθερές, μεταβλητές, κλπ. που βρίσκονται στο τμήμα δηλώσεων μιας διαδικασίας έχουν ισχύ μόνο για τη συγκεκριμένη διαδικασία (υποπρόγραμμα) γι αυτό ονομάζονται τοπικές μεταβλητές.

Θα πρέπει να γνωρίζουμε ότι μια διαδικασία μπορεί να καλέσει μια άλλη διαδικασία εφόσον έχει δηλωθεί πριν από αυτή στο ίδιο υποπρόγραμμα. Όταν όμως πρέπει να κληθεί μια διαδικασία πριν ορισθεί, θα πρέπει να χρησιμοποιήσουμε τη δήλωση **FORWARD**. Η δήλωση αυτή χρησιμοποιείται μόνο για την αλλαγή της τάξης των δηλώσεων και τοποθετείται πριν από τη διαδικασία στο οποίο καλείται η διαδικασία που δηλώνεται με την **FORWARD**.

Παραδείγματα

[1] Να γραφεί ένα πρόγραμμα το οποίο διαβάσει την αξία και το πλήθος των εμπορευμάτων, μας ρωτάει αν η πώληση είναι χονδρική και μας δίνει την συνολική αξία των εμπορευμάτων.



(Εδώ χρησιμοποιούμε δύο διαδικασίες στις οποίες η δεύτερη καλεί την πρώτη όταν η πώληση δεν είναι χονδρική.)

```
PROGRAM PARADEIGMA_9;
CONST KEROOS = 0.15;
VAR AXIA, SAXIA :REAL;
NUMBER:INTEGER;
XONDRIKH:CHAR;

PROCEDURE TOTAL (VAR P1:INTEGER; VAR P2, P3 :REAL);
BEGIN {total}
    P3:= P1 * (P2 + KEROOS * P2)
END; {total}

PROCEDURE AMOUNTDUE(VAR N1:INTEGER;
VAR N2, N3 :REAL; VAR C: CHAR);
BEGIN {amount due}
    IF C= 'Y' THEN
        N3 := N1 * N2
    ELSE
        TOTAL(N1,N2,N3)
END; {amount due}

BEGIN {main program}
    WRITE('DOSE AXIA EMPOREYMATOS: ');
    READLN(AXIA);
    WRITE('DOSE PLITHOS EMPOREYMATON: ');
    READLN(NUMBER);
    WRITE('EINAI XONDRIKH POLHSH; (Y/N) ');
    READLN(XONDRIKH);
    AMOUNTDUE(NUMBER, AXIA, SAXIA, XONDRIKH);
    WRITELN(NUMBER, ' EIDH ME EURO ',AXIA:0:2);
    WRITELN('SYNOIHKOS LOGARIASMOS EURO ',SAXIA:0:2);
    IF XONDRIKH = 'Y' THEN
        WRITELN('H POLHSH EINAI XONDRIKH')
    ELSE
        WRITELN('H POLHSH EINAI LIANIKH ')
END. {main program}
```

[2] Να γραφεί πρόγραμμα σε γλώσσα **Pascal** το οποίο να διαβάζει το μήκος και το πλάτος ενός ορθογώνιου παραλληλόγραμμου και να υπολογίζει και να εμφανίζει το εμβαδόν του.

```
PROGRAM PARADEIGMA_10;
VAR M,P:INTEGER;

PROCEDURE COUNT(VAR LENGTH, WIDTH,AP:INTEGER);
BEGIN
    WRITELN('TO ORTHOGONIO PARALLHLOGRAMMO EXEI DIASTASEIS: ');
    WRITELN(LENGTH,' EPI ', WIDTH, ' cm ');
    WRITELN('EXEI EMBADON ',LENGTH * WIDTH,' TETRAGONIKA cm ');
END;

BEGIN
    WRITE('DOSE MHKOS PLEVRAS ');
    READLN(M);
```



```
WRITE('DOSE PLATOS PLEVRAS ');  
READLN(P);  
COUNT(M, P);  
END.
```

Συναρτήσεις - Functions

Η δήλωση μιας συνάρτησης έχει την πιο κάτω μορφή:

```
FUNCTION <όνομα_συνάρτησης> (<λίστα_παραμέτρων>:<τύπος>):<τύπος_αποτελέσματος>;  
<Τμήμα δηλώσεων τοπικών μεταβλητών>  
BEGIN  
    Ομάδα εντολών;  
    όνομα_συνάρτησης := παράσταση  
END;
```

Η λέξη **FUNCTION** είναι μια δεσμευμένη λέξη η οποία ακολουθείται από το όνομα της συνάρτησης. Ακολουθούν οι τυπικές (**formal**) παράμετροι και ο τύπος τους καθορίζεται στη λίστα των τυπικών παραμέτρων. Όλες οι παράμετροι που εμφανίζονται μέσα στις παρενθέσεις ονομάζονται τυπικές παράμετροι π.χ. στη συνάρτηση **ABS(X)**, η παράμετρος X είναι μια τυπική παράμετρος. Οι τυπικές παράμετροι δίνουν την δυνατότητα του ορισμού των υπολογισμών μιας συνάρτησης με ένα γενικό τρόπο. Επειδή υπάρχει περίπτωση να μην υπάρχουν τυπικές παράμετροι, γι αυτό η παρένθεση με τη λίστα των τυπικών παραμέτρων είναι προαιρετική. Μετά τη δεξιά παρένθεση της λίστας των τυπικών παραμέτρων ακολουθεί ο τύπος του αποτελέσματος που λαμβάνει η συνάρτηση αφού προηγηθεί το σύμβολο :

Οι εντολές που υπάρχουν σε μια συνάρτηση μπορεί να είναι είτε μια μόνο εντολή είτε περισσότερες εντολές. Θα πρέπει όμως να υπάρχει μια εντολή της μορφής *όνομα_συνάρτησης := παράσταση*, γιατί η κλήση του υποπρογράμματος συνάρτησης αντιπροσωπεύει μια τιμή, η οποία δίνεται στο όνομά της με την παραπάνω εντολή.

Παραδείγματα

[1] Να γραφεί ένα πρόγραμμα που να χρησιμοποιεί μια συνάρτηση η οποία να δίνει το τετράγωνο ενός αριθμού.

```
PROGRAM PARADEIGMA_11;  
VAR N:INTEGER;  
  
FUNCTION SQUARE(X:INTEGER):INTEGER;  
    BEGIN{square}  
        SQUARE:=X*X;  
    END;{square}  
  
BEGIN{main program}  
    N:=SQUARE(4);  
    WRITELN('4 SQUARED IS ', N);  
    WRITELN('5 SQUARED IS ', SQUARE(5));  
END.{main program}
```

Το αποτέλεσμα που θα πάρουμε αν τρέξουμε το πρόγραμμά μας είναι το πιο κάτω:

4 SQUARED IS 16



5 SQUARED IS 25

[2] Να γραφεί πρόγραμμα σε γλώσσα **Pascal** το οποίο να διαβάζει δυο ακεραίους αριθμούς και να υπολογίζει και να τυπώνει το Μέγιστο Κοινό Διαιρέτη και το Ελάχιστο Κοινό Πολλαπλάσιο.

```
PROGRAM PARADEIGMA_12;  
VAR AR1, AR2: INTEGER;  
  
FUNCTION MKD(X, Y: INTEGER): INTEGER;  
  BEGIN{MKD}  
    WHILE X<>Y DO  
      BEGIN  
        IF X>Y THEN  
          X:=X-Y  
        ELSE  
          Y:=Y-X;  
        END;  
      MKD:=X;  
    END; {MKD}  
  
FUNCTION EKP(X, Y: INTEGER): INTEGER;  
  BEGIN{EKP}  
    EKP:=(X*Y) DIV MKD(X, Y);  
  END; {EKP}  
  
BEGIN{MAIN PROGRAM}  
  WRITE('DOSE TO PROTO AKERAIIO ARITHMO ');  
  READLN(AR1);  
  WRITE('DOSE TO DEYTERO AKERAIIO ARITHMO ');  
  READLN(AR2);  
  WRITELN('O MEGISTOS KOINOS DIAIRETHS EINAI ', MKD(AR1, AR2));  
  WRITELN('TO ELAXISTO KOINO POLLAPLASIO EINAI ', EKP(AR1, AR2));  
END. {MAIN PROGRAM}
```

Μαθηματικές συναρτήσεις βιβλιοθήκης της PASCAL

Ο προγραμματιστής έχει στη διάθεσή του κάποιες μαθηματικές συναρτήσεις βιβλιοθήκης, που μπορεί να χρησιμοποιήσει για την ανάπτυξη των προγραμμάτων του στη γλώσσα **Pascal**. Οι βασικές μαθηματικές συναρτήσεις που μπορούμε να χρησιμοποιήσουμε στην **Pascal** είναι οι ακόλουθες:

exp(x) : εκθετικό του **x** (**ex**) **ln(x)** : νεπέρειος λογάριθμος του **x** (**lnx**)
sqr(x) : τετράγωνο του **x** (**x2**) **sin(x)** : ημίτονο του **x** (**ημx**)
sqrt(x) : τετραγωνική ρίζα του **x** () **cos(x)** : συνημίτονο του **x** (**συνx**)
abs(x) : απόλυτη τιμή του **x** (**|x|**) **arctan(x)** : τόξο εφαπτομένης **x** (**τοξεφx**)
pi : Ο αριθμός **π** (**3.14**). Η συνάρτηση αυτή δεν έχει παραμέτρους.

