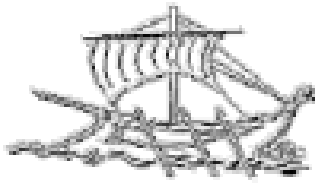


ΤΕΙ ΠΕΙΡΑΙΑ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Π.Μ.Σ. "ΕΦΑΡΜΟΣΜΕΝΑ ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ"

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Υβριδική μάθηση στην αλγοριθμική σκέψη

Μοσχόπουλος Δ. Παναγιώτης

Εισηγήτρια: Αναστασία Βελώνη, Καθηγήτρια

ΑΘΗΝΑ
ΜΑΡΤΙΟΣ 2018

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Υβριδική μάθηση στην αλγοριθμική σκέψη

Μοσχόπουλος Δ. Παναγιώτης
Α.Μ. ais0114

Εισηγήτρια: Αναστασία Βελώνη, Καθηγήτρια

Εξεταστική Επιτροπή:

Αναστασία Βελώνη, Καθηγήτρια

Ιωάννης Έλληνας, Καθηγητής

Πάρις Μαστοροκώστας, Καθηγητής

Ημερομηνία εξέτασης

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Ο/Η κάτωθι υπογεγραμμένος Μοσχόπουλος Παναγιώτης, του Δημητρίου, με αριθμό μητρώου ais0114 φοιτητής του Τμήματος Μηχανικών Η/Υ Συστημάτων Τ.Ε. του Α.Ε.Ι. Πειραιά Τ.Τ. πριν αναλάβω την εκπόνηση της Πτυχιακής Εργασίας μου, δηλώνω ότι ενημερώθηκα για τα παρακάτω:

«Η Πτυχιακή Εργασία (Π.Ε.) αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο του συγγραφέα, όσο και του Ιδρύματος και θα πρέπει να έχει μοναδικό χαρακτήρα και πρωτότυπο περιεχόμενο.

Απαγορεύεται αυστηρά οποιοδήποτε κομμάτι κειμένου της να εμφανίζεται αυτούσιο ή μεταφρασμένο από κάποια άλλη δημοσιευμένη πηγή. Κάθε τέτοια πράξη αποτελεί προϊόν λογοκλοπής και εγείρει θέμα Ηθικής Τάξης για τα πνευματικά δικαιώματα του άλλου συγγραφέα. Αποκλειστικός υπεύθυνος είναι ο συγγραφέας της Π.Ε., ο οποίος φέρει και την ευθύνη των συνεπειών, ποινικών και άλλων, αυτής της πράξης.

Πέραν των όποιων ποινικών ευθυνών του συγγραφέα σε περίπτωση που το Ίδρυμα του έχει απονείμει Πτυχίο, αυτό ανακαλείται με απόφαση της Συνέλευσης του Τμήματος. Η Συνέλευση του Τμήματος με νέα απόφασης της, μετά από αίτηση του ενδιαφερόμενου, του αναθέτει εκ νέου την εκπόνηση της Π.Ε. με άλλο θέμα και διαφορετικό επιβλέποντα καθηγητή. Η εκπόνηση της εν λόγω Π.Ε. πρέπει να ολοκληρωθεί εντός τουλάχιστον ενός ημερολογιακού 6μήνου από την ημερομηνία ανάθεσης της. Κατά τα λοιπά εφαρμόζονται τα προβλεπόμενα στο άρθρο 18, παρ. 5 του ισχύοντος Εσωτερικού Κανονισμού.»

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα πτυχιακή εργασία ολοκληρώθηκε μετά από επίμονες προσπάθειες, σε ένα ενδιαφέρον γνωστικό αντικείμενο, όπως αυτό της αλγοριθμικής σκέψης. Την προσπάθειά μου αυτή υποστήριξε η επιβλέπουσα καθηγήτρια Βελώνη Αναστασία, την οποία θα ήθελα να ευχαριστήσω.

Επίσης θα ήθελα να ευχαριστήσω την σύζυγό μου Παπαδοπούλου Παναγιώτα για τις ατελείωτες ώρες προσμονής, την υπομονή και την σιωπηρή υποστήριξη που έδειξε για την ολοκλήρωση αυτής της εργασίας.

ΠΕΡΙΛΗΨΗ

Η εφαρμογή της μικτής μάθησης στα σχολεία είναι ο πιο αποτελεσματικός και ευέλικτος τρόπος μάθησης καθώς συνδυάζει κλασικές μεθόδους διδασκαλίας και εξ΄ αποστάσεως διαδικτυακές μεθόδους διδασκαλίας. Η παρούσα πτυχιακή εργασία σκοπεύει στην βαθύτερη κατανόηση στην μεθοδολογία των αλγοριθμικών δομών και των βασικών αλγορίθμων συνδυάζοντας την εξ' αποστάσεως ατομική μάθηση με την παραδοσιακή εκπαιδευτική προσέγγιση. Απευθύνεται σε όλους εκείνους που θα ήθελαν να κάνουν μία αρχή πάνω στην αλγοριθμική σκέψη και ειδικά στους μαθητές της τρίτης Λυκείου του προσανατολισμού σπουδών οικονομίας και πληροφορικής που δίνουν πανελλαδικές εξετάσεις στο μάθημα «Ανάπτυξη εφαρμογών σε προγραμματιστικό περιβάλλον».

ΕΠΙΣΤΗΜΟΝΙΚΗ ΠΕΡΙΟΧΗ: Αλγοριθμική σκέψη

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Υβριδική μάθηση, μικτή μάθηση, Αλγόριθμοι, Δομή ακολουθίας, Δομή επιλογής, Δομή επανάληψης.

ABSTRACT

The implementation of mixed learning in schools is the most effective and flexible way of learning as it combines classical teaching methods and web-based teaching methods. This present thesis aims at a deeper understanding of the methodology of algorithmic structures and basic algorithms, by combining distance learning with the traditional educational approach. It is aimed at all those who would like to make a start on algorithmic thinking and especially students in High School of Economics Administration and Informatics.

SCIENTIFIC AREA: Algorithmic thought

KEY WORDS: Hybrid learning, mixed learning, Algorithms, Sequence structure, Selection structure, Repetition structure.

Περιεχόμενα

Πίνακας Σχημάτων/Εικόνων

Πίνακες

ΚΕΦΑΛΑΙΟ 1

1. ΕΙΣΑΓΩΓΗ

Σε αυτό το κεφάλαιο αναλύεται το αντικείμενο της πτυχιακής εργασίας και γίνεται μια λεπτομερέστερη περιγραφή των θέσεων που καταπιάνεται η συγκεκριμένη πτυχιακή εργασία

1.1. Περιγραφή του αντικειμένου της πτυχιακής εργασίας

Αντικείμενο της παρούσας πτυχιακής εργασίας είναι η εισαγωγή στη αλγοριθμική σκέψη και στην συνέχεια η ανάλυση και κατανόηση των δομών του δομημένου προγραμματισμού, χρησιμοποιώντας το μοντέλο της υβριδικής μάθησης. Στο μικτό αυτό μοντέλο, συνδυάζεται η πρόσωπο με πρόσωπο παραδοσιακή διδασκαλία σε έναν φυσικό χώρο όπως μια αίθουσα σχολείου, με την εξ΄ αποστάσεως διδασκαλία ως συνέχεια της προηγούμενης σε μια διαδικτυακή εικονική τάξη. Για την παραδοσιακή διδασκαλία έχουν γραφτεί ενότητες του παρόντος συγγράμματος που θα βοηθήσουν τον οποιοδήποτε εκπαιδευτικό να οργανώσει σε κατάλληλα τμήματα την διδασκαλία του και να διδάξει με συγκεκριμένα βήματα την αλγοριθμική σκέψη. Πέραν των γνώσεων στη αλγοριθμική σκέψη που μεταφέρονται στον μαθητή μέσα από μια συμβατική πρόσωπο με πρόσωπο διδασκαλία, ο μαθητής έχει την δυνατότητα να αυξήσει τις γνώσεις του πάνω στο συγκεκριμένο τμήμα της αλγοριθμικής σκέψης που τον ενδιαφέρει χρησιμοποιώντας την εικονική προσέγγιση μιας διαδικτυακής τάξης.

Για τον σκοπό αυτό έχει χρησιμοποιηθεί η ψηφιακή πλατφόρμα Schoology.com στην οποία υπάρχει η δυνατότητα της εγγραφής και παραλαβής κωδικού τάξης, έτσι ώστε ο ενδιαφερόμενος να έρθει σε επαφή με τα παρακάτω γνωστικά αντικείμενα:

- ✓ Ερωτήσεις σωστού λάθους
- ✓ Ερωτήσεις πολλαπλής επιλογής
- ✓ Ασκήσεις σε μορφή εικόνας και κειμένου.
- ✓ Βίντεο που αναλύουν την κάθε ενότητα.

Αρχικά αφού γίνεται μια ιστορική αναδρομή, αναλύεται η έννοια της υβριδικής μάθησης, τα διαφορετικά μοντέλα της καθώς και τα πλεονεκτήματά της. Έπειτα

αναλύονται σε βάθος όλες εκείνες οι έννοιες που χρειάζονται στο ξεκίνημα του προγραμματισμού, αλλά και βασικοί αλγόριθμοι όπως αυτός της εύρεσης μεγαλύτερου - μικρότερου στοιχείου, της σειριακής και δυαδικής αναζήτησης, της ταξινόμησης ευθείας ανταλλαγής (φυσαλίδα), της ταξινόμησης με επιλογή κ.α.

Το παρόν σύγγραμμα θα μπορούσε να χρησιμεύσει σε μαθητές λυκείου, αλλά και σε όποιον θα ήθελε να κάνει μια εισαγωγή στις βασικές αρχές της αλγοριθμικής σκέψης. Ο κώδικας που χρησιμοποιείται είναι μια ελληνική εκδοχή της Pascal, βασικής γλώσσας προγραμματισμού που εισήγαγε από τις πρώτες την έννοια του δομημένου προγραμματισμού.

1.2.Ενότητες

Το γνωστικό αντικείμενο χωρίστηκε στις εξής ενότητες:

- ✓ Δομή Ακολουθίας - Επιλογής
- ✓ Δομή επανάληψης
- ✓ Πίνακες
- ✓ Προγραμματιστικά περιβάλλοντα
- ✓ Υποπρογράμματα

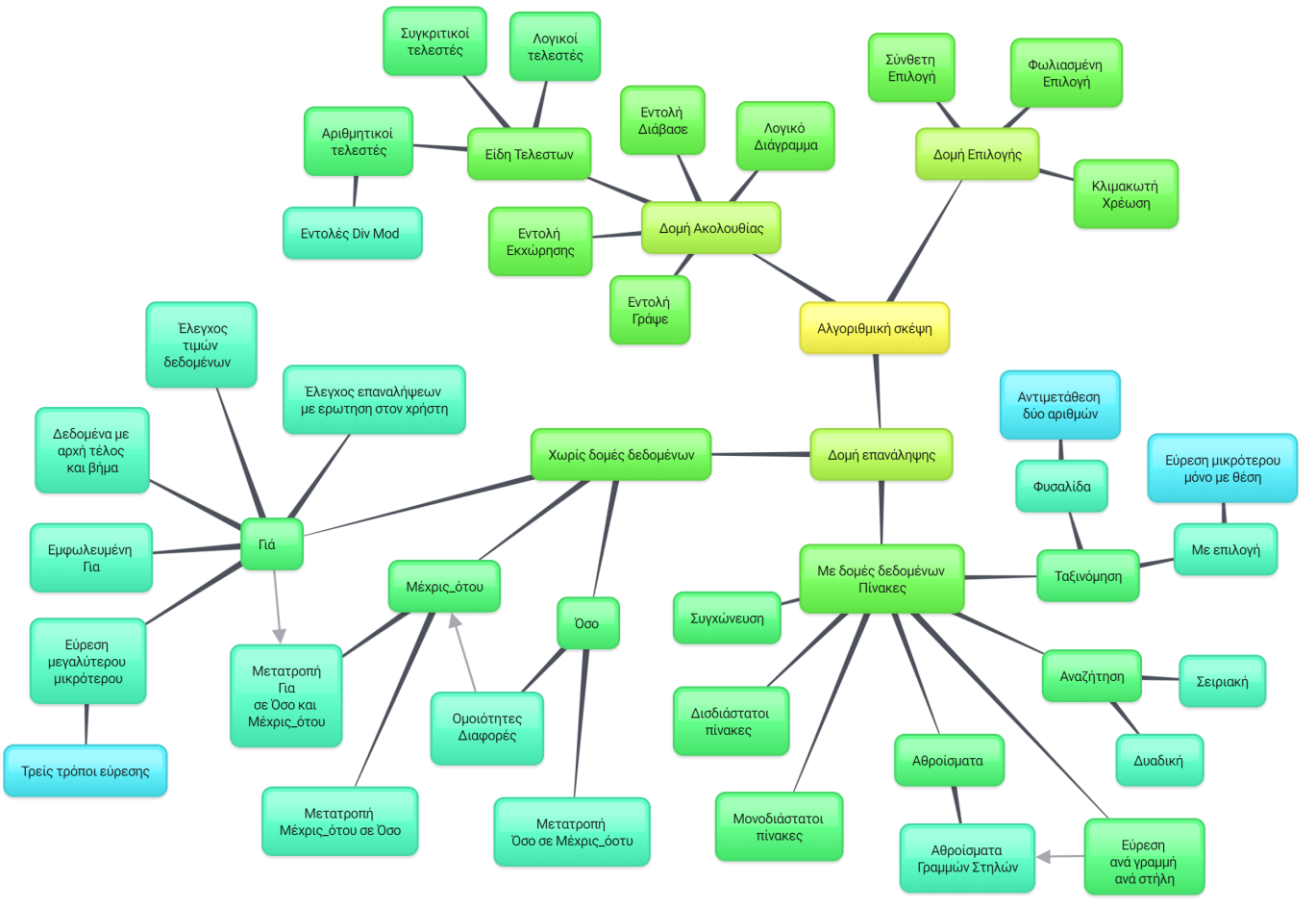
1.3.Βίντεο εκμάθησης

Έχουν δημιουργηθεί εκπαιδευτικά βίντεο με τους παρακάτω τίτλους:

1.1 Δομή Ακολουθίας
1.2 Δομή Επιλογής
1.3 Εμφωλευμένη Δομή Επιλογής - Σύνθετα Αν
1.4 Κλιμακωτή χρέωση
2.1.1 Δομή επανάληψης
2.1.2 Όσο vs Μέχρις_ότου
2.1.3 Για μέχρι
2.2 Εύρεση μεγαλύτερου – μικρότερου αριθμού
2.3 Μετατροπή Για σε Όσο και Μεχρις_οτου
2.4 Έλεγχος τιμών δεδομένων
2.5 Διπλή Επανάληψη
3.1 Τρόποι εισαγωγής Δεδομένων
3.2.1 Μονοδιάστατοι Πίνακες

1.4.Νοητικός χάρτης

Η δημιουργία των ενοτήτων της εργασίας και των βίντεο στηρίχτηκε στον παρακάτω νοητικό χάρτη αλγοριθμικής σκέψης, ο οποίος φτιάχτηκε online στον διαδικτυακό τόπο https://bubbl.us.



