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

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 απο το όνομα του αλγορίθμου. Στο παράδειγμα παρακάτω, φαίνεται η δομή ενός αλγορίθμου.

Παράδειγμα1:
Αλγόριθμος παράδειγμα_1
Εμφάνισε 'Διάβσε ένα νούμερο:'
Διάβασε x
z←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←3

y←46.5

Οι **ΑΛΦΑΡΙΘΜΗΤΙΚΕΣ** είναι μεταβλητές που περιέχουν κάποιο αλφαριθμητικό (δηλαδή χαρακτήρες με αριθμούς μαζί), πχ
α←'Σήμερα έχουμε 25 βαθμούς Κελσίου'
c←'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←ΑΛΗΘΗΣ
r←ΨΕΥΔΗΣ.
Οι ΛΟΓΙΚΕΣ μεταβλητές χρησιμοποιούνται κυρίως σε συνθήκες και επαναληπτικές εντολές.

ΑΡΙΘΜΗΤΙΚΟΙ ΤΕΛΕΣΤΕΣ

ΑΡΙΘΜΗΤΙΚΟΙ ΤΕΛΕΣΤΕΣ		ΣΥΝΑΡΤΗΣΕΙΣ	
+	Πρόσθεση	ΗΜ(x)	Ημίτονο
-	Αφαίρεση	ΣΥΝ(x)	Συνημίτονο
*	Πολλαπλασιασμός	ΕΦ(x)	Εφαπτομένη
/	Διαίρεση	Τ_Ρ(x)	Τετραγωνική ρίζα
^	Ύψωση σε δύναμη	ΛΟΓ(x)	Λογάριθμος
DIV	Πηλίκο Ακέραιας Διαίρεσης	Ε(x)	e^x
MOD	Υπόλοιπο Ακέραιας Διαίρεσης	Α_Μ(x)	Ακέραιο μέρος του x
		Α_Τ(x)	Απόλυτη τιμή του x

Π.χ.

x ← 2^3 αντιστοιχεί σε ($x=2^3$)

*y ← (3*5)-2* αντιστοιχεί σε (y=(3*5)-2)

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

c ← 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 ← έκφραση

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

Π.χ.

α ← 3

ημέρα ← '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←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 ← 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 ← 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 ίσο
>	μεγαλύτερο
>=	μεγαλύτερο 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.**

x	y
x MOD y	x DIV y

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

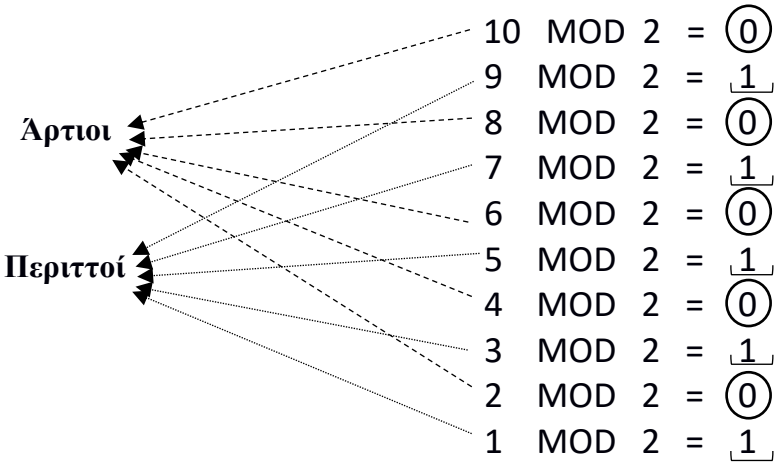
α. Οι αριθμοί μεταξύ των οποίων γίνεται η διαίρεση (τα x και y), πρέπει να είναι ακέραιοι αριθμοί. Δε μπορούμε δηλαδή να κάνουμε *12.5 DIV 2* ή *7 MOD 2.3*
β. Το αποτέλεσμα είτε της πράξης DIV είτε της πράξης MOD θα είναι ακέραιος αριθμός. Οπότε αν σε κάποιο ερώτημα δείτε κάτι τέτοιο *11 DIV 2 = 5.5* 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 MOD 9 = 4, 21 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 MOD 2 = 0** ελέγχουμε αν ο αριθμός x είναι άρτιος. Με την πράξη **x MOD 2 = 1** ελέγχουμε αν ο αριθμός x είναι περιττός.

2. Πολλαπλάσια ενός αριθμού

x MOD <αριθμός> = 0

Γνωρίζουμε ότι οι αριθμοί που είναι πολλαπλάσιοι ενός άλλου αριθμού δεν αφήνουν υπόλοιπο (π.χ. 70 MOD 7 = 0). Οπότε ο τελεστής MOD μπορεί να χρησιμοποιηθεί για να ελέγξουμε αν ένας αριθμός είναι πολλαπλάσιος ενός άλλου. Άρα αν ένας αριθμός x είναι πολλαπλάσιος ενός αριθμού *(<αριθμός>*) τότε το ακέραιο υπόλοιπο της διαίρεσης τους είναι 0 Πχ.

110 MOD 11 = 0

77 MOD 7 = 0

56 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 DIV 10 = 235

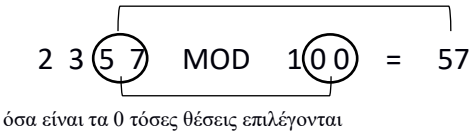
2357 DIV 100 = 23

2357 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 MOD 10 = 7
2357 MOD 100 = 57
2357 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:



ΥΠΟΛΟΓΙΣΜΟΣ ΤΙΜΗΣ ΑΠΟ ΕΝΑ ΠΟΣΟΣΤΟ

Σε πολλές περιπτώσεις ασκήσεων, 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:

*Ποσοστό_Τιμής = Τιμή * Ποσοστό*

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

Κρατήσεις = 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

Τελική_Τιμή = Αρχική_Τιμή + Ποσοστό_Τιμής ή αλλιώς

Τελική_Τιμή = Αρχική_Τιμή + (Αρχική_Τιμή*Ποσοστό)

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

*Τελικός_Μισθός = Μισθός – (Μισθός*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 (*Τελική_Τιμή = Αρχική_Τιμή + (Αρχική_Τιμή*Ποσοστό)*) μόνο που αυτή τη φορά θα κάνουμε αφαίρεση διότι η τελική τιμή προκύπτει έπειτα από έκπτωση. Οπότε:

*Τελική_Τιμή = Αρχική_Τιμή - (Αρχική_Τιμή*Ποσοστό)*

*12 = Αρχική_Τιμή – (Αρχική_Τιμή * 20/100)*

$$12 = \frac{80 * Αρχική\_Τιμή}{100}$$

$$Αρχική\_Τιμή = \frac{12}{0{,}8} = 15$$

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:

*Ποσοστό = (τιμή της οποίας το ποσοστό ψάχνουμε / συνολική τιμή) * 100*

Συνεπώς στο παράδειγμα με το βαρέλι το ποσοστό είναι:

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

1η) Σύνταξη εντολής

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

2η) Σύνταξη εντολής

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

3η) 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 ΑΝ 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 1η σύνταξη εντολής)
2. Αν η *<συνθήκη>* είναι ΑΛΗΘΗΣ, τότε εκτελούνται όλες οι *<εντολές>* (βλέπε 1η 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 2η 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 ΑΝ γίνεται όπως στη 3η 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 ΑΝ. 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 ΑΝ. 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 ΑΝ.

Παρακάτω φαίνονται κάποια παραδείγματα της εντολής ΑΝ:

```
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 3η σύνταξη της εντολής ΑΝ. Μπορεί να γραφεί και με τη χρήση της 1ης σύνταξης της εντολής ΑΝ, 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'
```

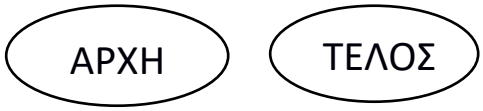
```
Τέλος_αν
Αν αριθμός = 0 τότε
    Εμφάνισε 'Ο αριθμός είναι 0'
Τέλος_αν
Τέλος παράδειγμα_3
```

ΔΙΑΓΡΑΜΜΑΤΑ ΡΟΗΣ

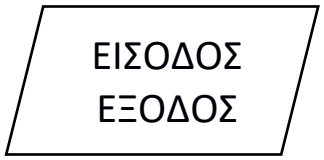
Το διάγραμμα ροής είναι η μετατροπή-αναπαράσταση ενός προγράμματος σε μορφή σχημάτων. Το διάγραμμα ροής βοηθάει τον προγραμματιστή στην υλοποίηση του προγράμματος και είναι η πιο γενική μορφή ενός προγράμματος, ακόμα πιο γενική και από τον αλγόριθμο.

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:

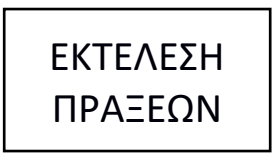
- έλλειψη, που δηλώνει την αρχή και το τέλος του κάθε αλγορίθμου,
- ρόμβος, που δηλώνει μία ερώτηση με δύο ή περισσότερες εξόδους για απάντηση,
- ορθογώνιο, που δηλώνει την εκτέλεση μίας ή περισσότερων πράξεων, και
- πλάγιο παραλληλόγραμμο, που δηλώνει είσοδο ή έξοδο στοιχείων.



