Χρυσάνθη Τζικούδη - Παπαγεωργίου

Εισαγωγή στη
γλώσσα προγραμματισμού

PASCAL

Θεσσαλονίκη, 2023

Χρυσάνθη Τζικούδη – Παπαγεωργίου

Εισαγωγή στη γλώσσα προγραμματισμού

PASCAL

Θεσσαλονίκη, 2023

ISBN: 978-618-00-4772-1

Αναφορά Δημιουργού - Μη Εμπορική Χρήση - παρόμοια Διανομή
CC BY-NC-SA

Περιεχόμενα

Εισαγωγή

Η γλώσσα προγραμματισμού Pascal είναι μία από τις πρώτες γλώσσες προγραμματισμού υψηλού επιπέδου που δημιουργήθηκαν. Πήρε το όνομά της από τον γάλλο μαθηματικό, φυσικό, εφευρέτη και φιλόσοφο Blaise Pascal. Δημιουργήθηκε από τον προγραμματιστή Niklaus Wirth στα τέλη της δεκαετίας του 1960 (1968). Ο Wirth ανέπτυξε την Pascal με στόχο να δημιουργήσει μια γλώσσα προγραμματισμού που να είναι εύκολη στη χρήση και κατανοητή από αρχάριους. Η πρώτη της έκδοση δημοσιεύτηκε το 1970. Έγινε γρήγορα δημοφιλής στην ακαδημαϊκή κοινότητα και την εκπαίδευση λόγω της ευκολίας της κατανόησης και της δομημένης προσέγγισης προγραμματισμού που προσφέρει. Έχει χρησιμοποιηθεί επίσης για τη δημιουργία λογισμικού σε διάφορους τομείς.

Μια από τις πιο γνωστές εφαρμογές της Pascal είναι το περιβάλλον προγραμματισμού Borland Turbo Pascal, το οποίο κυκλοφόρησε στις αρχές της δεκαετίας του 1980 (1983), επεκτάθηκε στο περιβάλλον MS-DOS και προσέλκυσε πολλούς προγραμματιστές λόγω της ευκολίας χρήσης και της ισχυρής υποστήριξης γραφικών και βιβλιοθηκών. Έγινε ιδιαίτερα δημοφιλής μέσω των προσωπικών υπολογιστών της εποχής όπως του IBM PC.

Σήμερα, η γλώσσα προγραμματισμού Pascal έχει χάσει κάποια από τη δημοτικότητά της σε σύγκριση με τα προηγούμενα χρόνια. Όμως, διάφορες οργανώσεις, πανεπιστήμια και εκπαιδευτικά ιδρύματα τη χρησιμοποιούν για διδασκαλία στους φοιτητές τους, καθώς η Pascal εξακολουθεί να θεωρείται μια αξιόπιστη γλώσσα για να μάθει κανείς τις βασικές αρχές του προγραμματισμού. Είναι ιδιαίτερα δημοφιλής στον εκπαιδευτικό τομέα, καθώς εισάγει τους νέους προγραμματιστές στις βασικές έννοιες του προγραμματισμού και της δομημένης προσέγγισης.

Μία εξέλιξη της Pascal που χρησιμοποιείται μέχρι και σήμερα είναι η γλώσσα προγραμματισμού Free Pascal (FPC). Είναι λογισμικό ανοικτού κώδικα που αναπτύχθηκε για να παρέχει μια σύγχρονη, δωρεάν παραλλαγή της Pascal. Παρέχει πληθώρα χαρακτηριστικών όπως υποστήριξη για διάφορες πλατφόρμες (Windows, Linux, macOS κ.τ.λ.), αναβαθμίσεις και υποστήριξη για πολλούς μεταγλωττιστές (compiler backends) που επιτρέπουν τη μεταγλώττιση του κώδικα σε διάφορες αρχιτεκτονικές. Οι προγραμματιστές που έχουν εμπειρία στη συγγραφή κώδικα σε Pascal μπορούν εύκολα να μεταβούν στη Free Pascal, καθώς οι βασικές αρχές και η σύνταξη της γλώσσας είναι παρόμοιες. Ωστόσο, η Free Pascal παρέχει περισσότερες δυνατότητες και ευελιξία, παράλληλα με την υποστήριξη πιο σύγχρονων τεχνολογιών και πλατφορμών.
Επίσημη ιστοσελίδα της FPC: https://www.freepascal.org/

Ορισμένοι από τους λόγους που η FPC συνεχίζει να έχει υποστηρικτές και να χρησιμοποιείται είναι:

1. *Διασυνδεσιμότητα*: Η Free Pascal μπορεί να μεταγλωττιστεί για πολλές πλατφόρμες, συμπεριλαμβανομένων Windows, Linux, macOS, και πολλών ακόμη. Αυτή η δυνατότητα της γλώσσας να τρέχει σε διαφορετικά λειτουργικά συστήματα την καθιστά ελκυστική για διασυνδεσιμότητα και διάθεση εφαρμογών.
2. *Εύκολη εκμάθηση*: Το συντακτικό της Free Pascal είναι σχετικά απλό και εύκολο, ειδικά για νέους προγραμματιστές, καθιστώντας την ιδανική για εκπαιδευτικούς σκοπούς.
3. *Εκτεταμένη βιβλιοθήκη*: Η Free Pascal διαθέτει μια ευρεία γκάμα βιβλιοθηκών που προσφέρουν πολλές λειτουργίες, συμπεριλαμβανομένων γραφικών, δικτύωσης, κρυπτογραφίας κ.ά.
4. *Λογισμικό ανοιχτού κώδικα*: Όντας ανοιχτού κώδικα, η Free Pascal επιτρέπει στους χρήστες να προσαρμόσουν τον μεταγλωττιστή και τις βιβλιοθήκες σύμφωνα με τις ανάγκες τους.

Αυτό το βιβλίο αποτελεί μια ηλεκτρονική έκδοση των σημειώσεων της συγγραφέα, οι οποίες συντάχθηκαν και χρησιμοποιήθηκαν για τις ανάγκες της διδασκαλίας της γλώσσας προγραμματισμού Pascal σε τεχνικά Επαγγελματικά Λύκεια (ΤΕΛ) κατά τη δεκαετία του 1990 (1996 - 1998), όταν δεν υπήρχε ακόμη ανάλογο σχολικό εγχειρίδιο. Σκοπός του βιβλίου είναι να προσφέρει μια συνολική και συνεκτική προσέγγιση της Pascal, για την εκμάθηση βασικών αρχών προγραμματισμού και αλγοριθμικής και την εξοικείωση με τον τρόπο σκέψης και τις δομές που απαιτούνται για την ανάπτυξη λογισμικού. Από τις βασικές αρχές της γλώσσας μέχρι την ανάπτυξη προηγμένων προγραμμάτων, το βιβλίο καλύπτει σημαντικά θέματα που θα βοηθήσουν τους αναγνώστες να κατανοήσουν τις βασικές αρχές του δομημένου προγραμματισμού, να ανακαλύψουν τις δυνατότητες που προσφέρει ο προγραμματισμός και να αναπτύξουν τις δεξιότητές τους στον τομέα αυτό.

Οι ασκήσεις του βιβλίου μπορούν να αποτελέσουν υλικό για την εκμάθηση, εμβάθυνση και ανάπτυξη εφαρμογών σε διάφορα προγραμματιστικά περιβάλλοντα.

1. Εισαγωγικές έννοιες

1.1 Πράξεις σε Pascal

ΠΡΑΞΗ	PASCAL
Πρόσθεση	+
Αφαίρεση	-
Πολλαπλασιασμός	*
Διαίρεση	/
Ακέραια διαίρεση	a div b
Υπόλοιπο διαίρεσης	a mod b
Τετραγωνική ρίζα	sqrt(x)
Ύψωση σε δύναμη	
Ύψωση στο τετράγωνο	**sqr**(x)
Ημίτονο χ	sin(x)
Συνημίτονο χ	cos(x)
Εφαπτομένη χ	
Απόλυτη τιμή	abs(x)
Αποκοπή δεκαδικών	trunc(x) ή **int**(x) *
Δεκαδικό μέρος πραγματικού αριθμού	frac(x) **
Στρογγύλευση δεκαδικού αριθμού	round(x)
lnx	ln(x)
e^x	exp(x)

* το αποτέλεσμα της trunc είναι longint, ενώ της int είναι real.

** π.χ. frac(12.57) = 0.57

1.2 Μορφή Turbo Pascal προγράμματος

```
program <όνομα προγράμματος>;

const
    <δηλώσεις σταθερών>;

var
    <δηλώσεις μεταβλητών>;

begin     (* κυρίως πρόγραμμα *)
    <εντολές>;
end.
```

Παράδειγμα 1

```
program greeting;
(* αυτό είναι ένα απλό pascal πρόγραμμα *)
begin
    writeln('H E L L O');
    readln;
end.
```

Κάθε δήλωση ή εντολή της Pascal πρέπει να έχει στο τέλος (;)

Παρατηρήσεις

- Το όνομα προγράμματος μπορεί να αποτελείται από λατινικά γράμματα, αριθμούς και την υπογράμμιση (_). πρέπει ν'αρχίζει από γράμμα και δεν πρέπει να ξεπερνά τους 118 χαρακτήρες.

- Οτιδήποτε γράφουμε ανάμεσα σε (* και *) θεωρείται σχόλιο και αγνοείται από τον μεταφραστή. Τα σχόλια μπορεί να παρεμβάλλονται ακόμη και στη μέση μιας εντολής.

- Οι εντολές του κυρίως προγράμματος μπαίνουν ανάμεσα σε begin και end. (Προσέξτε ότι το τελευταίο end του προγράμματος έχει το τέλος τελεία).

- Η εντολή readln μπαίνει στο τέλος των προγραμμάτων για να μπορούμε να δούμε τα αποτελέσματα στην οθόνη. Χωρίς αυτήν η turbo pascal μας επαναφέρει στον editor χωρίς να προλάβουμε να δούμε τα αποτελέσματα.

Παράδειγμα 2

```
program absum;

var
    a,b,sum: integer;   (* Δηλώσεις των μεταβλητών *)

begin
    a:=10;
    b:=20;
    sum:=a+b;
    writeln(sum);
end.
```

Όλες οι μεταβλητές ενός Pascal προγράμματος πρέπει να δηλώνονται. Με τη δήλωση μιας μεταβλητής, δεσμεύεται ο απαιτούμενος χώρος στη μνήμη του Η/Υ. Οι δηλώσεις των μεταβλητών γίνονται στο τμήμα δηλώσεων **var**.(Βλ. παρακάτω δηλώσεις μεταβλητών)

Στο παράδειγμα 2 υπάρχουν 3 μεταβλητές a, b, sum που είναι ακέραιου τύπου (integer). Ο τύπος integer καταλαμβάνει στη μνήμη χώρο 2 bytes (όπως θα μάθουμε παρακάτω). Ετσι για τις μεταβλητές του συγκεκριμένου προγράμματος δεσμεύονται στη μνήμη συνολικά 6 bytes.

1.3 Ονοματολογία μεταβλητών

Τα ονόματα των μεταβλητών πρέπει να ακολουθούν τους παρακάτω κανόνες:
α) Να αποτελούνται από 63 το πολύ χαρακτήρες (στην αρχική Pascal του Wirth μέχρι 8 το πολύ).
β) Μπορούν να περιέχουν λατινικά γράμματα, αριθμούς και την υπογράμμιση (_).
γ) πρέπει ν' αρχίζουν από γράμμα.
δ) Δεν πρέπει να είναι δεσμευμένες λέξεις π.χ. program, var, end κ.τ.λ.

Σωστά ονόματα: a, alpha, m2, Number_of_students
Λανθασμένα ονόματα: first+a, 2nd, begin, example 2

Το όνομα του προγράμματος δεν πρέπει να είναι το ίδιο όνομα με κάποια μεταβλητή του προγράμματος. Στην περίπτωση αυτή εμφανίζεται το μήνυμα λάθους dublicate identifier.

Κάθε μεταβλητή έχει όνομα, τύπο και τιμή. Το όνομα λέγεται και ταυτότητα. Αν ο τύπος δεν συμφωνεί με την τιμή εμφανίζεται το μήνυμα λάθους type mismatch.

Ερωτήσεις

1) Μπορεί να χρησιμοποιηθεί μια μεταβλητή χωρίς να δηλωθεί; .....
2) Είναι σωστή η εντολή; total (* άθροισμα *) := a+b; .....
3) Ποια από τα παρακάτω ονόματα μεταβλητών είναι λάθος και γιατί; a1, 2b, a1b2c3, a$, sum no, end, print

1.4 Τύποι της Turbo Pascal

Ονομα τύπου	**Εύρος τιμών**	**Χώρος στη μνήμη**
shortint (ακέραιος)	-128 .. 127	1 byte
byte (ακέραιος)	0 .. 255	1 byte
integer (ακέραιος)	-32768 .. 32767	2 bytes
word (ακέραιος)	0 .. 65535	2 bytes
longint (ακέραιος)	-2147483648 .. 2147483648	4 bytes
real (πραγματικός)	$-1.7\ 10^{38}$ .. $-2.9\ 10^{-39}$ και $2.9\ 10^{-39}$ .. $1.7\ 10^{38}$	6 bytes
char (1 χαρακτήρας)	1 χαρακτήρας	1 byte
string (αλφαριθμητικό)	Μέχρι 255 χαρακτήρες	256 bytes
boolean (λογικός)	true και false	1 byte

Ο ακέραιος τύπος integer

Δηλώνεται integer μια μεταβλητή που η τιμή της θα βρίσκεται μεταξύ του -32768 και του 32767 και θα είναι ακέραιος αριθμός.

Σωστές τιμές: 1234, 0, -234
Λανθασμένες τιμές: 1.25 (είναι πραγματικός και όχι ακέραιος)
200000 (είναι μεγαλύτερος του 32767)

1) Οι τελεστές div και mod χρησιμοποιούνται μόνο με ακέραιες μεταβλητές και όχι με real.
2) Όταν διαιρούμε τις τιμές δύο μεταβλητών ακέραιου τύπου, το αποτέλεσμα πρέπει να καταχωρείται σε μεταβλητή τύπου real.
Δηλ. στο παράδειγμα 2, αν αντί για πρόσθεση κάναμε διαίρεση (/), θα έπρεπε η μεταβλητή sum να είχε δηλωθεί τύπου real.

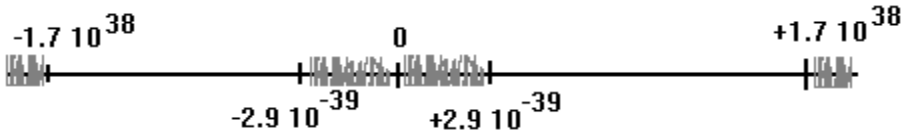
Προσοχή στις πράξεις με ακέραιους. π.χ. στον πολλαπλασιασμό δύο ακεραίων (integer), μπορεί το γινόμενο, ακόμη κι αν δηλωθεί longint ή real, να

ξεπερνά τα όρια του τύπου integer. Σ' αυτή την περίπτωση, το αποτέλεσμα θα είναι λανθασμένο. Για να έχουμε σωστό αποτέλεσμα, υπάρχουν δύο τρόποι:
α) η μία από τις δύο μεταβλητές που πολλαπλασιάζονται και η μεταβλητή του γινομένου να δηλωθούν τύπου longint ή real γιατί το γινόμενο integer * real δίνει αποτέλεσμα real και το γινόμενο integer * longint δίνει αποτέλεσμα longint.
β) Οι δύο μεταβλητές που πολλαπλασιάζονται (π.χ. i και j) να δηλωθούν integer, η μεταβλητή του γινομένου (π.χ. ginomeno) να δηλωθεί real και η εντολή καταχώρισης να είναι ως εξής:
ginomeno := **1.0** * i *j έτσι ώστε το γινόμενο 1.0 που θεωρείται real επί τους 2 ακέραιους να δίνει αποτέλεσμα real.

Γενικά ισχύει ο εμπειρικός κανόνας: **Κατά τον πολλαπλασιασμό δύο μεταβλητών, το γινόμενο μπορεί να πάρει τιμές που βρίσκονται μέσα στα όρια του μεγαλύτερου τύπου από τους τύπους των δύο μεταβλητών.** Υπάρχουν υποπεριπτώσεις οι οποίες όμως δεν επηρεάζουν σημαντικά τον κανόνα. π.χ. το γινόμενο byte*byte δημιουργεί πρόβλημα μόνο αν ξεπεράσει τα όρια του integer.

Ο πραγματικός τύπος real

Ο τύπος real χρησιμοποιείται για να καταχωρίσουμε στις μεταβλητές δεκαδικές τιμές ή ακέραιες τιμές που είναι έξω από τα όρια του longint.



Τα γραμμοσκιασμένα τμήματα του άξονα των πραγματικών αριθμών, δεν μπορούν να παρασταθούν από τον τύπο real.
Εμφανίζονται στη μορφή:
+ή- 1 ακέραιο ψηφίο.10 δεκαδικά ψηφία E +ή- 2 ακέραια ψηφία
π.χ. ο real 1.5000000000E+05 είναι ο αριθμός 150000
1.2560000000E+01 είναι ο αριθμός 12.56
-5.0000000000E+00 είναι ο αριθμός -5
2.5000000000E-01 είναι ο αριθμός 0.25

Εκτός από τον τύπο real, υπάρχουν κι άλλοι πραγματικοί τύποι με τους οποίους δε θ' ασχοληθούμε.

Ο τύπος string

Ο τύπος string χρησιμοποιείται για μεταβλητές, η τιμή των οποίων είναι ένα αλφαριθμητικό (γράμματα, ψηφία, σύμβολα). Μια μεταβλητή τύπου string

καταλαμβάνει στη μνήμη χώρο 256 bytes αλλά η τιμή της μπορεί να είναι μέχρι 255 χαρακτήρες. το ένα επιπλέον byte δεσμεύεται για να κρατηθεί εκεί το πραγματικό μήκος του string δηλ. πόσους πραγματικά χαρακτήρες πληκτρολογήσαμε για τιμή του string. π.χ. έστω η μεταβλητή onoma είναι τύπου string και της δώσαμε την τιμή ΜΑΡΙΑ. Στη μνήμη δεσμεύτηκαν 256 bytes αλλά η τιμή της είναι 5 χαρακτήρες. Ο αριθμός 5 αποθηκεύεται στο ένα επιπλέον byte που κρατήθηκε στη μνήμη.

Μπορούμε τον τύπο string να τον συμπληρώσουμε έτσι ώστε να δηλώνουμε πόσους το πολύ χαρακτήρες θα περιέχει η τιμή της μεταβλητής. Αυτό γίνεται με τη δήλωση:
string[n] όπου το n είναι ένας αριθμός από το 1 μέχρι το 255 και συμβολίζει το μέγιστο πλήθος των χαρακτήρων που μπορεί να δεχθεί η τιμή της μεταβλητής. π.χ. αν δηλώσουμε μια μεταβλητή τύπου **string[10]**, η τιμή της μπορεί να περιέχει το πολύ 10 χαρακτήρες. Σ' αυτή την περίπτωση **δεσμεύονται στη μνήμη 11 bytes**.