1.4-1 Νοητικός χάρτης αλγοριθμικής σκέψης

ΚΕΦΑΛΑΙΟ 2

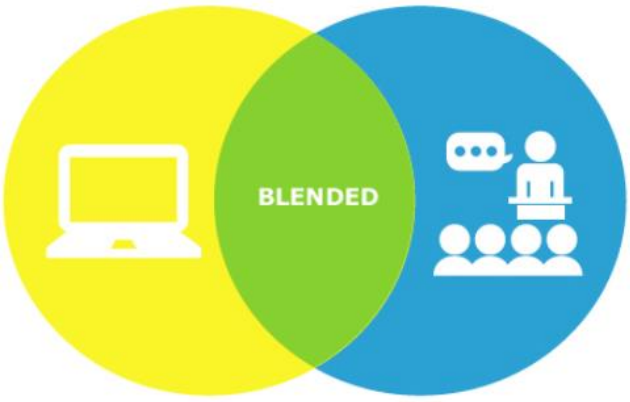
2. Υβριδική Μάθηση

2.1.Ιστορική αναδρομή

Η παρουσία του διαδικτύου έχει επηρεάσει όλους του τομείς τις ζωής μας. Με την ανάπτυξη του διαδικτύου που ολοένα και αντικαθιστά δραστηριότητες του φυσικού μας κόσμου, εμφανίστηκαν σταδιακά διαδικτυακά περιβάλλοντα τα οποία υποστήριζαν την ηλεκτρονική τάξη, δηλαδή έναν χώρο που για πρώτη φορά θα μπορούσε ο καθηγητής να δώσει ασκήσεις, οι μαθητές σε συγκεκριμένο χρονικό διάστημα να τις παραδώσουν και ο καθηγητής να τις διορθώσει σε δικό του προσωπικό χρόνο. Έτσι εξελίχθηκαν ασύγχρονες μορφές εκπαίδευσης σπάζοντας την παραδοσιακή λογική ότι μαθητές και εκπαιδευτικοί πρέπει αναγκαστικά να βρίσκονται στον ίδιο χώρο και χρόνο για να πραγματοποιηθεί η μαθησιακή διαδικασία. Οι σύγχρονοι μαθησιακοί σχεδιασμοί πλέον ενσωματώνουν δύο βασικές εκπαιδευτικές μορφές: την παραδοσιακή διδασκαλία στην αίθουσα, αλλά και την ηλεκτρονική εξ αποστάσεως μάθηση. Αποτέλεσμα των παραπάνω είναι ένα υβριδικό εκπαιδευτικό περιβάλλον το οποίο χρησιμοποιεί τις Τεχνολογίες Πληροφοριών και Επικοινωνιών (ΤΠΕ) στην εκπαιδευτική διαδικασία, δημιουργώντας ένα μικτό μοντέλο μάθησης.

2.2.Έννοια υβριδικής μάθησης.

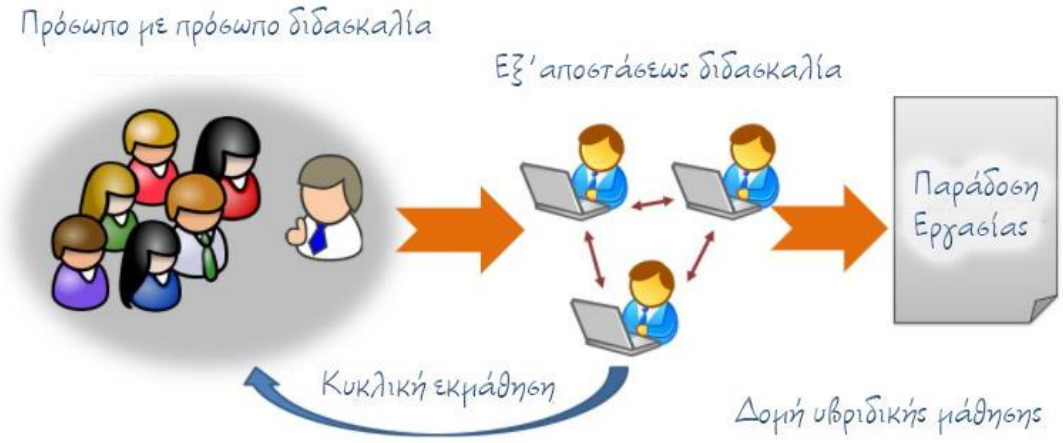
Ο όρος υβριδική - μικτή μάθηση (blended learning), αναφέρεται στη μάθηση που συνδυάζει την πρόσωπο με πρόσωπο διδασκαλία με τη μάθηση μέσω διαδικτυακών εξ΄ αποστάσεως δραστηριοτήτων, μειώνοντας ή και συμπληρώνοντας το χρόνο παρακολούθησης στη φυσική τάξη. Συνδυάζει δοκιμασμένες παραδοσιακές μεθόδους μάθησης με καινοτόμες μαθησιακές προσεγγίσεις, με αποτέλεσμα τη δημιουργία συνεργατικού και δυναμικού πλαισίου μάθησης. Είναι μία μίξη σύγχρονης και ασύγχρονης μάθησης,



2.2-1 Υβριδική Μάθηση

πρόσωπο με πρόσωπο διδακτικών μεθόδων και ανοικτών - εξ' αποστάσεως μορφών εκπαίδευσης, καθώς και ποικίλων τεχνολογικών και παραδοσιακών μέσων διδασκαλίας.

Σε ένα πρόγραμμα μικτής μάθησης ο εκπαιδευτής μπορεί να αναμείξει διαφορετικά είδη διδασκαλίας, διδακτικές πρακτικές και εκπαιδευτικές προσεγγίσεις, να χρησιμοποιήσει το διαδίκτυο για να συμπληρώσει μία ζωντανή διδασκαλία, ή να συνδυάσει τμήματα μίας δικτυακής εκπαίδευσης που δεν απαιτεί καθοδήγηση, με άλλα τμήματα εκπαίδευσης στα οποία είναι απαραίτητη η παρουσία και η καθοδήγηση του εκπαιδευτικού.



2.2-2 Κυκλική εκμάθηση

Η υβριδική διδασκαλία δεν χρησιμοποιεί απλά τις παραδοσιακές και τις εξ αποστάσεως δραστηριότητες, αλλά οργανώνει ένα αποτελεσματικό σύνολο δραστηριοτήτων στο οποίο εκμεταλλευόμαστε στο μέγιστο τα θετικά στοιχεία και των δύο προσεγγίσεων (Saliba, G., Rankine, L., Cortez, H. 2013).

Ενώ η συμβατική μάθηση επικεντρώνεται κυρίως στη μετάδοση πληροφοριών από τον εκπαιδευτικό στο εκπαιδευόμενο, η μικτή μάθηση απευθύνεται στο μεγαλύτερο μέρος της στον εκπαιδευόμενο αφού δεν περιορίζεται μόνο στο γνωστικό αντικείμενο, αλλά παράλληλα λαμβάνει υπόψιν τις ιδιαιτερότητες και την γενικότερη προσωπικότητα του μαθητή. Είναι μία μαθητοκεντρική προσέγγιση που συνδυάζει τη δυνατότητα κοινωνικοποίησης που παρέχουν οι συνθήκες της φυσικής τάξης με τις μαθησιακές δραστηριότητες που εμπλέκουν ενεργά το μαθητή και προσφέρονται σε ένα διαδικτυακό περιβάλλον.

Ένα ακόμα χαρακτηριστικό της υβριδικής μάθησης είναι η πολυδιάστατη επικοινωνία που υπάρχει σε αυτήν. Συγκεκριμένα επισημαίνεται αυξημένη αλληλεπίδραση ανάμεσα στο μαθητή και στον εκπαιδευτικό, μεταξύ των μαθητών, και ανάμεσα στους μαθητές και τις εξωτερικές πηγές γνώσεων.

Συνηθέστερα, οι εκπαιδευόμενοι ξεκινούν την εκπαιδευτική διαδικασία σε συμβατικές εκπαιδευτικές δομές όπως μια τάξη ενός σχολείου. Στη συνέχεια, δίνεται σε αυτούς η ελεγχόμενη δυνατότητα πρόσβασης σε επιπλέον υλικό, με τη χρήση των τεχνολογιών του διαδικτύου. Με αυτό τον τρόπο προσφέρεται αυτονομία, καθώς οι περιοριστικοί παράγοντες της φυσικής παρουσίας σε συγκεκριμένο χώρο και χρόνο δεν υφίστανται, αφού οι εκπαιδευόμενοι συνεχίζουν την διαδικασία της μάθησης μέσω μιας ψηφιακής πλατφόρμας που αναπαριστά μια εικονική τάξη. Βασική προϋπόθεση για τα παραπάνω είναι η εξοικείωση των χρηστών με τα ψηφιακά περιβάλλοντα μάθησης.

2.3.Μοντέλα υβριδικής-μικτής μάθησης

Πλέον χρησιμοποιούνται αρκετά μοντέλα μικτής μάθησης, καθώς διαρκείς αλλαγές εξακολουθούν να αναπτύσσονται αντίστοιχα με την εξέλιξη των τεχνολογικών μέσων και των παιδαγωγικών μεθόδων. Αυτά τα μοντέλα έχουν διαφοροποιημένες αρκετές παραμέτρους, όπως ο ρόλος των καθηγητών, ο φυσικός χώρος που πραγματοποιούνται καθώς και σειρά παρακολούθησης των ενοτήτων. Τα μοντέλα αυτά κατατάσσονται σε έξι κατηγορίες, όπως παρακάτω (Michael B. Horn and Heather Staker 2011).

2.3.1. Το πρόσωπο με πρόσωπο μοντέλο.

Θεωρείται το πιο κοντινό σε μια παραδοσιακή σχολική τάξη. Σύμφωνα αυτή την προσέγγιση, η εισαγωγή στην ηλεκτρονική διδασκαλία γίνεται μέσα στην τάξη, από έναν καθηγητή που με την χρησιμοποίηση πολυμέσων δείχνει στους μαθητές διαδικτυακό υλικό και συχνά διακόπτει την διαδικασία δίνοντας επιμέρους πληροφορίες σχετικά με το αντικείμενο διδασκαλίας. Το μοντέλο αυτό δίνει την δυνατότητα στους μαθητές να αυξήσουν την πρόοδο τους χρησιμοποιώντας τεχνολογικά μέσα που θα βρουν μέσα στην τάξη.

2.3.2. Το κυκλικό μοντέλο.

Οι μαθητές εναλλάσσουν σε συγκεκριμένα χρονικά διαστήματα την προσωπική διαδικτυακή μάθηση που πραγματοποιείται με τον δικό τους προσωπικό ρυθμό εκμάθησης και την παραδοσιακή πρόσωπο με πρόσωπο εκμάθηση που πραγματοποιείται ομαδικά στην τάξη με τον καθηγητή τους. Τα χρονικά διαστήματα εναλλαγής μπορεί να ποικίλλουν ανάλογα με τις ανάγκες των εκάστοτε μαθητών.

2.3.3. Το ευέλικτο μοντέλο.

Σε αυτή την μορφή, το εκπαιδευτικό υλικό υπάρχει σχεδόν εξ' ολοκλήρου μέσω μιας διαδικτυακής πλατφόρμας. Η μάθηση εδώ είναι κυρίως αυτό - καθοδηγούμενη. Οι μαθητές μαθαίνουν να εξασκούν καινούριες έννοιες σε ένα ψηφιακό περιβάλλον, με την περιστασιακή βοήθεια ενός εξ΄ αποστάσεως καθηγητή.

2.3.4. Το μοντέλο του διαδικτυακού εργαστηρίου.

Στο μοντέλο αυτό, οι μαθητές μαθαίνουν τον περισσότερο καιρό μέσω του διαδικτύου, έχοντας εξ΄ αποστάσεως βοήθεια από κάποιον καθηγητή. Τελικά όμως μεταφέρονται σε μια αίθουσα υπολογιστών για να ολοκληρώσουν την παρακολούθηση των μαθημάτων τους, που επιτηρείται από άτομα που δεν είναι αναγκαστικά γνώστες του αντικειμένου εκμάθησης. Κάτι τέτοιο, επιτρέπει σε σχολεία με ελλείψεις σε καθηγητές να προσφέρουν ανάλογα μαθήματα. Επίσης επιτρέπει στους μαθητές να δουλέψουν με τον δικό τους ρυθμό, χωρίς αυτό να επηρεάζει άμεσα τους άλλους μαθητές.

2.3.5. Το αυτό-αναμειγνυόμενο μοντέλο.

Συνδυάζει την προσωπική διδασκαλία με την ηλεκτρονική μάθηση. Αρκετά δημοφιλές στα λύκεια και γυμνάσια της Αμερικής, το μοντέλο αυτό δίνει στους μαθητές την δυνατότητα να επιλέξουν συμπληρωματικά μαθήματα πέραν των ωρών διδασκαλίας στο σχολείο τους. Έτσι οι μαθητές θα μπορούν επιπλέον να ενισχύσουν την μάθησή τους μέσα από διαδικτυακά εξ' αποστάσεως μαθήματα, ακόμα και σε πιο εξειδικευμένα πεδία γνώσης.

2.3.6. Το μοντέλο διαδικτυακού οδηγού

Αποτελεί μία μορφή μικτής μάθησης στην οποία οι μαθητές δουλεύουν απομακρυσμένα μέσω μιας διαδικτυακής πλατφόρμας και επικοινωνούν με

ηλεκτρονικά μηνύματα όποτε αυτό χρειάζεται με τον καθηγητή τους. Κάποιες φορές δε μπορεί να χρειάζεται και μια διαδικτυακή παρακολούθηση ή ραντεβού με τον επιβλέποντα καθηγητή. Αυτό το μοντέλο μικτής μάθησης είναι ιδανικό για μαθητές που χρειάζονται ευελιξία στην παρακολούθηση των μαθημάτων τους λόγω άλλων υποχρεώσεων (ενήλικές).

2.3.7. Αυτό-αναμειγνυόμενο μοντέλο και αλγοριθμική σκέψη.

Στο συγκεκριμένο παράδειγμα της εισαγωγής στην αλγοριθμική σκέψη χρησιμοποιήθηκε το αυτό-αναμειγνυόμενο μοντέλο ως το καταλληλότερο από τα παραπάνω. Έτσι πέραν των γνώσεων στη αλγοριθμική σκέψη που μεταφέρονται στον μαθητή μέσα από μια συμβατική πρόσωπο με πρόσωπο διδασκαλία, ο μαθητής έχει την δυνατότητα να αυξήσει τις γνώσεις του πάνω στο συγκεκριμένο τμήμα της αλγοριθμικής σκέψης που τον ενδιαφέρει.

2.4.Πλεονεκτήματα της υβριδικής μάθησης.

2.4.1. Φυσικότερος τρόπος εκμάθησης

Σύμφωνα με τον Elliot Masie οι άνθρωποι είμαστε εκ φύσεως «blended learners». Αποδίδουμε καλύτερα όταν εκπαιδευόμαστε με ποικίλους τρόπους και μέσα. Με άλλα λόγια, οι άνθρωποι μαθαίνουμε καλύτερα με τον συνδυασμό διαφορετικών μέσων και μεθόδων διδασκαλίας, ως συνέχεια της συνδυαστικής συμπεριφοράς που έχουμε για τον χειρισμό πραγματικών γεγονότων στην ζωή. Διαφαίνεται λοιπόν ότι η μικτή μάθηση προσφέρει περισσότερες διδακτικές και μαθησιακές επιλογές γεγονός που την κάνει πιο αποτελεσματική σε σχέση με τους απλούστερους τρόπους διδασκαλίας.

