



ΑΝΑΠΤΥΞΗ ΕΦΑΡΜΟΓΩΝ ΣΕ ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΟ ΠΕΡΙΒΑΛΛΟΝ Γ' ΛΥΚΕΙΟΥ

ΣΗΜΕΙΩΣΕΙΣ Α.Ε.Π.Π.

Μάκης Θωμάς, MSc.
Full Stack Developer

Πίνακας περιεχομένων

ΑΝΑΠΤΥΞΗ ΕΦΑΡΜΟΓΩΝ ΣΕ ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΟ ΠΕΡΙΒΑΛΛΟΝ	3
ΔΟΜΗ ΑΛΓΟΡΙΘΜΟΥ	3
ΕΙΔΗ ΜΕΤΑΒΛΗΤΩΝ	3
ΑΡΙΘΜΗΤΙΚΟΙ ΤΕΛΕΣΤΕΣ	4
ΕΝΤΟΛΗ ΕΚΧΩΡΗΣΗΣ	5
ΕΝΤΟΛΗ ΕΜΦΑΝΙΣΕ-ΕΚΤΥΠΩΣΕ	5
ΕΝΤΟΛΗ ΔΙΑΒΑΣΕ	6
ΣΥΓΚΡΙΤΙΚΟΙ ΤΕΛΕΣΤΕΣ	6
ΛΟΓΙΚΟΙ ΤΕΛΕΣΤΕΣ	7
ΤΕΛΕΣΤΕΣ DIV ΚΑΙ MOD	8
ΥΠΟΛΟΓΙΣΜΟΣ ΤΙΜΗΣ ΑΠΟ ΕΝΑ ΠΟΣΟΣΤΟ	10
ΥΠΟΛΟΓΙΣΜΟΣ ΠΟΣΟΣΤΟΥ ΑΠΟ ΜΙΑ ΤΙΜΗ	12
ΕΝΤΟΛΗ ΑΝ	12
ΔΙΑΓΡΑΜΜΑΤΑ ΡΟΗΣ	14
ΚΡΙΤΗΡΙΑ ΑΛΓΟΡΙΘΜΩΝ	16
ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΣΕ ΓΛΩΣΣΑ	18
ΔΟΜΗ ΕΝΟΣ ΠΡΟΓΡΑΜΜΑΤΟΣ	18
ΣΤΑΘΕΡΕΣ ΤΙΜΕΣ	19
ΕΙΔΗ ΜΕΤΑΒΛΗΤΩΝ	19
ΣΦΑΛΜΑΤΑ	20
ΑΣΚΗΣΕΙΣ	21
ΕΝΤΟΛΕΣ ΕΠΑΝΑΛΗΨΗΣ	24
ΕΝΤΟΛΗ ΓΙΑ	24
ΕΝΤΟΛΗ ΟΣΟ	26
ΕΝΤΟΛΗ ΜΕΧΡΙΣ_ΟΤΟΥ	27
ΥΠΟΛΟΓΙΣΜΟΣ ΕΛΑΧΙΣΤΟΥ, ΜΕΓΙΣΤΟΥ	30
ΥΠΟΛΟΓΙΣΜΟΣ ΑΘΡΟΙΣΜΑΤΟΣ, ΓΙΝΟΜΕΝΟΥ	32
ΥΠΟΛΟΓΙΣΜΟΣ ΠΛΗΘΟΥΣ	33
ΥΠΟΛΟΓΙΣΜΟΣ ΜΕΣΟΥ ΟΡΟΥ	34
ΑΣΚΗΣΕΙΣ	35
ΜΟΝΟΔΙΑΣΤΑΤΟΙ ΠΙΝΑΚΕΣ	36
ΑΝΑΖΗΤΗΣΗ	40
ΤΑΞΙΝΟΜΗΣΗ	44
ΑΣΚΗΣΕΙΣ	48

ΔΙΣΔΙΑΣΤΑΤΟΙ ΠΙΝΑΚΕΣ.....	50
ΑΣΚΗΣΕΙΣ	57
ΥΠΟΠΡΟΓΡΑΜΜΑΤΑ	59
ΚΥΡΙΩΣ ΠΡΟΓΡΑΜΜΑ.....	60
ΣΥΝΑΡΤΗΣΕΙΣ.....	60
ΔΙΑΔΙΚΑΣΙΕΣ.....	62
ΠΑΡΑΜΕΤΡΟΙ.....	64
ΔΙΑΦΟΡΕΣ ΣΥΝΑΡΤΗΣΕΩΝ ΜΕ ΔΙΑΔΙΚΑΣΙΕΣ	65
ΠΙΝΑΚΕΣ ΩΣ ΠΑΡΑΜΕΤΡΟΙ ΣΕ ΣΥΝΑΡΤΗΣΕΩΝ ΜΕ ΔΙΑΔΙΚΑΣΙΕΣ	65
ΣΥΓΚΡΙΣΗ ΔΙΑΔΙΚΑΣΙΩΝ ΚΑΙ ΣΥΝΑΡΤΗΣΕΩΝ	66
ΑΣΚΗΣΕΙΣ	68
ΕΡΓΑΣΙΕΣ	70
ΕΡΓΑΣΙΑ 1	70
ΕΡΓΑΣΙΑ 2	71
ΕΡΓΑΣΙΑ 3	73

ΑΝΑΠΤΥΞΗ ΕΦΑΡΜΟΓΩΝ ΣΕ ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΟ ΠΕΡΙΒΑΛΛΟΝ

ΔΟΜΗ ΑΛΓΟΡΙΘΜΟΥ

Αλγόριθμος <Όνομα_Αλγορίθμου>

<Κύριο μέρος>

Τέλος <Όνομα_Αλγορίθμου>

Ένας αλγόριθμος αρχίζει με τη λέξη **Αλγόριθμος** ακολουθούμενη από μια οποιαδήποτε λέξη που δηλώνει το όνομα του αλγορίθμου. Στη συνέχεια ακολουθεί το κύριο μέρος του αλγορίθμου και ο αλγόριθμος ολοκληρώνεται με την λέξη **Τέλος** ακολουθούμενη από το όνομα του αλγορίθμου. Στο παράδειγμα παρακάτω, φαίνεται η δομή ενός αλγορίθμου.

Παράδειγμα1:

Αλγόριθμος παράδειγμα_1

Εμφάνισε 'Διάβσε ένα νούμερο:'

Διάβασε x

$z \leftarrow x / 100$

Εμφάνισε 'Η διαίρεση του 'x, 'με το 100 είναι:', z

Τέλος παράδειγμα_1

ΕΙΔΗ ΜΕΤΑΒΛΗΤΩΝ

Υπάρχουν τριών ειδών μεταβλητές, οι ΑΡΙΘΜΗΤΙΚΕΣ, οι ΑΛΦΑΡΙΘΜΗΤΙΚΕΣ, και οι ΛΟΓΙΚΕΣ.

Οι **ΑΡΙΘΜΗΤΙΚΕΣ** μεταβλητές είναι μεταβλητές που μπορούν να πάρουν **μόνο** αριθμούς, πχ

$x \leftarrow 3$

$y \leftarrow 46.5$

Οι **ΑΛΦΑΡΙΘΜΗΤΙΚΕΣ** είναι μεταβλητές που περιέχουν κάποιο αλφαριθμητικό (δηλαδή χαρακτήρες με αριθμούς μαζί), πχ

$a \leftarrow$ 'Σήμερα έχουμε 25 βαθμούς Κελσίου'

$c \leftarrow$ 'Το Επώνυμο μου είναι Δημητρίου'.

Το περιεχόμενο των ΑΛΦΑΡΙΘΜΗΤΙΚΩΝ μεταβλητών, πρέπει να περιλαμβάνεται ανάμεσα σε ' '.

Τέλος οι **ΛΟΓΙΚΕΣ** μεταβλητές είναι μεταβλητές που μπορούν να πάρουν την τιμή **ΑΛΗΘΗΣ** ή τη τιμή **ΨΕΥΔΗΣ**, πχ

$g \leftarrow \text{ΑΛΗΘΗΣ}$

$r \leftarrow \text{ΨΕΥΔΗΣ}$.

Οι ΛΟΓΙΚΕΣ μεταβλητές χρησιμοποιούνται κυρίως σε συνθήκες και επαναληπτικές εντολές.

ΑΡΙΘΜΗΤΙΚΟΙ ΤΕΛΕΣΤΕΣ

ΑΡΙΘΜΗΤΙΚΟΙ ΤΕΛΕΣΤΕΣ		ΣΥΝΑΡΤΗΣΕΙΣ	
+	Πρόσθεση	HM(x)	Ημίτονο
-	Αφαίρεση	ΣΥΝ(x)	Συνημίτονο
*	Πολλαπλασιασμός	ΕΦ(x)	Εφαπτομένη
/	Διαίρεση	T_P(x)	Τετραγωνική ρίζα
^	Ύψωση σε δύναμη	ΛΟΓ(x)	Λογάριθμος
DIV	Πηλίκo Ακέραιας Διάρεσης	E(x)	e^x
MOD	Υπόλοιπο Ακέραιας Διάρεσης	A_M(x)	Ακέραιο μέρος του x
		A_T(x)	Απόλυτη τιμή του x

Π.χ.

$x \leftarrow 2^3$ αντιστοιχεί σε ($x=2^3$)

$y \leftarrow (3*5)-2$ αντιστοιχεί σε ($y=(3*5)-2$)

$z \leftarrow T_P(10)$ (όπου θα γίνει υπολογισμός της τετραγωνικής ρίζας του 10 και θα καταχωρηθεί στη μεταβλητή z)

$c \leftarrow A_M(4,15)$ (στη μεταβλητή c θα αποθηκευτεί το ακέραιο μέρος του 4.15 δηλαδή το 4.

Η προτεραιότητα των πράξεων είναι ίδια με την προτεραιότητα των εξισώσεων των μαθηματικών. Δηλαδή οι πράξεις εκτελούνται με την εξής σειρά:

α. οι πράξεις των παρενθέσεων,

β. η ύψωση σε δύναμη

γ. ο πολλαπλασιασμός η διαίρεση, DIV και MOD

δ. έπειτα η πρόσθεση και η αφαίρεση.

ε. στη συνέχεια εκτελούνται οι συγκριτικοί τελεστές (<, >, =...)

στ. και τέλος οι λογικές πράξεις (ΚΑΙ, ΟΧΙ, Η)

ΕΝΤΟΛΗ ΕΚΧΩΡΗΣΗΣ

Μεταβλητή \leftarrow έκφραση

Η εντολή \leftarrow θέτει τιμές σε μεταβλητές. Υπολογίζει την τιμή της έκφρασης που είναι στα δεξιά και την καταχωρεί στη μεταβλητή που είναι στα αριστερά.

Π.χ.

$\alpha \leftarrow 3$

ημέρα \leftarrow 'Τρίτη'

ΕΝΤΟΛΗ ΕΜΦΑΝΙΣΕ-ΕΚΤΥΠΩΣΕ

ΕΜΦΑΝΙΣΕ <Κείμενο>

Η εντολή **ΕΜΦΑΝΙΣΕ** χρησιμοποιείται για την εμφάνιση κειμένου ή του αποτελέσματος μιας μεταβλητής. Χρησιμοποιείται όταν θέλουμε να εμφανίσουμε το αποτέλεσμα ενός αλγορίθμου. π.χ.

ΜΟ \leftarrow 10

ΕΜΦΑΝΙΣΕ 'Ο Μέσος όρος είναι'

ΕΜΦΑΝΙΣΕ ΜΟ

Μπορούμε επίσης να εμφανίσουμε συνεχόμενες τιμές κειμένου και μεταβλητών χωρίζοντας τες με κόμμα (,) και τοποθετώντας το κείμενο ανάμεσα σε ' '.

$x \leftarrow 25$

ΕΜΦΑΝΙΣΕ 'Σήμερα έχουμε', x , 'βαθμούς Κελσίου'

Το οποίο θα εμφανίσει 'Σήμερα έχουμε 25 βαθμούς Κελσίου'.

ΕΝΤΟΛΗ ΔΙΑΒΑΣΕ

ΔΙΑΒΑΣΕ <μεταβλητή>

Με την εντολή ΔΙΑΒΑΣΕ το πρόγραμμα σταματάει την εκτέλεση του, περιμένοντας από τον χρήστη να πληκτρολογήσει κάτι (τιμή, κείμενο) και στη συνέχεια αποθηκεύει αυτό που πληκτρολόγησε ο χρήστης μέσα στη <μεταβλητή>. Πχ

ΕΜΦΑΝΙΣΕ 'Δώσε την ηλικία σου:'

ΔΙΑΒΑΣΕ Ηλικία

Σ' αυτή τη περίπτωση τυπώνεται στην οθόνη η φράση 'Δώσε την ηλικία σου:' και το πρόγραμμα περιμένει μέχρις ότου ο χρήστης πληκτρολογήσει ένα νούμερο (την ηλικία) το οποίο στη συνέχεια αποθηκεύει στη μεταβλητή "Ηλικία". Τις περισσότερες φορές η εντολή ΔΙΑΒΑΣΕ χρησιμοποιείται σε συνδυασμό με την εντολή ΕΜΦΑΝΙΣΕ, προκειμένου να διευκρινίζεται στο χρήστη τί πρέπει να πληκτρολογήσει.

ΣΥΓΚΡΙΤΙΚΟΙ ΤΕΛΕΣΤΕΣ

Το \leftarrow στο προγραμματισμό παίζει το ρόλο του μαθηματικού $=$. Παρόλα αυτά το $=$ χρησιμοποιείται και στο προγραμματισμό, και παίζει το ρόλο του τελεστή σύγκρισης.

<μέλος 1> = <μέλος 2>

Το $=$ **ελέγχει** αν το <μέλος 1> είναι ίσο με το <μέλος 2>. Πχ αν γράψουμε $5=3$ ο ρόλος του $=$ είναι να ελέγξει αν ότι γράψαμε μπροστά από το $=$ (δηλαδή το 5) είναι ίσο με αυτό που γράψαμε πίσω από το $=$ (δηλαδή 3).

Εκτός από το $=$ υπάρχουν και άλλοι συγκριτικοί τελεστές

Συγκριτικοί τελεστές	Έννοια
$=$	ισότητα
$<$	μικρότερο
$<=$	μικρότερο ή ίσο
$>$	μεγαλύτερο
$>=$	μεγαλύτερο ή ίσο
$<>$	διάφορο

Αυτή η έκφραση που προκύπτει από τη χρήση των συγκριτικών τελεστών ονομάζεται **συνθήκη**.

ΛΟΓΙΚΟΙ ΤΕΛΕΣΤΕΣ

Το αποτέλεσμα ενός συγκριτικού τελεστή μπορεί να είναι είτε ΑΛΗΘΕΣ είτε ΨΕΥΔΕΣ και τίποτα άλλο.

Πχ. Αν γράψουμε για παράδειγμα την συνθήκη

$$4*2 = 10-2$$

Ελέγχουμε αν το αποτέλεσμα αριστερά από το = (δηλαδή 8) , είναι ίσο με το αποτέλεσμα δεξιά από το =. Στη περίπτωση που ο **έλεγχος** που κάνουμε ισχύει τότε το αποτέλεσμα αυτής της πράξης είναι **ΑΛΗΘΕΣ** διαφορετικά είναι **ΨΕΥΔΕΣ**. Στη περίπτωση που ακολουθεί το αποτέλεσμα είναι ΨΕΥΔΕΣ

$$4*2 <> 10-2$$

Αν τώρα θέλουμε να ισχύουν ταυτόχρονα περισσότερες από μια συνθήκες χρησιμοποιούμε τους λογικούς τελεστές προκειμένου να ενώσουμε τις συνθήκες. Οι λογικοί τελεστές είναι:

Λογικοί τελεστές	Έννοια
<έκφραση1> ΚΑΙ <έκφραση2>	Ισχύουν ταυτόχρονα και η <έκφραση1> και η <έκφραση2>
<έκφραση1> Η <έκφραση2>	Ισχύει ή η <έκφραση1> ή <έκφραση2>
ΟΧΙ <έκφραση1>	Ισχύει το αντίθετο από την <έκφραση1>

Οι λογικοί τελεστές με παραδείγματα

Αν $x < 10$ και $y < 3$ τότε η λογική έκφραση $x > 4$ **ΚΑΙ** $y < 5$ είναι ΑΛΗΘΗΣ, διότι ισχύουν ταυτόχρονα και το $x > 4$ και το $y < 5$, εφόσον το x είναι 10 και το y είναι 3.

Αν $a < 7$, $b < 3$ και $c < 6$ τότε η λογική έκφραση $a > b$ **Η** $c > 6$ είναι ΑΛΗΘΗΣ, διότι η έκφραση $a > b$ ισχύει (είναι δηλαδή ΑΛΗΘΗΣ) και η έκφραση $c > 6$ δεν ισχύει (είναι δηλαδή ΨΕΥΔΗΣ). Οπότε εφόσον ισχύει μόνο μια από τις 2 εκφράσεις, τότε όλη η έκφραση επειδή έχει το λογικό τελεστή **Η** είναι ΑΛΗΘΗΣ.

Αν όνομα ← 'Πέτρος' τότε η λογική έκφραση **ΟΧΙ**(όνομα = 'Πέτρος') είναι ΨΕΥΔΗΣ, διότι η έκφραση όνομα = 'Πέτρος' είναι ΑΛΗΘΗΣ, και με τον τελεστή ΟΧΙ μπροστά το αποτέλεσμα είναι το αντίθετο της έκφρασης όνομα = 'Πέτρος', δηλαδή ΨΕΥΔΗΣ.

Παρακάτω φαίνονται αναλυτικά όλες οι περιπτώσεις των τελεστών.

Πίνακας Αληθείας

<έκφραση1>	<έκφραση2>	<έκφραση1> ΚΑΙ <έκφραση2>	<έκφραση1> Η <έκφραση2>	ΟΧΙ <έκφραση1>
ΑΛΗΘΗΣ	ΑΛΗΘΗΣ	ΑΛΗΘΗΣ	ΑΛΗΘΗΣ	ΨΕΥΔΗΣ
ΑΛΗΘΗΣ	ΨΕΥΔΗΣ	ΨΕΥΔΗΣ	ΑΛΗΘΗΣ	ΨΕΥΔΗΣ
ΨΕΥΔΗΣ	ΑΛΗΘΗΣ	ΨΕΥΔΗΣ	ΑΛΗΘΗΣ	ΑΛΗΘΗΣ
ΨΕΥΔΗΣ	ΨΕΥΔΗΣ	ΨΕΥΔΗΣ	ΨΕΥΔΗΣ	ΑΛΗΘΗΣ

ΤΕΛΕΣΤΕΣ DIV ΚΑΙ MOD

Οι τελεστές DIV και MOD χρησιμοποιούνται για να πάρουμε τα αποτελέσματα μια ακέραιας διαίρεσης. Ο τελεστής **DIV** μας δίνει το **ακέραιο πηλίκο** της διαίρεσης μεταξύ δυο **ακέραιων αριθμών**, ενώ ο τελεστής **MOD** μας δίνει το **ακέραιο υπόλοιπο**.

$$\begin{array}{c|c} x & y \\ \hline x \text{ MOD } y & x \text{ DIV } y \end{array}$$

Παραδείγματα:

- | | |
|-------------------|-------------------|
| I. 14 DIV 3 = 4 | II. 14 MOD 3 = 2 |
| III. 7 DIV 2 = 3 | IV. 7 MOD 2 = 1 |
| V. 8 DIV 2 = 4 | VI. 8 MOD 2 = 0 |
| VII. 3 DIV 7 = 0 | VIII. 3 MOD 7 = 3 |
| IX. 83 DIV 10 = 8 | |

ΠΡΟΣΟΧΗ

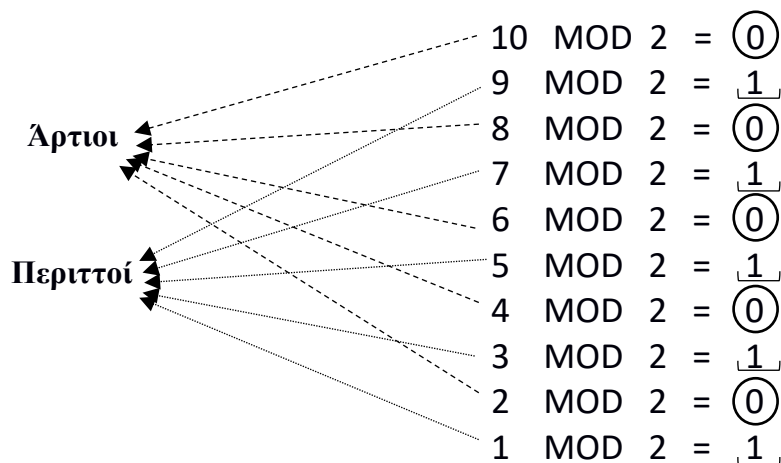
- α. Οι αριθμοί μεταξύ των οποίων γίνεται η διαίρεση (τα x και y), πρέπει να είναι ακέραιοι αριθμοί. Δε μπορούμε δηλαδή να κάνουμε $12.5 \text{ DIV } 2$ ή $7 \text{ MOD } 2.3$
- β. Το αποτέλεσμα είτε της πράξης DIV είτε της πράξης MOD θα είναι ακέραιος αριθμός. Οπότε αν σε κάποιο ερώτημα δείτε κάτι τέτοιο $11 \text{ DIV } 2 = 5.5$ αυτόματα με το που βλέπουμε ότι το αποτέλεσμα δεν είναι ακέραιος αριθμός καταλαβαίνουμε ότι είναι λάθος το αποτέλεσμα της πράξης.
- γ. Στην πράξη MOD εάν ο διαιρετέος είναι μικρότερος από τον διαιρέτη, τότε το αποτέλεσμα είναι ο διαιρετέος. Δηλαδή $4 \text{ MOD } 9 = 4$, $21 \text{ MOD } 33 = 21$.

ΠΟΥ ΧΡΗΣΙΜΟΠΟΙΟΥΝΤΑΙ;

Οι πιο συχνές περιπτώσεις στις οποίες χρησιμοποιούμε τα MOD και DIV είναι οι ακόλουθες:

1. Άρτιος – Περιττός

Χρησιμοποιώντας τον τελεστή MOD, μπορούμε να ελέγξουμε αν ένας αριθμός είναι άρτιος ή περιττός.



Όπως βλέπουμε στο παράδειγμα το ακέραιο υπόλοιπο της διαίρεσης ενός άρτιου αριθμού με το 2 είναι 0, ενώ το ακέραιο υπόλοιπο της διαίρεσης ενός περιττού αριθμού είναι 1. Συνεπώς αν το ακέραιο υπόλοιπο της διαίρεσης ενός αριθμού με το 2 είναι 0 τότε αυτός ο αριθμός είναι άρτιος. Αν το ακέραιο υπόλοιπο της διαίρεσης ενός περιττού αριθμού είναι 1 τότε αυτός ο αριθμός είναι περιττός. Άρα με την πράξη $x \text{ MOD } 2 = 0$ ελέγχουμε αν ο αριθμός x είναι άρτιος. Με την πράξη $x \text{ MOD } 2 = 1$ ελέγχουμε αν ο αριθμός x είναι περιττός.

2. Πολλαπλάσια ενός αριθμού

$$x \text{ MOD } \langle \text{αριθμός} \rangle = 0$$

Γνωρίζουμε ότι οι αριθμοί που είναι πολλαπλάσιοι ενός άλλου αριθμού δεν αφήνουν υπόλοιπο (π.χ. $70 \text{ MOD } 7 = 0$). Οπότε ο τελεστής MOD μπορεί να χρησιμοποιηθεί για να ελέγχουμε αν ένας αριθμός είναι πολλαπλάσιος ενός άλλου. Άρα αν ένας αριθμός x είναι πολλαπλάσιος ενός αριθμού ($\langle \text{αριθμός} \rangle$) τότε το ακέραιο υπόλοιπο της διαίρεσης τους είναι 0. Πχ.

$$110 \text{ MOD } 11 = 0$$

$$77 \text{ MOD } 7 = 0$$

$$56 \text{ MOD } 7 = 0$$

3. Ψηφία ενός αριθμού

Αν θέλουμε να πάρουμε συγκεκριμένα ψηφία από έναν αριθμό, τότε η διαίρεση αυτού του αριθμού με το 10 και τα πολλαπλάσια του (100, 1000, κλπ.) μπορούν να μας δώσουν τα ψηφία που θέλουμε. Πχ.

$$2357 \text{ DIV } 10 = 235$$

$$2357 \text{ DIV } 100 = 23$$

$$2357 \text{ DIV } 1000 = 2$$

Η DIV χρησιμοποιείται για να ξεχωρίσουμε τα μπροστά ψηφία ενός αριθμού. Όσα είναι τα μηδενικά του διαιρέτη, τόσα είναι τα ψηφία του διαιρετέου που απορρίπτονται.

$$2357 \text{ MOD } 10 = 7$$

$$2357 \text{ MOD } 100 = 57$$

$$2357 \text{ MOD } 1000 = 357$$

Η MOD χρησιμοποιείται για να πάρουμε τα πίσω ψηφία ενός αριθμού. Όσα είναι τα μηδενικά του διαιρέτη τόσα είναι τα ψηφία του διαιρετέου, από το τέλος προς την αρχή, που επιλέγονται. Αν θα θέλαμε να τα αναπαραστήσουμε γραφικά τις 2 αυτές περιπτώσεις των DIV και MOD θα ήταν ως εξής:

$$\begin{array}{c} \text{2 3 } \overline{57} \text{ DIV } 100 = 23 \\ \text{όσα είναι τα 0 τόσες θέσεις σβήνονται} \end{array}$$

$$2 \ 3 \ 5 \ 7 \ \text{MOD} \ 100 = 57$$

όσα είναι τα 0 τόσες θέσεις επιλέγονται

ΥΠΟΛΟΓΙΣΜΟΣ ΤΙΜΗΣ ΑΠΟ ΕΝΑ ΠΟΣΟΣΤΟ

Σε πολλές περιπτώσεις αοκήσεων, ζητείται ο υπολογισμός μιας τιμής από ένα ποσοστό. Π.χ. α) ο υπολογισμός του επιτοκίου ενός δανείου που είναι 3%, β) ο υπολογισμός του ΦΠΑ της τιμής ενός προϊόντος που είναι 23%, γ) ο υπολογισμός των κρατήσεων ενός μισθού που είναι 10% κλπ. Για να υπολογίσουμε τη τιμή ενός ποσοστού πρέπει να πολλαπλασιάσουμε την τιμή αυτή με το ποσοστό που αναζητάμε. Δηλαδή για να υπολογίσουμε το ΦΠΑ πρέπει να πολλαπλασιάσουμε την τιμή του προϊόντος με το ποσοστό 23%, για να υπολογίσουμε το επιτόκιο ενός δανείου πρέπει να πολλαπλασιάσουμε τη τιμή του δανείου με το ποσοστό του επιτοκίου. Δηλαδή ο γενικός τύπος υπολογισμού του ποσοστού μιας τιμής είναι:

$$\text{Ποσοστό Τιμής} = \text{Τιμή} * \text{Ποσοστό}$$

Αν δηλαδή μας ζητηθεί να υπολογίσουμε τις κρατήσεις ενός μισθού όταν γνωρίζουμε ότι οι κρατήσεις είναι 10%, τότε αυτό που κάνουμε είναι:

$$\text{Κρατήσεις} = \text{Μισθός} * 10/100$$

Υπολογισμός τελικής τιμής

Με το τρόπο που αναφέραμε πριν, υπολογίζουμε τη τιμή ενός ποσοστού. Αρκετές φορές όμως μας ζητείται ο υπολογισμός της **τελικής τιμής** συμπεριλαμβανομένης και της τιμής του ποσοστού που υπολογίστηκε πριν. Τέτοιες περιπτώσεις είναι α) ο υπολογισμός της τελικής τιμής του προϊόντος συμπεριλαμβανομένου και του ΦΠΑ, β) ο υπολογισμός του μισθούς συμπεριλαμβανομένων και των κρατήσεων, γ) ο υπολογισμός του δανείου συμπεριλαμβανομένου και του επιτοκίου κλπ. Σε αυτές τις περιπτώσεις αυτό που κάνουμε είναι να προσθέτουμε ή να αφαιρούμε (ανάλογα με το τί ζητάει η άσκηση) στην τιμή, το ποσοστό που υπολογίσαμε πριν. Δηλαδή στην περίπτωση υπολογισμού της τελικής τιμής μαζί με το ΦΠΑ

$$\text{Τελική_Τιμή} = \text{Αρχική_Τιμή} + \text{Ποσοστό_Τιμής} \quad \text{ή αλλιώς}$$

$$\text{Τελική_Τιμή} = \text{Αρχική_Τιμή} + (\text{Αρχική_Τιμή} * \text{Ποσοστό})$$

Στη περίπτωση υπολογισμού της τελικής τιμής του μισθού μαζί με τις κρατήσεις, θα είναι:

$$\text{Τελικός_Μισθός} = \text{Μισθός} - (\text{Μισθός} * \text{Ποσοστό})$$

Η αφαίρεση οφείλεται στο γεγονός ότι μέσα στη παρένθεση υπολογίζουμε τις κρατήσεις.