To x είναι η τυπική παράμετρος των συναρτήσεων. Όταν καλούμε τη συνάρτηση μπορούμε να έχουμε στη θέση του **x** οποιαδήποτε μεταβλητή ή έκφραση, αρκεί να μην παραβιάζονται βασικοί κανόνες (για παράδειγμα, οι **sqrt** και **ln** δεν δέχονται αρνητικές παραμέτρους).

```
PROGRAM PARADEIGMA_13;
```



```
CONST
    SENTINEL =0.0;
VAR
    X:REAL;
BEGIN
    WRITELN('AFTER EACH LINE ENTER A REAL NUMBER OR 0.0 TO STOP');
    WRITELN;
    WRITELN('X', 'ABS(X)' :10, 'SQR(X)' :10, 'SQRT(ABS(X))' :15);
    READLN (X);
    WHILE (X <> SENTINEL) DO
        BEGIN
            WRITELN (ABS(X) :10:2, SQR(X) :10:2, SQRT(ABS(X)) :10:2);
            READLN(X);
        END
    END.
END.
```

Ομοιότητες - Διαφορές Διαδικασιών και Συναρτήσεων.

- [1] Και οι δύο είναι υποπρογράμματα, αυτόνομες μονάδες προγράμματος που κάνουν μια συγκεκριμένη εργασία.
- [2] Δηλώνονται μετά την περιοχή του **VAR** και πριν από το **BEGIN** του κυρίως προγράμματος.
- [3] Η δήλωση της διαδικασίας ξεκινάει με τη λέξη **PROCEDURE** ενώ της συνάρτησης με τη λέξη **FUNCTION**.
- [4] Η διαδικασία όπως και η συνάρτηση έχει ένα όνομα.
- [5] Η συνάρτηση έχει στόχο να υπολογίσει μια τιμή, ενώ ο στόχος της διαδικασίας είναι γενικότερος.
- [6] Στη δήλωση της συνάρτησης περιλαμβάνεται και ο τύπος της τιμής που υπολογίζει, ενώ στη διαδικασία κάτι τέτοιο δεν έχει νόημα.
- [7] Η διαδικασία καλείται με το να γράψουμε το όνομά της σαν εντολή μέσα στο πρόγραμμα ενώ η αναφορά στη συνάρτηση γίνεται μέσα σε μια παράσταση.
- [8] Κατά την κλήση μιας διαδικασίας της δίνουμε πληροφορίες με παραμέτρους τιμής, παραμέτρους μεταβλητές και καθολικές μεταβλητές. Παίρνουμε πληροφορίες από τη διαδικασία με παραμέτρους μεταβλητές και καθολικές μεταβλητές. Ενώ από τις συναρτήσεις παίρνουμε πληροφορία από την ίδια την αναφορά στη συνάρτηση.



Προχωρημένα θέματα στην PASCAL

Μέρος Β΄

Αρχεία



Αρχεία

Το αρχείο είναι ένας σύνθετος τύπος δεδομένων με δόμηση. Σχηματίζεται από μια οργανωμένη συλλογή από συνιστώσες κάποιου άλλου τύπου. Όπως στα σύνολα και στις εγγραφές έτσι και στο αρχείο δίνεται ένα όνομα, η προσπέλαση όμως σε κάθε συνιστώσα γίνεται ξεχωριστά. Σε τύπους χωρίς δόμηση η προσπέλαση σε μία συνιστώσα γίνεται με το όνομά της ανεξάρτητα από τη θέση της, ενώ σε τύπους με δόμηση η προσπέλαση γίνεται σύμφωνα με τη θέση της στη συλλογή και όχι με το όνομά της. Επιτρεπτές πράξεις είναι η εκχώρηση τιμών, η αποθήκευση και η ανάκτηση μεμονωμένων συνιστωσών.

Αρχεία Κειμένου

Τα **αρχεία κειμένου (text files)** αποτελούνται από χαρακτήρες και χωρίζονται σε γραμμές οι οποίες τελειώνουν με το χαρακτήρα μεταφοράς στην επόμενη γραμμή (**CR/LF**). Οι χαρακτήρες των γραμμών μπορεί να σχηματίζουν λέξεις ή προτάσεις, να είναι ειδικοί χαρακτήρες, εκτυπώσιμοι ή όχι, ή αριθμητικοί χαρακτήρες που αντιστοιχούν στα ψηφία αριθμητικών δεδομένων. Για να χρησιμοποιήσουμε ένα αρχείο κειμένου, πρέπει όπως κάνουμε και για κάθε άλλη μεταβλητή, να του δώσουμε ένα όνομα και να δηλώσουμε τον τύπο του με τη λέξη **Text**. Επιπλέον, για να χρησιμοποιήσουμε ένα αρχείο για είσοδο ή έξοδο πρέπει να δώσουμε και το πραγματικό όνομα του αρχείου του δίσκου στο οποίο αναφέρεται. Αυτό επιτυγχάνεται με τη διαδικασία **assign**.

Παράδειγμα

```
VAR MATHITES: TEXT;  
...  
BEGIN  
...  
    ASSIGN (MATHITES, 'C:STUDENTS.DAT');
```

Το όνομα του αρχείου στο δίσκο (**c:students.dat**), στο οποίο αναφέρεται το αρχείο με τη χρήση της διαδικασίας **assign**, χρησιμοποιείται μόνο μία φορά. Όλες οι άλλες αναφορές του αρχείου στο πρόγραμμα γίνονται με το όνομά του, όπως αυτό δηλώθηκε στην περιοχή δήλωσης των μεταβλητών (**mathites**).

Οι εντολές που χρησιμοποιούμε στην **Turbo Pascal** για αρχεία κειμένου εξυπηρετούν τους παρακάτω σκοπούς:

Άνοιγμα ή κλείσιμο αρχείων

reset (F1)

Όπου **F1** είναι μεταβλητή τύπου αρχείου, όπως έχει καθοριστεί από την εντολή **assign**.

Ενέργεια: Η εντολή αυτή ανοίγει ένα αρχείο που υπάρχει, με δυνατότητα ανάγνωσης ή εγγραφής σε αυτό. Αν το αρχείο είναι τύπου **text**, τότε το αρχείο μπορεί να χρησιμοποιηθεί μόνο για ανάγνωση.

Παράδειγμα: ASSIGN(ARXΕΙΟ_SP, 'FOITITES.DAT');
RESET(ARXΕΙΟ_SP);

rewrite (F1)

Όπου **F1** είναι μεταβλητή τύπου αρχείου, όπως έχει καθοριστεί από την εντολή **assign**.

Ενέργεια: Η εντολή αυτή δημιουργεί ένα αρχείο για εγγραφή. Αν το αρχείο αυτό τότε τα δεδομένα που είναι αποθηκευμένα στο αρχείο καταστρέφονται.