Ο τύπος char

Ο τύπος char χρησιμοποιείται για μεταβλητές η τιμή των οποίων είναι ένας χαρακτήρας. Καταλαμβάνει στη μνήμη χώρο 1 byte (ενώ ο τύπος string[1] καταλαμβάνει χώρο 2 bytes). Πολλές εντολές και συναρτήσεις της Pascal δέχονται παραμέτρους τύπου char (και όχι string[1]).

Ο λογικός τύπος boolean

Οι μεταβλητές τύπου boolean μπορούν να πάρουν δύο τιμές: true και false (αληθές και ψευδές). Δεν μπορούμε να τις δώσουμε τιμή με την εντολή readln (που θα μάθουμε παρακάτω), μπορούμε όμως να εμφανίσουμε την τιμή τους με την εντολή writeln.

Αν γράψουμε την εντολή readln(b) όπου το b είναι μια μεταβλητή τύπου boolean, θα εμφανισθεί το μήνυμα λάθους: Can not read or write variables of this type (δεν μπορώ να διαβάσω ή να γράψω μεταβλητές αυτού του τύπου, παρ' όλ' αυτά, η εντολή writeln εμφανίζει την τιμή τους).

1.5 Δηλώσεις μεταβλητών (var)

Το τμήμα δηλώσεων var μπαίνει πριν από το κυρίως πρόγραμμα (το οποίο αρχίζει με begin και τελειώνει με end).

```
var
  <όνομα μεταβλητής>:<τύπος>;
```

Αν υπάρχουν πολλές μεταβλητές του ίδιου τύπου μπορούμε να τις δηλώσουμε στην ίδια γραμμή, χωρίζοντάς τες με κόμμα.

π.χ. a,b,c : integer;

π.χ.

```
var
   name:string;
   aker:integer;
   x,y:real;
   epil:char;
```

Προσέξτε ότι το όνομα της μεταβλητής χωρίζεται
από τον τύπο με (:)

1.6 Δηλώσεις σταθερών (const)

```
const
    <όνομα σταθεράς>=<τιμή>;
```

Το όνομα της σταθεράς μπορεί να χρησιμοποιείται μέσα στο πρόγραμμα αντί της τιμής της. Η τιμή αυτή δεν επιτρέπεται ν'αλλάξει μέσα στο πρόγραμμα. Αν προσπαθήσουμε να την αλλάξουμε εμφανίζεται το μήνυμα λάθους Error in statement.

π.χ.

```
const
    pi=3.14;
    synt=10;
    onoma='ΘΕΣ/ΝΙΚΗ';
```

Προσέξτε ότι το όνομα της σταθεράς χωρίζεται από την τιμή με (=)

2. Οι βασικές εντολές της Pascal

2.1. Η εντολή καταχώρησης

Γενικός τύπος

<μεταβλητή> := <σταθερά> ή <άλλη μεταβλητή> ή <παράσταση>

Υπολογίζεται η τιμή δεξιά του := και τοποθετείται στη μεταβλητή αριστερά του :=
Η τιμή στο 2ο μέλος, θα πρέπει να είναι του ίδιου τύπου με τη μεταβλητή στο 1ο μέλος. Μία εξαίρεση σ' αυτόν τον κανόνα είναι η εξής: <μεταβλητή τύπου real> := <τιμή ακέραιου τύπου>. Το αντίστροφο δεν γίνεται και εμφανίζεται το μήνυμα λάθους Type mismatch.

παραδείγματα

1. a=2;
b:=3;
sum:=a+b;
Υπολογίζεται η παράσταση του 2ου μέλους 2+3 και η τιμή 5 καταχωρείται στη μεταβλητή sum.

2. a:=2;
b:=a;
Το a παίρνει την τιμή 2 και το b την τιμή του a δηλ. το 2, άρα το b παίρνει την τιμή 2.

3. a:=1; Η μεταβλητή a παίρνει την τιμή 1
b:=2; Η μεταβλητή b παίρνει την τιμή 2
t:=a; Η μεταβλητή t παίρνει την τιμή της a δηλ. 1
a:=b; Η μεταβλητή a παίρνει την τιμή της b δηλ. 2
(άρα χάνει την τιμή 1 που είχε)
b:=t; Η μεταβλητή b παίρνει την τιμή της t δηλ. 1
(άρα χάνει την τιμή 2 που είχε)

4. a:=5;
a:=a+1; Η μεταβλητή a παίρνει την τιμή που είχε δηλ. 5 συν 1
δηλ. η νέα τιμή της a είναι 6.

5. s:='Η ΓΛΩΣΣΑ PASCAL';
Η μεταβλητή s παίρνει τιμή τη σειρά χαρακτήρων
Η ΓΛΩΣΣΑ PASCAL.

2.2. Οι εντολές write και writeln

Με τις εντολές αυτές **τυπώνονται στην οθόνη** τα αποτελέσματα των προγραμμάτων κ.τ.λ..

Μορφές της εντολής write

- **write(< μεταβλητή>);**

Εμφανίζει στην οθόνη την τιμή της μεταβλητής και στη συνέχεια δεν αλλάζει γραμμή εκτύπωσης.

- **write(< μεταβλητή_1>, <μεταβλητή_2>,....);**

Εμφανίζει στην οθόνη τις τιμές των μεταβλητών, όλες στην ίδια γραμμή χωρίς κανένα κενό (εκτός από τις μεταβλητές τύπου real που αν η τιμή τους είναι θετική αφήνεται ένα κενό για το πρόσημο +) και στη συνέχεια δεν αλλάζει γραμμή εκτύπωσης.

- **write('σειρά χαρακτήρων');**

Εμφανίζει τη σειρά χαρακτήρων όπως είναι και στη συνέχεια δεν αλλάζει γραμμή εκτύπωσης.

- **write('σειρά χαρακτήρων',<μεταβλητή>);**

Εμφανίζει τη σειρά χαρακτήρων όπως είναι, δίπλα (στην ίδια γραμμή χωρίς κανένα κενό) την τιμή της μεταβλητής και στη συνέχεια δεν αλλάζει γραμμή εκτύπωσης.

Μορφές της εντολής writeln

- **writeln(< μεταβλητή>);**

Εμφανίζει στην οθόνη την τιμή της μεταβλητής και στη συνέχεια αλλάζει γραμμή εκτύπωσης.

- **writeln(< μεταβλητή_1>, <μεταβλητή_2>,....);**

Εμφανίζει στην οθόνη τις τιμές των μεταβλητών, όλες στην ίδια γραμμή χωρίς κανένα κενό (εκτός από τις μεταβλητές τύπου real που αν η τιμή τους είναι θετική αφήνεται ένα κενό για το πρόσημο +) και στη συνέχεια αλλάζει γραμμή εκτύπωσης.

- **writeln('σειρά χαρακτήρων');**

Εμφανίζει τη σειρά χαρακτήρων όπως είναι και στη συνέχεια αλλάζει γραμμή εκτύπωσης.

- **writeln('σειρά χαρακτήρων',<μεταβλητή>);**

Εμφανίζει τη σειρά χαρακτήρων όπως είναι, δίπλα (στην ίδια γραμμή χωρίς κανένα κενό) την τιμή της μεταβλητής και στη συνέχεια αλλάζει γραμμή εκτύπωσης.

- **writeln;**

Αλλάζει απλώς η γραμμή εκτύπωσης. χρησιμοποιείται συνήθως για να αφήσουμε μια κενή γραμμή.

Αλλαγή της γραμμής εκτύπωσης σημαίνει ότι μία επόμενη write ή writeln θα τυπώσει στην επόμενη γραμμή.

Παραδείγματα

- writeln('PASCAL');
  Τυπώνεται στην οθόνη η λέξη PASCAL

- a:=5;
  writeln(a);
  Τυπώνεται στην οθόνη η τιμή της μεταβλητής a δηλ. 5.

- x:=15;
  y:=10;
  sum:=x+y;
  writeln('το άθροισμα του ',x, ' και του ',y,' είναι ',sum);
  Τυπώνεται στην οθόνη: το άθροισμα του 15 και του 10 είναι 25

Οι αριθμοί τύπου real τυπώνονται σε εκθετική μορφή:
X.XXXXXXXXXEXX όπου X: ψηφία

Διευθέτηση της εκτύπωσης

- **Ακέραιες αριθμητικές μεταβλητές**
  (τύπου shortint, byte, integer, word, longint)

<Ακέραια μεταβλητή>:n όπου n ακέραιος αριθμός.
Τυπώνεται η τιμή της μεταβλητής σε n θέσεις. Αν η τιμή αποτελείται από λιγότερα ψηφία, αφήνονται από αριστερά ανάλογα κενά.
π.χ. i:=12;
writeln(i:5);
Θα εμφανίσει στην οθόνη τον αριθμό 12 αφήνοντας όμως από αριστερά του αριθμού 3 κενά. Αν η τιμή της μεταβλητής έχει περισσότερα ψηφία από τον αριθμό n τυπώνεται ολόκληρη (δηλ. το n αυξάνεται αυτόματα όσο χρειάζεται).

- **Πραγματικές αριθμητικές μεταβλητές**
  (τύπου real)

<πραγματική μεταβλητή>:n:m όπου n και m ακέραιοι αριθμοί.
Η μορφή αυτή χρησιμοποιείται για να μην τυπώνονται οι real σε εκθετική μορφή.
Τυπώνεται η τιμή της μεταβλητής **σε n θέσεις συνολικά** (στις οποίες πρέπει να συμπεριλαμβάνεται το πρόσημο και η υποδιαστολή (.), με m δεκαδικά ψηφία. Αν η τιμή αποτελείται από λιγότερα ψηφία, αφήνονται από αριστερά ανάλογα κενά.
π.χ. a:=12.5;
writeln(i:7:2);

Θα εμφανίσει στην οθόνη τον αριθμό 12.50 αφήνοντας όμως από αριστερά του αριθμού 2 κενά (το 1 είναι του προσήμου που είναι το +). Αν η τιμή της μεταβλητής έχει περισσότερα ακέραια ψηφία από τον αριθμό n τυπώνεται ολόκληρη (δηλ. το n αυξάνεται αυτόματα όσο χρειάζεται). Αν η τιμή της μεταβλητής έχει περισσότερα δεκαδικά ψηφία από τον αριθμό m τότε τα επιπλέον δεν εμφανίζονται.

- **Αλφαριθμητικές μεταβλητές (τύπου string)**

<Αλφαριθμητική μεταβλητή>:n όπου n ακέραιος αριθμός.
τυπώνεται η τιμή της μεταβλητής σε n θέσεις. Αν η τιμή αποτελείται από λιγότερους χαρακτήρες, αφήνονται από αριστερά ανάλογα κενά.
π.χ. s:='pascal';
writeln(s:10);
Θα εμφανίσει στην οθόνη τη λέξη pascal αφήνοντας όμως από αριστερά 4 κενά. Αν η τιμή της μεταβλητής έχει περισσότερους χαρακτήρες από τον αριθμό n τυπώνεται ολόκληρη (δηλ. το n αυξάνεται αυτόματα όσο χρειάζεται).

Παράδειγμα
Δίνονται το όνομα, το επώνυμο, οι ώρες εργασίας και η αμοιβή ανά ώρα ενός υπαλλήλου. Να υπολογισθεί το ημερομίσθιό του και να τυπωθούν τα αποτελέσματα ως εξής:

ΠΕΤΡΟΥ ΙΩΑΝΝΗΣ

Ώρες εργασίας : 6
Αμοιβή ανά ώρα : 20 €
Ημερομίσθιο : 120 €

```
program misthodosia;
var
   onom,epon:string[20];
   wres:byte;
   amoibh, hmerom:word;
begin
      onom:='ΙΩΑΝΝΗΣ';
      epon:=' ΠΕΤΡΟΥ';
      wres:=5;
      amoibh:=3000;
      hmerom:=wres*amoibh;
      writeln(epon,' ',onom);
      writeln;
      writeln('Ώρες εργασίας   : ',wres);
      writeln('Αμοιβή ανά ώρα: ', amoibh);
```

```
    writeln('Ημερομίσθιο      : ',hmerom);
    readln;
end.
```

2.3. Η εντολή readln

readln(<όνομα μεταβλητής>;

Η εντολή reading χρησιμοποιείται για να δώσουμε τιμή σε μια μεταβλητή την ώρα της εκτέλεσης του προγράμματος. Μεταβλητές τύπου boolean δεν μπορούν να πάρουν τιμή με την εντολή αυτή.
Συνήθως πριν από την εντολή readln χρησιμοποιούμε την εντολή write έτσι ώστε να εμφανισθεί στην οθόνη ένα σχόλιο σχετικό με τη μεταβλητή που πρόκειται να πάρει τιμή από τη readln που ακολουθεί.

Παράδειγμα

```
program greetings;
var
   onom:string[20];
begin
    write('Πως σε λένε; ');
    readln(onom);
    writeln;
    writeln('Γεια σου, ',onom);
    readln;
end.
```

(* Εμφανίζεται το σχόλιο 'Πως σε λένε;' και δεν αλλάζει η γραμμή εκτύπωσης, έτσι ώστε το όνομα να δοθεί δίπλα στην ερώτηση *)

Ασκήσεις

1. Δίνονται δύο αριθμοί. Να υπολογισθεί το άθροισμα και το γινόμενό τους.

2. Δίνονται τρεις αριθμοί. Να υπολογισθεί ο μέσος όρος τους.

3. Δίνονται οι κάθετες πλευρές ορθογωνίου τριγώνου. Να υπολογισθεί η υποτείνουσά του.

4. Δίνονται οι πλευρές και το ύψος ενός τραπεζίου. Να υπολογισθεί το εμβαδό του.

5. Να γραφεί πρόγραμμα που θα διαβάζει την τιμή ενός προϊόντος και τα χρήματα που δίνει ο πελάτης στον ταμία και θα εμφανίζει τα παρακάτω:
ΤΙΜΗ : XXX.XX €
ΕΛΑΒΑ : XXX.XX €
ΡΕΣΤΑ : XXX.XX €

6. Να γραφεί πρόγραμμα μετατροπής των βαθμών fahreneit σε βαθμούς Κελσίου. Βαθμοί Κελσίου=5*(Βαθμοί fahreneit-32)/9

7. Δίνεται ένα ποσό σε €. Να μετατραπεί σε δολλάρια και γερμανικά μάρκα. Οι αντιστοιχίες των νομισμάτων να δηλωθούν ως σταθερές.

8. Δίνεται διψήφιος ακέραιος αριθμός. Να βρεθεί ο ακέραιος που προκύπτει από την αντιστροφή των ψηφίων του.

9. Δίνεται η περιγραφή, η τιμή χωρίς Φ.Π.Α. και ο συντελεστής Φ.Π.Α. ενός προϊόντος. Να υπολογισθεί η τελική τιμή του προϊόντος.

10. Δίνονται οι ημέρες που δουλεύει ένας εργάτης το μήνα και το ημερομίσθιό του (αμοιβή την ημέρα). Αν οι κρατήσεις του είναι το 20% των μηνιαίων αποδοχών του, να υπολογισθούν ο μικτός μηνιαίος μισθός του (πριν γίνουν οι κρατήσεις), οι κρατήσεις και ο καθαρός μηνιαίος μισθός του.

11. Μια εταιρεία φορολογείται με συντελεστή 30% επί των κερδών της. Αν δίνονται τα κέρδη της, να υπολογισθεί ο φόρος που θα πληρώσει και το ποσό που τελικά θα της μείνει.

12. Ένας πωλητής παίρνει ποσοστό 30% επί των πωλήσεων. Να δοθεί το ονοματεπώνυμό του και το ποσό 3 πωλήσεών του και να υπολογισθούν το ποσοστό του και το ποσό που θα πάρει η εταιρεία.

13. Δίνονται το κεφάλαιο που καταθέτει κάποιος στην τράπεζα για δύο χρόνια και το επιτόκιο. Να υπολογισθεί ο τόκος στο τέλος του 1ου χρόνου, ο τόκος στο τέλος του 2ου χρόνου και πόσο τελικά γίνεται το κεφάλαιο.

14. Με δεδομένο ότι οι υπερωρίες πληρώνονται 50% **επιπλέον**, να υπολογισθεί το ημερομίσθιο ενός υπαλλήλου, αν δίνονται οι ώρες εργασίας, η αμοιβή ανά ώρα και οι υπερωρίες.

15. το ημερομίσθιο ενός εργάτη αυξάνεται κατά 5% για κάθε παιδί που έχει. πληρώνει 20% ΙΚΑ και 10% φόρο. Αν δίνονται το ημερομίσθιο και οι μέρες εργασίας ανά μήνα, να υπολογισθούν οι μικτές μηνιαίες αποδοχές του, το ΙΚΑ, ο φόρος και οι καθαρές μηνιαίες αποδοχές του. (Ο φόρος υπολογίζεται αφού αφαιρεθεί το ΙΚΑ).

Μεταβλητές τύπου string

Μία μεταβλητή τύπου string όπως έχουμε μάθει, μπορεί να δεχθεί μέχρι το πολύ 255 χαρακτήρες και καταλαμβάνει στη μνήμη χώρο 256 bytes. Μπορεί όμως εμείς να της δώσουμε μία τιμή που να αποτελείται από λιγότερους χαρακτήρες.
Η συνάρτηση **length(<όνομα μεταβλητής string>)** δίνει το πλήθος των χαρακτήρων που πληκτρολογήσαμε. π.χ. αν στη μεταβλητή name δώσουμε την τιμή ΚΩΝΣΤΑΝΤΙΝΟΣ, η συνάρτηση length(name) θα πάρει την τιμή 12.

Το όνομα της μεταβλητής ακολουθούμενο από [<αριθμός>], (όπου αριθμός από 1 έως το μέγιστο μήκος του string), δίνει τον χαρακτήρα του string που βρίσκεται στην <αριθμός> θέση.

Παράδειγμα

```
name:='ΚΩΝΣΤΑΝΤΙΝΟΣ';
writeln(name[2]);
```
Θα εμφανίσει στην οθόνη το γράμμα Ω, που είναι το 2ο του name.
```
writeln(length(name));
```
Θα εμφανίσει τον αριθμό 12.

Για να εκτελεσθούν απ' την Pascal οι δύο επόμενες εντολές clrscr και gotoxy, πρέπει κάτω από τη γραμμή Program ... του προγράμματός μας, να γράψουμε την εντολή uses crt; Η crt είναι μία μονάδα (unit) της Pascal που περιέχει χρήσιμες εντολές της γλώσσας.

2.4. Η εντολή clrscr

clrscr

Με την εντολή clrscr (clear screen) "καθαρίζουμε" την οθόνη.