2.4.2. Ανεξαρτησία σε τόπο και χρόνο

Ένα πρόγραμμα εκπαίδευσης που πραγματοποιείται μόνο σε μια φυσική αίθουσα διδασκαλίας, επιτρέπει τη συμμετοχή μόνο σε όσους μπορούν να παρευρεθούν την συγκεκριμένη χρονική στιγμή, στον συγκεκριμένο φυσικό χώρο. Εν αντιθέσει ένα πρόγραμμα εκπαίδευσης σε εικονική τάξη δεν προσδιορίζει ούτε το χώρο ούτε το χρόνο μάθησης. Η συνδυαστική ηλεκτρονική μάθηση προσφέρει αυτονομία και παρέχει τη δυνατότητα της πρόσβασης σε γεωγραφικά απομακρυσμένους συμμετέχοντες, να μαθαίνουν οποτεδήποτε θέλουν εκείνοι, έχοντας πρόσβαση στην αναγκαία πληροφορία τη στιγμή που την χρειάζονται.

2.4.3. Μείωση χρόνου και κόστους εκμάθησης

Ένα εξ΄ ολοκλήρου διαδικτυακό πρόγραμμα είναι απαιτητικό σε μέσα, πηγές, χρόνο και αντίστοιχα χρηματικούς πόρους. Αντίθετα σε ένα υβριδικό – μικτό πρόγραμμα εκμάθησης, η φυσική παρουσία σε μία τάξη απαιτεί απλούστερα εργαλεία αυτό - καθοδηγούμενης ηλεκτρονικής μάθησης. Συνδυασμός γραπτών κειμένων, ηχογραφήσεων, βίντεο και παρουσιάσεων, δίνουν καλύτερα αποτελέσματα, σε μικρότερο χρόνο και με πολύ λιγότερα χρήματα. Κάνοντας χρήση διαδικτυακών εργαλείων σε συνδυασμό με τους ήδη γνωστούς τρόπους διδασκαλίας, μειώνεται η πολυπλοκότητα και ο χρόνος που χρειάζεται για να πραγματοποιηθεί ένα σύνθετο χρονοβόρο μαθησιακό αντικείμενο.

2.4.4. Ικανοποίηση διαφορετικών αναγκών

Οι μαθησιακές ανάγκες και τα ενδιαφέροντα κάθε εκπαιδευόμενου ποικίλλουν σε μικρότερο ή μεγαλύτερο βαθμό. Ο κλασικός τρόπος διδασκαλίας δεν αρκεί για να εκπληρωθεί αυτήν η διαφορετικότητα. Προκειμένου λοιπόν να ικανοποιηθούν οι απαιτήσεις και τα ενδιαφέροντα όλων των εκπαιδευόμενων, απαιτείται μια μικτή διδακτική προσέγγιση, που να περιλαμβάνει συνδυασμό μαθησιακών τεχνικών, ούτως ώστε ο κάθε ενδιαφερόμενος να έρθει σε επαφή με το κατάλληλο περιεχόμενο στην κατάλληλη μορφή την κατάλληλη χρονική στιγμή, με σκοπό την ικανοποίηση του σε ατομικό επίπεδο.

2.4.5. Κοινωνικοποίηση μαθητών

Οι εκπαιδευόμενοι, όπως αποδείχτηκε, προτιμούν την κοινωνική, αλληλεπιδραστική μάθηση έναντι της αποκλειστικά αυτό – καθοδηγούμενης μορφής μάθησης . Οι προγραμματισμένες συναντήσεις με τους συμμαθητές και του διδάσκοντες μέσα στα πλαίσια ενός προγράμματος εκπαιδευτικών δραστηριοτήτων, ικανοποιούν την ανθρώπινη ανάγκη για αλληλεπίδραση. Η φυσική παρουσία και η ανταλλαγή απόψεων βοηθάει στην μείωση του χρόνου ολοκλήρωσης προγραμματισμένων εργασιών και λειτουργεί ως κίνητρο για την αύξηση της ποιότητας του περιεχομένου τους.

2.4.6. Δεξιότητες σε νέες τεχνολογίες.

Οι αλματώδεις εξελίξεις στον τομέα του διαδικτύου δημιουργούν ψηφιακό αναλφαβητισμό. Η αμφίδρομη επικοινωνία διδασκόντων και διδασκομένων μέσω

εικονικών τάξεων και ψηφιακών διεπαφών χρηστών, δίνουν την δυνατότητα σε εκπαιδευτές και εκπαιδευόμενους να έρθουν σε επαφή με νέες τεχνολογίες και τεχνογνωσίες, να εξοικειωθούν σε ψηφιακά περιβάλλοντα και να έχουν μια καλύτερη προοπτική αποκατάστασης στον χώρο εργασίας.

ΚΕΦΑΛΑΙΟ 3

3. Εισαγωγικές έννοιες στην Αλγοριθμική σκέψη

3.1.Βασικές έννοιες

Εντολές: είναι λέξεις που δεσμεύομαι να μην τις χρησιμοποιήσω ως ονόματα παραμέτρων καθώς αυτό απαγορεύεται. Πρέπει να τις γράψω όπως έχει ορίσει ο κατασκευαστής της γλώσσας για να μπορέσουν να εκτελεστούν από τον υπολογιστή.

Δεσμευμένες λέξεις: Οι λέξεις που χρησιμοποιούνται στον ψευδοκώδικα και έχουν αυστηρά καθορισμένη έννοια και τρόπο χρήσης ονομάζονται δεσμευμένες λέξεις και δεν μπορούν να χρησιμοποιούνται ως ονόματα μεταβλητών.

Παράμετροι: μπορεί να είναι μεταβλητές , σταθερές, ή η ονομασία του προγράμματος

Σταθερά: Είναι η ποσότητα εκείνη που δεν μεταβάλλεται κατά την διάρκεια εκτέλεσης ενός αλγορίθμου

Μεταβλητή: Είναι η ποσότητα εκείνη που μπορεί να μεταβληθεί κατά την διάρκεια εκτέλεσης ενός αλγορίθμου. Στο μάθημα αυτό είναι μια «θέση μνήμης» όπου μπορούμε να τοποθετήσουμε μια τιμή.

Σταθερές–Μεταβλητές: Χωρίζονται σε αριθμητικές , αλφαριθμητικές και λογικές.

A ← 5 (Η αριθμητική μεταβλητή Α παίρνει την αριθμητική σταθερά 5)

Όνομα ← ’’Κώστας’’ (Η αλφαριθμητική μεταβλητή Όνομα παίρνει την αλφαριθμητική σταθερά ‘’Κώστας’’)

flag ← ψευδής. (Η λογική μεταβλητή flag παίρνει την λογική σταθερά ψευδής. Η λέξη ψευδής είναι δεσμευμένη και δηλώνει μία από τις δύο καταστάσεις που μπορεί να πάρει μια λογική μεταβλητή)

3.2.Κανόνες ονομασίας μεταβλητών και παραμέτρων

- Επιτρέπεται συνδυασμός γραμμάτων και αριθμών αρκεί ο πρώτος χαρακτήρας να είναι αναγκαστικά γράμμα (α - ω, Α - Ω, a - z, A - Z, 0 - 9).
- Απαγορεύεται οποιοσδήποτε άλλος ειδικός χαρακτήρας εκτός από την κάτω παύλα (_).

3.2.1. Παραδείγματα αποδεκτών ονομάτων

Αριθμός	Είναι αποδεκτό.
Αριθμός1	Είναι αποδεκτό.
Αριθμός-1	Δεν είναι αποδεκτό, γιατί ο χαρακτήρας "-" (πλην) είναι το σύμβολο της αφαίρεσης.
Αριθμός _2	Είναι αποδεκτό. Επιτρέπεται η (_).
2οςαριθμος	Δεν είναι αποδεκτό. Δεν επιτρέπεται το όνομα να ξεκινά από αριθμητικό ψηφίο.
A12345678	Είναι αποδεκτό. Ο αλγόριθμος δεν ενδιαφέρεται για την ορθογραφία.
Αριθμός.2	Δεν είναι αποδεκτός ο χαρακτήρας ".", ο οποίος χρησιμοποιείται στους δεκαδικούς αριθμούς.
Αριθμός 2	Δεν είναι αποδεκτός ο χαρακτήρας του κενού " " .
Λεφτά$	Δεν είναι αποδεκτό. Δεν επιτρέπεται άλλο ειδικό σύμβολο εκτός από (_).
Τέλος	Δεν είναι αποδεκτό, γιατί η λέξη Τέλος είναι εντολή και άρα δεσμευμένη.
Τέλος_όλων	Είναι αποδεκτό. Για τον αλγόριθμο οι λέξεις Τέλος και Τέλος_όλων είναι διαφορετικές.

3.3.Εντολή Διάβασε

Είναι μια από τις δύο εντολές που δίνουν τιμή σε μια μεταβλητή. Η δεύτερη εντολή είναι η εντολή εκχώρησης που θα την δούμε πιο κάτω. Δέχεται ως παραμέτρους μόνο μεταβλητές (μία ή περισσότερες. Όταν είναι περισσότερες από μία διαχωρίζονται με το σύμβολο του κόμματος). Όταν την συναντάει ο υπολογιστής στο στάδιο εκτέλεσης, σταματάει η ροή του προγράμματος και ο υπολογιστής περιμένει από τον χρήστη να δώσει τόσες τιμές, όσες και οι μεταβλητές που βρίσκονται δίπλα στην εντολή Διάβασε.

3.4.Εντολή Εμφάνισε – Εκτύπωσε - Γράψε

Δέχεται ως παραμέτρους μία ή περισσότερες σταθερές (μόνο αλφαριθμητικές) ή μεταβλητές. Οι σταθερές εμφανίζονται ως έχουν στην οθόνη του υπολογιστή (χωρίς τα “ “), ενώ στις μεταβλητές εμφανίζεται το περιεχόμενό τους. Οι μεταβλητές που

τυχόν υπάρχουν θα πρέπει να είναι γνωστές από προηγούμενες εντολές. Τα ίδια ισχύουν και για την Εκτύπωσε.

3.4.1. Παραδείγματα εντολής Εμφάνισε

Αν δεχτούμε ως αρχικές τιμές α = 5 , β = 12, Όνομα = ”Κώστας” τότε:

Εντολή	Οθόνη
Εμφάνισε α	5
Εμφάνισε “α”	α
Εμφάνισε “α=”,α	α=5
Εμφάνισε “Ο “, Όνομα, “ πήρε βαθμό “,β	Ο Κώστας πήρε βαθμό 12

3.5.Εντολή εκχώρησης

Είναι η δεύτερη εντολή που δίνει τιμή σε μια μεταβλητή και η μοναδική εντολή υπολογισμού - πράξεων στον προγραμματισμό. Στο αριστερό μέρος βρίσκεται πάντα μια μεταβλητή που περιμένει να πάρει τιμή από το δεξί μέρος. Στο δεξί μέρος βρίσκεται μια μεταβλητή ή σταθερά ή συνδυασμός αριθμητικών μεταβλητών και σταθερών δημιουργώντας μια παράσταση. Οι μεταβλητές που τυχόν υπάρχουν στο δεξί μέρος θα πρέπει να είναι γνωστές από προηγούμενες εντολές.

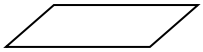
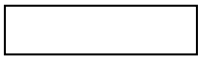
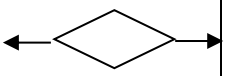
3.5.1. Παραδείγματα αποδεκτών εντολών εκχώρησης

τιμή ← "Καλά"	Σωστό. Η μεταβλητή τιμή είναι αλφαριθμητική.
τιμή ←τιμή + 5	Σωστό, αν η μεταβλητή τιμή είναι αριθμητική και είναι γνωστή από προηγούμενη εντολή.
α + τιμή ←9	Λάθος. Στο αριστερό μέρος επιτρέπεται μόνο μία μεταβλητή
τιμή ←ασ ←7	Λάθος.
τιμή ←α*β+2	Σωστό, αν οι μεταβλητές α, β και τιμή είναι αριθμητικές και οι α,β έχουν τιμές από προηγούμενη εντολή.
τιμή ←"δ" + 8	Λάθος.
τιμή ←τιμή + 1	Σωστό.
τιμή ←"τιμή" + 1	Λάθος, η παράμετρος "τιμή" δεν είναι αριθμητική μεταβλητή αλλά αλφαριθμητική σταθερά.

τιμή ←3 * τιμή ^ 2 + 11	Σωστό.
τιμή ←αριθμός	Σωστό, αν οι μεταβλητές τιμή και αριθμός είναι του ιδίου τύπου.
τιμή ← “Κώστας”	Σωστό. Η μεταβλητή τιμή είναι αλφαριθμητική.
Τιμή←χ>5	Σωστό. Η μεταβλητή τιμή είναι λογική και δέχεται δύο καταστάσεις “Αληθής” και “Ψευδής”.
τιμή = β + 5	Λάθος. Η εντολή εκχώρησης δεν έχει σχέση με την ισότητα.

3.6.Σύμβολα διαγράμματος ροής

3.6-1 Σύμβολα διαγράμματος ροής

Έλλειψη		Αρχή ή Τέλος του αλγορίθμου
Πλάγιο παραλληλόγραμμο		Είσοδος δεδομένων-Εξοδος αποτελεσμάτων
Ορθογώνιο παραλληλόγραμμο		Εκτέλεση πράξεων
Ρόμβος		Έλεγχος συνθήκης
Βέλος		Ροή εκτέλεσης ενεργειών

3.7.Πώς συντάσσεται ένας αλγόριθμος

Ένας αλγόριθμος σε ψευδογλώσσα ξεκινάει πάντα με τη λέξη Αλγόριθμος και ακολουθείται από το όνομα του αλγορίθμου. Παράδειγμα: Αλγόριθμος Πράξεις_Αριθμών. Αν δεν σας έρχεται στο μυαλό εύκολα κάποιο όνομα για τον Αλγόριθμο στις εξετάσεις χρησιμοποιείστε ως όνομα το θέμα. Έτσι, αν γράφετε το θέμα 3 γράψτε Αλγόριθμος Θέμα_3. Προσοχή στο όνομα! Πρέπει να αποφεύγεται η χρήση του ονόματος του αλγορίθμου και ως όνομα κάποιας μεταβλητής ή σταθεράς. Ο αλγόριθμος κλείνει με την εντολή Τέλος. Παράδειγμα: Τέλος Πράξεις_Αριθμών.

3.8.Είδη τελεστών

3.8.1. Αριθμητικοί τελεστές

3.8.1-1 Αριθμητικοί τελεστές

ΤΕΛΕΣΤΗΣ	ΣΥΜΒΟΛΟ
ΠΡΟΣΘΕΣΗ	+
ΑΦΑΙΡΕΣΗ	-
ΠΟΛΛΑΠΛΑΣΙΑΣΜΟΣ	*
ΔΙΑΙΡΕΣΗ	/
ΥΨΩΣΗ ΣΕ ΔΥΝΑΜΗ	^
ΑΚΕΡΑΙΟ ΜΕΡΟΣ ΔΙΑΙΡΕΣΗΣ(ΑΚΕΡΑΙΟΙ)	div
ΥΠΟΛΟΙΠΟ ΔΙΑΙΡΕΣΗΣ(ΑΚΕΡΑΙΟΙ)	mod

3.8.2. Λογικοί τελεστές

Είναι δυνατόν να δημιουργηθούν σύνθετες λογικές συνθήκες χρησιμοποιώντας τις λογικές πράξεις της **άρνησης, σύζευξης, διάζευξης**.

Άρνηση (ΟΧΙ) Είναι αληθής (Α) όταν η αρχική συνθήκη είναι ψευδής (Ψ) και αντίστροφα.

Σύζευξη (ΚΑΙ) Είναι αληθής (Α) όταν και οι δύο συνθήκες είναι αληθείς (Α) και ψευδής (Ψ) σε όλες τις άλλες περιπτώσεις.

Διάζευξη (Ή) Είναι αληθής (Α) όταν τουλάχιστον μία από τις δύο συνθήκες είναι αληθής (Α) και ψευδής (Ψ) όταν και οι δύο συνθήκες είναι ψευδείς(Ψ) .

