ΚΕΦΑΛΑΙΟ 8
Η ΓΛΩΣΣΑ PASCAL

Σκοπός κεφαλαίου:

Να γνωρίσουμε τις αρχές και τις δυνατότητες της γλώσσας Pascal.

Ειδικοί σκοποί:

- Να γνωρίσουμε τη χρησιμότητα της γλώσσας στη διδασκαλία του προγραμματισμού.
- Να γνωρίσουμε τις δυνατότητές της για την υλοποίηση του δομημένου προγραμματισμού.
- Να γνωρίσουμε τη δομή ενός προγράμματος σε γλώσσα Pascal.
- Να γνωρίσουμε τις ευκολίες που παρέχει για τη χρησιμοποίηση και τη δημιουργία προγραμμάτων.

8.1. ΕΙΣΑΓΩΓΗ

Η γλώσσα PASCAL σχεδιάστηκε από τον Nicklaus Wirth, διάσημο Ελβετό επιστήμονα της Πληροφορικής, το 1968 και αναθεωρήθηκε το 1972.Ο Wirth σχεδίασε την PASCAL προκειμένου να ξεπεραστούν τα μειονεκτήματα των γλωσσών προγραμματισμού της δεκαετίας του 1960. Πήρε το όνομά της προς τιμή του μαθηματικού και φιλοσόφου Blaise Pascal.

Η PASCAL σχεδιάστηκε με στόχο να χρησιμοποιηθεί ως διδακτικό εργαλείο των αρχών του προγραμματισμού. Λόγω όμως της πληρότητάς της, της απλότητας και της ευκολίας στην εκμάθησή της χρησιμοποιείται ευρέως στις επιχειρήσεις, τη βιομηχανία και τους προσωπικούς υπολογιστές.

Η PASCAL είναι γλώσσα γενικής χρήσης και υποστηρίζει τις αρχές του δομημένου και του τμηματικού προγραμματισμού. Μερικά από τα ιδιαίτερα χαρακτηριστικά της είναι τα εξής:

- Η δυνατότητα που δίνεται στον προγραμματιστή να δημιουργεί δικούς του τύπους δεδομένων.
- Η χρήση μεταβλητών τύπου **δείκτη (pointer)** και η δυνατότητα της δυναμικής διαχείρισης της κεντρικής μνήμης.
- Η **σύνθετη εντολή (compound statement)**, δηλαδή η χρήση μιας σειράς εντολών ως μία εντολή.

Γλώσσα προγραμματισμού
Είναι ένα σύνολο κανόνων, συμβόλων και ειδικών λέξεων που χρησιμοποιούνται για τη δημιουργία ενός προγράμματος.

Όπως οι περισσότερες γλώσσες, έτσι και αυτή, πέρασε από πολλές εκδόσεις διαφόρων κατασκευαστών που κάθε μια εμπλουτιζόταν με περισσότερες δυνατότητες. Αρχικά είχε μία αδυναμία στον αποτελεσματικό χειρισμό των αρχείων και των αλφαριθμητικών δεδομένων ή **συμβολοσειρών** (**strings**) που όμως έχει ήδη ξεπεραστεί σε μεταγενέστερες εκδόσεις. Η έκδοσή της για περιβάλλον Windows θεωρείται ως πρότυπο στο χώρο των μικροϋπολογιστών.

Το **αλφάβητο** της γλώσσας Pascal, αποτελείται από βασικά σύμβολα, όπως γράμματα του Ελληνολατινικού αλφαβήτου, τα αριθμητικά ψηφία (0 - 9) και τα ειδικά σύμβολα, όπως +, -, *, /, \, ., ; κλπ.

Συντακτικό(Syntax)
Είναι ένα σύνολο τυπικών κανόνων οι οποίοι προσδιορίζουν πώς γράφονται έγκυρες εντολές σε μία γλώσσα προγραμματισμού.

Η Pascal μας επιτρέπει να δίνουμε ταυτότητες ή **ονόματα** (**identifiers**) τα οποία αναφέρονται σε σταθερές, μεταβλητές, τύπους δεδομένων, διαδικασίες, συναρτήσεις κλπ. Ένα **όνομα** αποτελείται από μια σειρά χαρακτήρων (**γράμματα** του λατινικού αλφαβήτου., **αριθμοί** ή _), πρέπει να αρχίζει πάντοτε με γράμμα και δεν πρέπει να περιέχει κενά.

Σημασιολογία (Semantics)
Είναι ένα σύνολο κανόνων οι οποίοι προσδιορίζουν τη σημασία των εντολών που γράφονται σε μία γλώσσα προγραμματισμού.

Προκειμένου να εξασφαλίσουμε ένα ευανάγνωστο πρόγραμμα, δίνουμε ονόματα ενδεικτικά του περιεχομένου τους (meaningful) και αντί κενού βάζουμε τον χαρακτήρα "_". Ο χαρακτήρας αυτός δεν μπορεί να είναι ο τελευταίος του ονόματος. Πχ. basikos_mistos, kratiseis, pl_poso, είναι σωστά ονόματα.

Πολλές φορές για λόγους **τεκμηρίωσης** χρειάζεται να γράφουμε **σχόλια** στο πρόγραμμα. Τα σχόλια μπορεί να καταλαμβάνουν όσες γραμμές θέλουμε ή να εμφανίζονται μεταξύ των στοιχείων μιας εντολής. Τα σχόλια περιέχονται μεταξύ δύο παρενθέσεων της μορφής **{...} ή (*...*)**. Κάθε σχόλιο θεωρείται ως ένα κενό.

8.2. ΒΑΣΙΚΟΙ ΤΥΠΟΙ ΔΕΔΟΜΕΝΩΝ

Τύπος Δεδομένων (Data Type)
Είναι μια κατηγορία δεδομένων με ορισμένη απεικόνιση και ένα σύνολο λειτουργιών που μπορούν να εφαρμοστούν στο σύνολο των τιμών τους.

Ένα πρόγραμμα επεξεργάζεται δεδομένα τα οποία μπορεί να είναι αποθηκευμένα εσωτερικά στη μνήμη ή εξωτερικά στο δίσκο ή τη δισκέτα ή να γίνεται η εισαγωγή τους από το πληκτρολόγιο ή το σαρωτή κλπ. Στην Pascal κάθε δεδομένο πρέπει να είναι ορισμένου τύπου. Οι **τύποι δεδομένων** προσδιορίζουν τον τρόπο παράστασης των δεδομένων εσωτερικά στον υπολογιστή καθώς και το είδος της επεξεργασίας τους από τον υπολογιστή. Μερικοί τύποι δεδομένων χρησιμοποιούνται τόσο συχνά που η Pascal τους έχει προσδιορίσει και έτσι υπάρχουν έτοιμοι για το χρήστη. Στον προγραμματισμό όμως, ο προγραμματιστής προσδιορίζει και άλλους τύπους δεδομένων ανάλογα με το είδος της επεξεργασίας.

Οι τύποι δεδομένων χρησιμοποιούνται για τον ορισμό των μεταβλητών ή των συναρτήσεων που ορίζει ο χρήστης. Μία μεταβλητή θα είναι πάντοτε ενός συγκεκριμένου τύπου. Εάν προσπαθήσουμε να δώσουμε τιμή εκτός

των ορίων του τύπου της τότε τα αποτελέσματα είναι απρόβλεπτα. Υπάρχουν απλοί ή στοιχειώδεις και σύνθετοι τύποι δεδομένων.

Οι προσδιορισμένοι από την Pascal **απλοί** ή στοιχειώδεις τύποι δεδομένων είναι οι:

- Ο ακέραιος τύπος.
- Ο πραγματικός τύπος.
- Ο λογικός τύπος.
- Ο χαρακτήρας.

Σύνθετοι τύποι δεδομένων είναι αυτοί που ορίζονται από απλούς τύπους ή και από άλλους σύνθετους που ορίστηκαν παραπάνω. Στην ενότητα αυτή θα αναφερθεί ο αλφαριθμητικός τύπος (string).

8.2.1. Ακέραιος

Οι ακέραιοι τύποι είναι οι γνωστοί μας ακέραιοι που μπορεί να είναι θετικοί ή αρνητικοί, πχ. 10 100 –10009 +51 –52 κλπ. Αν δεν υπάρχει σημείο, ο αριθμός θεωρείται θετικός. Θεωρητικά ένας ακέραιος μπορεί να έχει οποιοδήποτε πλήθος ψηφίων. Πρακτικά όμως το πλήθος των ψηφίων περιορίζεται ανάλογα με τον τύπο του υπολογιστή. Η μεταβλητή **MaxInt** (μέγιστος ακέραιος) προσδιορίζει το εύρος του διαστήματος των ακεραίων, από **–MaxInt-1** έως **MaxInt** Στην Turbo Pascal MaxInt=32767 δηλαδή, το εύρος του διαστήματος των ακεραίων είναι από –32768 έως 32767.

Έτσι οι **ακέραιοι (integer)** της Pascal, όπως και στις περισσότερες γλώσσες προγραμματισμού, είναι ένα υποσύνολο των ακεραίων, που γνωρίζουμε από τα Μαθηματικά. Στις περισσότερες εκδόσεις της Pascal παίρνουν τιμές στο διάστημα [-32768, 32767]. Με τους ακεραίους γίνονται όλες οι γνωστές από τα Μαθηματικά πράξεις αλλά τα αποτελέσματα πρέπει να βρίσκονται στο σύνολο τιμών του ακέραιου τύπου.

Μερικές εκδόσεις της Pascal, όπως η Turbo Pascal, υποστηρίζουν διάφορους τύπους ακεραίων. Κάθε τέτοιος τύπος καθορίζεται από το πεδίο τιμών του, την ύπαρξη ή μη πρόσημου και το πλήθος των bytes (8bits/byte) που καταλαμβάνει στην κεντρική μνήμη (βλέπε Πίνακα 8-1).

Πίνακας 8-1

Ακέραιοι τύποι

Δήλωση τύπου	Διάστημα τιμών	πρόσημο	πλήθος bytes
shortint	-128 .. 127	ΝΑΙ	1
integer	-32768 .. 32767	ΝΑΙ	2
longint	-2148483648 .. 2147483647	ΝΑΙ	4
byte	0 .. 255	ΟΧΙ	1
word	0 .. 65535	ΟΧΙ	2

Οι επιτρεπτές πράξεις ακεραίων είναι:		Παραδείγματα
+	πρόσθεση	
-	αφαίρεση	
*	πολλαπλασιασμός	
div	ακέραια διαίρεση (πηλίκο)	27 div 6 = 4, 16 div 17 = 0 36 div 6 = 6
mod	υπόλοιπο διαίρεσης	27 mod 6 = 3 16 mod 17 = 16 36 mod 6 = 0

8.2.2. Πραγματικός

Πραγματικοί **τύποι**
- real
- single
- double
- extended
- comp

Ο **πραγματικός (real)** τύπος χρησιμοποιείται εκεί που οι αριθμητικές τιμές δεν είναι ακέραιοι αριθμοί ή οι αναμενόμενες τιμές τους είναι εκτός ορίων του ακέραιου τύπου. Επειδή οι αριθμοί αυτοί αποθηκεύονται με διαφορετικό τρόπο από τους ακέραιους, δίνουν μεγαλύτερο εύρος τιμών και μπορεί να χρησιμοποιηθούν για πολύ μεγάλους ή πολύ μικρούς αριθμούς (ακέραιους ή δεκαδικούς).

Οι πραγματικοί αριθμοί της Pascal ορίζονται ως ένα υποσύνολο των πραγματικών αριθμών, που ξέρουμε από τα Μαθηματικά. Περιέχονται στο διάστημα από 10^{-38} μέχρι 10^{+38} περίπου και έχουν από 6 μέχρι και 20 σημαντικά ψηφία, ανάλογα με την έκδοσή της Pascal και την επιμέρους επιλογή του πραγματικού τύπου. Τα σημαντικά ψηφία εκφράζονται με τη μορφή κινητής υποδιαστολής που διαθέτει ο υπολογιστής. Τα αποτελέσματα αριθμητικών πράξεων προσεγγίζονται με το πλήθος των σημαντικών ψηφίων που διαθέτει η έκδοση της Pascal.

Π.χ. ο αριθμός 4.934456E+04 αντιστοιχεί στο δεκαδικό αριθμό 49344,56 ($4.934456 \times 10^{+4}$).

Μερικές εκδόσεις Pascal όπως η Turbo Pascal υποστηρίζει διάφορους τύπους πραγματικών. Κάθε τέτοιος τύπος καθορίζεται από το πεδίο τιμών του, το πλήθος των σημαντικών ψηφίων του και το πλήθος των bytes που καταλαμβάνει στην κεντρική μνήμη (βλέπε Πίνακα 8-2).