2.5. Η εντολή gotoxy

gotoxy(<αριθμός στήλης>,<αριθμός γραμμής>);

Με την εντολή gotoxy μπορούμε να τοποθετήσουμε το δρομέα σε συγκεκριμένη θέση της οθόνης. π.χ. η gotoxy(30,10) τοποθετεί το δρομέα στη στήλη 30 και στη γραμμή 10.
Προσέξτε, πρώτα γράφουμε τη στήλη και μετά τη γραμμή.
τιμές στήλης: 1-80, τιμές γραμμής: 1-24.

2.6. Η εντολή if

1η μορφή

```
if <συνθήκη> then <εντολή>;
```

Αν ισχύει η συνθήκη τότε θα εκτελεσθεί η εντολή.

2η μορφή

```
if <συνθήκη> then <εντολή_1> else <εντολή_2>;
```

Αν ισχύει η συνθήκη τότε θα εκτελεσθεί η εντολή_1.
Αν δεν ισχύει η συνθήκη τότε θα εκτελεσθεί η εντολή_2.

3η μορφή

```
if <συνθήκη> then
begin
      <εντολή>;
      <εντολή>;
        ........
end;
```

Αν ισχύει η συνθήκη τότε θα εκτελεσθούν οι εντολές που βρίσκονται μεταξύ του begin και του end.

4η μορφή

```
if <συνθήκη> then
begin
      <εντολή>;
      <εντολή>;
        ........
end   (* προσοχή, αυτό το end δεν έχει ελληνικό ερωτηματικό *)
else
begin
      <εντολή>;
      <εντολή>;
        ........
end;
```

Αν ισχύει η συνθήκη τότε θα εκτελεσθούν οι εντολές που βρίσκονται μεταξύ του 1ου begin .. end.
Αν δεν ισχύει η συνθήκη τότε θα εκτελεσθούν οι εντολές που βρίσκονται μεταξύ του 2ου begin .. end.
Στη συνθήκη της if μπορεί να χρησιμοποιηθούν οι λογικοί τελεστές **and, or, not και xor.**
π.χ. if (x=0) or (x=1) then writeln('Σωστή τιμή');

Ασκήσεις

16. Δίνονται 3 θετικοί αριθμοί. Να υπολογισθούν τα τετράγωνα και οι τετραγωνικές τους ρίζες. Η εμφάνιση των αποτελεσμάτων να γίνει σε μορφή πίνακα (χρήση **gotoxy**)

17. Να γραφεί πρόγραμμα το οποίο θα εμφανίζει σε μορφή πίνακα τα ημίτονα, τα συνημίτονα και τις εφαπτομένες των γνωστών γωνιών 0^{o},30^{o},45^{o},60^{o},90^{o}. (χρήση **gotoxy**)

18. Να γραφεί πρόγραμμα που θα υπολογίζει το μέγιστο 2 ακέραιων αριθμών και θα τον τοποθετεί στη μεταβλητή max.

19. Να γραφεί πρόγραμμα που θα υπολογίζει το μέγιστο 3 ακέραιων αριθμών και θα τον τοποθετεί στη μεταβλητή max.

20. Να γραφεί πρόγραμμα το οποίο θα βρίσκει πόσοι από τους 5 αριθμούς που δίνονται είναι θετικοί.

21. Να γραφεί πρόγραμμα που θα δέχεται έναν ακέραιο θετικό μονοψήφιο αριθμό και θα τον τυπώνει. Αν δε δοθεί θετικός μονοψήφιος το πρόγραμμα θα τυπώνει "Λάθος αριθμός"

22. Να γραφεί πρόγραμμα που θα δέχεται έναν ακέραιο αριθμό.. Αν είναι μεγαλύτερος ή ίσος του 0 θα τυπώνεται η ένδειξη "ΘΕΤΙΚΟΣ ή ΜΗΔΕΝ" και θα υπολογίζεται η τετραγωνική του ρίζα, ενώ αν είναι αρνητικός θα τυπώνεται η ένδειξη "ΑΡΝΗΤΙΚΟΣ" και θα υπολογίζεται το τετράγωνό του.

23. Ενα προϊόν πωλείται ως εξής: α) για λιγότερα από 100 τεμάχια προς 150€ το ένα β) για 100 και περισσότερα τεμάχια προς 100 € το ένα. Να δοθεί ο αριθμός των τεμαχίων και να υπολογισθούν τα χρήματα που εισπράχθηκαν.

24. Δίνεται ακέραιος αριθμός. Να τυπωθεί ολογράφως το υπόλοιπο της διαίρεσής του με το 3.

25. Δίνεται η συνάρτηση $y=\begin{cases} x+1 & \text{αν } x<0 \\ x^2-1 & \text{αν } 0<=x<=1 \\ x+1 & \text{αν } x>1 \end{cases}$

Να δοθεί μια τιμή του x και να υπολογισθεί το αντίστοιχο y.

26. Δίνονται 2 θετικοί αριθμοί. Να υπολογισθεί η διαφορά τους έτσι ώστε να είναι κι αυτή θετική.

27. Μια εταιρεία δίνει επίδομα στους υπαλλήλους της με βάση τον αριθμό παιδιών που έχουν. Για 1 παιδί 5%, για 2 παιδιά 10%, για 3 παιδιά 15% και για περισσότερα από 3 20%. Να δοθεί ο μισθός ενός υπαλλήλου και ο αριθμός παιδιών του και να υπολογισθεί το επίδομά του.

28. Μια εταιρεία δίνει επίδομα στους υπαλλήλους της με βάση τις γραμματικές τους γνώσεις ως εξής:

1. 2% για αποφοιτους Γυμνασίου
2. 5% για αποφοιτους Λυκείου
3. 10% για πτυχιούχους ΤΕΙ
4. 15% για πτυχιούχους ΑΕΙ

Να δοθεί ο μισθός ενός υπαλλήλου και ο κωδικός γνώσεων (1,2,3,4) και να υπολογισθεί το επίδομα γνώσεων.

29. Να υπολογισθεί ο φόρος εισοδήματος ενός φορολογούμενου με βάση τα παρακάτω:
Για εισόδημα μέχρι 20000 € φόρος 10%, από 20001 μέχρι 40000 φόρος 20% και από 40001 και πάνω φόρος 25%.

30. Ένας πωλητής παίρνει ποσοστό 10% επί των πωλήσεων και δικαιούται bonus 50 € αν οι πωλήσεις του υπερβαίνουν τα 1000 €. Να δοθεί το ποσό των πωλήσεων και να υπολογισθούν τα χρήματα που θα πάρει ο πωλητής.

31. Κάποιος αγοράζει μια τηλεόραση με ισόποσες δόσεις. Να δοθεί η αρχική αξία της τηλεόρασης και ο αριθμός των δόσεων και να βρεθεί το ποσό που θα πληρώνει στην κάθε δόση με βάση τα παρακάτω:
1 δόση συνολική επιβάρυνση 0%
2 δόσεις συνολική επιβάρυνση 10%
3 ή 4 δόσεις συνολική επιβάρυνση 20%
5 ή 6 δόσεις συνολική επιβάρυνση 30%

32. Η κεντρική θέρμανση ενός εργοστασίου ξεκινά αν η θερμοκρασία σε 3 διαφορετικά σημεία είναι μικρότερη των 15^{o} Κελσίου. Να δοθούν οι 3 θερμοκρασίες και να τυπωθεί μήνυμα ΟΝ ή OFF ανάλογα αν πρέπει να λειτουργήσει η θέρμανση ή όχι.

33. Δίνεται ακέραιος. Να τυπώνεται η λέξη ΖΥΓΟΣ αν είναι ζυγός ή η λέξη ΜΟΝΟΣ αν είναι μονός.

34. Ένας μαθητής παίρνει τρεις προφορικούς βαθμούς και ένα γραπτό. Αν ο γραπτός έχει διπλάσια βαρύτητα, να υπολογισθεί ο μέσος όρος βαθμολογίας του, στη συνέχεια να στρογγυλευθεί και να τυπωθεί η ένδειξη ΠΕΡΑΣΕ (αν ο μέσος όρος είναι μεγαλύτερος ή ίσος του 10) ή ΑΠΕΤΥΧΕ.

35. Τα αυτοκίνητα που νοικιάζει ένα γραφείο χρεώνονται με 2 € το ΚΜ για τα πρώτα 100 ΚΜ και με 3 € το ΚΜ για τα επιπλέον ΚΜ. Στο ποσό αυτό προστίθεται πάγιο 50 €. Να δοθούν τα ΚΜ που διένυσε κάποιος και να υπολογισθεί η συνολική του χρέωση.

36. Να δοθούν οι συντελεστές μιας δευτεροβάθμιας εξίσωσης και να υπολογισθούν οι ρίζες της (αν υπάρχουν).

2.7. Η εντολή case

```
case <μεταβλητή> of
  <τιμή 1>: <εντολή 1>;
  <τιμή 2>: <εντολή 2>;
  <τιμή 3>: <εντολή 3>;
   .....
end;
```

Αν η τιμή της μεταβλητής είναι ίση με την <τιμή 1> τότε εκτελείται η <εντολή 1>

Αν η τιμή της μεταβλητής είναι ίση με την <τιμή 2> τότε εκτελείται η <εντολή 2>,

Αν η τιμή της μεταβλητής είναι ίση με την <τιμή 3> τότε εκτελείται η εντολή 3 κ.ο.κ.

Παρατηρήσεις

- Η μεταβλητή δεν μπορεί να είναι πραγματικού τύπου (real) ούτε τύπου string.
- Οι τιμές <τιμή 1>, <τιμή 2> κ.τ.λ. πρέπει να συμφωνούν ως προς τον τύπο με τη μεταβλητή. Αν είναι αριθμητικές δεν πρέπει να είναι έξω από τα όρια του integer (-32768 .. 32767), παρ' όλο που η μεταβλητή μπορεί να είναι και τύπου longint.
- Η μεταβλητή της case δεν πρέπει να αλλάζει τιμή μέσα στην case.

```
case <μεταβλητή> of
  <τιμή 1>: begin
              <εντολή 1-1>;
              <εντολή 1-2>;
              <εντολή 1-3>;
            end;
  <τιμή 2>: begin
              <εντολή 2-1>;
              <εντολή 2-2>;
              <εντολή 2-3>;
            end;
   .....
end;
```

Στην περίπτωση που θέλουμε να εκτελούνται περισσότερες από μία εντολές για κάποια τιμή της μεταβλητής μέσα στην case τις κλείνουμε σε begin .. end;
Αν πρέπει να εκτελείται η ίδια εντολή (ή οι ίδιες εντολές) για περισσότερες από μία τιμές, μπορούμε να τις γράψουμε μαζί, χωρίζοντάς τες με κόμμα.

```
case <μεταβλητή> of
  <τιμή 1>,<τιμή 2>: <εντολή>;
  .....
end;
```

Παράδειγμα

Να δίνεται ένας μήνας αριθμητικά και να τυπώνεται στην οθόνη η εποχή που ανήκει. π.χ. αν δοθεί 4 (δηλ. Απρίλιος) να εμφανισθεί στην οθόνη ΑΝΟΙΞΗ.

```
program epoxes;
var  month:byte;
begin
    write('Μήνας αριθμητικά: '); readln(month);
    case month of
       1,2,12: writeln('ΧΕΙΜΩΝΑΣ');
       3,4,5: writeln('ΑΝΟΙΞΗ');
       6,7,8: writeln('ΚΑΛΟΚΑΙΡΙ');
       9,10,11: writeln('ΦΘΙΝΟπΩΡΟ');
    end;
end.
```

Ασκήσεις

37. Να γραφεί πρόγραμμα στο οποίο θα δίνεται ένας μήνας αριθμητικά και θα τυπώνεται ολογράφως π.χ. αν δοθεί το 3 θα τυπώνεται ΜΑΡΤΙΟΣ.

38. Να γραφεί πρόγραμμα στο οποίο θα δίνονται 2 αριθμοί και θα εμφανίζεται το μενού:

1. πρόσθεση 2. Αφαίρεση
3. πολλαπλασιασμός 4. Διαίρεση

Ποια η επιλογή σου;

Ανάλογα με την επιλογή του χρήστη θα γίνεται η ανάλογη πράξη.

39. Στις εξετάσεις αν κάποιος μαθητής συγκέντρωσε:
91..100 βαθμούς θα παίρνει Α, 80..90 βαθμούς θα παίρνει Β,
70..79 βαθμούς θα παίρνει C, 60..69 βαθμούς θα παίρνει D,
0..59 βαθμούς θα παίρνει Ε. Να δοθεί ο βαθμός ενός μαθητής και να τυπωθεί ο ανάλογος χαρακτηρισμός.

40. Να γραφεί πρόγραμμα στο οποίο θα εμφανίζεται ένα μενού με τις επιλογές:

τ. Εμβαδό τριγώνου
π. Εμβαδό παραλληλογράμμου
Κ. Εμβαδό κύκλου

Να δίνονται τα απαραίτητα στοιχεία και να υπολογίζεται το εμβαδό.

41. Να γραφεί πρόγραμμα στο οποίο θα δίνεται αριθμητικά ένας μήνας και ένα έτος και θα εμφανίζεται ο αριθμός των ημερών που έχει ο μήνας αυτός.

42. Να γραφεί πρόγραμμα το οποίο θα υπολογίζει το φόρο εισοδήματος ως εξής: Για εισόδημα μέχρι 10000 φόρος 0%, από 10001 μέχρι 20000 φόρος 10%,από 20001 μέχρι 30000 φόρος 15%,από 30001 μέχρι 40000 φόρος 20% και από 40001 και πάνω φόρος 25%.

2.8. Η εντολή goto

goto <ετικέττα>;

Αποφεύγουμε, όσο είναι δυνατό, τη χρήση της εντολής αυτής.
Με την εντολή goto μπορούμε ν' αλλάξουμε τη ροή του προγράμματος δηλ. η εκτέλεση συνεχίζεται με την εντολή στην οποία μας πηγαίνει η goto και όχι με την επόμενη (της goto) εντολή.
Στην εντολή που θέλουμε να μεταφερθούμε, δίνουμε μια ετικέτα δηλ. **έναν ακέραιο αριθμό** μεταξύ του 0 και του 9999 (στην Turbo Pascal επιτρέπονται και χαρακτήρες) **ακολουθούμενο από :**
Οι ετικέτες πρέπει να δηλώνονται στην αρχή του προγράμματος με τη δήλωση **label** (συνήθως πριν από τη var).

Παράδειγμα
Να δίνονται συνεχώς απ' το πληκτρολόγιο αριθμοί, οι οποίοι να αθροίζονται μέχρι να δοθεί ο αριθμός 0.

```
program athroisma;
label 1;
var
   number,total:real;
begin
    total:=0;
1: write('Αριθμός:');  readln(number);
    total:=total+number;
    if number<>0 then goto 1;
    writeln('Αθροισμα=',total:6:1);
    readln;
end.
```

Ασκηση : τί παρατηρείτε στο παρακάτω πρόγραμμα;

```
program athroisma;
label 10;
var
   a,b,s:integer;
begin
      s:=0;
10:  readln(a);
      s:=s+a;
      goto 10;
      writeln('Αθροισμα=',s);
end.
```

2.9. Η εντολή repeat ... until

```
repeat
     <εντολή 1>;
     <εντολή 2>;
      ..........
until <συνθήκη>;
```

Με την εντολή repeat, οι εντολές (μεταξύ του repeat και του until) επαναλαμβάνονται μέχρι η συνθήκη του until να γίνει αληθής (να ισχύσει).

Παρατήρηση

- Οι εντολές εκτελούνται τουλάχιστον μία φορά, γιατί ο έλεγχος της συνθήκης γίνεται στο τέλος. Έτσι ακόμη και αν η συνθήκη ισχύει, οι εντολές θα εκτελεσθούν μια φορά και μετά θα συνεχισθεί η εκτέλεση με την εντολή που βρίσκεται κάτω απ' την until.

Προσοχή
Οι εντολές εκτελούνται όσο η συνθήκη είναι ψευδής (δεν ισχύει).

2.10. Η εντολή while

```
while <συνθήκη> do
begin
     <εντολή 1>;
     <εντολή 2>;
      ..........
end;
```

Με την εντολή while, οι εντολές (μεταξύ του begin και του end) επαναλαμβάνονται όσο η συνθήκη του while είναι αληθής.

Παρατηρήσεις

- Αν η συνθήκη δεν ισχύει απ' την αρχή, οι εντολές δεν θα εκτελεσθούν καμιά φορά και θα συνεχισθεί η εκτέλεση με την εντολή που βρίσκεται κάτω απ 'το end.
- Για μία μόνο εντολή, δεν χρειάζεται begin και end.

Προσοχή
Οι εντολές εκτελούνται όσο η συνθήκη είναι αληθής (ισχύει).

Ασκήσεις

43. Δίνονται Ν αριθμοί. Να βρεθεί ο μέσος όρος τους.

44. Να διαβαστούν 2 αριθμοί και να τυπωθεί το άθροισμά τους. Η διαδικασία να επαναλαμβάνεται μέχρι να δοθεί
α) σε έναν απ' τους δύο η τιμή 0
β) και στους δύο η τιμή 0

45. Να διαβαστούν 10 αριθμοί στην ίδια θέση μνήμης α και να υπολογισθεί το γινόμενό τους.

46. Δίνονται για Ν άτομα: Κωδικός φύλου (Γ: γυναίκα Α: άνδρας), βάρος και ηλικία. Να βρεθούν: α) Ο μέσος όρος ηλικίας των γυναικών με βάρος μεγαλύτερο των 60 κιλών β) πόσοι άνδρες έχουν βάρος μικρότερο των 85 κιλών και ηλικία μικρότερη των 60 χρόνων γ) πόσο ετών είναι ο βαρύτερος άνδρας.

47. Δίνονται Ν αριθμοί. Να βρεθεί πόσοι είναι θετικοί, πόσοι αρνητικοί και πόσοι μηδέν.

48. Ο πληθυσμός μιας χώρας αυξάνεται κάθε χρόνο με ποσοστό 10%. Αν σήμερα είναι 10000000, σε πόσα χρόνια θα ξεπεράσει τα 20000000.

49. Να βρεθεί ο μεγαλύτερος και ο μικρότερος από Ν αριθμούς.

50. Για Ν μαθητές δίνονται Μ βαθμοί (για τον καθένα). Να υπολογισθεί ο μέσος όρος βαθμολογίας του καθένα. (Ολα τα στοιχεία ενός μαθητή να εμφανίζονται σε μία οθόνη)

51. Για Ν μαθητές δίνονται τα στοιχεία: Ονοματεπώνυμο και αριθμός απουσιών. Να βρεθούν: α) Το ονοματεπώνυμο του μαθητή με τις περισσότερες απουσίες
β) Το πλήθος των μαθητών που έχουν από 0 έως 10 απουσίες.
Το πλήθος των μαθητών που έχουν από 11 έως 20 απουσίες
Το πλήθος των μαθητών που έχουν από 21 έως 30 απουσίες
Το πλήθος των μαθητών που έχουν από 31 έως 40 απουσίες
Το πλήθος των μαθητών που έχουν από 41 έως 50 απουσίες
γ) Ο μέσος όρος απουσιών

52. Δίνονται Ν αριθμοί μεταξύ του 1 και του 6. Να βρεθεί το πλήθος των άσσων, των 2ριών, των 3ριών, των 4ριων, των 5ριών και των 6ριών.