3.8.3. Τελεστές σύγκρισης

Οι συνθήκες παίζουν σημαντικό ρόλο στους αλγορίθμους γιατί ανάλογα με τη τιμή τους εκτελούνται κάποιες εντολές ή όχι. Για τη διατύπωση των συνθηκών χρησιμοποιούνται οι τελεστές σύγκρισης.

3.8.3-1 Συγκριτικοί τελεστές

ΤΕΛΕΣΤΗΣ	ΣΗΜΑΣΙΑ
=	ΙΣΟ
<>	ΔΙΑΦΟΡΕΤΙΚΟ
>	ΜΕΓΑΛΥΤΕΡΟ
<	ΜΙΚΡΟΤΕΡΟ
>=	ΜΕΓΑΛΥΤΕΡΟ Η ΙΣΟ
<=	ΜΙΚΡΟΤΕΡΟ Η ΙΣΟ

3.9.Προτεραιότητα των πράξεων

Πρώτα εκτελούνται οι πράξεις εντός παρενθέσεων. Προτεραιότητα έχουν οι αριθμητικές πράξεις. Πιο συγκεκριμένα η σειρά είναι: δυνάμεις, πολλαπλασιασμός και διαίρεση (με τα div και mod), πρόσθεση και αφαίρεση. Πράξεις με ίδια προτεραιότητα εκτελούνται από αριστερά προς τα δεξιά. Στη συνέχεια προτεραιότητα έχουν οι συγκριτικές πράξεις. Τέλος, αξιολογούνται οι τιμές των λογικών πράξεων

Μεγαλύτερη προτεραιότητα

1. παρενθέσεις
2. δυνάμεις
3. * , / , div , mod
4. + , -

Όσον αφορά την ιεραρχία στους τελεστές αυτή είναι:

1. πρώτα εκτελούνται οι αριθμητικοί τελεστές
2. στη συνέχεια οι συγκριτικοί
3. και τέλος οι λογικοί

3.10. Ακέραια Διαίρεση – Εντολές div και mod

div: Δίνει το πηλίκο της ακεραίας διαίρεσης δύο ακεραίων αριθμών.

mod: Δίνει το υπόλοιπο της ακεραίας διαίρεσης δύο ακεραίων αριθμών.

Η εντολή mod χρησιμοποιείται κυρίως σε δύο είδη ασκήσεων

- *Όταν θέλω να βρω αν ένας αριθμός είναι άρτιος ή περιττός*. Αν x mod 2=0 τότε ο αριθμός αυτός είναι άρτιος. Αν x mod 2 = 1 τότε ο αριθμός αυτός είναι περιττός.
- *Όταν θέλω να βρω αν ένας αριθμός είναι ακέραιο πολλαπλάσιο ενός άλλου*. Αν x mod y = 0 τότε αυτό σημαίνει ότι το x είναι ακέραιο πολλαπλάσιο του y

Ποιο είναι το αποτέλεσμα από την εκτέλεση των παρακάτω πράξεων:

1. 14 mod 5-25 mod 8
2. 3 * (3 mod 2) + 4 div (5 mod 3)
3. 13 mod (27 div 4)
4. 2 ^ 3 + 3 * (27 mod (25 mod 7))
5. 13/2-3 mod 2 - 3 div 2
6. 13 / 4 + 2 * (5 mod 3) * 4
7. 25 mod 22 div4
8. ((13+ 2) div 2) / (7 - 4 + 1)
9. 3 * (27 mod (23 mod 6))

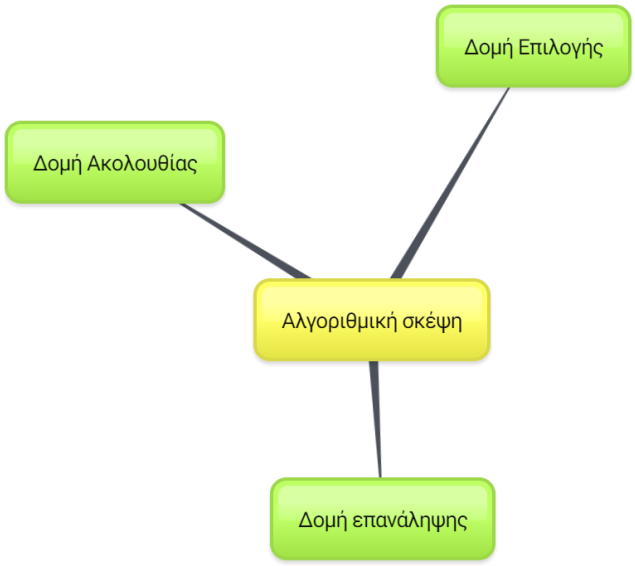
Λύση

1. 14 mod 5-25 mod 8 = 4 - 1 =3
2. 3 * (3 mod 2) + 4 div (5 mod 3) = 3 * 1 + 4 div 2 = 3 + 2 = 5
3. 13 mod (27 div 4)= 13 mod 6= 1
4. 2 ^ 3 + 3 * (27 mod (25 mod 7)) = 8 + 3 * (27 mod 4) = 8 + 3 * 3 = 8 + 9 = 17
5. 13/2-3 mod 2 - 3 div 2 = 6.5-1 -1 =4.5
6. 13 / 4 + 2 * (5 mod 3) * 4 = 3.25 + 2 * 2 * 4 = 3.25 + 16 = 19.25
7. 25 mod 22 div 4 = 3 div 4 = 0 (οι πράξεις εκτελούνται από αριστερά προς τα δεξιά)
8. ((13 + 2) div 2) /(7 - 4 + 1) = (15 div 2)/4 = 7/4 = 1.75
9. 3 * (27 mod (23 mod 6)) = 3 * (27 mod 5) = 3 * 2 = 6.

ΚΕΦΑΛΑΙΟ 4

4. Δομή Ακολουθίας

Στον δομημένος προγραμματισμό υπάρχουν τρείς δομές. Η δομή της Ακολουθίας, η δομή της Επιλογής και η δομή της επανάληψης. Αυτό φαίνεται και από το παρακάτω σχήμα:



4-1 Δομημένος προγραμματισμός - Τρείς δομές

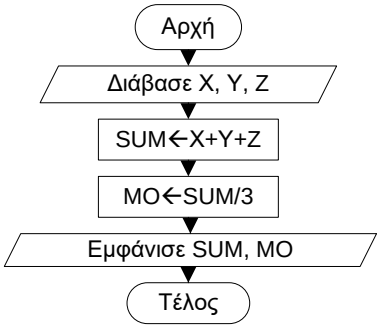
4.1.Πρόβλημα Α1

Να φτιαχτεί αλγόριθμος που να διαβάζει τρεις αριθμούς και να βρίσκει και εμφανίζει το άθροισμα και τον μέσο όρο αυτών.

Λύση: Στην ακολουθιακή δομή οι εντολές εκτελούνται η μία μετά την άλλη αδιαμφισβήτητα. Δεν υπάρχει περίπτωση να μην εκτελεστεί κάποια από αυτές τις εντολές. Στόχος μας είναι η εκτέλεση του προγράμματος από την αρχική πρώτη εντολή μέχρι και την τελευταία. Πρόκειται για την πιο απλή δομή. Σε μεγάλο βαθμό καθορίζεται η άσκηση αν έχουμε καταλάβει τι είναι αυτό που πρέπει να διαβάσουμε και τι είναι αυτό που πρέπει να εμφανίσουμε. Εδώ χρειάζεται να διαβάσουμε τρεις αριθμούς και να εμφανίσουμε δύο, συγκεκριμένα το άθροισμα και τον μέσο όρο των αριθμών. Άρα χρειαζόμαστε πέντε μεταβλητές και άρα πέντε ανάλογους χώρους στην μνήμη του υπολογιστή, που εμείς ονομάζουμε σύμφωνα με τους κανόνες ονομασίας, δηλαδή συνδυασμός γραμμάτων, αριθμών και κάτω παύλας, αρκεί ο

πρώτος χαρακτήρας να είναι αναγκαστικά γράμμα και οι κάτω παύλα να μην βρίσκεται στο τέλος.

Αλγόριθμος Μέσος_όρος
Διάβασε X, Y, Z
SUM← X+Y+Z
MO←SUM/3
Εμφάνισε “Το άθροισμα είναι”, SUM
Εμφάνισε “Ο μέσος όρος είναι”, MO
Τέλος Μέσος_όρος



4.1-1 Λογικό διάγραμμα

Εδώ παρατηρούμε ότι σε ένα πρόγραμμα χρησιμοποιούμε δύο είδη λέξεων. Τις εντολές που είναι γραμμένες έντονα και τις παραμέτρους των εντολών αυτές δηλαδή που βρίσκονται δίπλα από κάθε εντολή. Τις εντολές δεν μπορούμε να τις μεταβάλλουμε. Είναι συγκεκριμένες λέξεις του κώδικα που έχει ορίσει ο κατασκευαστής της γλώσσας και που ξέρουμε ότι αν τις χρησιμοποιήσουμε ακριβώς, τότε θα συμβεί κάτι συγκεκριμένο στον υπολογιστή. Οι εντολές ονομάζονται και δεσμευμένες λέξεις, καθώς δεσμευόμαστε να μην τις χρησιμοποιήσουμε ως ονόματα παραμέτρων.

Οι παράμετροι των εντολών μπορεί να είναι το όνομα του προγράμματος, μεταβλητές ή σταθερές αριθμητικές, αλφαριθμητικές και λογικές. Οι αλφαριθμητικές σταθερές βρίσκονται πάντα σε εισαγωγικά “ “ μονά ή διπλά.

Εμείς ονομάζουμε τις μεταβλητές του προγράμματος. Όπως θα βαπτίσουμε την κάθε μεταβλητή, έτσι θα την καλούμε σε όλη την διάρκεια εκτέλεσης του αλγορίθμου. Δεν είναι δυνατόν να ονομάσουμε το άθροισμα SUM και μετά να χρησιμοποιήσουμε την μεταβλητή με το όνομα SAM.

Για την χρησιμοποίηση της τιμής κάποιας μεταβλητής πρέπει οπωσδήποτε να έχει δοθεί τιμή σ' αυτή τη μεταβλητή, είτε με εντολή εκχώρησης είτε με είσοδο δεδομένων από το χρήστη. Στην αλγοριθμική σκέψη χρησιμοποιούμε μια μεταβλητή με δύο τρόπους: Την τοποθετούμε στο δεξί μέρος μιας εντολής εκχώρησης, ή την εμφανίζουμε στην οθόνη χρησιμοποιώντας την ως παράμετρο σε μια εντολή εμφάνισης.

Τέλος ο μοναδικός τρόπος της επεξεργασίας είναι να κάνουμε πράξεις με μία οι περισσότερες εντολές εκχώρησης.

4.2.Πρόβλημα Α2

Ένας καταστηματάρχης θέλει να αλλάξει τις τιμές των προϊόντων του λόγω εκπτώσεων. Να αναπτύξετε αλγόριθμο ο οποίος: α. να διαβάζει την τιμή ενός προϊόντος, β. να διαβάζει το ποσοστό της έκπτωσης, γ. να εμφανίζει το μήνυμα «Η τελική τιμή του προϊόντος μετά την έκπτωση είναι» καθώς και την τιμή του προϊόντος μετά την έκπτωση.

Λύση: Σε ασκήσεις με ποσοστά έχουμε τις εξής έννοιες:

1. Την αρχική τιμή.
2. Το ποσοστό επί της εκατό.
3. Την αξία του ποσοστού που προκύπτει από τον πολλαπλασιασμό του ποσοστού με την αρχική τιμή δια εκατό.
4. Και την τελική τιμή η οποία προκύπτει από την πρόσθεση (φόρος) ή από την αφαίρεση (έκπτωση) της αξίας του ποσοστού στην αρχική τιμή.

ΠΡΟΓΡΑΜΜΑ εκπτώσεις
ΜΕΤΑΒΛΗΤΕΣ
ΠΡΑΓΜΑΤΙΚΕΣ: αρχική_τιμή, τελική_τμή, αξία_ποσσοστού
ΑΚΕΡΑΙΕΣ: ποσοστό_έκπτωσης
ΑΡΧΗ
ΓΡΑΨΕ ‘Δώσε την αρχική τιμή του προιόντος’
ΔΙΑΒΑΣΕ αρχική_τιμή
ΓΡΑΨΕ ‘Δώσε την έκπτωση που θα γίνει το προιόν’
ΔΙΑΒΑΣΕ ποσοστό_έκπτωσης
αξία_ποσσοστού← αρχική_τιμή*ποσοστό_έκπτωσης / 100
τελική_τιμή ← αρχική_τιμή- αξία_ποσσοστού
ΓΡΑΨΕ ‘Η τελική τιμή του προϊόντος μετά την έκπτωση είναι ’, νέα_τιμή
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ

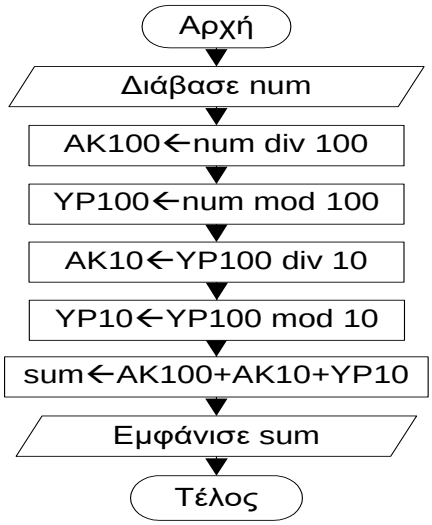
4.3.Πρόβλημα Α3

Να φτιαχτεί αλγόριθμος ο οποίος να διαβάζει έναν τριψήφιο αριθμό από το 100 έως το 999 και να εμφανίζει το άθροισμα των ψηφίων του.

Λύση: Ένας τριψήφιος αριθμός αποτελείται από εκατοντάδες, από δεκάδες και από μονάδες. Για να απομονώσουμε τις εκατοντάδες χρησιμοποιούμε την ακέραια

διαίρεση με το 100. Το υπόλοιπο με το 100 είναι αυτό που θα μας βοηθήσει να απομονώσουμε τις δεκάδες και τις μονάδες. Αν ο αρχικός αριθμός πχ. ήταν το 358, το υπόλοιπο με το εκατό θα μας δώσει 58. Διαιρώντας τον αριθμό αυτό με το δέκα, η ακέραια διαίρεση μας δίνει τις δεκάδες και το υπόλοιπο μας δίνει τις μονάδες.