Πίνακας 8-2

Τύποι πραγματικών αριθμών

Δήλωση τύπου	Διάστημα τιμών	πρόσημο	πλήθος bytes
real	$-2.9 \times 10^{-39} .. 1.7 \times 10^{38}$	11-12	6
single	$-1.5 \times 10^{-45} .. 3.4 \times 10^{38}$	7-8	4
double	$-5.0 \times 10^{-324} .. 1.7 \times 10^{308}$	15-16	8
extended	$-3.4 \times 10^{-4932} .. 1.1 \times 10^{4932}$	19-20	10
comp	$-263^{+1} .. 263^{-1}$	19-20	8

Επειδή η μορφή αναπαράστασης των πραγματικών αριθμών είναι περίπλοκη, πράξεις που περιλαμβάνουν πραγματικούς αριθμούς απαιτούν περισσότερο χρόνο για εκτέλεση από αυτές των ακεραίων, οι οποίοι αποθηκεύονται απλά σε ισοδύναμη δυαδική μορφή.

Οι επιτρεπτές πράξεις πραγματικών αριθμών είναι:

\+ πρόσθεση

\- αφαίρεση

* πολλαπλασιασμός

/ διαίρεση

Αριθμητικές εκφράσεις είναι οι απεικονίσεις αριθμητικών παραστάσεων που μπορεί να περιέχουν σταθερές, μεταβλητές, συναρτήσεις, αριθμητικά σύμβολα και παρενθέσεις. Κατά την εκτέλεση των πράξεων η προτεραιότητα των μαθηματικών τελεστών φαίνεται παρακάτω

Χαμηλότερη	*Υψηλότερη*
Προτεραιότητα	***Προτεραιότητα***
+ {πρόσθεση)	* (πολλαπλασιασμός)
- (αφαίρεση)	/ (διαίρεση)
	DIV (διαίρεση ακέραια)
	MOD (υπόλοιπο ακέραιας διαίρεσης)

Αυτό σημαίνει ότι κάθε πράξη, με τα σύμβολα *, /, DIV, MOD εκτελείται πρώτη, εκτός αν υπάρχει παρένθεση, οπότε εκτελείται πρώτα η πράξη που υπάρχει μέσα στην παρένθεση. Η πράξη του πολλαπλασιασμού, σε σχέση με τη διαίρεση, έχει την ίδια προτεραιότητα. Η πρόσθεση επίσης έχει την ίδια προτεραιότητα με την αφαίρεση. Όταν τα σύμβολα έχουν την ίδια προτεραιότητα οι πράξεις εκτελούνται από αριστερά προς τα δεξιά. Μερικά παραδείγματα εκφράσεων είναι τα παρακάτω:

Έκφραση	*Αποτέλεσμα*
20 DIV 3 * 4	24
20 MOD 3 * 4	8
6 * 3 / 2 * 4	36
6 * 3 / (2 * 4)	2,25
6 + 3 / (2 * 4)	1,125

8.2.3. *Λογικός*

Ο **λογικός τύπος** (**Boolean**) έχει δύο μόνο τιμές, την **Αληθή** (**true**) και την **Ψευδή** (**false**). Πολλές φορές ο σκοπός μιας μεταβλητής λογικού τύπου είναι η καταγραφή του αποτελέσματος ενός ελέγχου. Εάν σε κάποιο σημείο του προγράμματος χρειαστεί να εξετάσουμε το αποτέλεσμα ενός ελέγχου, αρκεί να εξετάσουμε αν η τιμή της λογικής μεταβλητής είναι **Αληθής** (**true**) ή **Ψευδής** (**false**).

Οι πράξεις που μπορούν να γίνουν με μεταβλητές ή εκφράσεις λογικού τύπου είναι η σύζευξη, η διάζευξη, η αποκλειστική διάζευξη και η άρνηση και γίνονται με χρήση των λογικών τελεστών **and**, **or**, **xor** και **not** αντίστοιχα.

Στον πίνακα 8-3 παρουσιάζεται τα αποτελέσματα των παραπάνω πράξεων μεταξύ δύο λογικών μεταβλητών με όλους του συνδυασμούς των τιμών τους.

Πίνακας 8-3

Η σύζευξη, η διάζευξη, η αποκλειστική διάζευξη και η άρνηση με χρήση δύο λογικών μεταβλητών P, Q

P	Q	P and Q	P or Q	P xor Q	not P
True	True	True	True	False	False
True	False	False	True	True	False
False	True	False	True	True	True
False	False	False	False	False	True

Λογικές εκφράσεις είναι οι απεικονίσεις παραστάσεων που μπορεί να περιέχουν σταθερές, μεταβλητές, συναρτήσεις, αριθμητικά σύμβολα και παρενθέσεις και μπορούν να πάρουν μια λογική τιμή (true ή false). Μία λογική έκφραση παράγεται από δύο μεταβλητές ή σταθερές μέσω των σχεσιακών τελεστών (βλέπε πίνακα 8-4).

Πίνακας 8-4

Σχεσιακοί τελεστές

Περιγραφή	Μαθηματικά	Pascal
Μεγαλύτερο από	$>$	>
Μικρότερο από	$<$	<
Μεγαλύτερο ή ίσο	$\geq$	>=
Μικρότερο ή ίσο	$\leq$	<=
Διάφορο	$\neq$	<>
Ανήκει	$\in$	in

Ο Λογικός τελεστής **and** δέχεται δύο λογικές εκφράσεις και δίνει τιμή true, μόνο όταν και οι δύο λογικές εκφράσεις έχουν τιμή true. Ο τελεστής and έχει χαρακτηριστεί ως τελεστής λογικού πολλαπλασιασμού.

Η έκφραση (x>1) and (x<10) έχει τιμή true, μόνο όταν ο x παίρνει τιμές από 2 μέχρι και 9. Ο τελεστής and έχει προτεραιότητα σε σχέση με τους <, >, γι' αυτό οι παρενθέσεις είναι απαραίτητες. Αν στην προηγούμενη σχέση γράψουμε x>1 and x<9, αυτή είναι ισοδύναμη με την x>(1 and x)<9, η οποία στερείται νοήματος.

Ο Λογικός τελεστής **or** δέχεται δύο λογικές εκφράσεις και δίνει τιμή true, όταν τουλάχιστον μία από τις δύο λογικές εκφράσεις έχει τιμή true. Ο τελεστής or έχει χαρακτηριστεί ως τελεστής λογικής πρόσθεσης και επομένως έχει μικρότερη προτεραιότητα από τον τελεστή and. Έτσι η σχέση:

if ((x>=0) and (x<=10)) or (x=20) then...

είναι ισοδύναμη με τη σχέση if (x>=0) and (x<=10) or (x=20) then...

Ο Λογικός τελεστής **xor** δέχεται δύο λογικές εκφράσεις και δίνει τιμή true, όταν μόνο μία από τις δύο λογικές εκφράσεις έχει τιμή true. Ο τελεστής xor έχει την ίδια προτεραιότητα με τον τελεστή or.

Ο τελεστής **not** είναι ο λογικός τελεστής που δέχεται ως παράμετρο μία μόνο λογική μεταβλητή ή έκφραση και επιστρέφει την αντίθετη τιμή της δηλαδή δίνει τιμή true, αν η παράμετρος έχει τιμή false, ή δίνει false, αν η παράμετρος έχει τιμή true.

8.2.4. Χαρακτήρας

Ο **τύπος** (**char**) περιγράφει δεδομένα ενός χαρακτήρα μέσα από το σύνολο των χαρακτήρων του υπολογιστή. Σε ένα πρόγραμμα Pascal μια τιμή ενός δεδομένου αυτού του τύπου γράφεται: 'A','B','$','&' κλπ.

Ο τύπος χαρακτήρας Char είναι ένας διατεταγμένος τύπος ο οποίος περιλαμβάνει το σύνολο των χαρακτήρων που διαθέτει ο υπολογιστής μας. Η διάταξη του συνόλου των χαρακτήρων διαφέρει από υπολογιστή σε υπολογιστή. Γενικά πάντως τα ψηφία 0,1,2,...9 είναι συνεχόμενα. Το ίδιο και τα γράμματα A,B,C,D,...Z, a,b,c,d,...,z και ακολουθούν οι Ελληνικοί χαρακτήρες Α,Β,...,Ω, α,β,...,ω.

8.2.5. Αλφαριθμητικός Τύπος

Ο **Αλφαριθμητικός** (**String**) τύπος δεν συναντάται στην Standard Pascal. Ο αλφαριθμητικός τύπος είναι μία σειρά από 255, το πολύ χαρακτήρες. Εάν στη δήλωση αυτού του τύπου δεν έχει αναφερθεί το μήκος, τότε λαμβάνεται το μέγιστο μήκος, δηλαδή 255 χαρακτήρες. Το περιεχόμενο μιας μεταβλητής αυτού του τύπου μπορεί να είναι μία λέξη ή μία φράση, όπως ονοματεπώ-

νυμο, διεύθυνση κατοικίας κλπ. τα οποία περιλαμβάνονται μεταξύ εισαγωγικών. Για παράδειγμα επιτρεπτές τιμές του τύπου αυτού μπορεί να είναι:

- 'Turbo Pascal',
- 'ΤΕΕ Τεχνικά Επαγγελματικά Εκπαιδευτήρια',
- 'Τομέας Πληροφορικής' κλπ.

Ένα string πρέπει να γράφεται στην ίδια γραμμή, ειδάλλως η Turbo Pascal θα δώσει το μήνυμα STRING CONSTANT EXCEEDS LINE (η σταθερά string ξεπερνάει τη γραμμή) Η τιμή μιας μεταβλητής αλφαριθμητικού τύπου είναι η σειρά των χαρακτήρων που περιλαμβάνονται μεταξύ των εισαγωγικών. Για παράδειγμα **'Vathmos'** είναι η σειρά των χαρακτήρων που αποτελείται από τους χαρακτήρες **V a t h m o s** χωρίς τα εισαγωγικά. **Vathmos** χωρίς εισαγωγικά είναι μία μεταβλητή, το όνομα μίας περιοχής της μνήμης. Η τιμή του string **'2468'** είναι η σειρά των χαρακτήρων **2 4 6 8**. Αν γράψουμε **2468** χωρίς τα εισαγωγικά είναι ένας ακέραιος τον οποίο μπορούμε να χρησιμοποιήσουμε σε υπολογισμούς.

Ένα string μπορεί να αποτελείται, όπως αναφέραμε, από 0 έως 255 το πολύ χαρακτήρες. Ένα string με 0 χαρακτήρες είναι το κενό (null string) και δηλώνεται με δύο εισαγωγικά χωρίς κενό μεταξύ τους. Η Turbo Pascal παρέχει λειτουργίες για τη συνένωση αλφαριθμητικών τύπων, για την απομάκρυνση χαρακτήρων από ένα string και για τη σύγκριση των τιμών αλφαριθμητικών τύπων.

8.3. ΔΟΜΗ ΠΡΟΓΡΑΜΜΑΤΟΣ PASCAL

8.3.1. Επικεφαλίδα

Ένα πρόγραμμα Pascal αρχίζει με τη λέξη **program** που ακολουθείται από το **όνομα** του προγράμματος και μία **λίστα με ονόματα αρχείων** τα οποία μπορεί να είναι προαιρετικά μονάδες εισόδου / εξόδου ή τα αρχεία που θα χρησιμοποιήσει κατά την εκτέλεσή του. Η πρώτη γραμμή είναι η **επικεφαλίδα** (**heading**) του προγράμματος. Η μορφή της επικεφαλίδας είναι:

Πρόγραμμα Pascal
program όνομα προγράμματος;
δηλώσεις
begin
εκτελέσιμες εντολές
end.

program όνομα προγράμματος(όνομα αρχείου, όνομα αρχείου,...);

Παραδείγματα:

program test_1 (input,output);

program test_2;

8.3.2. Δηλώσεις

Η επικεφαλίδα ακολουθείται από προτάσεις **δηλώσεων** (**declarations**) που αναφέρονται στους όρους που χρησιμοποιούνται στο πρόγραμμα. Οι προτάσεις δηλώσεων καταλήγουν σε ερωτηματικό (;). και ακολουθούν την εξής σειρά:

- Δηλώσεις **σταθερών** (**constants**), χρησιμοποιούνται για τον ορισμό ονομάτων δεδομένων τα οποία παραμένουν σταθερά. Κάθε σταθερή τιμή που περιλαμβάνεται σε ένα πρόγραμμα ονομάζεται literal. Πριν από τον ορισμό των σταθερών αναγράφεται η λέξη **const.**

Παραδείγματα:

```
const
      pososto=18;
      nomos='ΑΘΗΝΩΝ';
      pi=3.141592653;
      e=2.718281828;
      max= 100;
      min=-100;
      part_no=1234567;
      Invalid_d=-9999999;
      letter='y';
      name= 'Blaise Pascal';
```

Ο τύπος της σταθεράς προσδιορίζεται από την τιμή της. Αν υπάρχει δεκαδικό σημείο η σταθερά θεωρείται τύπου πραγματικού. Αν δεν υπάρχει δεκαδικό σημείο εξετάζεται αν η σταθερά βρίσκεται στα όρια των ακεραίων θεωρείται τύπου Integer, διαφορετικά θεωρείται LongInt. Αν η τιμή βρίσκεται ανάμεσα σε εισαγωγικά είναι τύπου χαρακτήρα ή αλφαριθμητικού ανάλογα με το πλήθος των χαρακτήρων που περιλαμβάνει.