Παράδειγμα: ASSIGN(ARXΕΙΟ_FOIT, 'FOITITES.DAT');
RESET(ARXΕΙΟ_FOIT);



close (F1)
Όπου F1 είναι μεταβλητή τύπου αρχείου, όπως έχει καθοριστεί από την εντολή assign .
<i>Ενέργεια:</i> Η εντολή αυτή κλείνει το αρχείο F1 και ενημερώνει τον κατάλογο (directory). Μετά τη χρήση του αρχείου και πριν τελειώσει το πρόγραμμα, χρησιμοποιείται η εντολή close για το κλείσιμο του αρχείου. Εάν δε χρησιμοποιηθεί και έχουμε ανοίξει αρχείο για εγγραφή, τότε είναι πιθανό να χάσουμε ορισμένα από τα τελευταία δεδομένα που γράφτηκαν.
<i>Παράδειγμα:</i> CLOSE(PELATES);
append (F1)
Όπου F1 είναι μεταβλητή τύπου αρχείου, όπως έχει καθοριστεί από την εντολή assign .
<i>Ενέργεια:</i> Ανοίγει ένα αρχείο τύπου κειμένου για εγγραφή και συγκεκριμένα για πρόσθεση νέων δεδομένων στο τέλος του αρχείου και τα νέα δεδομένα δε σβήνουν τα παλιά.
<i>Παράδειγμα:</i> APPEND(SXOLEIO);

Επεξεργασία αρχείων

read / readln (F1, M1, M2, ..., Mn)
Όπου F1 είναι μεταβλητή τύπου αρχείου, όπως έχει καθοριστεί από την εντολή assign και M1, M2,...,Mn είναι τα ονόματα των μεταβλητών.
<i>Ενέργεια:</i> Η εντολή read διαβάζει δεδομένα από το αρχείο F1 και τα αποθηκεύει στις μεταβλητές M1, M2,...,Mn . Η εντολή readln , η οποία χρησιμοποιείται μόνο σε αρχεία κειμένου, διαβάζει δεδομένα με τον ίδιο τρόπο αλλά μετά την εκτέλεσή της ο δείκτης του αρχείου μετακινείται προς την επόμενη γραμμή του κειμένου του αρχείου.
<i>Παράδειγμα:</i> VAR KOD: INTEGER; EPONYMIA: STRING[30]; LOIPA_STOICHEIA: STRING; ARXΕΙΟ_PELATWN: TEXT; BEGIN READLN(ARXΕΙΟ_PELATWN, KOD, EPONYMIA, LOIPA_STOICHEIA);
write / writeln (F1, M1, M2, ..., Mn)
Όπου F1 είναι μεταβλητή τύπου αρχείου, όπως έχει καθοριστεί από την εντολή assign και M1, M2,...,Mn είναι τα ονόματα των μεταβλητών.
<i>Ενέργεια:</i> Η εντολή write γράφει στο αρχείο F1 τα δεδομένα που είναι αποθηκευμένα στις μεταβλητές M1, M2,...,Mn . Η εντολή writeln , η οποία χρησιμοποιείται μόνο σε αρχεία κειμένου, γράφει δεδομένα με τον ίδιο τρόπο αλλά μετά την εκτέλεσή της ο δείκτης του αρχείου μετακινείται προς την επόμενη γραμμή του κειμένου του αρχείου.
<i>Παράδειγμα:</i> VAR KOD: INTEGER; EPONYMIA: STRING[30]; LOIPA_STOICHEIA: STRING; PELATES: TEXT; BEGIN WRITELN(PELATES, KOD, EPONYMIA, LOIPA_STOICHEIA);



eof (F1)
Όπου F1 είναι μεταβλητή τύπου αρχείου, όπως έχει καθοριστεί από την εντολή assign .
<i>Ενέργεια:</i> Η συνάρτηση είναι boolean και επιστρέφει την τιμή true , αν βρεθεί η ένδειξη τέλους του αρχείου (τέλος δεδομένων), αλλιώς, επιστρέφει την τιμή false . Αν παραληφθεί η μεταβλητή F1 , τότε ως αρχείο λαμβάνεται η τυποποιημένη μονάδα εισόδου (standard input) που είναι το πληκτρολόγιο.
<i>Παράδειγμα:</i> IF EOF(ARXΕΙΟ_MATH) THEN WRITELN('TELOS ARXΕΙΟΥ!!');
seekeof (F1)
Όπου F1 είναι μεταβλητή τύπου αρχείου, όπως έχει καθοριστεί από την εντολή assign .
<i>Ενέργεια:</i> Η συνάρτηση είναι boolean και επιστρέφει την τιμή true , αν βρεθεί η ένδειξη τέλους του αρχείου (τέλος δεδομένων), αλλιώς, επιστρέφει την τιμή false . Είναι αντίστοιχη της συνάρτησης eof με τη διαφορά ότι παραλείπει όλα τα κενά και τους χαρακτήρες τέλους γραμμής.
eoln (F1)
Όπου F1 είναι μεταβλητή τύπου αρχείου, όπως έχει καθοριστεί από την εντολή assign .
<i>Ενέργεια:</i> Η συνάρτηση δηλώνει αν υπάρχουν ακόμη χαρακτήρες στην τρέχουσα γραμμή. Η συνάρτηση επιστρέφει την τιμή true , αν διαβάζοντας βρει το τέλος της γραμμής ή το τέλος του αρχείου, αλλιώς επιστρέφει την τιμή false .
<i>Παράδειγμα:</i> IF EOLN(ARXΕΙΟ_MATH) THEN WRITELN('TELOS GRAMMIS!!');
IOresult
<i>Ενέργεια:</i> Η συνάρτηση επιστρέφει μία ακέραια τιμή που δείχνει την κατάσταση της τελευταίας εντολής εισόδου/ εξόδου/ Η συνάρτηση είναι χωρίς παράμετρο. Αν η τελευταία λειτουργία εισόδου/ εξόδου τελειώσει επιτυχώς, τότε η συνάρτηση επιστρέφει την τιμή 0 (μηδέν). Για να λειτουργήσει η συνάρτηση πρέπει να έχει δηλωθεί προς τον compiler της Pascal η δήλωση {SI-} που δίνει στο χρήστη τη δυνατότητα ελέγχου των λαθών που έχουν σχέση με είσοδο και έξοδο. Αν δεν έχει γίνει αυτή η δήλωση και υπάρχει λάθος σε λειτουργία εισόδου/ εξόδου, τότε σταματάει η εκτέλεση που προγράμματος με αντίστοιχο μήνυμα.
<i>Παράδειγμα:</i> READLN (ARXΕΙΟ_PELATWN,KOD, EPONYMIA,LOIPA_STOIXEIA); IF IORESULT <> 0 THEN REWRITE(MATHITES);

Χειρισμός αρχείων

assign (F1, F2)
Όπου F1 είναι μεταβλητή τύπου αρχείου, και F2 το όνομα του αρχείου στο Λειτουργικό Σύστημα.
<i>Ενέργεια:</i> Η εντολή αυτή αντιστοιχίζει στη μεταβλητή F1 το όνομα του αρχείου F2 , έτσι όπως έχει οριστεί στο Λειτουργικό Σύστημα.
<i>Παράδειγμα:</i> ASSIGN(ARX_PELATES, 'C:PELATES.DAT');
erase (F1)
Όπου F1 είναι μεταβλητή τύπου αρχείου, όπως έχει καθοριστεί από την εντολή assign .



<i>Ενέργεια:</i> Η εντολή διαγράφει από το δίσκο το αρχείο που αντιστοιχεί στην μεταβλητή F1 .
<i>Παράδειγμα:</i> ERASE(ARX_PELATES);
rename (F1, F2)
Όπου F1 είναι μεταβλητή τύπου αρχείου, όπως έχει καθοριστεί από την εντολή assign , και F2 μεταβλητή τύπου string .
<i>Ενέργεια:</i> Αλλάζει το όνομα του αρχείου F1 με το όνομα που αντιστοιχεί στο string F2 .
<i>Παράδειγμα:</i> RENAME(ARXΕΙΟ_SP, 'FOITITES.DAT');

Χρήση του εκτυπωτή

Η χρήση του εκτυπωτή από την **Pascal** γίνεται μέσα από εντολές διαχείρισης των αρχείων. Ο εκτυπωτής μπορεί να θεωρηθεί ότι είναι ένα αρχείο κειμένου στο οποίο μπορούμε μόνο να γράψουμε. Παρακάτω δίνεται ένα παράδειγμα χρήσης του εκτυπωτή.

```
PROGRAM EKTYPOTI;
USES WINCRT;
{δήλωση για χρήση εντολών εισόδου/ εξόδου σε Turbo Pascal για Windows}
VAR
    EKTYP: TEXT;
    {δήλωση της μεταβλητής ektyp ως αρχείου τύπου text}
BEGIN
    ASSIGN(EKTYP, 'LPT1');
    {αντιστοίχιση της θύρας LPT1 του υπολογιστή στη μεταβλητή ektyp}
    REWRITE(EKTYP); {άνοιγμα επικοινωνίας με τον εκτυπωτή}
    WRITELN(EKTYP, 'THIS IS A TEST');
    CLOSE(EKTYP); {Κλείσιμο επικοινωνίας με τον εκτυπωτή}
END.
```