Υπολογισμός αρχικής τιμής

Αρκετές φορές χρειάζεται να υπολογίσουμε την αρχική τιμή όταν γνωρίζουμε την τελική τιμή και το ποσοστό. Για παράδειγμα μπορεί να μας ζητηθεί η αρχική τιμή ενός προϊόντος όταν η τελική τιμή του είναι 12€ και προέκυψε μετά από έκπτωση 20%. Σε αυτές πάλι εφαρμόζουμε τον τύπο που είδαμε παραπάνω ($\text{Τελική_Τιμή} = \text{Αρχική_Τιμή} + (\text{Αρχική_Τιμή} * \text{Ποσοστό})$) μόνο που αυτή τη φορά θα κάνουμε αφαίρεση διότι η τελική τιμή προκύπτει έπειτα από έκπτωση. Οπότε:

$$\text{Τελική_Τιμή} = \text{Αρχική_Τιμή} - (\text{Αρχική_Τιμή} * \text{Ποσοστό})$$

$$12 = \text{Αρχική_Τιμή} - (\text{Αρχική_Τιμή} * 20/100)$$

$$12 = \frac{80 * \text{Αρχική_Τιμή}}{100}$$

$$\text{Αρχική_Τιμή} = \frac{12}{0,8} = 15$$

ΥΠΟΛΟΓΙΣΜΟΣ ΠΟΣΟΣΤΟΥ ΑΠΟ ΜΙΑ ΤΙΜΗ

Εκτός από τον υπολογισμό μια τιμής από ένα ποσοστό, μπορεί να χρειαστεί να υπολογίσουμε και το αντίθετο, δηλαδή τον υπολογισμό ενός ποσοστού από μια τιμή. Τέτοιες περιπτώσεις είναι α) ο υπολογισμός του ποσοστού των κομμάτων που έλαβαν στις βουλευτικές εκλογές, β) το ποσοστό των καπνιστών στην Ελλάδα, γ) το ποσοστό ευστοχίας μιας ομάδας σε ένα αγώνα μπάσκετ κ.α.

Στη περίπτωση υπολογισμού ενός ποσοστού, χρειάζεται να γνωρίζουμε δύο τιμές α) τη συνολική τιμή και β) την τιμή της οποίας το ποσοστό ψάχνουμε. Δηλαδή, αν π.χ. έχουμε ένα βαρέλι που χωράει 150 λίτρα και αδειάσουμε μέσα στο βαρέλι 90 λίτρα, τότε η συνολική τιμή είναι τα 150 λίτρα (αφού είναι η χωρητικότητα όλου του βαρελιού) και η τιμή της οποίας το ποσοστό ψάχνουμε είναι τα 90 λίτρα. Οπότε ο τύπος για τον υπολογισμό του ποσοστού είναι:

$$\text{Ποσοστό} = (\text{τιμή της οποίας το ποσοστό ψάχνουμε} / \text{συνολική τιμή}) * 100$$

Συνεπώς στο παράδειγμα με το βαρέλι το ποσοστό είναι:

$$\text{Ποσοστό} = (90 / 150) * 100 = 60\%$$

Οπότε αν σε ένα βαρέλι που χωράει 150 λίτρα ρίξουμε 90 λίτρα νερό τότε θα έχει καλυφτεί το 60% του βαρελιού με νερό.

ΕΝΤΟΛΗ ΑΝ

1^η) Σύνταξη εντολής

**Αν <συνθήκη> τότε
<εντολές>
Τέλος_αν**

2^η) Σύνταξη εντολής

**Αν <συνθήκη> τότε
<εντολές_1>
αλλιώς
<εντολές_2>
Τέλος_αν**

3^η) Σύνταξη εντολής

**Αν <συνθήκη_1> τότε
<εντολές_1>
Αλλιώς_αν <συνθήκη_2> τότε
<εντολές_2>
.....
αλλιώς
<εντολές_n>
Τέλος_αν**

Η πιο σημαντική και συχνή, εντολή στον προγραμματισμό, είναι η εντολή ΑΝ. Η σύνταξη της εντολής αλλάζει ανάλογα με το πόσες συνθήκες σκοπεύουμε να χρησιμοποιήσουμε μέσα στην εντολή. Κάθε εντολή ΑΝ αποτελείται από 3 μέρη: α) τη <συνθήκη>, β) τις <εντολές>, γ) την εντολή Τέλος_αν που δηλώνει το τέλος της εντολής ΑΝ.

Η εντολή AN όταν έχουμε μία <συνθήκη> δουλεύει με τον εξής τρόπο κατά σειρά:

1. Το πρώτο πράγμα που κάνει είναι να ελέγξει αν ισχύει η <συνθήκη> (αν είναι δηλαδή είναι ΑΛΗΘΗΣ ή ΨΕΥΔΗΣ) (βλέπε 1^η σύνταξη εντολής)
2. Αν η <συνθήκη> είναι ΑΛΗΘΗΣ, τότε εκτελούνται όλες οι <εντολές> (βλέπε 1^η σύνταξη εντολής)

Στο σημείο <εντολές> μπορούμε να έχουμε περισσότερες από μια εντολές.

Στη περίπτωση που θέλουμε να εκτελέσουμε κάποιες εντολές όταν η <συνθήκη> είναι ΑΛΗΘΗΣ και κάποιες άλλες όταν η <συνθήκη> είναι ΨΕΥΔΗΣ, τότε χρησιμοποιούμε τη 2^η σύνταξη εντολής, η οποία δουλεύει ως εξής:

1. Αρχικά ελέγχεται αν η <συνθήκη> είναι ΑΛΗΘΗΣ.
2. Αν είναι ΑΛΗΘΗΣ, εκτελούνται όλες οι εντολές <εντολές_1>
3. Αν η συνθήκη είναι ΨΕΥΔΗΣ, **δεν εκτελούνται** οι <εντολές_1> και εκτελούνται οι <εντολές_2>

Τέλος στη περίπτωση που έχουμε πολλές συνθήκες η σύνταξη της εντολής AN γίνεται όπως στη 3^η περίπτωση σύνταξης, και δουλεύει ως εξής:

1. Ελέγχει αν η <συνθήκη_1> είναι ΑΛΗΘΗΣ. Στη περίπτωση που είναι ΑΛΗΘΗΣ, εκτελούνται οι <εντολές_1> και τελειώνει η εκτέλεση της AN. Αν η <συνθήκη_1> είναι ΨΕΥΔΗΣ, **δεν εκτελούνται** οι <εντολές_1>, αλλά ελέγχεται η <συνθήκη_2>
2. Αν η <συνθήκη_2> είναι ΑΛΗΘΗΣ, εκτελούνται οι <εντολές_2> και τελειώνει η εκτέλεση της AN. Στη περίπτωση που η <συνθήκη_2> είναι ΨΕΥΔΗΣ, δεν εκτελούνται οι <εντολές_2>, αλλά ελέγχεται η <συνθήκη_3>, κλπ.
3. Αν στο τέλος δεν ικανοποιείται καμία από τις συνθήκες, τότε και μόνο τότε εκτελούνται οι <εντολές_n> και τερματίζεται η AN.

Παρακάτω φαίνονται κάποια παραδείγματα της εντολής AN:

Αλγόριθμος παράδειγμα_1
Εμφάνισε 'Δώσε ένα αριθμό:'
Διάβασε αριθμός
Αν αριθμός >= 0 τότε
 Εμφάνισε 'Ο αριθμός είναι
 θετικός'
Τέλος_αν
Τέλος παράδειγμα_1

Αλγόριθμος παράδειγμα_2
Εμφάνισε 'Δώσε ένα αριθμό:'
Διάβασε αριθμός
Αν αριθμός >= 0 τότε
 Εμφάνισε 'Ο αριθμός είναι
 θετικός'
Αλλιώς
 Εμφάνισε 'Ο αριθμός είναι
 αρνητικός'
Τέλος_αν
Τέλος παράδειγμα_2

Αλγόριθμος παράδειγμα_3
Εμφάνισε 'Δώσε ένα αριθμό:'
Διάβασε αριθμός
Αν αριθμός > 0 τότε
 Εμφάνισε 'Ο αριθμός είναι
 θετικός'
Αλλιώς_αν αριθμός < 0 τότε
 Εμφάνισε 'Ο αριθμός είναι
 αρνητικός'
Αλλιώς
 Εμφάνισε 'Ο αριθμός είναι 0'
Τέλος_αν
Τέλος παράδειγμα_3

Προσοχή!!! Αν έχουμε πολλές συνθήκες, δεν είναι δεσμευτική η 3^η σύνταξη της εντολής AN. Μπορεί να γραφεί και με τη χρήση της 1^{ης} σύνταξης της εντολής AN, για κάθε συνθήκη. Για παράδειγμα ο αλγόριθμος παράδειγμα_3 θα μπορούσε να γραφεί και με τον εξής τρόπο:

Αλγόριθμος παράδειγμα_3
Εμφάνισε 'Δώσε ένα αριθμό:'
Διάβασε αριθμός
Αν αριθμός > 0 τότε
 Εμφάνισε 'Ο αριθμός είναι θετικός'
Τέλος_αν
Αν αριθμός < 0 τότε
 Εμφάνισε 'Ο αριθμός είναι αρνητικός'

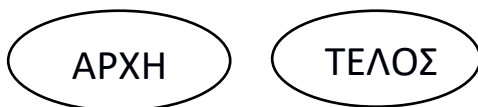
Τέλος_αν
Αν αριθμός = 0 τότε
 Εμφάνισε 'Ο αριθμός είναι 0'
Τέλος_αν
Τέλος παράδειγμα_3

ΔΙΑΓΡΑΜΜΑΤΑ ΡΟΗΣ

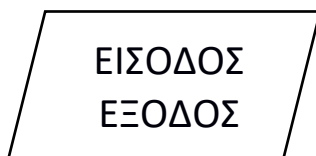
Το διάγραμμα ροής είναι η μετατροπή-αναπαράσταση ενός προγράμματος σε μορφή σχημάτων. Το διάγραμμα ροής βοηθάει τον προγραμματιστή στην υλοποίηση του προγράμματος και είναι η πιο γενική μορφή ενός προγράμματος, ακόμα πιο γενική και από τον αλγόριθμο.

Ένα διάγραμμα ροής αποτελείται από ένα σύνολο γεωμετρικών σχημάτων, όπου το καθένα δηλώνει μία συγκεκριμένη ενέργεια ή λειτουργία. Τα γεωμετρικά σχήματα ενώνονται μεταξύ τους με βέλη, που δηλώνουν τη σειρά εκτέλεσης των ενεργειών αυτών. Τα κυριότερα χρησιμοποιούμενα γεωμετρικά σχήματα είναι τα εξής:

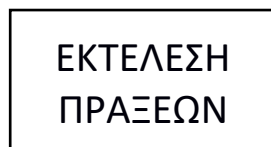
- έλλειψη, που δηλώνει την αρχή και το τέλος του κάθε αλγορίθμου,
- ρόμβος, που δηλώνει μία ερώτηση με δύο ή περισσότερες εξόδους για απάντηση,
- ορθογώνιο, που δηλώνει την εκτέλεση μίας ή περισσότερων πράξεων, και
- πλάγιο παραλληλόγραμμο, που δηλώνει είσοδο ή έξοδο στοιχείων.



Έλλειψη που δηλώνει την αρχή ή το τέλος του κάθε αλγορίθμου



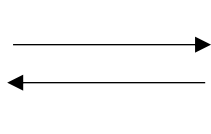
Πλάγιο παραλληλόγραμμο που δηλώνει είσοδο ή έξοδο στοιχείων (εισαγωγή δεδομένων ή έξοδο αποτελεσμάτων)



Ορθογώνιο που δηλώνει την εκτέλεση μιας ή περισσότερων πράξεων



Ρόμβος που δηλώνει μια ερώτηση με δυο εξόδους για απάντηση



Βέλη που δηλώνουν τη σειρά εκτέλεσης των ενεργειών

Τα πλάγια παραλληλόγραμμα συνήθως χρησιμοποιούνται για τις εντολές **ΕΜΦΑΝΙΣΕ** και **ΔΙΑΒΑΣΕ** αφού αυτές οι εντολές στη πλειοψηφία των προγραμμάτων και των αλγορίθμου είναι τα κομμάτια εισόδου και εξόδου ενός αλγορίθμου.

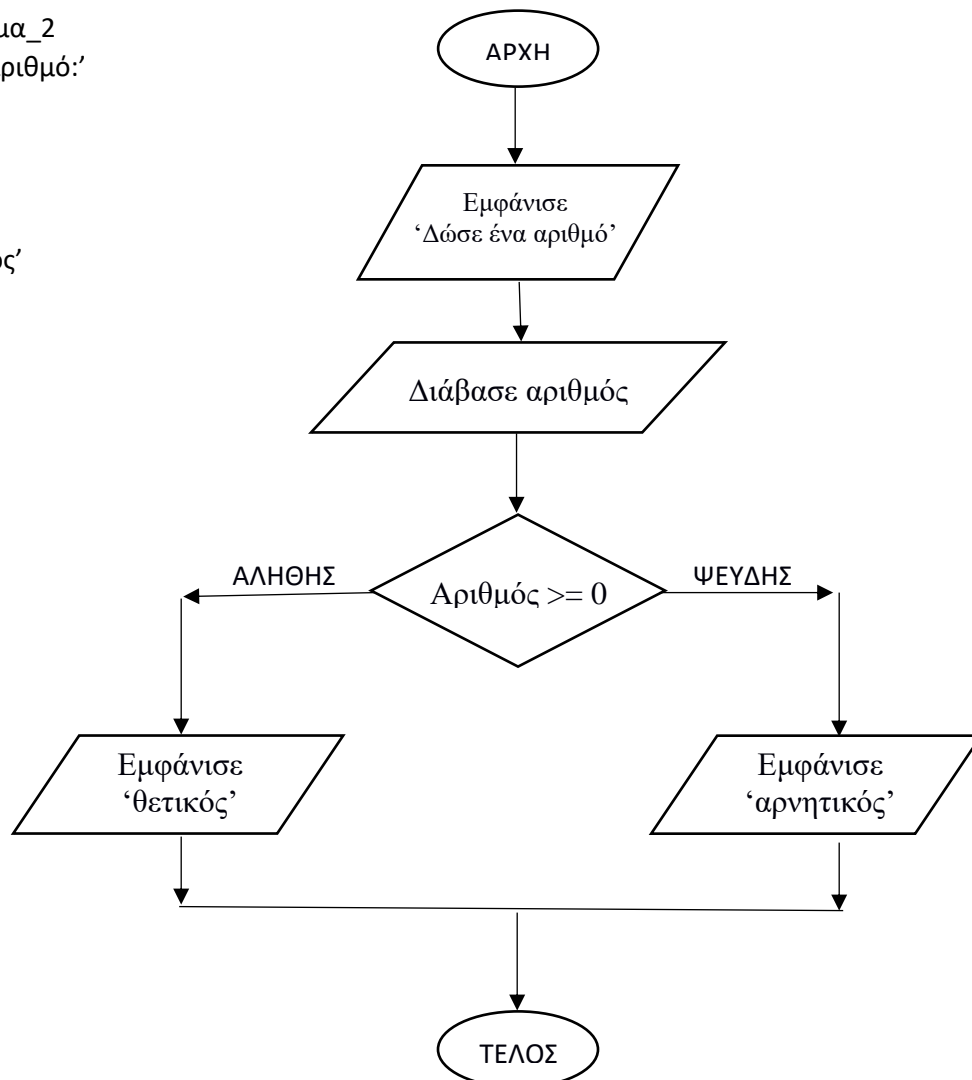
Τα ορθογώνια συνήθως χρησιμοποιούνται για τα κομμάτια του αλγορίθμου που έχουν την **εντολή εκχώρησης** (\leftarrow).

Οι ρόμβοι χρησιμοποιούνται για τα σημεία του αλγορίθμου στα οποία υπάρχουν συνθήκες, όπως η εντολή **ΑΝ** και οι **εντολές επανάληψης** που θα δούμε αργότερα.

Τέλος τα βέλη μπαίνουν ανάμεσα στα σχήματα, προκειμένου να δηλώσουν ποιο είναι το επόμενο σχήμα στο οποίο μεταβαίνει η ροή του αλγορίθμου. Ποια είναι δηλαδή η επόμενη εντολή που εκτελείται.

Παρακάτω φαίνεται ένα παράδειγμα διαγράμματος ροής με τον αλγόριθμό του.

Αλγόριθμος παράδειγμα_2
 Εμφάνισε 'Δώσε ένα αριθμό:'
 Διάβασε αριθμός
 Αν αριθμός ≥ 0 τότε
 Εμφάνισε 'θετικός'
 Αλλιώς
 Εμφάνισε 'αρνητικός'
 Τέλος_αν
 Τέλος παράδειγμα_2



ΚΡΙΤΗΡΙΑ ΑΛΓΟΡΙΘΜΩΝ

- Είσοδος: Σε κάθε αλγόριθμος θα πρέπει να δίνουμε τουλάχιστον ένα δεδομένο ως είσοδο ώστε μετά στην συνέχεια να το επεξεργάζεται και να μας δίνει τα αποτελέσματα. Όταν δηλαδή σε ένα αλγόριθμο υπάρχει η εντολή ΔΙΑΒΑΣΕ αμέσως καταλαβαίνουμε ότι ο αλγόριθμος πληροί το κριτήριο της εισόδου.
Εξαίρεση αποτελεί όταν ο αλγόριθμος φτιάχνει δικά του δεδομένα. Όταν δηλαδή ο αλγόριθμος δημιουργεί μια μεταβλητή με κάποιο αποτέλεσμα ($x < 5$), επίσης πληροί το κριτήριο της εισόδου.
- Εξοδος: Κάθε αλγόριθμος θα πρέπει οπωσδήποτε να μας δίνει τουλάχιστον ένα αποτέλεσμα σαν έξοδο. Για παράδειγμα δεν έχει κανένα νόημα να υπολογίσει το εμβαδόν του τριγώνου και να μην το εμφανίσουμε.

Παράδειγμα 1

Διάβασε x
Αν $x \geq 0$ τότε
 Εμφάνισε 'Θετικός'
Αλλιώς
 Εμφάνισε 'Αρνητικός'
Τέλος_αν

Παράδειγμα 2

$x \leftarrow -3$
Αν $x \geq 0$ τότε
 Εμφάνισε 'Θετικός'
Τέλος_αν

Στο παράδειγμα 1 πληρείται και το κριτήριο της εισόδου και το κριτήριο της εξόδου, αφού ο αλγόριθμος ξεκινάει με την εντολή ΔΙΑΒΑΣΕ και σαν έξοδο μας δίνει το μήνυμα 'Θετικός' ή 'Αρνητικός' ανάλογα με το τι είναι ο αριθμός x

Στο παράδειγμα 2 πληρείται το κριτήριο της εισόδου, αφού ξεκινάει με την εκχώρηση στη μεταβλητή x της τιμής -3 , αλλά δεν πληρείται το κριτήριο της εξόδου. Ο αλγόριθμος δεν θα μας εμφανίσει τίποτα στην οθόνη ως αποτέλεσμα, αφού δε θα μπορέσει να εκτελεστεί ποτέ η εντολή ΕΜΦΑΝΙΣΕ.

- Καθοριστικότητα: Κάθε εντολή που του λέμε να κάνει, θα πρέπει να είναι πλήρως καθορισμένη παίρνοντας υπ' όψη μας όλες τις διαφορετικές πιθανές περιπτώσεις που μπορεί να προκύψουν ανάλογα με τα δεδομένα.

Παράδειγμα 3

$i \leftarrow 9$
Διάβασε x
 $y \leftarrow i/x$
Εμφάνισε y

Παράδειγμα 4

$i \leftarrow 9$
Διάβασε x
Αν $x \neq 0$ τότε
 $y \leftarrow i/x$
 Εμφάνισε y
Αλλιώς
 Εμφάνισε 'Δε μπορεί να γίνει η διαίρεση'
Τέλος_αν

Στο παράδειγμα 3 πληρείται το κριτήριο της εισόδου αφού στην αρχή δημιουργείται μια μεταβλητή i που παίρνει τη τιμή 9 και με μια ΔΙΑΒΑΣΕ δίνουμε εμείς τιμή στην μεταβλητή x . Πληρείται το κριτήριο της εξόδου, αφού ο αλγόριθμος τελειώνει με την εντολή ΕΜΦΑΝΙΣΕ και μας δίνει αποτελέσματα στην οθόνη. Δεν πληρείται όμως το κριτήριο της καθοριστικότητας, διότι η εντολή $y \leftarrow i/x$ δεν είναι πλήρως ξεκάθαρη. Δεν λαμβάνεται υπόψη η περίπτωση όπου το x μπορεί να είναι 0, οπότε δε θα μπορεί να γίνει η διαίρεση. Συνεπώς δεν είναι πλήρως ξεκάθαρη η πράξη της διαίρεσης στη συγκεκριμένη περίπτωση.

Αν θα θέλαμε να κάνουμε τον αλγόριθμο του παραδείγματος 3 να πληροί και το κριτήριο της καθοριστικότητας, έπρεπε να μετατραπεί όπως στο παράδειγμα 4. Όπου με μια ΑΝ εξασφαλίζουμε ότι όταν μπει ο αλγόριθμος στο κομμάτι της διαίρεσης, ο παρονομαστής (δηλαδή το x) θα είναι σίγουρα διάφορος του 0.

- Περατότητα: Θα πρέπει κάποια στιγμή ο αλγόριθμος να τελειώνει. Η δομή του δηλαδή να είναι τέτοια έτσι ώστε σίγουρα να τελειώνει κάποια στιγμή. Υπάρχει περίπτωση η δομή του αλγορίθμου να είναι τέτοια που να μην τελειώνει ποτέ ο αλγόριθμος (θα το δούμε σε άλλα μαθήματα), κάτι το οποίο δεν είναι αποδεκτό στο προγραμματισμό.
- Αποτελεσματικότητα: Κάθε εντολή-βήμα του αλγορίθμου πρέπει να είναι απλή, εκτελέσιμη και υπαρκτή.

Παράδειγμα 5

Διάβσε t
 $y \leftarrow T_R(t)$
Εμφάνισε y

Στο παράδειγμα 5 δεν πληρείται το κριτήριο της αποτελεσματικότητας, διότι η εντολή ΔΙΑΒΑΣΕ δεν είναι σωστά γραμμένη (Διάβσε), οπότε δεν αναγνωρίζεται σαν εντολή. Επίσης δεν υπάρχει η συνάρτηση T_R . Η συνάρτηση που έχουμε μάθει για τον υπολογισμό της τετραγωνικής ρίζας είναι η T_P .

ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΣΕ ΓΛΩΣΣΑ

ΔΟΜΗ ΕΝΟΣ ΠΡΟΓΡΑΜΜΑΤΟΣ

```
ΠΡΟΓΡΑΜΜΑ <Όνομα_Προγράμματος>  
ΣΤΑΘΕΡΕΣ  
    <Σταθερές Τιμές του προγράμματος>  
ΜΕΤΑΒΛΗΤΕΣ  
    <Μεταβλητές του προγράμματος>  
ΑΡΧΗ  
    <Πρόγραμμα>  
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Ένα πρόγραμμα αρχίζει με τη λέξη ΠΡΟΓΡΑΜΜΑ ακολουθούμενη από μια οποιαδήποτε λέξη που δηλώνει το όνομα του προγράμματος. Στη συνέχεια δηλώνονται οι ΣΤΑΘΕΡΕΣ τιμές του προγράμματος π.χ. $\pi=3.14$. Ακολουθεί η δήλωση των ΜΕΤΑΒΛΗΤΩΝ τιμών και στη συνέχεια αρχίζει η δημιουργία του κυρίου κορμού του προγράμματος. Παρακάτω φαίνεται η δομή ενός προγράμματος

Παράδειγμα1:

```
ΠΡΟΓΡΑΜΜΑ παραδειγμα  
ΣΤΑΘΕΡΕΣ  
     $y=0.01$   
ΜΕΤΑΒΛΗΤΕΣ  
    ΠΡΑΓΜΑΤΙΚΕΣ:  $x, z$   
ΑΡΧΗ  
    ΓΡΑΨΕ 'Δώσε ένα νούμερο:'  
    ΔΙΑΒΑΣΕ  $x$   
     $z \leftarrow x * y$   
    ΓΡΑΨΕ 'Η διαίρεση του ' $x$ ,' με το 100 μας δίνει ' $z$   
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Η δομή ενός προγράμματος είναι συγκεκριμένη και αμετάβλητη. Δεν γίνεται π.χ. να δηλωθούν πρώτα οι ΜΕΤΑΒΛΗΤΕΣ τιμές και στη συνέχεια οι ΣΤΑΘΕΡΕΣ. Στη περίπτωση που δεν έχουμε ΣΤΑΘΕΡΕΣ παραλείπεται το κομμάτι των ΣΤΑΘΕΡΩΝ τιμών και δεν το γράφουμε καθόλου. Οπότε η δομή γίνεται:

```
ΠΡΟΓΡΑΜΜΑ <Όνομα_Προγράμματος>  
ΜΕΤΑΒΛΗΤΕΣ  
    <Μεταβλητές του προγράμματος>  
ΑΡΧΗ  
    <Πρόγραμμα>  
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

ΣΤΑΘΕΡΕΣ ΤΙΜΕΣ

Στο κομμάτι των σταθερών τιμών δηλώνονται οι μεταβλητές των οποίων οι τιμές παραμένουν σταθερές σε όλη τη διάρκεια του προγράμματος και δε γίνεται να αλλάξουν κατά τη διάρκεια εκτέλεσης του κώδικα, μπορούν όμως να χρησιμοποιηθούν στην εκτέλεση πράξεων.

Πχ $\pi=3.14$, $g=6.68 \cdot 10^{-11}$ (το σύμβολο $^$ σημαίνει 'δύναμη')

ΕΙΔΗ ΜΕΤΑΒΛΗΤΩΝ

Υπάρχουν τριών ειδών μεταβλητές, οι ΑΚΕΡΑΙΕΣ, οι ΠΡΑΓΜΑΤΙΚΕΣ, οι ΛΟΓΙΚΕΣ και οι ΧΑΡΑΚΤΗΡΕΣ.

Οι ΑΚΕΡΑΙΕΣ μεταβλητές είναι μεταβλητές που μπορούν να πάρουν **μόνο** ακέραιες τιμές, πχ $x = 3$, $y = 46$.

Οι ΠΡΑΓΜΑΤΙΚΕΣ είναι μεταβλητές που μπορούν να πάρουν **μόνο** πραγματικές τιμές, πχ $z=5.34$, $t=14.0000$.

Οι μεταβλητές ΧΑΡΑΚΤΗΡΕΣ είναι μεταβλητές που περιέχουν κάποιο αλφαριθμητικό (δηλαδή χαρακτήρες με αριθμούς μαζί), πχ $a='Σήμερα έχουμε 25 βαθμούς Κελσίου'$, $c='Το Επώνυμο μου είναι Δημητρίου'$. Το περιεχόμενο των μεταβλητών ΧΑΡΑΚΤΗΡΕΣ, πρέπει να περιλαμβάνεται ανάμεσα σε $' '$. Τέλος οι ΛΟΓΙΚΕΣ μεταβλητές είναι μεταβλητές που μπορούν να πάρουν την τιμή ΑΛΗΘΗΣ ή τη τιμή ΨΕΥΔΗΣ, πχ $g=ΑΛΗΘΗΣ$, $r=ΨΕΥΔΗΣ$. Οι ΛΟΓΙΚΕΣ μεταβλητές χρησιμοποιούνται κυρίως σε συνθήκες και επαναληπτικές εντολές.

Παράδειγμα2:

ΠΡΟΓΡΑΜΜΑ παραδειγμα

ΜΕΤΑΒΛΗΤΕΣ

ΑΚΕΡΑΙΕΣ: i

ΠΡΑΓΜΑΤΙΚΕΣ: x

ΧΑΡΑΚΤΗΡΕΣ: c

ΛΟΓΙΚΕΣ: l

ΑΡΧΗ

ΓΡΑΨΕ 'Δώσε ένα νούμερο(ακέραιο):'

ΔΙΑΒΑΣΕ i

ΓΡΑΨΕ 'Δώσε ένα νούμερο(πραγματικό αριθμό):'

ΔΙΑΒΑΣΕ x

ΓΡΑΨΕ 'Δώσε κάποιο κείμενο:'

ΔΙΑΒΑΣΕ c

ΓΡΑΨΕ 'ΓΡΑΨΕ ΑΛΗΘΗΣ Ή ΨΕΥΔΗΣ:'

ΔΙΑΒΑΣΕ l

ΓΡΑΨΕ 'Ακέραιος = ',i

ΓΡΑΨΕ 'Πραγματικός = ',x

ΓΡΑΨΕ 'Κείμενο = ',c

ΓΡΑΨΕ 'Λογική μεταβλητή = ',l

ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ

ΣΦΑΛΜΑΤΑ

Στο προγραμματισμό υπάρχουν δύο ειδών σφάλματα. Τα συντακτικά και τα λογικά.

- A. **Συντακτικά Σφάλματα**: Είναι τα σφάλματα που εμφανίζονται όταν έχουμε κάποιο λάθος στη σύνταξη του προγράμματος ή του αλγορίθμου. Αν για παράδειγμα αντί για την εντολή *Διάβασε x* γράψουμε *Διαβσε x* όταν θα εκτελέσουμε το πρόγραμμα μας, μόλις φτάσει να εκτελέσει αυτή την εντολή θα μας εμφανίσει ένα συντακτικό σφάλμα και θα σταματήσει την εκτέλεση του προγράμματος χωρίς αποτέλεσμα.

Στο παρακάτω πρόγραμμα θα σταματήσει η εκτέλεσή του και θα εμφανίσει συντακτικό σφάλμα στην 5^η γραμμή, διότι στη μεταβλητή *x* αποθηκεύουμε ένα πραγματικό αριθμό, ενώ η μεταβλητή έχει δηλωθεί ακέραια.

1. ΠΡΟΓΡΑΜΜΑ παραδ3
2. ΜΕΤΑΒΛΗΤΕΣ
3. Ακέραιες: *x*
4. ΑΡΧΗ
5. $x \leftarrow 3.5$
6. Γράψε *x*
7. ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ

- B. **Λογικά Σφάλματα**: Είναι το πιο δύσκολο είδος σφαλμάτων όπου ακόμα και έμπειροι προγραμματιστές δε μπορούν να τα εντοπίσουν. Τα λογικά σφάλματα δεν εμφανίζουν κανένα μήνυμα-σφάλμα όπως τα συντακτικά, αλλά αφήνουν τον αλγόριθμο να εκτελεστεί κανονικά μέχρι το τέλος. Καταλαβαίνουμε ότι ο αλγόριθμος μας έχει λογικό σφάλμα, όταν τα αποτελέσματα του δεν είναι τα αναμενόμενα.

Το παρακάτω πρόγραμμα θα εκτελεστεί κανονικά, το αποτέλεσμα όμως που θα μας δώσει δεν θα είναι σωστό, αφού δίνοντας έναν θετικό αριθμό θα μας εμφανίσει το μήνυμα 'Ο αριθμός που έδωσες, είναι αρνητικός' και το αντίστροφο. Συνεπώς πρέπει να εντοπίσουμε σε ποιο σημείο είναι το λογικό σφάλμα του αλγορίθμου, όπου στη συγκεκριμένη περίπτωση είναι στην 7^η γραμμή, στη συνθήκη της ΑΝ. Πρέπει να γίνει $y < 0$.

1. ΠΡΟΓΡΑΜΜΑ παραδ4
2. ΜΕΤΑΒΛΗΤΕΣ
3. ΠΡΑΓΜΑΤΙΚΕΣ: *y*
4. ΑΡΧΗ
5. Γράψε 'Δώσε αριθμό:'
6. Διάβασε *y*
7. Αν $y > 0$ τότε
8. Γράψε 'Ο αριθμός που έδωσες, είναι αρνητικός'
9. Αλλιώς
10. Γράψε 'Ο αριθμός που έδωσες, είναι θετικός ή μηδέν'
11. Τέλος_αν
12. Τέλος_προγράμματος

ΑΣΚΗΣΕΙΣ

1. Να γραφεί αλγόριθμος ο οποίος να διαβάζει έναν βαθμό και να τυπώνει ένα από τα ακόλουθα μηνύματα: 'Άριστα' αν ο βαθμός είναι μεγαλύτερος ή ίσος του 18.5. 'Προάγεται' αν ο βαθμός είναι μικρότερος του 18.5 και μεγαλύτερος ή ίσος του 9.5. 'Απορρίπτεται' αν ο βαθμός είναι μικρότερος του 9.5
2. Να γραφεί αλγόριθμος ο οποίος να διαβάζει δυο ακέραιους αριθμούς X,Y. Θα διαβάζει επίσης μια από τις 3 λέξεις (ΠΡΟΣΘΕΣΗ, ΑΦΑΙΡΕΣΗ, ΠΟΛ). Ανάλογα με τη λέξη που διαβάζει, θα κάνει και την αντίστοιχη πράξη και θα εμφανίζει το αποτέλεσμα.
3. Να κάνετε τις αντιστοιχίσεις μεταξύ των στοιχείων της στήλης Α με τα κατάλληλα στοιχεία της στήλης Β

Στήλη Α Όνομα Μεταβλητής	Στήλη Β Χαρακτηρισμός
1. ΦΠΑ	α. Αποδεκτή
2. 2AB	β. Μη αποδεκτή
3. ΒΑΘΜΟΣ	
4. 'ΜΙΣΘΟΣ'	
5. A32	
6. ΑΚΕΡΑΙΟΣ	

4. Να γραφεί πρόγραμμα που ζητάει από το χρήστη να δώσει ένα αριθμό και ελέγχει αν είναι πολλαπλάσιος του 5 και του 7, εμφανίζοντας αντίστοιχα μηνύματα.
5. Να γραφεί πρόγραμμα που υπολογίζει το μέσο όρο 2 αριθμών που δίνει ο χρήστης.
6. Να γραφεί αλγόριθμος στον οποίο ο χρήστης δίνει την ακτίνα ενός κύκλου και υπολογίζεται το εμβαδόν του και η περίμετρος του.
7. Να γραφεί πρόγραμμα που διαβάζει τους βαθμούς κελσίου που δίνει ο χρήστης και τους μετατρέπει σε βαθμούς fahrenheit. ($F=9C/5+32$)
8. Να γραφεί πρόγραμμα που υλοποιεί το πυθαγόρειο θεώρημα. Ο χρήστης δίνει τις διαστάσεις των δυο πλευρών ενός ορθογωνίου τριγώνου και το πρόγραμμα υπολογίζει την τρίτη πλευρά.
9. Σε μια σχολή ένας μαθητής προβιβάζεται όταν ο βαθμός του είναι μεγαλύτερος ή ίσος του 10 και οι απουσίες του είναι λιγότερες ή ίσες από 70. Αν ο βαθμός του είναι μικρότερος από 10 τότε πρέπει να δώσει εξετάσεις, αν οι απουσίες του είναι περισσότερες από 70 τότε πρέπει να φέρει δικαιολογητικό, ενώ αν ο βαθμός του είναι μικρότερος από 10 και οι απουσίες του περισσότερες από 70 τότε πρέπει να επαναλάβει την τάξη. Να γίνει αλγόριθμος που θα διαβάζει το βαθμό και τις απουσίες ενός μαθητή και θα εμφανίζει το κατάλληλο μήνυμα.
10. Να γραφεί αλγόριθμος που διαβάζει 2 αριθμούς και σε περίπτωση που ο πρώτος αριθμός είναι μικρότερος του δεύτερου, να υπολογισθεί και να εκτυπωθεί το άθροισμα τους, διαφορετικά να υπολογισθεί και να εκτυπωθεί το γινόμενό τους.

11. Να μετατρέψετε σε μαθηματικές σχέσεις τις παρακάτω εντολές εκχώρησης

A. $X \leftarrow \alpha + \beta / 2$

Γ. $X \leftarrow (\alpha + \beta) / (2 * \alpha)$

E. $K \leftarrow (\alpha + T_P(\alpha + \beta)) / (\alpha + 1)$

B. $X \leftarrow (\alpha + \beta) / 2 * \alpha$

Δ. $E \leftarrow 2 * T_P(x + y) / x + y$

Z. $E \leftarrow (\alpha + B)^2 / \alpha + 1$

12. Αν οι τιμές των μεταβλητών A, B και Γ είναι 2, 5 και 10, αντίστοιχα, να βρεθεί η τιμή των ακόλουθων λογικών προτάσεων: (μπλέ-Σ44-A7)

- $A > B$ ΚΑΙ $B > \Gamma$
- $(A + B) < \Gamma$ Η $(B + \Gamma) < A$
- $(B - 2) > A$ ΚΑΙ $B < A$ ΚΑΙ $\Gamma > (A - B)$
- ΟΧΙ $(A > 0$ Η $B < 0)$ Η $\Gamma < 20$
- $B + A > \Gamma / 2$ Η $A < B$ ΚΑΙ $A * B < \Gamma$
- $B + A > \Gamma / 2$ Η $(A < B$ ΚΑΙ $A * B < \Gamma)$

13. Να γραφεί αλγόριθμος ο οποίος να διαβάσει την αξία ενός Η/Υ χωρίς ΦΠΑ, την τιμή του ΦΠΑ και να υπολογίζει την αξία του με ΦΠΑ. Η τιμή του ΦΠΑ θα δίνεται με τη μορφή ποσοστού δηλαδή για ΦΠΑ 23% ο χρήστης θα δίνει 23.

14. Μια τράπεζα έχει ετήσιο επιτόκιο καταθέσεων 3,5%. Ένας καταθέτης καταθέτει σ' αυτή την τράπεζα κεφάλαιο ίσο με K. Να γραφεί αλγόριθμος που να διαβάσει το κεφάλαιο K και να υπολογίζει το κεφάλαιο του καταθέτη μετά από:

A) ένα χρόνο B) δύο χρόνια

15. Να γραφεί πρόγραμμα που ζητάει από το χρήστη να δώσει έναν αριθμό, γίνεται έλεγχος αν ο αριθμός είναι θετικός και υπολογίζει το υπόλοιπο του αριθμού με τη διαίρεση του με το 2. Αν το υπόλοιπο είναι 1 εμφανίζει το υπόλοιπο αλλιώς εμφανίζει κατάλληλο μήνυμα.

16. Τι θα εμφανίσει ο παρακάτω αλγόριθμος για τιμές εισόδου i) 1,2 ii) 5,5 Να τον μετατρέψετε σε διάγραμμα ροής.

Αλγόριθμος άσκηση

Διάβασε x, y

Εμφάνισε x, y

$x \leftarrow x + 3$

$y \leftarrow x + y$

Εμφάνισε x, y

Τέλος άσκηση

17. Να γίνει αλγόριθμος ο οποίος θα διαβάσει έναν αριθμό και θα εμφανίζει αν ο αριθμός είναι άρτιος ή περιττός

18. Δίνεται το παρακάτω τμήμα αλγορίθμου (μπλέ-Σ83-A4)

Αν ποσότητα ≤ 50 τότε

κόστος \leftarrow ποσότητα * 580

αλλιώς_αν ποσότητα > 50 ΚΑΙ ποσότητα ≤ 100 τότε

κόστος \leftarrow ποσότητα * 520

αλλιώς_αν ποσότητα > 100 ΚΑΙ ποσότητα ≤ 200 τότε

κόστος \leftarrow ποσότητα * 470

αλλιώς

κόστος \leftarrow ποσότητα $\ast 440$

Τέλος_αν

Στο παραπάνω τμήμα αλγορίθμου, για το οποίο θεωρούμε ότι η ποσότητα είναι θετικός αριθμός, περιλαμβάνονται περιττοί έλεγχοι. Να το ξαναγράψετε παραλείποντας τους περιττούς ελέγχους.

Εξετάσεις 2006

ΕΝΤΟΛΕΣ ΕΠΑΝΑΛΗΨΗΣ

Πολλές φορές σε ένα αλγόριθμο χρειάζεται να επαναλάβουμε την εκτέλεση ενός κομματιού κώδικα αρκετές φορές. Για παράδειγμα αν ένα πρόγραμμα ζητάει την εκτύπωση του μηνύματος 'Σήμερα είναι Τρίτη' 100 φορές, με βάση τα όσα γνωρίζουμε θα πρέπει να γράψουμε μέσα στο πρόγραμμα:

```
ΓΡΑΨΕ 'Σήμερα είναι Τρίτη'  
ΓΡΑΨΕ 'Σήμερα είναι Τρίτη'  
ΓΡΑΨΕ 'Σήμερα είναι Τρίτη'  
ΓΡΑΨΕ 'Σήμερα είναι Τρίτη'  
ΓΡΑΨΕ 'Σήμερα είναι Τρίτη'
```

....

Πράγμα το οποίο όπως καταλαβαίνουμε είναι χρονοβόρο και καθόλου πρακτικό. Σε τέτοιες περιπτώσεις όπου ένα σύνολο εντολών απαιτείται να εκτελεστεί περισσότερες από μια φορές, χρησιμοποιούμε τις εντολές επανάληψης.

Οι εντολές επανάληψης αυτό που κάνουν είναι να επαναλαμβάνουν την εκτέλεση ολόκληρων κομματιών κώδικα, τα οποία γράφονται μόνο μια φορά. Σε περιπτώσεις προγραμμάτων όπως αυτό παραπάνω όπου μια ή περισσότερες εντολές επαναλαμβάνονται πολλές φορές μπορούμε να χρησιμοποιήσουμε μια εντολή επανάληψης, προκειμένου να δηλώσουμε ποια-ες εντολή-ες θέλουμε να επαναληφθεί, και με κάποια συνθήκη το μέχρι πότε να επαναληφθεί.

Υπάρχουν 3 εντολές επανάληψης, η **ΟΣΟ**, η **ΜΕΧΡΙΣ_ΟΤΟΥ** και η **ΓΙΑ**.

ΕΝΤΟΛΗ ΓΙΑ:

```
ΓΙΑ <μεταβλητή> ΑΠΟ τιμή1 ΜΕΧΡΙ τιμή2 ΜΕ ΒΗΜΑ τιμή3  
    εντολή-1  
    εντολή-2  
    ...  
    εντολή-ν  
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