- Δηλώσεις **τύπων** (**type**), εφόσον ορίζονται νέοι τύποι.
- Δηλώσεις **μεταβλητών** (**variables**), χρησιμοποιούνται για τον ορισμό των ονομάτων δεδομένων τα οποία μεταβάλλονται και για τα οποία αναφέρεται ο τύπος τους. Δύο ή περισσότερες μεταβλητές μπορεί να είναι του ιδίου τύπου οπότε διαχωρίζονται με ένα κόμμα. Ο τύπος αναφέρεται στο τέλος της γραμμής μετά το σύμβολο (:). Πριν από τον ορισμό των μεταβλητών αναγράφεται η λέξη **var.**

Παραδείγματα

```
var
  pososto_forou,mistos,kath_apodoxes:real;
  eponymo, onoma : string[25];
```

- Δηλώσεις των υποπρογραμμάτων που ορίζει ο χρήστης
  Συναρτήσεις (**functions**) και
  Διαδικασίες (**procedures**) εφόσον υπάρχουν.

8.3.3. Κύριο Πρόγραμμα

Σύνθετη Εντολή (Compound statement)
Είναι μία ομαδοποίηση εντολών οι οποίες χρησιμοποιούνται όπως μία απλή εντολή και περικλείονται μεταξύ των λέξεων begin και end.

Η λέξη **begin** (**αρχή**) δηλώνει την αρχή των εντολών του προγράμματος που εκτελεί ο υπολογιστής, δηλαδή το **κύριο** πρόγραμμα, ενώ η λέξη **end** (**τέλος**) δηλώνει το τέλος του προγράμματος.

Το κύριο πρόγραμμα αποτελείται από μια σειρά εντολών, σύμφωνα με τον αλγόριθμο του προβλήματος. Κάθε εντολή που γράφεται τελειώνει υποχρεωτικά με ένα ερωτηματικό (;). Το σύμβολο αυτό είναι διαχωριστικό εντολών. Δεν είναι υποχρεωτικό να γραφεί μόνο όταν η επόμενη γραμμή αρχίζει με end (τέλος προγράμματος ή σύνθετης εντολής ή υποπρογράμματος ή προγράμματος). Μετά το end πρέπει να γραφεί μία τελεία (.) η οποία δηλώνει και το τέλος του προγράμματος

Παραδείγματα προγράμματος

1.

```
program thermokrasia;
{Το πρόγραμμα υπολογίζει τη μέση θερμοκρασία }
{πήξης και βρασμού                             }
 uses wincrt;
const  {δηλώσεις σταθερών}
  pixis_c=0;      {θερμοκρασία πήξης βαθμοί Κελσίου}
  vrasmou_c=100; {θερμοκρασία βρασμού βαθ.Κελσίου }
var    {δηλώσεις μεταβλητών}
  mesi_th_k :real; {μέση τιμή, πήξης - βρασμού}
{κύριο πρόγραμμα}
  begin
   writeln('θερμ. πήξης    ', pixis_c,    ' βαθμοί');
   writeln('θερμ. βρασμού ', vrasmou_c, ' βαθμοί');
   mesi_th_k:=( pixis_c + vrasmou_c)/2;
   writeln('μέση τιμή      ', vrasmou_c, ' βαθμοί');
  end.
```

2.

```
program mistos;
    uses wincrt;
    const  {δηλώσεις σταθερών}
         orio=2000000;    {όριο υπολογισμού φόρου}
         pos_forou=0.20; {ποσοστό υπολογισμού φόρου}
         wrom=1500;       {ποσό ανά υπερωριακή ώρα}
    var    {δηλώσεις μεταβλητών}
       bm,yper,foros,akath,plir:real;
         {bm    = βασικός μισθός
         yper  = ώρες υπερωρίας
         foros = φόρος
         akath = ακαθάριστες αποδοχές
         plir  = πληρωτέο ποσό}
         yp_yper:real;     {αποδοχές από υπερωρίες}

function ypol_ap(basikos,yp:real):real;
  {συνάρτηση υπολογισμού αποδοχών}
```

```
    begin
      yp_yper:=yp*wrom;              {υπολ. υπερωριών}
      ypol_ap:=yp_yper+basikos;      {υπολ. αποδοχών}
    end;

function ypol_forou(synolo:real):real;
   {συνάρτηση υπολογισμού φόρου}
    begin
      if synolo<=orio     {συνθήκη υπολογισμού φόρου}
       then ypol_forou:=0
       else ypol_forou:=synolo*pos_forou;
    end;

procedure  ektyposi;
   {διαδικασία εκτύπωσης αποτελεσμάτων}
 begin
    writeln(bm:6:0,' ',yp_yper:6:0,' ',akath:6:0,'
          ',foros:6:0,' ',plir:6:0);
 end;

{κύριο πρόγραμμα}
  begin
     writeln('δώσε βασικό μισθό, ώρες υπερωρίας ');
     readln(bm,yper);
     akath:=ypol_ap(bm,yper);     {ακαθάριστες αποδοχές}
     foros:=ypol_forou(akath);    {φόρος}
     plir:=akath-foros;           {πληρωτέο - καθαρά}
     ektyposi;                 {εκτύπωση αποτελεσμάτων}
end.
```

Ανακεφαλαίωση

Γνωρίσαμε την δομή και τη φιλοσοφία της γλώσσας Pascal. Γνωρίσαμε τους βασικούς τύπους δεδομένων της γλώσσας και τον τρόπο ορισμού σταθερών και μεταβλητών. Κατανοήσαμε τη δομή ενός προγράμματος σε γλώσσα Pascal. Γνωρίσαμε την έννοια της σύνθετης εντολής. Γνωρίσαμε τη δυνατότητες δημιουργίας τύπων δεδομένων από τον προγραμματιστή και της δυναμικής διαχείρισης της μνήμης με δείκτες.

Ερωτήσεις

1. Να αναφέρετε μερικά από τα ιδιαίτερα χαρακτηριστικά της Pascal.
2. Ποιό είναι το αλφάβητο της Pascal;
3. Ποιά είναι τα ονόματα- ταυτότητες και σε τι χρησιμεύουν;
4. Σε τί χρησιμεύει το Συντακτικό (Syntax) της γλώσσας;
5. Σε τί χρησιμεύει η Σημασιολογία (Semantics);
6. Ποιοί είναι οι απλοί ή στοιχειώδεις τύποι δεδομένων;
7. Πότε χρησιμοποιείται ο πραγματικός τύπος;
8. Ποιές είναι οι πράξεις που μπορεί να γίνουν με μεταβλητές ή εκφράσεις λογικού τύπου;
9. Να εξηγήσετε τη σημασία των: 2000, '2000', etos, 'etos' στη γλώσσα Pascal.
10. Ποιά είναι η δομή ενός προγράμματος Pascal;

Ασκήσεις

1. Να γράψετε με τον editor, να μεταφράσετε και να εκτελέσετε το παρακάτω πρόγραμμα χρησιμοποιώντας τις κατάλληλες εντολές.

```
program charset(output);
{  το πρόγραμμα εκτυπώνει το σύνολο των χαρακτήρων
ASCII }
var
 ch:char;
begin
 for ch:=chr(32) to chr(255) do
    write(ch);
 writeln
end.
```

2. Να γράψετε με τον editor,να μεταφράσετε και να εκτελέσετε το παρακάτω πρόγραμμα χρησιμοποιώντας τις κατάλληλες εντολές.

```
program table(output);
{το πρόγραμμα τυπώνει τα τετράγωνα και τους κύβους
 των αριθμών 1 έως και 10 }
var
 i:integer;
begin
 writeln('Number':10,'square':10,'cube':10);
 for i:=1 to 10 do
writeln(i:10,sqr(i):10,i*sqr(i):10)
end.
```

ΚΕΦΑΛΑΙΟ 9
ΒΑΣΙΚΕΣ ΕΝΤΟΛΕΣ

Σκοπός κεφαλαίου:

Να μπορούμε γρήγορα να υλοποιήσουμε ένα απλό πρόγραμμα σε γλώσσα Pascal.

Ειδικοί σκοποί:

- Να κατανοήσουμε τις εντολές εισόδου/εξόδου.
- Να κατανοήσουμε τη σημασία της εντολής αντικατάστασης.
- Να κατανοήσουμε τον τρόπο της διαδοχικής εκτέλεσης των εντολών ενός προγράμματος.

9.1. ΕΝΤΟΛΕΣ ΕΙΣΟΔΟΥ / ΕΞΟΔΟΥ

Τα δεδομένα των προγραμμάτων λαμβάνονται από μια μονάδα εισόδου, όπως για παράδειγμα το πληκτρολόγιο. Για την εισαγωγή των δεδομένων η Pascal χρησιμοποιεί τις διαδικασίες (**procedures**) **read** ή **readln**. Για την εμφάνιση των περιεχομένων μεταβλητών ή σταθερών στην οθόνη, στον εκτυπωτή ή σ' ένα αρχείο χρησιμοποιούνται οι διαδικασίες **write** ή **writeln**.

Οι εντολές read, readln, write, writeln χρησιμοποιούνται και για ανάγνωση/εγγραφή από αρχεία τα οποία είναι καταχωρημένα στο δίσκο. Στην παρούσα ενότητα, θα αναφερθεί η χρήση των παραπάνω εντολών για ανάγνωση από το πληκτρολόγιο και εγγραφή στην οθόνη. Όταν οι εντολές εισόδου/εξόδου χρησιμοποιούνται στην Turbo Pascal για Windows, θα πρέπει να γραφεί αμέσως μετά τη δήλωση του προγράμματος η δήλωση **uses wincrt**.

Για πρακτικούς λόγους οι διαδικασίες εισόδου / εξόδου ονομάζονται και εντολές εισόδου / εξόδου.

Η χρήση των εντολών εισόδου/εξόδου σε αρχεία θα γίνει σε ξεχωριστό κεφάλαιο.

9.1.1. read, readln

***Μορφή*:** **read (parameter,parameter,...,)**

Ενέργεια: Η διαδικασία read ακολουθείται από μία ή περισσότερες παραμέτρους. Οι παράμετροι αυτές είναι τα ονόματα των διευθύνσεων της μνήμης όπου θα αποθηκευτούν τα δεδομένα, δηλαδή ονόματα **μεταβλητών**

που χωρίζονται με ',' (κόμμα) και περιέχονται σε μία παρένθεση. Όταν εκτελείται η διαδικασία, τα δεδομένα διαβάζονται και αποδίδονται, κατά τη σειρά που είναι γραμμένα, στις μεταβλητές όπως αυτές εμφανίζονται στη διαδικασία read. Όταν γίνεται εισαγωγή δεδομένων, και εφόσον αυτά είναι αριθμητικά και εμφανίζονται περισσότερα από ένα ανά γραμμή, πρέπει να χωρίζονται με ένα τουλάχιστον κενό. Τα δεδομένα τύπου char δεν πρέπει να χωρίζονται με κενά, γιατί το κενό είναι κι αυτό ένας χαρακτήρας ο οποίος θα διαβαστεί.

Αν για παράδειγμα οι παράμετροι της εντολής read είναι ακέραιοι για κάθε παράμετρο, αρχίζοντας από την πρώτη, εκτελούνται τα παρακάτω:

- 1. Διαβάζεται ένας αριθμός από το πληκτρολόγιο.
- 2. Αποθηκεύεται ο αριθμός στη μεταβλητή (αντίστοιχη διεύθυνση της μνήμης)
- 3. Επαναλαμβάνονται τα βήματα 1 και 2 για την επόμενη μεταβλητή.