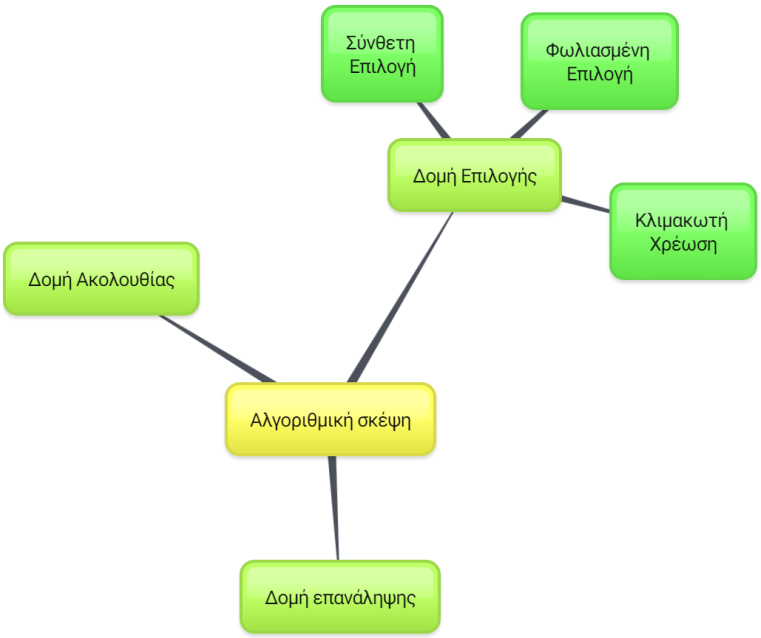
Αλγόριθμος άθροισμα_ψηφίων
Διάβασε num
ΑΚ100←num div 100
ΥΡ100←num mod 100
ΑΚ10← ΥΡ100 div 10
ΥΡ10← ΥΡ100 mod 10
sum←ΑΚ100+ΑΚ10+ΥΡ10
Εμφάνισε sum
Τέλος άθροισμα_ψηφίων



4.3-1 Λογικό διάγραμμα

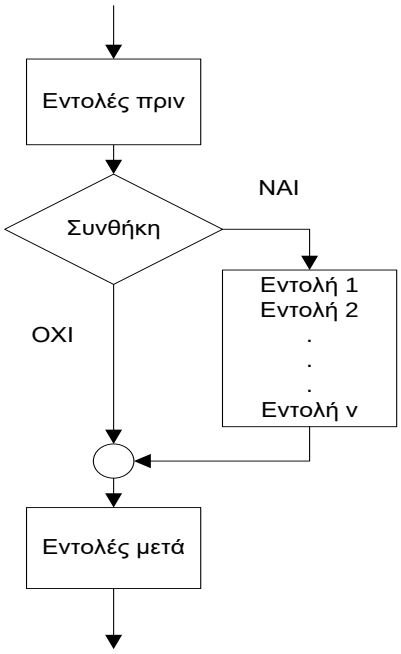
ΚΕΦΑΛΑΙΟ 5

5. Δομή επιλογής



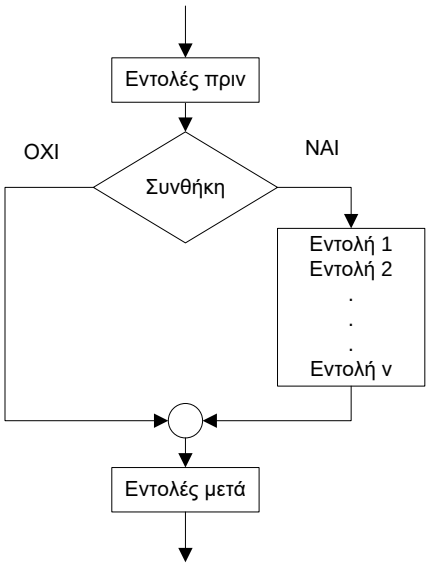
5-1 Νοητικός χάρτης δομής επιλογής

5.1.Απλή δομή επιλογής



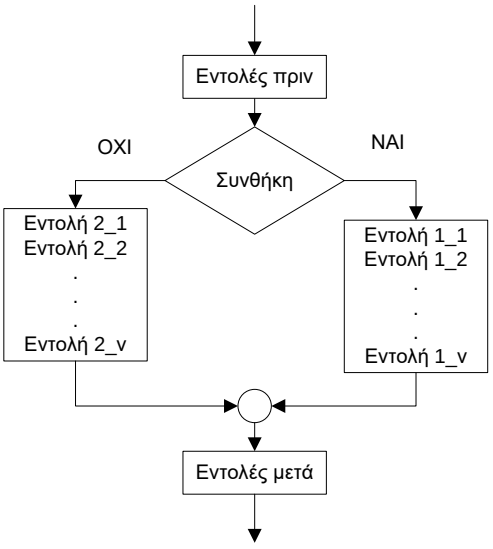
5.1-1 Απλή δομή επιλογής-Σχήμα 1

Εντολές πριν
Αν Σ τότε
Εντολή 1
Εντολή 2
…
Εντολή v
Τέλος_αν
Εντολές μετά



5.1-2 Απλή δομή επιλογής-Σχήμα 2

Στο παραπάνω σχήμα έχουμε ένα κομμάτι κώδικα εκφρασμένο σε λογικό διάγραμμα. Οι εντολές που βρίσκονται πριν την συνθήκη εκτελούνται αδιαμφισβήτητα. Οι εντολές που βρίσκονται μετά την συνθήκη εκτελούνται αδιαμφισβήτητα. Η ομάδα των εντολών που βρίσκεται μέσα στη συνθήκη εκτελείται μόνο όταν η απάντηση στην ερώτηση που βρίσκεται μέσα στην συνθήκη είναι ναι, ή αλλιώς όταν η πρόταση που βρίσκεται μέσα στην συνθήκη είναι αληθής. Σε διαφορετική περίπτωση από τις εντολές που βρίσκονται πριν την συνθήκη (Εντολές πριν) πηγαίνουμε στις εντολές που βρίσκονται μετά την συνθήκη (εντολές μετά) χωρίς να υπάρχει καμία επιπλέον εκτέλεση ενδιάμεσων εντολών. Πολλοί μπερδεύονται νομίζοντας ότι οι Εντολές μετά εκτελούνται όταν δεν ισχύει η παραπάνω συνθήκη. Αυτό είναι μια εσφαλμένη εντύπωση. Οι Εντολές μετά θα εκτελεστούν ούτως ή άλλως είτε ισχύει η παραπάνω συνθήκη είτε όχι.




```
Εντολές πριν
Αν Σ τότε
    Εντολή 1_1
    Εντολή 1_2
    .,,
    Εντολή 1_v
αλλιώς
    Εντολή 2_1
    Εντολή 2_2
    .,,
    Εντολή 2_v
Τέλος_αν
Εντολές μετά
```

5.1-3 Δομή επιλογής με αλλιώς

Στο παραπάνω σχήμα παρατηρώ επιπλέον ότι μεταξύ των προηγούμενων εντολών και των επόμενων εντολών θα εκτελεστούν μια ή περισσότερες διαφορετικές εντολές ανάλογα με το αν η συνθήκη είναι αληθής ή είναι ψευδής.

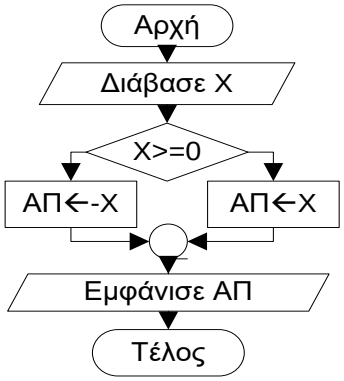
Η εντολή επιλογής «Αν» ακολουθείται πάντα από «Τέλος_Αν». Όταν όμως χρησιμοποιήσουμε την δομή περιορισμένης επιλογής Αν…τότε…Τέλος_αν και εκτελείται μόνο μια εντολή τότε μπορείτε να γράψετε την δομή περιορισμένης επιλογής σε μια μόνο σειρά χωρίς το «Τέλος_ΑΝ». Δηλαδή : Αν χ>0 τότε Άθροισμα←Άθροισμα + χ

5.1.1. Πρόβλημα Β1

Να φτιαχτεί αλγόριθμος που θα διαβάζει έναν αριθμό και να εμφανίζει την απόλυτη τιμή του.

Λύση: Αν η συνθήκη ισχύει, εκτελείται το μπλοκ εντολών που περιέχεται μεταξύ της δεσμευμένης λέξης **τότε** και του **αλλιώς**. Εδώ σε περίπτωση που το Χ είναι θετικό ή μηδέν θα εκτελεστεί η εντολή ΑΠ←Χ. Στην περίπτωση που η συνθήκη δεν ισχύει, τότε εκτελείται το μπλοκ εντολών που υπάρχει μεταξύ της δεσμευμένης λέξης **αλλιώς** και του . Αν λοιπόν ο αριθμός Χ είναι μικρότερος του μηδενός τότε θα εκτελεστεί η εντολή Α Π←-Χ. Παρατηρώ ότι η μεταβλητή ΑΠ θα πάρει τιμή είτε η συνθήκη είναι αληθής, είτε είναι ψευδής. Έτσι στην επόμενη εντολή Εμφάνισε ΑΠ η μεταβλητή ΑΠ έχει σίγουρα τιμή και θα εμφανιστεί το περιεχόμενό της.

Αλγόριθμος Απόλυτη_τιμή
Διάβασε Χ
Αν Χ>= 0 **τότε**
    ΑΠ←Χ
αλλιώς
    ΑΠ←-Χ
Τέλος_αν
Εμφάνισε ΑΠ
Τέλος Απόλυτη_τιμή



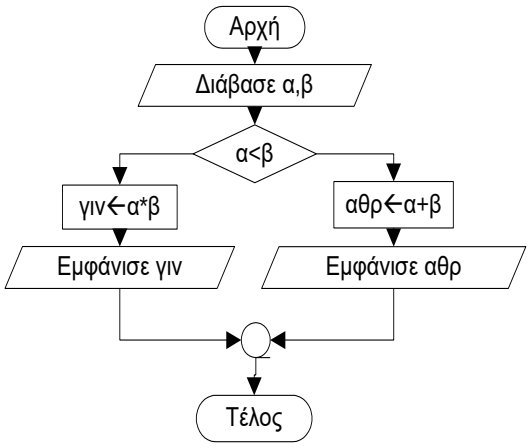
5.1.1-1 Λογικό διάγραμμα

5.1.2. Πρόβλημα Β2

Να φτιαχτεί αλγόριθμος που διαβάζει δύο αριθμούς a,β. Αν a<β τότε να εμφανίζεται το άθροισμά τους, αλλιώς να εμφανίζεται το γινόμενό τους.

Λύση: Εδώ στο παρακλάδι «τότε» της **Αν** δίνουμε τιμή στην μεταβλητή αθρ και στο άλλο παρακλάδι δίνουμε τιμή στην μεταβλητή γιν. Έτσι δεν θα μπορούσαμε να εμφανίσουμε μετά το τέλος της δομής της επιλογής με μία εντολή **Εμφάνισε** αθρ, γιν, γιατί θα είχαν πάρει τιμή ή η μία ή η άλλη μεταβλητή και ποτέ και οι δύο μαζί. Για αυτό και αν και εφόσον πάρει τιμή η μεταβλητή αθρ, στο ίδιο παρακλάδι την εμφανίζω. Με την ίδια λογική αν και εφόσον πάρει τιμή η μεταβλητή γιν, την εμφανίζω στο ίδιο παρακλάδι.

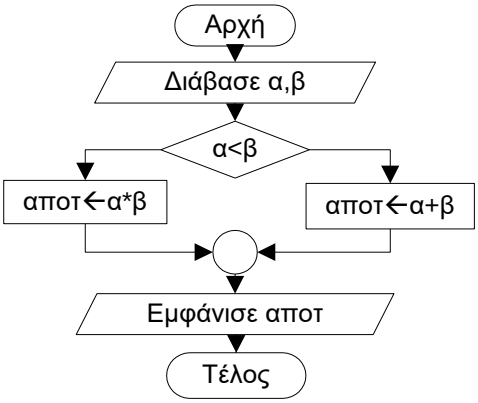
Αλγόριθμος Πρόβλημα
Διάβασε α,β
Αν α<β τότε
    αθρ← α+β
    Εμφάνισε αθρ
αλλιώς
    γιν←α*β
    Εμφάνισε γιν
Τέλος_αν
Τέλος Πρόβλημα



5.1.2-1 Λογικό διάγραμμα

Αν ήθελα να εμφανίσω ένα γενικό αποτέλεσμα κάτω από την δομή της επιλογής τότε θα έπρεπε ο παραπάνω κώδικας να μετασχηματιστεί:

Αλγόριθμος Πρόβλημα
Διάβασε α,β
Αν α<β τότε
    αποτέλεσμα ← α+β
αλλιώς
    αποτέλεσμα ←α*β
Τέλος_αν
Εμφάνισε αποτέλεσμα
Τέλος Πρόβλημα



5.1.2-2 Λογικό διάγραμμα

Εδώ σε κάθε παρακλάδι η μεταβλητή αποτέλεσμα δέχεται τιμή και άρα μπορεί να εμφανιστεί κάτω από το **Τέλος_αν .**

5.2.Εμφωλευμένη ή Φωλιασμένη Επιλογή

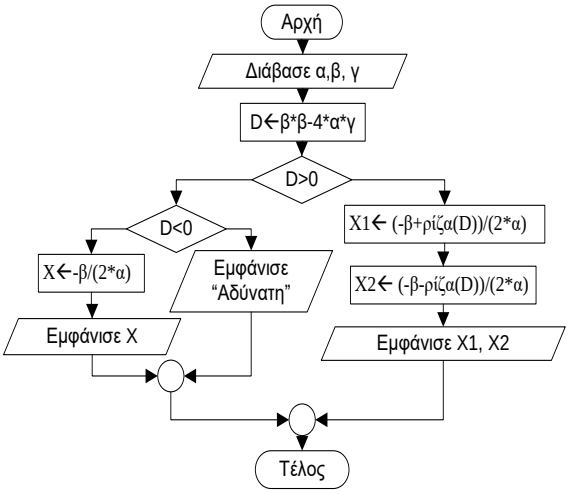
Όταν χρησιμοποιώ δομή επιλογής μέσα σε δομή επιλογής τότε ονομάζω την επιλογή μου **εμφωλευμένη επιλογή**.

5.2.1. Πρόβλημα Γ1

Να φτιαχτεί ο αλγόριθμος επίλυσης της δευτεροβάθμιας εξίσωσης $a*x^2+b*x+c=0$. Να εμφανιστούν οι πιθανές λύσεις της εξίσωσης.

Λύση: Οι πιθανές περιπτώσεις της εξίσωσης είναι τρείς. Να έχει δύο ρίζες, να έχει μία ρίζα και να μην έχει ρίζες στο σύνολο των πραγματικών αριθμών. Όταν έχω περισσότερες από δύο περιπτώσεις τότε μπορώ να τοποθετήσω στο όχι της πρώτης συνθήκης μια νέα συνθήκη κ.ο.κ. Σε κάθε συνθήκη που προσθέτω, μπορώ να αντιστοιχίσω και μία παραπάνω περίπτωση. Οι περιπτώσεις που μπορούν να αντιστοιχιστούν είναι πάντα κατά μία περισσότερες από τις συνθήκες που χρησιμοποιώ. Εδώ που έχω τρείς περιπτώσεις θα χρησιμοποιήσω δύο συνθήκες.

Αλγόριθμος Βθμια_εξίσωση
Διάβασε α,β,γ
D←β*β-4*α*γ
Αν D>0 **τότε**
X1←(-β+ρίζα(D))/(2*α)
X2←(-β-ρίζα(D))/(2*α)
Εμφάνισε X1,X2
αλλιώς
Αν D<0 **τότε**
Εμφάνισε “αδύνατη”
αλλιώς
X←-b/(2*α)
Εμφάνισε X
Τέλος_αν
Τέλος_αν
Τέλος Βθμια_εξίσωση



5.2.1-1 Λογικό διάγραμμα

5.2.2. Πρόβλημα Γ2

Να γραφεί αλγόριθμος που να διαβάζει το βαθμό ενός μαθητή και να εμφανίζει το χαρακτηρισμό «*Απορρίπτεται*» αν είναι μικρότερος ή ίσος του 10, «*Σχεδόν Καλά*» αν είναι έως και 13, «*Καλά*» αν είναι έως και 16, «*Πολύ καλά*» αν είναι έως και 18 και «*Άριστα*» αν είναι άνω του 18.