Η εντολή επανάληψης ΓΙΑ είναι ίσως η πιο απλή από τις 3 εντολές επανάληψης. Δίνεται στη <μεταβλητή> ως αρχική τιμή η τιμή <τιμή1> που είναι ένας αριθμός, εκτελούνται οι εντολές που βρίσκονται μέσα στην επανάληψη και στην επόμενη επανάληψη, αυξάνεται η τιμή της μεταβλητής κατά <τιμή3> και ξανά εκτελούνται οι εντολές που βρίσκονται μέσα στην επανάληψη. Η επανάληψη τερματίζεται όταν η μεταβλητή γίνει ίση με την <τιμή2> ή την ξεπεράσει. Πχ

```
ΓΙΑ k ΑΠΟ 1 ΜΕΧΡΙ 10 ΜΕ ΒΗΜΑ 1  
    ΓΡΑΨΕ 'Η τιμή του k είναι:', k  
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

ΕΠΑΝΑΛΗΨΕΙΣ	ΤΡΟΠΟΣ ΕΚΤΕΛΕΣΗΣ ΕΠΑΝΑΛΗΨΗΣ
1η επανάληψη: k=1	Το k γίνεται 1, εμφανίζεται στην οθόνη 'Η τιμή του k είναι: 1'

2η επανάληψη: k=2	Το k γίνεται 2, διότι αυξάνει με την τιμή του βήματος που δώσαμε, δηλ $k=1+τιμή_βήματος$. Αν η τιμή του βήματος ήταν 2, τότε το k θα γινόταν 3. Εμφανίζεται το μήνυμα "Η τιμή του k είναι: 2"
3η επανάληψη: k=3	Το k γίνεται 3 και εμφανίζεται στην οθόνη "Η τιμή του k είναι: 3"
.....
10η επανάληψη: k=10	Το k γίνεται 10, εκτελείται η εντολή μέσα στην επανάληψη.
11η επανάληψη: k=11	Το k γίνεται 11, αλλά επειδή έχει ξεπεράσει το όριο τερματισμού της επανάληψης, που είναι ο αριθμός 10, δεν εκτελείται η επανάληψη σταματάει.

Το τελευταίο κομμάτι της εντολή ΜΕ ΒΗΜΑ, στη περίπτωση που η τιμή του είναι 1, μπορεί να παραληφθεί και να μη γραφτεί καθόλου, διότι η εντολή από μόνη της δέχεται ως προεπιλεγμένη τιμή για το βήμα το 1. Στη περίπτωση που είναι οποιοσδήποτε άλλος αριθμός, πρέπει υποχρεωτικά να γράφεται.

Το πρόγραμμα στο παράδειγμα11 εμφανίζει όλες τους περιττούς αριθμούς από το -100 μέχρι το 100, χρησιμοποιώντας την εντολή επανάληψης ΓΙΑ.

Παράδειγμα3:

Αλγόριθμος παραδειγμα3

ΓΙΑ αριθμος ΑΠΟ -99 ΜΕΧΡΙ 100 ΜΕ ΒΗΜΑ 2

 ΓΡΑΨΕ αριθμος

ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

Τέλος παραδειγμα3

Για την επαναληπτική εντολή ΓΙΑ υπάρχουν και κάποιες προκαθορισμένες περιπτώσεις τιμών οι οποίες μας δίνουν συγκεκριμένα αποτελέσματα. Παρακάτω φαίνονται αυτές οι περιπτώσεις, ανάλογα με τις τιμές των μεταβλητών τιμή1, τιμή2 και τιμή3.

ΓΙΑ κ ΑΠΟ τιμή1 ΜΕΧΡΙ τιμή2 ΜΕ ΒΗΜΑ τιμή3

 εντολή-1

 εντολή-2

ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

- Αν $τιμή1=τιμή2$, ανεξάρτητα από την τιμή3 η επανάληψη θα τρέξει **μία φορά**.
- Αν $τιμή1<τιμή2$ και $τιμή3<0$ η επανάληψη δεν θα τρέξει **καμία φορά**.
- Αν $τιμή1>τιμή2$ και $τιμή3>0$ η επανάληψη δεν θα τρέξει **καμία φορά**.
- Αν $τιμή3=0$ τότε προκύπτει **ατέρμονος βρόγχος** και η επανάληψη δεν τελειώνει ποτέ (οπότε παραβιάζει και το αλγοριθμικό κριτήριο της περατότητας, όπως είδαμε στο κεφάλαιο με τα κριτήρια αλγορίθμων)

ΕΝΤΟΛΗ ΟΣΟ:

ΟΣΟ <συνθήκη> ΕΠΑΝΑΛΑΒΕ

 εντολή-1

 εντολή-2

 εντολή-3

 ...

 εντολή-ν

ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

Αυτό που κάνει η εντολή επανάληψης ΟΣΟ, είναι να εκτελεί συνέχεια τις εντολές που υπάρχουν στη δομή της (μέχρι το ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ), για όσο διάστημα ισχύει η <συνθήκη>. Στο πρόγραμμα που ακολουθεί, υπολογίζεται η προπέδια του 3, χρησιμοποιώντας την εντολή επανάληψης ΟΣΟ.

Παράδειγμα1:

Αλγόριθμος παράδειγμα1

πολ1 ← 3

πολ2 ← 1

ΟΣΟ πολ2 <= 10 ΕΠΑΝΑΛΑΒΕ

 γινόμενο ← πολ2 * πολ1

 ΕΜΦΑΝΙΣΕ πολ2, '*', πολ1, '=', γινόμενο

 πολ2 ← πολ2 + 1

ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

ΓΡΑΨΕ 'Αυτή ήταν η προπέδια του', πολ1

ΤΕΛΟΣ παράδειγμα1

Στο παρακάτω πίνακα αναλύεται ο τρόπος εκτέλεσης την εντολής ΟΣΟ, στο παράδειγμα1

ΕΠΑΝΑΛΗΨΕΙΣ	ΤΡΟΠΟΣ ΕΚΤΕΛΕΣΗΣ ΕΠΑΝΑΛΗΨΗΣ
1η επανάληψη: πολ2=1	Η μεταβλητή πολ2 παίρνει την τιμή 1. Γίνεται έλεγχος αν ισχύει η συνθήκη πολ2<=10. Η συνθήκη ισχύει, οπότε εκτελούνται οι εντολές μέσα στην επανάληψη: γινόμενο= 1 * 3 ΓΡΑΨΕ 1 * 3 = 3 πολ2 = 1 + 1 οπότε η πολ2 γίνεται πλέον πολ2=2
2η επανάληψη: πολ2=2	Η μεταβλητή πολ2 έχει πλέον την τιμή 2. Γίνεται έλεγχος αν ισχύει η συνθήκη πολ2<=10. Η συνθήκη ισχύει, οπότε εκτελούνται οι εντολές μέσα στην επανάληψη: γινόμενο= 2 * 3 ΓΡΑΨΕ 2 * 3 = 6 πολ2 = 2 + 1 οπότε η πολ2 γίνεται πλέον πολ2=3
3η επανάληψη: πολ2=3	Η μεταβλητή πολ2 έχει πλέον την τιμή 3. Γίνεται έλεγχος αν ισχύει η συνθήκη πολ2<=10. Η συνθήκη ισχύει, οπότε εκτελούνται οι εντολές μέσα στην επανάληψη: γινόμενο= 3 * 3 ΓΡΑΨΕ 3 * 3 = 9 πολ2 = 3 + 1 οπότε η πολ2 γίνεται πλέον πολ2=4

.....
10η επανάληψη: πολ2=10	<p>Η μεταβλητή πολ2 έχει πλέον την τιμή 10. Γίνεται έλεγχος αν ισχύει η συνθήκη πολ2<=10. Η συνθήκη ισχύει, οπότε εκτελούνται οι εντολές μέσα στην επανάληψη:</p> <p>γινόμενο= 10 * 3 ΓΡΑΨΕ 10* 3 =30 πολ2 = 10 + 1 οπότε η πολ2 γίνεται πλέον πολ2=11</p>
11η επανάληψη: πολ2=11	<p>Η μεταβλητή πολ2 έχει πλέον την τιμή 2. Γίνεται έλεγχος αν ισχύει η συνθήκη πολ2<=10. Η συνθήκη δεν ισχύει, οπότε δεν εκτελούνται οι εντολές μέσα στην επανάληψη. Η συνέχεια του προγράμματος συνεχίζεται μετά το ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ, οπότε εκτελείται η εντολή:</p> <p>ΓΡΑΨΕ 'Αυτή ήταν η προπέδια του', πολ1 όπου πολ1=3</p>

ΕΝΤΟΛΗ ΜΕΧΡΙΣ_ΟΤΟΥ:

ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ

εντολή-1

εντολή-2

...

εντολή-ν

ΜΕΧΡΙΣ_ΟΤΟΥ <συνθήκη>

Η εντολή ΜΕΧΡΙΣ_ΟΤΟΥ δουλεύει με τον αντίθετο τρόπο απ' ότι η εντολή ΟΣΟ. Δηλαδή πρώτα εκτελούνται οι εντολές που βρίσκονται μέσα στην επανάληψη και έπειτα γίνεται ο έλεγχος αν ισχύει η συνθήκη ή όχι. **Στη περίπτωση που ισχύει η συνθήκη, τότε σταματάει η επανάληψη και συνεχίζεται η εκτέλεση των εντολών έξω από την επανάληψη.** Συνήθως η εντολή ΜΕΧΡΙΣ_ΟΤΟΥ δέχεται για συνθήκη την αντίθετη συνθήκη από την επιθυμητή, διότι προκαλεί την έξοδο από την επανάληψη στη περίπτωση που ισχύει. Πχ αν θέλουμε να υπολογίζεται η τετραγωνική ρίζα του αριθμού που μας δίνει ο χρήστης, μέχρι να μας δώσει τον αριθμό 1, το πρόγραμμα θα ήταν:

ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ

ΓΡΑΨΕ 'Δώσε τον αριθμό:'

ΔΙΑΒΑΣΕ αριθμος

τετρ_ριζα ← Τ_Ρ(αριθμός)

ΓΡΑΨΕ 'Η τετραγωνική ρίζα του ',αριθμός, 'είναι ', τετρ_ριζα

ΜΕΧΡΙΣ_ΟΤΟΥ αριθμός = 1

Δηλαδή οι εντολές μέσα στην επανάληψη θα εκτελούνται μέχρι να πληκτρολογήσει ο χρήστης τον αριθμό 1. Όταν ο χρήστης πληκτρολογήσει 1 τότε θα υπολογιστεί η τετραγωνική ρίζα του 1, θα εμφανίσει το αποτέλεσμα, θα ελέγξει αν ισχύει η συνθήκη 'αριθμός=1' και εφόσον ισχύει θα σταματήσει η εκτέλεση της επανάληψης.

Το πρόγραμμα στο παράδειγμα2 προσθέτει διάφορους αριθμούς που δίνει ο χρήστης, έως ότου δώσει τον αριθμό 0.

Παράδειγμα2:

Αλγόριθμος παράδειγμα2

σύνολο $\leftarrow 0$

ΕΜΦΑΝΙΣΕ 'Δώσε ένα αριθμό:'

ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ

 ΔΙΑΒΑΣΕ μεταβλητή1

 σύνολο \leftarrow σύνολο + μεταβλητή1

ΜΕΧΡΙΣ_ΟΤΟΥ μεταβλητή1 = 0

ΕΜΦΑΝΙΣΕ 'Το άθροισμα των αριθμών που πληκτρολογήθηκε είναι:', σύνολο

ΤΕΛΟΣ παράδειγμα2

ΜΕΤΡΗΤΗΣ

Όπως βλέπουμε στις περισσότερες περιπτώσεις των εντολών ΟΣΟ και ΜΕΧΡΙΣ_ΟΤΟΥ, χρησιμοποιείται μια έκφραση μέσα στην επανάληψη της μορφής:

<μεταβλητή συνθήκης> \leftarrow *<μεταβλητή συνθήκης>* + *<τιμή>*

Η εντολή αυτή παίζει το ρόλο του 'μετρητή' μέσα στην δομή της επανάληψης και γι' αυτό στον προγραμματισμό χαρακτηρίζεται ως μετρητής η μεταβλητή που τροποποιείται. Αν δεν συνέβαινε αυτή η τροποποίηση και η *<μεταβλητή συνθήκης>* διατηρούσε την ίδια τιμή σε όλες τις επαναλήψεις, τότε η επανάληψη δε θα τερματιζόταν ποτέ και κατά συνέπεια δε θα συνεχιζόταν και η εκτέλεση του προγράμματος. Αυτή η έκφραση είναι πολύ σημαντική στις περισσότερες περιπτώσεις των εντολών ΟΣΟ και ΜΕΧΡΙΣ_ΟΤΟΥ και χρησιμοποιείται σχεδόν πάντα σε αυτές τις δύο δομές επανάληψης.

Αντιθέτως η εντολή ΓΙΑ, δεν απαιτεί τη χρήση αυτής της έκφρασης, αφού από μόνη της ο τρόπος λειτουργίας της εφαρμόζει αυτή την αλλαγή στον μετρητή (ΜΕ ΒΗΜΑ *<τιμή>*). Με άλλα λόγια η έκφραση αυτή παίζει για τις εντολές ΟΣΟ και ΜΕΧΡΙΣ_ΟΤΟΥ, το ρόλο που παίζει η εντολή ΜΕ ΒΗΜΑ για την ΓΙΑ.

ΣΥΓΚΡΙΣΗ ΕΝΤΟΛΩΝ ΕΠΑΝΑΛΗΨΗΣ

Βασική αρχή και για τρεις εντολές επανάληψης είναι ότι και οι τρεις μπορούν να χρησιμοποιηθούν σε οποιαδήποτε περίπτωση απαιτείται κάποια εντολή επανάληψης. Δεν υπάρχει δηλαδή κάποιος κανόνας σε ποιες περιπτώσεις χρησιμοποιούμε τη μία εντολή και σε ποιες την άλλη. Μπορούμε να χρησιμοποιήσουμε όποια επανάληψη θέλουμε, αρκεί φυσικά να υλοποιήσουμε το επιθυμητό αποτέλεσμα. Πχ το πρόγραμμα στο παράδειγμα1 υπολογίζει τη προπέδια ενός αριθμού, χρησιμοποιώντας την εντολή επανάληψης ΟΣΟ. Παρακάτω, το ίδιο πρόγραμμα, έχει υλοποιηθεί, αυτή τη φορά με τις εντολές ΜΕΧΡΙΣ_ΟΤΟΥ και ΓΙΑ.

ΟΣΟ	ΜΕΧΡΙΣ_ΟΤΟΥ	ΓΙΑ
πολ1 \leftarrow 3 πολ2 \leftarrow 1 ΟΣΟ πολ2 <=10 ΕΠΑΝΑΛΑΒΕ γινόμενο \leftarrow πολ2 * πολ1	πολ1 \leftarrow 3 πολ2 \leftarrow 1 ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ γινόμενο \leftarrow πολ2 * πολ1	πολ1 \leftarrow 3 ΓΙΑ πολ2 ΑΠΟ 1 ΜΕΧΡΙ 10 γινόμενο \leftarrow πολ2 * πολ1 ΓΡΑΨΕ πολ2, '*', πολ1, '=', γινόμενο

ΓΡΑΨΕ πολ2,'*',πολ1,'=',γινόμενο
 πολ2 ← πολ2 + 1
 ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
 ΓΡΑΨΕ 'Η προπέδια του',πολ1

ΓΡΑΨΕ πολ2,'*',πολ1,'=',γινόμενο
 πολ2 ← πολ2+1
 ΜΕΧΡΙΣ_ΟΤΟΥ πολ2 = 11
 ΓΡΑΨΕ 'Η προπέδια του',πολ1

ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
 ΓΡΑΨΕ 'Η προπέδια του',πολ1

Βλέπουμε ότι και με τις τρεις εντολές επανάληψης παίρνουμε το ίδιο αποτέλεσμα. Παρ' όλη την ομοιότητα όμως και την έλλειψη περιορισμών τότε χρησιμοποιείται η μία ή άλλη εντολή επανάληψης; θα δούμε κάποιες κατευθύνσεις πότε είναι προτιμότερο να χρησιμοποιούμε κάποια επανάληψη έναντι της άλλης, χωρίς όμως να είναι δεσμευτικό.

Εντολή ΟΣΟ:

- Είναι η ποιο γενική εντολή επανάληψης και η ποιο συνηθισμένη. Χρησιμοποιείται στις περισσότερες περιπτώσεις έναντι των άλλων δυο.
- Απαιτεί αρκετά συχνά την αρχικοποίηση των μεταβλητών πριν χρησιμοποιηθούν μέσα στην επανάληψη (βλέπε παράδειγμα1 όπου η μεταβλητή πολ2 πρέπει να πάρει τη τιμή 1 πριν χρησιμοποιηθεί μέσα στην επανάληψη).
- Τις περισσότερες φορές απαιτείται τη χρήση μετρητή.

Εντολή ΜΕΧΡΙΣ_ΟΤΟΥ:

1. Χρησιμοποιείται περισσότερο όταν θέλουμε οι εντολές της επανάληψης να εκτελεστούν τουλάχιστον μια φορά.
2. Η συνθήκη της αρκετά συχνά είναι ακριβώς αντίθετη από αυτή της εντολής ΟΣΟ.
3. Οι εντολές μέσα στην επανάληψη **εκτελούνται τουλάχιστον μια φορά**.
4. Τις περισσότερες φορές απαιτείται τη χρήση μετρητή.

Εντολή ΓΙΑ:

1. Χρησιμοποιείται περισσότερο σε περιπτώσεις όπου θέλουμε η συνθήκη να πάρει ένα συγκεκριμένο εύρος τιμών και δεν θέλουμε σε καμία περίπτωση να ξεφύγει από αυτό.
2. Χρησιμοποιείται περισσότερο στις δομή πινάκων, που θα δούμε πιο μετά.
3. Με την εντολή ΓΙΑ, συνήθως ο κώδικας ελαττώνεται κατά 2-3 γραμμές μέσα στην επανάληψη σε σχέση με τις άλλες εντολές.
4. Σπάνια απαιτείται η χρήση μετρητή.

Όπως αναφέραμε οι μεγάλη διαφορά της εντολής ΓΙΑ με τις εντολές ΟΣΟ και ΜΕΧΡΙ_ΟΤΟΥ, είναι η ύπαρξη μετρητή. Η εντολή ΓΙΑ μπορούμε να πούμε ότι έχει έναν ενσωματωμένο μετρητή που αυξάνει ή μικραίνει με την εντολή ΜΕ ΒΗΜΑ. Αντίθετα, οι εντολές ΟΣΟ και ΜΕΧΡΙΣ_ΟΤΟΥ δεν εμπεριέχουν κάποιο μετρητή, γι' αυτό και χρησιμοποιούμε εμείς κάποιον (μετρητής ← μετρητής + <τιμή>). Αν θέλαμε να μετατρέψουμε τις εντολές ΟΣΟ και ΜΕΧΡΙΣ_ΟΤΟΥ να δουλεύουν ακριβώς με τον ίδιο τρόπο όπως η ΓΙΑ, τότε θα γίνονταν:

Εντολή ΓΙΑ	Εντολή ΟΣΟ	Εντολή ΜΕΧΡΙΣ_ΟΤΟΥ
ΓΙΑ μετρητής ΑΠΟ 1 ΜΕΧΡΙ 100 ΜΕ ΒΗΜΑ 1 <εντολές> ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ	μετρητής ← 1 ΟΣΟ μετρητής <=100 ΕΠΑΝΑΛΑΒΕ <εντολές> μετρητής ← μετρητής + 1 ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ	μετρητής ← 1 ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ <εντολές> μετρητής ← μετρητής + 1 ΜΕΧΡΙΣ_ΟΤΟΥ μετρητής > 100

ΥΠΟΛΟΓΙΣΜΟΣ ΕΛΑΧΙΣΤΟΥ, ΜΕΓΙΣΤΟΥ

Σύνταξη Υπολογισμού Ελαχίστου:

$min \leftarrow <αρχική\ τιμή\ συνόλου>$

[αρχή_επανάληψη]

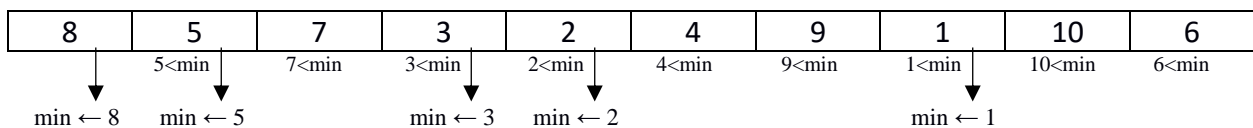
 Αν $<τιμή\ συνόλου> < min$ τότε

$min \leftarrow <τιμή\ συνόλου>$

 Τέλος_αν

[τέλος_επανάληψης]

Αυτό το κομμάτι αλγορίθμου είναι ο γενικός τρόπος υπολογισμού της ελάχιστης τιμής μέσα από ένα σύνολο τιμών που παίρνει η *<μεταβλητή>*. Τις περισσότερες φορές ο υπολογισμός του ελαχίστου, αλλά και του μεγίστου που θα δούμε στη συνέχεια, εμπεριέχεται μέσα σε μια εντολή επανάληψης (Παράδειγμα1), με μόνη διαφορά ως προς τη σύνταξη που είδαμε παραπάνω, την αρχικοποίηση της μεταβλητής min, η οποία μπαίνει έξω και πριν την επανάληψη.



Τα βήματα για τον υπολογισμό του ελαχίστου, είναι τα εξής:

- Δημιουργούμε μια μεταβλητή που θα περιέχει την ελάχιστη τιμή (min)
- Δίνουμε σε αυτή τη μεταβλητή την πρώτη τιμή του συνόλου
- Ελέγχουμε αν η μεταβλητή min είναι **μικρότερη** από την επόμενη τιμή του συνόλου. Στη περίπτωση που είναι, τότε για νέα τιμή στη μεταβλητή min δίνουμε την τρέχουσα τιμή που είναι μικρότερη.
- Επαναλαμβάνεται το βήμα iii) για την επόμενη τιμή, μέχρι να ελεγχθεί όλο το σύνολο τιμών.

Παράδειγμα1

Πρόγραμμα Παράδειγμα1

Μεταβλητές

 Ακέραιες: αριθμός, min

Αρχή

 Γράψε 'Δώσε ένα αριθμό:'

 Διάβασε αριθμός

$min \leftarrow \text{αριθμός}$

 Οσο αριθμός > 0 Επανάλαβε

 Αν αριθμός < min τότε

$min \leftarrow \text{αριθμός}$

 Τέλος_αν

 Γράψε 'Δώσε ένα αριθμό:'

 Διάβασε αριθμός

 Τέλος_Επανάληψης

 Γράψε 'Ο μικρότερος αριθμός είναι:', min

Τέλος_Προγράμματος

Στο παράδειγμα ο χρήστης δίνει συνεχόμενους αριθμούς, έως ότου δώσει κάποιον αρνητικό και εμφανίζεται ο μικρότερος από αυτούς που έδωσε. Οι συνεχόμενοι αριθμοί που δίνει ο χρήστης παίζουν το ρόλο του συνόλου, οπότε σε κάθε επανάληψη θα πρέπει να ελέγχουμε τον αριθμό που έδωσε ο χρήστης αν είναι μικρότερος ή όχι του min.

Σύνταξη Υπολογισμού Μεγίστου:

$max \leftarrow \langle \text{αρχική τιμή συνόλου} \rangle$

[αρχή_επανάληψη]

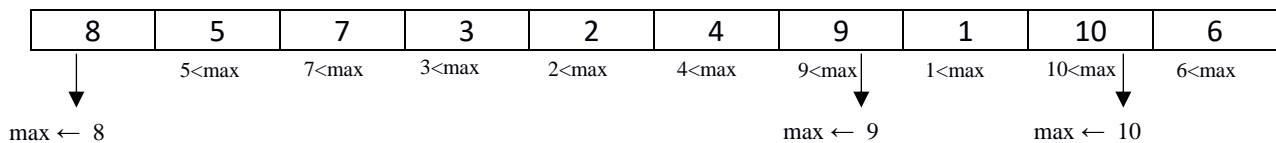
Αν $\langle \text{τιμή συνόλου} \rangle > max$ τότε

$max \leftarrow \langle \text{τιμή συνόλου} \rangle$

Τέλος_αν

[τέλος_επανάληψης]

Ο υπολογισμός του μεγίστου γίνεται ακριβώς με τον ίδιο τρόπο με αυτόν του ελαχίστου, με τη διαφορά ότι στον έλεγχο εξετάζουμε αν η τιμή της *<μεταβλητής>* είναι μεγαλύτερη από την εκάστοτε μέγιστη τιμή (max).



Στο Παράδειγμα2 βλέπουμε τη μετατροπή του Παραδείγματος1, για τον υπολογισμό του μεγίστου αυτή τη φορά.

Παράδειγμα2

Πρόγραμμα Παράδειγμα2

Μεταβλητές

Ακέραιες: αριθμός, max

Αρχή

Γράψε 'Δώσε ένα αριθμό:'

Διάβασε αριθμός

$max \leftarrow \text{αριθμός}$

Οσο αριθμός > 0 Επανάλαβε

Αν αριθμός > max τότε

$max \leftarrow \text{αριθμός}$

Τέλος_αν

Γράψε 'Δώσε ένα αριθμό:'

Διάβασε αριθμός

Τέλος_Επανάληψης

Γράψε 'Ο μεγαλύτερος αριθμός είναι:', max

Τέλος_Προγράμματος

ΥΠΟΛΟΓΙΣΜΟΣ ΑΘΡΟΙΣΜΑΤΟΣ, ΓΙΝΟΜΕΝΟΥ

Σύνταξη:

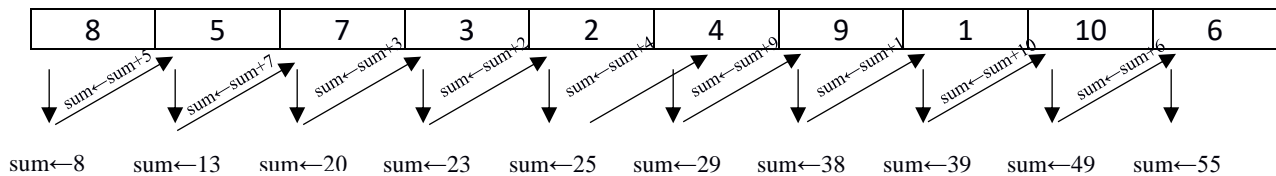
$sum \leftarrow 0$ //Αρχικοποίηση

[αρχή_επανάληψη]

$sum \leftarrow sum + <τιμή\ συνόλου>$ //Κυρίως μέρος

[τέλος_επανάληψης]

Όταν θέλουμε να υπολογίσουμε το άθροισμα ενός συνόλου αριθμών, αυτό που κάνουμε είναι να δημιουργήσουμε μια μεταβλητή (πχ sum) στην οποία δίνουμε ως αρχική τιμή το 0. Έπειτα προσθέτουμε σε αυτή την μεταβλητή μια-μια όλες τις τιμές του συνόλου ($sum \leftarrow sum + τιμή$). Έτσι η μεταβλητή (sum) που δημιουργήσαμε, αφού προστεθούν σε αυτή όλες οι τιμές του συνόλου, στο τέλος θα έχει το άθροισμα όλου του συνόλου.



Με αντίστοιχο τρόπο υπολογίζεται και το γινόμενο μόνο που αντί για πρόσθεση γίνεται η πράξη του πολλαπλασιασμού.

Σύνταξη:

$γινόμενο \leftarrow 0$ //Αρχικοποίηση

[αρχή_επανάληψη]

$γινόμενο \leftarrow γινόμενο * <τιμή\ συνόλου>$ //Κυρίως μέρος

[τέλος_επανάληψης]

Στο παράδειγμα που ακολουθεί, υπολογίζεται το άθροισμα τριών αριθμών που δίνει ο χρήστης.

Παράδειγμα3

Πρόγραμμα Παράδειγμα3

Μεταβλητές

Ακέραιες: sum, κ, n

Αρχή

$sum \leftarrow 0$

$κ \leftarrow 0$

Γράψε 'Δώσε Αριθμό:'

Διάβασε n

Όσο $κ \leq 3$ Επανάλαβε

$sum \leftarrow sum + n$

Γράψε 'Δώσε Αριθμό:'

Διάβασε n

$κ \leftarrow κ + 1$

Τέλος_Επανάληψης

Γράψε 'Το άθροισμα των τριών αριθμών είναι:', sum
Τέλος_Προγράμματος

Στο Παράδειγμα3 παρατηρούμε τα εξής:

- αρχικοποιούμε τη τιμή που θα περιέχει το άθροισμα, με την τιμή 0.
- Κάθε φορά που αλλάζει η τιμή της μεταβλητής n που θέλουμε να υπολογίσουμε το άθροισμα (Διάβασε n), το άθροισμα sum αυξάνει κατά n.

ΥΠΟΛΟΓΙΣΜΟΣ ΠΛΗΘΟΥΣ

Όταν θέλουμε να υπολογίσουμε το πλήθος ενός συνόλου, η διαδικασία είναι παρόμοια με αυτή του αθροίσματος, με τη μόνη διαφορά ότι δεν προσθέτουμε την τιμή, αλλά αυξάνουμε την μεταβλητή μας κατά ένα. Αν για παράδειγμα στη μεταβλητή count θέλουμε να υπολογίσουμε το πλήθος των τιμών, αυτό που κάνουμε είναι να αυξάνουμε την μεταβλητή count κατά 1 κάθε φορά που βρίσκεται σε νέο αριθμό. Και εδώ τις περισσότερες φορές ο υπολογισμός τους πλήθους συνοδεύεται με τη χρήση μιας εκ των τριών δομών επανάληψης.

Η διαδικασία για τον υπολογισμό του πλήθους είναι ίδια με αυτή του μετρητή που είδαμε στις επανάληψεις, γι' αυτό πολλές φορές ο ίδιο ο μετρητής παίζει το ρόλο του υπολογισμού του πλήθους στις ασκήσεις.

Παράδειγμα4

Πρόγραμμα Παράδειγμα4

Μεταβλητές

Ακέραιες: count, αριθμός

Αρχή

count \leftarrow 0

Γράψε 'Δώσε ένα αριθμό:'

Διάβασε αριθμός

Οσο αριθμός > 0 Επανάλαβε

count \leftarrow count+1

Γράψε 'Δώσε ένα αριθμό:'

Διάβασε αριθμός

Τέλος_Επανάληψης

Γράψε 'Το σύνολο των αριθμών που δόθηκαν είναι:', count

Τέλος_Προγράμματος

Στο Παράδειγμα4 ο χρήστης δίνει διάφορους αριθμούς έως ότου δώσει έναν αρνητικό και υπολογίζεται το πλήθος των αριθμών που δόθηκαν. Στο Παράδειγμα4 παρατηρούμε τα εξής:

1. Θέλουμε να υπολογίσουμε το πλήθος των αριθμών που αλλάζει η μεταβλητή 'αριθμός'.
2. Αρχικοποιούμε τη μεταβλητή (count) που περιέχει το πλήθος των τιμών της μεταβλητής 'αριθμός'.
3. Μετά από κάθε αλλαγή της τιμής της μεταβλητής 'αριθμός' (Διάβασε αριθμός), αυξάνουμε τη τιμή του πλήθους κατά 1(count \leftarrow count+1), αφού η μεταβλητή 'αριθμός' έλαβε νέα τιμή.

ΥΠΟΛΟΓΙΣΜΟΣ ΜΕΣΟΥ ΟΡΟΥ

Βασική προϋπόθεση για τον υπολογισμό του μέσου όρου, είναι ο υπολογισμός του αθροίσματος (sum) και του πλήθους (count). Άρα πριν υπολογίσουμε το μέσο όρο, πρέπει να εφαρμόσουμε την διαδικασία υπολογισμού του πλήθους και του αθροίσματος και στη συνέχεια διαιρώντας τα μεταξύ τους, προκύπτει ο μέσος όρος ($\text{μέσος_όρος} = \text{άθροισμα} / \text{πλήθος}$). Το πλήθος πολλές φορές μας δίνεται στις ασκήσεις ως δεδομένο και δε χρειάζεται να το υπολογίσουμε, σε περίπτωση όμως που δεν δίνεται πρέπει να εκτελεστεί κανονικά η διαδικασία υπολογισμού του.

Παράδειγμα5

Πρόγραμμα Παράδειγμα5

Μεταβλητές

Ακέραιες: count, sum, αριθμός

Πραγματικές: MO

Αρχή

count \leftarrow 0

sum \leftarrow 0

Γράψε 'Δώσε αριθμό'

Διάβασε αριθμός

Όσο αριθμός \geq 0 Επανάλαβε

count \leftarrow count + 1

sum \leftarrow sum + αριθμός

Γράψε 'Δώσε αριθμό'

Διάβασε αριθμός

Τέλος_Επανάληψης

MO \leftarrow sum/count

Γράψε 'Ο μέσος όρος είναι:', MO

Τέλος_Προγράμματος

Στο Παράδειγμα5 υπολογίζεται ο μέσος όρος των αριθμών που δίνει ο χρήστης έως ότου δώσει ένα αρνητικό αριθμό. Στο παράδειγμα βλέπουμε τα εξής στοιχεία:

1. Υπολογίζεται το πλήθος των αριθμών (count) σύμφωνα με τη μέθοδο που είδαμε πριν.
2. Υπολογίζεται το άθροισμα των αριθμών (sum) σύμφωνα με τη μέθοδο που είδαμε πριν.
3. Αφού τελειώσει η επανάληψη και έξω αυτήν, υπολογίζεται ο μέσος όρος σύμφωνα με τον τύπο $\text{μέσος_όρος} = \text{άθροισμα} / \text{πλήθος}$.

Παρατηρήσεις

- Στη πλειοψηφία των περιπτώσεων που απαιτείται κάποιος από τους παραπάνω υπολογισμούς (άθροισμα, μέγιστο, ελάχιστο κλπ.) χρειάζεται η χρήση κάποιας επανάληψης.
- Τις περισσότερες φορές το ρόλο του 'συνόλου τιμών' παίζει κάποια μονάδα (όπως βαθμοί, θερμοκρασίες, άνθρωποι κλπ.) της οποίας την τιμή μας ζητάει η άσκηση να δίνουμε εμείς (πχ 'δώσε 10 αριθμούς', 'δώσε 8 θερμοκρασίες' κλπ).

ΑΣΚΗΣΕΙΣ

1. Να γραφεί πρόγραμμα και με τις 3 εντολές επανάληψης, που εμφανίζει τους αριθμούς από το 100 μέχρι το 0.
2. Να γραφεί πρόγραμμα και με τις 3 εντολές επανάληψης, που εμφανίζει το άθροισμα των αριθμών από το 0 μέχρι το 100.
3. Να γραφεί πρόγραμμα με τις εντολές ΟΣΟ και ΜΕΧΡΙΣ_ΟΤΟΥ, που διαβάσει διάφορους αριθμούς που δίνει ο χρήστης έως ότου δώσει το 0 και υπολογίζει και εμφανίζει τον μικρότερο.
4. Να γραφεί πρόγραμμα με τις εντολές ΟΣΟ και ΜΕΧΡΙΣ_ΟΤΟΥ, που διαβάσει διάφορους αριθμούς που δίνει ο χρήστης έως ότου δώσει το 0 και υπολογίζει και εμφανίζει τον μεγαλύτερο.
5. Να γραφεί πρόγραμμα και με τις 3 εντολές επανάληψης το οποίο διαιρεί έναν-έναν όλους τους αριθμούς από το 1 μέχρι το 100 με το 2 και μας εμφανίζει το ακέραιο μέρος του αποτελέσματος. (P_15)
6. Δίνεται ο αλγόριθμος:
 Αλγόριθμος Έλεγχος_Ανάθεσης
 Διάβασε x
 Όσο x > 1 επανάλαβε
 Αν x είναι άρτιος τότε
 x ← x/2
 αλλιώς
 x ← 3*x+1
 Τέλος_αν
 Τέλος_επανάληψης
 Αποτελέσματα //x//
 Τέλος Έλεγχος_Ανάθεσης
 Να γράψεις τα αποτελέσματα αυτού του αλγορίθμου για x=13, x=9 και x=22.
7. Σε ένα Λύκειο κάθε μαθητής αξιολογείται με βάση το μέσο όρο που θα έχει σε 5 βασικά μαθήματα. Να γραφεί ένας αλγόριθμος που θα διαβάσει τη βαθμολογία για τα 5 αυτά μαθήματα και θα υπολογίζει το μέσο όρο του μαθητή. (P_14)
8. Να γραφεί πρόγραμμα το οποίο διαβάσει το όνομα ενός μαθητή, τους βαθμούς του σε τρία μαθήματα και υπολογίζει και τυπώνει το μέσο όρο. Το πρόγραμμα να σταματάει, όταν για όνομα δοθεί το κενό. (P_13)
9. Να γραφεί πρόγραμμα που να υπολογίζει τη συνάρτηση $y(x) = \frac{x-3x}{2}$ για όλες τις τιμές του x από -1 έως 3 σε βήματα του 0.1 (P_12)
10. Πόσες φορές θα εκτελεστεί η παρακάτω επανάληψη (η ΜΕΧΡΙΣ_ΟΤΟΥ):
 ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
 A ← 0
 ΓΙΑ I ΑΠΟ 1 ΜΕΧΡΙ 5
 A ← A-1
 ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
 ΜΕΧΡΙΣ_ΟΤΟΥ A=0

ΜΟΝΟΔΙΑΣΤΑΤΟΙ ΠΙΝΑΚΕΣ

Μέχρι τώρα έχουμε δει προγράμματα με πολλές μεταβλητές οι περισσότερες από τις οποίες παίζουν το ρόλο ενδιάμεσων μεταβλητών προκειμένου να κάνουμε διάφορους υπολογισμούς. Επίσης έχουμε δει προγράμματα τα οποία περιέχουν αρκετές μεταβλητές και μάλιστα σε ορισμένα από αυτά τα προγράμματα κάποιες άλλαζαν συνέχεια τιμές με μια εντολή επανάληψης. Π.χ.

Αρχή_Επανάληψης

Γράψε 'Δώσε ένα νούμερο:'

Διάβασε x

Μέχρις_Ότου x=0

Σε παραδείγματα σαν το προηγούμενο η τιμή αποθηκεύετε στη μεταβλητή μέχρι ο χρήστης να δώσει μια νέα τιμή, οπότε και σβήνεται η παλιά τιμή για να αποθηκεύσει την καινούργια. Αν όμως θελήσουμε να εμφανίσουμε μια τιμή που ήδη έχει πάρει το x και την έχει σβήσει για να αποθηκεύσει μια νέα τιμή; Αν π.χ. θέλουμε να εμφανίσουμε τη 2η τιμή που πήρε το x όταν έτρεχε το πρόγραμμα και που τώρα την έχει σβήσει;

Σε τέτοιες περιπτώσεις χρησιμοποιούμε τους πίνακες προκειμένου να μη σβηστεί καμία ενδιάμεση τιμή που παίρνει η x αλλά τις αποθηκεύουμε όλες σε ένα πίνακα και τις **προσπελαύνουμε** όποτε θέλουμε.

Ένας πίνακας είναι μια δομή δεδομένων, αλλά μπορούμε να τον φανταστούμε σαν μια μεταβλητή η οποία παίρνει πολλές τιμές. Δηλαδή, αν έχουμε τις τιμές 3,6,4,9,17 και η ροή του προγράμματος μας επιβάλει να αποθηκευτούν σε μια μεταβλητή γ, τότε δηλώνουμε τη μεταβλητή γ ως ακέραιο πίνακα οπότε η μεταβλητή γ θα γίνει:

θέση: 1 2 3 4 5
γ:

3	6	4	9	17
---	---	---	---	----

όπου η θέση μας δείχνει σε ποια θέση της μεταβλητής βρίσκεται η κάθε τιμή. Δηλαδή η τιμή 3 θα βρίσκεται στη θέση 1 της μεταβλητής γ ή αλλιώς γ[1], η τιμή 4 θα βρίσκεται στη θέση 3 της μεταβλητής γ ή αλλιώς γ[3], η τιμή 9 στην γ[4] κλπ. Με αυτό τον τρόπο σε μια μεταβλητή μπορούμε να αποθηκεύσουμε πολλές τιμές και να **προσπελάσουμε** τη μεταβλητή που θέλουμε με τον εξής τρόπο: γ[θέση]. Βλέπουμε λοιπόν ότι με μία μόνο μεταβλητή αποθηκεύσαμε 5 τιμές και δε χρειάστηκε να δημιουργήσουμε 5 διαφορετικές μεταβλητές.

ΔΗΛΩΣΗ ΠΙΝΑΚΑ

Άρα λοιπόν ένα πίνακα μπορούμε να τον φανταστούμε σαν μια μεταβλητή που παίρνει πολλές τιμές. Έτσι λοιπόν όπως και στις μεταβλητές έτσι και στους πίνακες πρέπει να τους δηλώνουμε. Ο πίνακας που παίρνει ακέραιες τιμές θα δηλωθεί ως ακέραιος, αυτός που παίρνει πραγματικές ως πραγματικός, αυτός που παίρνει χαρακτήρες ως χαρακτήρα. Δεν μπορούμε να έχουμε πίνακες που έχουν μέσα τους

ταυτόχρονα δύο τύπους δεδομένων. Για παράδειγμα δεν μπορούμε να έχουμε ένα πίνακα που η μία θέση παίρνει έναν ακέραιο και κάποια άλλη θέση παίρνει κάποιο χαρακτήρα, όλες οι θέσεις πρέπει να έχουν τον ίδιο τύπο δεδομένων.

Κάθε πίνακας στη δήλωση του θα πρέπει να συνοδεύεται και από τον αριθμό των θέσεων που έχει. Δηλαδή αν θέλουμε ένα πίνακα p που παίρνει 30 ακέραιες τιμές τότε η δήλωση του θα είναι:

```
Πρόγραμμα A
Μεταβλητές
    ΑΚΕΡΑΙΕΣ: p[30]
Αρχή
.....
Τέλος_Προγράμματος
```

ΚΑΤΑΧΩΡΗΣΗ ΤΙΜΩΝ

Η καταχώρηση μιας τιμής σε ένα πίνακα γίνεται όπως σε μια μεταβλητή με τη διαφορά ότι στο πίνακα πρέπει να αναφέρουμε και τη θέση στην οποία θέλουμε να αποθηκευτεί η τιμή. Δηλαδή αν θέλουμε να αποθηκεύσουμε τη τιμή 5 στη τρίτη θέση του πίνακα x τότε θα γίνει:

```
Πρόγραμμα A
Μεταβλητές
    Ακέραιες: x[6]
Αρχή
    x[3] ← 5
Τέλος_Προγράμματος
```

Αν θέλαμε να αποθηκεύσουμε την τιμή 2 στην 7η θέση τότε θα προκύψει σφάλμα, διότι έχουμε δηλώσει ότι ο πίνακας έχει 6 θέσεις, οπότε δε γίνεται να αποθηκεύσουμε μια τιμή στην 7η θέση που δεν υπάρχει. Άρα μεγάλη προσοχή πρέπει να δίνεται στη δήλωση του πίνακα και στον αριθμό των θέσεων που θα δηλώνονται.

ΠΡΟΣΠΕΛΑΣΗ ΤΙΜΩΝ

Συνήθως τους πίνακες τους χρησιμοποιούμε για να αποθηκεύσουμε συνεχόμενες τιμές ενός είδους π.χ. “τις θερμοκρασίες μιας χώρας όλη την εβδομάδα” (αντί να χρησιμοποιήσουμε 7 μεταβλητές, μια για κάθε μέρα, χρησιμοποιούμε ένα πίνακα με 7 θέσεις), “τις αποδοχές ενός υπαλλήλου όλο το χρόνο” (ένα πίνακα με 12 θέσεις, μια θέση για κάθε μήνα), “τους βαθμούς 5 μαθημάτων” (έναν πίνακα με 5 θέσεις). Καταλαβαίνουμε λοιπόν ότι για να προσπελάσουμε όλες τις θέσεις ενός πίνακα απαιτείται η χρήση κάποιας εντολής επανάληψης.

Αν υποθέσουμε ότι έχουμε ένα πίνακα 10 θέσεων στον οποίο θέλουμε να καταχωρήσουμε 10 ονόματα που θα δίνει ο χρήστης, τότε θα πρέπει να χρησιμοποιήσουμε μια εντολή επανάληψης προκειμένου να καταχωρήσουμε αυτά τα 10 ονόματα στον πίνακα. Η πιο συνηθισμένη εντολή επανάληψης στους

πίνακες είναι η ΓΙΑ...ΑΠΟ...ΜΕΧΡΙ, χωρίς αυτό να σημαίνει ότι δεν μπορούμε να χρησιμοποιήσουμε και κάποια από τις άλλες δυο που γνωρίζουμε (ΟΣΟ, ΜΕΧΡΙΣ_ΟΤΟΥ). Οπότε αν θα θέλαμε να αποθηκεύσουμε σε ένα πίνακα 10 ονόματα που δίνει ο χρήστης, ο αλγόριθμος και με τις 3 εντολές επανάληψης θα ήταν ο εξής:

ΓΙΑ...ΑΠΟ...ΜΕΧΡΙ	ΟΣΟ	ΜΕΧΡΙΣ_ΟΤΟΥ
Πρόγραμμα Α Μεταβλητές Ακέραιες: i Χαρακτήρες: x[10] Αρχή Για i Απο 1 Μέχρι 10 Γράψε 'Δώσε όνομα:' Διάβασε x[i] Τέλος_Επανάληψης Τέλος_Προγράμματος	Πρόγραμμα Α Μεταβλητές Ακέραιες: i Χαρακτήρες: x[10] Αρχή i ← 1 Όσο i <= 10 Επανάλαβε Γράψε 'Δώσε όνομα:' Διάβασε x[i] i ← i + 1 Τέλος_Επανάληψης Τέλος_Προγράμματος	Πρόγραμμα Α Μεταβλητές Ακέραιες: i Χαρακτήρες: x[10] Αρχή i ← 1 Αρχή_Επανάληψης Γράψε 'Δώσε όνομα:' Διάβασε x[i] i ← i + 1 Μέχρις_Ότου i > 10 Τέλος_Προγράμματος

Βλέπουμε λοιπόν ότι η πιο σύντομη και εύχρηστη εντολή επανάληψης για την προσπέλαση πινάκων είναι η ΓΙΑ...ΑΠΟ...ΜΕΧΡΙ.

Είδαμε πως γίνεται η καταχώρηση τιμών σε ένα πίνακα χρησιμοποιώντας και τις τρεις εντολές επανάληψης. Στη συνέχεια θα δούμε πως εμφανίζουμε τα δεδομένα ενός πίνακα. Αν υποθέσουμε ότι έχουμε ένα πίνακα με τις θερμοκρασίες της προηγούμενης εβδομάδας μιας περιοχής. Δηλαδή:

	1	2	3	4	5	6	7
θ:	28.3	27.4	29.1	28.1	26.3	25.8	26.7

όπου η θερμοκρασία στη πρώτη θέση αντιστοιχεί στη θερμοκρασία της Δευτέρας, στη δεύτερη θέση στη θερμοκρασία της Τρίτης κλπ. Οπότε η εμφάνιση των θερμοκρασιών και με τις τρεις εντολές επανάληψης θα είναι:

ΓΙΑ...ΑΠΟ...ΜΕΧΡΙ	ΟΣΟ	ΜΕΧΡΙΣ_ΟΤΟΥ
Για i Απο 1 Μέχρι 7 Γράψε i, 'η μέρα:', θ[i] Τέλος_Επανάληψης	i ← 1 Όσο i <= 7 Επανάλαβε Γράψε i, 'η μέρα:', θ[i] i ← i + 1 Τέλος_Επανάληψης	i ← 1 Αρχή_Επανάληψης Γράψε i, 'η μέρα:', θ[i] i ← i + 1 Μέχρις_Ότου i > 7

Πέρα από την αποθήκευση και εμφάνιση τιμών μέσα σε ένα πίνακα, όπως και με τις μεταβλητές έτσι και εδώ μπορούν να γίνουν και πράξεις με τα στοιχεία του πίνακα και γενικά η συμπεριφορά των στοιχείων του πίνακα είναι ακριβώς η ίδια με αυτή των μεταβλητών. Δηλαδή, μπορούμε να υπολογίσουμε τη μέγιστη τιμή ενός πίνακα, το μέσο όρο των τιμών του, να κάνουμε διάφορες πράξεις μεταξύ των στοιχείων, να προσθέσουμε να αφαιρέσουμε μια μεταβλητή στα στοιχεία, και άλλα.

ΕΛΑΧΙΣΤΟ, ΜΕΓΙΣΤΟ ΣΕ ΕΝΑ ΠΙΝΑΚΑ

Στη συνέχεια θα δούμε πως υπολογίζουμε την ελάχιστη και μέγιστη τιμή ενός πίνακα και συγκεκριμένα θα χρησιμοποιήσουμε το παράδειγμα των θερμοκρασιών προκειμένου να βρούμε την ελάχιστη και τη μέγιστη θερμοκρασία του πίνακα θ .

ΕΛΑΧΙΣΤΟ		
ΓΙΑ...ΑΠΟ...ΜΕΧΡΙ	ΟΣΟ	ΜΕΧΡΙΣ_ΟΤΟΥ
$\min \leftarrow \theta[1]$ Για i Απο 2 Μέχρι 7 Αν $\theta[i] < \min$ τότε $\min \leftarrow \theta[i]$ Τέλος_αν Τέλος_Επανάληψης	$\min \leftarrow \theta[1]$ $i \leftarrow 2$ Όσο $i \leq 7$ Επανάλαβε Αν $\theta[i] < \min$ τότε $\min \leftarrow \theta[i]$ Τέλος_αν $i \leftarrow i + 1$ Τέλος_Επανάληψης	$\min \leftarrow \theta[1]$ $i \leftarrow 2$ Αρχή_Επανάληψης Αν $\theta[i] < \min$ τότε $\min \leftarrow \theta[i]$ Τέλος_αν $i \leftarrow i + 1$ Μέχρις_Ότου $i > 7$

ΜΕΓΙΣΤΟ		
ΓΙΑ...ΑΠΟ...ΜΕΧΡΙ	ΟΣΟ	ΜΕΧΡΙΣ_ΟΤΟΥ
$\max \leftarrow \theta[1]$ Για i Απο 2 Μέχρι 7 Αν $\theta[i] > \max$ τότε $\max \leftarrow \theta[i]$ Τέλος_αν Τέλος_Επανάληψης	$\max \leftarrow \theta[1]$ $i \leftarrow 2$ Όσο $i \leq 7$ Επανάλαβε Αν $\theta[i] > \max$ τότε $\max \leftarrow \theta[i]$ Τέλος_αν $i \leftarrow i + 1$ Τέλος_Επανάληψης	$\max \leftarrow \theta[1]$ $i \leftarrow 2$ Αρχή_Επανάληψης Αν $\theta[i] > \max$ τότε $\max \leftarrow \theta[i]$ Τέλος_αν $i \leftarrow i + 1$ Μέχρις_Ότου $i > 7$

Βλέπουμε ότι και στον υπολογισμό του ελαχίστου και σε αυτόν του μεγίστου, αυτό που κάνουμε είναι οι \max (μέγιστο) και \min (ελάχιστο) να αρχικοποιούνται παίρνοντας ως πρώτη τιμή το πρώτο στοιχείο του πίνακα, και στη συνέχεια οι επαναλήψεις να ξεκινούν από το δεύτερο στοιχείο του πίνακα.

ΜΕΣΟΣ ΟΡΟΣ ΣΕ ΕΝΑ ΠΙΝΑΚΑ

Αν τώρα θελήσουμε να υπολογίσουμε τη μέση θερμοκρασία (avg) του πίνακα των θερμοκρασιών θ , τότε ο αλγόριθμος θα ήταν:

ΓΙΑ...ΑΠΟ...ΜΕΧΡΙ	ΟΣΟ	ΜΕΧΡΙΣ_ΟΤΟΥ
$\text{sum} \leftarrow 0$	$\text{sum} \leftarrow 0$	$\text{sum} \leftarrow 0$

Για i Απο 1 Μέχρι 7
 sum \leftarrow sum + $\theta[i]$
 Τέλος_Επανάληψης
 avg \leftarrow sum/7

i \leftarrow 1
 Όσο i \leq 7 Επανάλαβε
 sum \leftarrow sum + $\theta[i]$
 i \leftarrow i + 1
 Τέλος_Επανάληψης
 avg \leftarrow sum/7

i \leftarrow 1
 Αρχή_Επανάληψης
 sum \leftarrow sum + $\theta[i]$
 i \leftarrow i + 1
 Μέχρις_Ότου i > 7
 avg \leftarrow sum/7

Όπως βλέπουμε αρχικά υπολογίζεται το άθροισμα των τιμών του πίνακα μέσα στην επανάληψη και στη συνέχεια το άθροισμα αυτό διαιρείται με το πλήθος των θέσεων του πίνακα.

ΠΡΟΣΘΕΤΕΣ ΔΥΝΑΤΟΤΗΤΕΣ ΠΙΝΑΚΩΝ

Πέρα όμως από τη μαζική αποθήκευση τιμών και την συμπεριφορά των στοιχείων του πίνακα σαν να είναι μεταβλητές, οι πίνακες παρέχουν κάποιες πρόσθετες δυνατότητες ιδιαίτερα χρήσιμες τις οποίες δεν μας παρέχουν οι απλές μεταβλητές. Τέτοιες δυνατότητες είναι η Αναζήτηση και η Ταξινόμηση.

ΑΝΑΖΗΤΗΣΗ

Με τον όρο Αναζήτηση, εννοούμε το έλεγχο των στοιχείων ενός πίνακα προκειμένου να δούμε αν υπάρχει η τιμή που επιθυμούμε. Δηλαδή όταν θέλουμε να δούμε αν υπάρχει κάποια συγκεκριμένη τιμή μέσα σε ένα πίνακα, τότε έχουμε τη δυνατότητα να εφαρμόσουμε έναν αλγόριθμο αναζήτησης προκειμένου να δούμε αν υπάρχει αυτή η τιμή.

Υπάρχουν πολλοί αλγόριθμοι αναζήτησης με διαφορετικούς τρόπους λειτουργίας ο καθένας. Εμείς όμως θα δούμε μόνο τη σειριακή αναζήτηση, η οποία είναι η πιο απλή αναζήτηση αλλά και η λιγότερο αποδοτική.

Γενική Μορφή Σειριακής Αναζήτησης:

Αλγόριθμος ΣειριακήΑναζήτηση

Δεδομένα n, table, key

done \leftarrow ψευδής

position \leftarrow 0

i \leftarrow 1

Όσο done=ψευδής και i \leq n επανάλαβε

 Αν table[i] = key τότε

 done \leftarrow αληθής

 position \leftarrow i

 αλλιώς

 i \leftarrow i + 1

 Τέλος_αν

Τέλος_Επανάληψης

Αποτελέσματα done, position

Τέλος ΣειριακήΑναζήτηση

Αν θα θέλαμε να αναζητήσουμε την τιμή 29.1 στο πίνακα θ των θερμοκρασιών, τότε ο αλγόριθμος θα ήταν ο εξής:

```

done ← ψευδής
position ← 0
i ← 1
Όσο done=ψευδής και i<=7 επανάλαβε
    Αν θ[i] = 29.1 τότε
        done ← αληθής
        position ← i
    αλλιώς
        i ← i + 1
Τέλος_αν
Τέλος_Επανάληψης

```

	1	2	3	4	5	6	7
θ:	28.3	27.4	29.1	28.1	26.3	25.8	26.7

Επανάληψεις	Αποτελέσματα
1η επανάληψη, i=1: done ← ψευδής position ← 0 i ← 1 Όσο done=ψευδής και 1<=7 επανάλαβε Αν θ[1] = 29.1 τότε done ← αληθής position ← 1 αλλιώς i ← 1 + 1 Τέλος_αν Τέλος_Επανάληψης	Γίνεται έλεγχος αν το θ[1] είναι 29.1. Δεν είναι, οπότε εκτελείται η “Αλλιώς” και αυξάνει το i κατά 1. Οπότε το i=2
2η επανάληψη, i=2: done ← ψευδής position ← 0 i ← 1 Όσο done=ψευδής και 2<=7 επανάλαβε Αν θ[2] = 29.1 τότε done ← αληθής position ← 2 αλλιώς i ← 2 + 1 Τέλος_αν	Γίνεται έλεγχος αν το θ[2] είναι 29.1. Δεν είναι, οπότε εκτελείται η “Αλλιώς” και αυξάνει το i κατά 1. Οπότε το i=3

Τέλος_Επανάληψης	
3η επανάληψη, i=3: done ← ψευδής position ← 0 i ← 1 Όσο done=ψευδής και 3<=7 επανάλαιβε Αν θ[3] = 29.1 τότε done ← αληθής position ← 3 αλλιώς i ← 3 + 1 Τέλος_αν Τέλος_Επανάληψης	Γίνεται έλεγχος αν το θ[3] είναι 29.1. Είναι 29.1, οπότε η μεταβλητή done γίνεται 'αληθής', και η μεταβλητή position παίρνει την τιμή 3, που είναι η θέση του πίνακα που περιέχει την τιμή 29.1. Στην επόμενη επανάληψη η συνθήκη done=ψευδής παύει να ισχύει αφού πλέον η done είναι 'αληθής' (δηλαδή η μεταβλητή done παίζει το ρόλο διακόπτη)

Αυτός που περιγράψαμε ήταν ο τρόπος λειτουργίας της σειριακής αναζήτησης στη περίπτωση που υπάρχει μέσα στον πίνακα η τιμή που αναζητάμε. Αν δεν υπάρχει η τιμή, τότε μετά τον έλεγχο και του τελευταίου στοιχείου του πίνακα θα σταματήσει να ισχύει η συνθήκη $i \leq n$ και θα σταματήσει η επανάληψη. Γι' αυτό και η συνθήκη της επανάληψης έχει δυο περιορισμούς (done=ψευδής και $i \leq n$). Ο ένας (done=ψευδής) ισχύει για όσο διάστημα δεν έχει βρεθεί η τιμή που αναζητάμε μέσα στον πίνακα και ο άλλος ($i \leq n$) ισχύει μέχρι να επιλεγεί και το τελευταίο στοιχείο του πίνακα για σύγκριση.

Το μεγάλο πλεονέκτημα της Σειριακής Αναζήτησης είναι το γεγονός ότι δεν απαιτείται τα δεδομένα μέσα στον πίνακα να είναι ταξινομημένα, εν' αντιθέσει με άλλες μεθόδους αναζήτησης όπου η ταξινόμηση των στοιχείων είναι απαραίτητη προϋπόθεση για να δουλέψει η αναζήτηση. Επίσης να τονίσουμε ότι ο αλγόριθμος αυτός της Σειριακής Αναζήτησης ισχύει μόνο για την περίπτωση που το στοιχείο που αναζητάμε υπάρχει **μόνο μια φορά** μέσα στον πίνακα. Στη περίπτωση που το στοιχείο που αναζητάμε υπάρχει περισσότερες από μια φορές, τότε πρέπει να παραλείψουμε εντελώς τη μεταβλητή done, τόσο μέσα στην 'Αν' αλλά και στην συνθήκη της 'Όσο'.

ΑΝΑΖΗΤΗΣΗ ΧΩΡΙΣ ΤΗ ΧΡΗΣΗ ΔΙΑΚΟΠΤΗ

Η μεταβλητή done παίζει το ρόλο του διακόπτη της επανάληψης. Όταν βρεθεί δηλαδή μέσα στον πίνακα το στοιχείο που ψάχνουμε, η μεταβλητή done εξασφαλίζει την απότομη διακοπή της επανάληψης και συνεπώς της αναζήτησης. Καταλαβαίνουμε λοιπός ότι η κλασική μορφή της Σειριακής Αναζήτησης δε μπορεί να λειτουργήσει όταν το στοιχείο το οποίο ψάχνουμε υπάρχει μέσα στον πίνακα περισσότερες από μια φορές. Οπότε σε τέτοιες περιπτώσεις πρέπει να τροποποιήσουμε τη Σειριακή Αναζήτηση κατάλληλα, έτσι ώστε να συνεχίζεται μέχρι το τέλος του πίνακα, προκειμένου να βρεθούν και οι υπόλοιπες θέσεις του πίνακα στις οποίες βρίσκεται το στοιχείο που ψάχνουμε. Πρέπει δηλαδή να αφαιρεθεί ο διακόπτης από τη Σειριακή Αναζήτηση. Οπότε ο αλγόριθμος τροποποιημένος για τέτοιες περιπτώσεις, γίνεται ως εξής:

position ← 0