Όταν αποθηκευτεί τιμή και στην τελευταία μεταβλητή της λίστας των παραμέτρων, τελειώνει και η εκτέλεση της εντολής **read.**

Θα πρέπει να είμαστε προσεκτικοί κατά την εισαγωγή των δεδομένων. Η εισαγωγή πραγματικού αριθμού σε ακέραια μεταβλητή δεν είναι επιτρεπτή. Μπορούμε όμως να διαβάσουμε μια ακέραια τιμή σε πραγματική μεταβλητή.

	Εντολή	Δεδομένα	Περιεχόμενα μετά την εκτέλεση
1	read(x)	22	x=22
2	read(a, b, c)	33 44 55	a=33 b=44 c=55
3	read(d, e)	66 77	d=66 e=77
4	read(f, g)	11.99 21 31 41 51	f=11.99 g=21
5	read(x, a, d)		x=31 a=41 d=51

Παρατηρήσεις:

- Τα δεδομένα της γραμμής 3 δεν είναι αρκετά για όλες της παραμέτρους κι έτσι η εισαγωγή δεδομένων συνεχίζεται στην επόμενη γραμμή.
- Οι μεταβλητές της εντολής read στη γραμμή 4 είναι λιγότερες από τα δεδομένα. Για το λόγο αυτό, τα επιπλέον δεδομένα είναι διαθέσιμα για την επόμενη εντολή read στη γραμμή 5.

Στα προηγούμενα παραδείγματα όλες οι μεταβλητές είναι **ακέραιες (integer)** εκτός από την f η οποία είναι **πραγματική(real)**.

Οι εντολές read(a), read(b) και read(a, b) είναι ισοδύναμες. Η λίστα των μεταβλητών διευκολύνει τον προγραμματιστή στην κωδικοποίηση.

***Μορφή*: readln(parameter, parameter,...,)**

Ενέργεια Η διαδικασία readln, μπορεί να ακολουθείται από μία ή περισσότερες παραμέτρους (**μεταβλητές**) που χωρίζονται με ',' (κόμμα) και περιέχονται σε μία παρένθεση. που είναι ισοδύναμη των read και readln. Κατ' αυτό τον τρόπο τα ζητούμενα διαβάζονται από την τρέχουσα γραμμή (read) και στη συνέχεια ο έλεγχος περνάει στην επόμενη γραμμή (readln). Η διαδικασία readln, όταν εμφανίζεται χωρίς παραμέτρους, έχει ως αποτέλεσμα να περνάει ο έλεγχος στην επόμενη γραμμή.

Ας υποθέσουμε ότι έχουμε δύο ακέραιες τιμές σε κάθε γραμμή δεδομένων. Ο πίνακας που ακολουθεί δείχνει τα περιεχόμενα των μεταβλητών μετά την εκτέλεση της εντολής readln.

	Εντολή	Δεδομένα	Περιεχόμενα μετά την εκτέλεση
1	readln(a) readln(b) readln(c)	10 20 30 40 50 60	a=10 b=30 c=50
2	readln(a,b,c) readln(d,e)	10 20 30 40 50 60	a=10 b=20 c=30 d=50 e=60
3	readln (a, b) readln(c,d) readln(e)	10 20 30 40 50 60	a=1 b=20 c=30 d40 e=50
4	readln(a) readln(b) readln(c,d,e)	10 20 30 40 50 60	a=10 b=30 c=50 d=60 e=?
5	readln(a,b) readln readln(c)	10 20 30 40 50 60	a=10 b=20 c=50

Παρατηρήσεις:

- Δεν υπάρχουν δεδομένα για τη μεταβλητή e η εκτέλεση σταματάει μέχρι να γίνει εισαγωγή από το πληκτρολόγιο.
- Μια εντολή readln χωρίς παραμέτρους έχει ως αποτέλεσμα να παραλειφθεί μία γραμμή δεδομένων.

9.1.2. write, writeln

***Μορφή*: write (parameter,parameter,...,)**

Ενέργεια: Οι παράμετροι μπορεί να είναι σταθερές, μεταβλητές ή εκφράσεις, σε αντίθεση με την εντολή read η οποία δέχεται ως παραμέτρους μόνο μεταβλητές. Η τιμή καθεμιάς παραμέτρου κατά σειρά, τυπώνεται στη γραμμή από αριστερά προς τα δεξιά Στην απλούστερη περίπτωση οι παράμετροι μπορεί να είναι αριθμοί, χαρακτήρες ή **αλφαριθμητικά στοιχεία (συμβολοσειρές) (strings)**. Εάν πρόκειται να γραφούν **συμβολοσειρές** σταθερών, αυτές θα πρέπει να περιέχονται μεταξύ απλών εισαγωγικών.

Παράδειγμα:

write ('αυτή είναι μία συμβολοσειρά')

***Μορφή*: writeln(parameter, parameter,...,)**

Ενέργεια: Η διαδικασία writeln, αν ακολουθείται από παραμέτρους, είναι ισοδύναμη των write και writeln. Η τιμή κάθε μιας παραμέτρου κατά σειρά, τυπώνεται στην γραμμή από αριστερά προς τα δεξιά και στη συνέχεια η επόμενη εντολή εισόδου / εξόδου εκτελείται στην αρχή της επόμενης γραμμής. Η διαδικασία writeln, όταν εμφανίζεται χωρίς παραμέτρους, έχει ως αποτέλεσμα να περνάει ο έλεγχος στην αρχή της επόμενης γραμμής.

Ο πίνακας που ακολουθεί δείχνει τα αποτελέσματα μετά την εκτέλεση εντολής **writeln.**

Περιεχόμενα παραμέτρων	Εντολή	Αποτελέσματα
i=2	writeln (i)	2
	writeln('i=', i)	i=2
	writeln('Error=',i)	Error=2
	writeln('Error Message')	Error Message
la ='Pascal'	writeln(ty, ' ',la)	TURBO Pascal
ty ='TURBO'	writeln(ty, la)	TURBOPascal
	writel writeln('Language=', la)	Language=Pascal

Η εισαγωγή των δεδομένων γίνεται με την εντολή read. Η εισαγωγή απλοποιείται, όταν με τη διαδικασία write το ίδιο το πρόγραμμα δίνει ένα προτρεπτικό μήνυμα, δηλαδή εμφανίζει στην οθόνη ποια είναι τα ζητούμενα δεδομένα.

Παράδειγμα:

```
write('Δώσε βασικό μισθό :  ');
readln(mistos);
```

Μορφοποιημένη εκτύπωση

Οι δύο βασικές τεχνικές για τη μορφοποίηση της εκτύπωσης είναι οι κενές γραμμές και η εισαγωγή κενών στις γραμμές εκτύπωσης. Για να έχουμε κενές γραμμές αρκεί να δώσουμε την εντολή writeln χωρίς παραμέτρους τόσες φορές όσες είναι οι κενές γραμμές που θέλουμε. Για την εισαγωγή κενών στη γραμμή μεταξύ των παραμέτρων περιλαμβάνουμε και όσα κενά θέλουμε μεταξύ εισαγωγικών.

Ο έλεγχος της μορφής των αποτελεσμάτων στη γραμμή εκτύπωσης γίνεται με πληροφορία που ακολουθεί την παράμετρο στη διαδικασία write ή writeln. Η πληροφορία αυτή δίνει το επιθυμητό εύρος του πεδίου δηλαδή τις θέσεις που θα καταλάβει η παράμετρος κατά την εκτύπωσή της..

Για σταθερές / μεταβλητές **Ακέραιου** ή **Αλφαριθμητικού** τύπου, η μορφή εκτύπωσης είναι **write(x:m)**. Όπου:

x η μεταβλητή που θα τυπωθεί,

m το συνολικό πλήθος των θέσεων εκτύπωσης.

Η διαδικασία write(x:m), όπου **x** είναι ακέραια έκφραση, τυπώνει την τιμή της μεταβλητής x αρχίζοντας από τα δεξιά ενός πεδίου εύρους **m**. Αν το εύρος είναι μεγαλύτερο από το μήκος του πεδίου που πρόκειται να τυπωθεί, εμφανίζονται κενά στην αριστερή πλευρά της τιμής. Αν το εύρος είναι μικρότερο από το μήκος του πεδίου που πρόκειται να τυπωθεί, αγνοείται η οδηγία μορφοποίησης και χρησιμοποιείται το ελάχιστο απαιτούμενο εύρος.

Η μορφοποιημένη εκτύπωση μπορεί να χρησιμοποιηθεί και για σταθερές ή μεταβλητές αλφαριθμητικού τύπου (string). Η χρήση γίνεται όπως και στους ακεραίους, δηλαδή δίνεται το πλήθος των θέσεων εκτύπωσης της μεταβλητής ή της σταθεράς. Η στοίχιση γίνεται από δεξιά.

Π.χ. Η εντολή `writeln ('Αριθμός μητρώου.':20);` θα έχει ως αποτέλεσμα να αφήσει 4 κενές θέσεις εκτύπωσης από την αρχή της γραμμής ή από την τελευταία θέση εκτύπωσης και μετά να γράψει το περιεχόμενο της σταθεράς δηλαδή: `Αριθμός μητρώου.`

Για σταθερές / μεταβλητές **Πραγματικού** τύπου η μορφή εκτύπωσης είναι **write(x:m:n)**. Όπου:

x η μεταβλητή πραγματικού τύπου που θα τυπωθεί,

m το συνολικό πλήθος των θέσεων εκτύπωσης, συμπεριλαμβανομένης της υποδιαστολής και

n το πλήθος των δεκαδικών ψηφίων.

Με τον τρόπο αυτό γίνεται η στρογγυλοποίηση και ο αριθμός εμφανίζεται με τη γνωστή από τα Μαθηματικά, μορφή σταθερής υποδιαστολής. Π.χ. `write(3.14159265:10:4)` θα δώσει ως αποτέλεσμα `3.1416`. Γίνεται δηλαδή στρογγυλοποίηση στο τέταρτο δεκαδικό ψηφίο και εμφανί-ζονται και 4 κενά πριν από τον αριθμό, για να συμπληρωθεί το πλήθος των 10 θέσεων.

9.2. ΕΝΤΟΛΗ ΑΝΤΙΚΑΤΑΣΤΑΣΗΣ

***Μορφή*: μεταβλητή := έκφραση**

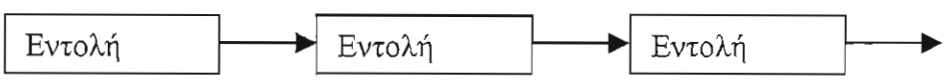
Ενέργεια: Το περιεχόμενο της μνήμης που ορίζεται με το όνομα της μεταβλητής που βρίσκεται αριστερά του := αντικαθίσταται από την τιμή που ορίζεται από την έκφραση η οποία βρίσκεται δεξιά του := (συμβόλου αντικατάστασης). Ο συμβολισμός := θα πρέπει να θεωρείται ως ένα μόνο σύμβολο και για το λόγο αυτό δεν επιτρέπονται κενά. Η έκφραση του δεξιού μέρους μπορεί να είναι και μία σταθερά ή άλλη μεταβλητή και τότε η τιμή τους αντικαθιστά την τιμή της μεταβλητής που βρίσκεται αριστερά.

Παραδείγματα:

```
x:=1;          {η μεταβλητή x λαμβάνει την τιμή 1}
x:=x+1;        {το περιεχόμενο της μεταβλητής x αυξάνει κατά 1 και
                το αποτέλεσμα καταχωρείται στη μεταβλητή x}
y:=(3*x+5)/(x+3);
               {το περιεχόμενο της μεταβλητής y αντικαθίσταται
          από την τιμή της αλγεβρικής παράστασης (3x+5)/(x+3)
onoma:='ΑΡΗΣ';
               {Η μεταβλητή onoma, η οποία πρέπει να έχει
                ορισθεί ως string, λαμβάνει την τιμή ΑΡΗΣ}
star := '*' ;  {Η μεταβλητή star, η οποία πρέπει να έχει ορισθεί
                ως  string ή  char λαμβάνει την τιμή *}
```

9.3. ΑΚΟΛΟΥΘΙΑ

Ακολουθία είναι μια σειρά από εντολές οι οποίες εκτελούνται η μία μετά την άλλη.



Το παρακάτω πρόβλημα είναι ένα χαρακτηριστικό παράδειγμα ακολουθίας:

Με δεδομένα την προφορική βαθμολογία των δύο τετραμήνων και τη γραπτή βαθμολογία, να υπολογιστεί ο μέσος όρος της βαθμολογίας.