53. Δίνεται το ενοίκιο που πληρώνει κάποιος σήμερα, το οποίο αυξάνεται κάθε 2 χρόνια κατά 15%. Να υπολογισθεί σε πόσα χρόνια θα διπλασιασθεί.

2.11. Η εντολή for

1η μορφή

```
for <μεταβλητή>:=<αρχική τιμή> to <τελική τιμή> do
begin
    <εντολή 1>;
    <εντολή 2>;
     .....
end;
```

Η μεταβλητή αρχικά παίρνει την <αρχική τιμή>, εκτελούνται οι εντολές μεταξύ του begin και του end, στη συνέχεια η μεταβλητή **αυξάνεται κατά 1**, εκτελούνται πάλι οι εντολές μεταξύ του begin και του end κ.ο.κ. μέχρι που η μεταβλητή παίρνει την <τελική τιμή> και εκτελούνται για τελευταία φορά οι εντολές μεταξύ του begin και του end.

Παρατηρήσεις

- Η μεταβλητή δεν πρέπει να είναι τύπου real (ή τύπου string)
- Η τιμή της μεταβλητής της for μπορεί να χρησιμοποιηθεί στις εντολές μέσα στη for, αλλά καλό είναι να μην αλλάζει μέσα στη for.
- Κατά την έξοδο από το βρόγχο της for, η μεταβλητή έχει την τελική τιμή.
- Αν η τελική τιμή είναι μικρότερη της αρχικής, οι εντολές δεν θα εκτελεσθούν καμιά φορά και θα συνεχισθεί η εκτέλεση με την εντολή που βρίσκεται κάτω απ' το end.
- Για μία μόνο εντολή, δεν χρειάζεται begin και end.
- Μπορούμε, με χρήση της εντολής goto, να βγούμε έξω από μία for, δεν μπορούμε όμως να μπούμε σε μία for από άλλο σημείο παρά μόνο από την αρχή της.

2η μορφή

```
for <μεταβλητή>:=<αρχική τιμή> downto <τελική τιμή> do
begin
    <εντολή 1>;
    <εντολή 2>;
     .....
end;
```

Η μεταβλητή αρχικά παίρνει την <αρχική τιμή>, εκτελούνται οι εντολές μεταξύ του begin και του end, στη συνέχεια η μεταβλητή **μειώνεται κατά 1**, εκτελούνται πάλι οι εντολές μεταξύ του begin και του end κ.ο.κ. μέχρι που η μεταβλητή παίρνει την <τελική τιμή> και εκτελούνται για τελευταία φορά οι

εντολές μεταξύ του begin και του end. Η αρχική τιμή πρέπει να είναι μεγαλύτερη της τελικής.

for μέσα σε for

Μία for μπορεί να βρίσκεται εξ ολοκλήρου μέσα σε μία άλλη for. Σ' αυτή την περίπτωση, **για κάθε τιμή** της μεταβλητής **της εξωτερικής** for, **η εσωτερική** for κάνει **όλες τις επαναλήψεις** της.

```
for <μεταβλητή 1>:=<αρχική τιμή 1> to <τελική τιμή 1> do
begin
    for <μεταβλητή 2>:=<αρχική τιμή 2> to <τελική τιμή 2> do
   begin
     <εντολή 1>;
     <εντολή 2>;
      ..........
   end;
   ..........
end;
```

Παράδειγμα

```
program ginomeno;
var
  i,j,p:integer;
begin
    for i:=1 to 3 do        (* Η εξωτερική for δεν εχει begin και end *)
      for j:=1 to 2 do      (* γιατί περιέχει μία μόνο εντολή, την *)
      begin                  (*  εσωτερική for *)
         p:=i*j;
          writeln('Γινόμενο: ',p);
      end;
```

Για i:=1
    j:=1 τυπώνει: Γινόμενο: 1
    j:=2 τυπώνει: Γινόμενο: 2
Για i:=2
    j:=1 τυπώνει: Γινόμενο: 2
    j:=2 τυπώνει: Γινόμενο: 4
Για i:=3
    j:=1 τυπώνει: Γινόμενο: 3
    j:=2 τυπώνει: Γινόμενο: 6

Ασκήσεις

54. Να υπολογισθούν:
α)1*2+2*3+...+(N-1)*N
β) 1+2+3+...+N
γ)1*2*3*...*N

55. Να εκτυπωθούν όλοι οι ζυγοί από το 2 έως το Ν (αν το Ν δεν είναι ζυγός να ξαναδίνεται)

56α. Για Ν είδη δίνονται τα εξής στοιχεία: Κωδικός προέλευσης (1. Γερμανία 2. Αγγλία), ποσότητα και τιμή μονάδος. Να βρεθεί η συνολική αξία των ειδών με προέλευση την Αγγλία και η συνολική ποσότητα που εισάγεται από τη Γερμανία.

56β. Για Ν είδη δίνονται τα εξής στοιχεία: Κωδικός, ποσότητα, τιμή μονάδος και όριο ασφάλειας. Να βρεθεί η συνολική αξία των ειδών και να τυπωθούν οι κωδικοί των ειδών που η ποσότητα τους είναι κάτω από το όριο ασφάλειας.

57. Για Ν αυτοκίνητα δίνονται ο αριθμός κυκλοφορίας και ο αριθμός ατυχημάτων. Να υπολογισθεί ο μέσος όρος ατυχημάτων και να βρεθεί ο αριθμός κυκλοφορίας του αυτοκινήτου με τα περισσότερα ατυχήματα

58. Δίνονται το ονοματεπώνυμο και οι επιδόσεις σε 3 προσπάθειες (στο άλμα σε μήκος) Ν αθλητών. Να βρεθεί ο νικητής του αγωνίσματος και η επίδοσή του (η καλύτερη από τις 3).

59. Για Ν μαθητές δίνονται : Ονοματεπώνυμο, κωδικός φύλου (1 αγόρι 2 κορίτσι) και βαθμός. Να βρεθεί ποιο αγόρι έχει το μικρότερο βαθμό.

60. Δίνονται οι βαθμοί σε 5 μαθήματα Ν μαθητών. Να βρεθεί ο μέσος όρος του κάθε μαθητή.

61. Να υπολογισθεί το άθροισμα: 1!+2!+3!+...+N!

Έλεγχος υπέρβασης τιμών τύπου

Έστω ότι έχουμε δηλώσει μία μεταβλητή i τύπου byte. Αν στη συνέχεια, στο πρόγραμμά μας, γράψουμε την εντολή i:=1000; (το 1000 είναι έξω από τα όρια του byte), ή την εντολή for i:=1 to 1000 do ... τότε εμφανίζεται το μήνυμα Constant out of range. Αν όμως δώσουμε την εντολή readln(i) και ο χρήστης δώσει στη μεταβλητή i κατά την εκτέλεση του προγράμματος την τιμή 1000 δεν θα εμφανισθεί μήνυμα λάθους, και η μεταβλητή i θα πάρει για τιμή "σκουπίδια". Για να γίνεται έλεγχος τιμών, θα πρέπει να ενεργοποιήσουμε τον έλεγχο λαθών. Αυτό γίνεται προσθέτοντας στο πρόγραμμά μας το διακόπτη {$R+} τον οποίο μπορούμε να απενεργοποιήσουμε αργότερα με {$R-}.

Παράδειγμα

```
program check_number;
var
  i:byte;
begin
       write('Αριθμός: ');
{$R+}                    (*Αν κατά την εκτέλεση του προγράμματος*)
      readln(i);         (* δώσουμε την i π.χ. την τιμή 1000, θα *)
      .....              (* εμφανισθεί το μήνυμα λάθους *)
{$R-}                    (*    Range check error    *)
end.
```

Δηλώσεις νέων τύπων

Στην Pascal, μπορούμε εκτός από τους γνωστούς τύπους (integer, real κ.τ.λ.) να ορίσουμε και νέους τύπους. Αυτό γίνεται με τη δήλωση type που γράφεται πριν από τις δηλώσεις var.
Υπάρχουν τρεις τρόποι για να δηλώσουμε ένα νέο τύπο

1. <όνομα τύπου>=<υπάρχον τύπος>;

Παράδειγμα

```
Type
   akeraios=integer;
var
    i:akeraios;
```

Με την παραπάνω δήλωση το akeraios είναι ο ίδιος τύπος με τον integer.

2. <όνομα τύπου>=<τιμή 1>..<τιμή 2>;
τύπος υποπεριοχής τιμών.

Οι τιμές 1 και 2 δεν μπορεί να είναι πραγματικές ή string. Μπορεί να είναι οποιοιδήποτε ακέραιοι (και longint).
Η <τιμή 1>..<τιμή 2> ονομάζεται υποπεριοχή τιμών.

Παράδειγμα

```
Type
   bathmos=0..20;
var
   b:bathmos;
```

Η μεταβλητή b θα παίρνει τιμές από 0 έως 20.
Για να γίνεται έλεγχος υπέρβασης τιμών, χρησιμοποιούμε το διακόπτη {$R+} όπως τον περιγράψαμε παραπάνω.

Παραδείγματα

```
type
  m=0..100;
  bit=0..1;
  letter='A'..'Z';     (* Υποπεριοχή char *)
```

Μπορούμε να γράψουμε την υποπεριοχή τιμών απευθείας στη δήλωση var. Σ' αυτή την περίπτωση ο τύπος ονομάζεται ανώνυμος.
π.χ.

```
    var bathmos:0..20;
```

3. <όνομα τύπου>=(<λέξη 1>,<λέξη 2>, ...);
Απαριθμητός τύπος.

- Οι λέξεις 1, 2 κ.τ.λ. δεν μπορεί να περιέχουν ελληνικούς χαρακτήρες.
- Δεν γίνεται διάκριση μεταξύ κεφαλαίων και πεζών.
- Η ίδια λέξη δεν μπορεί να περιέχεται σε δύο διαφορετικούς απαριθμητους τύπους.
- Οι μεταβλητές που θα δηλωθούν απαριθμητού τύπου, δεν μπορούν να πάρουν τιμή με την εντολή readln ή να εκτυπωθούν με την writeln.

Παράδειγμα

```
Program kok;
uses crt;
Type
   colors=(red, orange, green);
   ar=1..3;
var
    c:ar;
    xroma:colors;

begin
     clrscr;
     writeln(1. Κόκκινο');
     writeln(2. πορτοκαλί');
     writeln(3. πράσινο');
     write('Επιλογή χρώματος: '); readln(c);
     case c of
        1:  xroma:=red;
        2:  xroma=orange;
        3:  xroma= green;
      end;
     if xroma=red then
         writeln ('το αυτοκίνητο πρέπει να σταματήσει');
     if xroma=green then
         writeln('το αυτοκίνητο μπορεί να περάσει');
end.
```

3. Υποπρογράμματα

Στην Pascal υπάρχουν δύο ειδών υποπρογράμματα:
οι functions (συναρτήσεις) και οι procedures (διαδικασίες).

3.1. Συναρτήσεις (functions)

```
function <όνομα συνάρτησης>(παράμετρος1:τύπος1;
                            παράμετρος2:τύπος2;...):τύπος της function;
Δηλώσεις (var κ.τ.λ. )
                (* Οι μεταβλητές που θα δηλωθούν εδώ *)
                (* ονομάζονται τοπικές μεταβλητές.  *)
begin
    <εντολές>;
     .....
     <όνομα function>:= <τιμή>;
      (*Η τιμή πρέπει να είναι ίδιου τύπου με τον τύπο της function*)
end;
```

Οι συναρτήσεις γράφονται πριν από το κυρίως πρόγραμμα και πριν απ'το σημείο που θα κληθούν.

Οι παράμετροι 1,2 κ.τ.λ. ονομάζονται **τυπικές παράμετροι** και παίρνουν τιμές από αντίστοιχες μεταβλητές κατά την κλήση της συνάρτησης, οι οποίες ονομάζονται **πραγματικές παράμετροι.**

Τα ονόματα των τυπικών παραμέτρων δεν είναι υποχρεωτικό να είναι ίδια με τα ονόματα των πραγματικών παραμέτρων.

Δεν μπορεί οι τυπικές παράμετροι ή ο τύπος της function να είναι τύπου υποπεριοχής τιμών (π.χ. 0..20). Στην περίπτωση αυτή, δηλώνουμε στο type έναν τύπο (π.χ. bathmos=0..20) και οι τυπικές παράμετροι στην παρένθεση της function ή ο τύπος της δηλώνονται αυτού του τύπου π.χ. bathmos. Επίσης, δεν μπορεί να είναι τύπου string[n]. Όπως και στην προηγούμενη περίπτωση δημιουργούμε ένα τύπο π.χ. string20=string[20];

Οι τυπικές παράμετροι, παίρνουν τιμή από τις αντίστοιχες **πραγματικές** παραμέτρους κατά την κλήση της function (περιγράφεται παρακάτω). Αν αλλαχθεί η τιμή κάποιας τυπικής παραμέτρου μέσα στη function, δεν επιστρέφεται αλλαγμένη μετά την έξοδο από τη function, δηλ. οι αντίστοιχες **πραγματικές** παράμετροι εξακολουθούν να έχουν τις τιμές που είχαν τη στιγμή που κλήθηκε η function. Συνήθως χρησιμοποιούμε τις τιμές των τυπικών παραμέτρων μέσα στη function χωρίς να τις αλλάζουμε.

Ο τύπος της συνάρτησης αφορά στο όνομα της συνάρτησης.

Οι τύποι των τυπικών παραμέτρων θα πρέπει να είναι ακριβώς ίδιοι με τους τύπους των **πραγματικών** παραμέτρων, διαφορετικά εμφανίζεται μήνυμα λάθους.

Μία τουλάχιστον από τις εντολές της function δίνει τιμή στο όνομα της function δηλ. <όνομα function>:=<τιμή>.

Το όνομα της function όμως δεν πρέπει να γράφεται δεξιά του := μέσα στη function γιατί θα εμφανισθεί μήνυμα λάθους. (Η κλήση της function μπορεί να μπει δεξιά του := σε μία ειδική περίπτωση συναρτήσεων, τις αναδρομικές συναρτήσεις με τις οποίες δε θα ασχοληθούμε).

Η **κλήση της function** γίνεται από κάποιο σημείο του προγράμματος συνήθως ως εξής:
<όνομα μεταβλητής>:=<όνομα function>(μεταβλ1,μεταβλ2,...)
Η μεταβλητή αριστερά του := πρέπει να είναι ίδιου τύπου με τη function.
Η function επιστρέφει στο σημείο που κλήθηκε μια τιμή η οποία αποθηκεύεται στη μεταβλητή αριστερά του :=

Οι μεταβλ1, μεταβλ2, ... ονομάζονται **πραγματικές** παράμετροι και πρέπει να χωρίζονται με κόμμα.

Παράδειγμα

Να γραφεί πρόγραμμα το οποίο να υπολογίζει το μικρότερο από 2 αριθμούς με χρήση function. Δηλ. να δίνονται στη function οι δύο αριθμοί και να επιστρέφεται ο μικρότερος.

```
program mikroteros;
var
   a,b,mik:integer;

function min(x:integer; y:integer):integer;
var
    z:integer;
begin
     if x<y then z:=x else z:=y;
     min:=z;
end;

begin
    readln(a);
    readln(b);
    mik:=min(a,b);
```

```
    writeln('Μικρότερος είναι ο ',mik);
end.
```

Παρατηρήσεις

Η γραμμή function min(x:integer; y:integer):integer; θα μπορούσε να αντικατασταθεί με την function min(x,y:integer):integer; αφού το x και το y είναι του ίδιου τύπου.

Στο σώμα της function θα μπορούσαμε να παραλείψουμε τη μεταβλητή z και να γράψουμε απευθείας:
if x<y then min:=x else min:=y; Δεν θα συνηθίζουμε όμως να χρησιμοποιούμε το όνομα της function σε διάφορα σημεία μέσα στη function γιατί υπάρχει πιθανότητα να κάνουμε λάθος.

Η μεταβλητή a του κυρίως προγράμματος δίνει την τιμή της στην παράμετρο x της function.
Η μεταβλητή b του κυρίως προγράμματος δίνει την τιμή της στην παράμετρο y της function.

Αν αλλάζαμε τις τιμές των x και y μέσα στη function, δεν θα επηρεάζονταν οι τιμές των a και b στο κυρίως πρόγραμμα.

Θα μπορούσαμε την τιμή που επέστρεψε η fuction να την εμφανίσουμε απευθείας με την εντολή writeln χωρίς να την αποθηκεύσουμε στη μεταβλητή mik δηλ. να γράφαμε την εντολή:
writeln('Μικρότερος είναι ο ',min(a,b));

Ασκήσεις

62. Να γραφεί function που θα αθροίζει 2 αριθμούς.

63. Να γραφεί function που θα υπολογίζει το μέσο όρο 3 αριθμών

64. Να γραφεί function που θα υπολογίζει την υποτείνουσα ορθογωνίου τριγώνου.

65. Να γραφεί πρόγραμμα το οποίο θα έχει 3 επιλογές:
1. Εμβαδό τριγώνου
2. Εμβαδό ορθογωνίου παραλληλογράμμου
3. Εμβαδό κύκλου.

Να δίνονται τα απαραίτητα στοιχεία και τα εμβαδά να υπολογίζονται με χρήση τριών functions.

66. Να γραφεί function η οποία θα υπολογίζει την εφχ για τις γωνίες από 0 έως 90 μοίρες.

67α. Να γραφεί πρόγραμμα στο οποίο θα δίνεται ένα string και με χρήση function θα υπολογίζεται η στήλη στην οποία πρέπει να τυπώνεται ώστε να ειναι κεντραρισμένο στη γραμμή.

67β. Να γραφεί πρόγραμμα στο οποίο θα δίνεται ένα string και με χρήση function θα δημιουργείται νέο string το οποίο θα είναι το αντίστροφο του αρχικού.

68. Να γραφεί πρόγραμμα στο οποίο θα δίνεται η βάση a:real και ο εκθέτης n:byte και θα υπολογίζεται η δύναμη a^n.

69. Να γραφεί πρόγραμμα στο οποίο θα δίνονται οι αστικές και οι υπεραστικές μονάδες που πραγματοποίησε κάποιος από σταθερό τηλέφωνο και θα υπολογίζει το πληρωτέο ποσό όταν η χρέωση είναι: Αστικές μονάδες: 0.05 € η μία. Υπεραστικές μονάδες: 0..100 0.10 €, 101..500 0.20 € , 501..1000 0.35 € και 1001.. 0.50 € η μία

70. Να υπολογισθεί το N! με χρήση function.