-----*

Παραδείγματα

[1] Εγγραφή σε αρχείο κειμένου

```
PROGRAM PARADEIGMA_14;
VAR FNAME, TXT : STRING;
USERFILE : TEXT;
BEGIN
    WRITE('DOSE ONOMA ARXEIOU (PX.C:\TEST.TXT) :');
    READLN(FNAME);
    ASSIGN(USERFILE, FNAME);
    REWRITE(USERFILE);
    WRITELN('GRAPSE TO KIMENO : ');
    READLN(TXT);
    WRITELN(USERFILE, TXT);
    CLOSE(USERFILE);
END.
```

Επεξήγηση Προγράμματος

UserFile : Text Δηλώνουμε ότι το αρχείο θα περιέχει κείμενο(**text**).



Assign(UserFile, FName); Αντιστοίχιση του αρχείου. Περνούν τα ορίσματα τύπος αρχείου και όνομα αρχείου(πλήρης διαδρομή). Δηλαδή το αρχείο **FName** θα είναι τύπου **UserFile** δηλαδή κείμενο.

Άνοιγμα του αρχείου **FName** για εγγραφή (αν έχει ήδη κείμενο το αρχείο σβήνεται και γράφει το νέο).

Κάνοντας χρήση του **Append(UserFile);** προσθέτουμε το κείμενο στο τέλος του αρχείου (εφόσον υπάρχει το αρχείο !!!) κρατώντας το προηγούμενο κείμενο.

WriteIn(UserFile,Txt); Εγγραφή κειμένου στο αρχείο (όχι στην οθόνη).

Close(UserFile); Κλείσιμο του αρχείου **FName** (απαιτείτε).

[2] Ανάγνωση από αρχείο κειμένου

```
PROGRAM PARADEIGMA_15;  
VAR USERFILE : TEXT;  
FILENAME, TFILE : STRING;  
BEGIN  
    WRITE( 'DOSE ONOMA ARXEIOU (PX.C:\TEST.TXT) :' );  
    READLN(FILENAME);  
    ASSIGN(USERFILE, FILENAME);  
    RESET(USERFILE);  
    REPEAT  
        READLN(USERFILE, TFILE);  
        WRITELN(TFILE);  
    UNTIL EOF(USERFILE);  
    CLOSE(USERFILE);  
    READLN;  
END.
```

Επεξήγηση Προγράμματος

Reset(UserFile);

Άνοιγμα του αρχείου **FileName** για ανάγνωση.

...

Repeat

ReadIn(UserFile,TFile);

WriteIn(TFile);

Until Eof(UserFile);

...

Διάβαζε από το αρχείο **ReadIn(UserFile,TFile);** Και να παρουσιάζεις στην οθόνη **WriteIn(TFile);** μέχρι να φτάσεις στο τέλος του αρχείου **Until Eof(UserFile);**.

(Eof – End of file)

Close(UserFile); Κλείσιμο του αρχείου **FileName** (απαιτείτε).

Παρατηρήσεις: Η ανάγνωση του αρχείου γίνεται γραμμή - γραμμή μέχρι το τέλος του **EOF**.

[3] Διαγραφή Αρχείου (Προσοχή Κίνδυνος !!!)

```
PROGRAM PARADEIGMA_16;  
VAR UFILE : TEXT;  
FILENAME : STRING;  
BEGIN
```



```
WRITE( 'DOSE ONOMA ARXEIOU (PX.C:\TEST.TXT) :');  
READLN(FILENAME);  
ASSIGN(UFILE, FILENAME);  
ERASE(UFILE);  
END.
```

Επεξήγηση Προγράμματος

Erase(UFile); Άνοιγμα του αρχείου **FileName** για διαγραφή.

[4] Να γράψετε μια διαδικασία (**procedure**) στην **Pascal** που θα ζητά τις πληροφορίες: όνομα, επίθετο, **email**, τηλέφωνο και να τις αποθηκεύει σε αρχείο κειμένου(**Append**). Στη συνέχεια θα γίνεται ερώτηση αν θέλουμε να συνεχίσουμε να προσθέτουμε δεδομένα στο αρχείο. Αν η απάντηση είναι θετική θα ξανακαλείται η διαδικασία, ενώ αν είναι αρνητική θα τερματίζεται το πρόγραμμα.

Τα δεδομένα να αποθηκεύονται στο αρχείο στην εξής μορφή:

Onoma : Toto
Epitheto : Blabla
email: mymail@teiep.com
Tilefono : 2109999999

```
PROGRAM PARADEIGMA_17;  
  
PROCEDURE INSERT();  
VAR  
    USERFILE : TEXT;  
    NAME, SURNAME, MAIL, PHONE : STRING;  
    YN : CHAR;  
BEGIN  
    ASSIGN(USERFILE, 'C:\PHONEBOOK.TXT');  
{TO ARXEIO C:\PHONEBOOK.TXT PREPEI NA YPARXEI GIA NA MPORESEI NA GINEI  
APPEND !!!}  
    APPEND(USERFILE);  
    WRITE( 'ONOMA: ');  
    READLN(NAME);  
    WRITE( 'EPITHETO: ');  
    READLN(SURNAME);  
    WRITE( 'EMAIL: ');  
    READLN(MAIL);  
    WRITE( 'TILEFONO: ');  
    READLN(PHONE);  
    WRITELN(USERFILE, 'ONOMA: ' + NAME);  
    WRITELN(USERFILE, 'EPITHETO: ' + SURNAME);  
    WRITELN(USERFILE, 'EMAIL: ' + MAIL);  
    WRITELN(USERFILE, 'TILEFONO: ' + PHONE);  
    WRITELN(USERFILE, '');  
    CLOSE(USERFILE);  
    WRITE( 'THELETE NA SINEXISETE (Y/N) :');  
    READLN(YN);  
    IF(YN='Y') THEN  
        BEGIN  
            CLRSCR; {KATHARISMOS TIS OTHONIS}  
            INSERT();  
        END  
    ELSE  
        BEGIN  
            WRITE( 'BYE BYE ...');  
            EXIT; {EXODOS APO TO PROGRAMMA}
```



```
        END;  
    END;  
  
BEGIN  
    INSERT( );  
    READLN;  
END.
```

Αρχεία Με Τύπο

Τα **αρχεία με τύπο (typed files)** είναι αρχεία, όπου οι αριθμητικές τιμές απεικονίζονται σε αριθμητική μορφή και όχι σε μορφή χαρακτήρων, όπως στα αρχεία κειμένου. Οι συνιστώσες του αρχείου μπορεί να είναι κάθε τύπου, απλού, σύνθετου, με δόμηση κτλ. με εξαίρεση τον τύπο αρχείου. Τα αρχεία αποτελούνται από εγγραφές ενός συγκεκριμένου τύπου πχ ακεραίου, πραγματικού κτλ. Η δόμηση των αρχείων ορισμένου τύπου προσδιορίζεται από τον τύπο των δεδομένων τα οποία θα αποθηκευτούν σε αυτά.

Για να χρησιμοποιήσουμε ένα αρχείο, πρέπει να του δώσουμε ένα όνομα και να δηλώσουμε ότι είναι **αρχείο (file)**, και να δηλώσουμε τον τύπο των δεδομένων που αποθηκεύονται σε αυτό.

Παραδείγματα

```
VAR  
    ETH: FILE OF INTEGER;  
    TEMP: FILE OF REAL;  
    SURNAMES: FILE OF STRINGS[20];
```

Τα δεδομένα είναι οργανωμένα σε **εγγραφές (records)** και κάθε δεδομένο αποτελεί μία εγγραφή. Το μήκος μίας εγγραφής είναι ο αριθμός των **bytes** τα οποία απαιτούνται για την αποθήκευση του συγκεκριμένου τύπου.