Έλλειψη που δηλώνει την αρχή ή το τέλος του κάθε αλγορίθμου

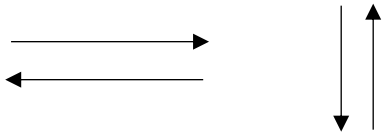


Πλάγιο παραλληλόγραμμο που δηλώνει είσοδο ή έξοδο στοιχείων (εισαγωγή δεδομένων ή έξοδο αποτελεσμάτων)



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**.**

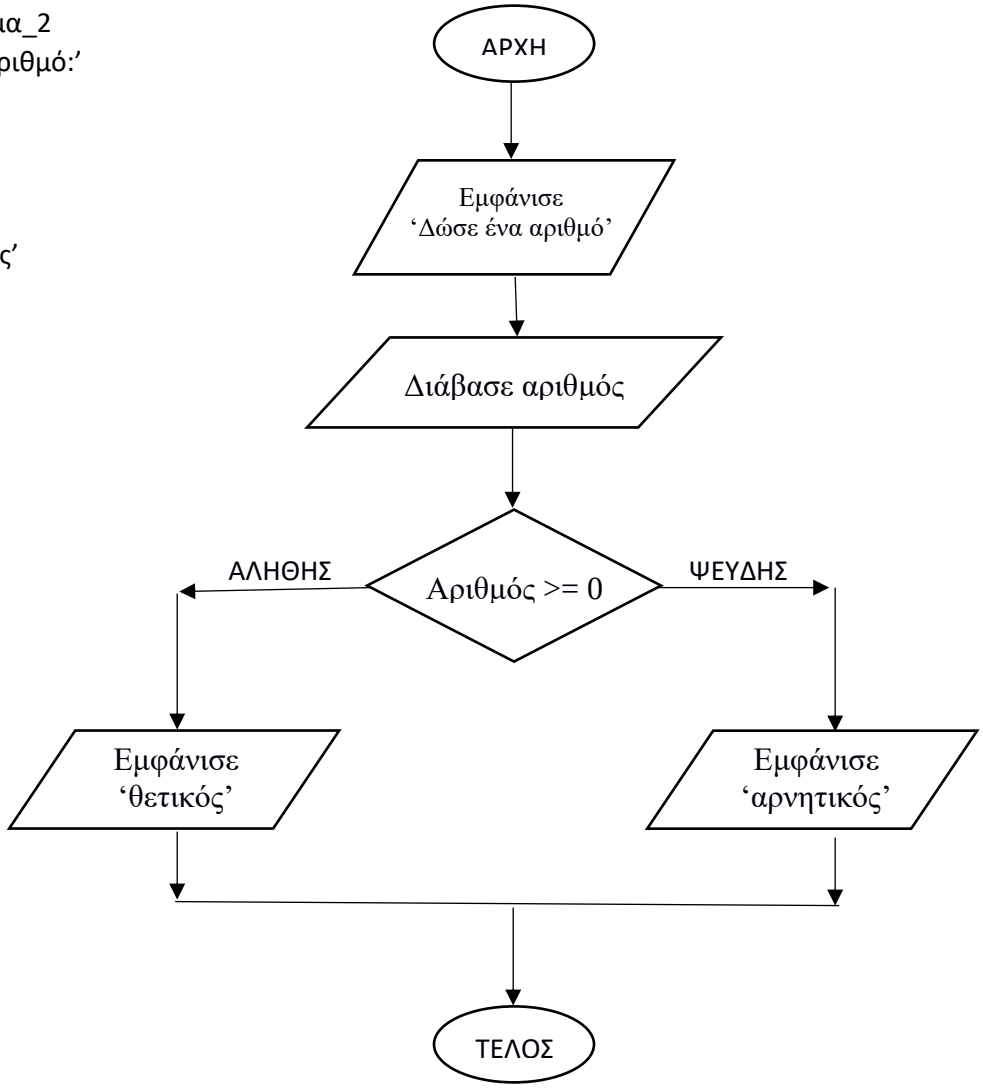
Τα ορθογώνια συνήθως χρησιμοποιούνται για τα κομμάτια του αλγορίθμου που έχουν την **εντολή εκχώρησης** (←).

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.

- *Έξοδος*: 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>=0 τότε
   Εμφάνισε 'Θετικός'
Αλλιώς
   Εμφάνισε 'Αρνητικός'
Τέλος_αν
```

Παράδειγμα 2

```
x← -3
Αν x>=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←9
Διάβασε x
y← i/x
Εμφάνισε y
```

Παράδειγμα 4

```
i←9
Διάβασε x
Αν x<>0 τότε
   y← 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←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 αλγόριθμος 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← 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. π=3.14. Ακολουθεί η δήλωση των ΜΕΤΑΒΛΗΤΩΝ τιμών και στη συνέχεια αρχίζει η δημιουργία του κυρίους 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 ← 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 π=3.14, g=6.68*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 α='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 'Κειμενο = ',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 5η 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←3.5
6.     Γράψε x
7. ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

B. **Λογικά Σφάλματα**: Είναι το ποιο 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 7η 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
```

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 ποιο 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, δλδ 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 παράδειγμα3

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	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: γινόμενο= 10 * 3 ΓΡΑΨΕ 10* 3 =30 πολ2 = 10 + 1 οπότε η πολ2 γίνεται πλέον πολ2=11
11η επανάληψη: πολ2=11	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: ΓΡΑΨΕ 'Αυτή ήταν η προπέδια του',πολ1 όπου πολ1=3

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 χρήστη 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 ← 0
ΕΜΦΑΝΙΣΕ 'Δώσε ένα αριθμό:'
ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
        ΔΙΑΒΑΣΕ μεταβλητή1
        σύνολο ← 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> ← <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←3 πολ2 ← 1 ΟΣΟ πολ2 <=10 ΕΠΑΝΑΛΑΒΕ γινόμενο ← πολ2 * πολ1	πολ1←3 πολ2 ← 1 ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ γινόμενο ← πολ2 * πολ1	πολ1←3 ΓΙΑ πολ2 ΑΠΟ 1 ΜΕΧΡΙ 10 γινόμενο ← 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 πότε χρησιμοποιείται η μία ή άλλη εντολή επανάληψης; θα δούμε κάποιες κατευθύνσεις πότε είναι προτιμότερο να χρησιμοποιούμε κάποια επανάληψη έναντι της άλλη, 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:

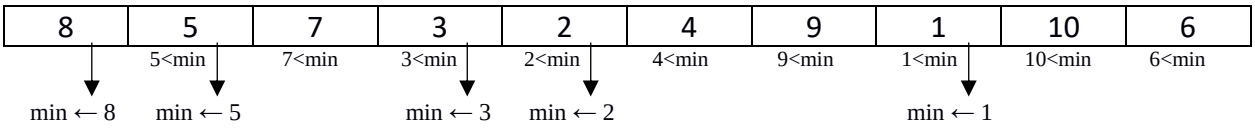
Εντολή ΓΙΑ	**Εντολή ΟΣΟ**	**Εντολή ΜΕΧΡΙΣ_ΟΤΟΥ**

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 ← <αρχική τιμή συνόλου>
[αρχή_επανάληψη]
    Αν <τιμή συνόλου> < min τότε
        min ← <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:

i. Δημιουργούμε μια μεταβλητή που θα περιέχει την ελάχιστη τιμή (min)
ii. Δίνουμε σε αυτή τη μεταβλητή την πρώτη τιμή του συνόλου
iii. 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.
iv. Επαναλαμβάνεται το βήμα iii) για την επόμενη τιμή, μέχρι να ελεγχθεί όλο το σύνολο τιμών.

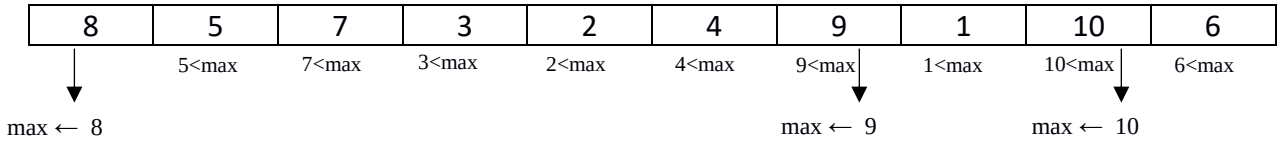
Παράδειγμα1

```
Πρόγραμμα Παράδειγμα1
Μεταβλητές
        Ακέραιες: αριθμός, min
Αρχή
        Γράψε 'Δώσε ένα αριθμό:'
        Διάβασε αριθμός
        min ←  αριθμός
        Οσο αριθμός > 0 Επανάλαβε
                Αν αριθμός < min τότε
                        min ← 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 ← <αρχική τιμή συνόλου>
[αρχή_επανάληψη]
    Αν <τιμή συνόλου> > max τότε
        max ← <τιμή συνόλου>
    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.

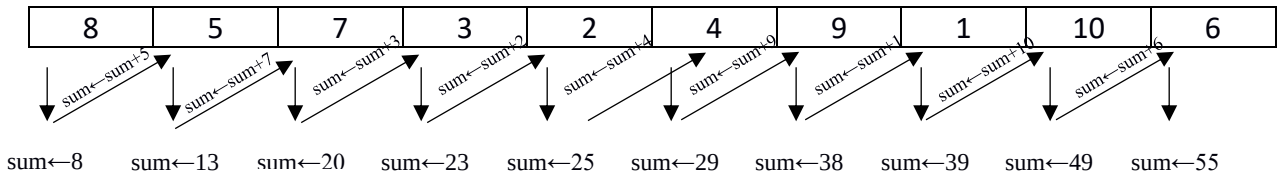
Παράδειγμα2

```
Πρόγραμμα Παράδειγμα2
Μεταβλητές
        Ακέραιες: αριθμός, max
Αρχή
        Γράψε 'Δώσε ένα αριθμό:'
        Διάβασε αριθμός
        max ← αριθμός
        Οσο αριθμός > 0 Επανάλαβε
                Αν αριθμός > max τότε
                        max ← αριθμός
                Τέλος_αν
                Γράψε 'Δώσε ένα αριθμό:'
                Διάβασε αριθμός
        Τέλος_Επανάληψης
        Γράψε 'Ο μεγαλύτερος αριθμός είναι:', max
Τέλος_Προγράμματος
```

ΥΠΟΛΟΓΙΣΜΟΣ ΑΘΡΟΙΣΜΑΤΟΣ, ΓΙΝΟΜΕΝΟΥ

Σύνταξη:
sum ← 0 //Αρχικοποίηση
[αρχή_επανάληψη]
    sum ← 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←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← 0 //Αρχικοποίηση
[αρχή_επανάληψη]
    *γινόμενο ← 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 ← 0
    κ ← 0
    Γράψε 'Δώσε Αριθμό:'
    Διάβασε n
    Όσο κ <=3 Επανάλαβε
        sum ← sum + n
        Γράψε 'Δώσε Αριθμό:'
        Διάβασε n
        κ ← 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 επανάληψης, 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 ← 0
    Γράψε 'Δώσε ένα αριθμό:'
    Διάβασε αριθμός
    Οσο αριθμός > 0 Επανάλαβε
        count ← 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 ← 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 (μέσος_όρος=άθροισμα/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: ΜΟ
Αρχή
    count ← 0
    sum ← 0
    Γράψε 'Δώσε αριθμό'
    Διάβασε αριθμός
    Όσο αριθμός >= 0 Επανάλαβε
        count ← count +1
        sum ← sum + αριθμός
        Γράψε 'Δώσε αριθμό'
        Διάβασε αριθμός
    Τέλος_Επανάληψης
    ΜΟ ← sum/count
    Γράψε 'Ο μέσος όρος είναι:',ΜΟ
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 μέσος_όρος=άθροισμα/πλήθος.