71. Να δημιουργήσετε functions για το ημx και το συνx οι οποίες θα δέχονται τη γωνία σε μοίρες.

72. Να γραφεί πρόγραμμα το οποίο θα υπολογίζει το άθροισμα:
1/2 - 1/4 + 1/6 - 1/8 + ... 1/2*N

73. Να τυπωθεί τριγωνομετρικός πίνακας για τις γωνίες από 0 ώς 359 μοίρες (ημ, συν, εφ). Οι γωνίες με τους τριγωνομετρικούς αριθμούς τους να εμφανίζονται στην οθόνη ανά 20.

74. Κάποιος καταθέτει στην τράπεζα ένα κεφάλαιο. Να υπολογισθεί με χρήση function το τελικό κεφάλαιο που θα πάρει μετά από X χρόνια.

3.2. Procedures (διαδικασίες)

Οι Procedures είναι τμήματα του προγράμματος με αυτόνομη κατά κάποιο τρόπο εργασία. Όπως σε ένα κείμενο χωρίζουμε παραγράφους για να είναι πιο ευανάγνωστο, έτσι ένα πρόγραμμα Pascal το χωρίζουμε σε procedures που η καθεμιά εκτελεί μία ανεξάρτητη εργασία.

3.2.1. Procedures χωρίς παραμέτρους

```
procedure <όνομα διαδικασίας>;
.....
var
   <τοπικές μεταβλητές>;

begin
    <εντολές>;
     .............
end;
```

Οι τοπικές μεταβλητές που ορίζονται μέσα σε μια procedure έχουν υπόσταση μόνο μέσα σ' αυτήν και καταστρέφονται μετά την έξοδο από την procedure.

Για να καλέσουμε μια procedure χωρίς παραμέτρους από κάποιο σημείο του προγράμματός μας, γράφουμε απλώς το όνομά της.
Οι procedures γράφονται πριν από το κυρίως πρόγραμμα και πριν από το σημείο που καλούνται.

Παράδειγμα

```
program add_subtract;
var                          Οι μεταβλητές του κυρίως προγράμματος
  a,b:real;                   ονομάζονται ολικές μεταβλητές και έχουν
  epil:1..2;                  υπόσταση σε όλο το πρόγραμμα δηλ. και
                              μέσα στις fuctions και τις procedures.
procedure add;
var
    sum:real;
begin
    sum:=a+b;
    writeln('Αθροισμα=',sum:5:1);
end;
procedure subtract;
var
    sub:real;
begin
```

```
    sub:=a-b;
    writeln('Διαφορά=',sub:5:1);
end;

procedure wait;
begin
    write('πατήστε <enter> για συνέχεια');
    readln;
end;

begin  (* κυρίως προγράμματος *)
    write('1ος αριθμός: ');  readln(a);
    write('2ος αριθμός: ');  readln(b);
    repeat
        write('1. πρόσθεση  2. Αφαίρεση : '); readln(epil);
    until (epil=1) or (epil=2);
    if epil=1 then add else subtract;
    wait;
end.
```

3.2.2. Procedures με παραμέτρους

```
procedure <όνομα διαδικασίας>(παράμετρος1:τύπος1;
                              παράμετρος2:τύπος2;...);
Δηλώσεις
var
    τοπικές μεταβλητές;
begin
    <εντολές>;
     .....
end;
```

Οι παράμετροι της procedure χωρίζονται με ελληνικό ερωτηματικό **(;)**

Οι παράμετροι 1,2 κ.τ.λ. **αν δεν έχουν αριστερά τους τη λέξη var** ονομάζονται **τυπικές παράμετροι** και παίρνουν τιμές από αντίστοιχες μεταβλητές κατά την κλήση της διαδικασίας, οι οποίες ονομάζονται **πραγματικές παράμετροι.**
Οι τυπικές παράμετροι, παίρνουν τιμή από τις αντίστοιχες **πραγματικές** παραμέτρους κατά την κλήση της procedure (περιγράφεται παρακάτω). **Αν αλλαχθεί η τιμή κάποιας τυπικής παραμέτρου μέσα στη procedure, δεν επιστρέφεται αλλαγμένη** μετά την έξοδο από τη procedure, δηλ. οι αντίστοιχες **πραγματικές** παράμετροι εξακολουθούν να έχουν τις τιμές που

είχαν τη στιγμή που κλήθηκε η procedure. Συνήθως χρησιμοποιούμε τις τιμές των τυπικών παραμέτρων μέσα στη procedure χωρίς να τις αλλάζουμε.

Οι παράμετροι 1,2 κ.τ.λ. **αν έχουν αριστερά τους τη λέξη var** ονομάζονται **var παράμετροι** και παίρνουν, όπως και οι τυπικές παράμετροι, τιμές από αντίστοιχες μεταβλητές κατά την κλήση της διαδικασίας**.**

Αν αλλαχθεί όμως η τιμή κάποιας var παραμέτρου μέσα στη procedure, επιστρέφεται αλλαγμένη μετά την έξοδο από τη procedure, δηλ. η αντίστοιχη **πραγματική** παράμετρος παίρνει τη νέα τιμή της var παραμέτρου. Δηλαδή, **ορίζουμε μία παράμετρο var όταν θέλουμε να επιστραφεί αλλαγμένη η τιμή της**.

Δεν μπορεί οι τυπικές και οι var παράμετροι να είναι τύπου υποπεριοχής τιμών (π.χ. 0..20) ή τύπου string[n]. (Βλ. functions)
Οι τύποι των τυπικών παραμέτρων και των var παραμέτρων θα πρέπει να είναι ακριβώς ίδιοι με τους τύπους των **πραγματικών** παραμέτρων, διαφορετικά εμφανίζεται μήνυμα λάθους.

Οι τυπικές παράμετροι καταλαμβάνουν νέες θέσεις στη μνήμη.
Οι var παράμετροι δεν καταλαμβάνουν νέες θέσεις στη μνήμη (είναι οι ίδιες θέσεις μνήμης των αντίστοιχων **πραγματικών** παραμέτρων).

Η **κλήση της procedure** γίνεται από κάποιο σημείο του προγράμματος συνήθως ως εξής:
<όνομα procedure>(μεταβλ1,μεταβλ2,...)
Προσέξτε ότι στην κλήση της procedure οι **πραγματικές** παράμετροι χωρίζονται με κόμμα.

Μπορούμε να βγούμε από μία procedure χωρίς να έχει ολοκληρωθεί η εκτέλεση όλων των εντολών της, χρησιμοποιώντας την εντολή **exit.**

Παράδειγμα

```
program mesos;
var
  x,y:integer;
  mo:real;

procedure mesos(a,b:integer; var mes:real);
begin
    mes:=(a+b)/2;
end;
begin  (* κυρίως προγράμματος *)
    write('1ος αριθμός: ');  readln(x);
```

```
    write('2ος αριθμός: ');  readln(y);
    mesos(x,y,mo);
    writeln('Μέσος όρος=',mo:6:2);
end.
```

- Η x δίνει την τιμή της στην a.
- Η y δίνει την τιμή της στην b.
- Η mo δίνει την τιμή της στη mes. (δηλ. δίνει σκουπίδια)

Μετά την εκτέλεση της procedure

- Η a δεν επιστρέφει την τιμή της στη x.
- Η b δεν επιστρέφει την τιμή της στη y.
- Η mes επιστρέφει την τιμή της στη mo γιατί είναι var παράμετρος.

Ασκήσεις

75. Να γραφεί πρόγραμμα το οποίο θα έχει τις επιλογές:
1. Εύρεση μεγίστου 3 αριθμών
2. Εύρεση ελαχίστου 3 αριθμών
3. Εύρεση μέσου όρου 3 αριθμών.

Η κάθε επιλογή να είναι procedure χωρίς παραμέτρους.

76. Να γραφεί procedure η οποία θα επιστρέφει την υποτείνουσα ορθογωνίου τριγώνου και μια μεταβλητή τύπου boolean η οποία θα έχει την τιμή true αν η υποτείνουσα είναι ακέραιος.

77. Να συμπληρωθούν κατά τον καλύτερο τρόπο τα κενά:

```
program sales_week;
var .....

procedure compute(.....);
var .....
begin
    write('πωλήσεις 1ου τμήματος: ');
    readln(sales1);
    write('πωλήσεις 2ου τμήματος: ');
    readln(sales2);
    s:=sales1+sales2;
end;

begin
    compute(.....);
    writeln('Σύνολο: ',total);
end.
```

78. Τι τυπώνεται στην οθόνη κατά την εκτέλεση:

```
program parameters;
var g1,g2: integer;
procedure varval(pm1: integer; var pm2: integer);
var pr1,pr2: integer;
begin
  pr1:=1; pr2:=2; pm1:=pm1+pr1+pr2;  pm2:=pm2+pr1+pr2;
end;
function showscope(pm1: integer): integer;
var g1,fn: integer;
begin
  g1:=0; fn:=2; pm1:=pm1+fn; writeln(pm1); writeln(g1);
  showscope:=g1;
end;
begin  g1:=1; g2:=2;
       varval(g1,g2); writeln(g1); writeln(g2);
       g2:=showscope(g1); writeln(g1); writeln(g2);
end.
```

79. Να γραφεί procedure που θα μετατρέπει ένα string από πεζά σε κεφαλαία γράμματα. το νέο string να τυπώνεται στο κυρίως πρόγραμμα.

80. Δίνεται η τωρινή ώρα και λεπτά και Χ λεπτά που θέλουμε να περάσουν. Να γραφεί procedure που θα επιστρέφει τη νέα ώρα και λεπτά μετά την πρόσθεση των Χ λεπτών.

81. Να γραφεί procedure που θα ανταλλάσσει τις τιμές 2 μεταβλητών.

82. Να γραφεί procedure που θα υπολογίζει τον ελάχιστο 3 αριθμών.

83. Τι τυπώνεται στην οθόνη κατά την εκτέλεση:

```
program parameters;
var a,b,c: integer;
procedure one(x,y: integer; var z: integer);
var a: integer;
begin
  a:=1; b:=7; x:=y;  z:=a+x;
end;
begin
       a:=4; b:=5; c:=12;
       one(a,b,c); writeln(a,b,c);
end.
```

84. Να γραφεί procedure η οποία θα επιστρέφει τον αριθμό των ημερών ανάμεσα σε 2 ημερομηνίες.

3.3. Επιλογή χρωμάτων φόντου και χαρακτήρων

Για να επιλέξουμε χρώματα φόντου (textbackground) και χαρακτήρων (textcolor) πρέπει να συμπεριλάβουμε τη μονάδα crt, δηλ. κάτω από το Program <όνομα προγράμματος>; να γράψουμε: **uses crt.**

Επιλογή χρώματος χαρακτήρων

```
textcolor(<χρώμα>);
```

Επιλογή χρώματος φόντου

```
textbackground(<χρώμα>);
```

τιμές χρωμάτων

Μαύρο	0	black
Μπλε	1	blue
πράσινο	2	green
Γαλάζιο	3	cyan
Κόκκινο	4	red
Ροζ	5	magenta
Καφέ	6	brown
Ανοικτό γκρι	7	lightgray
Σκούρο γκρι	8	darkgray
Ανοικτό μπλε	9	lightblue
Ανοικτό πράσινο	10	lightgreen
Ανοικτό γαλάζιο	11	lightcyan
Ανοικτό κόκκινο	12	lightred
Ανοικτό ροζ	13	lightmagenta
Κίτρινο	14	yellow
Άσπρο	15	white

Στις διαδικασίες textcolor και textbackground, μπορούμε να γράφουμε τα χρώματα είτε με τον αντίστοιχο αριθμό τους (2η στήλη) είτε με το όνομά τους στα αγγλικά (3η στήλη).
Δηλ. η textcolor(yellow) έχει το ίδιο αποτέλεσμα με την textcolor(14).
Αν στην textcolor, προσθέσουμε δίπλα στο χρώμα **+blink** (ή +128), τότε οι χαρακτήρες αναβοσβήνουν π.χ. textcolor(yellow+blink);

Αν επιλέξουμε το χρώμα του φόντου ίδιο με το χρώμα των χαρακτήρων, τότε δεν εμφανίζεται τίποτε στην οθόνη. Αυτό είναι χρήσιμο όταν εισάγουμε κωδικούς, για να μην εμφανίζονται στην οθόνη την ώρα που τους πληκτρολογούμε.

Για να χρησιμοποιήσουμε τις συναρτήσεις whereY και whereX, πρέπει να συμπεριλάβουμε τη unit crt (δηλ. κάτω από το program <όνομα προγράμματος>; να γράψουμε **uses crt**;

Εύρεση της τρέχουσας γραμμής του δρομέα στην οθόνη

whereY

Η συνάρτηση whereY επιστρέφει τη γραμμή στην οποία βρίσκεται ο δρομέας.

Εύρεση της τρέχουσας στήλης του δρομέα στην οθόνη

whereX

Η συνάρτηση whereX επιστρέφει τη στήλη στην οποία βρίσκεται ο δρομέας.

Η συνάρτηση readkey

<μεταβλητή τύπου char>:=readkey;

Με τη συνάρτηση readkey διαβάζεται από το πληκτρολόγιο μία τιμή ενός χαρακτήρα και αποθηκεύεται στη μεταβλητή τύπου char. Αφού πληκτρολογηθεί ο χαρακτήρας, εκτελείται η επόμενη εντολή (χωρίς να πατήσουμε το <enter>).

Παράδειγμα

```
procedure wait;
var
    c:char;
begin
    write('πατήστε οποιδήποτε πλήκτρο για συνέχεια ');
    c:=readkey;
end;
```

4. Πίνακες

4.1. Μονοδιάστατοι πίνακες

Έστω ότι θέλουμε να ταξινομήσουμε σε αύξουσα σειρά 100 αριθμούς. Οι 100 αριθμοί θα πρέπει να βρίσκονται όλοι αποθηκευμένοι στη μνήμη του υπολογιστή, έτσι ώστε να γίνουν οι απαραίτητες συγκρίσεις μεταξύ τους. Άρα θα πρέπει να δημιουργηθούν 100 θέσεις μνήμης, στις οποίες να αποθηκευθούν οι 100 αριθμοί. πρακτικά δηλαδή, με τις εντολές που μάθαμε μέχρι τώρα, θα πρέπει να δηλώσουμε 100 διαφορετικά ονόματα μεταβλητών και να κάνουμε ένα τεράστιο αριθμό συγκρίσεων. Οι παραπάνω δυσκολίες αντιμετωπίζονται με μία νέα δομή δεδομένων που ονομάζεται πίνακας.
Πίνακας είναι μία δομή δεδομένων που αποτελείται από ίδιου τύπου δεδομένα π.χ. 100 αριθμοί τύπου integer, 20 ονοματεπώνυμα τύπου string κ.τ.λ. **Κάθε στοιχείο του πίνακα προσδιορίζεται από το όνομα του πίνακα και από την τιμή του δείκτη**. π.χ. το 3ο στοιχείο του πίνακα marks είναι το marks[3].

Δήλωση πίνακα

α' τρόπος
```
Type
   <τύπος πίνακα>=array[1..<ακέραιος>] of <τύπος>;
Var
   <όνομα πίνακα>:<τύπος πίνακα>;
```
β' τρόπος
```
Var
   <όνομα πίνακα>:array[1..<ακέραιος>] of <τύπος>;
```

Ο ακέραιος μπορεί να είναι ένας αριθμός ή μία σταθερά που έχει δηλωθεί παραπάνω με const και συμβολίζει τον (μέγιστο) αριθμό των θέσεων του πίνακα.
Η καταχώριση τιμών στον πίνακα γίνεται συνήθως με μία for.

1ο παράδειγμα
```
type
    arith=array[1..10] of integer;    (* πίνακας 10 ακεραίων *)
var
    num:arith;
```

2ο παράδειγμα
```
var
    num:array[1..10] of integer;   (* ανώνυμος τύπος πίνακα *)
```

Για να αναφερθούμε σε συγκεκριμένο στοιχείο του πίνακα γράφουμε num[i] όπου το i είναι ακέραιος που προσδιορίζει τη θέση του στοιχείου που θέλουμε. Συνήθως, όταν θέλουμε ένας πίνακας να είναι παράμετρος σε μία function ή procedure, **τον δηλώνουμε var παράμετρο** για οικονομία μνήμης ακόμη κι αν δεν θέλουμε οι τιμές του να επιστραφούν αλλαγμένες.

Ο τύπος μιας function δεν μπορεί να είναι τύπος πίνακα, ενώ οι παράμετροι της function μπορεί να είναι πίνακες.

4.2. Πίνακες δύο διαστάσεων

Παράδειγμα

Μαθήματα	**Α' τριμ.**	**Β' τριμ.**	**Γ' τριμ.**	**Γραπτά**
Cobol	**18**	**19**	**20**	**19**
Pascal	**19**	**20**	**20**	**20**
Λειτ. συστήματα	**15**	**15**	**18**	**17**

Οι παραπάνω βαθμοί μπορούν να αποθηκευθούν σε έναν πίνακα δύο διαστάσεων με 3 γραμμές (μία για κάθε μάθημα) και 4 στήλες (μία για κάθε τρίμηνο και τα γραπτά). π.χ. ο βαθμός του Γ'τριμήνου στην Pascal είναι το στοιχείο [2,3] δηλ. 2η γραμμή (Pascal) και 3η στήλη (Γ'τρίμηνο)

4.2.1 Δήλωση πίνακα 2 διαστάσεων

α' τρόπος
```
Type
    <τύπος πίνακα>=array[1..<ακέρ-1>,1..<ακέρ-2>]of <τύπος>;
Var
   <όνομα πίνακα>:<τύπος πίνακα>;
```
β' τρόπος
```
Var
    όνομα πίνακα>=array[1..<ακέρ-1>,1..<ακέρ-2>]of <τύπος>;
```

Ο <ακέρ-1> συμβολίζει το μέγιστο πλήθος των γραμμών του πίνακα και ο <ακέρ-2> το μέγιστο πλήθος των στηλών του.