Οι εντολές **reset, rewrite, read, write, append, assign, close, erase, rename** και **eof** που είναι διαθέσιμες από τη γλώσσα **Pascal** για τα αρχεία κειμένου ισχύουν με την ίδια σημασία και για τα αρχεία με τύπο. Οι εντολές **read** και **write** είναι απλοποιημένες, αφού μία μόνο συνιστώσα διαβάζεται ή γράφεται στο αρχείο κάθε φορά που εκτελούνται. Ακόμη, επειδή δεν είναι αρχεία κειμένου, δε γίνεται μετάφραση από το αρχείο στη μνήμη και από τη μνήμη στο αρχείο.

Τα αρχεία με τύπο ονομάζονται και **δυναδικά (binary)** γιατί η εσωτερική απεικόνιση των δεδομένων στη μνήμη είναι ίδια με την εξωτερική απεικόνιση των δεδομένων στο αρχείο. Τα δυναδικά αρχεία διαθέτουν έναν αριθμό σειράς για κάθε συνιστώσα (εγγραφή) αρχίζοντας από το μηδέν. Εκτός από τη σειριακή προσπέλαση που μπορεί να γίνει με την εντολή **read**, η οποία κάθε φορά διαβάζει την επόμενη εγγραφή, μία τυχαία εγγραφή μπορεί να βρεθεί αν δοθεί ο αριθμός της σειράς της. Η άμεση ή τυχαία προσπέλαση επιτυγχάνεται με έναν ειδικό δείκτη, το **δείκτη εγγραφής αρχείου (file pointer)**. Οι ενέργειες είναι:

1. Ενημερώνεται ο δείκτης αρχείου με τον αριθμό της εγγραφής του αρχείου που θέλουμε να διαβάσουμε ή να γράψουμε.
2. Χρησιμοποιείται η αντίστοιχη εντολή ανάγνωσης ή εγγραφής, η οποία εκτελείται πάνω στην τρέχουσα εγγραφή.

seek (F1, R)

Όπου **F1** είναι μεταβλητή τύπου αρχείου, όπως έχει καθοριστεί από την εντολή **assign**, και **R** η θέση της προς αναζήτηση εγγραφής στο αρχείο.

Ενέργεια: Η εντολή, η οποία είναι διαδικασία, τοποθετεί το δείκτη ανάγνωσης/εγγραφής στο **record** με αριθμό **R** για μελλοντικό **read** ή **write**. Η εντολή δε λειτουργεί σε αρχεία τύπου **TEXT**.



Η πρώτη εγγραφή του αρχείου έχει αριθμό μηδέν (**0**).

Παράδειγμα: SEEK (PELATES, AR);

READ (PELATES, ΣΤΟΙΧΕΙΑ_PELATH);

{ΤΟΠΟΘΕΤΗΣΕ ΤΟ ΔΕΙΚΤΗ ΑΡΧΕΙΟΥ (FILE POINTER) ΣΤΗΝ ΕΓΓΡΑΦΗ ΜΕ ΑΡΙΘΜΟ AR ΤΟΥ ΑΡΧΕΙΟΥ PELATES, ΔΙΑΒΑΣΕ ΑΠΟ ΤΟ ΑΡΧΕΙΟ PELATES ΤΗΝ ΕΓΓΡΑΦΗ ΠΟΥ ΔΕΙΧΝΕΙ Ο ΔΕΙΚΤΗΣ ΤΟΥ ΑΡΧΕΙΟΥ ΚΑΙ ΤΟΠΟΘΕΤΗΣΕ ΤΑ ΔΕΔΟΜΕΝΑ ΣΤΗ ΜΕΤΑΒΛΗΤΗ ΣΤΟΙΧΕΙΑ_PELATH }

filepos (F1)

Όπου **F1** είναι μεταβλητή τύπου αρχείου, όπως έχει καθοριστεί από την εντολή **assign**.

Ενέργεια: Η συνάρτηση επιστρέφει τη θέση του αρχείου όπου θα διαβαστεί ή θα γραφεί η επόμενη εγγραφή. Στην αρχή του αρχείου η συνάρτηση επιστρέφει την τιμή μηδέν (**0**), στο τέλος του αρχείου επιστρέφει την ίδια τιμή με τη **filesize**.

Παράδειγμα: WRITELN('THESEI ΤΟΥ RECORD:', FILEPOS(MATHITES));

filesize (F1)

Όπου **F1** είναι μεταβλητή τύπου αρχείου, όπως έχει καθοριστεί από την εντολή **assign**.

Ενέργεια: Η συνάρτηση επιστρέφει το μέγεθος του αρχείου σε πλήθος **record**. Η συνάρτηση δε μπορεί να χρησιμοποιηθεί σε αρχεία τύπου **TEXT**.

Παράδειγμα: WRITELN('MAGETHOS ARXEIOU:', FILESIZE(MATHITES));

Παράδειγμα

```
PROGRAM PARADEIGMA_18;
VAR A, I: INTEGER;
VAR FP1: FILE OF INTEGER;

BEGIN
    ASSIGN(FP1, 'TEMPO.DAT');
    REWRITE(FP1);
    FOR I:=1 TO 5 DO
        BEGIN
            WRITE('DOSE ENAN AKERAIIO: ');
            READLN(A);
            WRITE(FP1, A);
        END;
    CLOSE(FP1);
    RESET(FP1);
    WRITELN('TA DEDOMENA ΤΟΥ ΑΡΧΕΙΟΥ ΕΙΝΑΙ:');
    WHILE NOT EOF(FP1) DO
        BEGIN
            READ(FP1, A);
            WRITELN(A);
        END;
    WRITELN;
    CLOSE(FP1);
END.
```



Προχωρημένα θέματα στην PASCAL

Μέρος Γ΄

*Πίνακες,
Εγγραφές*



Πίνακες

Χρησιμοποιούμε **πίνακες**, όταν θέλουμε να επεξεργαστούμε ένα πλήθος ομοειδών στοιχείων. Τα στοιχεία ενός πίνακα είναι πάντα του ίδιου τύπου, π.χ. ακέραιοι αριθμοί, πραγματικοί αριθμοί, συμβολοσειρές κλπ., και αναφέρονται με κοινό όνομα (το όνομα του πίνακα) και τη θέση τους στο πίνακα. Οι διαστάσεις του πίνακα στην **Pascal** πρέπει να είναι γνωστές εκ των προτέρων (να δηλώνονται δηλαδή στο τμήμα δηλώσεων του προγράμματος). Τα στοιχεία του πίνακα αποθηκεύονται σε συνεχόμενες θέσεις στη μνήμη.

Μια μεταβλητή πίνακα δηλώνεται ως εξής:

```
VAR <όνομα πίνακα>:ARRAY[<διαστάσεις>] OF <τύπος στοιχείων>;
```

Συγκεκριμένα για το μονοδιάστατο πίνακα **A** με **10** στοιχεία:

```
VAR A:ARRAY[1..10] OF INTEGER;
```

Το όνομα του πίνακα είναι **A** (επιλέγεται από τον προγραμματιστή), τα στοιχεία του πίνακα είναι ακέραιοι αριθμοί και καθένα από αυτά αναφέρεται ως **A[1]**, **A[2]**, **A[3]**, ..., **A[10]**

Για πίνακα δύο διαστάσεων **A** με **3** γραμμές και **4** στήλες:

```
VAR A:ARRAY[1..3,1..4] OF REAL;
```

Το όνομα του πίνακα είναι **A** (επιλέγεται από τον προγραμματιστή), τα στοιχεία του πίνακα είναι πραγματικοί αριθμοί και καθένα από αυτά αναφέρεται ως **A[I,J]**, όπου το **I** έχει τιμές **1, 2, ή 3** και το **J** έχει τιμές **1, 2, 3 ή 4**.

```
A[1,1] A[1,2] A[1,3] A[1,4]
A[2,1] A[2,2] A[2,3] A[2,4]
A[3,1] A[3,2] A[3,3] A[3,4]
```

Βασικές λειτουργίες που μπορεί να γίνουν στα στοιχεία ενός πίνακα είναι:

- Ανάγνωση των στοιχείων του πίνακα από το πληκτρολόγιο ή απόδοση τιμής στα στοιχεία με κάποιο τύπο.
- Εκτύπωση των στοιχείων του πίνακα ή εμφάνισή τους στην οθόνη.
- Διαπέραση του πίνακα για αναζήτηση των στοιχείων που ικανοποιούν κάποια συνθήκη.
- Επεξεργασία των στοιχείων του πίνακα (για εύρεση αθροίσματος, μέσου όρου, μέγιστου και ελάχιστου στοιχείου, κλπ.).
- Ταξινόμηση των στοιχείων του πίνακα, με στόχο την ευκολότερη και ταχύτερη αναζήτηση.