Παρατηρήσεις

- Στη πλειοψηφία των περιπτώσεως που απαιτείται κάποιος από τους παραπάνω υπολογισμούς (άθροισμα, μέγιστο, ελάχιστο κλπ.) χρειάζεται η χρήση κάποιας επανάληψης.
- Τις περισσότερε 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).

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 y, τότε δηλώνουμε τη μεταβλητή y ως ακέραιο πίνακα οπότε η μεταβλητή y θα γίνει:

θέση:	*1*	*2*	*3*	*4*	*5*
y:	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 y ή αλλιώς y[1], η τιμή 4 θα βρίσκεται στη θέση 3 της μεταβλητής y ή αλλιώς y[3], η τιμή 9 στην y[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: y[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 Α
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 Α
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” (ένας πίνακας με 5 θέσεις). Καταλαβαίνουμε λοιπόν ότι για να προσπελάσουμε όλες τις θέσεις ενός πίνακας 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 ποιο 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 πράξει 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 ← θ[1]
Για i Απο 2 Μέχρι 7
  Αν θ[i] <min τότε
  min ← θ[i]
  Τέλος_αν
Τέλος_Επανάληψης
	min ← θ[1]
	i ← 2
	Όσο i <= 7 Επανάλαβε
	  Αν θ[i] <min τότε
	  min ← θ[i]
	  Τέλος_αν
	  i ← i + 1
	Τέλος_Επανάληψης
		min ← θ[1]
		i ← 2
		Αρχή_Επανάληψης
		  Αν θ[i] <min τότε
		  min ← θ[i]
		  Τέλος_αν
		  i ← i + 1
		Μέχρις_Ότου i>7

ΜΕΓΙΣΤΟ

ΓΙΑ...ΑΠΟ...ΜΕΧΡΙ	ΟΣΟ	ΜΕΧΡΙΣ_ΟΤΟΥ

max ← θ[1]
Για i Απο 2 Μέχρι 7
  Αν θ[i] > max τότε
  max ← θ[i]
  Τέλος_αν
Τέλος_Επανάληψης
	max ← θ[1]
	i ← 2
	Όσο i <= 7 Επανάλαβε
	  Αν θ[i] > max τότε
	  max ← θ[i]
	  Τέλος_αν
	  i ← i + 1
	Τέλος_Επανάληψης
		max ← θ[1]
		i ← 2
		Αρχή_Επανάληψης
		  Αν θ[i] > max τότε
		  max ← θ[i]
		  Τέλος_αν
		  i ← 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

sum ← 0
Για i Απο 1 Μέχρι 7
  sum ← sum + θ[i]
Τέλος_Επανάληψης
avg ← sum/7
	sum ← 0
	i ← 1
	Όσο i <= 7 Επανάλαβε
	  sum ← sum + θ[i]
	  i ← i + 1
	Τέλος_Επανάληψης
	avg ← sum/7
		sum ← 0
		i ← 1
		Αρχή_Επανάληψης
		  sum ← sum + θ[i]
		  i ← i + 1
		Μέχρις_Ότου i>7
		avg ← 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 ← ψευδής
position ← 0
i ← 1
Όσο done=ψευδής  και  i<=n  επανάλαβε
        Αν table[i] = key  τότε
                done ← αληθής
                position ← i
        αλλιώς
                i ← 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

Επαναλήψεις	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	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

αλλιώς i ← 2 + 1 Τέλος_αν Τέλος_Επανάληψης	
3η επανάληψη, i=3: done ← ψευδής position ← 0 i ← 1 Όσο done=ψευδής και 3<=7 επανάλαβε 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<=n και θα σταματήσει η επανάληψη. Γι' αυτό και η συνθήκη της επανάληψης έχει δυο περιορισμούς (done=ψευδής και i<=n). Ο ένας (done=ψευδής) ισχύει για όσο διάστημα δεν έχει βρεθεί η τιμή που αναζητάμε μέσα στον πίνακα και ο άλλος (i<=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← 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
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	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

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
```

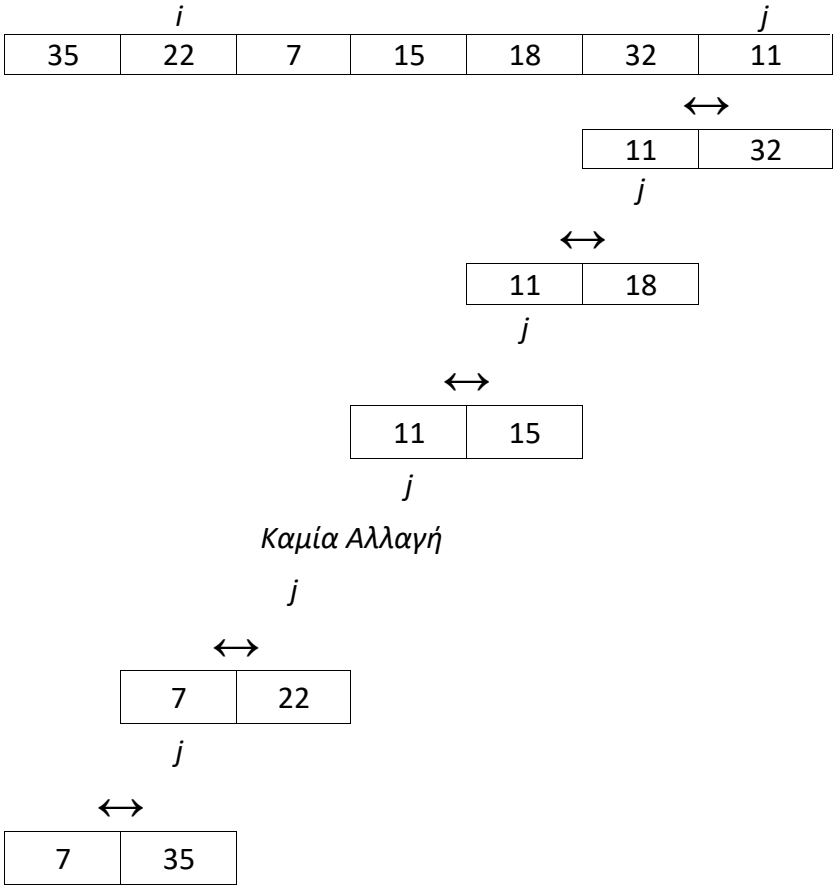
Με τον όρο “Αντιμετάθεσε”, εννοούμε την αλλαγή των τιμών μεταξύ τους. Δηλαδή η τιμή της μιας μεταβλητής θα μεταφερθεί στην άλλη. Αυτό για να γίνει προγραμματιστικά χρειάζεται μια ενδιάμεση τρίτη μεταβλητή, προκειμένου να αποθηκεύσει προσωρινά μια από τις 2 τιμές. Δηλαδή η αντιμετάθεση γίνεται με τον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

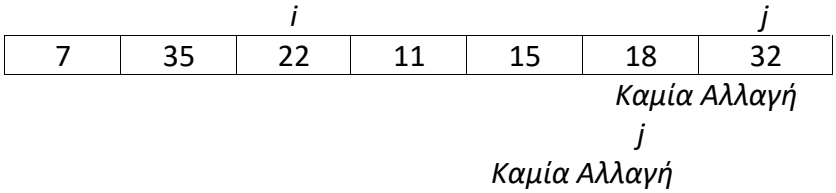
1η επανάληψης:

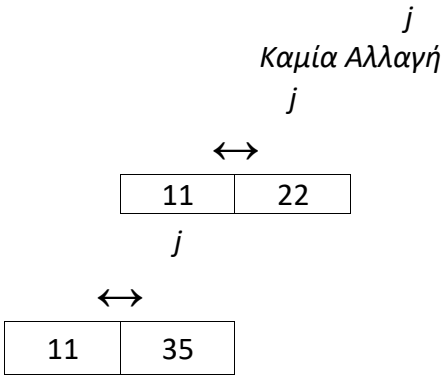


Η λογική του αλγορίθμου είναι σε κάθε προσπέλαση όλου του πίνακα να βρίσκεται το μικρότερο στοιχείο και να τοποθετείται στην αριστερότερη θέση. Στην επόμενη επανάληψη να βρίσκεται το αμέσως μικρότερο στοιχείο και να τοποθετείται στην αμέσως αριστερότερη θέση κλπ.
Όπως βλέπουμε χρησιμοποιούνται δυο μετρητές (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 τι 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 αντιμετάθεση 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 επανάληψης:



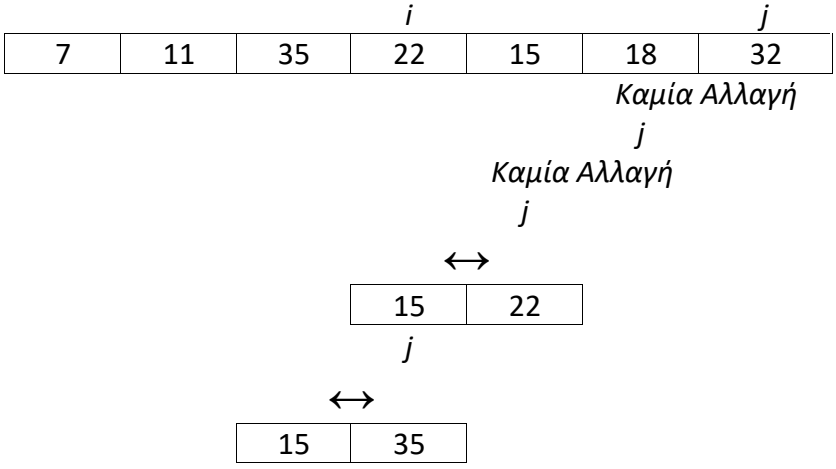


Άρα ο πίνακας στο τέλος της 2ης επανάληψης θα είναι:

7	11	35	22	15	18	32

Παρατηρούμε ότι στης 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.

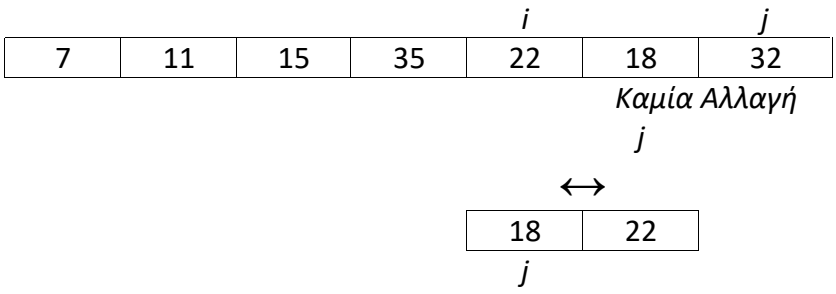
3η επανάληψης:

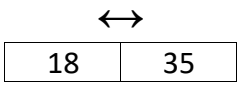


Ο πίνακας μετά την 3η επανάληψη θα είναι:

7	11	15	35	22	18	32

4η επανάληψης:

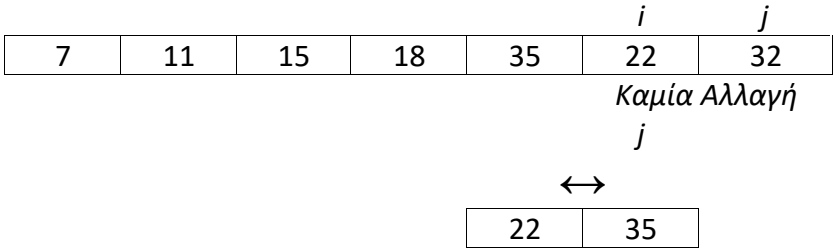




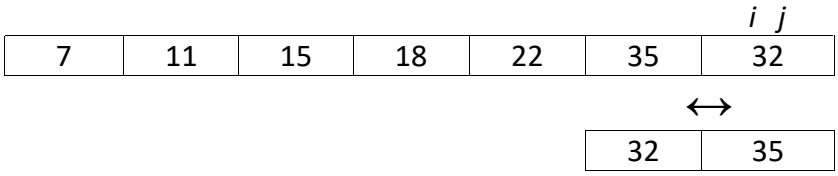
Ο πίνακας μετά την 4η επανάληψη θα είναι:

7	11	15	18	35	22	32

5η επανάληψης:



6η επανάληψης:



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*
θ1:	28.3	27.4	29.1	28.1	26.3	25.8	26.7
θ2:	24.2	23.5	24.6	25	23.9	22	22.6
θ3:	20.5	21.3	21.7	22.3	23.1	22.9	23.5
θ4:	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 πίνακας, 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: θ[4,7]

Προσπέλαση Δισδιάστατοι Πίνακα

	j	*j*	*j*	*j*	*j*	*j*	*j*
i	28.3	27.4	29.1	28.1	26.3	25.8	26.7
i	24.2	23.5	24.6	25	23.9	22	22.6
i	20.5	21.3	21.7	22.3	23.1	22.9	23.5
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 θ[2,3].
Έτσι λοιπόν η καταχώρηση δεδομένων σε ένα δισδιάστατο πίνακα θα έχει την εξής μορφή:

ΓΙΑ...ΑΠΟ...ΜΕΧΡΙ	ΟΣΟ	ΜΕΧΡΙΣ_ΟΤΟΥ

Πρόγραμμα Α
Μεταβλητές
  Ακέραιες: i,j
  Πραγματικές: θ[4,7]
Αρχή
  Για i Απο 1 Μέχρι 4
	Πρόγραμμα Α
	Μεταβλητές
	  Ακέραιες: i,j
	  Πραγματικές: θ[4,7]
	Αρχή
	  i ← 1
		Πρόγραμμα Α
		Μεταβλητές
		  Ακέραιες: i, j
		  Πραγματικές: θ[4,7]
		Αρχή
		  i ← 1