5.2.2.1. Λύση από τα μικρά όρια στα μεγάλα

Η άσκηση αυτή επιδέχεται πολλές λύσεις χρησιμοποιώντας τη δομή της επιλογής (σύνθετη, εμφωλευμένη, πολλαπλή επιλογή). Εδώ αρχικά δίνεται η πιο

ενδεδειγμένη λύση με εμφωλευμένη ή πολλαπλή επιλογή. ***Πολλαπλή επιλογή*** (Μέθοδος αλλιώς_αν) χρησιμοποιούμε πάντα σε περιπτώσεις διαστημάτων. Με την εντολή «αλλιώς_αν Βαθμός <= 13 τότε» έχω πετύχει δύο συγκρίσεις και όχι μία. Το γεγονός ότι βρίσκεται στο αλλιώς της εντολής «Αν Βαθμός <=10 τότε» δείχνει ότι ο βαθμός είναι μεγαλύτερος του 10, ενώ από την εντολή «αν Βαθμός <= 13 τότε» φαίνεται ότι ο βαθμός είναι μικρότερος ή ίσος από το 13. Άρα βρίσκομαι στο διάστημα δέκα μέχρι δεκατρία (10<Βαθμός<=13). Το ίδιο ισχύει και για τις υπόλοιπες εντολές αλλιώς_αν.

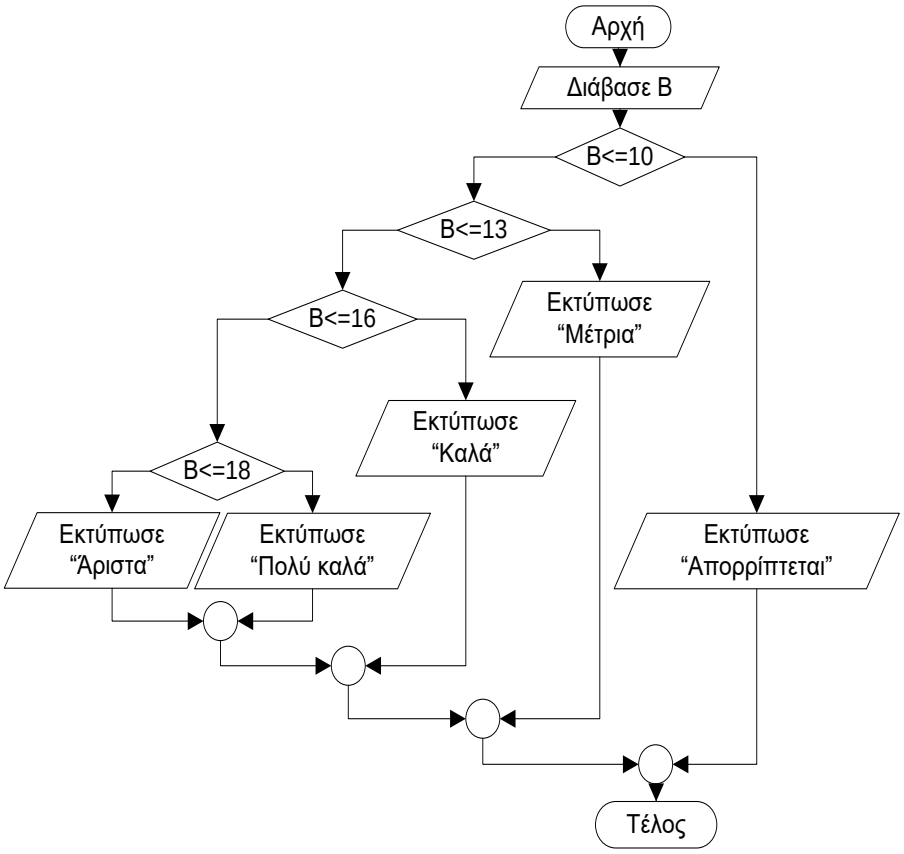
Εμφωλευμένα Αν

```
Αλγόριθμος Βαθμολογία
Διάβασε Βαθμός
Αν Βαθμός <=10 τότε
    Εμφάνισε  “Απορρίπτεται”
αλλιώς
    Αν Βαθμός <= 13 τότε
        Εμφάνισε “Σχεδόν Καλά”
    αλλιώς
        Αν Βαθμός <= 16 τότε
            Εμφάνισε “Καλά”
        αλλιώς
            Αν Βαθμός <= 18 τότε
                Εμφάνισε “Πολύ Καλά”
            αλλιώς
                Εμφάνισε “Άριστα”
            Τέλος_αν
        Τέλος_αν
    Τέλος_αν
Τέλος_αν
Τέλος Βαθμολογία
```

Πολλαπλή επιλογή Αλλιώς_αν

```
Αλγόριθμος Βαθμολογία
Διάβασε Βαθμός
Αν Βαθμός <=10 τότε
    Εμφάνισε  “Απορρίπτεται”
αλλιώς_αν Βαθμός <= 13 τότε
    Εμφάνισε “Σχεδόν Καλά”
αλλιώς_αν Βαθμός <= 16 τότε
    Εμφάνισε “Καλά”
αλλιώς_αν Βαθμός <= 18 τότε
    Εμφάνισε “Πολύ Καλά”
αλλιώς
    Εμφάνισε “Άριστα”
Τέλος_αν
Τέλος Βαθμολογία
```

Και για του δύο παραπάνω κώδικες φαίνεται και το λογικό διάγραμμα στο σχήμα



5.2.2.1-1 Λογικό διάγραμμα

5.2.2.2. Λύση από τα μεγάλα όρια στα μικρά

Σε αυτού του είδους τις ασκήσεις με όρια, συνήθως ξεκινάμε από τα μικρά όρια προς τα μεγάλα (10-13-16-18) χρησιμοποιώντας την ανισότητα < ή <=. Μπορούμε όμως να πάμε και από τα μεγάλα στα μικρά όρια (18-16-13-10) > ή >=. Η ίδια άσκηση θα μπορούσε να λυθεί και ως εξής:

Αλγόριθμος Βαθμολογία
Διάβασε Βαθμός
Αν Βαθμός >18 **τότε**
    Εμφάνισε "Άριστα"
αλλιώς_αν Βαθμός >16 **τότε**
    Εμφάνισε " Πολύ Καλά "
αλλιώς_αν Βαθμός > 13 **τότε**
    Εμφάνισε "Καλά"
αλλιώς_αν Βαθμός > 10 **τότε**
    Εμφάνισε " Σχεδόν καλά "

αλλιώς

Εμφάνισε " Απορρίπτεται "

Τέλος_αν

Τέλος Βαθμολογία

5.2.2.3. Λύση με σύνθετα Αν

Επίσης η άσκηση αυτή θα μπορούσε να λυθεί με συνδυασμό πολλών απλών δομών επιλογής χρησιμοποιώντας σύνθετα αν. Αυτός ο τρόπος είναι και ο τελευταίος που ενδείκνυται σε περιπτώσεις με όρια.

Αλγόριθμος Βαθμολογία
Διάβασε Βαθμός
Αν Βαθμός <=10 **τότε**
Εμφάνισε "Απορρίπτεται"
Τέλος_αν
Αν Βαθμός >10 **και** Βαθμός<= 13 **τότε**
Εμφάνισε "Σχεδόν Καλά"
Τέλος_αν
Αν Βαθμός >13 **και** Βαθμός<= 16 **τότε**
Εμφάνισε "Καλά"
Τέλος_αν
Αν Βαθμός >16 **και** Βαθμός<= 18 **τότε**
Εμφάνισε "Πολύ Καλά"
Τέλος_αν
Αν Βαθμός > 18 τότε
Εμφάνισε "Άριστα"
Τέλος_αν
Τέλος Βαθμολογία

5.3.Σύνθετα Αν

Πολλές φορές χρειάζεται μία συνθήκη να σπάσει σε περισσότερες από μία υποσυνθήκες. Τότε ελέγχουμε τις υποσυνθήκες μία προς μία και μετά τις συνδέουμε με την αντίστοιχη λογική πράξη. Οι λογικές πράξεις είναι τρείς: «**και**», «**ή**», «**όχι**». Το αποτέλεσμα της ολικής συνθήκης εξαρτάται από την λογική πράξη

που συνδέει τις υποσυνθήκες και από το αν αυτές είναι αληθείς ή ψευδείς σύμφωνα με τον παρακάτω πίνακα.

5.3-1 Πίνακας λογικών πράξεων

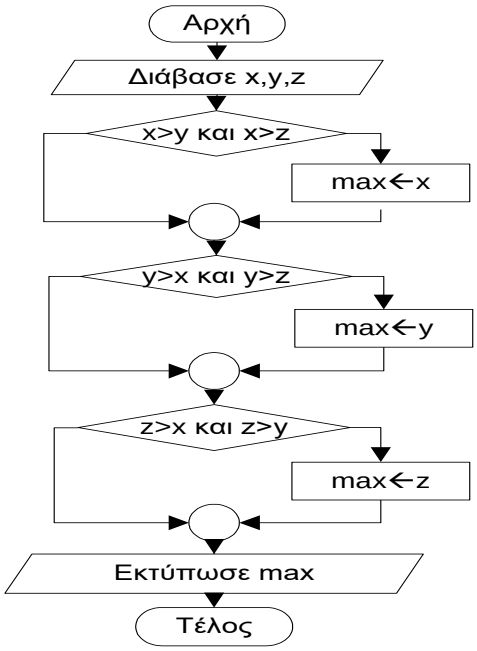
Πρόταση Α	Πρόταση Β	Α ή Β	Α και Β	όχι Α
Αληθής	Αληθής	Αληθής	Αληθής	Ψευδής
Αληθής	Ψευδής	Αληθής	Ψευδής	Ψευδής
Ψευδής	Αληθής	Αληθής	Ψευδής	Αληθής
Ψευδής	Ψευδής	Ψευδής	Ψευδής	Αληθής

5.3.1. Πρόβλημα Δ1

Να φτιαχτεί αλγόριθμος που να διαβάζει τρεις αριθμούς και να βρίσκει και να εκτυπώνει τον μεγαλύτερο.

Λύση: Εδώ στο συγκεκριμένο παράδειγμα για να μπορέσει ένας αριθμός να είναι ο μεγαλύτερος θα πρέπει να είναι μεγαλύτερος ταυτόχρονα και από τους άλλους δύο. Έτσι για παράδειγμα το x για να είναι ο μεγαλύτερος αριθμός από τους άλλους δύο θα πρέπει να είναι ταυτόχρονα μεγαλύτερο και από το y και από το z κ.ο.κ. Η λογική πράξη «και» για να γίνει αληθής θα πρέπει και οι δύο υποσυνθήκες να είναι αληθείς.

Αλγόριθμος μεγαλύτερος
Διάβασε χ, y, z
Αν (x>y) **και** (x>z) **τότε**
    max←x
Τέλος_αν
Αν (y>x) **και** (y>z) **τότε**
    max←y
Τέλος_αν
Αν (z>x) **και** (z>y) **τότε**
    max←z
Τέλος_αν
Εκτύπωσε max
Τέλος μεγαλύτερος



5.3.1-1 Λογικό διάγραμμα

5.4.Κλιμακωτή Χρέωση

Όταν έχω ασκήσεις με διαστήματα έχω δύο τρόπους να υπολογίσω το τελικό αποτέλεσμα ενός μεγέθους (πχ χρέωσης) ή κλιμακωτά ή μη κλιμακωτά. Έστω ότι έχω Ν μηνύματα sms και τις εξής πληροφορίες για την χρέωση τους:

Πλήθος μηνυμάτων sms	**Χρέωση ανά μήνυμα σε λεπτά**
N<=50	8
50<N<=100	6
100<N<=200	4
N>200	2

Τα μηνύματα αυτά μπορώ να τα χρεωθώ με δύο τρόπους

1ος τρόπος: Μη κλιμακωτά

- Έστω **N=60**. Επειδή το 60 βρίσκεται μεταξύ 50 και 100 η χρέωση για κάθε μήνυμα θα είναι 6 λεπτά του ευρώ. Άρα η χρέωση θα είναι: **χρ←60*6**.
- Έστω **N=120**. Επειδή το 120 βρίσκεται μεταξύ 100 και 200 η χρέωση για κάθε μήνυμα θα είναι 4 λεπτά του ευρώ. Άρα η χρέωση θα είναι: **χρ←120*4**.
- Αν **N=240** με την ίδια λογική η χρέωση θα είναι: **χρ←240*2**

2ος τρόπος: Κλιμακωτά

- Έστω **N=60**. Επειδή το 60 έχει περάσει το πρώτο όριο των 50 μηνυμάτων, εδώ η χρέωση προκύπτει ως έξής: Τα πρώτα 50 μηνύματα χρεώνονται με 8 λεπτά του ευρώ και τα επόμενα 10 (60-50) μηνύματα χρεώνονται με 6 λεπτά του ευρώ αφού βρίσκονται σε διαφορετική κλίμακα. Άρα η χρέωση θα είναι: **χρ←50*8+(60-50)*6**
- Έστω **N=120.** Επειδή το 120 έχει περάσει δύο όρια η χρέωση προκύπτει ως εξής: Τα πρώτα 50 μηνύματα έχουν χρέωση 8 λεπτά του ευρώ. Τα επόμενα 50 (100-50) μηνύματα έχουν χρέωση 6 λεπτά του ευρώ . Τα μηνύματα που περισσεύουν είναι 20 (120-100) και θα χρεωθούν με 4 λεπτά του ευρώ Άρα η χρέωση θα είναι: **χρ←50*8+50*6+(120-100)*4.**
- Αν **N=240** με την ίδια λογική η χρέωση θα είναι:

  χρ← 50*8+50*6+100*4+(240-200)*2

5.4.1. Πρόβλημα Ε1

Μια ηλεκτρική εταιρεία χρεώνει κλιμακωτά την ηλεκτρική κατανάλωση σύμφωνα με τον παρακάτω πίνακα:

- Τις πρώτες 200 μονάδες 25 δρχ/μονάδα
- Τις επόμενες 1000 μονάδες 40 δρχ/μονάδα
- Τις πέρα των 1200 μονάδων προς 50 δρχ/μονάδα

Να γίνει πρόγραμμα που θα δίνεται ο αριθμός των μονάδων που καταναλώθηκαν από έναν πελάτη και θα υπολογίζει και εμφανίζει το ποσό των χρημάτων που χρωστάει ο πελάτης στην ηλεκτρική Εταιρεία. (Εξετάσεις αποφοίτων ΤΕΛ 1997)

```
Αλγόριθμος ηλεκτρική_εταιρεία
Διάβασε μονάδες
Αν  μονάδες< =200 τότε
   λογαριασμός←μονάδες* 25
αλλιώς
   Αν μονάδες< =1200 τότε
      λογαριασμός←200*25+(μονάδες-200)* 40
   αλλιώς
      λογαριασμός←200*25+1000*40+(μονάδες-1200)* 50
   Τέλος_αν
Τέλος_αν
Εμφάνισε  λογαριασμός
Τέλος ηλεκτρική_εταιρεία
```