4.2.2. Καταχώριση τιμών σε πίνακα δύο διαστάσεων

Για την καταχώριση τιμών σε πίνακα δύο διαστάσεων χρησιμοποιούνται συνήθως δύο for (η μία μέσα στην άλλη). Στο παραπάνω παράδειγμα, η καταχώριση τιμών στον πίνακα βαθμολογίας θα μπορούσε να γίνει είτε κατά γραμμές (η for των γραμμών είναι η εξωτερική και η for των στηλών είναι η εσωτερική) είτε κατά στήλες (η for των στηλών είναι η εξωτερική και η for των γραμμών είναι η εσωτερική).

```
program bathmologia;   (* Καταχώριση τιμών κατά γραμμές *)
var
    a:array[1..3,1..4] of byte;
    i,j:byte;
begin
    for i:=1 to 3 do
      for j:=1 to 4 do
      begin
          write (i,'η γραμμή ',j',η στήλη : ');
          readln(a[i,j]);
       end;
  .....
end.
```

4.2.3. Εφαρμογή 1 (Μέθοδος Bubble Sort)

ταξινόμηση πίνακα σε αύξουσα σειρά με τη μέθοδο Bubble Sort.

```
program bubble_sort_1;
uses crt;
type pinakas=array[1..10] of integer;
var
   pin:pinakas;  n:byte;

procedure input_array(pl:byte; var p:pinakas);
var   i:byte;
begin
    clrscr;
    for i:=1 to pl do
    begin
        write(i:2,'o στοιχείο : '); readln(p[i]);
    end;
end;
```

```
procedure bubble1(pl:byte; var p:pinakas);
var
  i,j:byte;
  t:integer;
begin
    for i:=1 to pl-1 do
      for j:=i+1 to pl do
      begin
          if p[i]>p[j] then
          begin
              t:=p[i];  p[i]:=p[j];  p[j]:=t;
          end;
      end;
end;

procedure print_array(pl:byte; var p:pinakas);
var
 i:byte;   t:integer;
begin
    clrscr;
    for i:=1 to pl do writeln(i:2,'ο στοιχείο : ',p[i]);
end;

begin
     clrscr;  write('πλήθος στοιχείων πίνακα : ');  readln(n);
     input_array(n,pin);   bubble1(n,pin);
     print_array(n,pin);   readln;
end.
```

4.2.4. Εφαρμογή 2 (Βελτιωμένη μέθοδος Bubble Sort)

ταξινόμηση πίνακα σε αύξουσα σειρά με τη βελτιωμένη μέθοδο Bubble Sort.

```
procedure bubble2(pl:byte; var p:pinakas);
var
     i:byte;  t:integer;  change:boolean;
begin
         repeat
              change:=false;
              for i:=1 to pl-1 do
                   if p[i]>p[i+1] then
                   begin
                        t:=p[i];  p[i]:=p[i+1];
```

```
                    p[i+1]:=t;  change:=true;
                end;
            until change=false;
end;
```

4.2.5. Εφαρμογή 3 (Δυαδική αναζήτηση στοιχείου)

Να δοθεί πίνακας ακεραίων ταξινομημένος σε αύξουσα σειρά και ένας ακέραιος αριθμός. Να βρεθεί μία θέση του ακεραιου μέσα στον πίνακα ή να τυπωθεί το μήνυμα "Δεν βρέθηκε" αν δεν ανήκει στον πίνακα.

```
(* Ο ΠΙΝΑΚΑΣ ΠΡΕΠΕΙ ΝΑ ΕΙΝΑΙ ΤΑΞΙΝΟΜΗΜΕΝΟΣ *)
program binary;
uses crt;
type pinakas=array[1..10] of integer;
var
   pin:pinakas;   n:byte;   find:integer;
procedure input_array(pl:byte; var p:pinakas);
var  i:byte;
begin
    clrscr;
    for i:=1 to pl do
    repeat        (* Η repeat αναγκάζει το χρήστη να εισάγει τα στοιχεία
                     του πίνακα ταξινομηνένα σε αύξουσα σειρά *)
        write(i:2,'ο στοιχείο : '); readln(p[i]);
    until p[i]>=p[i-1];
end;
procedure binary_search(pl:byte; var p:pinakas; x:integer);
var  l,l1,l2:byte;
begin
    l1:=1;  l2:=pl;
    while l1<=l2 do
    begin
        l:=(l1+l2) div 2;
        if p[l]>x then l2:=l-1;  if p[l]<x then l1:=l+1;
        if p[l]=x then
        begin
            writeln('το στοιχείο βρέθηκε στη θέση ',l);  exit;
        end;
    end; (* while *)
    writeln('Δεν βρέθηκε');
end;
begin
     clrscr;  write('πλήθος στοιχείων πίνακα : ');  readln(n);
```

```
    input_array(n,pin); writeln;
    write('Ζητούμενο στοιχείο : ');  readln(find);  writeln;
    binary_search(n,pin,find);  readln;
end.
```

Ασκήσεις

85. Να γραφεί function που θα υπολογίζει το άθροισμα των Ν στοιχείων ενός πίνακα.

86. Να γραφεί procedure που θα γεμίζει έναν πίνακα Ν (όπου Ν<=30) στοιχείων με την τιμή του ίδιου του δείκτη (δηλ. α[1]=1, α[2]=2 κ.τ.λ.) και θα υπολογίζει το γινόμενό τους με function.

87. Δίνεται πίνακας Ν στοιχείων. Να βρεθεί το μεγαλύτερο στοιχείο του.

88. Δίνεται πίνακας Ν στοιχείων, ταξινομημένος σε αύξουσα σειρά. (Να δοθούν τα στοιχεία του ταξινομημένα απ' τον χρήστη). Κατόπιν δίνεται αριθμός Χ. Να καταχωρηθεί ο Χ μέσα στον πίνακα έτσι ώστε να συνεχίσει να είναι ταξινομημένος.

89. Να γραφεί πρόγραμμα το οποίο θα διαγράφει ένα στοιχείο από έναν πίνακα. Τα στοιχεία του πίνακα να δοθούν ταξινομημένα και διαφορετικά μεταξύ τους.

90. Δίνονται στοιχεία για Ν άτομα: Κωδικός φύλου (Γ:γυναίκα, Α:άνδρας), βάρος και ηλικία. Να βρεθούν
α) ο μέσος όρος ηλικίας των γυναικών με βάρος >60 κιλών
β) πόσο ετών είναι ο βαρύτερος άνδρας.

91. Να ταξινομηθεί ένας πίνακας Ν ονοματεπωνύμων σε αλφαβητική σειρά. (**Μέθοδος Bubble sort**)

92. Να ταξινομηθεί ένας πίνακας Ν ακεραίων σε φθίνουσα σειρά. (**Βελτιωμένη μέθοδος Bubble sort**)

93. Δίνεται πίνακας Ν στοιχείων και ένας αριθμός Χ. Να βρεθεί η θέση ή οι θέσεις του Χ μέσα στον πίνακα. Αν δεν ανήκει στον πίνακα, να τυπωθεί ανάλογο μήνυμα. (Χρήση μεταβλητής τύπου boolean π.χ. της found: Αν βρεθεί το στοιχείο τότε found:=true αλλιώς found:=false). (**Σειριακή αναζήτηση**)

94. Δυαδική αναζήτηση ονοματεπωνύμου σε πίνακα με ονοματεπώνυμα (**Binary search**).

95. Δίνονται 2 ταξινομημένοι πίνακες Α[Ν] & Β[Μ]. Να ενωθούν σε έναν πίνακα C ώστε να είναι επίσης ταξινομημένος.

96. Να βρεθεί το άθροισμα κάθε γραμμής του Α[Ν,Μ] και να καταχωρηθεί σε πίνακα Β[Ν].

97. Να βρεθεί το άθροισμα των στοιχείων της κυρίας διαγωνίου ενός τετραγωνικού πίνακα Α[Ν,Ν].

98. Να δημιουργηθεί τετραγωνικός πίνακας A[N,N] έτσι ώστε όλα τα στοιχεία πάνω από την κύρια διαγώνιο (I<J) να έχουν την τιμή 1, τα στοιχεία της κυρίας διαγωνίου (I=J) την τιμή 0 και τα στοιχεία κάτω από την κύρια διαγώνιο (I>J) την τιμή 2.

99. Να προστεθούν δύο πίνακες A[N,M] και B[N,M]

100. Δίνονται οι βαθμοί 5 μαθητών σε 3 μαθήματα. (Ενας πίνακας A[5,3]). Να υπολογισθούν οι μέσοι όροι βαθμολογίας και να τοποθετηθούν σε πίνακα B[5]

101. Δίνονται τα αποτελέσματα των εκλογών σε 4 εκλογικές περιφέρειες 3 υποψηφίων. Να υπολογισθούν α) το σύνολο των ψήφων της καθε περιφέρειας β) το σύνολο των ψήφων του κάθε υποψηφίου. (Να γραφεί μόνο μία procedure η οποία ανάλογα θα υπολογίζει ή το άθροισμα των γραμμών του πίνακα ή το άθροισμα των στηλών)

102. Τα strings αντιμετωπίζονται σαν πίνακας μιας διάστασης.
Να γραφεί πρόγραμμα στο οποίο ο χρήστης θα δίνει μια πρόταση με κεφαλαία ελληνικά γράμματα η οποία θα τελειώνει μόλις δοθεί τελεία.
το πρόγραμμα θα υπολογίζει:
α) αριθμό λέξεων (οι λέξεις χωρίζονται με ένα ή περισσότερα κενά ή με ένα κόμμα)
β) αριθμός γραμμάτων (αλφαβήτου)
γ) ποσοστό (%) εμφάνισης του γράμματος με τη μεγαλύτερη συχνότητα και του γράμματος με τη μικρότερη συχνότητα.
δ) ποια γράμματα δεν εμφανίζονται καθόλου στην πρόταση

103. Δίνεται η μέση βροχή και θερμοκρασία κάθε μήνα για ένα έτος (να δημιουργηθούν 2 πίνακες, ένας για τη βροχή και ένας για τη θερμοκρασία). Να υπολογισθούν η μέγιστη βροχή και η μέγιστη θερμοκρασία με χρήση μιας μόνο συνάρτησης η οποία θα καλείται από το κυρίως πρόγραμμα 2 φορές, μία για τη βροχή και μία για τη θερμοκρασία.

5. Procedures και functions βιβλιοθήκης με παραμέτρους τύπου string

5.1 Procedures

delete(<string>,<αρχή>,<n>);

Διαγράφει από το <string>, <n> χαρακτήρες αρχίζοντας από το χαρακτήρα στη θέση <αρχή>.

Παράδειγμα

```
st:='Αυτό είναι ένα string';
delete(st,12,3);
writeln(st);    (* Θα τυπώσει: Αυτό είναι string *)
```

insert(<string-1>,<string-2>,<n>);

Παρεμβάλλει το <string-1> στο <string-2> στη θέση <n>.

Παράδειγμα

```
s:='ένα';
t:='Αυτό είναι string';
insert(s,t,12);
writeln(t);   (* Θα τυπώσει: Αυτό είναι ένα string *)
```

str(<i>, <string>);

Μετατρέπει τον αριθμό <i> σε αλφαριθμητικό που το αποθηκεύει στη μεταβλητή <string>.

Παράδειγμα

```
var
  i:integer;
  st:string;
begin
   i:=351;
   str(i,st);
   writeln(st);    (* Θα τυπώσει: 351 *)
end.
```

val(<string>,<i>,<error>);

Μετατρέπει το <string> σε αριθμό που τον αποθηκεύει στη μεταβλητή <i>. Η μεταβλητή <error> παίρνει την τιμή 0 αν η μετατροπή ήταν επιτυχής ή την τιμή της θέσης του χαρακτήρα που προκάλεσε το λάθος.

Παράδειγμα

```
var
    i, error:integer;
    st:string;
begin
    st:='123';
    val(st,i,error);
    if error<>0 then halt;
    (* Η εντολή halt διακόπτει την εκτέλεση του προγράμματος *)
    writeln('Αριθμός=',i);
end.
```

5.2 Functions

copy(<string-1>,<αρχή>,<n>):string;

Αντιγράφει <n> χαρακτήρες του <string-1> αρχίζοντας από τη θέση <αρχή>.

Παράδειγμα

```
st1:=' Αυτό είναι ένα string';
cs:=copy(st1,12,3);
writeln(cs);   (* Θα τυπώσει: ένα *)
```

concat(<string-1>,<string-2>,...,<string-n>):string;

Ενώνει τα strings σε ένα string το οποίο δεν πρέπει να ξεπερνά τους 255 χαρακτήρες.
Η συνένωση μπορεί να γίνει πιο εύκολα και με το σύμβολο + .

Παράδειγμα

```
st1:='Η γλώσσα προγραμματισμού ');
st2:='PASCAL');
st:=concat(st1,st2);   (*  ή st:=st1+st2;  *)
writeln(st);   (* Θα τυπώσει: Η γλώσσα προγραμματισμού PASCAL');
```

length(<string>):integer;

Δίνει το πραγματικό μήκος του string δηλ. από πόσους χαρακτήρες αποτελείται η τιμή του (όχι πως δηλώθηκε).

Παράδειγμα

```
var
  st:string[30];
begin
  st:='ΘΕΣ/ΝΙΚΗ';
  l:=length(st);
  writeln(l);   (* Θα τυπώσει: 8 *)
end.
```

pos(<string-1>,<string-2>):integer;

Δίνει τη θέση του 1ου χαρακτήρα του <string-1> μέσα στο <string-2> Αν δεν υπάρχει το <string-1> μέσα στο <string-2> η pos δίνει την τιμή 0.

Παράδειγμα

```
st1:='PASCAL';
st2:='Η γλώσσα PASCAL είναι εύκολη';
i:=pos(st1,st2);
writeln(i);  (* Θα τυπώσει: 10 *)
```

upcase(<χαρακτήρας>):<χαρακτήρας>;

Μετατρέπει το <χαρακτήρα> από πεζό σε κεφαλαίο. Χρησιμοποιείται μόνο για λατινικούς χαρακτήρες.

Παράδειγμα

```
writeln(upcase('a'));  (* Θα τυπώσει: A *)
```

Ασκήσεις

104. Δίνονται τα ονοματεπώνυμα και τα υπόλοιπα των πελατών μιας επιχείρησης σε **έναν** πίνακα δύο διαστάσεων. Να υπολογισθεί το συνολικό υπόλοιπο. (τα υπόλοιπα μέσα στον πίνακα καταχωρούνται σαν strings και για να λάβουν μέρος σε πράξεις μετατρέπονται σε αριθμούς με τη συν. **VAL**).

105. Να γραφεί procedure που θα δέχεται ένα string, ένα χαρακτήρα που θα αντικατασταθεί (old) και έναν χαρακτήρα που θα αντικαταστήσει τον προηγούμενο (new). Η procedure θα επιστρέφει το string διορθωμένο. Η άσκηση να λυθεί με δύο τρόπους: α) να αντικαθιστάται μόνο ένας χαρακτήρας old (ο πρώτος που βρίσκεται μέσα στο string, β) να αντικαθιστούνται όλοι οι χαρακτήρες old που υπάρχουν μέσα στο string με το νέο χαρακτήρα new)

106. Να επαναληφθεί η προηγούμενη άσκηση έτσι ώστε να γίνεται αντικατάσταση ολόκληρης λέξης με μια άλλη.

107. Να γραφεί πρόγραμμα που θα υπολογίζει πόσες φορές υπάρχει ένας χαρακτήρας μέσα σ' ένα string.

108. Να γραφεί procedure η οποία θα μετατρέπει ένα ελληνικό string πεζών χαρακτήρων σε κεφαλαία.

109. Να γραφεί πρόγραμμα που θα τυπώνει κάθετα ένα string.

110. Να γραφεί πρόγραμμα που θα τυπώνει 3 strings σε σχήμα π.

111. Να γραφεί πρόγραμμα το οποίο θα δίνει πρόσβαση στον Η/Υ του χρήστη αν πληκτρολογήσει σωστά τον κωδικό πρόσβασης. Οι προσπάθειές του είναι 3. Αν δεν τον βρει, το πρόγραμμα θα τερματίζει. Οι χαρακτήρες του κωδικού, να μην εμφανίζονται στην οθόνη καθώς πληκτρολογούνται.

112. Να γραφεί procedure στην οποία θα δίνεται ένα ονοματεπώνυμο (πρώτα το όνομα και μετά το επώνυμο) και θα επιστρέφει το αρχικό του ονόματος και το επώνυμο.

113. Δίνεται πίνακας Ν θέσεων με ονοματεπώνυμα. Να δημιουργηθούν δύο πίνακες ένας για τα ονόματα και ένας για τα επώνυμα.

6. Εγγραφές (records)

Όπως μάθαμε, τα στοιχεία ενός πίνακα πρέπει να είναι όλα του ίδιου τύπου. πολλές φορές όμως θέλουμε να αποθηκεύσουμε στη μνήμη, διάφορα στοιχεία που αφορούν ένα πρόσωπο ή αντικείμενο. π.χ. για κάθε πελάτη της μία εταιρεία θα ήθελε να κρατάει Ονοματεπώνυμο, Διεύθυνση, τηλέφωνο, Υπόλοιπο λογαριασμού κ.ά.

Ένα record είναι μία συλλογή στοιχείων όχι αναγκαστικά του ίδιου τύπου που όμως έχουν σχέση μεταξύ τους (π.χ. αφορούν έναν πελάτη, ένα προϊόν κ.τ.λ.). Τα στοιχεία αυτά ονομάζονται **πεδία** του record.
Πολλά records μπορούν να τοποθετηθούν σε έναν πίνακα (π.χ. για να αποθηκευθούν στη μνήμη του υπολογιστή) ή σε ένα **αρχείο** (π.χ. για να αποθηκευθούν στο δίσκο).

6.1. Δήλωση record

```
type
    <τύπος record>=record
            <όνομα πεδίου-1>:<τύπος-1>;
            <όνομα πεδίου-2>:<τύπος-2>;
             ..................
            <όνομα πεδίου-n>:<τύπος-n>;
    end;
var
    <όνομα record>:<τύπος record>;
```

Για να αναφερθούμε σε συγκεκριμένο πεδίο του record γράφουμε <όνομα record>.<όνομα πεδίου>

Παράδειγμα 1

```
type
    t_customer=record
        fname:string;
        lname:string;
        address:string;
        total:longint;
    end;
var
    customer:t_customer;
```

Παράδειγμα 2

```
type
    date=record
        year:1900..2100;
        month:1..12;
        day:1..31;
    end;
var
    today:date;
```

Για να αναφερθούμε π.χ. στο πεδίο μήνας της today γράφουμε today.month

Παράδειγμα

```
if (today.month=12)and(today.day=25) then writeln('Χριστούγεννα');
```

Στην προηγούμενη εντολή if χρησιμοποιήσαμε το record today δύο φορές γιατί αναφερθήκαμε δύο φορές σε κάποιο πεδίο του record. Σε παρόμοιες περιπτώσεις μπορούμε αντί να γράφουμε κάθε φορά το όνομα του record, να "κλείνουμε" τις εντολές που περιέχουν τα πεδία του record σε μία εντολή with. Όσες εντολές βρίσκονται μέσα στη with και δεν περιέχουν κανένα πεδίο του record, δεν επηρεάζονται.

```
with <όνομα record> do
begin
    <εντολή-1>;
    <εντολή-2>;
        .....
    <εντολή-n>;
end;
```

Παράδειγμα

```
with today do
begin
    write('Μέρα  :'); readln(day);
    write('Μήνας :');readln(month);
    write('¸τος    :');readln(year);
    if (month=12)and(day=25) then write('Χριστούγεννα ');
    writeln(year);
end;
```

6.2. Πίνακας με στοιχεία τύπου record

```
type
    <τύπος record>=record
            <όνομα πεδίου-1>:<τύπος-1>;
            <όνομα πεδίου-2>:<τύπος-2>;
             ..................
            <όνομα πεδίου-n>:<τύπος-n>;
    end;
    <τύπος πίνακα>=array[1..<n>] of <τύπος record>
var
    <όνομα πίνακα>: <τύπος πίνακα>;
```

Για να αναφερθούμε σε συγκεκριμένο πεδίο συγκεκριμένου record του πίνακα γράφουμε: **<όνομα πίνακα>[<θέση>].<όνομα πεδιου>**

Παράδειγμα

```
program phones;
type
    tpers=record
        name:string[30];
        phone:string[15];
    end;
    arraypers=array[1..5] of tpers;
var
    persons:arraypers;    i:byte;
begin  (* κυρίως πρόγραμμα *)
    for i:=1 to 5 do
    begin
        write('Ονοματεπώνυμο: ');readln(persons[i].name);
        write('τηλέφωνο       : ');readln(persons[i].phone);
        writeln;
    end;
end.
```

[1] καλύτερα **με την εντολή with**, το κυρίως πρόγραμμα θα μπορούσε να είναι ως εξής:

```
for i:=1 to 5 do   (*Η for δεν έχει begin γιατί περιέχει μόνο τη with*)
with persons[i] do
begin
    write('Ονοματεπώνυμο: ');readln(name);
    write('τηλέφωνο       : ');readln(phone);
    writeln;
end;
```

Ασκήσεις

114. Το record ενός πελάτη περιέχει τα στοιχεία: Ονοματεπώνυμο, διεύθυνση και τηλέφωνο. Να δοθούν τα στοιχεία ενός πελάτη και να τυπωθούν σε νέα οθόνη το ονοματεπώνυμο και το τηλέφωνό του.