```
Για j Απο 1 Μέχρι 7
  Γράψε 'Δώσε Θερμοκρασία:'
  Διάβασε θ[i,j]
 Τέλος_Επανάληψης
Τέλος_Επανάληψης
Τέλος_Προγράμματος
```

```
Όσο i <= 4 Επανάλαβε
  j ← 1
  Όσο <=7 Επανάλαβε
   Γράψε 'Δώσε Θερμοκρασία:'
   Διάβασε θ[i,j]
   j ← j + 1
   Τέλος_Επανάληψης
   i ← i + 1
 Τέλος_Επανάληψης
Τέλος_Προγράμματος
```

```
Αρχή_Επανάληψης
 j ← 1
 Αρχή_Επανάληψης
  Γράψε 'Δώσε Θερμοκρασία:'
  Διάβασε θ[i,j]
  j ← j + 1
  Μέχρις_Ότου j>7
  i ← i + 1
 Μέχρις_Ότου i>4
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 θ[1,1]	2η προσπέλαση θ[1,2]	3η προσπέλαση θ[1,3]	4η προσπέλαση θ[1,4]	5η προσπέλαση θ[1,5]	6η προσπέλαση θ[1,6]	7η προσπέλαση θ[1,7]
i	8η προσπέλαση θ[2,1]	9η προσπέλαση θ[2,2]	10η προσπέλαση θ[2,3]	11η προσπέλαση θ[2,4]	12η προσπέλαση θ[2,5]	13η προσπέλαση θ[2,6]	14η προσπέλαση: θ[2,7]
i	15η προσπέλαση θ[3,1]	16η προσπέλαση θ[3,2]	17η προσπέλαση θ[3,3]	18η προσπέλαση θ[3,4]	19η προσπέλαση θ[3,5]	20η προσπέλαση θ[3,6]	21η προσπέλαση θ[3,7]
i	22η προσπέλαση θ[4,1]	23η προσπέλαση θ[4,2]	24η προσπέλαση θ[4,3]	25η προσπέλαση θ[4,4]	26η προσπέλαση θ[4,5]	27η προσπέλαση θ[4,6]	28η προσπέλαση θ[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 θ[10,4] τότε ο αλγόριθμο θα είναι (και με τις τρεις εντολές επανάληψης):

ΕΛΑΧΙΣΤΟ

ΓΙΑ...ΑΠΟ...ΜΕΧΡΙ	ΟΣΟ	ΜΕΧΡΙΣ_ΟΤΟΥ

min ← θ[1,1]
Για i Από 1 Μέχρι 10
  Για j Από 1 Μέχρι 4
  Αν θ[i,j] <min τότε
  min ← θ[i,j]
  Τέλος_αν
  Τέλος_Επανάληψης
Τέλος_Επανάληψης
	min ← θ[1,1]
	i ← 1
	Όσο i <= 10 Επανάλαβε
	  j←1
	  Όσο j <=4 Επανάλαβε
	  Αν θ[i,j] <min τότε
	  min ← θ[i,j]
	  Τέλος_αν
	  j ← j + 1
	  Τέλος_Επανάληψης
	  i← i + 1
	Τέλος_Επανάληψης
		min ← θ[1,1]
		i ← 1
		Αρχή_Επανάληψης
		  j← 1
		  Αρχή_Επανάληψης
		  Αν θ[i,j] <min τότε
		  min ← θ[i,j]
		  Τέλος_αν
		  j ← j + 1
		  Μέχρις_Όπου j>4
		  i ← i + 1
		Μέχρις_Όπου i>10

ΜΕΓΙΣΤΟ

ΓΙΑ...ΑΠΟ...ΜΕΧΡΙ	ΟΣΟ	ΜΕΧΡΙΣ_ΟΤΟΥ

max ← θ[1,1]
Για i Από 1 Μέχρι 10
  Για j Από 1 Μέχρι 4
  Αν θ[i,j] > max τότε
  max ← θ[i,j]
  Τέλος_αν
  Τέλος_Επανάληψης
Τέλος_Επανάληψης
	max ← θ[1,1]
	i ← 1
	Όσο i <= 10 Επανάλαβε
	  j←1
	  Όσο j <=4 Επανάλαβε
	  Αν θ[i,j] > max τότε
	  max ← θ[i,j]
	  Τέλος_αν
	  j ← j + 1
	  Τέλος_Επανάληψης
	  i← i + 1
	Τέλος_Επανάληψης
		max ← θ[1,1]
		i ← 1
		Αρχή_Επανάληψης
		  j← 1
		  Αρχή_Επανάληψης
		  Αν θ[i,j] > max τότε
		  max ← θ[i,j]
		  Τέλος_αν
		  j ← j + 1
		  Μέχρις_Όπου j>4
		  i ← i + 1
		Μέχρις_Όπου 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 τρείς αλγόριθμοι (ένας για κάθε επανάληψη) για τον υπολογισμό του μέσου όρου ενός δισδιάστατου πίνακα θ[10,4] είναι:

ΓΙΑ...ΑΠΟ...ΜΕΧΡΙ	ΟΣΟ	ΜΕΧΡΙΣ_ΟΤΟΥ
	sum ← 0	sum ← 0
	i ← 1	i ← 1
sum ← 0	Όσο i <= 10 Επανάλαβε	Αρχή_Επανάληψης
Για i Απο 1 Μέχρι 10	j←1	j← 1
Για j Από 1 Μέχρι 4	Όσο j <=4 Επανάλαβε	Αρχή_Επανάληψης
sum ← sum + θ[i,j]	sum ← sum + θ[i,j]	sum ← sum + θ[i,j]
Τέλος_Επανάληψης	j ← j + 1	j ← j + 1
Τέλος_Επανάληψης	Τέλος_επανάληψης	Μέχρις_Ότου j>4
avg ← sum/40	i ← i + 1	i ← i + 1
	Τέλος_Επανάληψης	Μέχρις_Ότου i>10
	avg ← sum/40	avg ← sum/40