Επειδή οι λειτουργίες αυτές εκτελούνται σε όλα τα στοιχεία του πίνακα και δεδομένου ότι οι διαστάσεις του πίνακα είναι γνωστές (άρα και ο αριθμός των στοιχείων) για την επεξεργασία του πίνακα ενδείκνυται η χρήση της εντολής **FOR**.

Π.χ. για να διαβάσουμε το μονοδιάστατο πίνακα **A** με **10** στοιχεία που ορίσαμε πιο πάνω χρησιμοποιούμε:

```
FOR I:=1 TO 10 DO
  READLN(A[I]);
```

Αν θέλουμε να είναι πιο ολοκληρωμένο και να ξέρουμε πιο στοιχείο του πίνακα διαβάζουμε κάθε φορά χρησιμοποιούμε:

```
FOR I:=1 TO 10 DO
  BEGIN
    WRITE('ΔΟΣΕ ΣΤΟΙΧΕΙΟ A[ ',I,' ]');
    READLN(A[I]);
  END;
```



Για να διαβάσουμε το πίνακα δύο διαστάσεων **A** με **3** γραμμές και **4** στήλες που ορίσαμε πιο πάνω χρησιμοποιούμε:

```
FOR I:=1 TO 3 DO
FOR J:=1 TO 4 DO
    READLN(A[I, J]);
```

Αν θέλουμε να είναι πιο ολοκληρωμένο και να ξέρουμε πιο στοιχείο του πίνακα διαβάζουμε κάθε φορά χρησιμοποιούμε:

```
FOR I:=1 TO 3 DO
FOR J:=1 TO 4 DO
BEGIN
    WRITE('DOSE STOIXEIO A[ ', I, ', ', J, ' ]');
    READLN(A[I, J]);
END;
```

Αναζήτηση Στοιχείων Σε Πίνακα: Σειριακή Αναζήτηση (Serial Search)

Θα θεωρήσουμε πίνακα **A** με στοιχεία ακέραιους αριθμούς και θα ελέγξουμε αν υπάρχει στο πίνακα αυτό στοιχείο του οποίου η τιμή να είναι ίση με ένα συγκεκριμένο αριθμό **N**. Αν τέτοιο στοιχείο υπάρχει, η ζητούμενη απάντηση είναι η θέση του στο πίνακα. Αν όχι, θα πάρουμε σαν απάντηση τον αριθμό **0**. Θα εξετάσουμε όλα τα στοιχεία του πίνακα ξεκινώντας από το πρώτο στοιχείο του συγκρίνοντάς το με το αριθμό **N**. Ο αλγόριθμος σε γλώσσα **Pascal** που λύνει το παραπάνω πρόβλημα είναι ο ακόλουθος:

```
PROGRAM SERIAL_SEARCH_A;
VAR A:ARRAY[1..10] OF INTEGER;
I, N, M: INTEGER; F: BOOLEAN;
BEGIN
    FOR I:=1 TO 10 DO
        BEGIN
            WRITE('DOSE STOIXEIO PINAKA A[', I, ']= ');
            READLN(A[I]);
        END;
    WRITELN;
    WRITE('DOSE ARITHMO TON OPOIO ANAZHTAS ');
    READLN(N);
    M:=0;
    F:=FALSE;
    FOR I:=1 TO 10 DO
        IF A[I]=N THEN
            BEGIN
                M:=I;
                F:=TRUE;
            END;
    WRITELN('O ARITHMOS ', N, ' BRETHIKE STH THESI ', M);
END.
```

Η μεταβλητή **I** δείχνει το πλήθος των στοιχείων που έχουν εξεταστεί. Η τιμή της αυξάνει κατά **1** σε κάθε εκτέλεση της εντολής **FOR**, για να εξεταστεί το επόμενο στοιχείο του πίνακα. Η μεταβλητή **M** έχει αρχική τιμή **0**. Η τιμή αυτή δεν αλλάζει παρά μόνο όταν βρεθεί στοιχείο με τιμή **N**. Αν αυτό είναι το **I**-οστό στοιχείο του πίνακα, τότε στο **M** εκχωρείται η τιμή **I**.

Στο παραπάνω πρόγραμμα σειριακής αναζήτησης η επαναληπτική δομή **FOR** συνεχίζει τον έλεγχο, ακόμη και αν το στοιχείο του πίνακα βρεθεί, ώσπου να φτάσει και στο τελευταίο



στοιχείο του πίνακα. Αν στον πίνακα υπάρχουν περισσότερα από ένα στοιχεία με την τιμή N, ως αποτέλεσμα θα έχουμε την τελευταία από τις θέσεις στην οποία υπάρχει το στοιχείο N.

Μια πιο βελτιωμένη έκδοση του αλγόριθμου θα ήταν να σταματήσει η επαναληπτική δομή, μόλις βρεθεί το πρώτο στοιχείο με τιμή N. Ο αλγόριθμος σε γλώσσα **Pascal** που λύνει το παραπάνω πρόβλημα είναι ο ακόλουθος:

```
PROGRAM SERIAL_SEARCH_B;  
VAR A:ARRAY[1..10] OF INTEGER;  
I, N, M: INTEGER; F: BOOLEAN;  
BEGIN  
    FOR I:=1 TO 10 DO  
        BEGIN  
            WRITE('DOSE STOIXEIO PINAKA A[' , I, ']= ');  
            READLN(A[I]);  
        END;  
    WRITELN;  
    WRITE('DOSE ARITHMO TON OPOIO ANAZHTAS ');  
    READLN(N);  
    M:=0;  
    F:=FALSE;  
    I:=0;  
    WHILE (I<>10) AND (F=FALSE) DO  
        BEGIN  
            I:=I+1;  
            IF A[I]=N THEN  
                BEGIN  
                    M:=I;  
                    F:=TRUE;  
                END;  
            END;  
        END;  
    WRITELN('O ARITHMOS ', N, ' BRETHIKE STH THESI ', M);  
END.
```

Ταξινόμηση Στοιχείων Σε Πίνακα: Ταξινόμηση Φυσαλίδας (Bubble Sort)

Θα θεωρήσουμε πίνακα A με στοιχεία **8** ακέραιους αριθμούς. Για να κατανοήσουμε καλύτερα τη λειτουργία του αλγόριθμου φυσαλίδας, ας φανταστούμε το πίνακα A κατακόρυφο και τους αριθμούς σαν φυσαλίδες. Ο αριθμός με τη μικρότερη τιμή είναι η πιο ελαφριά φυσαλίδα, ενώ ο αριθμός με τη μεγαλύτερη τιμή η πιο βαριά φυσαλίδα.

Στο πρώτο πέρασμα του πίνακα ξεκινάμε από το κάτω μέρος του πίνακα και πραγματοποιούμε διαδοχικές αλλαγές γειτονικών φυσαλίδων. Σε κάθε ζευγάρι φυσαλίδων που συναντάμε, αν η πιο κάτω φυσαλίδα είναι πιο ελαφριά τότε πηγαίνει από πάνω. Όταν ολοκληρωθεί το πρώτο πέρασμα η πιο ελαφριά φυσαλίδα βρίσκεται στην κορυφή.

Στο δεύτερο πέρασμα του πίνακα ξεκινάμε και πάλι από το κάτω μέρος του πίνακα και πραγματοποιούμε διαδοχικές ανταλλαγές μεταξύ γειτονικών φυσαλίδων. Σε κάθε ζευγάρι φυσαλίδων που συναντάμε, αν η πιο κάτω φυσαλίδα είναι πιο ελαφριά τότε πηγαίνει από πάνω.

Γνωρίζουμε ότι η πάνω-πάνω θέση έχει καταληφθεί από τη μικρότερη φυσαλίδα στο πρώτο πέρασμα, οπότε δεν έχει έννοια να φτάσουμε ως την κορυφή των φυσαλίδων. Όταν ολοκληρωθεί το δεύτερο πέρασμα, η δεύτερη πιο ελαφριά φυσαλίδα βρίσκεται στη δεύτερη θέση του πίνακα.