115. Να επαναληφθεί η προηγούμενη άσκηση για 5 πελάτες τα στοιχεία των οποίων θα τοποθετηθούν σε πίνακα (κάθε στοιχείο του πίνακα θα είναι ένα record)

116. Το record ενός πελάτη περιέχει τα στοιχεία: Ονοματεπώνυμο, τηλέφωνο και υπόλοιπο λογαριασμού. Να δοθούν τα στοιχεία Ν πελατών και να τυπωθούν τα ονοματεπώνυμα των πελατών που το υπόλοιπό τους ξεπερνά της 100000 δρχ.

117. Να δοθούν 10 ημερομηνίες (record: μέρα, μήνας, έτος) και να τοποθετηθούν σε πίνακα. Στη συνέχεια να δοθεί μια ημερομηνία και να βρεθεί ποιες απ' τις 10 ημερομηνίες του πίνακα είναι μικρότερες από τη δοσμένη.

118. Για κάθε φορολογούμενο κρατούνται τα εξής στοιχεία: Όνομα, επώνυμο και αποδοχές. Αν οι αποδοχές είναι λιγότερες από 10000 τότε δεν πληρώνει φόρο αλλιώς πληρώνει φόρο 10%. Το πρόγραμμα να έχει τις επιλογές:

1. Εισαγωγή στοιχείων φορολογούμενου
2. Εμφάνιση ονοματεπωνύμων όσων δεν φορολογούνται
3. Εμφάνιση ονοματεπωνύμων και ποσού που θα πληρώσουν όσων φορολογούνται.

119. Σ' ένα μαιευτήριο, για κάθε βρέφος κρατούνται τα εξής στοιχεία: Όνομα πατέρα, όνομα μητέρας, βάρος, ύψος, ημερομηνία και ώρα γέννησης (η ημερομηνία και η ώρα να είναι records μέσα στο record του βρέφους). Να τυπωθούν τα στοιχεία των βρεφών που γεννήθηκαν μια συγκεκριμένη μέρα.

120. Δίνονται το ονοματεπώνυμο και οι βαθμοί σε 5 μαθήματα ενός μαθητή. Να βρεθεί ο μέσος όρος βαθμολογίας του και να τοποθετηθεί σ'ένα πεδίο του record του.

121. Για κάθε επιταγή κρατούνται: ποσό, ημερομηνία λήξης (record: μέρα, μήνας, έτος), χρεώστης (record: όνομα, επώνυμο, τηλέφωνο). Να εμφανισθούν τα στοιχεία της επιταγής σε νέα οθόνη αν έχει λήξει.

Επαναληπτικές ασκήσεις

122. Να γραφεί πρόγραμμα το οποίο με χρήση function θα υπολογίζει τη δύναμη $α^{ν}$.

123. Να γραφεί πρόγραμμα το οποίο θα υπολογίζει το άθροισμα θετικών αριθμών. Η διαδικασία να σταματά όταν δοθεί αριθμός αρνητικός ή μηδέν. (Με while και με repeat)

124. Να γραφεί procedure color που θα περιέχει τις εντολές textcolor & textbackground στην οποία θα δίνονται τιμές για το χρώμα των χαρακτήρων και του φόντου.

125. Να γραφεί function που θα επιστρέφει τη στήλη από την οποία πρέπει να ξεκινά η εκτύπωση ενός string έτσι ώστε να είναι κεντραρισμένο στη γραμμή. Στη συνέχεια, στο κυρίως πρόγραμμα να εμφανίζεται το string σε καθαρή οθόνη και να υπογραμμίζεται ακριβώς.

126. Να υπολογισθεί το άθροισμα $1^2+3^2+5^2+...+N^2$
όπου Ν: περιττός αριθμός

127. Δίνεται πίνακας 12 θέσεων (μία για κάθε μήνα ενός έτους) ο οποίος περιέχει τον αριθμό των βροχερών ημερών κάθε μήνα. Να υπολογισθούν:
α) Με function, ο μέσος όρος των βροχερών ημερών
β) Με procedure, ποιος μήνας ήταν ο πιο βροχερός και πόσες μέρες του μήνα αυτού έβρεχε.
Όλες οι εκτυπώσεις να γίνονται στο κυρίως πρόγραμμα.

128. Η μέτρηση της μόλυνσης της ατμόσφαιρας γίνεται σε κλίμακα από 0 έως 100. Δίνονται μετρήσεις για Ν συνεχείς μέρες. Να γραφεί πρόγραμμα το οποίο θα προσδιορίζει το συνολικό πλήθος των ημερών αιχμής δηλ. των ημερών που η μόλυνση ήταν μεγαλύτερη από την προηγούμενη και την επόμενη μέρα. Επίσης να τυπώνονται οι μέρες αιχμής (π.χ. 3η, 7η κ.τ.λ.)

129. Δίνεται μία ημερομηνία (τύπου record με πεδία: ημέρα, μήνας και έτος σαν τετραψήφιος αριθμός). Να γραφεί procedure που θα εμφανίζει την ημερομηνία στη μορφή **/**/**

7. Σύνολα

Τα σύνολα είναι συλλογές αντικειμένων ίδιου τύπου που ονομάζονται στοιχεία του συνόλου. Κάθε σύνολο μπορεί να περιέχει μέχρι 256 στοιχεία.

Δήλωση συνόλου

```
Type
    <τύπος συνόλου>=set of <τύπος στοιχείων>;
Var
    <όνομα συνόλου>:<τύπος συνόλου>
```

Ο <τύπος στοιχείων> μπορεί να είναι ένας από τους γνωστούς τύπους ή απαριθμητός ή τύπος υποπεριοχής τιμών. Τα σύνολα με στοιχεία απαριθμητού τύπου δεν θα μας απασχολήσουν.

Παράδειγμα

```
Type
    one_digit=set of 0..9;
    set_char=set of char;
    alphabet=set of 'A'..'Z';
```

Συμμετοχή σε σύνολο **<μεταβλητή> in <σύνολο>**

Μη συμμετοχή στο σύνολο **not (<μεταβλητή> in <σύνολο>)**

Το σύνολο μπορεί να δηλωθεί και να πάρει τιμές με := ή να γραφεί απευθείας στην εντολή που χρησιμοποιεί τη συμμετοχή μεταβλητής σε σύνολο.

Παράδειγμα

1ος τρόπος (επώνυμο σύνολο)

```
var
    grades:set of char;
    grade:char;
begin
    grades:=['A','B','C'];
    write('Χαρακτ. βαθμολ: ');
    grade:=readkey;
    if grade in grades then
    writeln('Ο μαθητής πέτυχε');
end;
```

2ος τρόπος (ανώνυμο σύνολο)

```
var
    grade:char;
begin
    write('Χαρακτ. βαθμολ: ');
    grade:=readkey;
    if grade in ['A','B','C'] then
    writeln('Ο μαθητής πέτυχε');
end;
```

Παράδειγμα

Να δοθούν 3 σύνολα: το σύνολο all που θα περιέχει τους μονοψήφιους αριθμούς από το 1 μέχρι το 9, το σύνολο odd που θα περιέχει τους περιττούς μονοψήφιους και το σύνολο even που θα περιέχει τους άρτιους μονοψήφιους αριθμούς. Στη συνέχεια να δίνεται ένας αριθμός. Αν δεν είναι μονοψήφιος από το 1 μέχρι το 9, το πρόγραμμα να σταματά διαφορετικά να τυπώνεται η λέξη 'ΠΕΡΙΤΤΟΣ' αν ανήκει στο σύνολο odd και η λέξη 'ΑΡΤΙΟΣ' αν ανήκει στο σύνολο even.

```
program odd_even;
uses crt;
type
    one_digit=set of 1..9;
var
    i:byte;
    all,odd,even:one_digit;

begin
     clrscr;
     all:=[1..9];              { Το  all περιέχει συνεχόμενα στοιχεία }
                               { γι' αυτό χρησιμοποιήθηκαν .. }
     odd:=[1,3,5,7,9];         { το  odd και το even δεν περιέχουν }
     even:=[2,4,6,8];          { συνεχόμενα στοιχεία γι' αυτό }
     write('Αριθμός: ');       { χωρίστηκαν με κόμμα }
     readln(i);
     if not(i in all) then halt;
     if i in odd then writeln(' ΠΕΡΙΤΤΟΣ ') else writeln('ΑΡΤΙΟΣ');
     readln;
end.
```

Παραδείγματα χρήσης ανώνυμων συνόλων

Έστω ότι η μεταβλητή any είναι τύπου char.

- if any in['N','n','N','ν'] then writeln('Η απάντησή σου είναι ΝΑΙ');
- if any in['O','o','O','o'] then writeln('Η απάντησή σου είναι ΟΧΙ);
- if not (any in['N','n','N','ν','O','o','O','o']) then
  writeln('Η απάντησή σου δεν είναι σωστή');

8. Αρχεία καθορισμένου τύπου

8.1 Δήλωση αρχείου

```
type
    <τύπος αρχείου>= file of <τύπος>;
var
    <όνομα αρχείου>: <τύπος αρχείου>;
```

Παράδειγμα

```
type
    intfile=file of integer;

    obj=record
        no:word;
        description:string;
        price:longint;
    end;
    objfile=file of obj;
```

8.2. Σύνδεση με το DOS

Πριν ανοίξουμε ένα αρχείο, πρέπει να συνδέσουμε πρώτα το όνομά του με το όνομα ενός αρχείου στο DOS.

```
assign(<όνομα αρχείου>,'<όνομα>.<επέκταση>');
```

Το <όνομα>.<επέκταση> πρέπει να είναι σύμφωνο με τους κανόνες ονοματολογίας αρχείων στο DOS (το όνομα μέχρι 8 λατινικοί χαρακτήρες, η επέκταση μέχρι 3 λατινικοί χαρακτήρες, χωρίς ειδικά σύμβολα και κενά). Συνηθίζουμε να χρησιμοποιούμε την επέκταση DAT για να δηλώνουμε ότι πρόκειται για αρχείο δεδομένων.

Αν δεν γίνει η σύνδεση εμφανίζεται το μήνυμα λάθους:
File not assigned.

8.3. Άνοιγμα αρχείου

α) Με διαγραφή

```
rewrite(<όνομα αρχείου>);
```

Αν υπάρχει το αρχείο, το διαγράφει και το καθιστά έτοιμο για εγγραφή. (Ο νοητός δείκτης του αρχείου τοποθετείται στην αρχή του).

β) Άνοιγμα υπάρχοντος αρχείου

```
reset(<όνομα αρχείου>);
```

Τοποθετείται ο νοητός δείκτης του αρχείου στην αρχή του. Η reset ανοίγει το αρχείο και για ανάγνωση και για εγγραφή. **πρέπει όμως το αρχείο να υπάρχει.** Αν το αρχείο δεν υπάρχει εμφανίζεται το μήνυμα λάθους: File not found.

Μπορούμε να χρησιμοποιήσουμε τη reset για ένα αρχείο που δεν υπάρχει και να μην εμφανισθεί μήνυμα λάθους αν χρησιμοποιήσουμε το διακόπτη {$I-}. Με το διακόπτη {$I-} απενεργοποιούμε τον έλεγχο λαθών κατά το χειρισμό αρχείων κι έτσι σε περίπτωση προβλήματος, δεν διακόπτεται η εκτέλεση του προγράμματος και η συνάρτηση IOresult της Pascal παίρνει τιμή διάφορη του 0. Έτσι, ελέγχοντας την τιμή της IOresult, αν είναι διάφορη του 0, διορθώνουμε το λάθος που προέκυψε μέσα από τον κώδικα του προγράμματος χωρίς να διακοπεί η εκτέλεσή του. Η ενεργοποίηση του ελέγχου λαθών γίνεται με το διακόπτη {$I+}.

Παράδειγμα 1

Έστω ότι επιχειρούμε να ανοίξουμε ένα αρχείο και να διαβάσουμε κάποια εγγραφή ενώ το αρχείο δεν υπάρχει το δίσκο.

```
Type
        stud=record
            no:word;        (* Κωδικός μαθητή *)
            name:string;    (* Ονοματεπώνυμο *)
            mark:real;      (* Βαθμός *)
    end;
    studfile=file of stud;
var
    students:studfile;
    studrec:stud;
.....
    assign(students,'students.dat');
    {$i-}
```

```
        reset(students);
    {$i+}
    if ioresult<>0 then exit;
.....
```

Παράδειγμα 2

Έστω ότι επιθυμούμε να ανοίξουμε ένα αρχείο για εγγραφή. Αν το αρχείο υπάρχει η εγγραφή γίνεται κανονικά. Αν δεν υπάρχει, πρέπει να ανοιχθεί με την εντολή rewrite.

```
.....
    assign(students,'students.dat');
    {$i-}
        reset(students);
    {$i+}
    if ioresult<>0 then rewrite(students);
.....
```

8.4. Κλείσιμο αρχείου

```
close(<όνομα αρχείου>);
```

Οι θέσεις του αρχείου αριθμούνται αρχίζοντας την αρίθμηση από το 0. Συνηθίζουμε να μην γράφουμε εγγραφές στη θέση 0 ειδικά όταν χρησιμοποιούμε κωδικούς που θέλουμε να ταυτίζονται με τη θέση των εγγραφών μέσα στο αρχείο.

Το αρχείο του παραδείγματος 1:

Θέση	no	name	mark
0			
1	1	Αντωνίου Κ.	17.5
2	2	Βασιλειάδης Γ.	18
3	3	Πέτρου Χ.	12.5
4	4	Γεωργίου Μ.	19

Στο παραπάνω αρχείο, ο κωδικός (no) του μαθητή ταυτίζεται με τη θέση του μέσα στο αρχείο.
Το αρχείο τυπικά έχει 5 εγγραφές (συμπεριλαμβανομένης και της εγγραφής 0), ενώ στην ουσία έχει 4.

8.5. Ανάγνωση εγγραφής από το αρχείο

```
read(<όνομα αρχείου>,<μεταβλητή>);
```

Διαβάζεται από το αρχείο η εγγραφή στην οποία βρίσκεται ο νοητός δείκτης, η τιμή της τοποθετείται στη <μεταβλητή> και ο νοητός δείκτης του αρχείου μεταφέρεται αυτόματα στην επόμενη εγγραφή (π.χ. μία επόμενη read θα διάβαζε την επόμενη εγγραφή του αρχείου).

Προσοχή! Η <μεταβλητή> παίρνει τιμή από το αρχείο.

8.6. Εγγραφή στο αρχείο

```
write(<όνομα αρχείου>,<μεταβλητή>);
```

Εγγράφεται μέσα στο αρχείο, στη θέση που βρίσκεται ο νοητός δείκτης, η τιμή της <μεταβλητής>. Αν στη θέση που γίνεται η εγγραφή υπάρχουν πληροφορίες, χάνονται. Μετά την εγγραφή, ο νοητός δείκτης του αρχείου μεταφέρεται αυτόματα στην επόμενη θέση.

Προσοχή! Η <μεταβλητή> δίνει τιμή που αποθηκεύεται στο αρχείο.

8.7. Η συνάρτηση eof

```
eof(<όνομα αρχείου>);
```

Η συνάρτηση eof επιστρέφει την τιμή true αν ο δείκτης του αρχείου βρίσκεται μετά την τελευταία εγγραφή του αρχείου (δηλ. στο τέλος του) ή την τιμή false αν ο δείκτης δεν βρίσκεται στο τέλος του αρχείου. Χρησιμοποιείται κυρίως για σειριακή ανάγνωση των εγγραφών. Συνήθως χρησιμοποιείται η μορφή not eof(<όνομα αρχείου>) η οποία έχει την τιμή true όταν δεν έχει τελειώσει το αρχείο και την τιμή false όταν ο δείκτης βρίσκεται στο τέλος του.

Παράδειγμα

Το παρακάτω τμήμα του προγράμματος, διαβάζει όλες τις εγγραφές του αρχείου και τυπώνει για κάθε μαθητή το ονοματεπώνυμό του (name) και το βαθμό του (mark). Οι δηλώσεις των μεταβλητών είναι αυτές του παραδείγματος 1 στην εντολή reset.

```
.....
    assign(students,'students.dat');
```

```
{$i-}
   reset(students);
{$i+}
   if ioresult<>0 then exit;
   while not eof(students) do
   begin
      read(students,studrec);
      with studrec do writeln(name,'   ',mark);
      if wherey=23 then
      begin
         readln;
         clrscr;
      end;
   end; (* while *)
.....
```

8.8. Η συνάρτηση filesize

filesize(<όνομα αρχείου>);

Η συνάρτηση filesize επιστρέφει το μέγεθος του αρχείου. Στο σχήμα του αρχείου του παραδείγματος 1 η filesize(students) θα είχε την τιμή 5 (συμπεριλαμβάνεται και η θέση 0). Άρα για το συγκεκριμένο αρχείο, οι πραγματικές εγγραφές είναι filesize(students)-1. Η τελευταία θέση είναι επίσης filesize(students)-1 ενώ η επόμενη της τελευταίας (δηλ. εκεί όπου μπορούμε να γράψουμε μία νέα εγγραφή) είναι η filesize(students).