<u>Αναζήτηση Δισδιάστατου Πίνακα</u>

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 ← 1	I ← 1	
Όσο i<=n επανάλαβε	***Αρχή_Επανάληψης***	
j ← 1	***j ← 1***	***Για i απο 1 μέχρι n***
Όσο j<=m επανάλαβε	Αρχή_επανάληψης	Για j απο 1 μέχρι m
Αν Πίνακας[i,j] = ΣΑ τότε	Αν Πίνακας[i,j] = ΣΑ τότε	Αν Πίνακας[i,j] = ΣΑ τότε
Γράψε Το στοιχείο βρέθηκε΄	Γράψε Το στοιχείο βρέθηκε΄	Γράψε 'Το στοιχείο βρέθηκε'
Τέλος_Αν	Τέλος_Αν	Τέλος_Αν
j← j + 1	j← j + 1	Τέλος_Επανάληψης
Τέλος_Επανάληψης	Μέχρις_'Οτου j>m	***Τέλος_Επανάληψης***
i←i+1	***i←i+1***	
Τέλος_Επανάληψης	***Μέχρις_Ότου i>n***	

Όπου 'Πίνακας[]' 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 επανάληψης 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:
Α) η πρώτη επανάληψη πρέπει αυτή τη φορά να αναφέρεται στις στήλες,
Β) η δεύτερη εσωτερική επανάληψη να αναφέρεται στις γραμμές
Δηλαδή να έχουμε τον προηγούμενο δισδιάστατο πίνακα Α[4,7] και θέλουμε να υπολογίσουμε το άθροισμα ανά στήλη αυτή τη φόρα, 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

Α) 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.
Β) 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 Α 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.

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 το 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. 1ο υποπρόγραμμα: Υπολογίζουμε τη περίμετρο.
3. 2ο υποπρόγραμμα: Υπολογίζουμε το εμβαδό.
4. 3ο 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:

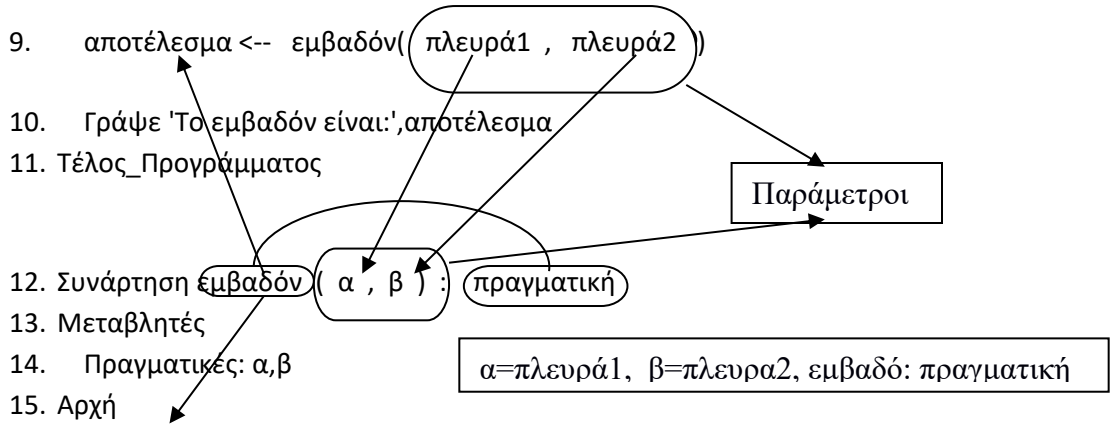
ΣΥΝΤΑΞΗ ΣΥΝΑΡΤΗΣΗΣ
ΣΥΝΑΡΤΗΣΗ Όνομα_Συνάρτησης(Παράμετροι) : είδος_τιμής_που_επιστρέφεται
Μεταβλητές
<Δήλωση μεταβλητών>
Αρχή
<Εντολές>
ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ

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 α=πλευρά1 και β=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
9. αποτέλεσμα <-- εμβαδόν(πλευρά1 , πλευρά2)
10. Γράψε 'Το εμβαδόν είναι:',αποτέλεσμα
11. Τέλος_Προγράμματος
12. Συνάρτηση εμβαδόν (α , β) : πραγματική
13. Μεταβλητές
14. Πραγματικές: α,β
15. Αρχή
16. εμβαδόν <-- α*β
17. Τέλος_συνάρτησης



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 ίδιο, 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 περίπτωσή 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 περίπτωσή μας "εμβαδό←…".

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.     π=3.14
13. Μεταβλητές
14.     Πραγματικές: ρ, εμβαδό
15. Αρχή
16.     Γράψε 'Δώσε την ακτίνα του κύκλου:'
17.     Διάβασε ρ
18.     εμβαδό <-- π*ρ^2
19. Τέλος_Διαδικασίας

20. ΔΙΑΔΙΚΑΣΙΑ περίμετρος_κύκλου(ρ, περίμετρος)
21. Σταθερές
22.     π=3.14
23. Μεταβλητές
24.     Πραγματικές: ρ, περίμετρος
25. Αρχή
26.     περίμετρος <-- 2*π*ρ
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 ειδάλλως προκύπτει σφάλμα στο πρόγραμμα κάτι το οποίο **δεν ισχύει για τις διαδικασίες**. Στο παράδειγμα που ακολουθεί στη περίπτωση της κλήση της συνάρτησης (Z← ηλικίες(X,Y)) αν δεν υπήρχαν πριν από την κλήση οι εντολές (X←15 , Y←'Κώστας') όπου δίνουν αρχικές τιμές στις παραμέτρους, η εκτέλεση της εντολής Z← 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←15
Y←'Κώστας'
Z← ηλικίες(X,Y)
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ

ΣΥΝΑΡΤΗΣΗ ηλικίες(α, β): ακέραια
……
Ηλικίες← α
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 εμβαδό_κύκλου
ΜΕΤΑΒΛΗΤΕΣ
   ΑΚΕΡΑΙΕΣ: Α[5], Ι
   ΠΡΑΓΜΑΤΙΚΕΣ: εμβ
ΑΡΧΗ
    ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 5
       Γράψε ‘Δώσε ακτίνα’
       Διάβασε Α[Ι]
       εμβ← εμβαδό(Α[Ι})
       ΓΡΑΨΕ εμβ
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ

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

```
ΠΡΟΓΡΑΜΜΑ εμβαδό_κύκλου
ΜΕΤΑΒΛΗΤΕΣ
   ΑΚΕΡΑΙΕΣ: Α[5], Ι
ΑΡΧΗ
    ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 5
       Γράψε ‘Δώσε ακτίνα’
       ΔΙΑΒΑΣΕ Α[Ι]
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
    ΚΑΛΕΣΕ εμβαδό(Α)
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ

ΔΙΑΔΙΚΑΣΙΑ εμβαδό(Β)
ΜΕΤΑΒΛΗΤΕΣ
  Ακέραιες: Β[5], Ι
ΑΡΧΗ
    ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 5
       ΓΡΑΨΕ 3.14*Β[Ι]^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
	Δεν μπορεί να έχει σαν παραμέτρους μεταβλητές χωρίς προηγουμένως να έχουν

Μπορεί να έχει σαν παραμέτρους μεταβλητές χωρίς προηγουμένως να έχουν πάρει τιμές.	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.

ΘΕΜΑΤΑ ΠΑΝΕΛΛΗΝΙΩΝ

ΘΕΜΑΤΑ 2014

ΘΕΜΑ Α

Α1. Να γράψετε στο τετράδιό σας τον αριθμό καθεμιάς από τις παρακάτω προτάσεις **1-5** και, δίπλα, τη λέξη **ΣΩΣΤΟ**, αν η πρόταση είναι σωστή, ή τη λέξη **ΛΑΘΟΣ**, αν η πρόταση είναι λανθασμένη.

1. Οι εκφράσεις διαμορφώνονται από τους τελεστέους και τους τελεστές. (μονάδες 2)

2. Σκοπός της ταξινόμησης είναι να διευκολυνθεί στη συνέχεια η αναζήτηση των στοιχείων του ταξινομημένου πίνακα. (μονάδες 2)

3. Το εκτελέσιμο πρόγραμμα δημιουργείται ακόμα και στην περίπτωση που το αρχικό πρόγραμμα περιέχει λογικά, αλλά όχι συντακτικά λάθη. (μονάδες 2)

4. Οι λογικές τιμές είναι οι εξής: ΟΧΙ, ΚΑΙ, Ή. (μονάδες 2)

5. Μεταξύ των εντολών του σώματος μιας συνάρτησης πρέπει υποχρεωτικά να υπάρχει τουλάχιστον μία εντολή εκχώρησης τιμής στο όνομα της συνάρτησης. (μονάδες 2)

Μονάδες 10

Α2. Να γράψετε στο τετράδιό σας:

α. Ένα συγκριτικό τελεστή. (μονάδα 1)

β. Ένα λογικό τελεστή. (μονάδα 1)

γ. Μία λογική σταθερά. (μονάδα 1)

δ. Μία απλή λογική έκφραση. (μονάδα 1)

ε. Μία σύνθετη λογική έκφραση. (μονάδα 1)

Μονάδες 5

Α3. Δίνονται οι τιμές των μεταβλητών Χ=8 και Ψ=4 και η παρακάτω έκφραση:

(ΟΧΙ (9mod5 = 20-4*2^2)) ΄Η (Χ>Ψ ΚΑΙ "Χ">"Ψ")

Να υπολογίσετε την τιμή της έκφρασης αναλυτικά, ως εξής:
α. Να αντικαταστήσετε τις μεταβλητές με τις τιμές τους. (μονάδα 1)
β. Να εκτελέσετε τις αριθμητικές πράξεις. (μονάδα 1)
γ. Να αντικαταστήσετε τις συγκρίσεις με την τιμή ΑΛΗΘΗΣ, αν η σύγκριση είναι αληθής, ή με την τιμή ΨΕΥΔΗΣ, αν η σύγκριση είναι ψευδής. (μονάδα 1)
δ. Να εκτελέσετε τις λογικές πράξεις, ώστε να υπολογίσετε την τελική τιμή της έκφρασης. (μονάδες 2)

Μονάδες 5

Α4. α. Να γράψετε τους κανόνες που πρέπει να ακολουθούνται στη χρήση των εμφωλευμένων βρόχων με εντολές ΓΙΑ. (μονάδες 6)
β. Ποιος είναι ο ρόλος του συντάκτη σε ένα προγραμματιστικό περιβάλλον; (μονάδες 2)
γ. Ποιος είναι ο ρόλος του συνδέτη-φορτωτή σε ένα προγραμματιστικό περιβάλλον; (μονάδες 2)
δ. Ποιος είναι ο ρόλος του μεταγλωττιστή σε ένα προγραμματιστικό περιβάλλον; (μονάδες 2)

Μονάδες 12

Α5. Δίνεται το παρακάτω ημιτελές τμήμα αλγορίθμου:

```
Α ← ...
Β ← ...
Αρχή_επανάληψης
        Β ← ...
        Α ← ...
Μέχρις_ότου Α>200
Εμφάνισε Β
```

Να ξαναγράψετε στο τετράδιό σας το παραπάνω τμήμα αλγορίθμου με τα κενά συμπληρωμένα, έτσι ώστε να υπολογίζει και να εμφανίζει το άθροισμα των περιττών ακεραίων από το 100 έως το 200.

Μονάδες 8

ΘΕΜΑ Β

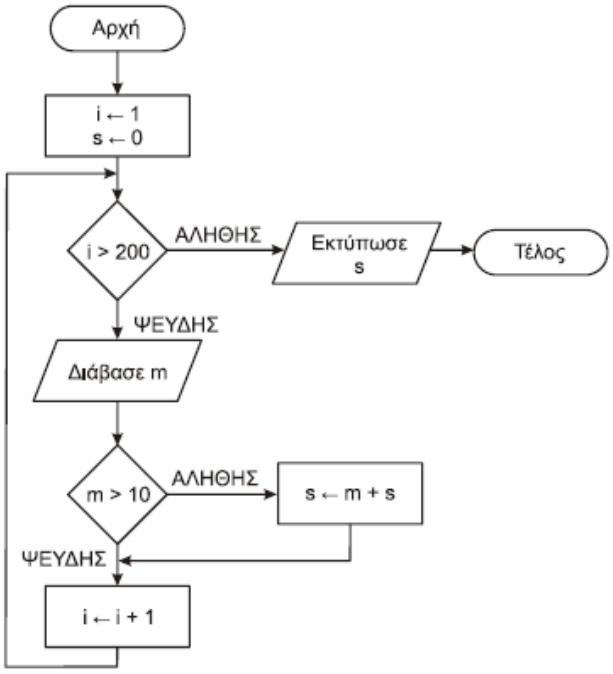
Β1. Για την ταξινόμηση, σε φθίνουσα σειρά, των στοιχείων ενός μονοδιάστατου πίνακα αριθμών Π[30] μπορεί να ακολουθηθεί η παρακάτω διαδικασία:
Αρχικά, ο πίνακας σαρώνεται από την αρχή μέχρι το τέλος του, προκειμένου να βρεθεί το μεγαλύτερο στοιχείο του. Αυτό το στοιχείο τοποθετείται στην αρχή του πίνακα, ανταλλάσσοντας θέσεις με το στοιχείο της πρώτης θέσης του πίνακα. Η σάρωση του πίνακα επαναλαμβάνεται, ξεκινώντας τώρα από το δεύτερο στοιχείο του πίνακα. Το μεγαλύτερο από τα στοιχεία που απέμειναν ανταλλάσσει θέσεις με το στοιχείο της δεύτερης θέσης του πίνακα. Η σάρωση επαναλαμβάνεται, ξεκινώντας από το τρίτο στοιχείο του πίνακα, μετά από το τέταρτο στοιχείο του πίνακα κ.ο.κ.
Το παρακάτω ημιτελές τμήμα αλγορίθμου κωδικοποιεί την παραπάνω διαδικασία:

```
Για k από 1 μέχρι 29
    θ ← (1) ...
    Για i από k μέχρι 30
        Αν Π[i] (2) ... Π[θ] τότε
            θ ← (3) ...
        Τέλος_αν
    Τέλος_επανάληψης
    αντιμετάθεσε (4) ... , (5) ...
Τέλος_επανάληψης
```

Να γράψετε στο τετράδιό σας τους αριθμούς (1) έως (5), που αντιστοιχούν στα κενά του αλγορίθμου και, δίπλα σε κάθε αριθμό, ό,τι πρέπει να συμπληρωθεί, ώστε να γίνεται σωστά η ταξινόμηση.

Μονάδες 10

Β2. Δίνεται ο παρακάτω αλγόριθμος:



Να κωδικοποιήσετε τον παραπάνω αλγόριθμο σε ψευδογλώσσα.

Μονάδες 10

ΘΕΜΑ Γ

Ένας πελάτης αγοράζει προϊόντα από ένα κατάστημα. Να αναπτύξετε αλγόριθμο ο οποίος:

Γ1. Για κάθε προϊόν που αγοράζει ο πελάτης, να διαβάζει τον κωδικό του, τον αριθμό τεμαχίων που αγοράστηκαν και την τιμή τεμαχίου. Η διαδικασία ανάγνωσης να σταματά, όταν δοθεί ως κωδικός ο αριθμός 0.

Μονάδες 3

Γ2. Αν ο λογαριασμός δεν υπερβαίνει τα 500 ευρώ, να εμφανίζει το μήνυμα «ΠΛΗΡΩΜΗ ΜΕΤΡΗΤΟΙΣ». Διαφορετικά, να υπολογίζει και να εμφανίζει το πλήθος των απαιτούμενων για την εξόφληση δόσεων, όταν η εξόφληση γίνεται με άτοκες μηνιαίες δόσεις, ως εξής: Τον πρώτο μήνα η δόση θα είναι 20 ευρώ και κάθε επόμενο μήνα θα αυξάνεται κατά 5 ευρώ, μέχρι να εξοφληθεί το συνολικό ποσό.

Μονάδες 6

Γ3. Να υπολογίζει και να εμφανίζει τον συνολικό αριθμό των τεμαχίων με τιμή τεμαχίου μεγαλύτερη των 10 ευρώ.

Μονάδες 5

Γ4. Να υπολογίζει και να εμφανίζει τον συνολικό αριθμό των τεμαχίων με τη μέγιστη τιμή τεμαχίου.

Μονάδες 6

ΘΕΜΑ Δ

Μια εταιρεία Πληροφορικής καταγράφει, για δέκα ιστότοπους, τον αριθμό των επισκέψεων που δέχεται ο καθένας, κάθε μέρα, για τέσσερις εβδομάδες.

Να αναπτύξετε αλγόριθμο, ο οποίος:

Δ1. Για καθένα από τους ιστότοπους να διαβάζει το όνομά του και τον αριθμό των επισκέψεων που δέχθηκε ο ιστότοπος για καθεμιά ημέρα.
Δεν απαιτείται έλεγχος εγκυρότητας τιμών.

Μονάδες 2

Δ2. Να εμφανίζει το όνομα κάθε ιστοτόπου και τον συνολικό αριθμό των επισκέψεων που δέχθηκε αυτός στο διάστημα των τεσσάρων εβδομάδων.

Μονάδες 3

Δ3. Να εμφανίζει τα ονόματα των ιστοτόπων που κάθε μέρα στο διάστημα των τεσσάρων εβδομάδων δέχθηκαν περισσότερες από 500 επισκέψεις.

Αν δεν υπάρχουν τέτοιοι ιστότοποι, να εμφανίζει κατάλληλο μήνυμα.

Μονάδες 6

Δ4. Να διαβάζει το όνομα ενός ιστοτόπου. Αν το όνομα αυτό δεν είναι ένα από τα δέκα ονόματα που έχουν δοθεί, να το ξαναζητά, μέχρι να δοθεί ένα από αυτά τα ονόματα. Να εμφανίζει τους αριθμούς των εβδομάδων (1-4) κατά τη διάρκεια των οποίων ο συνολικός (εβδομαδιαίος) αριθμός επισκέψεων στον ιστότοπο αυτό είχε τη μέγιστη τιμή.

Μονάδες 9

ΘΕΜΑΤΑ 2015

ΘΕΜΑ Α

Α1. Να γράψετε στο τετράδιό σας τον αριθμό καθεμίας από τις παρακάτω προτάσεις 1-5 και , δίπλα , τη λέξη **ΣΩΣΤΟ** , αν η πρόταση είναι σωστή, ή τη λέξη **ΛΑΘΟΣ** , αν η πρόταση είναι λανθασμένη.

1. Η επαναληπτικότητα των διαδικασιών είναι ένας από τους λόγους ανάθεσης της επί λυσης ενός προβλήματος σε υπολογιστή.

(μονάδες 2)

2. Ο βρόχος **Για κ από 5 μέχρι 5** εκτελείται μία φορά.

(μονάδες 2)

3. Δεν υπάρχουν δομές δεδομένων δευτερεύουσας μνήμης

(μονάδες 2)

4. Ένας από τους παράγοντες από τους οποίους εξαρτάται η επιλογή της καταλληλότερης γλώσσας προγραμματισμού για την ανάπτυξη μιας εφαρμογής είναι το είδος της εφαρμογής

(μονάδες 2)

5. Ένα υποπρόγραμμα μπορεί να καλείται μόνο από το κύριο πρόγραμμα.

(μονάδες 2)

Μονάδες 10

Α2.α. Να αναφέρετε ονομαστικά τις κατηγορίες στις οποίες διακρίνονται τα προβλήματα με κριτήριο το είδος της επίλυσης που επιζητούν.

(μονάδες 3)

β. Έστω τα παρακάτω επιλύσιμα προβλήματα:

1. Δίδεται ένας ακέραιος αριθμός Ν και ζητείται ποια είναι η παραγοντοποίηση του Ν με το μεγαλύτερο πλήθος παραγόντων.

2. Δίδεται ένας ακέραιος αριθμός Ν και το πρόβλημα που τίθεται είναι αν ο Ν είναι άρτιος.

3. Δίδεται ένας ακέραιος αριθμός Ν και ζητείται να βρεθεί πόσες διαφορετικές παραγοντοποιήσεις του Ν υπάρχουν.

Για καθένα από τα προβλήματα αυτά, να γράψετε στο τετράδιό σας τον αριθμό του (1, 2 ή 3) και δίπλα την κατηγορία στην οποία ανήκει με κριτήριο το είδος της επίλυσης που επιζητεί. (μονάδες 3)

Μονάδες 6

Α3. α. Πόσοι δείκτες απαιτούνται για την υλοποίηση μιας ουράς με μονοδιάστατο πίνακα (μονάδες 2) και τι δείχνει ο καθένας; (μονάδες 2)

β. Ποιος δείκτης της ουράς μεταβάλλεται κατά τη λειτουργία της εξαγωγής; (μονάδες 2)

Μονάδες 6

Α4.α. Δίνονται οι παρακάτω εντολές:

λ←λ+1

λ←λ-2

λ←λ+3

Να γράψετε στο τετράδιό σας μία εντολή εκχώρησης που παράγει το ίδιο αποτέλεσμα.

(μονάδες 3)

β. Δίνονται τα τμήματα αλγορίθμου Ι και ΙΙ :

Ι	ΙΙ
Αν Χ>Υ και Υ≠1 τότε **Ζ← Χ/(Υ-1)** **Εμφάνισε Ζ** **αλλιώς_αν Χ>Υ και Υ=1 τότε** **Ζ←Υ/Χ** **Εμφάνισε Ζ** **Τέλος_αν**	**Αν ..... τότε** **Αν ..... τότε** **.....** **αλλιώς** **.....** **Τέλος_αν** **.....** **Τέλος_αν**

Να γράψετε στο τετράδιό σας το τμήμα αλγορίθμου ΙΙ με συμπληρωμένα τα κενά, ώστε να παράγει το ίδιο αποτέλεσμα με το τμήμα αλγορίθμου Ι.

(μονάδες 5)

Μονάδες 8

Α5.α. Δίνονται οι παρακάτω προτάσεις σε φυσική γλώσσα:

1. Αύξησε το Χ κατά 2.
2. Εκχώρησε στο Υ τον μέσο όρο των Κ, Λ, Μ.
3. Το τελευταίο ψηφίο του Α είναι 5.
4. Ο Β είναι διψήφιος.

Να θεωρήσετε ότι οι Α και Β είναι θετικοί ακέραιοι. Να γράψετε στο τετράδιό σας τον αριθμό της κάθε πρότασης και δίπλα την κωδικοποίησή της σε ΓΛΩΣΣΑ. (μονάδες 4)

β. Δίνεται το παρακάτω τμήμα αλγορίθμου:

```
Διάβασε Χ
Αν Χ>15 τότε
        Γράψε 1
αλλιώς _αν Χ>23 τότε
        Γράψε 2
αλλιώς
        Γράψε 3
Τέλος_αν
```

Μια εντολή εξόδου στο παραπάνω τμήμα δεν πρόκειται να εκτελεστεί, όποια και αν είναι η τιμή του Χ.

1. Ποια είναι η εντολή αυτή; (μονάδες 2)
2. Να γράψετε τις εντολές εξόδου που είναι δυνατόν να εκτελεστούν και, δίπλα σε κάθε μία από αυτές, το διάστημα τιμών του Χ για το οποίο θα εκτελεστεί η εντολή.(μονάδες 4)

Μονάδες 10

ΘΕΜΑ Β

Β1.

Δίνεται το παρακάτω τμήμα αλγορίθμου, όπου η μεταβλητή x έχει θετική ακέραια τιμή:

```
Αν x>1τότε
   y←x
   Αρχή_επανάληψης
        y←y-2
        Εμφάνισε y
   Μέχρις_ότου y≤0
```

Τέλος_αν

α. Να σχεδιάσετε στο τετράδιό σας το ισοδύναμο διάγραμμα ροής.(μονάδες 6)

β. Να ξαναγράψετε το τμήμα αυτό στο τετράδιό σας, χρησιμοποιώντας την εντολή **Για** αντί της εντολής **Μέχρις_ότου.** (μονάδες 8)

Μονάδες 14

Β2. Το παρακάτω ημιτελές τμήμα αλγορίθμου εισάγει αριθμητικές τιμές σε πίνακα 100 θέσεων ώστε:
α. οι τιμές να είναι διαφορετικές μεταξύ τους,
β. οι τιμές να εισάγονται σε αύξουσα σειρά.
Εάν κάποια εισαγόμενη τιμή δεν ικανοποιεί τις συνθήκες (α) και (β), επανεισάγεται.

```
Διάβασε Π[(1)...]
Για i από (2)... μέχρι (3)...
    Αρχή_επανάληψης
        Διάβασε Π[i]
    Μέχρις_ότου Π[(4)...] (5)... Π[(6)...]
Τέλος_επανάληψης
```

Να γράψετε στο τετράδιό σας τους αριθμούς (1) έως (6), που αντιστοιχούν στα κενά του αλγορίθμου και, δίπλα σε κάθε αριθμό, ό,τι πρέπει να συμπληρωθεί, ώστε το τμήμα αλγορίθμου να επιτελεί τη λειτουργία που περιγράφεται.

Μονάδες 6

ΘΕΜΑ Γ

Μία εταιρεία μεταφοράς δεμάτων διαθέτει δύο αποθήκες, Α και Β, στο αεροδρόμιο. Κατά την παραλαβή δεμάτων, κάθε δέμα τοποθετείται στην αποθήκη που έχει εκείνη τη στιγμή τον περισσότερο ελεύθερο χώρο. Αν ο ελεύθερος χώρος της αποθήκης Α είναι ίσος με τον ελεύθερο χώρο της αποθήκης Β, το δέμα τοποθετείται στην αποθήκη Α. Όταν όμως το δέμα δεν χωρά σε καμία από τις δύο αποθήκες, προωθείται στις κεντρικές εγκαταστάσεις της εταιρείας, που βρίσκονται εκτός αεροδρομίου.

Γ1. Να κατασκευάσετε πρόγραμμα που:

- **α.** Να περιλαμβάνει κατάλληλο τμήμα δηλώσεων. (μονάδες 2)
- **β.** Να διαβάζει τα μεγέθη ελεύθερου χώρου των αποθηκών Α και Β. (μονάδες 2)
- **γ.** Να διαβάζει το μέγεθος κάθε εισερχόμενου δέματος και να εμφανίζει το όνομα της αποθήκης (Α ή Β) στην οποία θα τοποθετηθεί αυτό ή να εμφανίζει το μήνυμα «Προώθηση», όταν το δέμα δεν χωρά σε καμία από τις αποθήκες Α ή Β. Η διαδικασία παραλαβής τερματίζεται, όταν εισαχθεί ως μέγεθος δέματος η τιμή 0.(μονάδες 6)
- **δ.** Στη συνέχεια, να καλεί υποπρόγραμμα, το οποίο να βρίσκει και να εμφανίζει το όνομα της αποθήκης (Α ή Β) στην οποία τοποθετήθηκαν τα περισσότερα δέματα, ή το μήνυμα «Ισάριθμα» σε περίπτωση που στις δύο αποθήκες Α και Β τοποθετήθηκαν ισάριθμα δέματα, ή το μήνυμα «Καμία αποθήκευση στο αεροδρόμιο», αν κανένα δέμα δεν τοποθετήθηκε σε οποιαδήποτε από τις αποθήκες Α ή Β. (μονάδες 2)

Μονάδες 12

Γ2. Να κατασκευάσετε το υποπρόγραμμα που περιγράφεται στο ερώτημα **Γ1.δ.**

Μονάδες 8

ΘΕΜΑ Δ

Ένας διαγωνισμός τραγουδιού διεξάγεται σε δύο φάσεις.
Στην πρώτη φάση γίνεται ακρόαση των 45 τραγουδιών που διαγωνίζονται και κάθε μέλος της επταμελούς κριτικής επιτροπής βαθμολογεί το κάθε τραγούδι με βαθμό από 1 έως 10.
Στη δεύτερη φάση προκρίνεται κάθε τραγούδι που συγκέντρωσε συνολική βαθμολογία μεγαλύτερη του 50 και το οποίο όλοι οι κριτές έχουν βαθμολογήσει τουλάχιστον με 5.

Να γραφεί αλγόριθμος , ο οποίος:

Δ1. Για κάθε τραγούδι να διαβάζει τον τίτλο του και τον βαθμό που έδωσε κάθε κριτής. Δεν απαιτείται έλεγχος εγκυρότητας.

Μονάδες 3

Δ2. Να υπολογίζει και να εμφανίζει τη συνολική βαθμολογία του κάθε τραγουδιού, η οποία προκύπτει ως το άθροισμα των βαθμών όλων των κριτών.

Μονάδες 2

Δ3. Να βρίσκει και να εμφανίζει τους τίτλους των τραγουδιών που προκρίνονται στη δεύτερη φάση του διαγωνισμού. Αν κανένα τραγούδι δεν προκρίνεται στη δεύτερη φάση, να εμφανίζει κατάλληλο μήνυμα.

Μονάδες 6

Δ4. Να βρίσκει και να εμφανίζει το πλήθος των κριτών που έδωσαν τον μέγιστο βαθμό τους σε ένα μόνο τραγούδι.

Μονάδες 9

ΘΕΜΑΤΑ 2016

ΘΕΜΑ Α

Α1. *Να γράψετε στο τετράδιό σας τον αριθμό καθεμιάς από τις παρακάτω προτάσεις **1-5** και, δίπλα, τη λέξη **ΣΩΣΤΟ**, αν η πρόταση είναι σωστή, ή τη λέξη **ΛΑΘΟΣ**, αν η πρόταση είναι λανθασμένη.*

1. Ο χρόνος εκτέλεσης κάθε αλγορίθμου εξαρτάται από τη Γλώσσα προγραμματισμού που θα χρησιμοποιηθεί.
2. Οι στατικές δομές στηρίζονται στην τεχνική της δυναμικής παραχώρησης μνήμης.
3. Σε μια δομή σύνθετης επιλογής, μετά από τις εντολές που βρίσκονται μεταξύ των λέξεων ΤΟΤΕ και ΑΛΛΙΩΣ, εκτελούνται οι εντολές που βρίσκονται μεταξύ των λέξεων ΑΛΛΙΩΣ και ΤΕΛΟΣ_ΑΝ.
4. Στο τμήμα δηλώσεων ενός προγράμματος, εκτός από τον τύπο ενός πίνακα, πρέπει να δηλώνεται και ο μεγαλύτερος αριθμός στοιχείων που μπορεί να έχει ο συγκεκριμένος πίνακας.
5. Το πρόγραμμα Συντάκτης εντοπίζει τα συντακτικά λάθη του προγράμματος.

Μονάδες 10

Α2. Δίδεται η λίστα:

α. Να περιγράψετε τη διαδικασία για την εισαγωγή του κόμβου με δεδομένα Ε ανάμεσα στον δεύτερο και τρίτο κόμβο της λίστας. (μονάδες 3)

β. Να περιγράψετε τη διαδικασία για τη διαγραφή του κόμβου με δεδομένα Κ από την αρχική λίστα. (μονάδες 3)

Μονάδες 6

Α3. **α.** Ποιες μεταβλητές ονομάζονται καθολικές; (μονάδες 2)

β. Η χρήση καθολικών μεταβλητών σε ένα πρόγραμμα καταστρατηγεί μία από τις βασικές αρχές του τμηματικού προγραμματισμού (ιδιότητες που πρέπει να διακρίνουν τα υποπρογράμματα). Να αναφέρετε ποια είναι αυτή η ιδιότητα και να εξηγήσετε γιατί καταστρατηγείται. (μονάδες 4)

Μονάδες 6

Α4. Έστω ο μονοδιάστατος πίνακας Α:

5	2	3	8	7	4	10	12

Να σχεδιάσετε τον πίνακα Β[6] μετά την εκτέλεση των παρακάτω εντολών:

1. Β[Α[1] – Α[3]] ← Α[5]
2. Β[Α[7] – Α[5]] ←Α[2] + Α[7]
3. Β[Α[6]] ←Α[4]
4. Β[Α[1] + Α[4] – Α[8]] ←Α[3] + Α[8]
5. Β[Α[8] DIV 2] ←Α[3] MOD 2
6. Β[Α[1] MOD Α[4]] ←Α[6] + 4

Μονάδες 12

Α5. Δίδεται πίνακας ΠΙΝ[7] με τις παρακάτω τιμές:

2	5	8	12	15	17	22

και το παρακάτω τμήμα αλγορίθμου