Βλέπουμε ότι απαιτείται ένα βήμα λιγότερο, εφόσον η σύγκριση των δύο πάνω στοιχείων δεν έχει έννοια. Παρόμοια είναι η διαδικασία στο τρίτο, τέταρτο, πέμπτο, έκτο και έβδομο πέρασμα όπως φαίνεται και στο ακόλουθο σχήμα.

1° πέρασμα								2° πέρασμα								3° πέρασμα							
3	3	3	3	3	3	3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
7	7	7	7	7	7	1	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2
8	8	8	8	8	1	7	7	7	7	7	7	7	2	3	3	3	3	3	3	3	3	3	3
2	2	2	2	1	8	8	8	8	8	8	8	2	7	7	7	7	7	7	7	6	6	6	6
6	6	6	1	2	2	2	2	2	2	2	2	2	8	8	8	8	8	8	8	6	7	7	7
10	10	1	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	8	8	8	8	8
1	1	10	10	10	10	10	10	10	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10

4° πέρασμα				5° πέρασμα			6° πέρασμα		7° πέρασμα	
1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3	3
6	6	6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9	9	9	9
10	10	10	10	10	10	10	10	10	10	10

Ο αλγόριθμος σε γλώσσα **Pascal** που λύνει το παραπάνω πρόβλημα είναι ο ακόλουθος:

```

PROGRAM BUBBLE_SORT;
VAR A:ARRAY[1..8] OF INTEGER;
I, J, TEMP:INTEGER;
BEGIN
    FOR I:=1 TO 8 DO
        BEGIN
            WRITE('DOSE STOIXEIO PINAKA A[' , I, ']= ');
            READLN(A[I]);
        END;
    WRITELN;
    FOR I:=2 TO 8 DO
        FOR J:=8 DOWNT0 I DO
            IF A[J-1]>A[J] THEN
                BEGIN
                    TEMP:=A[J-1];
                    A[J-1]:=A[J];
                    A[J]:=TEMP;
                END;
        FOR I:=1 TO 8 DO

```



```
WRITELN( 'A[',I,']= ',A[I]);  
END.
```

Παραδείγματα

[1] (Παράδειγμα σε μονοδιάστατο πίνακα) Να γραφεί πρόγραμμα σε γλώσσα **Pascal** το οποίο να διαβάζει ένα μονοδιάστατο πίνακα ακέραιων αριθμών A με **10** θέσεις και υπολογίζει και να τυπώνει το μέσο όρο, το άθροισμα και το γινόμενο των στοιχείων του πίνακα A.

```
PROGRAM PARADEIGMA_19;  
USES CRT;  
VAR I,SUM:INTEGER; GIN:LONGINT; MO:REAL;  
A:ARRAY[1..10] OF INTEGER;  
BEGIN  
    CLRSCR;  
    FOR I:=1 TO 10 DO  
        BEGIN  
            WRITE( 'DOSE A[',I,']= ');  
            READ(A[I]);  
        END;  
    SUM:=0;GIN:=1;  
    FOR I:=1 TO 10 DO  
        BEGIN  
            SUM:=SUM+A[I];  
            GIN:=GIN*A[I];  
        END;  
    MO:=SUM/10;  
    WRITELN( 'TO ATHROISMA EINAI : ',SUM);  
    WRITELN( 'TO GINOMENO EINAI : ',GIN);  
    WRITELN( 'O MESOS OROS EINAI : ',MO:5:2);  
END.
```

[2] (Παράδειγμα σε πίνακα δύο διαστάσεων) 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.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

```
PROGRAM PARADEIGMA_20;  
USES CRT;  
VAR I,J,SUM:INTEGER; GIN:LONGINT; MO:REAL;  
A:ARRAY[1..3,1..3] OF INTEGER;  
BEGIN  
    CLRSCR;  
    FOR I:=1 TO 3 DO  
        FOR J:=1 TO 3 DO  
            BEGIN  
                WRITE( 'DOSE A[',I,' ',J,']= ');  
                READ(A[I,J]);  
            END;  
    SUM:=0;GIN:=1;  
    FOR I:=1 TO 3 DO  
        FOR J:=1 TO 3 DO  
            BEGIN
```



```
SUM:=SUM+A[I, J];
GIN:=GIN*A[I, J];

END;
MO:=SUM/9;
FOR I:=1 TO 3 DO
  BEGIN
    FOR J:=1 TO 3 DO
      WRITE(A[I, J], ' ');
      WRITELN;
    END;
    WRITELN('ΤΟ ΑΘΡΟΙΣΜΑ ΕΙΝΑΙ : ', SUM);
    WRITELN('ΤΟ ΓΙΝΟΜΕΝΟ ΕΙΝΑΙ : ', GIN);
    WRITELN('Ο ΜΕΣΟΣ ΟΡΟΣ ΕΙΝΑΙ : ', MO:5:2);
  END.
END.
```

Εγγραφές - Records

Η **εγγραφή (record)** είναι ένα σύνολο στοιχείων, τα οποία δεν είναι (υποχρεωτικά) του ίδιου τύπου μεταξύ τους. Τα στοιχεία του **record** λέγονται **πεδία (fields)**. Η δήλωση ενός **record** είναι η ακόλουθη:

```
TYPE <ΟΝΟΜΑ ΤΥΠΟΥ> = RECORD
  <ΟΝΟΜΑ ΠΕΔΙΟΥ 1>: <ΤΥΠΟΣ ΠΕΔΙΟΥ 1>;
  <ΟΝΟΜΑ ΠΕΔΙΟΥ 2>: <ΤΥΠΟΣ ΠΕΔΙΟΥ 2>;
  ...
  <ΟΝΟΜΑ ΠΕΔΙΟΥ N>: <ΤΥΠΟΣ ΠΕΔΙΟΥ N>;
END;
```

Παραδείγματα:

[1] Ας υποθέσουμε ότι θέλουμε να επεξεργαστούμε τα στοιχεία των σπουδαστών του Τμήματος Τεχνολογίας Πληροφορικής και Τηλεπικοινωνιών. Τα στοιχεία που θέλουμε να έχουμε είναι: Επώνυμο, Όνομα, Αριθμός Μητρώου, Χρονολογία Γέννησης, Φύλο και Βαθμολογία ενός Μαθήματος.

SURNAME	NAME	AM	BIRTHDAY	SEX	GRADE
ΣΩΤΗΡΙΟΥ	ΓΕΩΡΓΙΟΣ	145	1982	ΑΓΟΡΙ	7,5
ΓΕΩΡΓΙΟΥ	ΜΑΡΙΑ	210	1981	ΚΟΡΙΤΣΙ	8,2
ΣΤΑΜΑΤΙΟΥ	ΝΙΚΟΛΑΟΣ	172	1982	ΑΓΟΡΙ	6,7

```
TYPE STUDENT = RECORD
  SUR, NAME:STRING[20];
  AM, BIRTH:INTEGER;
  SEX:CHAR;
  GRADE:REAL;
END;
VAR STUD:STUDENT;
```

Στη παραπάνω δήλωση τύπου (**type**) ορίζουμε τον τύπο **STUDENT** με στοιχεία: το επώνυμο (**SUR**) και το όνομα (**NAME**) του σπουδαστή ως συμβολοσειρές μήκους **20** χαρακτήρων, τον αριθμό μητρώου (**AM**) και την χρονολογία γέννησης (**BIRTH**) ως ακέραιους αριθμούς, το φύλο (**SEX**) ως χαρακτήρα και τη βαθμολογία (**GRADE**) ως πραγματικό αριθμό.



Στη **VAR** δηλώνουμε μια μεταβλητή **STUD** η οποία είναι τύπου **STUDENT**. Το όνομα **STUDENT** δεν μπορεί να χρησιμοποιηθεί στις εντολές του προγράμματος, αφού έχει δηλωθεί ως τύπος δεδομένων και όχι ως μεταβλητή. Η αναφορά των πεδίων ενός **RECORD** γίνεται με τον τελεστή .

Η εντολή **STUD.SUR := 'ΣΩΤΗΡΙΟΥ'**; δίνει στο πεδίο **SUR** του **RECORD STUD** την τιμή ΣΩΤΗΡΙΟΥ.

Η εντολή **STUD.NAME := 'ΓΕΩΡΓΙΟΣ'**; δίνει στο πεδίο **NAME** του **RECORD STUD** την τιμή ΓΕΩΡΓΙΟΣ.