5.4.2. Πρόβλημα Ε2

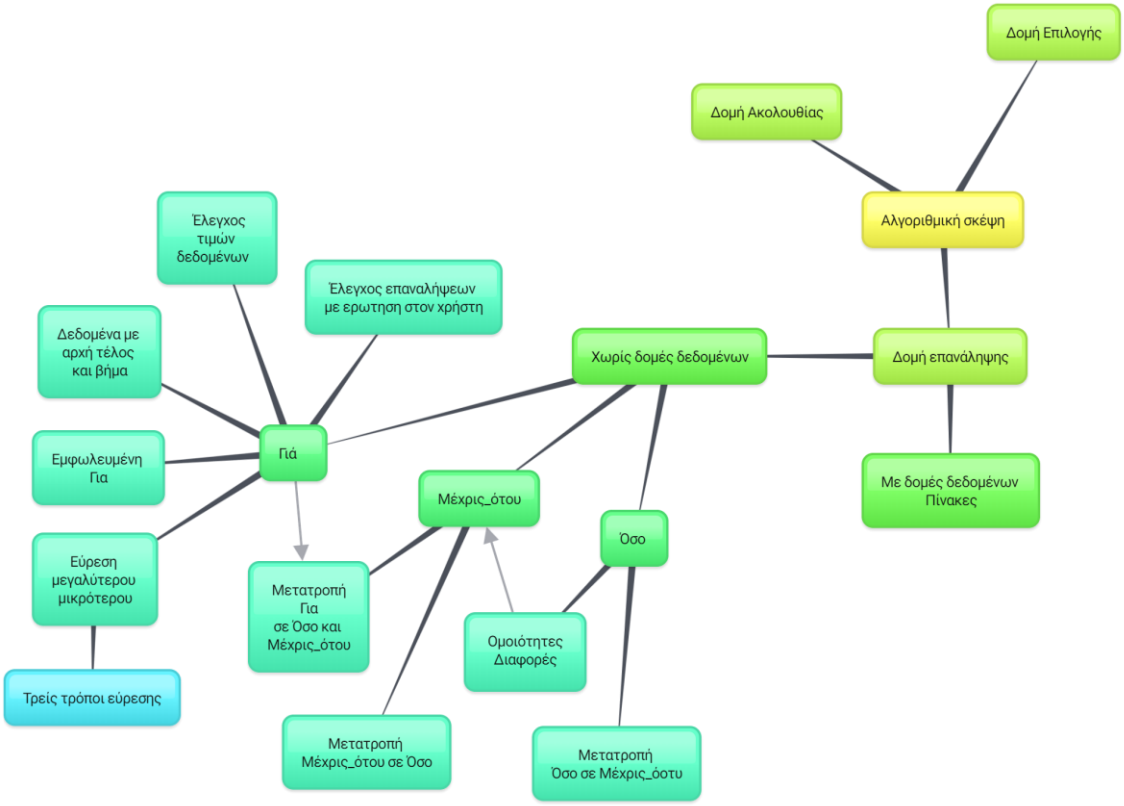
Να γραφεί αλγόριθμος που υπολογίζει και εμφανίζει το ποσό που πρέπει να πληρωθεί στον **ΟΤΕ** σαν αντίτιμο Ν τηλεφωνικών μονάδων (Ν γνωστό) όταν γνωρίζουμε ότι η χρέωση είναι κλιμακωτή και ότι:

- Οι πρώτες 100 μονάδες στοιχίζουν από 5 δρχ. η μία.
- Οι επόμενες 100 μονάδες στοιχίζουν από 4,5 δρχ. η μία.
- Οι επόμενες 100 μονάδες στοιχίζουν από 4 δρχ. η μία.
- Οι επόμενες μονάδες στοιχίζουν από 3,5 δρχ. η μία.

```
Αλγόριθμος ΟΤΕ
Διάβασε Ν
Αν Ν<=100 τότε
    ποσό←Ν*5
αλλιώς
    Αν Ν<=200 τότε
        ποσό←100*5+(Ν-100)*4,5
    αλλιώς
        Αν Ν<=300 τότε
            ποσό←100*5+100*4,5+ (Ν-200)*4
        αλλιώς
            ποσό←100*5+100*4,5+100*4+ (Ν-300)*3,5
        Τέλος_αν
    Τέλος_αν
Τέλος_αν
Εμφάνισε ποσό
Τέλος ΟΤΕ
```

ΚΕΦΑΛΑΙΟ 6

6. Δομή Επανάληψης



6-1 Νοητικός χάρτης δομής επανάληψης

Συχνά είναι απαραίτητο σ' έναν αλγόριθμο μια εντολή ή μια ομάδα εντολών να εκτελείται περισσότερες από μία φορές. Σε μια τέτοια περίπτωση δεν ξαναγράφονται οι ίδιες εντολές αλλά χρησιμοποιούνται οι **δομές της επανάληψης**. Σε αυτή την περίπτωση δεν έχω διαφορετικές μεταβλητές να παίρνουν διαφορετικά δεδομένα, αλλά περισσότερα του ενός δεδομένα να δίνουν τιμή στην ίδια μεταβλητή. Έτσι αν θέλω να διαβάσω τρείς αριθμούς, τότε χωρίς επανάληψη η εντολή είναι διάβασε α,β,γ , ενώ με επανάληψη η εντολή είναι διάβασε x, που θα εκτελεστεί τρεις φορές παίρνοντας το x σε διαφορετικές χρονικές στιγμές 3 διαφορετικά δεδομένα, το ένα μετά το άλλο.

6.1.Βασικές έννοιες επαναληπτικών διαδικασιών

6.1.1. Αθροιστής

```
S←0
Αρχή Επανάληψης
    Διάβασε x
    S←S+x
Τέλος Επανάληψης
```

Είναι μια μεταβλητή που χρησιμοποιούμε για να βρούμε το άθροισμα μιας σειράς αριθμών σε μια επαναληπτική διαδικασία. Έξω από την επανάληψη παίρνει την τιμή 0 ενώ μέσα στην επανάληψη συμμετέχει σε μια εντολή εκχώρησης της μορφής S←S+x.

ΕΠ.	x	S
		0
1	3	3
2	4	7
3	2	9

Στο δεξί μέρος το οποίο και προηγείται χρονικά από το αριστερό, αντικαθίσταται η προηγούμενη τιμή του αθροιστή , αθροίζεται με την τωρινή τιμή της μεταβλητής x και το αποτέλεσμα (που είναι μία σταθερά) εκχωρείται στο αριστερό μέρος ως η νέα τιμή του αθροιστή.

6.1.2. Μετρητής

```
M←0
Αρχή Επανάληψης
    M←M+1
Τέλος Επανάληψης
```

Αποτελεί περιορισμό του αθροιστή αφού κάθε φορά προσθέτουμε στη προηγούμενη τιμή του την μονάδα αυξάνοντας έτσι την επόμενη τιμή του κατά ένα. Έξω από την επανάληψη παίρνει την τιμή 0 ενώ μέσα στην επανάληψη συμμετέχει σε μια εντολή εκχώρησης της μορφής M←M+1.

ΕΠ.	M
	0
1	1
2	2
3	3

6.1.3. Πολλαπλασιαστής

```
γιν←1
Αρχή Επανάληψης
    Διάβασε x
    γιν←γιν*x
Τέλος Επανάληψης
```

Είναι μια μεταβλητή που χρησιμοποιούμε για να βρούμε το γινόμενο μιας σειράς αριθμών σε μια επαναληπτική διαδικασία. Έξω από την επανάληψη παίρνει την τιμή 1 ενώ μέσα στην επανάληψη συμμετέχει σε μια εντολή εκχώρησης της μορφής γιν←γιν*x. Στο δεξί μέρος το οποίο και προηγείται χρονικά από το αριστερό, αντικαθίσταται η προηγούμενη τιμή του πολλαπλασιαστή, πολλαπλασιάζεται με την τωρινή τιμή της μεταβλητής x και το αποτέλεσμα (που είναι μία σταθερά) εκχωρείται στο αριστερό μέρος ως η νέα τιμή του πολλαπλασιαστή.

ΕΠ.	x	γιν
		1
1	3	3
2	4	12
3	2	24

6.1.4. Διαφορά μετρητή – αθροιστή

Οι μεταβλητές που χρησιμοποιούνται σαν αθροιστές και σαν μετρητές πρέπει να αρχικοποιούνται (συνήθως 0) στην αρχή μιας δομής επανάληψης δηλαδή πριν αρχίσει η επανάληψη. Ο μετρητής είναι πάντα της μορφής Μ←Μ+σταθερά και συνήθως η σταθερά είναι το 1. Ο αθροιστής είναι πάντα της μορφής S←S+μεταβλητή και συνήθως η μεταβλητή είναι κάτι που ή διαβάζεται από το πληκτρολόγιο ή υπολογίζεται εκμεταλλευόμενο κάποια δεδομένα.

6.1.5. Πίνακας τιμών

Στις ασκήσεις που ζητείται πίνακας παρακολούθησης τιμών μεταβλητών, δίνεται ένας αλγόριθμος και ζητείται να παρακολουθήσουμε τις αλλαγές στις μεταβλητές του αλγορίθμου. Σχηματίζουμε λοιπόν έναν πίνακα με τόσες στήλες, όσες και οι μεταβλητές που υπάρχουν. Ως τελευταία στήλη χρησιμοποιούμε την στήλη που θα εμφανιστούν τα αποτελέσματα. Επίσης αν υπάρχει επαναληπτική διαδικασία όσο ή μεχρις_ότου συχνά μας βοηθάει να χρησιμοποιούμε ως πρώτη στήλη την στήλη που μετράμε τις επαναλήψεις. Εκτελούμε τις εντολές του αλγορίθμου μια προς μια, χρησιμοποιώντας τις τιμές που καταγράφονται στον πίνακα. Όταν μια μεταβλητή αλλάξει τιμή, αλλάζουμε γραμμή στον πίνακα και τοποθετούμε τη νέα τιμή. Κάθε στιγμή, ως τιμή κάθε μεταβλητής θεωρούμε την τελευταία (από κάτω) τιμή της στήλης που αντιστοιχεί στην μεταβλητή αυτή.
Δίνεται ο παρακάτω αλγόριθμος. Να παρουσιαστεί ο πίνακας τιμών των μεταβλητών και οι τιμές που θα εμφανιστούν.

Αλγόριθμος Άσκηση
α←3
β←α+14
γ ← α * β - 20
α ← (γ - α) div 3
β ← β mod α
γ ← γ - (α + β)
Εμφάνισε α, β, γ
Τέλος Άσκηση

6.1.5-1 Πίνακας τιμών

α	**β**	**γ**	**οθόνη**
3	17	31	
9	8	14	9 8 14

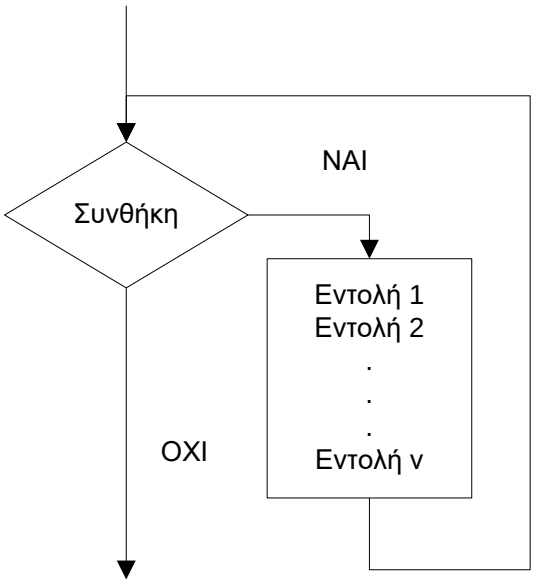
Έχω τρείς δομές επανάληψης. Την «Όσο» την «Μέχρις_ότου» και την «Για». Οι δύο πρώτες επαναληπτικές διαδικασίες χρησιμοποιούνται συνήθως σε μη

προκαθορισμένο αριθμό επαναλήψεων και η τρίτη επαναληπτική διαδικασία χρησιμοποιείται όταν γνωρίζουμε το πλήθος των επαναλήψεων.

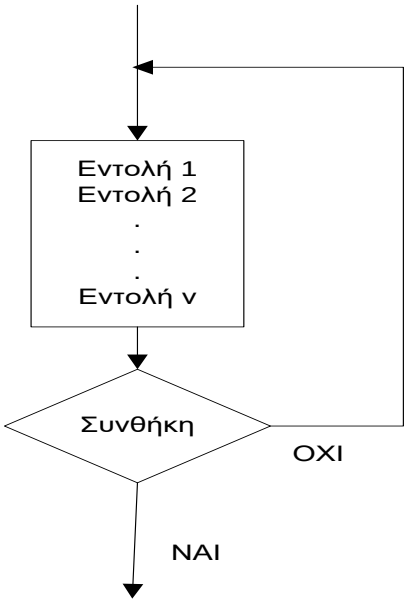
Κάποιες φορές έχουμε **μη προκαθορισμένο πλήθος επαναλήψεων**. Αυτό συμβαίνει όταν δεν έχουμε συγκεκριμένο αριθμό δεδομένων από την αρχή του προγράμματος. Όλοι μας έχουμε παίξει κάποιο παιχνίδι και μας έχει ρωτήσει στο τέλος αν θέλουμε να συνεχίσουμε ή όχι. Αν η απάντηση είναι ναι συνεχίζουμε, ενώ αν η απάντηση είναι όχι σταματάμε τη διαδικασία του παιχνιδιού. Ο προγραμματιστής την ώρα που έφτιαχνε το παιχνίδι δεν γνώριζε ο καθένας από εμάς πόσες φορές θα απαντήσει ναι, άρα θα πρέπει να συνεχίσει από την πίστα που έχασε την τελευταία φορά και ποια είναι εκείνη φορά που θα απαντήσουμε όχι.

Με την ίδια λογική υπάρχουν προβλήματα τα οποία για να απαντηθούν θα πρέπει να ρωτάμε κάθε φορά αν μια μεταβλητή έχει πάρει μία συγκεκριμένη τιμή και αν δεν έχει πάρει αυτή τη συγκεκριμένη τιμή τότε συνεχίζουμε την εκτέλεση των επαναλήψεων

6.2.Όσο-Μέχρις_ότου



6.2-1 Δομή Όσο



6.2-2 Δομή Μέχρις_ότου

Όσο συνθήκη **επανάλαβε**

ΟΕ

Τέλος_επανάληψης

Αρχή_επανάληψης

ΟΕ

Μέχρις_ότου <συνθήκη>

Και οι δύο αυτές επαναληπτικές διαδικασίες χρησιμοποιούνται συνήθως για μη προκαθορισμένο αριθμό επαναλήψεων. Είναι δύο διαφορετικές όψεις του ίδιου νομίσματος.

Στην όσο η συνθήκη προηγείται της ομάδας εντολών, η οποία και επαναλαμβάνεται όσο η συνθήκη είναι αληθής. Επειδή στην Όσο η συνθήκη είναι πριν την ομάδα εντολών, εάν η συνθήκη ευθύς εξαρχής είναι ψευδής, τότε ενδέχεται η ομάδα των εντολών να μην εκτελεστεί καμία φορά.

Στην Μέχρις_ότου η συνθήκη ακολουθεί την ομάδα εντολών, η οποία και επαναλαμβάνεται μέχρι η συνθήκη να γίνει αληθής, δηλαδή όσο η συνθήκη είναι ψευδής. Η ομάδα εντολών εκτελείται τουλάχιστον μία φορά.

Και στις δύο παραπάνω επαναληπτικές διαδικασίες, υπάρχει μία τιμή που ευθύνεται για το γεγονός ότι φεύγουμε από την επαναληπτική διαδικασία. Στην Όσο αυτή η τιμή είναι ξεκάθαρη και μπορούμε εύκολα να την ονομάσουμε. Είναι όταν δώσουμε την τιμή 0, τον πρώτο αρνητικό αριθμό ή τον πρώτο περιττό κ.ο.κ.

Η τιμή αυτή καλείται τιμή φρουρός. Η τιμή φρουρός, είναι κάτι εκτός ορίου δεδομένων, κάτι άχρηστο μόνο και μόνο για να φύγουμε από την επαναληπτική διαδικασία. Έτσι αν έχουμε βαθμούς στην κλίμακα από 0 έως 20, η τιμή φρουρός μπορεί να είναι κάτω από το μηδέν, ή πάνω από 20 π.χ. ή να έχει την τιμή -1, ή την τιμή 21. Όταν έχουμε για παράδειγμα να διαβάσουμε 10 βαθμούς, τότε η τιμή φρουρός θα είναι ο εντέκατος βαθμός.