Όπως φαίνεται, το πρόγραμμα που ακολουθεί αποτελείται από διαδοχικές εντολές, εισόδου, εξόδου και αντικατάστασης.

```
program mo_bathmwn;
{Μέσος όρος βαθμολογίας}
uses wincrt;
var pr1, pr2, gr, sum     : integer;
    mo, mo_pr              : real;

begin
  write('δώσε βαθμολογία προφορικών Α ΤΕΤΡΑΜΗΝΟΥ  ');
  readln(pr1);
  write ('δώσε βαθμολογία προφορικών Β ΤΕΤΡΑΜΗΝΟΥ  ');
  readln(pr2);
  write ('δώσε βαθμολογία ΓΡΑΠΤΩΝ                 ');
  readln(gr);
  mo_pr:=(pr1+pr2)/2;
  mo:= (mo_pr+ gr)/2;
  writeln('προφορικά =',mo_pr :3:1);
  writeln('γραπτά    =',gr :2);
  writeln('μέσος όρος=',mo :3:1);
end.
```

Ανακεφαλαίωση

Γνωρίσαμε τη σύνταξη και τον τρόπο χειρισμού των εντολών εισόδου στοιχείων στον υπολογιστή από το πληκτρολόγιο. Γνωρίσαμε τη σύνταξη και τον τρόπο χειρισμού των εντολών εξόδου στοιχείων στην οθόνη του υπολογιστή. Κατανοήσαμε τη χρήση και τη σημασία των εντολών αυτών μέσα από χαρακτηριστικά παραδείγματα. Γνωρίσαμε τις δυνατότητες να δημιουργούμε μορφοποιημένες εκτυπώσεις. Επίσης κατανοήσαμε την έννοια της εντολής αντικατάστασης και της ακολουθίας εντολών. Τέλος χρησιμοποιήσαμε τις εντολές αυτές για τη δημιουργία των πρώτων εντολών Pascal.

Ερωτήσεις

1. Να εξηγήσετε πώς λειτουργούν οι διαδικασίες
   read(parameter, parameter,...) και readln(parameter,parameter,...);
   Ποιες είναι οι ομοιότητες και ποιες είναι οι διαφορές τους;

2. Να εξηγήσετε πώς λειτουργούν οι διαδικασίες write(parameter, parameter,...) και writeln(parameter,parameter,...);
   Ποιές είναι οι ομοιότητες και ποιές οι διαφορές τους;

3. Να εξηγήσετε πώς λειτουργούν οι διαδικασίες write(x : m) και writeln(x :m :n); Ποιες είναι οι ομοιότητες και ποιες οι διαφορές τους;
4. Ποιο θα είναι το αποτέλεσμα της εντολής write(3.14159265:10:3);
5. Ποια είναι τα αποτελέσματα των παρακάτω εντολών, αν αυτές εκτελούνται διαδοχικά x:=2; x:=2*x+4; y:=(4*x-1)*(5*x-4);
6. Να συμπληρώσετε τα κενά στον πίνακα που ακολουθεί.

	Εντολή	Δεδομένα	Περιεχόμενα μετά την εκτέλεση
1	read(x)	122	x=
2	read(a, b, c)	133 244 355	a= b= c=
3	read(d, e)	166 277 3888	d= e=
4	read(f, g)	11.99 21 31 41 51	f= g= h=
5	read(x, a, d)	10 20 30	x= a= d=

7. Να συμπληρώσετε τα κενά στον πίνακα που ακολουθεί.

Περιεχόμενα παραμέτρων	Εντολή	Αποτελέσματα
i=2	writeln (i) writeln('i=', i) writeln('Error=',i) writeln('Error Message')	i= _____ _____ _____= ____ _____ ___
la ='name=' ty ='Ttttt'	writeln(ty, ' ',la) writeln(ty, la) writel writeln('Language=', la)	_____ _____

Ασκήσεις

1. Να γράψετε πρόγραμμα που να τυπώνει:
   στη γραμμή 1 αυτό είναι το πρώτο μου πρόγραμμα
   στη γραμμή 2 το όνομά σας
   στη γραμμή 3 ημερομηνία
2. Να γράψετε πρόγραμμα στο οποίο να γίνεται εισαγωγή των δεδομένων μήκος, πλάτος, ύψος έτσι, ώστε να υπολογίζονται και να τυπώνονται τα παρακάτω αποτελέσματα:

   το εμβαδόν της βάσης είναι .....τετρ.εκ.
   το εμβαδόν της έδρας με διαστάσεις .., ..
   είναι .....τετρ.εκ.

το εμβαδόν της έδρας με διαστάσεις .., ..
είναι .....τετρ.εκ.
ο όγκος του παραλληλεπιπέδου είναι .....κυβ. εκ.

3. Να γράψετε πρόγραμμα το οποίο να διαβάζει έναν αριθμό και να υπώνει το διπλάσιο και το τριπλάσιό του. Να γίνει η ίδια διαδικασία για τους 2 επόμενους απ΄ αυτόν αριθμούς. Η μορφή της εκτύπωσης να είναι σύμφωνα με το παρακάτω υπόδειγμα:

```
 8  16 24
 9  18 27
10  20 30
```

4. Να γράψετε πρόγραμμα στο οποίο να γίνεται εισαγωγή των ψήφων τριών ατόμων για το καθένα χωριστά. Να βρεθεί ο μέσος όρος των ψήφων και να τυπωθεί η διαφορά των ψήφων καθενός από το μέσο όρο. Οι αριθμοί κατά την εκτύπωση να συνοδεύονται από τις κατάλληλες επεξηγήσεις σύμφωνα με το υπόδειγμα.

Η μορφή της εκτύπωσης θα είναι:
Ο Γιώργος πήρε ..... ψήφους
Ο Χάρης πήρε ..... ψήφους
Ο Χρίστος πήρε ..... ψήφους
– –
ο μέσος όρος των ψήφων είναι ..... ψήφοι
– –
Η διαφορά από το μέσο όρο είναι:
για το Γιώργο ..... ψήφοι
για το Χάρη ..... ψήφοι
για το Χρίστο ..... ψήφοι

5. Να γράψετε πρόγραμμα που να διαβάζει το μήκος της ακτίνας ενός κύκλου και να τυπώνει τη διάμετρο, το μήκος και το εμβαδόν αυτού του κύκλου.

αποτελέσματα:
για κύκλο ακτίνας **** εκ.
η διάμετρος είναι ***** εκ.
το μήκος της περιφέρειας είναι ***** εκ.
το εμβαδόν του κύκλου είναι ******* τετρ.εκ.

6. Να γράψετε πρόγραμμα που να διαβάζει ένα τριψήφιο ακέραιο αριθμό και να τον τυπώνει ανάστροφα. Π.χ.

ο αριθμός είναι 123
ο ανάστροφος είναι 321

7. Να γράψετε το ίδιο πρόγραμμα για έναν τετραψήφιο αριθμό.
8. Να γράψετε πρόγραμμα που να διαβάζει έναν πραγματικό αριθμό και να τυπώνει τις 5 πρώτες δυνάμεις του και το άθροισμα των 3 πρώτων δυνάμεών του.

Τα αποτελέσματα να έχουν τη μορφή:
αριθμός :
1η δύναμη του αριθμού :
2η δύναμη του αριθμού :
3η δύναμη του αριθμού :
4η δύναμη του αριθμού :
5η δύναμη του αριθμού :

άθροισμα των δυνάμεων 1,2,3 :

9. Να γράψετε πρόγραμμα που να διαβάζει θερμοκρασία σε βαθμούς Fahrenheit και να τη μετατρέπει σε βαθμούς Κελσίου. Ο τύπος μετατροπής είναι:

$$\frac{(f-32)}{9}=\frac{c}{5}$$

10. Να γράψετε πρόγραμμα που να διαβάζει τους συντελεστές α,β, της πρωτοβάθμιας εξίσωσης αx+β=0 και να υπολογίζει τη ρίζα της.
(Υποτίθεται ότι α, β διάφορα του μηδενός).

11. Να γράψετε πρόγραμμα που να διαβάζει ημερομηνία με τη μορφή μέρα/μήνας/έτος και να την τυπώνει με τη μορφή έτος/μήνας/μέρα
αποτελέσματα:
διαβάστηκε η ημερομηνία ../../..
που μετατρέπεται στην ../../..

Δραστηριότητες

1. Να γράψετε 3 δικά σας προβλήματα όπου θα χρησιμοποιήσετε εντολές εισόδου/εξόδου και αντικατάστασης και θα κωδικοποιήσετε τον αλγόριθμο καθενός από αυτά

ΚΕΦΑΛΑΙΟ 10
ΕΝΤΟΛΕΣ ΕΠΙΛΟΓΗΣ ΚΑΙ ΑΠΟΦΑΣΕΩΝ

Σκοπός κεφαλαίου:

Να κατανοήσουμε τις δυνατότητες της αλλαγής της διαδοχικής εκτέλεσης των εντολών ενός προγράμματος ανάλογα με τα αποτελέσματα συνθηκών που περιλαμβάνονται στις ίδιες τις εντολές.

Ειδικοί σκοποί:

- Να κατανοήσουμε τη σύνταξη και τη λειτουργία της εντολής if σε όλες της δυνατές μορφές.
- Να κατανοήσουμε τη σύνταξη και τη λειτουργία της εντολής case καθώς και τον τρόπο με τον οποίο αυτή χρησιμοποιείται για την καλύτερη δόμηση του προγράμματος.

Αλλαγή σειράς εκτέλεσης των εντολών

Όπως είναι γνωστό, οι εντολές ενός προγράμματος εκτελούνται διαδοχικά, η μία μετά την άλλη. Αυτό δεν είναι πάντοτε επιθυμητό. Η σειρά εκτέλεσης των εντολών μπορεί να αλλάξει με τη χρήση των εντολών ελέγχου και επιλογής. Με τις εντολές αυτές επιτυγχάνεται η εκτέλεση ορισμένων εντολών υπό συνθήκες.

10.1. IF

***Μορφή*:** **if Λογική έκφραση**
then εντολή-1

***Ενέργεια*:** Αν ή τιμή της λογικής έκφρασης είναι true (σωστό – αληθής), τότε εκτελείται ή εντολή που ακολουθεί το then.

***Μορφή*:** **if Λογική έκφραση**
then εντολή-1
else εντολή-2

***Ενέργεια*:** : Αν ή τιμή της λογικής έκφρασης είναι true (σωστό – αληθής), τότε εκτελείται ή εντολή-1 που ακολουθεί το then, αλλιώς εκτελείται η εντολή-2 που ακολουθεί το else.

Παραδείγματα:

α.
```
if  wres  <= 40
    then  plir:=wr_apoz  * wres
    else  plir:=wr_apoz  ( 40+ (wres-40)*1.5);
```

Δηλαδή αν οι ώρες εργασίας δεν είναι περισσότερες από 40 πολλαπλασιάζουμε την ωριαία αποζημίωση επί τις ώρες και βρίσκουμε το πληρωτέο ποσό. Οι επί πλέον των 40 ωρών θεωρούνται υπερωριακές και πολλαπλασιάζονται με το συντελεστή 1.5. Μερικές φορές η εντολή if χρησιμοποιείται και για έλεγχο των δεδομένων. Για παράδειγμα, πριν γίνει διαίρεση πρέπει να γίνει έλεγχος μήπως ο διαιρέτης έχει τιμή μηδέν οπότε δεν είναι δυνατή η διαίρεση και τυπώνεται έτσι το κατάλληλο μήνυμα, όπως φαίνεται στο παράδειγμα που ακολουθεί.

β.
```
if diaireths <>0
    then  piliko:=diaireteos div diaireths
    else  writeln('ΛΑΘΟΣ διαίρεση με μηδέν δε γίνε-
                   ται');
```

Αν μετά το then ή το else ακολουθούν περισσότερες από μία εντολές, θεωρούμε ότι αποτελούν μία ακολουθία από εντολές. Τις εντολές αυτές βάζουμε μεταξύ των begin – end και το μεταφραστικό πρόγραμμα τις θεωρεί ως μία εντολή, **σύνθετη** εντολή **compound** statement. Το προηγούμενο παράδειγμα θα μπορούσε να έχει την εξής μορφή:

```
if diaireths <>0
then
 begin
  piliko:=diaireteos div diaireths´;
  writeln('διαίρεση δυνατή');
 end
 else
  begin
   writeln('ΛΑΘΟΣ  διαίρεση με μηδέν δε γίνεται');
   piliko:=maxint;
  end;
```

Η εντολή η οποία ακολουθεί το then ή το else μπορεί να είναι μια άλλη εντολή επιλογής (ένα άλλο if). Τότε λέμε ότι έχουμε **φωλιά** (**nest**) επιλογών ή φωλιά από (nested) if.