Η εντολή **STUD.AM := 145**; δίνει στο πεδίο **AM** του **RECORD STUD** την τιμή **145**.

Η εντολή **STUD.BIRTH := 1982**; δίνει στο πεδίο **BIRTH** του **RECORD STUD** την τιμή **1982**.

Η εντολή **STUD.SEX := 'M'**; δίνει στο πεδίο **SEX** του **RECORD STUD** την τιμή **M (Male -> Αγόρι)**.

Η εντολή **STUD.GRADE := 7.5**; δίνει στο πεδίο **GRADE** του **RECORD STUD** την τιμή **7.5**.

Πεδία του **RECORD** μπορεί να είναι και πίνακες. Στο πιο πάνω παράδειγμα θα μπορούσαμε να είχαμε τους βαθμούς (**GRADE**) **5** μαθημάτων οπότε η δήλωση του **RECORD** θα γινόταν ως εξής:

```
TYPE STUDENT = RECORD
    SUR, NAME:STRING[20];
    AM, BIRTH:INTEGER;
    SEX:CHAR;
    GRADE:ARRAY[1..5] OF REAL;
END;
VAR STUD:STUDENT;
```

Η εντολή **STUD.GRADE[1] := 7,5** δίνει στο πεδίο **GRADE[1]** του **RECORD STUD** την τιμή **7.5**. (Βαθμό για το 1ο μάθημα).

Επίσης στοιχεία ενός πίνακα μπορεί να είναι **RECORDS**. Στο πρώτο παράδειγμα που χρησιμοποιήσαμε θα μπορούσαμε να είχαμε τα στοιχεία για **50** μαθητές οπότε η δήλωση του **RECORD** θα γινόταν ως εξής:

```
TYPE STUDENT = RECORD
    SUR, NAME:STRING[20];
    AM, BIRTH:INTEGER;
    SEX:CHAR;
    GRADE:REAL;
END;
VAR STUD:ARRAY[1..50] OF STUDENT;
```

Η εντολή **STUD[1].SUR := 'ΣΩΤΗΡΙΟΥ'**; δίνει στο πεδίο **SUR** του **RECORD STUD[1]** την τιμή ΣΩΤΗΡΙΟΥ. (Για το 1ο Σπουδαστή)

Η εντολή **STUD[1].NAME := 'ΓΕΩΡΓΙΟΣ'**; δίνει στο πεδίο **NAME** του **RECORD STUD[1]** την τιμή ΓΕΩΡΓΙΟΣ. (Για το 1ο Σπουδαστή)

Η εντολή **STUD[1].AM := 145**; δίνει στο πεδίο **AM** του **RECORD STUD[1]** την τιμή **145**. (Για το 1ο Σπουδαστή)

Η εντολή **STUD[1].BIRTH := 1982**; δίνει στο πεδίο **BIRTH** του **RECORD STUD[1]** την τιμή **1982**. (Για το 1ο Σπουδαστή)

Η εντολή **STUD[1].SEX := 'M'**; δίνει στο πεδίο **SEX** του **RECORD STUD[1]** την τιμή **M (Male -> Αγόρι)**. (Για το 1ο Σπουδαστή)

Η εντολή **STUD[1].GRADE := 7.5**; δίνει στο πεδίο **GRADE** του **RECORD STUD[1]** την τιμή **7.5**. (Για το 1ο Σπουδαστή)

Επίσης θα μπορούσαμε να είχαμε συνδυασμό ή συνδυασμούς των πιο πάνω περιπτώσεων. π.χ. να είχαμε τα στοιχεία για **50** μαθητές και τους βαθμούς (**GRADE**) **5** μαθημάτων οπότε η δήλωση του **RECORD** θα γινόταν ως εξής:



```
TYPE STUDENT = RECORD
    SUR, NAME:STRING[20];
    AM, BIRTH:INTEGER;
    SEX:CHAR;
    GRADE:ARRAY[1..5] OF REAL;
END;
VAR STUD:ARRAY[1..50] OF STUDENT;
```

Η εντολή **STUD[1].GRADE[1] := 7.5;** δίνει στο πεδίο **GRADE[1]** του **RECORD STUD[1]** την τιμή **7.5**. (Το πρώτο μάθημα για το 1^ο Σπουδαστή)

[2] Να γραφεί πρόγραμμα σε γλώσσα **Pascal** το οποίο να διαβάζει τα στοιχεία **5** σπουδαστών (επώνυμο, αριθμό μητρώου, χρονολογία γέννησης και βαθμό στο μάθημα Προγραμματισμός Ι) του Τμήματος Τεχνολογίας Πληροφορικής και Τηλεπικοινωνιών. Να υπολογίζει το μέσο όρο στο μάθημα Δομές Δεδομένων και να τυπώνει τα στοιχεία των σπουδαστών που έχουν βαθμό μεγαλύτερο από το μέσο όρο.

```
PROGRAM PARADEIGMA_21;
USES CRT;
TYPE STUDENT=RECORD
    SUR:STRING[20];
    AM,BIRTH:INTEGER;
    DD:REAL;
END;

VAR STUD:ARRAY[1..5] OF STUDENT;
I:INTEGER;SUM,MO:REAL;

BEGIN
    CLRSCR;
    SUM:=0;
    FOR I:=1 TO 5 DO
        BEGIN
            WRITE('DOSE EPONYMO ',I,' OU SPOUDASTH : ');
            READLN(STUD[I].SUR);
            WRITE('DOSE AR. MHTROOU ',I,' OU SPOUDASTH : ');
            READLN(STUD[I].AM);
            WRITE('DOSE HM. GENNHSHS ',I,' OU SPOUDASTH : ');
            READLN(STUD[I].BIRTH);
            WRITE('DOSE BATHMO ',I,' OU SPOUDASTH : ');
            READLN(STUD[I].DD);
            SUM := SUM + STUD[I].DD;
        END;
    WRITELN;
    MO := SUM / 5;
    WRITELN('O MESOS OROS EINAI : ',MO:0:2);
    WRITELN;
    FOR I:=1 TO 5 DO
        IF STUD[I].DD > MO THEN
            WRITELN('EPONYMO:',STUD[I].SUR,'A.M.:',STUD[I].AM,'
            HM.GENNHSHS:',STUD[I].BIRTH,'BATHMOS:',STUD[I].DD:0:2);
    END.
```



Χρήσιμες αναφορές για την PASCAL

Μέρος Δ΄

Βιβλιογραφία

Βιβλιογραφία

- [1] Άρης Ν. Μπακάλης, «Η Γλώσσα Προγραμματισμού **Pascal**», Σχολή Διοίκησης Και Οικονομίας, Α.Τ.Ε.Ι. Πάτρας, **2009**.
- [2] Α. Σιδερίδης, Κ. Γιαλούρης, Σ. Μπακογιάννης, Κ. Σταθόπουλος, «Προγραμματισμός Υπολογιστών», Τ.Ε.Ε., Τομέας Πληροφορικής & Δικτύων Η/Υ, 1ος Κύκλος, Β' Τάξη, Έκδοση: ΟΕΔΒ, **2005**.
- [3] **Rodnay Zaks**, «Εισαγωγή στην **PASCAL** και την **Turbo PASCAL**», Εκδότης: Μ. Γκιούρδας.
- [4] **Jacques Tiberghien**, «Εγχειρίδιο **PASCAL**», Εκδότης: Μ. Γκιούρδας.
- [5] Ι. Στρουζας, Δ. Γκαρμόλας, «Δομές Δεδομένων Οργάνωση Αρχείων & Τεχνικές Προγραμματισμού σε Γλώσσα **Pascal & C++**», Εκδόσεις ΙΩΝ.
- [6] Π. Παπάζογλου, «**Turbo Pascal 6, 7 & Εφαρμογές**», Εκδόσεις ΙΩΝ.
- [7] Χ. Κοίλια, «Δομές Δεδομένων και Οργάνωση Αρχείων», Εκδόσεις Νέων Τεχνολογιών.
- [8] Γ. Γεωργακόπουλος, «Δομές Δεδομένων - Έννοιες, Τεχνικές και Αλγόριθμοι», Πανεπιστημιακές Εκδόσεις Κρήτης.
- [9] **Niklaus Wirth**, «Αλγόριθμοι & Δομές Δεδομένων», Εκδόσεις Κλειδάριθμος.