```

i ← 1
Όσο i ≤ 7 επανάλαβε
    Αν θ[i] = 29.1 τότε
        position ← i
    Τέλος_αν
    i ← i + 1
Τέλος_Επανάληψης

```

ΠΡΟΣΟΧΗ: στις περιπτώσεις που δεν χρησιμοποιούμε διακόπτη στην Σειριακή Αναζήτηση ο μετρητής μας ($i \leftarrow i + 1$) πρέπει να μετακινηθεί εκτός της ΑΝ, διαφορετικά δε θα μπορεί να αλλάξει τιμή όταν βρεθεί την πρώτη φορά το στοιχείο που ψάχνουμε.

ΑΝΑΖΗΤΗΣΗ ΜΕ ΕΜΦΑΝΙΣΗ ΜΗΝΥΜΑΤΟΣ ΜΟΝΟ ΜΙΑ ΦΟΡΑ

Πολλές φορές ζητείται στις ασκήσεις στην περίπτωση που δεν βρεθεί στον πίνακα το στοιχείο που ψάχνουμε να βρούμε, να εμφανιστεί κάποιο αντίστοιχο μήνυμα. Σε τέτοιες περιπτώσεις χρησιμοποιούμε τη μεταβλητή done εκτός επανάληψης προκειμένου να ελέγξουμε αν βρέθηκε κάποιο στοιχείο που ψάχνουμε. Αν είμαστε στη περίπτωση όπου δεν χρησιμοποιούμε διακόπτη στη Σειριακή Αναζήτηση, τότε δημιουργούμε εμείς ένα διακόπτη (founded) και τον χρησιμοποιούμε ΜΟΝΟ για ελέγξουμε αν βρέθηκε το στοιχείο και ΟΧΙ για να διακόψουμε την επανάληψη.

Εμφάνιση μηνύματος χρησιμοποιώντας τον υπάρχων διακόπτη της επανάληψης	Εμφάνιση μηνύματος δημιουργώντας καινούργιο διακόπτη αποκλειστικά για το μήνυμα
<pre> done ← ψευδής position ← 0 i ← 1 Όσο done=ψευδής και i ≤ 7 επανάλαβε Αν θ[i] = 29.1 τότε done ← αληθής position ← i αλλιώς i ← i + 1 Τέλος_αν Τέλος_Επανάληψης Αν done = ΨΕΥΔΗΣ τότε Εμφάνισε 'Το στοιχείο δεν βρέθηκε' Τέλος_αν </pre>	<pre> position ← 0 i ← 1 founded ← ΨΕΥΔΗΣ Όσο i ≤ 7 επανάλαβε Αν θ[i] = 29.1 τότε position ← i founded ← ΑΛΗΘΗΣ Τέλος_αν i ← i + 1 Τέλος_Επανάληψης Αν founded = ΨΕΥΔΗΣ τότε Εμφάνισε 'Το στοιχείο δεν βρέθηκε' Τέλος_αν </pre>

Αυτή η τεχνική μπορεί να χρησιμοποιηθεί και για τις ακριβώς ανάποδες περιπτώσεις όπου το μήνυμα θα εμφανίζεται ΜΟΝΟ ΜΙΑ ΦΟΡΑ και όταν βρεθεί το στοιχείο που ψάχνουμε (δηλαδή να ελέγξουμε αν done=ΑΛΗΘΗΣ στην πρώτη περίπτωση και founded = ΑΛΗΘΗΣ στη δεύτερη περίπτωση).

ΤΑΞΙΝΟΜΗΣΗ

Μια άλλη δυνατότητα των πινάκων είναι η Ταξινόμηση. Με τον όρο Ταξινόμηση εννοούμε την τοποθέτηση των στοιχείων ενός πίνακα σε μία σειρά, είτε αύξουσα είτε φθίνουσα. Η ταξινόμηση μπορεί να φανεί πολύ χρήσιμη σε αρκετές περιπτώσεις πινάκων. Για παράδειγμα, στην εφαρμογή της δυαδικής αναζήτησης για την εύρεση μια τιμής, στον υπολογισμό της μέσης τιμής ενός πίνακα συνεχόμενων τιμών απλά και μόνο εμφανίζοντας την τιμή που έχει η μεσαία θέση του πίνακα. π.χ.

f:

1	2	3	4	5	6	7
---	---	---	---	---	---	---

↑
Μέση
Τιμή

Η πιο γνωστή μέθοδος ταξινόμησης, αλλά όχι και η πιο αποτελεσματική, είναι η μέθοδος της Φυσσαλίδας (Bubblesort).

Αλγόριθμος Φυσσαλίδα

//Δεδομένα table, n//

Για i από 2 μέχρι n

 Για j από n μέχρι i με_βήμα -1

 Αν table[j-1] > table[j] τότε

 Αντιμετάθεσε table[j-1], table[j]

 Τέλος_αν

 Τέλος_Επανάληψης

Τέλος_Επανάληψης

//Αποτελέσματα table//

Τέλος Φυσσαλίδα

Με τον όρο “Αντιμετάθεσε”, εννοούμε την αλλαγή των τιμών μεταξύ τους. Δηλαδή η τιμή της μιας μεταβλητής θα μεταφερθεί στην άλλη. Αυτό για να γίνει προγραμματιστικά χρειάζεται μια ενδιάμεση τρίτη μεταβλητή, προκειμένου να αποθηκεύσει προσωρινά μια από τις 2 τιμές. Δηλαδή η αντιμετάθεση γίνεται με τον εξής τρόπο:

temp ← table[j-1]

table[j-1] ← table[j]

table[j] ← temp

Η μεταβλητή temp παίζει το ρόλο της ενδιάμεσης μεταβλητής.

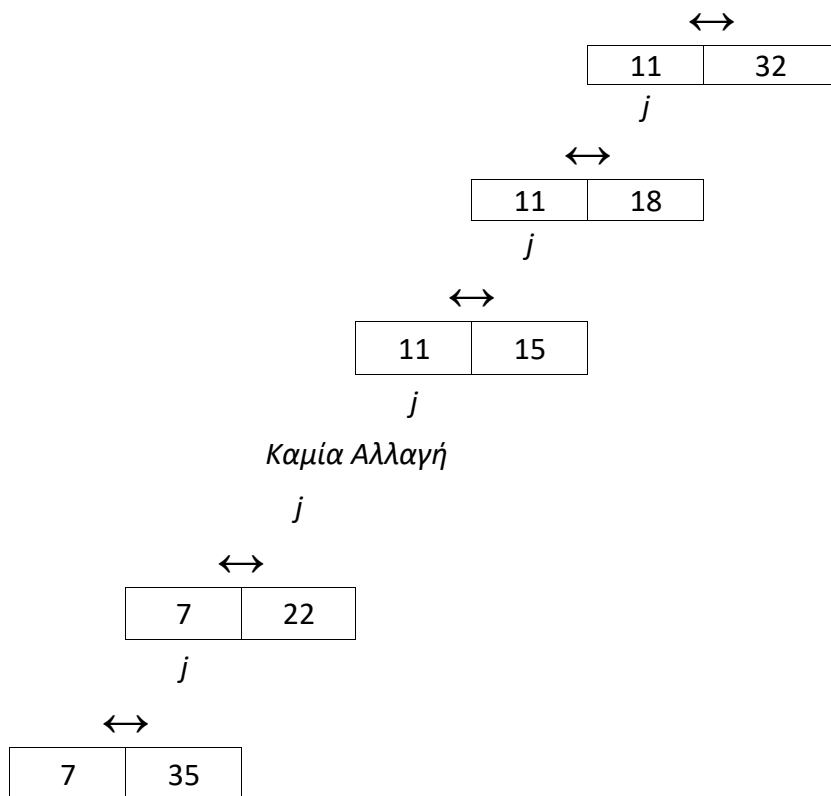
Ας υποθέσουμε ότι έχουμε τον πίνακα x στον οποίο θέλουμε να εφαρμόσουμε τη μέθοδο φυσσαλίδα για να τον ταξινομήσουμε. Τότε τα βήματα της ταξινόμησης θα ήταν τα ακόλουθα.

x:

35	22	7	15	18	32	11
----	----	---	----	----	----	----

1η επανάληψης:

<i>i</i>							<i>j</i>
35	22	7	15	18	32	11	

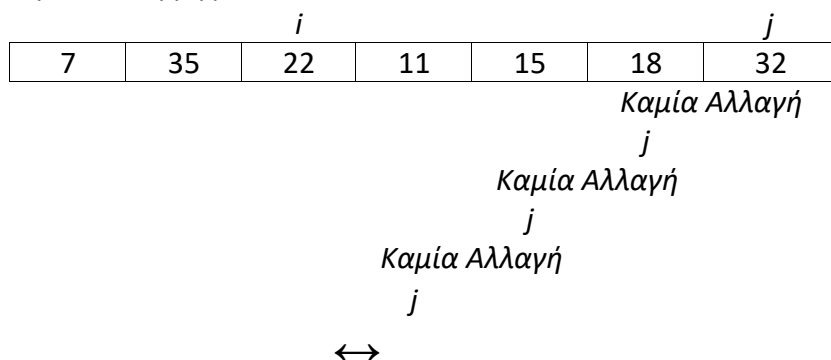


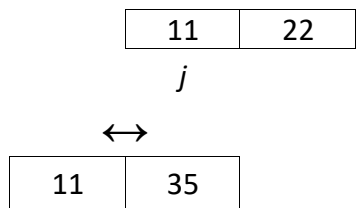
Η λογική του αλγορίθμου είναι σε κάθε προσπέλαση όλου του πίνακα να βρίσκεται το μικρότερο στοιχείο και να τοποθετείται στην αριστερότερη θέση. Στην επόμενη επανάληψη να βρίσκεται το αμέσως μικρότερο στοιχείο και να τοποθετείται στην αμέσως αριστερότερη θέση κλπ.

Όπως βλέπουμε χρησιμοποιούνται δυο μετρητές (i και j). Ο ένας μετρητής (i) μένει σταθερός κατά τη διάρκεια της επανάληψης και ο άλλος μετρητής αρχίζει κάθε φορά από το τέλος του πίνακα και κατεβαίνει μέχρι τη θέση του πρώτου μετρητή. Σε κάθε αλλαγή του μετρητή j γίνεται έλεγχος αν το εκάστοτε στοιχείο είναι μικρότερο από το προηγούμενο ($table[j-1] > table[j]$) και αν είναι, τότε γίνεται αντιμετάθεση των δυο στοιχείο. Αυτή η διαδικασία σύγκρισης και αντιμετάθεσης συνεχίζεται μέχρι το j να φτάσει τη τιμή του i, με αποτέλεσμα στο τέλος της επανάληψης το μικρότερο στοιχείο να έχει μεταφερθεί στη θέση i-1. Οπότε ο πίνακας μετά την πρώτη επανάληψη γίνεται:

7	35	22	11	15	18	32
---	----	----	----	----	----	----

2η επανάληψη:



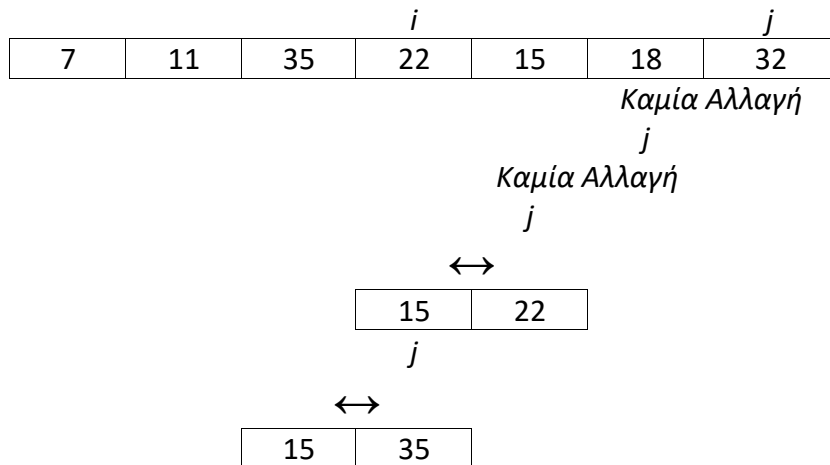


Άρα ο πίνακας στο τέλος της 2ης επανάληψης θα είναι:

7	11	35	22	15	18	32
---	----	----	----	----	----	----

Παρατηρούμε ότι στις δυο πρώτες επαναλήψεις του αλγορίθμου τα 2 μικρότερα στοιχεία έχουν μεταφερθεί στην 1η και 2η θέση. Με παρόμοιο τρόπο και μέχρι το τέλος του αλγορίθμου θα συνεχιστούν και οι υπόλοιπες επαναλήψεις και θα τοποθετηθούν τα στοιχεία στις ανάλογες θέσεις.

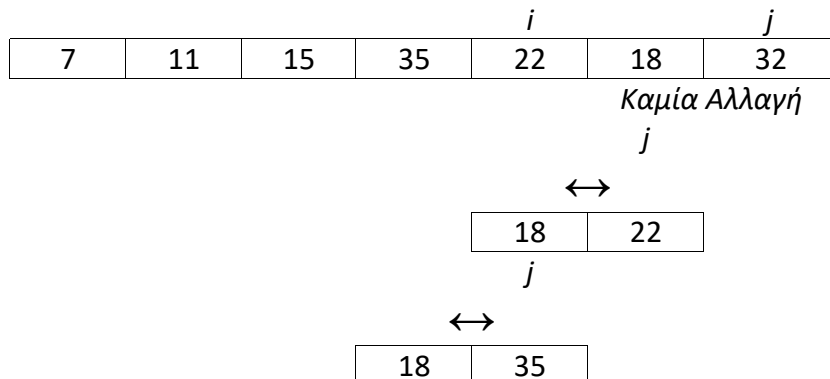
3η επανάληψη:



Ο πίνακας μετά την 3η επανάληψη θα είναι:

7	11	15	35	22	18	32
---	----	----	----	----	----	----

4η επανάληψη:



Ο πίνακας μετά την 4η επανάληψη θα είναι:

7	11	15	18	35	22	32
---	----	----	----	----	----	----

5η επανάληψη:

					<i>i</i>	<i>j</i>
7	11	15	18	35	22	32

Καμία Αλλαγή

j



22	35
----	----

6η επανάληψη:

					<i>i</i>	<i>j</i>
7	11	15	18	22	35	32



32	35
----	----

ΑΣΚΗΣΕΙΣ

1. Να γίνει αλγόριθμος που αποθηκεύει σε ένα πίνακα A τα αποτελέσματα της προπέδια του 7. Στη συνέχεια να εμφανίζει τα στοιχεία του πίνακα A.

2. Να γίνει αλγόριθμος που υπολογίζει το άθροισμα των στοιχείων των περιττών θέσεων και το άθροισμα των στοιχείων των άρτιων θέσεων του πίνακα που ακολουθεί

A:

7	9	15	3	19	20	8	4	11	17	14	6
---	---	----	---	----	----	---	---	----	----	----	---

3. Να γίνει αλγόριθμος που υπολογίζει το υπόλοιπο του κάθε στοιχείου του πίνακα με τη θέση του, και το αποθηκεύει σε έναν άλλο πίνακα στην αντίστοιχη θέση

A:

25	68	89	45	74	22	50	82	48
----	----	----	----	----	----	----	----	----

4. Να γίνει αλγόριθμος που υπολογίζει και εμφανίζει τον μέσο όρο των στοιχείων του πίνακα που ακολουθεί

A:

12	17	5	34	22	71	6	15	19	35	11	61
----	----	---	----	----	----	---	----	----	----	----	----

5. Να γίνει αλγόριθμος που προσθέτει τα στοιχεία των δυο πινάκων που ακολουθούν και τα αποθηκεύει σε ένα πίνακα Γ.

A:

2	4	6	8	10	12	14	16	18	20
---	---	---	---	----	----	----	----	----	----

B:

1	3	5	7	9	11	13	15	17	19
---	---	---	---	---	----	----	----	----	----

6. Να γίνει αλγόριθμος ο οποίος αποθηκεύει και εμφανίζει τα στοιχεία του πίνακα X σε ένα πίνακα Y αντίστροφα

X:

4	8	2	6	3	7	12	9
---	---	---	---	---	---	----	---

7. Να γίνει αλγόριθμος ο οποίος ζητάει από τον χρήστη να δώσει 10 θερμοκρασίες τις οποίες αποθηκεύει σε ένα πίνακα, και στη συνέχεια υπολογίζει τη μέγιστη και την ελάχιστη τιμή αυτών των θερμοκρασιών.

8. Να γίνει αλγόριθμος που υπολογίζει πόσες φορές υπάρχει ο αριθμός 7 στον πίνακα που ακολουθεί

Π:

4	9	7	2	11	8	7	5	3	12	7
---	---	---	---	----	---	---	---	---	----	---

9. Να γίνει αλγόριθμος ο οποίος αποθηκεύει τους άρτιους αριθμούς από το 100 μέχρι το 1 σε ένα πίνακα A και τους περιττούς σε ένα πίνακα B. Στη συνέχεια να γίνει ταξινόμηση των δυο πινάκων A και B. Έπειτα να βρεθεί σε ποια θέση του πίνακα A βρίσκεται ο αριθμός 18, και σε ποια θέση του πίνακα B ο αριθμός 57.

10. Να γίνει αλγόριθμος ο οποίος αποθηκεύει σε ένα πίνακα 10 θέσεων, διάφορες λέξεις που δίνει ο χρήστης. Στη συνέχεια ζητάει από τον χρήστη να δώσει μια λέξη προς αναζήτηση και ψάχνει να εντοπίσει τη λέξη αυτή μέσα στο πίνακα. Αν η λέξη βρεθεί, να εμφανίζει το μήνυμα “Η λέξη υπάρχει”, αν δεν υπάρχει η λέξη να εμφανίζει το μήνυμα “Δεν βρέθηκε η λέξη”.

11. Ο πίνακας B που ακολουθεί περιέχει 8 αριθμούς και ο πίνακας A τους αντίστοιχους αριθμούς με τα ονόματά τους. Να γίνει αλγόριθμος που ταξινομεί τον πίνακα B μεταθέτοντας ταυτόχρονα και τα αντίστοιχα στοιχεία του πίνακα A (δηλαδή αν μετά την ταξινόμηση ο αριθμός 8 βρίσκεται στη δεύτερη θέση του πίνακα B, τότε και στη δεύτερη θέση του πίνακα A να υπάρχει το όνομα του αριθμού).

A:

εννέα	δώδεκα	έντεκα	οχτώ	τρία	δέκα	δεκαπέντε
-------	--------	--------	------	------	------	-----------

B:	9	12	11	8	3	10	15
----	---	----	----	---	---	----	----

12. Να γίνει αλγόριθμος που υπολογίζει και εμφανίζει το μισό των πρώτων έξι στοιχείων του πίνακα A, και το διπλάσιο των υπόλοιπων στοιχείων.

A:	14	18	6	20	22	15	4	7	13	21	12	23
----	----	----	---	----	----	----	---	---	----	----	----	----

ΔΙΣΔΙΑΣΤΑΤΟΙ ΠΙΝΑΚΕΣ

Πέρα από τους απλούς πίνακες υπάρχουν και οι δισδιάστατοι πίνακες. Ένας δισδιάστατος πίνακας είναι ένα σύνολο πολλών συνεχόμενων απλών (μονοδιάστατων) πινάκων ο ένας μετά τον άλλο. Π.χ. ο πίνακας θ που είδαμε πριν, είχε τις θερμοκρασίες μιας περιοχής για όλη την εβδομάδα (μια θερμοκρασία ανά μέρα). Αν θα θέλαμε ένα πίνακα με τις θερμοκρασίες ενός μηνός ανά βδομάδα, τότε θα χρειαζόμασταν έναν δισδιάστατο πίνακα που περιέχει τέσσερις απλούς πίνακες (αφού τέσσερις βδομάδες έχει ένας μήνας) τον ένα κάτω από τον άλλον. Δηλαδή ο δισδιάστατος πίνακας μας θα γινόταν :

θέση:	1	2	3	4	5	6	7
$\theta 1:$	28.3	27.4	29.1	28.1	26.3	25.8	26.7
$\theta 2:$	24.2	23.5	24.6	25	23.9	22	22.6
$\theta 3:$	20.5	21.3	21.7	22.3	23.1	22.9	23.5
$\theta 4:$	26.3	25.4	24.7	23.5	22.9	21.3	23.8

Όπως βλέπουμε ο δισδιάστατος πίνακας μας αποτελείται από τέσσερις μονοδιάστατους πίνακες καθένας από τους οποίους περιέχει τις θερμοκρασίες μιας εβδομάδας.

Δήλωση Δισδιάστατου Πίνακα

Για να δηλώσουμε ένα μονοδιάστατο πίνακα, αναφέρουμε πόσες θέσεις περιέχει ο πίνακας. Δηλαδή:

Ακέραιες: $x[15]$

Στη δήλωση του δισδιάστατου πίνακα εκτός από τις θέσεις του πίνακα, δηλώνουμε και πόσοι μονοδιάστατοι πίνακες υπάρχουν σε έναν δισδιάστατο πίνακα, ή αλλιώς πρέπει να δηλώσουμε πόσες γραμμές και πόσες στήλες έχει ένας δισδιάστατος πίνακας. Δηλαδή η δήλωση του δισδιάστατου πίνακα με τις θερμοκρασίες θα είναι:

Πραγματικές: $\theta[4,7]$

Προσπέλαση Δισδιάστατοι Πίνακα

	<i>j</i>	<i>j</i>	<i>j</i>	<i>j</i>	<i>j</i>	<i>j</i>	<i>j</i>
<i>i</i>	28.3	27.4	29.1	28.1	26.3	25.8	26.7
<i>i</i>	24.2	23.5	24.6	25	23.9	22	22.6
<i>i</i>	20.5	21.3	21.7	22.3	23.1	22.9	23.5
<i>i</i>	26.3	25.4	24.7	23.5	22.9	21.3	23.8

Όπως είδαμε στη προσπέλαση του μονοδιάστατου πίνακα, χρησιμοποιείται ένας μετρητής ο οποίος συμβολίζει τη θέση του πίνακα, και αλλάζει σταδιακά, προκειμένου να πάει στο επόμενο στοιχείο του πίνακα. Στον δισδιάστατο πίνακα, χρειάζονται δυο μετρητές, ο ένας (*j*) αλλάζει προκειμένου να μεταβούμε στην επόμενη στήλη του πίνακα και ο άλλος (*i*) αλλάζει προκειμένου να μεταβούμε στην επόμενη γραμμή του πίνακα. Δηλαδή αν θέλουμε να εμφανίσουμε το 3ο στοιχείο της 2ης γραμμής τότε αυτό συμβολίζεται ως $\theta[2,3]$.

Έτσι λοιπόν η καταχώρηση δεδομένων σε ένα δισδιάστατο πίνακα θα έχει την εξής μορφή:

ΓΙΑ...ΑΠΟ...ΜΕΧΡΙ	ΟΣΟ	ΜΕΧΡΙΣ_ΟΤΟΥ
Πρόγραμμα Α	Πρόγραμμα Α	Πρόγραμμα Α
Μεταβλητές	Μεταβλητές	Μεταβλητές
Ακέραιες: <i>i, j</i>	Ακέραιες: <i>i, j</i>	Ακέραιες: <i>i, j</i>
Πραγματικές: $\theta[4,7]$	Πραγματικές: $\theta[4,7]$	Πραγματικές: $\theta[4,7]$
Αρχή	Αρχή	Αρχή
Για <i>i</i> Απο 1 Μέχρι 4	$i \leftarrow 1$	$i \leftarrow 1$
Για <i>j</i> Απο 1 Μέχρι 7	Όσο $i \leq 4$ Επανάλαβε	Αρχή_Επανάληψης
Γράψε 'Δώσε Θερμοκρασία:'	$j \leftarrow 1$	$j \leftarrow 1$
Διάβασε $\theta[i,j]$	Όσο ≤ 7 Επανάλαβε	Αρχή_Επανάληψης
Τέλος_Επανάληψης	Γράψε 'Δώσε Θερμοκρασία:'	Γράψε 'Δώσε Θερμοκρασία:'
Τέλος_Επανάληψης	Διάβασε $\theta[i,j]$	Διάβασε $\theta[i,j]$
Τέλος_Προγράμματος	$j \leftarrow j + 1$	$j \leftarrow j + 1$
	Τέλος_Επανάληψης	Μέχρις_Ότου $j > 7$
	$i \leftarrow i + 1$	$i \leftarrow i + 1$
	Τέλος_Επανάληψης	Μέχρις_Ότου $i > 4$
	Τέλος_Προγράμματος	Τέλος_Προγράμματος

Όπως βλέπουμε σε σχέση με τους μονοδιάστατους πίνακες στους δισδιάστατους προστίθεται μια επιπλέον επανάληψη. Με αυτή την επιπλέον επανάληψη η σειρά προσπέλασης των στοιχείων ενός πίνακα θα είναι αυτή που φαίνεται στον πίνακα που ακολουθεί:

	j	j	j	j	j	j	j
i	1η προσπέλαση $\theta[1,1]$	2η προσπέλαση $\theta[1,2]$	3η προσπέλαση $\theta[1,3]$	4η προσπέλαση $\theta[1,4]$	5η προσπέλαση $\theta[1,5]$	6η προσπέλαση $\theta[1,6]$	7η προσπέλαση $\theta[1,7]$
i	8η προσπέλαση $\theta[2,1]$	9η προσπέλαση $\theta[2,2]$	10η προσπέλαση $\theta[2,3]$	11η προσπέλαση $\theta[2,4]$	12η προσπέλαση $\theta[2,5]$	13η προσπέλαση $\theta[2,6]$	14η προσπέλαση: $\theta[2,7]$
i	15η προσπέλαση $\theta[3,1]$	16η προσπέλαση $\theta[3,2]$	17η προσπέλαση $\theta[3,3]$	18η προσπέλαση $\theta[3,4]$	19η προσπέλαση $\theta[3,5]$	20η προσπέλαση $\theta[3,6]$	21η προσπέλαση $\theta[3,7]$
i	22η προσπέλαση $\theta[4,1]$	23η προσπέλαση $\theta[4,2]$	24η προσπέλαση $\theta[4,3]$	25η προσπέλαση $\theta[4,4]$	26η προσπέλαση $\theta[4,5]$	27η προσπέλαση $\theta[4,6]$	28η προσπέλαση $\theta[4,7]$

Η ανάγνωση δεδομένων από ένα δισδιάστατο πίνακα, είναι ακριβώς ίδια με την καταχώρηση αλλάζοντας όμως τις εντολές μέσα στην δεύτερη εσωτερική επανάληψη με μία εντολή εμφάνισης.

ΥΠΟΛΟΓΙΣΜΟΙ ΣΕ ΔΙΣΔΙΑΣΤΑΤΟΥΣ ΠΙΝΑΚΕΣ

Οι υπολογισμοί που μπορούν να ζητηθούν από ένα δισδιάστατο πίνακα χωρίζονται σε τρεις κατηγορίες:

1. Στους υπολογισμούς με όλα τα στοιχεία του πίνακα
2. Στους υπολογισμούς ανά γραμμή
3. Στους υπολογισμούς ανά στήλη

1. Υπολογισμοί με όλα τα στοιχεία του πίνακα

Μέγιστο-Ελάχιστο Δισδιάστατου Πίνακα

Ο υπολογισμός του μέγιστου και του ελάχιστου ενός δισδιάστατου πίνακα, είναι ακριβώς ίδιο με αυτόν ενός μονοδιάστατου. Η μόνη διαφορά είναι ότι για τη προσπέλαση του πίνακα χρησιμοποιούμε τη διπλή επανάληψη που απαιτείται, αφού ο πίνακάς μας είναι δισδιάστατος. Συνεπώς αν θέλουμε να υπολογίζουμε την ελάχιστη και τη μέγιστη τιμή ενός πίνακα $\theta[10,4]$ τότε ο αλγόριθμος θα είναι (και με τις τρεις εντολές επανάληψης):

ΕΛΑΧΙΣΤΟ

ΓΙΑ...ΑΠΟ...ΜΕΧΡΙ	ΟΣΟ	ΜΕΧΡΙΣ_ΟΤΟΥ
$\min \leftarrow \theta[1,1]$ Για i Από 1 Μέχρι 10 Για j Από 1 Μέχρι 4 Αν $\theta[i,j] < \min$ τότε $\min \leftarrow \theta[i,j]$ Τέλος_αν Τέλος_Επανάληψης Τέλος_Επανάληψης	$\min \leftarrow \theta[1,1]$ $i \leftarrow 1$ Όσο $i \leq 10$ Επανάλαβε $j \leftarrow 1$ Όσο $j \leq 4$ Επανάλαβε Αν $\theta[i,j] < \min$ τότε $\min \leftarrow \theta[i,j]$ Τέλος_αν $j \leftarrow j + 1$ Τέλος_Επανάληψης $i \leftarrow i + 1$ Τέλος_Επανάληψης	$\min \leftarrow \theta[1,1]$ $i \leftarrow 1$ Αρχή_Επανάληψης $j \leftarrow 1$ Αρχή_Επανάληψης Αν $\theta[i,j] < \min$ τότε $\min \leftarrow \theta[i,j]$ Τέλος_αν $j \leftarrow j + 1$ Μέχρις_Ότου $j > 4$ $i \leftarrow i + 1$ Μέχρις_Ότου $i > 10$

ΜΕΓΙΣΤΟ

ΓΙΑ...ΑΠΟ...ΜΕΧΡΙ	ΟΣΟ	ΜΕΧΡΙΣ_ΟΤΟΥ
$\max \leftarrow \theta[1,1]$ Για i Από 1 Μέχρι 10 Για j Από 1 Μέχρι 4 Αν $\theta[i,j] > \max$ τότε $\max \leftarrow \theta[i,j]$ Τέλος_αν Τέλος_Επανάληψης Τέλος_Επανάληψης	$\max \leftarrow \theta[1,1]$ $i \leftarrow 1$ Όσο $i \leq 10$ Επανάλαβε $j \leftarrow 1$ Όσο $j \leq 4$ Επανάλαβε Αν $\theta[i,j] > \max$ τότε $\max \leftarrow \theta[i,j]$ Τέλος_αν $j \leftarrow j + 1$ Τέλος_Επανάληψης $i \leftarrow i + 1$ Τέλος_Επανάληψης	$\max \leftarrow \theta[1,1]$ $i \leftarrow 1$ Αρχή_Επανάληψης $j \leftarrow 1$ Αρχή_Επανάληψης Αν $\theta[i,j] > \max$ τότε $\max \leftarrow \theta[i,j]$ Τέλος_αν $j \leftarrow j + 1$ Μέχρις_Ότου $j > 4$ $i \leftarrow i + 1$ Μέχρις_Ότου $i > 10$

Μέσος - Όρος Δισδιάστατου Πίνακα

Και ο μέσος όρος ενός δισδιάστατου πίνακα υπολογίζεται με τον ίδιο τρόπο όπως στο μονοδιάστατο, με τη διαφορά ότι στο δισδιάστατο χρησιμοποιείται διπλή επανάληψης για την προσπέλαση του πίνακα. Δηλαδή οι τρεις αλγόριθμοι (ένας για κάθε επανάληψη) για τον υπολογισμό του μέσου όρου ενός δισδιάστατου πίνακα $\theta[10,4]$ είναι:

ΓΙΑ...ΑΠΟ...ΜΕΧΡΙ	ΟΣΟ	ΜΕΧΡΙΣ_ΟΤΟΥ
$sum \leftarrow 0$ Για i Απο 1 Μέχρι 10 Για j Από 1 Μέχρι 4 $sum \leftarrow sum + \theta[i,j]$ Τέλος_Επανάληψης Τέλος_Επανάληψης $avg \leftarrow sum/40$	$sum \leftarrow 0$ $i \leftarrow 1$ Όσο $i \leq 10$ Επανάλαβε $j \leftarrow 1$ Όσο $j \leq 4$ Επανάλαβε $sum \leftarrow sum + \theta[i,j]$ $j \leftarrow j + 1$ Τέλος_επανάληψης $i \leftarrow i + 1$ Τέλος_Επανάληψης $avg \leftarrow sum/40$	$sum \leftarrow 0$ $i \leftarrow 1$ Αρχή_Επανάληψης $j \leftarrow 1$ Αρχή_Επανάληψης $sum \leftarrow sum + \theta[i,j]$ $j \leftarrow j + 1$ Μέχρις_Ότου $j > 4$ $i \leftarrow i + 1$ Μέχρις_Ότου $i > 10$ $avg \leftarrow sum/40$

Αναζήτηση Δισδιάστατου Πίνακα

Για την αναζήτηση σε όλα τα στοιχεία ενός δισδιάστατου πίνακα, ισχύει ότι στην αναζήτηση σε ένα μονοδιάστατο πίνακα, με τη διαφορά ότι στην αναζήτηση σε ένα δισδιάστατο, χρησιμοποιείται μια επιπλέον επανάληψη, προκειμένου να γίνεται προσπέλαση όλων των γραμμών. Δηλαδή η αναζήτηση σε όλα τα στοιχεία ενός δισδιάστατου πίνακα είναι:

Όσο	Μέχρις_Ότου	Για
$i \leftarrow 1$ Όσο $i \leq n$ επανάλαβε $j \leftarrow 1$ Όσο $j \leq m$ επανάλαβε Αν Πίνακας[i,j] = ΣΑ τότε Γράψε Το στοιχείο βρέθηκε' Τέλος_Αν $j \leftarrow j + 1$ Τέλος_Επανάληψης $i \leftarrow i + 1$ Τέλος_Επανάληψης	$i \leftarrow 1$ Αρχή_Επανάληψης $j \leftarrow 1$ Αρχή_επανάληψης Αν Πίνακας[i,j] = ΣΑ τότε Γράψε Το στοιχείο βρέθηκε' Τέλος_Αν $j \leftarrow j + 1$ Μέχρις_Ότου $j > m$ $i \leftarrow i + 1$ Μέχρις_Ότου $i > n$	Για i απο 1 μέχρι n Για j απο 1 μέχρι m Αν Πίνακας[i,j] = ΣΑ τότε Γράψε 'Το στοιχείο βρέθηκε' Τέλος_Αν Τέλος_Επανάληψης Τέλος_Επανάληψης

Όπου 'Πίνακας[i,j]' είναι ο πίνακας στον οποίο αναζητούμε το στοιχείο που ψάχνουμε, 'n' είναι το σύνολο των γραμμών του πίνακα, 'm' το σύνολο των στηλών του πίνακα και 'ΣΑ' αυτό που θέλουμε να βρούμε.

2. Υπολογισμοί ανά γραμμή

Σε πολλές ασκήσεις θα χρειαστεί να κάνουμε κάποιους υπολογισμούς ανά γραμμή. Δηλαδή να υπολογίσουμε κάποια τιμή από τη πρώτη γραμμή του δισδιάστατου πίνακα, να την εμφανίσουμε και στη συνέχεια να μεταβούμε στη δεύτερη γραμμή κάνοντας τον ίδιο υπολογισμό, κ.ο.κ. μέχρι τη τελευταία γραμμή του δισδιάστατου πίνακα. Ένα τέτοιο παράδειγμα είναι ο υπολογισμός του μέσου όρου ανά γραμμή ενός δισδιάστατου πίνακα.

Όπως έχουμε πει για την προσπέλαση ενός δισδιάστατου πίνακα, χρειάζεται να χρησιμοποιήσουμε δυο εντολές επανάληψης (η μια μέσα στην άλλη). Η πρώτη επανάληψη μας εξασφαλίζει την προσπέλαση του πίνακα κατά γραμμή. Δηλαδή αν ο πίνακας που θέλουμε να προσπελάσουμε έχει 10 γραμμές, τότε η πρώτη επανάληψη θα πρέπει να είναι:

Για i από 1 μέχρι 10

 Για j από 1 μέχρι 5

 <Υπόλοιπος κώδικας>

 Τέλος_Επανάληψης

Τέλος_επανάληψης

Η δεύτερη επανάληψη στη προσπέλαση του δισδιάστατου πίνακα, μας εξασφαλίζει την προσπέλαση του πίνακα κατά στήλη. Δηλαδή αν ο πίνακας που θέλουμε να προσπελάσουμε έχει 10 γραμμές και 5 στήλες, τότε ο αλγόριθμος προσπέλασης του πίνακα θα ήταν:

Για i από 1 μέχρι 10

Για j από 1 μέχρι 5

 <Υπόλοιπος κώδικας>

Τέλος_Επανάληψης

Τέλος_επανάληψης

Έτσι λοιπόν όταν θέλουμε να κάνουμε υπολογισμούς ανά γραμμή, όλοι οι υπολογισμοί θα πρέπει να γίνουν μέσα στη δεύτερη επανάληψη. Οι αρχικοποιήσεις τιμών πρέπει να γίνουν μετά τη πρώτη επανάληψη και πριν τη δεύτερη, και η εμφάνιση των αποτελεσμάτων ανά γραμμή, μετά τη δεύτερη επανάληψη και πριν την πρώτη. Δηλαδή η δομή του αλγορίθμου για τον υπολογισμό ανά γραμμή, είναι η ακόλουθη:

Για i από 1 μέχρι n

 <Αρχικοποιήσεις τιμών>

 Για j από 1 μέχρι m

 <Υπόλοιπος κώδικας>

 Τέλος_Επανάληψης

 <Εμφάνιση υπολογισμών ανά γραμμή>

Τέλος_επανάληψης

όπου n είναι ο αριθμός των γραμμών του πίνακα και m ο αριθμός των στηλών του πίνακα.

Έτσι λοιπόν αν θελήσουμε να υπολογίσουμε το άθροισμα ανά γραμμή σε ένα δισδιάστατο πίνακα A[4,7], τότε ο αλγόριθμος θα ήταν:

1. Για i από 1 μέχρι 4
2. sum←--0
3. Για j από 1 μέχρι 7
4. sum←--sum+A[i,j]
5. Τέλος_Επανάληψης
6. Γράψε 'Το άθροισμα της', i, 'ης γραμμής είναι:',sum
7. Τέλος_επανάληψης

Με αντίστοιχο τρόπο υπολογίζονται ανά γραμμή, και όλες οι άλλες τιμές που μάθαμε (μέσος όρος, count, max, min) καθώς επίσης και η αναζήτηση ανά γραμμή.