Παραδείγματα:

α.
```
if  a1
then  a2
else  if  a3
      then a4
```

γ.
```
program exisosi_a_bathmou;
var
   a,b,x:real;
begin
  write('δώσε το a: ');
```

```
        else a5

β.

 if a=b
then write ('a=b')
else
 if a>b
 then
writeln('a>b')
     else
    writeln('a<b');
```

```
    readln(a);
    write('δώσε το b: ');
    readln(b);
    if a<>0
    then
      begin
      x:= a/b;
      writeln('x =', x:8:2);
      end
    else
      if b<>0
      then writeln('αδύνατη
      εξίσωση')
      else writeln('αόριστη
      εξίσωση');
end.
```

10.2. CASE

Μορφή: **case** έκφραση **of**

case label, case label ...: εντολή-1;

case label, case label ...: εντολή-2;

.

.

.

else εντολή-λ

end;

Ενέργεια: Η εντολή case προσφέρει δυνατότητα πολλαπλής επιλογής.

Επιλέγεται για εκτέλεση η εντολή-κ, όταν η έκφραση έχει ως τιμή μια ετικέτα (label) από τα στοιχεία της λίστας ετικετών της εντολής-κ.

Η **ετικέτα** (**case label**) μπορεί να είναι:

μια καθορισμένη τιμή (literal)

μια σταθερά με όνομα

οποιοσδήποτε διατεταγμένος τύπος (ακέραιος, χαρακτήρας κλπ)

Η **έκφραση** είναι ο επιλογέας (selector) της case.

Η λίστα ετικετών της case είναι μια λίστα διατεταγμένων τιμών του ίδιου τύπου με τον επιλογέα της case.

Αν η έκφραση δεν πάρει καμιά από τις αναφερόμενες τιμές, τότε εκτελείται η εντολή-λ, εφόσον έχει δηλωθεί μέσω της else. Αν σε μία επιλογή πρέπει να

εκτελεστούν περισσότερες εντολές, τότε χρησιμοποιούμε τη σύνθετη εντολή μέσω των begin-end.

Ακολουθούν χαρακτηριστικά παραδείγματα για την κατανόηση της case.

Μία πρόταση case μπορεί να είναι της μορφής:

α. case boolean έκφραση of

true: εντολή1

false: εντολή2

end {case}

Εδώ επιλογέας είναι μια λογική έκφραση. Οι τιμές της μπορεί να είναι true ή false. Εκτελείται η εντολή που έχει ως ετικέτα την τιμή της λογικής έκφρασης, δηλαδή η εντολή1, αν η τιμή της λογικής έκφρασης είναι true ή η εντολή2, αν η τιμή της λογικής έκφρασης είναι false.

Μία τέτοια μορφή που ελέγχεται από μία λογική έκφραση μπορεί βέβαια να έχει το πολύ δύο δυνατές επιλογές. Η μορφή αυτή είναι ισοδύναμη της εντολής if.

Παραδείγματα:

α.

```
 case letter of
      'x'          : εντολή-1
      'l', 'm'     : εντολή-2
      's'          : εντολή-3
 end; {case}
εντολή4
```

Εδώ επιλογέας είναι ο διατεταγμένος τύπος γράμμα (letter). Οι ετικέτες είναι literal του ιδίου τύπου με τον επιλογέα της **case.** Υπάρχει μόνο μία λίστα με ετικέτες που αντιστοιχούν στην εντολή-2. Αν ο επιλογέας έχει ως τιμή μια από τις ετικέτες της λίστας, δηλαδή l ή m, τότε εκτελείται η εντολή-2.

β.

```
 case vathmologia  of
  'A','B'      : writeln ('πολύ καλά'); {εντολή-1}
  'C','D'      : writeln ('καλά');      {εντολή-2}
  'E','F','G'  : begin                   {εντολή-3}
                   writeln ('καλά);
                   new_test:=new-test+1;
                 end;
else writeln(' βαθμολογία εκτός ορίων')
 end; {case}
εντολή-4
```

Εδώ επιλογέας είναι **vathmologia** τύπου Char (χαρακτήρα), δηλαδή διατεταγμένου τύπου. Οι ετικέτες είναι literal του ιδίου τύπου με τον επιλογέα

της **case.** Υπάρχουν τρεις λίστες με ετικέτες που αντιστοιχούν σε κάθε μία από τις τρεις εντολές.. Η τρίτη εντολή είναι σύνθετη. Αν ο επιλογέας έχει ως τιμή μια από τις ετικέτες της λίστας, τότε εκτελείται η αντίστοιχη εντολή, αλλιώς τυπώνεται βαθμολογία εκτός ορίων.

γ.

```
var
     arxika_epilogis:char;
.........
case arxika_epilogis of
'Ε': writeln('ΕΙΣΑΓΩΓΗ ');
'Μ': writeln('ΜΕΤΑΒΟΛΗ');
'Δ': writeln('ΔΙΑΓΡΑΦΗ');
'Τ': writeln('ΤΕΛΟΣ ΕΡΓΑΣΙΑΣ -  ΕΞΟΔΟΣ');
else writeln('Λάθος αρχικό επιλογής');
end{case}
```

δ.

```
var i:integer;
     case  i of
          1:  a:=a+1;
          2:  b:=b+1;
          3:  c:=c+1;
     end {case}
```

Η εντολή case επιτρέπει την απαρίθμηση διαφόρων εναλλακτικών λύσεων και την επιλογή μίας εξ αυτών για εκτέλεση.

Ανακεφαλαίωση

Κατανοήσαμε τη σύνταξη των εντολών επιλογής και τον τρόπο χειρισμού τους για τη λήψη απόφασης μέσα σε ένα πρόγραμμα Pascal. Κατανοήσαμε τις διάφορες μορφές σύνταξης της εντολής **if** και τον έλεγχο της συνθήκης που μπορεί να έχει μία ή δύο επιλογές. Κατανοήσαμε την ανάγκη για σύνθετες συνθήκες με περισσότερες επιλογές με τη χρήση εμφωλευμένων δομών if. Τέλος κατανοήσαμε τη χρησιμότητα της εντολής **case** η οποία βοηθάει τη δόμηση ενός προγράμματος και το καθιστά πιο ευανάγνωστο και κατανοητό αντικαθιστώντας πολλά if.

Ερωτήσεις

1. Πώς μπορεί να αλλάξει η σειρά εκτέλεσης των εντολών;
2. Πώς επιτυγχάνεται η εκτέλεση ορισμένων εντολών υπό συνθήκες;
3. Να εξηγήσετε την παρακάτω εντολή και να την τροποποιήσετε ώστε να περιλαμβάνει και την περίπτωση εκείνη στην οποία ο αριθμός είναι πε-

ριτός. Να δώσετε παραδείγματα και τα αποτελέσματα της εκτέλεσης των εντολών.

```
if arithm mod 2 = 0
then writeln(' ο αριθμός είναι άρτιος ')';
```

4. Να εξηγήσετε την παρακάτω εντολή και να δώσετε παραδείγματα και τα αποτελέσματα της εκτέλεσης της εντολής.

```
if diairetis <>0
then pililko:=diaireteos div diairetis
else writeln('διαίρεση με μηδέν δεν είναι επιτρεπτή');
```

5. Να χρησιμοποιήσετε την εντολή if για τον έλεγχο της ηλικίας ενός ψηφοφόρου και να δώσετε το κατάλληλο μήνυμα ανάλογα με την περίπτωση, για ηλικία > 65 , για ηλικία <18 και για 18<ηλικία<65

6. Να χρησιμοποιήσετε την εντολή case για τον έλεγχο της ηλικίας ενός ψηφοφόρου και να δώσετε το κατάλληλο μήνυμα ανάλογα με την περίπτωση, για ηλικία > 65, για ηλικία <18 και για 18<ηλικία<65

7. Να συγκρίνετε την εντολή case με την if και να δώσετε δύο παραδείγματα χρήσης της εντολής case.

8. Να γράψετε την επόμενη κωδικοποίηση χρησιμοποιώντας την εντολή case αντί της if

```
if  n=5
then   tm5:=tm5+1
else   if  n=15
          then  tm15:=tm15+1
          else  if n=25
                   then  tm25:=tm25+1;
```

9. Να γράψετε μία εντολή case η οποία ανάλογα με το χαρακτηρισμό της βαθμολογίας τυπώνει το αντίστοιχο μήνυμα. Αν ο χαρακτηρισμός=1 τύπωσε άριστα, 2 λίαν καλώς, 3 καλώς, 4 σχεδόν καλώς, 5 απορρίπτεται, αλλιώς λάθος χαρακτηρισμός. Να επαναλάβετε τα προηγούμενα με την εντολή if.

10. Να συμπληρώσετε με **Σωστό – Λάθος** τις παρακάτω προτάσεις

    i. Η έκφραση της εντολής case μπορεί να δώσει τιμές τύπου ακέραιου, πραγματικού ή λογικού. _____

    ii. Οι τιμές που παίρνουν οι ετικέτες στην εντολή case μπορεί να έχουν οποιαδήποτε σειρά, αλλά οι ετικέτες δεν επαναλαμβάνονται. _____

    iii. Όλες οι δυνατές τιμές της έκφρασης σε μια εντολή case πρέπει να περιλαμβάνονται στις λίστες τιμών των ετικετών. _____

Ασκήσεις

1. Να γράψετε πρόγραμμα που να ελέγχει το ποσό ανάληψης ενός καταθέτη σε μία Τράπεζα. Σε περίπτωση που η ανάληψη είναι μεγαλύτερη από τις καταθέσεις τυπώνει απαγορευτικό μήνυμα, αλλιώς τυπώνει το ποσό ανάληψης και το υπόλοιπο των καταθέσεων.

2. Να γράψετε πρόγραμμα που να διαβάζει 4 βαθμούς, να βρίσκει το μέσο όρο τους και, αν αυτός είναι μεγαλύτερος από 15, τυπώνει ΕΠΙΤΥΧΩΝ.

3. Η ωριαία αμοιβή εργαζομένου είναι 2000 δρχ. Αν οι ώρες εργασίας είναι περισσότερες από 18, παίρνει επιπλέον υπερωριακή αποζημίωση 1000 δρχ γιά κάθε υπερωριακή ώρα. Να γράψετε πρόγραμμα που να διαβάζει τις ώρες εργασίας και να υπολογίζει τις αποδοχές του εργαζομένου. Τα αποτελέσματα να ακολουθήσουν το υπόδειγμα:

   κανονική αμοιβή .....
   αμοιβή υπερωριών .....
   ---------- ----------
   συνολικές αποδοχές .....

   Υπόδειξη: Να γίνουν δύο τουλάχιστον εκτελέσεις του προγράμματος, μία με υπερωρίες και μία χωρίς.

4. Να μετατρέψετε τα προηγούμενα προγράμματα if -- then--else κάνοντας χρήση της εντολής case.

5. Να γράψετε πρόγραμμα που να διαβάζει το αρχικό ενός ονόματος και να τυπώνει το αντίστοιχο όνομα (ονόματα: Πέτρος, Χριστίνα, Νίκος, 'Αννα, Γιώργος).

6. Να γράψετε πρόγραμμα που να'εμφανίζει menu με τις 4 πράξεις:

   1 πρόσθεση
   2 αφαίρεση
   3 πολλαπλασιασμός
   4 διαίρεση,

   να διαβάζει τον κωδικό της πράξης και τους δύο αριθμούς και να εμφανίζει το αποτέλεσμα.(Διαίρεση με μηδέν δε γίνεται).

7. Να γράψετε πρόγραμμα που να διαβάζει δύο ηλικίες και να δίνει τη διαφορά τους. Η εισαγωγή στοιχείων και τα αποτελέσματα να είναι σύμφωνα με το υπόδειγμα:

   Δεδομένα : 9 6
   Αποτελέσματα:
   Ο Νίκος είναι 9 ετών
   Ο Γιώργος είναι 6 ετών
   Ο Νίκος είναι 3 έτη μεγαλύτερος απ' το Γιώργο

Να γίνουν άλλες δύο εκτελέσεις με δεδομένα 6, 9 και 9, 9.

8. Για να ψηφίσει ένας πολίτης, πρέπει να είναι τουλάχιστον 18 ετών. Αν όμως είναι άνω των 70, δεν υποχρεώνεται να ψηφίσει. Να γράψετε πρόγραμμα που να διαβάζει την ηλικία και να δίνει το κατάλληλο μήνυμα.