8.9. Τοποθέτηση του δείκτη του αρχείου σε συγκεκριμένη θέση

seek(<όνομα αρχείου>,<θέση>);

Μεταφέρει το δείκτη του αρχείου στη <θέση>. **προσέξτε ότι η αρίθμηση των θέσεων ξεκινά από το 0.**
Αν ακολουθεί read η <θέση> μπορεί να είναι μέχρι η τελευταία θέση του αρχείου αλλιώς εμφανίζεται το μήνυμα Disk read error. Αν ακολουθεί write η <θέση> μπορεί να είναι μέχρι η **επόμενη** της τελευταίας θέσης του αρχείου (π.χ. για να γραφεί εκεί μία νέα εγγραφή).

8.10. Εύρεση της τρέχουσας θέσης του δείκτη του αρχείου

filepos(<όνομα αρχείου>);

Επιστρέφει την τρέχουσα θέση του δείκτη του αρχείου. Μετά από read ή write, το filepos αυξάνεται αυτόματα κατά 1.

Εφαρμογή 1

```
program phone_book;        { το πρόγραμμα αυτό εισάγει εγγραφές }
uses crt;                  { στο τέλος του αρχείου. Δεν ελέγχει }
type                       { αν υπάρχουν ενδιάμεσες κενές θέσεις }
    atomo=record           { που μπορεί να προέκυψαν από διαγραφή }
       epon:string[20];
       onom:string[20];
       phone:string[15];
    end;
    filetype=file of atomo;
var
  at:atomo; f:filetype;
  ch:char;

procedure new_atomo;
begin
    clrscr;
    {$i-} reset(f); {$i+}
    if ioresult<>0 then
    begin
        with at do
        begin
           epon:=' '; onom:=' '; phone:=' ';
        end;
        rewrite(f);
        write(f,at);    (* Γεμίζει τη 0 θέση με κενή εγγραφή *)
    end;
    with at do
    begin
       write('Επώνυμο  :');readln(epon);
       write('Ονομα    :');readln(onom);
       write('Τηλέφωνο :');readln(phone);
    end;
    seek(f,filesize(f)); (*Μεταφέρει το δείκτη στο τέλος του αρχείου*)
    write(f,at);
    close(f);
end;
```

```
procedure all_file;
var c:char;
begin
    {$i-} reset(f); {$i+}
    if ioresult<>0 then exit;
    clrscr;
    while not eof(f) do
    begin
        read(f,at);
        with at do
            if onom<>' ' then writeln(epon:20,onom:20,phone:15);
            {Για να μην εμφανίζονται οι κενές εγγραφές που
             προέκυψαν από διαγραφή}
        if wherey=23 then
        begin
            c:=readkey;
            clrscr;
        end;
    end;
    close(f);
    READLN;
end;

procedure emf_atomo;
var zep:string[20];
begin
    {$i-} reset(f); {$i+}
    if ioresult<>0 then exit;
    clrscr; write('Επώνυμο: ');readln(zep);
    while not eof(f) do
    begin
        read(f,at);
        with at do if epon=zep then writeln(onom:20,phone:15);
    end;
    close(f); readln;
end;

procedure del_atomo;
var zep:string[20];
    d:char;
begin
    {$i-} reset(f); {$i+}
    if ioresult<>0 then exit;
    clrscr;  write('Επώνυμο: ');readln(zep);
```

```
    while not eof(f) do
    begin
        read(f,at);      {Μετά την εκτέλεση της read, ο δείκτης του αρχείου
                          μεταφέρεται αυτόματα στην επόμενη εγγραφή}
        with at do
        if epon=zep then
        begin
            write(epon:20,onom:20,phone:15,'  Διαγραφή<N/O> ');
            d:=readkey;
            if d in ['N','n','ν','Ν'] then
            begin
                writeln('N');
                epon:=' '; onom:=' '; phone:=' ';
                seek(f,filepos(f)-1);
                  {Με τη seek γυρνάει ο δείκτης του αρχείου
                   μια θέση πίσω γιατί η προηγούμενη read τον είχε
                   μεταφέρει μια θέση μπροστά}
                write(f,at);
                 {Μετά την εκτέλεση της write, ο δείκτης του
                  αρχείου μεταφέρεται αυτόματα στην επόμενη εγγραφή
                  την οποία διαβάζει η read που ακολουθεί}
            end else writeln('O');
        end; {if epon=zep}
    end; {while}
    close(f);
end;

begin   {main}
  assign(f,'phones.dat');
  REPEAT
    CLRSCR;
    writeln('0. τΕΛΟΣ');
    writeln('1. Εισαγωγή εγγραφής');
    writeln('2. Εμφάνιση όλου του αρχείου');
    writeln('3. Εμφάνιση εγγραφής');
    writeln('4. Διαγραφή εγγραφής');
    write('ποια η επιλογή σου; ');ch:=readkey;
    case ch of
      '1': new_atomo;
      '2': all_file;
      '3': emf_atomo;
      '4': del_atomo;
    end;
  until ch='0';
end.
```

Εφαρμογή 2

```
program files;          { Το πρόγραμμα αυτό ελέγχει  αν  }
                        { υπάρχουν ενδιάμεσες κενές θέσεις που }
                        { μπορεί να προέκυψαν από διαγραφή }
uses crt,printer;
type
  r=record
    c:integer;
       (* Ο κωδικός της εγγραφής ταυτίζεται με τη θέση στο αρχείο *)
    n:string;
    t:longint;
  end;
  arxeio=file of r;
var
  rec:r; recs:arxeio;
  i:integer; telos,ok:boolean; epil,any:char;

procedure message;
begin
     gotoxy(30,23); write('πατήστε οποιοδήποτε πλήκτρο για συνέχεια...');
     any:=readkey;
end;  {procedure message}

procedure openfile(var f:arxeio);
begin
      clrscr;
{$i-}
      reset(f);
{$i+}
      if ioresult<>0 then
      begin
          rewrite(f);
          with rec do
          begin
              c:=0; n:=''; t:=0;
          end;
          write(f,rec); close(f);
          reset(f);
      end;
      READ(F,REC); (* για να κάνει skip τη μηδενική θέση *)
end;   {procedure openfile}
```

```
procedure newcustomer(var f:arxeio);
label 1;
var
  i:integer;
begin
     openfile(f);
     with rec do
     begin
       for i:=1 to filesize(f)-1 do
       begin                          (*  εδώ ελέγχεται αν υπάρχει  *)
            read(f,rec);              (*  κενή θέση στο αρχείο η    *)
            if i<>c   then            (*  οποία θα είχε προκύψει    *)
            begin                     (*  από διαγραφή εγγραφής     *)
                 c:=i  ;              (*  i : η θέση στο αρχείο     *)
                 goto 1;              (*  c : ο κωδικός             *)
            end;
       end;
       c:=filesize(f);
       (*  Η νέα εγγραφή παίρνει κωδικό το μέγεθος του αρχείου *)
1:      write('Κωδικός  : ');writeln(c);
       write('Ονομ/μο  : ');readln(n);
       write('Υπόλοιπο : ');readln(t);
     end;
     seek(f,rec.c); (* για να μεταφερθεί ο δείκτης στο τέλος *)
     write(f,rec);
     close(f);
     writeln; writeln('Η εγγραφή έχει γίνει'); message;
end;   {procedure newcustomer}

procedure displaycust(var f:arxeio;var ok:boolean);
var
   zk:integer;
begin
     openfile(f);
     clrscr;
     repeat
       gotoxy(20,5);
       writeln('                    ');
       gotoxy(20,5);
       write('Κωδικός  : ');
        readln(zk);
     until (zk<filesize(f)) and (zk>0);
     seek(f,zk);
     read(f,rec);
     if zk=rec.c then
```

```
    begin
      with rec do
      begin
         ok:=true;
         gotoxy(20,6);  write('Ονομ/μο  : ');writeln(n);
         gotoxy(20,7);  write('Υπόλοιπο : ');writeln(t);
      end;   {with}
    end
    else   writeln('Ανύπαρκτος κωδικός');
    message;
end;  {procedure displaycust}

procedure displayname(var f:arxeio);
var
  zn:string;
  found:boolean;
begin
    openfile(f);
    clrscr;
    found:=false;
    write('Ονομ/μο  : ');readln(zn);
    WHILE NOT EOF(f) do
    begin
        read(f,rec);
        if zn=rec.n then
        begin
           found:=true;
           with rec do
           begin
              clrscr;
              gotoxy(20,5);   writeln('Κωδικός  : ',c);
              gotoxy(20,6);   writeln('Ονομ/μο  : ',n);
              gotoxy(20,7);   writeln('Υπόλοιπο : ',t);
              message;
           end;  {with}
        end;
    end;   {while}
    if found=false then
    begin
       writeln('Ανύπαρκτο ονοματεπώνυμο'); message;
    end;
    close(f);
end;  {procedure displayname}
```

```
procedure update(var f:arxeio);
var
  ok:boolean;
begin
    ok:=false;
    displaycust(f,ok);
    if ok=false then exit;
    repeat
        gotoxy(30,20);
        textcolor(black);
        textbackground(white);
        writeln('<O>');
        gotoxy(30,21);
        writeln('<Y>');
        textcolor(white);
        textbackground(black);
        gotoxy(33,20);
        writeln('νοματεπώνυμο');
        gotoxy(33,21);
        writeln('πόλοιπο');
        any:=readkey;
    until any in['O','o','O','o','Y','υ','Y','y'];
    if any in['O','o','O','o'] then
    begin
        with rec do
        begin
           gotoxy(31,6);writeln('                              ');
           gotoxy(31,6);readln(n);
        end;
    end;
    if any in['Y','υ','Y','y'] then
    begin
        with rec do
        begin
           gotoxy(31,7);writeln('                              ');
           gotoxy(31,7);readln(t);
        end;
    end;
    seek(f,rec.c); write(f,rec);
    close(f);
    message;
end;   {procedure update}

procedure deletecust(var f:arxeio);
var
```

```
  ok:boolean;
  position:integer;
begin
    ok:=false;
    displaycust(f,ok);
    if ok=false then exit;
    writeln; write('Επιβεβαίωση διαγραφής <N>'); any:=readkey;
    if any in['N','n','N','ν'] then
    begin
        with rec do
        begin
            position:=c;
            c:=0; n:=''; t:=0;
         end;
    end
    else
    begin
        close(f); exit;
    end;
    seek(f,position); write(f,rec);
    close(f);
end;   {procedure deletecust}

procedure displayall(var f:arxeio);
var
  max:1..22;
  m:integer;
begin
     clrscr;
     write('Αριθμός εγγραφών ανά οθόνη [1-22] ');
     readln(max); m:=0;
     openfile(f);
     WHILE NOT EOF(f) do
     begin
          read(f,rec);
          if rec.c<>0 then
                  (*   ΓΙΑ ΝΑ ΑΓΝΟΟΥΝτΑΙ ΟΙ ΕΓΓΡΑΦΕΣ
               πΟΥ ΕΧΟΥΝ ΔΙΑΓΡΑΦΕΙ δηλ. ΚΩΔΙΚΟΣ=0  *)
          begin
             writeln(rec.c:10,rec.n:30,rec.t:15);
             m:=m+1;
          end;
         if m=max then
          begin
            m:=0;
```

```
            message;
            clrscr;
          end;
      end;
      message;
      close(f);
end;  {procedure displayall}

begin
  assign(RECS,'random.dat');
  epil:=' ';
  telos:=false;
  while telos=false do
  begin
     clrscr;
     writeln('0. τέλος προγράμματος');
     writeln('1. προσθήκη νέων εγγραφών');
     writeln('2. Εμφάνιση εγγραφών από κωδικό');
     writeln('3. Εμφάνιση εγγραφών από επώνυμο');
     writeln('4. Διόρθωση πεδίων εγγραφής');
     writeln('5. Διαγραφή εγγραφής');
     writeln('6. Εμφάνιση όλων των εγγραφών');
     writeln;writeln('ποια η επιλογή σας  ');
     epil:=readkey;
     case epil of
         '1':newcustomer(recs);
         '2':BEGIN
                displaycust(recs,ok);
                close(recs);
                (* το αρχείο κλείνει εδώ γιατί η procedure
                χρησιμοποιείται και από τις update & deletecust *)
             END;
         '3':displayname(recs);
         '4':update(recs);
         '5':deletecust(recs);
         '6':displayall(recs);
         '0':telos:=true;
     end; {case}
  end;  {while}
end.
```

Ασκήσεις

130. Να δημιουργηθεί ένα αρχείο που θα περιέχει τους βαθμούς ενός μαθητή σε Ν μαθήματα.

131. Να διαβαστούν οι βαθμοί από το αρχείο της προηγούμενης άσκησης και να υπολογισθεί ο μέσος όρος τους π.χ. 18 και 5/12

132. Να διαβαστούν οι βαθμοί από το αρχείο της προηγούμενης άσκησης και να γραφούν σε νέο αρχείο μόνο όσοι είναι μεγαλύτεροι του 15.

133. Να δημιουργηθεί αρχείο στο οποίο για κάθε μαθητή κρατούνται: Κωδικός (ταυτίζεται με τη θέση μέσα στο αρχείο), ονοματεπώνυμο και βαθμός. Να εισαχθούν στοιχεία για διάφορους μαθητές (μέχρι ο χρήστης να επιλέξει τέλος εισαγωγής) και στη συνέχεια να εμφανισθεί όλο το αρχείο στην οθόνη. τέλος να εμφανισθεί στην οθόνη το ονοματεπώνυμο του μαθητή με το μεγαλύτερο βαθμό.

134. Να δημιουργηθεί αρχείο στο οποίο για κάθε μαθητή κρατούνται: Κωδικός (ταυτίζεται με τη θέση μέσα στο αρχείο), ονοματεπώνυμο και Ν βαθμοί. Να εισαχθούν στοιχεία για διάφορους μαθητές (μέχρι ο χρήστης να επιλέξει τέλος εισαγωγής) και στη συνέχεια να εμφανισθούν στην οθόνη τα ονοματεπώνυμα και οι μέσοι όροι βαθμολογίας όλων των μαθητών.

135. Να δημιουργηθεί αρχείο επιταγών. Για κάθε επιταγή κρατούνται: Αύξων αριθμός (να προτείνεται από το πρόγραμμα), αριθμός επιταγής, εκδότης, ημερομηνία λήξης και ποσό. Στη συνέχεια να εμφανισθούν στην οθόνη οι επιταγές που έληξαν (αυτές που η ημερομηνία λήξης τους είναι μικρότερη από τη σημερινή).

Οι διαδικασίες getdate και gettime της unit dos

Η σημερινή ημερομηνία μπορεί να δοθεί από το χρήστη ή να εισαχθεί αυτόματα από το σύστημα (αν είναι η σωστή). Στη δεύτερη περίπτωση, πρέπει να συμπεριλάβουμε στο πρόγραμμά μας τη μονάδα dos δηλ. να γράψουμε **uses dos;** Στη συνέχεια πρέπει να δηλώσουμε 4 μεταβλητές τύπου word ή καλύτερα μία μεταβλητή τύπου record με 4 πεδία τύπου word π.χ. year, month, day, dow όπου η year συμβολίζει το έτος, η month το μήνα, η day την ημέρα και η dow τη μέρα της εβδομάδας (η Δευτέρα είναι ο αριθμός 1 κλπ). Η διαδικασία **getdate**(year,month,day,dow) θα επιστρέψει στις μεταβλητές τις ανάλογες τιμές. Ανάλογα μπορούμε να εισάγουμε αυτόματα από το σύστημα και την ώρα με τη διαδικασία **gettime**(hour,minutes,seconds,sec100) όπου όλες οι παράμετροι είναι τύπου word και συμβολίζουν αντίστοιχα την ώρα, τα λεπτά, τα δευτερόλεπτα και τα εκατοστά του δευτερολέπτου.

9. Χρήσιμες εφαρμογές

9.1. Μενού με βελάκια

```
program arrows;
uses crt;

var ch,any:char;
   i:integer;
   menu:array[1..7] of string[10];

begin
 repeat
    cursoroff;
    clrscr;
    menu[1]:='  PASCAL ';
    menu[2]:='  BASIC  ';
    menu[3]:='  COBOL  ';
    menu[4]:=' FORTRAN ';
    menu[5]:='    C    ';
    menu[6]:='  DBASE  ';
    menu[7]:='  ΤΕΛΟΣ  ';
    textcolor(black); textbackground(white);
    gotoxy(17,13);write('ΠΑΡΑΚΑΛΩ ΕΠΙΛΕΞΤΕ ');
    textcolor(white); textbackground(black);
    for i:=1 to 7 do
    begin
        gotoxy(21,i+4);writeln(menu[i]);
    end;
    textcolor(black); textbackground(white);
    gotoxy(21,5);write(menu[1]);
    i:=1;
    ch:=readkey; (* 1ος κωδικός του extended key *)
    while ch=#0 do
    begin
     ch:=readkey;  (* 2ος κωδικός του extended key *)
     case ch  of
     #72 :begin
          textcolor(white);textbackground(black);
          gotoxy(21,i+4); write(menu[i]);
          textcolor(black);textbackground(white);
          if i=1 then i:=7 else i:=i-1;gotoxy(21,i+4);write(menu[i]);
         end;
     #80:begin
```

```
          textcolor(white);textbackground(black);
          gotoxy(21,i+4); write(menu[i]);
          textcolor(black);textbackground(white);
          if i=7 then i:=1 else i:=i+1;gotoxy(21,i+4);write(menu[i]);
        end;
      end;   (*CASE*)
      ch:=readkey;  (* 1ος κωδικός του extended key *)
    end;   (*WHILE*)
    textcolor(white);textbackground(black);
    case i of 1,2,3,4,5,6,7:begin
                              clrscr;  gotoxy(21,7);write(menu[i]);
                            end;
    end;
    any:=readkey;
  until i=7;
end.
```

9.2. Εμφάνιση / εξαφάνιση του δρομέα

Για να εξαφανίσουμε/εμφανίσουμε τον δρομέα, πρέπει στο πρόγραμμά μας να χρησιμοποιήσουμε τη unit dos δηλ. κάτω από το program να γράψουμε: **uses crt,dos;** Για να εξαφανίσουμε τον δρομέα καλούμε την procedure **cursoroff** ενώ για να τον εμφανίσουμε την **cursoron**.
Οι δύο αυτές procedures καθώς και η sizecursor που καλείται απ' τις δύο προηγούμενες πρέπει να γραφούν στο πρόγραμμά μας.

```
procedure sizecursor(scantop,scanbot:byte);
var register:registers;
begin
   with register do
    begin
        ax:=1 shl 8;   cx:=scantop shl 8+scanbot;  intr($10,register);
    end;
end;
procedure cursoron;
begin
    if mem[$0000:$0449]=7 then sizecursor(12,13)
                          else sizecursor(6,7);
end;
procedure cursoroff;
begin
   sizecursor(14,0);
end;
```

ISBN: 978-618-00-4772-1

9 786180 047721