3. Υπολογισμοί ανά στήλη

Στους υπολογισμούς ανά στήλη, η δομή του αλγορίθμου μας είναι ακριβώς ίδια με αυτή των υπολογισμών ανά γραμμή, με μόνες διαφορές ότι:

A) η πρώτη επανάληψη πρέπει αυτή τη φορά να αναφέρεται στις στήλες,

B) η δεύτερη εσωτερική επανάληψη να αναφέρεται στις γραμμές

Δηλαδή να έχουμε τον προηγούμενο δισδιάστατο πίνακα A[4,7] και θέλουμε να υπολογίσουμε το άθροισμα ανά στήλη αυτή τη φορά, τότε ο αλγόριθμος θα είναι:

1. Για i από 1 μέχρι 7
2. sum←-0
3. Για j από 1 μέχρι 4
4. sum←-sum+A[i,j]
5. Τέλος_Επανάληψης
6. Γράψε 'Το άθροισμα της', i,'ης στήλης είναι:',sum
7. Τέλος_επανάληψης

A) Η πρώτη επανάληψη (γραμμή 1) αυτή τη φορά φτάνει μέχρι το 7, αφού όπως είπαμε η πρώτη επανάληψη αυτή τη φορά αναφέρεται στη στήλη και οι στήλες του πίνακα μας είναι 7.

B) Η δεύτερη επανάληψη (γραμμή 3) αυτή τη φορά φτάνει μέχρι το 4, αφού η δεύτερη επανάληψη θα πρέπει να αναφέρεται στη γραμμή και ο πίνακας A έχει 4 γραμμές.

Με αντίστοιχο τρόπο υπολογίζονται ανά στήλη, και όλες οι άλλες τιμές που μάθαμε (μέσος όρος, count, max, min) καθώς επίσης και η αναζήτηση ανά στήλη.

ΑΣΚΗΣΕΙΣ

1. Να γίνει αλγόριθμος που δημιουργεί ένα δισδιάστατο πίνακα $X[8,8]$ και στη συνέχεια υπολογίζει και εμφανίζει το άθροισμα των 2 κύριων διαγωνίων του πίνακα. (P_70)
2. Να γίνει αλγόριθμος που αποθηκεύει τις μηνιαίες κρατήσεις τεσσάρων ξενοδοχείων για μία σεζόν. Στη συνέχεια να εμφανίζεται το σύνολο των κρατήσεων ανά μήνα. (Θεωρούμε ότι μια σεζόν ξεκινάει τον Μάιο και ολοκληρώνεται τον Οκτώβριο)
3. Να γραφεί αλγόριθμος ο οποίος να διαβάζει ένα πίνακα ακέραιων αριθμών A, δύο διαστάσεων 3×4 και να υπολογίζει και να τυπώνει το μέσο όρο, το άθροισμα και το γινόμενο των στοιχείων του πίνακα A.
4. Να γίνει αλγόριθμος που υπολογίζει και εμφανίζει το άθροισμα των στοιχείων ανά στήλη και ανά γραμμή του παρακάτω πίνακα (P_69)

5	9	12	4
6	2	8	10
3	15	1	7

5. Σε έναν πίνακα καταχωρούνται οι εισπράξεις μιας αλυσίδας 4 καταστημάτων τα τελευταία 3 χρόνια. Να γίνει αλγόριθμος που να καταχωρεί τα παραπάνω στοιχεία σε δισδιάστατο πίνακα και να εμφανίζει τα σύνολα του κάθε καταστήματος για αυτά τα 3 χρόνια. (P_54)
6. Να γραφεί πρόγραμμα το οποίο θα δημιουργεί τον ακόλουθο δισδιάστατο πίνακα:

1	10	10	10	10
20	1	10	10	10
20	20	1	10	10
20	20	20	1	10
20	20	20	20	1

(P_63)

7. Να γραφεί πρόγραμμα το οποίο θα δημιουργεί τον ακόλουθο δισδιάστατο πίνακα:

α	β	α	β
β	α	β	α
α	β	α	β
β	α	β	α

(P_64)

8. Τετραγωνικός πίνακας 5×5 περιέχει τον αριθμό 25. Να γράψετε αλγόριθμο που να βρίσκει και εμφανίζει την θέση του πίνακα (γραμμή, στήλη) στην οποία βρίσκεται το 25. Η διαδικασία να σταματά όταν βρεθεί για πρώτη φορά ο αριθμός
9. Δίνεται το παρακάτω τμήμα αλγορίθμου:

$N \leftarrow 5$

Για i από 1 μέχρι N

 Για k από 1 μέχρι N

 Αν $i=k$ Η $i+k = N + 1$ τότε

$A[i,k] \leftarrow 1$

 Αλλιώς

$A[i,k] \leftarrow 0$

 Τέλος_αν

Τέλος_επανάληψης

Τέλος_επανάληψης

Να σχεδιάσετε τον πίνακα A συμπληρωμένο, όπως είναι μετά την εκτέλεση αυτού του τμήματος αλγορίθμου.

10. Να γραφεί πρόγραμμα που θα διαβάζει έναν πίνακα 100×100 και θα υπολογίζει το πλήθος των άρτιων και το πλήθος των περιττών στοιχείων ανά γραμμή.(P_66)
11. Με δεδομένο πίνακα 10×20 αριθμών, να γραφεί αλγόριθμος που θα υπολογίζει και εμφανίζει τα ακόλουθα μεγέθη:
5. Το άθροισμα των στοιχείων του πίνακα.
 6. Το άθροισμα κάθε στήλης του πίνακα.
 7. Το ελάχιστο κάθε γραμμής του πίνακα.
 8. Το μέγιστο στοιχείο του πίνακα και οι θέσεις στον πίνακα που υπάρχει η μέγιστη αυτή τιμή.(P_68)

ΥΠΟΠΡΟΓΡΑΜΜΑΤΑ

Τα υποπρογράμματα είναι ένας διαφορετικός τρόπος προγραμματισμού από αυτόν που έχουμε δει μέχρι τώρα. Το σκεπτικό πάνω στο οποίο βασίζεται αυτό το είδος προγραμματισμού είναι ο διαχωρισμός ενός μεγάλου προγράμματος σε πολλά μικρότερα. Για να συμβεί όμως αυτό, θα πρέπει αντίστοιχα και ο χρήστης που γράφει το πρόγραμμα να αναλύσει την άσκηση σε πολλά μικρότερα κομμάτια. Δηλαδή αν έχουμε μια άσκηση όπου θα ζητείται από τον χρήστη να δώσει τις διαστάσεις ενός παραλληλογράμμου και το πρόγραμμα να υπολογίζει τη περίμετρό και το εμβαδό του, στον προγραμματισμό που μάθαμε μέχρι τώρα τα βήματα που θα ακολουθήσουμε είναι τα εξής:

Κυρίως Πρόγραμμα

1. Αρχικά θα ζητήσουμε από το χρήστη να δώσει τις τιμές
2. Θα τις αποθηκεύσουμε σε μεταβλητές
3. Θα υπολογίσουμε τη περίμετρο και το εμβαδό
4. Θα εμφανίσουμε τη περίμετρο και το εμβαδό

Όπως είπαμε στα υποπρογράμματα θα πρέπει να αναλύσουμε το πρόβλημα μας σε πολλά μικρότερα προβλήματα. Δηλαδή το παράδειγμα μας θα πρέπει να το διαχωρίσουμε σε μικρότερα υποπρογράμματα. Άρα, αντί για ένα πρόγραμμα που κάνει τα 4 βήματα που αναφέραμε, θα δημιουργήσουμε το κυρίως πρόγραμμα και 3 υποπρογράμματα που κάνουν τα εξής:

1. Κυρίως Πρόγραμμα: Ζητάμε από το χρήστη να δώσει τις μεταβλητές και τις αποθηκεύουμε σε μεταβλητές.
2. 1^ο υποπρόγραμμα: Υπολογίζουμε τη περίμετρο.
3. 2^ο υποπρόγραμμα: Υπολογίζουμε το εμβαδό.
4. 3^ο υποπρόγραμμα: Εμφανίζουμε τα αποτελέσματα.

Το κέρδος από τη μέθοδο των υποπρογραμμάτων είναι εμφανές σε μεγάλα προγράμματα, όπου γίνεται ένας σαφής διαχωρισμός του προγράμματος σε πολλά μικρότερα και δεν μπερδεύεται ο προγραμματιστής ανάμεσα σε ένα πλήθος εντολών. Επίσης ένα άλλο μεγάλο πλεονέκτημα των υποπρογραμμάτων, είναι η δυνατότητα χρησιμοποίησης τους πολλές φορές μέσα στο πρόγραμμα, γράφοντας μόνο μια φορά τον κώδικα (θα δούμε πρακτικά πως γίνεται), μικραίνοντας έτσι το μέγεθος του προγράμματος μας όταν χρειάζεται να επαναλάβουμε κομμάτια κώδικα.

Κύριο χαρακτηριστικό των υποπρογραμμάτων είναι ότι αποτελούνται από 2 κομμάτια. Το κομμάτι της δημιουργίας τους και το κομμάτι της κλήσης του (θα δούμε στη συνέχεια τι είναι αυτά τα δύο). Επίσης χωρίζονται σε 2 μεγάλες κατηγορίες, τις ΣΥΝΑΡΤΗΣΕΙΣ και τις ΔΙΑΔΙΚΑΣΙΕΣ, τις οποίες θα αναλύσουμε στη συνέχεια.

ΚΥΡΙΩΣ ΠΡΟΓΡΑΜΜΑ

Με τον όρο «Κυρίως πρόγραμμα» αναφερόμαστε στο πρόγραμμα που έχει το βασικό κομμάτι του κώδικά μας και το οποίο εκτελείται κατά την εκτέλεση του κώδικά μας. Είναι δηλαδή το κομμάτι του κώδικα που αρχίζει με την εντολή «ΠΡΟΓΡΑΜΜΑ <όνομα>» και τελειώνει με το «ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ» όταν πρόκειται για κώδικα σε ΓΛΩΣΣΑ ή το κομμάτι μεταξύ των λέξεων «ΑΛΓΟΡΙΘΜΟΣ <όνομα>» και «ΤΕΛΟΣ_ΑΛΓΟΡΙΘΜΟΥ» όταν πρόκειται για αλγόριθμο

ΠΡΟΓΡΑΜΜΑ παράδειγμα

<Δήλωση μεταβλητών>

ΑΡΧΗ

<εντολές>

.....

ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ

ΑΛΓΟΡΙΘΜΟΣ παράδειγμα

<εντολές>

.....

ΤΕΛΟΣ_ΑΛΓΟΡΙΘΜΟΥ

ΣΥΝΑΡΤΗΣΕΙΣ

Όπως είπαμε το πρώτο είδος υποπρογραμμάτων είναι οι συναρτήσεις. Η σύνταξη μιας συνάρτησης είναι η ακόλουθη:

ΣΥΝΤΑΞΗ ΣΥΝΑΡΤΗΣΗΣ

ΣΥΝΑΡΤΗΣΗ Όνομα_Συνάρτησης(Παράμετροι) : είδος_τιμής_που_επιστρέφεται

Μεταβλητές

<Δήλωση μεταβλητών>

Αρχή

<Εντολές>

ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ

ΣΥΝΤΑΞΗ ΚΛΗΣΗΣ ΣΥΝΑΡΤΗΣΗΣ

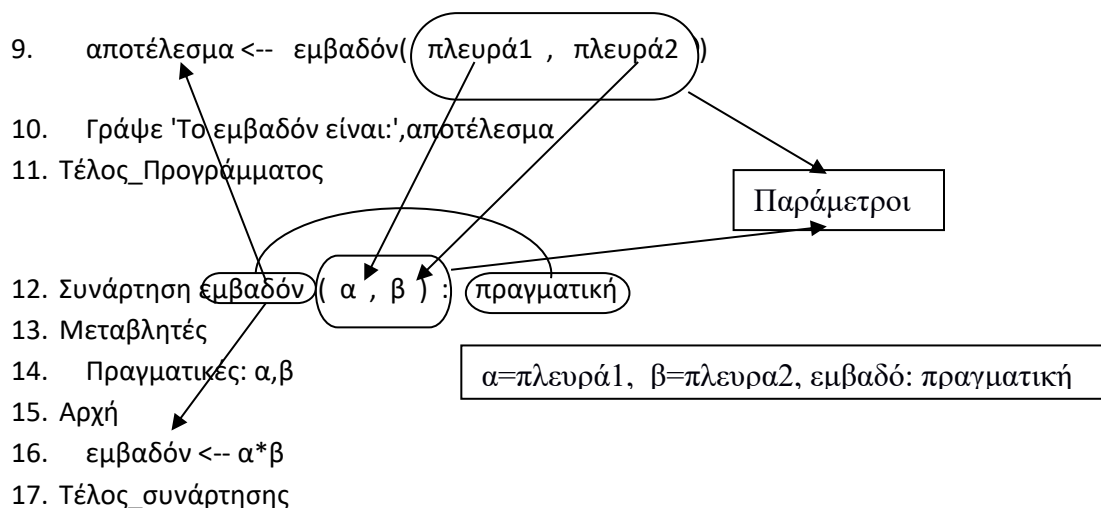
<μεταβλητή> ← Όνομα_Συνάρτησης(Παράμετροι)

Ας δούμε με ένα παράδειγμα το πώς δουλεύει μια συνάρτηση. Ας υποθέσουμε ότι θέλουμε να υπολογίσουμε το εμβαδόν ενός παραλληλογράμμου, όπου ο χρήστης θα δίνει τις διαστάσεις και με τη χρήση μιας συνάρτησης θα υπολογίζουμε το εμβαδό, το οποίο στη συνέχεια θα εμφανίζουμε

Στο πρόγραμμα που ακολουθεί αρχικά ζητάμε από το χρήστη με τις εντολές ΓΡΑΨΕ, ΔΙΑΒΑΣΕ να δώσει τις πλευρές του παραλληλογράμμου. Στη συνέχεια καλούμε τη συνάρτηση που υπολογίζει το εμβαδόν. Το όνομα αυτής της συνάρτησης είναι “εμβαδόν” και παίρνει ως παραμέτρους τις τιμές των μεταβλητών *πλευρά1* και *πλευρά2*. Στη συνέχεια η εκτέλεση του προγράμματος μεταφέρεται στη γραμμή 12. Εκεί στη μεταβλητή *α* αποθηκεύεται η τιμή της μεταβλητής *πλευρά1* και στην *β* αυτής της

πλευρά2. Δηλαδή $\alpha = \text{πλευρά1}$ και $\beta = \text{πλευρά2}$. Το όνομα της συνάρτησης (δηλαδή η λέξη “εμβαδό”) **παίζει το ρόλο μεταβλητής** και η τιμή της είναι αυτή που θα επιστραφεί στη μεταβλητή αποτέλεσμα στη γραμμή 9. Στη γραμμή 12 αυτό που ακολουθεί μετά το σύμβολο “:” είναι το είδος της μεταβλητής που είναι η λέξη “εμβαδό”. Μέσα στη συνάρτηση υπολογίζεται το εμβαδό στη γραμμή 16 και στη συνέχεια η τιμή της μεταβλητής “εμβαδό” (αφού όπως είπαμε το όνομα της συνάρτησης παίζει το ρόλο μεταβλητής), επιστρέφεται και καταχωρείται στη μεταβλητή “αποτέλεσμα” στη γραμμή 9. Στη συνέχεια συνεχίζεται η εκτέλεση του προγράμματος από τη γραμμή 9 και κάτω και μέχρι τη γραμμή 11 όπου τελειώνει το πρόγραμμα. Έτσι μας εμφανίζεται το αποτέλεσμα (γραμμή 10).

1. Πρόγραμμα εμβαδό
2. Μεταβλητές
3. Πραγματικές: πλευρά1, πλευρά2, αποτέλεσμα
4. Αρχή
5. Γράψε 'Δώσε τη 1η πλευρά:'
6. Διάβασε πλευρά1
7. Γράψε 'Δώσε τη 2η πλευρά:'
8. Διάβασε πλευρά2



Βλέπουμε λοιπόν σε όλη τη διαδικασία εκτέλεσης του προγράμματος, αρχικά να ξεκινάει το πρόγραμμα την εκτέλεση του κανονικά (γραμμές 1-9), στη συνέχεια όταν καλείται η συνάρτηση να συνεχίζεται η εκτέλεση του προγράμματος στη συνάρτηση (γραμμές 12-17) και έπειτα επιστρέφει η εκτέλεση του προγράμματος από εκεί που έμεινε (γραμμές 9-11).

Να σημειώσουμε ότι για τις συναρτήσεις ισχύουν τα εξής σημεία:

- Ο αριθμός των παραμέτρων τόσο στη κλήση της συνάρτησης όσο και στη δημιουργία της πρέπει να είναι ο ίδιος, δηλαδή στη κλήση της συνάρτησης έχουμε τις μεταβλητές *πλευρά1* και *πλευρά2* τις οποίες “περνάμε” μέσα στη συνάρτηση. Στη συνάρτηση τώρα έχουμε και εκεί 2 μεταβλητές που δέχονται τις τιμές των *πλευρά1* και *πλευρά2*, τις α και β .
- Η συνάρτηση επιστρέφει **οπωσδήποτε μόνο** μια τιμή.

- Το όνομα της συνάρτησης παίζει το ρόλο μεταβλητής η τιμή της οποίας καταχωρείται στη μεταβλητή που καλεί τη συνάρτηση, δηλαδή στη περίπτωση μας στη μεταβλητή “αποτέλεσμα”.
- Μέσα στη συνάρτηση πρέπει να δηλώσουμε τι είδους μεταβλητές είναι οι παράμετροί της, δηλαδή η τιμές α και β .
- Οι εντολές που μπορούν να εκτελεστούν μέσα σε μία συνάρτηση, μπορούν να είναι περισσότερες από μία.
- Κάποια από τις εντολές μέσα στη συνάρτηση πρέπει αναγκαστικά να εκχωρεί ένα αποτέλεσμα στη τιμή του ονόματος της συνάρτησης (αφού είπαμε το όνομα της συνάρτησης παίζει το ρόλο μεταβλητής). Στη περίπτωση μας “εμβαδό \leftarrow ...”.

Στην αρχή του μαθήματος είδαμε κάποιες ειδικές συναρτήσεις όπως οι $A_M(x)$, $A_T(x)$, $HM(x)$ κλπ. Αυτές οι συναρτήσεις δουλεύουν ακριβώς με τον ίδιο τρόπο που δουλεύουν και οι συναρτήσεις που είδαμε εδώ. Απλά σε αυτές δεν βλέπουμε τον κώδικα τους (αφού είναι αποθηκευμένος σε άλλο αρχείο), αλλά μόνο τις καλούμε.

ΔΙΑΔΙΚΑΣΙΕΣ

Η άλλη μεγάλη κατηγορία των υποπρογραμμάτων είναι οι **ΔΙΑΔΙΚΑΣΙΕΣ**. Ο τρόπος λειτουργίας των διαδικασιών είναι ίδιος με αυτόν των συναρτήσεων. Δηλαδή ξεκινάει κανονικά η εκτέλεση του προγράμματος και όταν βρει τη γραμμή όπου καλείται η διαδικασία, μεταβαίνει μέσα στη διαδικασία να εκτελέσει τις εντολές που βρίσκονται εκεί και στη συνέχεια επιστρέφει η εκτέλεση του προγράμματος από εκεί που σταμάτησε. Όπως ακριβώς συμβαίνει και με τις συναρτήσεις.

ΣΥΝΤΑΞΗ ΔΙΑΔΙΚΑΣΙΑΣ

ΔΙΑΔΙΚΑΣΙΑ Όνομα_Διαδικασίας (Παράμετροι)

Μεταβλητές

<Δήλωση μεταβλητών>

Αρχή

<Εντολές>

ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ

ΣΥΝΤΑΞΗ ΚΛΗΣΗΣ ΔΙΑΔΙΚΑΣΙΑΣ

ΚΑΛΕΣΕ Όνομα_Διαδικασίας (Παράμετροι)

Ας δούμε όμως με ένα παράδειγμα τον ακριβή τρόπο λειτουργίας μιας διαδικασίας. Το πρόγραμμα που ακολουθεί, ζητάει από το χρήστη να δώσει την ακτίνα του κύκλου και στη συνέχεια με μια διαδικασία υπολογίζει και εμφανίζει το εμβαδό του και με μια άλλη διαδικασία υπολογίζει και εμφανίζει τη περίμετρο του κύκλου.

1. Πρόγραμμα εμβαδό_κυκλ
 2. Μεταβλητές
 3. Πραγματικές: ρ, εμβαδό, περίμετρος
 4. Αρχή
 5. Κάλεσε εμβαδό_κύκλου (ρ , εμβαδό)
 6. Γράψε 'Το εμβαδό του κύκλου είναι:', εμβαδό
 7. Κάλεσε περίμετρος_κύκλου(ρ, περίμετρος)
 8. Γράψε 'Η περίμετρος είναι: ', περίμετρος
 9. Τέλος_Προγράμματος
-
10. ΔΙΑΔΙΚΑΣΙΑ εμβαδό_κύκλου (ρ , εμβαδό)
 11. Σταθερές
 12. $\pi=3.14$
 13. Μεταβλητές
 14. Πραγματικές: ρ, εμβαδό
 15. Αρχή
 16. Γράψε 'Δώσε την ακτίνα του κύκλου:'
 17. Διάβασε ρ
 18. $\text{εμβαδό} \leftarrow \pi * \rho^2$
 19. Τέλος_Διαδικασίας
-
20. ΔΙΑΔΙΚΑΣΙΑ περίμετρος_κύκλου(ρ, περίμετρος)
 21. Σταθερές
 22. $\pi=3.14$
 23. Μεταβλητές
 24. Πραγματικές: ρ, περίμετρος
 25. Αρχή
 26. $\text{περίμετρος} \leftarrow 2 * \pi * \rho$
 27. Τέλος_Διαδικασίας

Το πρώτο πράγμα που συμβαίνει όταν εκτελείται το πρόγραμμά μας είναι η κλήση της διαδικασίας “εμβαδό_κύκλου”, στέλνοντας μέσα στη διαδικασία “εμβαδό_κύκλου” τις τιμές των μεταβλητών “ρ” και “εμβαδό”, οι οποίες όμως δεν περιέχουν καμία τιμή. Μέσα στη διαδικασία “εμβαδό_κύκλου”, εκτελούνται οι εντολές τις και στη συνέχεια επιστρέφονται στις μεταβλητές “ρ” και “εμβαδό” του κυρίως προγράμματος οι αντίστοιχες τιμές των “ρ” και “εμβαδό” μέσα στη διαδικασία. Έπειτα εκτελείται η γραμμή 6, η οποία μας εμφανίζει το εμβαδό και στη συνέχεια η γραμμή 7 η οποία καλεί τη διαδικασία “περίμετρος_κύκλου”. Σαν παράμετροι αυτή τη φορά έχουμε τις μεταβλητές “ρ” και “περίμετρος”. Η “περίμετρος” δεν έχει καμία τιμή ακόμα μέσα στο κυρίως πρόγραμμα, οπότε η τιμή που θα περάσει στη μεταβλητή “περίμετρος” της διαδικασίας “περίμετρος_κύκλου”, θα είναι κενή. Αντίθετα η μεταβλητή “ρ” έχει πάρει τιμή πλέον στο κυρίως πρόγραμμα, την οποία και θα περάσει στη παράμετρο “ρ” μέσα στη διαδικασία “περίμετρος_κύκλου”. Στη συνέχεια εκτελούνται οι εντολές της διαδικασίας “περίμετρος_κύκλου” και έπειτα επιστρέφει η εκτέλεση του προγράμματος στο κυρίως πρόγραμμα και στη γραμμή 8, όπου τελειώνει η εκτέλεση του προγράμματος.

Για τις διαδικασίες πρέπει να τονίσουμε τα εξής σημεία:

- Ο αριθμός των παραμέτρων τόσο στη κλήση όσο και στην ίδια τη διαδικασία πρέπει να είναι ίδιος. Δηλαδή στη γραμμή 5 του παραδείγματος έχουμε 2 παραμέτρους, όσες είναι και οι παράμετροι στη γραμμή 9.
- Οι τιμές που επιστρέφονται από τις διαδικασίες μπορούν να είναι περισσότερες από μια, εν αντιθέσει με τις συναρτήσεις όπου η τιμή που επιστέφει μπορεί να είναι μόνο μία. **Αυτή είναι στην ουσία και η βασική διαφορά μεταξύ συναρτήσεων και διαδικασιών**
- Οι παράμετροι κατά την κλήση της συνάρτησης δεν είναι υποχρεωτικό να περιέχουν τιμές. Μπορεί δηλαδή να χρειαστεί να χρησιμοποιήσουμε σαν παράμετρο μια μεταβλητή που δεν έχει τιμή αρχικά, αλλά να της επιστρέφεται μια τιμή από τη διαδικασία.
- Και στις διαδικασίες όπως και στις συναρτήσεις τόσο οι παράμετροι όσο και οι μεταβλητές του προγράμματος και του υποπρογράμματος, πρέπει να δηλώνονται τι είδους μεταβλητές είναι.

ΠΑΡΑΜΕΤΡΟΙ

Οι μεταβλητές των υποπρογραμμάτων είναι ανεξάρτητες από αυτές του κυρίως προγράμματος, εκτός από τις παραμέτρους οι οποίες συνδέονται μεταξύ τους. Δηλαδή αν μέσα σε μια διαδικασία ή μία συνάρτηση έχουμε μια μεταβλητή που έχει το ίδιο όνομα με μια μεταβλητή του κυρίως προγράμματος, τότε αυτές οι δύο μεταβλητές δεν έχουν καμία σχέση μεταξύ τους (εκτός αν είναι παράμετροι). Καταλαβαίνουμε λοιπόν ότι ο λόγος που υπάρχουν οι παράμετροι είναι για να περνάμε τιμές από το κυρίως πρόγραμμα στο υποπρόγραμμα και αντίστροφα, αφού οι εκάστοτε μεταβλητές είναι ανεξάρτητες μεταξύ του. Οι παράμετροι του κυρίως προγράμματος λέγονται **πραγματικές** ενώ αυτές των υποπρογραμμάτων λέγονται **τυπικές**.

Γενικές Παρατηρήσεις για τις Συναρτήσεις και τις Διαδικασίες:

- Τόσο οι συναρτήσεις όσο και οι διαδικασίες πρέπει να θεωρούνται ξεχωριστά προγράμματα από το αρχικό πρόγραμμα.
- Μια διαδικασία ή μια συνάρτηση μπορεί να κληθεί πολλές φορές μέσα σε ένα πρόγραμμα, αλλά το υποπρόγραμμα της θα γραφεί μια φορά. Δηλαδή, στο προηγούμενο παράδειγμα τη διαδικασία “εμβαδό_κύκλου” τη γράψαμε μια φορά, αλλά μπορούμε να την καλέσουμε πολλές φορές στο κυρίως πρόγραμμα. Το ίδιο ισχύει και για τις συναρτήσεις. Αυτό είναι και το μεγάλο πλεονέκτημα των υποπρογραμμάτων.

ΔΙΑΦΟΡΕΣ ΣΥΝΑΡΤΗΣΕΩΝ ΜΕ ΔΙΑΔΙΚΑΣΙΕΣ

Όπως βλέπουμε ο τρόπος λειτουργίας των «Διαδικασιών» και των «Συναρτήσεων» είναι ίδιος. Έχουν όμως κάποιες πολύ σημαντικές διαφορές που τις κάνουν και να ξεχωρίζουν μεταξύ τους.

- Οι συναρτήσεις επιστρέφουν **μόνο** μία τιμή στο κυρίως πρόγραμμα σε αντίθεση με τις διαδικασίες όπου μπορούν να επιστρέψουν στο κυρίως πρόγραμμα περισσότερες από μία τιμές μέσω των παραμέτρων.
- Μια συνάρτηση πρέπει υποχρεωτικά να έχει έστω μια παράμετρο, ενώ η διαδικασία μπορεί να χρησιμοποιηθεί και χωρίς παραμέτρους.
- Όταν καλούμε μία συνάρτηση, οι παράμετροι που στέλνουμε στη συνάρτηση πρέπει να έχουν μία τιμή ειδικά προκύπτει σφάλμα στο πρόγραμμα κάτι το οποίο **δεν ισχύει για τις διαδικασίες**. Στο παράδειγμα που ακολουθεί στη περίπτωση της κλήσης της συνάρτησης ($Z \leftarrow \text{ηλικίες}(X,Y)$) αν δεν υπήρχαν πριν από την κλήση οι εντολές ($X \leftarrow 15$, $Y \leftarrow \text{'Κώστας'}$) όπου δίνουν αρχικές τιμές στις παραμέτρους, η εκτέλεση της εντολής $Z \leftarrow \text{ηλικίες}(X,Y)$ θα έβγαζε σφάλμα διότι πρέπει να δοθούν τιμές στις παραμέτρους πριν γίνει η κλήση της συνάρτησης. Στη περίπτωση της διαδικασίας δεν είναι υποχρεωτική η αρχικοποίηση των παραμέτρων.

ΠΡΟΓΡΑΜΜΑ παράδειγμα

.....

$X \leftarrow 15$

$Y \leftarrow \text{'Κώστας'}$

$Z \leftarrow \text{ηλικίες}(X,Y)$

ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ

ΣΥΝΑΡΤΗΣΗ ηλικίες(α , β): ακέραια

.....

$\text{Ηλικίες} \leftarrow \alpha$

ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ

ΠΡΟΓΡΑΜΜΑ παράδειγμα

.....

ΚΑΛΕΣΕ ηλικίες(X,Y)

ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ

ΔΙΑΔΙΚΑΣΙΑ ηλικίες(α , β)

.....

ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ

ΠΙΝΑΚΕΣ ΩΣ ΠΑΡΑΜΕΤΡΟΙ ΣΕ ΣΥΝΑΡΤΗΣΕΩΝ ΜΕ ΔΙΑΔΙΚΑΣΙΕΣ

Τέλος τόσο στις συναρτήσεις όσο και στις διαδικασίες, εκτός από τιμές, μπορούμε να στείλουμε μέσω τον παραμέτρων και ολόκληρες δομές δεδομένων, όπως τους πίνακες. Ωστόσο, μια συνάρτηση δεν πρέπει να επιστρέφει έναν πίνακα. Στην περίπτωση που απαιτείται η επιστροφή ενός πίνακα, τότε θα χρησιμοποιούμε πάντα διαδικασία.

ΠΡΟΓΡΑΜΜΑ εμβασό_κύκλου

ΜΕΤΑΒΛΗΤΕΣ

ΑΚΕΡΑΙΕΣ: $A[5]$, I

ΠΡΟΓΡΑΜΜΑ εμβασό_κύκλου

ΜΕΤΑΒΛΗΤΕΣ

ΑΚΕΡΑΙΕΣ: $A[5]$, I

ΠΡΑΓΜΑΤΙΚΕΣ: εμβ
 ΑΡΧΗ
 ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 5
 Γράψε 'Δώσε ακτίνα'
 Διάβασε A[I]
 εμβ ← εμβαδό(A[I])
 ΓΡΑΨΕ εμβ
 ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
 ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ

 ΣΥΝΑΡΤΗΣΗ εμβαδό(α) : πραγματική
 ΜΕΤΑΒΛΗΤΕΣ
 Ακέραιες: α
 ΑΡΧΗ
 εμβαδό ← 3.14*α^2
 ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ

ΑΡΧΗ
 ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 5
 Γράψε 'Δώσε ακτίνα'
 ΔΙΑΒΑΣΕ A[I]
 ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
 ΚΑΛΕΣΕ εμβαδό(A)
 ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ

 ΔΙΑΔΙΚΑΣΙΑ εμβαδό(B)
 ΜΕΤΑΒΛΗΤΕΣ
 Ακέραιες: B[5], I
 ΑΡΧΗ
 ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 5
 ΓΡΑΨΕ 3.14*B[I]^2
 ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
 ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ

ΣΥΓΚΡΙΣΗ ΔΙΑΔΙΚΑΣΙΩΝ ΚΑΙ ΣΥΝΑΡΤΗΣΕΩΝ

ΔΙΑΔΙΚΑΣΙΑ	ΣΥΝΑΡΤΗΣΗ
Μπορεί να επιστρέφει περισσότερες από μία τιμές.	Επιστρέφει μόνο μια τιμή.
Επιστρέφει τιμές μέσω των παραμέτρων.	Πρέπει οπωσδήποτε να επιστρέφει μια τιμή.
Μπορεί να μην έχει παραμέτρους.	Πρέπει υποχρεωτικά να έχει τουλάχιστον μια παράμετρο
Μπορεί να έχει σαν παραμέτρους μεταβλητές χωρίς προηγουμένως να έχουν πάρει τιμές.	Δεν μπορεί να έχει σαν παραμέτρους μεταβλητές χωρίς προηγουμένως να έχουν πάρει τιμές, εκτός από τους πίνακες όπου μπορούμε να τους στείλουμε άδειους.

Μπορούμε να στείλουμε σαν παράμετρο ολόκληρη δομή δεδομένων π.χ. πίνακα.	Μπορούμε να στείλουμε σαν παράμετρο ολόκληρη δομή δεδομένων π.χ. πίνακα.
Μπορούμε να επιστρέψουμε ολόκληρη δομή δεδομένων π.χ. πίνακα.	Δεν μπορούμε να επιστρέψουμε ολόκληρη δομή δεδομένων π.χ. πίνακα.
Δηλώνουμε κανονικά τις σταθερές μεταβλητές της διαδικασίας.	Δηλώνουμε κανονικά τις σταθερές μεταβλητές της συνάρτησης.

ΑΣΚΗΣΕΙΣ

1. Να μετατραπούν οι ακόλουθες συναρτήσεις σε διαδικασίες και να συμπληρωθούν τα τμήματα δηλώσεων των μεταβλητών

A) ΣΥΝΑΡΤΗΣΗ $f_1(a,b,c)$: ΑΚΕΡΑΙΑ