```
low ←1
high ←7
found ←ΨΕΥΔΗΣ
Όσο low ≤ high ΚΑΙ found=ΨΕΥΔΗΣ  επανάλαβε
        mid ← (low+high) DIV 2
        Εμφάνισε ΠΙΝ[mid]
        Αν  ΠΙΝ[mid] <  Χτότε
                low ←mid+1
        Αλλιώς_αν ΠΙΝ[mid] > Χ τότε
                high ←mid-1
        Αλλιώς
                found ← ΑΛΗΘΗΣ
        Τέλος_αν
Τέλος_Επανάληψης
```

Να γράψετε στο τετράδιό σας τις τιμές οι οποίες θα εμφανιστούν για: α) Χ=22 (μονάδες 3)

β) Χ=7 (μονάδες 3)

Μονάδες 6

ΘΕΜΑ Β

Β1. Ο αριθμός π εκφράζει το πηλίκο της περιμέτρου ενός κύκλου προς τη διάμετρό του. Η τιμή του μπορεί να υπολογιστεί, κατά προσέγγιση, από την παρακάτω παράσταση:

$$\pi = 4\left(\frac{1}{1}-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\cdots\right)$$

Ο υπολογισμός της τιμής της παράστασης, για 100 όρους του αθροίσματος, γίνεται από το παρακάτω τμήμα αλγορίθμου που περιλαμβάνει 5 κενά.

```
παρονομαστής ← (1) ...
Σ ← 0
πρόσημο ← 1
Για i από 1 μεχρι 100
    όρος ← 1/παρονομαστής
    όρος← (2) ... * πρόσημο
    (3) ... ← Σ + όρος
    πρόσημο ← πρόσημο * (...(4))
    παρονομαστής ← παρονομαστής + 2
Τέλος_Επανάληψης
π ← (5) ... * Σ
```

Να γράψετε στο τετράδιό σας τους αριθμούς 1 έως 5, που αντιστοιχούν στα κενά του αλγορίθμου, και, δίπλα σε κάθε αριθμό, ό,τι πρέπει να συμπληρωθεί, ώστε ο αλγόριθμος να υπολογίζει την τιμή του π όπως περιγράφηκε.

Μονάδες 10

Β2. Κατά την είσοδό τους σε μια τράπεζα οι πελάτες παίρνουν διαδοχικούς αριθμούς προτεραιότητας 1, 2, 3… που καθορίζουν τη σειρά τους στην ουρά του μοναδικού ταμείου.

Κάθε 2 λεπτά της ώρας προσέρχεται ένας νέος πελάτης και προστίθεται στην ουρά. Ο ταμίας εξυπηρετεί κάθε φορά τον πρώτο πελάτη στην ουρά και η εξυπηρέτησή του διαρκεί 3 λεπτά ακριβώς. Μετά την εξυπηρέτησή του ο πελάτης αποχωρεί από την ουρά.

Κατά την αρχή της διαδικασίας (χρόνος 0) στην ουρά υπάρχει μόνο ο πελάτης με αριθμό προτεραιότητας 1.

Να γράψετε διαδοχικά, σε ξεχωριστές γραμμές, με τη σωστή σειρά, τους αριθμούς προτεραιότητας των πελατών που βρίσκονται στην ουρά του ταμείου αμέσως μετά το $1^ο$, $2^ο$, $3^ο$, $4^ο$, $5^ο$ και $6^ο$ λεπτό.

Μονάδες 10

ΘΕΜΑ Γ

Μία εταιρεία πληροφορικής προσφέρει υπολογιστές σε τιμές οι οποίες μειώνονται ανάλογα με την ποσότητα της παραγγελίας, όπως φαίνεται στον παρακάτω πίνακα:

ΠΟΣΟΤΗΤΑ	ΤΙΜΗ ΜΟΝΑΔΑΣ
1-50	580
51-100	520
101-200	470
Πάνω από 200	440

Να κατασκευάσετε πρόγραμμα το οποίο:

Γ1. Να περιλαμβάνει κατάλληλο τμήμα δηλώσεων.

Μονάδες 2

Γ2. Να διαβάζει τον αριθμό υπολογιστών που έχει προς πώληση (απόθεμα), ελέγχοντας ότι δίνεται θετικός αριθμός

Μονάδες 2

Γ3. Για κάθε παραγγελία, να διαβάζει την απαιτούμενη ποσότητα και, εφόσον το απόθεμα επαρκεί για την κάλυψη της ποσότητας να εκτελεί την παραγγελία με την ποσότητα που ζητήθηκε. Αν το απόθεμα δεν επαρκεί, διατίθεται στον πελάτη το διαθέσιμο απόθεμα. Η εισαγωγή παραγγελιών τερματίζεται, όταν εξαντληθεί το απόθεμα.

Μονάδες 6

Για κάθε παραγγελία να εμφανίζει:

Γ4. το κόστος της παραγγελίας **Μονάδες 4**

Γ5. το επιπλέον ποσό που θα κόστιζε η παραγγελία, εάν ο υπολογισμός γινόταν κλιμακωτά με τις τιμές που φαίνονται στον πίνακα.

Μονάδες 6

ΘΕΜΑ Δ

Το Πανελλήνιο Σχολικό Δίκτυο παρέχει πρόσβαση στο Διαδίκτυο (Ίντερνετ) σε 150.000 μαθητές και διατηρεί τα στοιχεία τους, καθώς και στατιστικά στοιχεία, σχετικά με την πρόσβασή τους στο Διαδίκτυο.

Να κατασκευάσετε πρόγραμμα το οποίο:

Δ1. Να περιλαμβάνει κατάλληλο τμήμα δηλώσεων.

Μονάδες 2

Δ2. Για κάθε μαθητή να διαβάζει:

α) τον αλφαριθμητικό κωδικό του και να τον καταχωρίζει σε μονοδιάστατο πίνακα με όνομα ΚΩΔ

β) το φύλο του, «Α» αν είναι αγόρι και «Κ» αν είναι κορίτσι, και να το καταχωρίζει σε μονοδιάστατο πίνακα με όνομα Φ

γ) τον συνολικό χρόνο πρόσβασής του στο Διαδίκτυο ανά μήνα, για ένα έτος, και να τον καταχωρίζει σε δισδιάστατο πίνακα ΧΡ.

Μονάδες 3

Δ3. Να υπολογίζει και να καταχωρίζει σε πίνακα ΣΧ το συνολικό ετήσιο χρόνο πρόσβασης κάθε μαθητή.

Μονάδες 3

Δ4. Να εμφανίζει τον κωδικό του αγοριού με το μεγαλύτερο συνολικό χρόνο πρόσβασης και, στη συνέχεια, τον κωδικό του κοριτσιού με το μεγαλύτερο συνολικό χρόνο πρόσβασης, καλώντας τη συνάρτηση ΘΕΣΗ_ΜΑΧ, που περιγράφεται στο ερώτημα Δ5, μία φορά για τα αγόρια και μία για τα κορίτσια.

Μονάδες 4

Δ5. Να αναπτύξετε συνάρτηση ΘΕΣΗ_ΜΑΧ η οποία:

α) να δέχεται ως παραμέτρους: τον πίνακα του φύλου, τον πίνακα του συνολικού ετήσιου χρόνου πρόσβασης των μαθητών και τον χαρακτήρα

«Α» ή «Κ» που αντιστοιχεί στο φύλο (μονάδες 2)

β) να βρίσκει τη θέση της μέγιστης τιμής του ετήσιου χρόνου πρόσβασης αγοριών ή κοριτσιών, ανάλογα με την τιμή «Α» ή «Κ» του φύλου (μονάδες 4)

γ) να επιστρέφει τη θέση της μέγιστης τιμής (μονάδες 2)

Μονάδες 8

(Σημείωση: Δεν απαιτείται έλεγχος εγκυρότητας. Να θεωρήσετε ότι όλες οι εισαγωγές γίνονται σωστά και όλες οι συνολικές τιμές χρόνου πρόσβασης είναι μοναδικές).