Υπόδειξη: Να χρησιμοποιήσετε το δυαδικό τελεστή and. Να γίνουν 3 διαφορετικές εκτελέσεις με αποτελέσματα:

Είναι – – χρονών υποχρεώνεται να ψηφίσει.
Είναι – – χρονών δεν υποχρεώνεται να ψηφίσει.
Είναι – – χρονών δεν μπορεί να ψηφίσει.

ΚΕΦΑΛΑΙΟ 11
ΕΝΤΟΛΕΣ ΕΠΑΝΑΛΗΨΗΣ

Σκοπός κεφαλαίου:

Να κατανοήσουμε τη χρησιμότητα της δομής επανάληψης, που παρέχει ο προγραμματισμός.

Ειδικοί σκοποί:

- ❑ Να κατανοήσουμε τη λειτουργία της εντολής **while** η οποία είναι μία από τις τρεις βασικές δομές του δομημένου προγραμματισμού.
- ❑ Να κατανοήσουμε τη λειτουργία της εντολής **repeat** και πώς αυτή διαφοροποιείται σε σχέση με την while.
- ❑ Να κατανοήσουμε τη λειτουργία της εντολής **for** και πώς αυτή διαφοροποιείται από τις δύο προηγούμενες δομές επανάληψης.

11.1. Η ΕΝΝΟΙΑ ΤΗΣ ΕΠΑΝΑΛΗΨΗΣ

Συχνά, ορισμένοι υπολογισμοί σε ένα πρόγραμμα είναι αναγκαίο να εκτελούνται περισσότερες από μία φορές. Υπάρχουν δύο τύποι επαναλήψεων:

- Οι προκαθορισμένοι, όπου το πλήθος των επαναλήψεων είναι δεδομένο πριν αρχίσουν οι επαναλήψεις.
- Οι μη προκαθορισμένοι ή απροσδιόριστοι, όπου το πλήθος των επαναλήψεων καθορίζεται κατά τη διάρκεια της εκτέλεσης των εντολών του σώματος επανάληψης.

11.2. WHILE

Η εντολή while χρησιμοποιείται για μη προκαθορισμένο αριθμό επαναλήψεων, όπου υπάρχει περίπτωση να μην εκτελεστούν οι επαναλήψεις και όπου ο έλεγχος γίνεται πριν από την εκτέλεση των εντολών επανάληψης.

Μορφή: **while** boolean expression **do** statement
εφόσον λογική έκφραση **εκτέλεσε** εντολή

Ενέργεια: Κάθε φορά, ακόμη και την πρώτη, πριν εκτελεστεί η εντολή που αποτελεί το σώμα της επανάληψης (το statement μετά το do), εξετάζεται η λογική έκφραση η οποία ακολουθεί τη λέξη while και, μόνο σε περίπτωση που δίνει τιμή **true**, εκτελείται η προς επανάληψη εντολή. Αν δώσει τιμή

false, η εντολή που αποτελεί το σώμα της επανάληψης δεν εκτελείται και ο έλεγχος από το while περνάει στην επόμενη εντολή. Το σώμα της επανάληψης είναι μία εντολή. Σε περίπτωση που θέλουμε να εκτελέσουμε περισσότερες από μία εντολές τότε αυτές θα πρέπει να αποτελέσουν μία σύνθετη εντολή με τη χρήση των **begin - end**.

Υπάρχουν δύο τύποι επαναλήψεων. Ο ένας τύπος χρησιμοποιεί κάποιο ***μετρητή επαναλήψεων (count-controlled loop)***. Ο άλλος τύπος χρησιμοποιεί κάποιο γεγονός και η επανάληψη συνεχίζεται μέχρι να συμβεί το γεγονός στη διάρκεια της επανάληψης (***event-controlled loop***).

Στο παράδειγμα που ακολουθεί χρησιμοποιείται ένας μετρητής για τον έλεγχο των επαναλήψεων. Ο μετρητής παίρνει αρχική τιμή έξω από τον κύκλο των επαναλήψεων, και αυξάνει κατά 1 στην τελευταία εντολή μέσα στον κύκλο των επαναλήψεων έτσι ώστε να μπορεί να γίνει ο έλεγχος και να σταματήσουν οι επαναλήψεις, όταν η λογική έκφραση πάρει τιμή ***false*** δηλαδή, όταν ο μετρητής γίνει ίσος με 101.

Παράδειγμα με μετρητή

Για τον έλεγχο των επαναλήψεων (count-controlled loop).

```
metritis:=1;      {αρχική τιμή στη μεταβλητή ελέγχου}
while count <= 100 do   {έλεγχος της επανάληψης}
begin
   **                   {εντολές επανάληψης}
   **                   {εντολές επανάληψης}
   metritis:=metritis+1;   {αύξηση του μετρητή}
 end
```

Παράδειγμα με μεταβλητή

Για τον έλεγχο των επαναλήψεων (event-controlled loop).

Για κάποια επεξεργασία χρειαζόμαστε ημερομηνίες μήνα και ημέρα. Διαβάζουμε την πρώτη ημερομηνία και για τον έλεγχο χρησιμοποιούμε την ίδια μεταβλητή. Η επανάληψη θα σταματήσει όταν η μεταβλητή πάρει τιμή μη παραδεκτή πχ. 31 Φεβρουαρίου. Όταν συμβεί το γεγονός αυτό η επανάληψη θα σταματήσει.

```
read(mina,imera);                {αρχική ημερομηνία}
while not ((month=2) and (day=31)) do {έλεγχος
                                        επανάληψης}
   begin
     *
     *                                 {επεξεργασία}
      read(mina,imera);                {επόμενη ημερομηνία}
   end
```

Το παράδειγμα που ακολουθεί είναι ένα πλήρες πρόγραμμα όπου εφαρμόζονται όσα αναφέραμε παραπάνω. Η επανάληψη των εντολών σταματάει όταν συμβεί το γεγονός " το άθροισμα να γίνει μεγαλύτερο από το δεδομένο όριο".

Παράδειγμα:

```
program sum1;
{Ανάγνωση αριθμών μέχρις ότου το άθροισμά τους γίνει
μεγαλύτερο από ένα δεδομένο αριθμό}
var
orio,ar,sum :real;
begin
     read(orio);
     sum:=0;
     while sum <= orio do
       begin
        readln(ar);
        sum:=sum+ar;
       end;
     writeln(sum:10:4);
end.
```

11.3. REPEAT - UNTIL

Η εντολή repeat χρησιμοποιείται για μη προκαθορισμένο αριθμό επαναλήψεων. Ο έλεγχος για την επανάληψη γίνεται στο τέλος του κύκλου (**loop**) των εντολών που επαναλαμβάνονται. Όπως φαίνεται και από τη μορφή της εντολής που ακολουθεί

Μορφή: **repeat**
εντολή [;
εντολή ;
.
.
.....]
until (boolean expression);

- Οι ορθογώνιες παρενθέσεις δηλώνουν ότι η δεύτερη καθώς και οι επόμενες εντολές επανάληψης μπορεί να μην υπάρχουν.
- Υπάρχει πάντοτε μία τουλάχιστον εντολή επανάληψης.
- Δεν απαιτούνται begin, end γιατί αντί αυτών υπάρχουν οι λέξεις **repeat**, **until**

Επειδή ο έλεγχος επανάληψης γίνεται στο τέλος, οι εντολές επανάληψης εκτελούνται τουλάχιστον μια φορά ακόμη και αν η λογική έκφραση είχε εξ αρχής την τιμή true.

Ενέργεια: Αφού εκτελεστούν για πρώτη φορά οι εντολές που αποτελούν το σώμα της επανάληψης (μεταξύ των repeat και until), εξετάζεται η λογική έκφραση η οποία ακολουθεί την λέξη until, και μόνο σε περίπτωση που δίνει τιμή false εκτελούνται και πάλι οι εντολές επανάληψης. Αν δώσει τιμή true, ο έλεγχος από το repeat, περνάει στην επόμενη του repeat-until εντολή. Όπως

φαίνεται από τον ορισμό της repeat οι εντολές εκτελούνται απαραίτητα τουλάχιστον μία φορά ακόμη και όταν η λογική τιμή είναι αληθής σε αντίθεση με την while, που δεν εκτελούνται ποτέ, αν η λογική έκφραση έχει τιμή false.

Ακολουθεί μια σύγκριση των εντολών repeat until και while do πρώτα για επαναλήψεις ελεγχόμενες από μετρητή και στη συνέχεια για επαναλήψεις ελεγχόμενες από κάποιο γεγονός.

Παράδειγμα με μετρητή

Για τον έλεγχο των επαναλήψεων (count-controlled loop).

```
metritis:=1;
while metritis <= 100 do
begin
   εντολή1
   εντολή2
      ...
    metritis:=metritis+1;

end
```

```
metritis:=1;

repeat
εντολή1
   εντολή2
      ...
metritis:=metritis+1
until metritis > 100
```

Στο **while** ο έλεγχος εκτελείται πριν από τον κύκλο των εντολών επανάληψης.

Στο **repeat** η επανάληψη των εντολών συνεχίζεται όσο η λογική έκφραση έχει τιμή false, ενώ στο **while** η επανάληψη των εντολών συνεχίζεται όσο η λογική έκφραση έχει τιμή true..

Παραδείγματα:

```
program sum2;
{Ανάγνωση αριθμών μέχρις ότου το
άθροισμά τους γίνει μεγαλύτερο
από ένα δεδομένο αριθμό}
var
orio,ar,sum :real;
begin
   read(orio);
   sum:=0;
   repeat
      readln(ar);
      sum:=sum+ar;
   until  sum > orio
   writteln(sum:10:4);
end.
```

```
Program sum2;
{Ανάγνωση αριθμών μέχρις ότου το
άθροισμά τους γίνει μεγαλύτερο
από ένα δεδομένο αριθμό}


var
orio,ar,sum :real;

begin
   read(orio);
   sum:=0;
   while sum <= orio do
      begin
         readln(ar);
         sum:=sum+ar;
      end;
   writeln(sum:10:4);
end.
```

Στο προηγούμενο παράδειγμα και στο repeat και στο while η επανάληψη σταματάει, όταν το άθροισμα ***(sum)*** γίνει μεγαλύτερο από το όριο ***(orio)*** Οι λογικές εκφράσεις είναι συμπληρώματα η μία της άλλης (αντίθετες) (***sum > orio*** και ***sum*** <= orio).

Εδώ πρέπει να τονιστεί ότι το **while** χρησιμοποιεί τη λογική έκφραση για να συνεχίσει την επανάληψη, ενώ το **repeat/until** για να τη σταματήσει.

Επιλέγουμε το repeat/until όταν οι εντολές επανάληψης εκτελούνται οπωσδήποτε τουλάχιστον μια φορά αλλιώς επιλέγουμε το while.

Αν έχουμε τη δυνατότητα να χρησιμοποιήσουμε και τα δύο πρέπει να επιλέξουμε εκείνο που αντιπροσωπεύει σημασιολογικά το πρόβλημα. Αν το πρόβλημα αναφέρεται στη συνέχεια της επανάληψης καλύτερα είναι να χρησιμοποιήσουμε το while. Αν το πρόβλημα αναφέρεται στη διακοπή της επανάληψης καλύτερα είναι να χρησιμοποιήσουμε το repeat/until. Αν υπάρχει αμφιβολία καλύτερα να χρησιμοποιούμε το While.

11.4. FOR

Η εντολή **for** σχεδιάστηκε για την απλοποίηση του ελέγχου των επαναλήψεων με μετρητή (**count-controlled loop**).

Η εντολή **for** metritis **:=** 1 **to** n **do**
statement; ***σημαίνει:***

- βάλε 1 στη μεταβλητή ελέγχου (metritis).
- αν το n είναι μικρότερο από το 1 τότε δεν εκτελείται ο κύκλος των εντολών επανάληψης

αλλιώς εκτελείται ο κύκλος των εντολών επανάληψης

και ο μετρητής αυξάνεται κατά 1.

- ο κύκλος επαναλαμβάνεται μέχρι ο μετρητής να γίνει μεγαλύτερος από n

Η προηγούμενη εντολή **for** εμφανίζεται μαζί με την ισοδύναμή της while στον πίνακα που ακολουθεί.

```
for metritis := 1 to n do
    statement;
```

```
metritis:=1;
while metritis <= n do
   begin
     statement;
     i:=i+1;
   end;
```

Υπάρχουν δύο μορφές της εντολής αυτής στην Pascal, μία που η μεταβλητή αυξάνεται (με τη χρήση του **to**) και μία που η μεταβλητή ελαττώνεται (με τη χρήση του **downto**).