```
...
ΑΡΧΗ
  ΑΝ  $A\_T(a-b) < A\_T(b-c)$  ΤΟΤΕ
     $f_1 \leftarrow a$ 
  ΑΛΛΙΩΣ
     $f_1 \leftarrow c$ 
  ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ
```

B) ΣΥΝΑΡΤΗΣΗ $f_2(x,y)$: ΧΑΡΑΚΤΗΡΑΣ

```
...
ΑΡΧΗ
   $x \leftarrow y \text{ div } 2$ 
   $y \leftarrow y \text{ div } 3$ 
  ΑΝ  $x=y$  ΤΟΤΕ
     $f_2 \leftarrow '1'$ 
  ΑΛΛΙΩΣ
     $f_2 \leftarrow '2'$ 
  ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ
```

2. Να μετατραπεί το ακόλουθο πρόγραμμα, ώστε να μη γίνεται χρήση της διαδικασίας.

ΠΡΟΓΡΑΜΜΑ Ασκ ΜΕΤΑΒΛΗΤΕΣ ΑΚΕΡΑΙΕΣ: r,p,d,e ΑΡΧΗ ΔΙΑΒΑΣΕ r, p ΚΑΛΕΣΕ ΔΙΑΔ(r,p) ΓΡΑΨΕ r, p $e \leftarrow p \text{ DIV } 1$ $d \leftarrow r+9$ ΚΑΛΕΣΕ ΔΙΑΔ(e,d) ΓΡΑΨΕ d, e ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ	ΔΙΑΔΙΚΑΣΙΑ ΔΙΑΔ(d,e) ΜΕΤΑΒΛΗΤΕΣ ΑΚΕΡΑΙΕΣ: d,e ΑΡΧΗ $d \leftarrow d+2$ $e \leftarrow e-2$ ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ
---	--

3. Να μετατραπεί η ακόλουθη συνάρτηση σε διαδικασία:

```
ΣΥΝΑΡΤΗΣΗ ΣΥΝΑΡΤ(X,Z) : ΠΡΑΓΜΑΤΙΚΗ
ΜΕΤΑΒΛΗΤΕΣ
  ΠΡΑΓΜΑΤΙΚΕΣ: X,Z
ΑΡΧΗ
  ΑΝ  $X>Z$  ΤΟΤΕ
    ΣΥΝΑΡΤ  $\leftarrow X$ 
  ΑΛΛΙΩΣ
    ΣΥΝΑΡΤ  $\leftarrow (X+Z)/2$ 
  ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ
```

4. Να γίνει πρόγραμμα που εφαρμόζει την ταξινόμηση φυσαλίδας σε ένα πίνακα 10 θέσεων, χρησιμοποιώντας τη μέθοδο της “διαδικασίας” για την αντιμετάθεση των στοιχείων
5. Να γίνει πρόγραμμα που υπολογίζει και εμφανίζει το άθροισμα 10 βαθμών που δίνει ο χρήστης και στη συνέχεια το μέσο όρο τους. Το άθροισμα να υπολογιστεί με τη χρήση συνάρτησης και ο μέσος όρος με τη χρήση διαδικασίας. (P_90)

6. Να γίνει πρόγραμμα που υπολογίζει τη προπέδια ενός αριθμού που δίνει ο χρήστης, με τη χρήση υποπρογράμματος. (P_88)
7. Να γραφεί πρόγραμμα το οποίο με τη χρήση διαδικασιών:
- Γεμίζει ένα πίνακα με 10 θέσεων με αριθμούς που δίνει ο χρήστης
 - Ταξινομεί αυτό τον πίνακα
 - Εμφανίζει το πίνακα (P_87)
8. Να γραφεί πρόγραμμα που ζητάει από το χρήστη να συμπληρώσει ένα πίνακα 12 θέσεων και με τη χρήση υποπρογράμματος να υπολογίζει το εύρος των τιμών του πίνακα. Το εύρος των τιμών υπολογίζεται από τη διαφορά την μεγαλύτερης με τη μικρότερη τιμή. Για τον υπολογισμό της μικρότερης και της μεγαλύτερης τιμής να χρησιμοποιηθεί υποπρόγραμμα. (P_96)
9. Μια εταιρεία παροχής υπηρεσιών σταθερής τηλεφωνίας έχει πάγιο ίσο με 33€ και ο συνδρομητής της εταιρείας μπορεί να μιλήσει σε σταθερά απεριόριστα. Το ίδιο πρόγραμμα δίδει μία ώρα δωρεάν κλήσεων σε κινητά τηλέφωνα. Κάθε λεπτό κλήσεων προς κινητά, πέραν της μίας ώρας, κοστίζει 5 λεπτά του Ευρώ. Να γραφεί πρόγραμμα στη ΓΛΩΣΣΑ το οποίο:
- Διαβάζει τα λεπτά κλήσεων προς κινητά που έχει μιλήσει ένας συνδρομητής του προγράμματος.
 - Με χρήση συνάρτησης που θα καλείται στο πρόγραμμα να υπολογίζει το ποσό πληρωμής για τις κλήσεις προς κινητά.
 - Να υπολογίζει και εμφανίζει το τελικό ποσό πληρωμής που προκύπτει από το πάγιο και τις κλήσεις προς κινητά, στο οποίο όμως πρέπει να προστεθεί ΦΠΑ 23% (P_92)
10. Μια εταιρία κινητής τηλεφωνίας χρεώνει τους πελάτες της σύμφωνα με τον παρακάτω πίνακα:

ΠΑΓΙΟ : 8,3 €	
ΜΗΝΥΜΑΤΑ	ΧΡΕΩΣΗ
Μέχρι 100	0,012 €/μην
Από 101 και πάνω	0,01 €/μην
ΧΡΟΝΟΣ ΟΜΙΛΙΑΣ (sec)	ΧΡΕΩΣΗ
Χρόνος <= 3000 sec	0,05 €/sec
3000sec < Χρόνος <= 6000 sec	0,04 €/sec
Χρόνος > 6000sec	0,02 €/sec

Να γραφτεί κυρίως πρόγραμμα το οποίο αφού διαβάσει τον αριθμό μηνυμάτων ενός συνδρομητή και το χρόνο ομιλίας σε sec (να ελεγχθεί ότι είναι θετικοί αριθμοί), να εμφανίζει την χρέωση που του αναλογεί. Ο υπολογισμός της χρέωσης να γίνει με την βοήθεια κατάλληλου υποπρογράμματος.(P_99)

11. Να γίνει πρόγραμμα που με τη χρήση υποπρογράμματος ψάχνει να βρει τις ακόλουθες τιμές στους αντίστοιχους πίνακες, εμφανίζοντας τη θέση στην οποία βρίσκονται οι εκάστοτε τιμές:

A:	3	8	10	5	6	8	1	9	11	12	Τιμή 8	
B:	22	18	15	22	13	17	19	20	24	27	22	Τιμή 22
Γ:	4	9	17	11	33	54	Τιμή 54					

12. Να γίνει προσομοίωση της συνάρτησης $T_P()$ χωρίς τη χρήση της. (P_122)

ΕΡΓΑΣΙΕΣ

ΕΡΓΑΣΙΑ 1

Να αναπτύξετε πρόγραμμα σε ΓΛΩΣΣΑ που υλοποιεί το παρακάτω animation. Το πρόγραμμα να σταματάει όταν ο χρήστης πατήσει το κουμπί STOP της γλωσσομάθειας.

```
@
@@
@@@
@@@@
@@@@@
@@@@@
@@@@
@@@
@@
@
@@
@@@
@@@@
@@@@@
@@@@@
@@@@
@@@
@@
@
@@
@@@
@@@@
@@@@@
@@@@@
@@@@
@@@
@@
@
```


ΕΡΓΑΣΙΑ 2

Να αναπτύξετε πρόγραμμα το οποίο υλοποιεί το παιχνίδι της τρίλιζας όπως φαίνεται στα παραδείγματα παρακάτω:

ΤΡΙΛΙΖΑ ΠΑΡΑΔΕΙΓΜΑ 1

Δώσε το όνομα του 1ου παίκτη:

Κώστας

Δώσε το όνομα του 2ου παίκτη:

Πέτρος

- - -

- - -

- - -

Κώστας δώσε την επιλογή σου (π.χ. 2):

5

Η επιλογή καταχωρήθηκε

- - -

- O -

- - -

Πέτρος δώσε την επιλογή σου (π.χ. 2):

7

Η επιλογή καταχωρήθηκε

- - -

- O -

X - -

Κώστας δώσε την επιλογή σου (π.χ. 2):

3

Η επιλογή καταχωρήθηκε

- - O

- O -

X - -

Πέτρος δώσε την επιλογή σου (π.χ. 2):

2

Η επιλογή καταχωρήθηκε

- X O

- O -

X - -

Κώστας δώσε την επιλογή σου (π.χ. 2):

6

Η επιλογή καταχωρήθηκε

- X O

- O O

X - -

Πέτρος δώσε την επιλογή σου (π.χ. 2):

1

Η επιλογή καταχωρήθηκε

X X O

- O O

X - -

Κώστας δώσε την επιλογή σου (π.χ. 2):

4

Η επιλογή καταχωρήθηκε

X X O

O O O

X - -

ΝΙΚΗΤΗΣ ΕΙΝΑΙ Ο: Κώστας

Τ Ε Λ Ο Σ Π Α Ι Χ Ν Ι Δ Ι Ο Υ

ΤΡΙΛΙΖΑ ΠΑΡΑΔΕΙΓΜΑ 2

Δώσε το όνομα του 1ου παίκτη:

Μαρία

Δώσε το όνομα του 2ου παίκτη:

Στέλλα

- - -

- - -

- - -

Μαρία δώσε την επιλογή σου (π.χ. 2):

4

Η επιλογή καταχωρήθηκε

- - -

O - -

- - -

Στέλλα δώσε την επιλογή σου (π.χ. 2):

5

Η επιλογή καταχωρήθηκε

- - -

O X -

- - -

Μαρία δώσε την επιλογή σου (π.χ. 2):

1

Η επιλογή καταχωρήθηκε

O - -

O X -

- - -

Στέλλα δώσε την επιλογή σου (π.χ. 2):

7

Η επιλογή καταχωρήθηκε

O - -

O X -

X - -

Μαρία δώσε την επιλογή σου (π.χ. 2):

3

Η επιλογή καταχωρήθηκε

O - O

O X -

X - -

Στέλλα δώσε την επιλογή σου (π.χ. 2):

2

Η επιλογή καταχωρήθηκε

O X O

O X -

X - -

Μαρία δώσε την επιλογή σου (π.χ. 2):

8

Η επιλογή καταχωρήθηκε

O X O

O X -

X O -

Στέλλα δώσε την επιλογή σου (π.χ. 2):

9

Η επιλογή καταχωρήθηκε

O X O

O X -

X O X

Μαρία δώσε την επιλογή σου (π.χ. 2):

6

Η επιλογή καταχωρήθηκε

O X O

O X O

X O X

ΤΟ ΠΑΙΧΝΙΔΙ ΕΛΗΞΕ ΙΣΟΠΑΛΟ

Τ Ε Λ Ο Σ Π Α Ι Χ Ν Ι Δ Ι Ο Υ

ΕΡΓΑΣΙΑ 3

Να σχεδιασθεί και να δημιουργηθεί με τη γλώσσα προγραμματισμού ΓΛΩΣΣΑ εφαρμογή που προσομοιώνει τη λειτουργία του ATM.

Η εφαρμογή δημιουργεί 30 λογαριασμούς τραπεζής με αριθμό λογαριασμού από το 2000 μέχρι το 2030. Στους λογαριασμούς αυτούς αναθέτει 1000€.

Το ATM διαθέτει θυρίδες χαρτονομισμάτων των 20€ και 50€ για τις ανάγκες ανάληψης και κατάθεσης από τους πελάτες. Τις θυρίδες αυτές γεμίζει υπάλληλος της τράπεζας με συγκεκριμένο όνομα χρήστη, στην περίπτωση μας το επώνυμό σας με λατινικούς κεφαλαίους χαρακτήρες και κωδικό (password) ένα τετραψήφιο νούμερο της επιλογής σας.

Για τις συναλλαγές το ATM ζητά από τον χρήστη να πληκτρολογήσει τον τραπεζικό του λογαριασμό και το PIN του. Το PIN για κάθε τραπεζικό λογαριασμό, είναι ο αριθμός λογαριασμού αντεστραμμένος, π.χ. για τον αριθμό λογαριασμού 2016, το PIN είναι 6102.

Στο ATM ο πελάτης μπορεί να κάνει ΑΝΑΛΗΨΗ, ΕΝΗΜΕΡΩΣΗ, ΜΕΤΑΦΟΡΑ από λογαριασμό σε λογαριασμό και ΚΑΤΑΘΕΣΗ με λογαριασμό.

Η εφαρμογή λειτουργεί μέσα από το μενού επιλογών

ΠΕΙΡΑΜΑΤΙΚΟ ATM ΑΕΠΠ

**_*_*_*_*_*_*_*_*_*_*_*

- | | |
|--------------|-------------|
| 1. ΕΝΗΜΕΡΩΣΗ | 2. ΑΝΑΛΗΨΗ |
| 3. ΜΕΤΑΦΟΡΑ | 4. ΚΑΤΑΘΕΣΗ |

ΠΑΡΑΚΑΛΩ ΕΠΙΛΕΞΤΕ ΤΙ ΘΕΛΕΤΕ(1-4):

Ο υπάλληλος της τράπεζας, έχει τρεις κρυφές επιλογές, την 101, 102, και 103.

Μέσω της επιλογής 101 μπορεί να ενημερωθεί για την κατάσταση των θυρίδων χαρτονομισμάτων, δηλαδή να μάθει πόσα χαρτονομίσματα υπάρχουν στην κασέτα των 50ευρώ και πόσα στην κασέτα των 20ευρώ.

Μέσω της επιλογής 102 ενεργοποιεί την ΤΡΟΦΟΔΟΣΙΑ τοποθετώντας περιοδικά στο ATM νέα χαρτονομίσματα.

Με την επιλογή 103 αδρανοποιεί το ATM, κλείνει οριστικά τη λειτουργία του, και ολοκληρώνεται βέβαια το πρόγραμμα.

Η εφαρμογή ξεκινά έχοντας στο ATM 20 20ευρα και 10 50εύρα, οπότε το ATM μπορεί να εξυπηρετήσει για ανάληψη τους πελάτες, αρχικά μέχρι το ποσό των 900€.

Σε κάθε μια από τις ενέργειες 1, 2, 3 το σύστημα ζητά από τον χρήστη τον τραπεζικό του λογαριασμό και το PIN. Μόνο αν μπορεί να επιβεβαιώσει το συνδυασμό (λογαριασμό-PIN) προχωρά στην εξυπηρέτηση του πελάτη. Αλλιώς του απαντά ότι κάνει λάθος στο PIN και του εμφανίζει πόσες προσπάθειες έχει ακόμα στη διάθεσή του μέχρι τις τρεις. Αν κάνει 3 λανθασμένες προσπάθειες το σύστημα αυτόματα ΟΛΟΚΛΗΡΩΝΕΙ ΤΗ ΣΥΝΑΛΛΑΓΗ και επιστρέφει στο αρχικό μενού.

Για να κάνει ανάληψη ο χρήστης επιλέγει 3, την ΑΝΑΛΗΨΗ, και στις προτροπές που του εμφανίζονται πληκτρολογεί τον τραπεζικό του λογαριασμό, το PIN του και το ποσό που επιθυμεί. Το ATM ελέγχει αν ο λογαριασμός του πελάτη καλύπτει ή όχι το αιτούμενο ποσό. Στη συνέχεια ελέγχει αν μπορεί να σχηματίσει με 20ευρα και 50ευρα το ποσό αυτό και του απαντά ανάλογα. Για παράδειγμα δεν μπορεί να κάνει ανάληψη 10€ ή 125€. Αν ο λογαριασμός καλύπτει το αιτούμενο ποσό, το ATM προσπαθεί να του δώσει το ποσό με τον ελάχιστο αριθμό χαρτονομισμάτων. Για το σκοπό αυτό ελέγχει την κατάσταση των θυρίδων. Αν για παράδειγμα έχουν εξαντληθεί τα 50ευρα, θα του δώσει όλο το ποσό σε 20ευρα, και αυτό βέβαια αν έχει τη δυνατότητα αυτή. Διαφορετικά θα του δώσει όσα περισσότερα

50ευρα μπορεί και τα υπόλοιπα θα τα συμπληρώσει με 20ευρα. Το ίδιο ισχύει σε περίπτωση που δεν επαρκούν τα 20ευρα για το σχηματισμό του ποσού. Π.χ. για ανάληψη 210€ θα δώσει 3 50ευρα και 3 20ευρα, αν όμως έχουν εξαντληθεί πλήρως τα 50€ θα ενημερώσει τον πελάτη ότι δεν μπορεί να τον ικανοποιήσει. Το ίδιο ισχύει και αν έχουν εξαντληθεί τα 20ευρα. Για παράδειγμα αν ζητηθεί ανάληψη 150€ και το ΑΤΜ δεν διαθέτει κανένα 50ευρο θα τον ενημερώσει ότι αδυνατεί να τον εξυπηρετήσει. Σε περίπτωση κατάθεσης ισχύει το ίδιο, ο πελάτης μπορεί να καταθέσει ποσά που μπορούν να σχηματισθούν με 50ευρα και 20ευρα. Στην περίπτωση κατάθεσης όμως το σύστημα ζητά από τον πελάτη να το ενημερώσει πόσα 50ευρα και πόσα 20ευρα καταθέτει, για να ενημερώσει και τις αντίστοιχες θυρίδες για τις επόμενες κινήσεις. Στην ΚΑΤΑΘΕΣΗ το σύστημα ζητά μόνο τον τραπεζικό λογαριασμό στον οποίο θα γίνει η κατάθεση και δεν έχει ανάγκη από το PIN του. Ζητά το ποσό της κατάθεσης σύμφωνα με τους όρους που προδιαγράφηκαν, τον αριθμό των χαρτονομισμάτων σε 50ευρα και σε 20ευρα και τέλος εμφανίζει το νέο υπόλοιπο.

Στη μεταφορά από λογαριασμό σε λογαριασμό το σύστημα ζητά από τον χρήστη που θα κάνει τη μεταφορά τον τραπεζικό του λογαριασμό και το PIN του και το σύστημα τον ενημερώνει για το υπόλοιπο του λογαριασμού. Στη συνέχεια ζητά το ποσό που θέλει να μεταφέρει ο χρήστης, το οποίο μπορεί να είναι οποιοδήποτε, ακόμα και δεκαδικό, και το σύστημα ελέγχει αν επαρκεί το υπόλοιπο του λογαριασμού για τη μεταφορά αυτή. Μετά ζητά το λογαριασμό στον οποίο θα γίνει η μεταφορά. Το σύστημα πραγματοποιεί τη μεταφορά και ενημερώνει για το υπόλοιπο του λογαριασμού.

Για ΕΝΗΜΕΡΩΣΗ το σύστημα ζητά τον τραπεζικό λογαριασμό και το PIN του πελάτη και εμφανίζει το υπόλοιπο του τραπεζικού λογαριασμού που έχει εκείνη τη στιγμή.

Όταν ο χρήστης ολοκληρώνει μια ενέργεια το σύστημα επανέρχεται στο αρχικό μενού και περιμένει την επόμενη κίνηση.

Όταν ολοκληρώνει οποιαδήποτε συναλλαγή μηδενίζει τις μεταβλητές στις οποίες είχε καταχωρήσει πληροφορίες, όπως ο τραπεζικό λογαριασμός και το PIN του χρήστη.

ΟΔΗΓΙΕΣ

Η εργασία είναι πολυσύνθετη και απαιτεί πολλές απόψεις και γνώμες. Γι' αυτό σας συνιστώ να συνεργαστείτε.

Θα σας πρότεινα και αυτός είναι άλλωστε ο σκοπός της εργασίας, να επιμερίσετε τις δραστηριότητες σε επιμέρους κατάλληλα υποπρογράμματα. Για παράδειγμα η ενημέρωση θα αποτελούσε ένα υποπρόγραμμα, η μεταφορά ένα άλλο, το αρχικό μενού τρίτο, ο έλεγχος των στοιχείων χρήστη άλλο, η δημιουργία του PIN άλλο κλπ.

Η επιλογή των παραμέτρων και τους είδους των υποπρογραμμάτων έχει καθοριστικό ρόλο. Φροντίστε λοιπόν να επιλέξετε τα υποπρογράμματα με κατάλληλο τρόπο, αν επιστρέφουν ή όχι τιμή, αν οι τιμές των παραμέτρων τους μεταβάλλονται στο εσωτερικό τους και πρέπει η μεταβολή αυτή να είναι γνωστή και αλλού.

Δουλέψτε το πρόγραμμα όσο περισσότερο μπορείτε. Θέλω να αποκτήσετε εμπειρία στην ανάλυση ενός προβλήματος και τη σχεδίαση και οργάνωση του κατά τμήματα.

ΣΗΜΕΙΩΣΗ: Στη ΓΛΩΣΣΟΜΑΘΕΙΑ και **ΜΟΝΟ** στο συγκεκριμένο λογισμικό και **ΟΧΙ ΣΤΗ ΓΛΩΣΣΑ** μπορείτε να κάνετε χρήση της εντολής ΓΡΑΨΕ_. **Η χρήση της είναι αποδεκτή ΜΟΝΟ για τη συγκεκριμένη άσκηση και για καμία άλλη άσκηση.** Πληροφορίες για τη χρήση της θα βρείτε στη βοήθεια της ΓΛΩΣΣΟΜΑΘΕΙΑΣ.

ΠΡΟΣΟΧΗ ΔΕΝ ΙΣΧΥΕΙ Η ΓΡΑΨΕ_ ΓΙΑ ΤΟ ΜΑΘΗΜΑ ΣΑΣ.