Μορφή:

for variable identifier: = expression **to** expression **do**
statement;

για μεταβλητή ελέγχου από αρχική τιμή **μέχρι** τελική τιμή **εκτέλεσε** εντολή

Ενέργεια: Η **εντολή** που ακολουθεί το do, που μπορεί να είναι **απλή** ή **σύνθετη** εντολή, εκτελείται μέχρις ότου η μεταβλητή ελέγχου αποκτήσει τιμή ίση με την τελική τιμή. Η αύξηση της μεταβλητής ελέγχου γίνεται με βήμα τη μονάδα (1). Αν η αρχική τιμή είναι μεγαλύτερη από την τελική τιμή, τότε οι επαναλαμβανόμενες εντολές που ακολουθούν το do δεν εκτελούνται.

Αν η μεταβλητή ελέγχου αρχίζει από τη μεγαλύτερη τιμή μέχρι τη μικρότερη, τότε η μορφή της εντολής είναι η παρακάτω:

for variable identifier: = expression **downto** expression **do**

statement;

για μεταβλητή ελέγχου από αρχική τιμή **μέχρι** τελική τιμή **εκτέλεσε** εντολή

Ενέργεια: Η εντολή που ακολουθεί το do εκτελείται, μέχρις ότου η μεταβλητή ελέγχου αποκτήσει τιμή μικρότερη από την τελική τιμή. Η ελάττωση της μεταβλητής ελέγχου γίνεται με βήμα 1. Αν η αρχική τιμή είναι μικρότερη από την τελική τιμή, τότε οι εντολές που ακολουθούν το do δεν εκτελούνται.

Στον πίνακα που ακολουθεί ο αριθμός τωγ επαναλήψεων είναι ίδιος και για τις δύο μορφές της εντολής for

```
for metritis := katwtero_orio to
                anwtero_orio do
   statement;
```

```
for metritis := anwtero_orio
        downto katwtero _orio do
   statement;
```

Η αρχική και η τελική τιμή του κύκλου for μπορεί να είναι μια έκφραση οποιουδήποτε διατεταγμένου τύπου. Έτσι, εκτός από ακέραιες τιμές μπορούμε να χρησιμοποιήσουμε μεταβλητές με τιμές τύπου χαρακτήρα char ή λογικού τύπου (boolean).

Με το παράδειγμα που ακολουθεί τυπώνουμε το Αγγλικό αλφάβητο.

```
for letter:= 'A ' to 'Z' do
    write(letter);
```

Η εντολή for όπως και άλλες μορφές επανάληψης μπορεί να είναι και σε φωλιά -nested (η μία μέσα στην άλλη). Στον πίνακα που ακολουθεί υπάρχει ένα παράδειγμα φωλιάς for και τα αποτελέσματά του.

Παράδειγμα:

```
for teleytaio_gramma:='A' to 'G' do
   begin
     for typwse_gramma:= 'A ' to
          teleytaio_gramma ' do
            write(letter);
   end;
```

A
AB
ABC
ABCD
ABCDE
ABCDEF
ABCDEFG

Η εντολή for είναι πολύ εύχρηστη. Πρέπει να θυμόμαστε ότι έχει σχεδιαστεί για επαναλήψεις ελεγχόμενες με μετρητή, για τον οποίο γνωρίζουμε την αρχική και την τελική τιμή. Για αποτελεσματική χρήση της εντολής:

- Η μεταβλητή ελέγχου δεν πρέπει να αλλάζει τιμές μέσα στον κύκλο της επανάληψης. Μπορεί να εμφανίζεται σε μία έκφραση, αλλά όχι στο αριστερό μέλος της εντολής αντικατάστασης, οπότε αλλάζει η τιμή της
- Η μεταβλητή ελέγχου μεταβάλλεται κατά βήματα παίρνοντας την επόμενη ή την προηγούμενη τιμή από το πεδίο τιμών της. Αν η μεταβλητή είναι ακέραιος, τότε το βήμα είναι 1. Αν χρειαζόμαστε άλλο βήμα, επιλέγουμε την εντολή while.
- Μετά το τέλος της επανάληψης η μεταβλητή ελέγχου έχει απροσδιόριστη τιμή. Αν τη χρησιμοποιήσουμε ή θα πάρουμε μήνυμα λάθους ή λάθος αποτελέσματα ανάλογα με το μεταφραστικό πρόγραμμα (compiler) που χρησιμοποιούμε.
- Ο κύκλος επανάληψης εκτελείται με τη μεταβλητή ελέγχου να παίρνει τιμές την αρχική, την τελική και τα ενδιάμεσα βήματα. Αν η αρχική τιμή είναι ίση με την τελική, ο κύκλος εκτελείται μια μόνο φορά. Στην περίπτωση το αν η αρχική τιμή είναι μεγαλύτερη από την τελική, ο κύκλος δεν εκτελείται. Στην περίπτωση downto αν η αρχική τιμή είναι μικρότερη από την τελική τιμή, ο κύκλος δεν εκτελείται.

Παράδειγμα

```
program sum3;
use wincrt;
{Ανάγνωση αριθμών και υπολογισμός του αθροίσματός
τους μέχρις ότου το πλήθος τους γίνει μεγαλύτερο από
ένα δεδομένο αριθμό}
var
ar,sum :real;
orio:integer;
begin
    read(orio);
    sum:=0;
    for k:=1 to orio do
    begin
    readln(ar);
    sum:=sum+ar;
    end;
    writeln(sum:10:4);
end.
```

Ανακεφαλαίωση

Κατανοήσαμε τη σύνταξη και τον τρόπο χειρισμού των εντολών επανάληψης. Διακρίναμε τρεις διαφορετικές εντολές επανάληψης και κατανοήσαμε τις διαφορές μεταξύ τους και την ανάγκη ύπαρξη τους μέσα από χαρακτη-

ριστικά παραδείγματα. Οι εντολές **while** και **repeat** χρησιμοποιούνται για μη προκαθορισμένο αριθμό επαναλήψεων και στις δύο ο έλεγχος για την επανάληψη γίνεται μέσα από μία συνθήκη. Στη περίπτωση της **while** οι εντολές επανάληψης εκτελούνται εφόσον το αποτέλεσμα της συνθήκης είναι αληθές ενώ στη **repeat** οι εντολές επανάληψης εκτελούνται όσο το αποτέλεσμα της συνθήκης είναι ψευδές. Σημειώνεται ότι στη **repeat** οι εντολές θα εκτελεστούν τουλάχιστον μία φορά διότι ο έλεγχος της συνθήκης γίνεται μετά το σώμα των εντολών επανάληψης. Η εντολής **for** χρησιμοποιείται για προκαθορισμένο αριθμό επαναλήψεων.

Ερωτήσεις

1. Να δώσετε τη μορφή της εντολής επανάληψης while, να εξηγήσετε τη λειτουργία της και να δώσετε δύο δικά σας παραδείγματα.
2. Να δώσετε τη μορφή της εντολής επανάληψης repeat until, να εξηγήσετε τη λειτουργία της και να δώσετε δύο δικά σας παραδείγματα.
3. Να δώσετε τη μορφή της εντολής επανάληψης for, να εξηγήσετε τη λειτουργία της και να δώσετε δύο δικά σας παραδείγματα.
4. Να συγκρίνετε τις εντολές επανάληψης και να αναφέρετε ποιες είναι οι ομοιότητες και ποιες οι διαφορές τους.
5. Να δώσετε τα αποτελέσματα από την εκτέλεση των παρακάτω εντολών

```
arithmos:=1;
  while arithmos < 11 do
  begin
    arithmos:=arithmos+1;
    write(arithmos:5);
  end;
```

6. Να αλλάξετε τη σειρά των εντολών της προηγούμενης ερώτησης ώστε το αποτέλεσμα να είναι η εκτύπωση των αριθμών 1 έως και 10.
7. Να χρησιμοποιήσετε τις εντολές επανάληψης repeat until και for ώστε να έχετε τα ίδια αποτελέσματα με την ερώτηση 6.
8. Πόσες επαναλήψεις θα έχουμε από την εκτέλεση των παρακάτω εντολών;

```
telos:= false;
  while not telos  do
  begin
    arithmos:=arithmos+2;
    if arithmos > 100
    then telos :=true;
  end;
```

9. Να χρησιμοποιήσετε τις εντολές επανάληψης repeat until και for, ώστε να έχετε τον ίδιο αριθμό επαναλήψεων με την ερώτηση 8.

10. Να δώσετε τα αποτελέσματα από την εκτέλεση των παρακάτω εντολών και να εξηγήσετε τη διαφορά που υπάρχει στη χρήση των μεταβλητών athroisma και metritis

```
athroisma:=0;
metritis :=0;
while metritis < 10  do
begin
  athroisma:=athroisma + metritis;
  writeln(metritis:10,athroisma:10);
  metritis:= metritis + 1;
end;
```

Να τροποποιήσετε τον προηγούμενο κώδικα ώστε, να δίνει τα ίδια αποτελέσματα με τη χρήση των εντολών repeat until και for

Ασκήσεις

1. Να γράψετε ένα πρόγραμμα που να υπολογίζει τη μικρότερη δύναμη του 2 που είναι μεγαλύτερη από ένα δεδομένο αριθμό και να τυπώνει ποια είναι η δύναμη αυτή.

   Τα αποτελέσματα να δοθούν με τη μορφή:
   Η μικρότερη δύναμη του 2
   μεγαλύτερη από τον αριθμό .....
   είναι .....
   δηλ. είναι το 2 στην ..... δύναμη

Υπόδειξη: Η ασκηση να λυθεί με δύο τρόπους (με χρήση WHILE και REPEAT-UNTIL).

2. Να γράψετε πρόγραμμα που να υπολογίζει το μικρότερο αριθμό Fibonacci που δεν είναι μικρότερος από ένα προκαθορισμένο όριο. Κάθε αριθμός της σειράς Fibonacci, εκτός από τους δύο πρώτους που είναι 0 και 1, σχηματίζεται από το άθροισμα των δύο προηγουμένων του.

Υποδείξεις:

   Να χρησιμοποιηθεί η εντολή while.
   Τα αποτελέσματα να δοθούν με τη μορφή:
   αποτελέσματα:
   ο μικρότερος αριθμός Fibonacci
   που δεν είναι μικρότερος από .....
   είναι ο αριθμός .....

3. Να γράψετε πρόγραμμα που να διαβάζει:

   α. το πλήθος των εργατών μίας εταιρείας

   β. πόσες είναι οι υποχρεωτικές ώρες εργασίας (κάθε επιπλέον ώρα είναι υπερωρία)

γ. την ωριαία αμοιβή

δ. την επιπλέον αμοιβή γιά κάθε υπερωριακή ώρα

ε. τις ώρες εργασίας κάθε εργάτη

και να υπολογίζει την αποζημίωση κάθε εργάτη.

Υποδείξεις: Να χρησιμοποιηθεί η εντολή while. Τα αποτελέσματα να δοθούν με τη μορφή:

πλήθος εργατών: - -
υποχρεωτικές ώρες εργασίας: - -
ωριαία αποζημίωση : - -
πρόσθετη υπερωριακή αποζημ: - -
αποδοχές εργατών

Η εκτύπωση να γίνει με τη μορφή:

α/α	ώρες εργ.	αποζημ.	ώρες υπερ.	υπερ. αποζ.	σύνολο
1					
2					
...	...	.....	...	.....	.....

Να δοθεί δυνατότητα να σταματήσει η επανάληψη, σε περίπτωση που θα δοθεί αριθμός αρνητικός για ώρες εργασίας ενός εργάτη.

4. Να τροποποιηθεί η προηγούμενη άσκηση, ώστε το πρόγραμμα να λειτουργεί και με την εντολή repeat.
5. Να γράψετε πρόγραμμα που να διαβάζει το πλήθος των μαθητών και το βαθμό του καθενός και να τυπώνει ένα χαρακτηρισμό ανάλογα με το βαθμό του.

   (9 < βαθμός < = 12 μέτρια
   12 < βαθμός < = 15 καλά
   15 < βαθμός < = 18 πολύ καλά
   18 < βαθμός < = 20 άριστα)

   Να χρησιμοποιήσετε την εντολή repeat και δυαδικούς τελεστές (and...). Στο τέλος να τυπωθεί το πλήθος κάθε κατηγορίας, δηλ. άριστα, πολύ καλά κλπ. καθώς και το σύνολο όλων των μαθητών.
6. Να γράψετε πρόγραμμα που να διαβάζει τις 4 πλευρές και μία γωνία ενός τετραπλεύρου και να υπολογίζει το είδος του (τετράγωνο, ρόμβο, ορθογώνιο, παραλληλόγραμμο, τυχαίο). Να χρησιμοποιήσετε την εντολή repeat και δυαδικούς τελεστές.