6.2.1. Πρόβλημα Ζ1

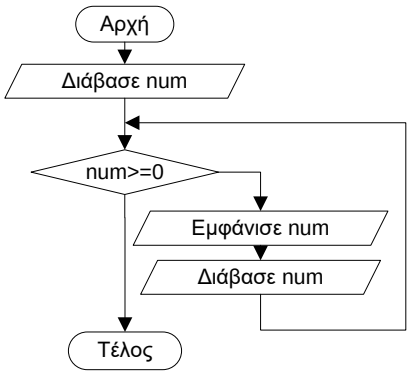
Να διαβάζονται και να εμφανίζονται αριθμοί από το πληκτρολόγιο μέχρι να δοθεί αρνητικός αριθμός.

6.2.1.1. Λύση με Όσο

Πρέπει να τονιστεί ιδιαίτερα ότι η μεταβλητή ή οι μεταβλητές που περιέχονται στη συνθήκη, πρέπει να έχουν μία **αρχική τιμή** πριν εκτελεστεί η δομή της επανάληψης, η οποία (τιμή) πρέπει να μεταβάλλεται μέσα στη δομή επανάληψης, γιατί σε διαφορετική περίπτωση η συνθήκη δεν θα γίνει ποτέ **ψευδής** και θα προκύψει **ατέρμων βρόχος**.

Αυτό μας οδηγεί στο γεγονός, να διαβάζουμε την μεταβλητή num λίγο πριν μπούμε στην επαναληπτική διαδικασία για τον πρώτο αριθμό, αλλά και λίγο πριν βγούμε από αυτήν για κάθε έναν από τους επόμενους αριθμούς. Αυτό είναι αναγκαίο, αλλά είναι και ταυτόχρονα ένα τέχνασμα που μας αποτρέπει να χρησιμοποιήσουμε την τελευταία τιμή που θα πάρει το num (δηλαδή την τιμή φρουρό) στην ομάδα των εντολών. Εδώ η τιμή φρουρός είναι ο πρώτος αρνητικός αριθμός. Έτσι εμφανίζονται όλοι οι αριθμοί που είναι μεγαλύτεροι ή ίσοι με το μηδέν, αλλά όχι η τελευταία αρνητική τιμή της μεταβλητής num.

Αλγόριθμος θετικοί_αριθμοί
Διάβασε num
Όσο num >=0 **επανάλαβε**
    Εμφάνισε num
    Διάβασε num
Τέλος_επανάληψης
Τέλος θετικοί_αριθμοί



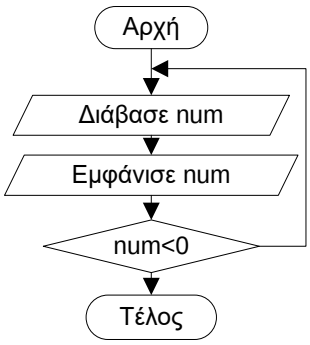
6.2.1.1-1 Λύση με Όσο

Συμπέρασμα: Όποτε δεν χρειάζεται η τιμή φρουρός να συμμετέχει στην επαναληπτική διαδικασία, χρησιμοποιούμε την όσο με το γνωστό τέχνασμα «διαβάζω λίγο πριν μπω και λίγο πριν βγω»

6.2.1.2. Λύση με Μέχρις_ότου

Ο έλεγχος της μεταβλητής γίνεται στο τέλος της ομάδας των εντολών, οπότε το διάβασμα της μεταβλητής num, μπορεί να γίνει μία φορά στην αρχή της ομάδας των εντολών μέσα στην επανάληψη. Γενικότερα οι εντολές που βρίσκονται μεταξύ της εντολής που δίνει τιμή στην μεταβλητή (Διάβασε num) και του ελέγχου της (Μέχρις_ότου num<0), εκτελούνται την τελευταία φορά. Η τελευταία τιμή θα συμμετάσχει στην ομάδα των εντολών. Έτσι θα εμφανιστεί και ο αρνητικός αριθμός, που αναγκαστικά θα δοθεί για να φύγουμε από την επαναληπτική διαδικασία.

Αλγόριθμος θετικοί_αριθμοί
Αρχή_επανάληψης
Διάβασε num
Εμφάνισε num
Μέχρις_ότου num<0
Τέλος θετικοί_αριθμοί



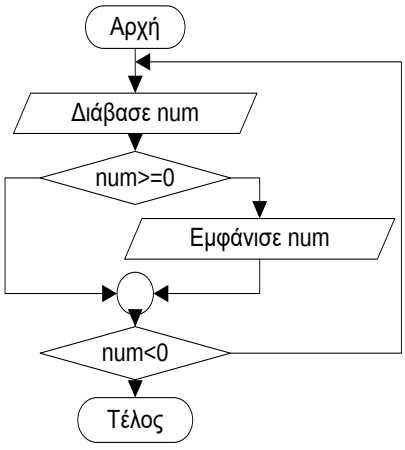
6.2.1.2-1 Λύση με Μέχρις_ότου

Συμπέρασμα: Όποτε χρειάζεται η τελευταία τιμή να συμμετέχει στην επαναληπτική διαδικασία χρησιμοποιούμε την μέχρις_ότου, διαβάζοντας μία φορά την μεταβλητή που θέλουμε στην αρχή της επανάληψης.

6.2.1.3. Λύση με Μέχρις_ότου χωρίς να συμμετέχει η τιμή φρουρός

Αν επιβάλλεται για κάποιο λόγο η άσκηση να λυθεί με «**μεχρις_ότου**» και να μην θέλουμε την τελευταία τιμή να συμμετέχει στην ομάδα των εντολών, τότε βάζουμε την ομάδα αυτή των εντολών σε μια δομή επιλογής, με συνθήκη ακριβώς αντίθετη από αυτή της μεχρις_ότου. Εδώ η «**μεχρις_ότου**» έχει την συνθήκη num<0 και η δομή της επιλογής που βάζω μέσα της την ομάδα των εντολών (μοναδική εντολή **Εμφάνισε** num) έχει την συνθήκη num>=0.

Αλγόριθμος θετικοί_αριθμοί
Αρχή_επανάληψης
Διάβασε num
Αν num>=0 **τότε**
Εμφάνισε num
Τέλος_αν
Μέχρις_ότου num<0
Τέλος θετικοί_αριθμοί



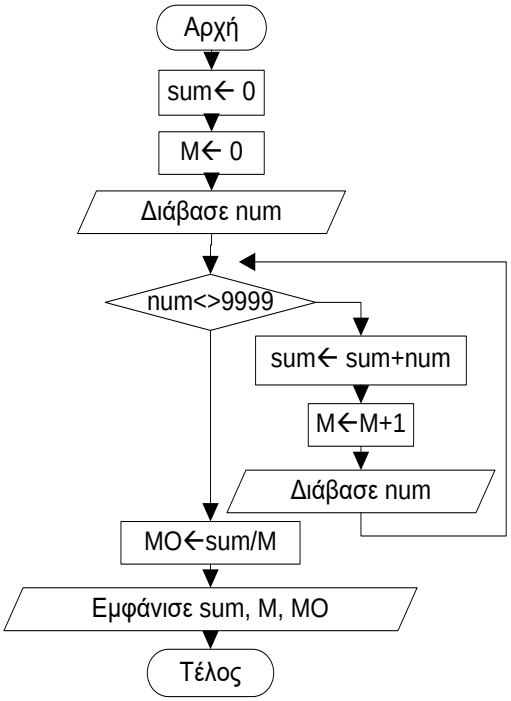
6.2.1.3-1 Μέχρις_ότου - Επιλογή

6.2.2. Πρόβλημα Ζ2

Να φτιαχτεί αλγόριθμος που να διαβάζει αριθμούς μέχρι να δοθεί ο αριθμός 9999. Να υπολογιστεί και να εμφανιστεί το άθροισμα, το πλήθος και ο μέσος όρος αυτών χωρίς το 9999.

Λύση με Όσο: Η τιμή φρουρός εδώ είναι το 9999. Αφού η εκφώνηση της άσκησης μας λέει ότι θέλει το άθροισμα το πλήθος και τον μέσο όρο των αριθμών χωρίς το 9999, συμπεραίνουμε ότι δεν θέλουμε να συμμετάσχει στην ομάδα των εντολών η τιμή φρουρός, άρα λύνουμε την άσκηση χρησιμοποιώντας την επαναληπτική διαδικασία «΄**Οσο**», με το γνωστό μοντέλο «διαβάζω λίγο πριν μπω και λίγο πριν βγω» . Πλήθος σε επαναληπτική διαδικασία σημαίνει μετρητής, που έξω από την επανάληψη πρέπει να μηδενιστεί. Άθροισμα σε επαναληπτική διαδικασία σημαίνει αθροιστής που έξω από την επανάληψη πρέπει επίσης να μηδενιστεί. Τόσο ο αθροιστής όσο και ο μετρητής μέσα στην επανάληψη αλλάζουν διαρκώς τιμή. Μετά την επανάληψη έχουν και οι δύο σταθεροποιήσει τις τιμές τους. Ο μέσος όρος δεν μπαίνει ποτέ μέσα στην επανάληψη καθώς είναι το τελικό άθροισμα δια του τελικού πλήθους. Έτσι παίρνει τιμή μετά την επαναληπτική διαδικασία.

Αλγόριθμος πρόβλημα
sum←0
M←0
Διάβασε num
Όσο num <>9999 **επανάλαβε**
    sum←sum+num
    M←M+1
    Διάβασε num
Τέλος_επανάληψης
MO←sum/M
Εμφάνισε sum, M, MO
Τέλος πρόβλημα



6.2.2-1 Λογικό διάγραμμα

Αν οι αρχικές τιμές του χ είναι 5,4,6, 9999 τότε ο πίνακας τιμών είναι ο κάτωθι:

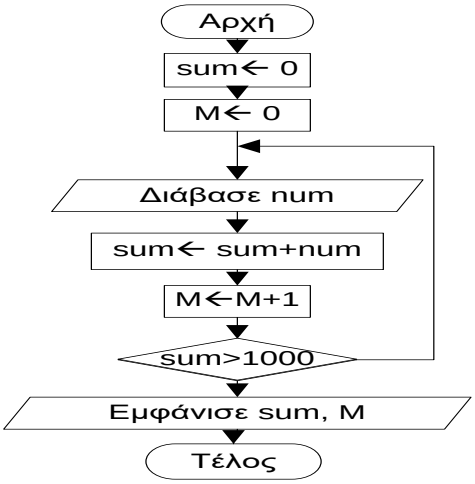
Επαν.	**Sum**	**M**	**x**	**MO**	**Οθόνη**
	0	0	5		
1	5	1	4		
2	9	2	6		
3	15	3	9999		
				5	15 3 5

6.2.3. Πρόβλημα Ζ3

Να φτιαχτεί αλγόριθμος που να διαβάζει αριθμούς μέχρι το άθροισμά τους να ξεπεράσει την τιμή 1000. Να εμφανιστεί το άθροισμα και το πλήθος των αριθμών που δόθηκαν.

Λύση με Μέχρις_ότου: Εδώ η τελευταία τιμή που θα δοθεί, είναι ο πρώτος αριθμός που όταν προστεθεί στο άθροισμα των προηγούμενων, το συνολικό άθροισμα θα ξεπεράσει την τιμή 1000. Αν είχα δηλαδή ως αρχικές τιμές του x=400, 500, 150 , η τελευταία τιμή θα ήταν το 150. Πώς όμως θα ξεπεράσω την τιμή 1000 αν στο προηγούμενο άθροισμα που είναι το 900 δεν προσθέσω και το 150. Καταλαβαίνω ότι το 150 πρέπει να προστεθεί, πρέπει να συμμετέχει στην ομάδα των εντολών, άρα χρησιμοποιώ την «μέχρις_ότου».

Αλγόριθμος Άθροισμα
sum←0
M←0
Αρχή_επανάληψης
  Διάβασε num
  sum←sum+num
  M←M+1
Μέχρις_ότου sum>1000
Εμφάνισε sum, M
Τέλος Άθροισμα



6.2.3-1 Λογικό διάγραμμα

Αν οι αρχικές τιμές του χ είναι 400, 500, 150 τότε ο πίνακας τιμών είναι ο κάτωθι:

Επαν.	Sum	M	x	Οθόνη
0	0	0		
1	400	1	400	
2	900	2	500	
3	1050	3	150	
				1050 3

6.2.4. Όσο vs Μέχρις_ότου

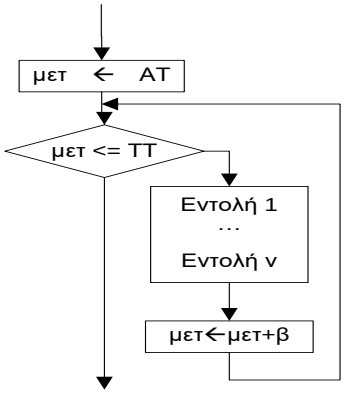
Γενικότερα χρησιμοποιούμε την «**Όσο**» όταν υπάρχει ξεκάθαρη τιμή φρουρός, οπότε με το τέχνασμα «διαβάζω λίγο πριν μπω και λίγο πριν βγω», η τελευταία τιμή δεν επεμβαίνει στην ομάδα των εντολών. Όποτε θέλω η τελευταία τιμή να επεμβαίνει στην ομάδα των εντολών τότε πάω με «**μεχρις_ότου**».

Υπάρχουν ασκήσεις που λύνονται μόνο με «**Όσο**», υπάρχουν ασκήσεις που λύνονται μόνο με «**μέχρις_ότου**», αλλά υπάρχουν και ασκήσεις που λύνονται και με «**Όσο**» και με « **μέχρις_ότου**», οπότε εκεί είναι προσωπική εκτίμηση του καθενός, η επιλογή της επαναληπτικής διαδικασίας.

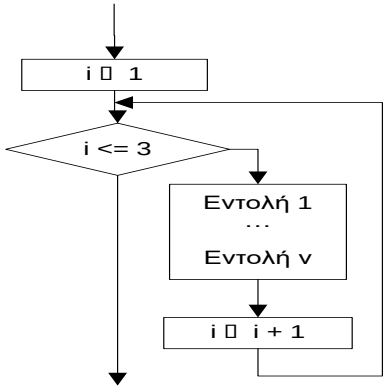
6.3.Για μετ από ΑΤ μέχρι ΤΤ με_βήμα β

Χρησιμοποιείται μόνο όταν γνωρίζουμε τον συγκεκριμένο αριθμό επαναλήψεων. Το πλήθος των επαναλήψεων εξαρτάται από τρεις παραμέτρους:

1. την αρχική τιμή ενός μετρητή ΑΤ,
2. την τελική τιμή του μετρητή ΤΤ
3. και την τιμή του βήματος β που αυτή θα προστίθεται σε κάθε επανάληψη.



Η «**Για**» είναι μια «**Όσο**» με έναν ενσωματωμένο μετρητή. Δεν είναι αναγκαίο να αρχικοποιηθεί και να αυξηθεί ο μετρητής «μετ» με ειδικές εντολές. Αυτό γίνεται αυτόματα από την επαναληπτική διαδικασία «Για». Όταν το βήμα είναι ένα δεν χρειάζεται να το γράφουμε. Έτσι όταν δεν γράφουμε καθόλου το βήμα, εννοείται ότι είναι ένα. Κάθε φορά προσθέτουμε στην αρχική τιμή της μεταβλητής το βήμα, μέχρι να φτάσουμε ή να ξεπεράσουμε την τελική τιμή. Το πλήθος των επαναλήψεων εξαρτάται από τις διαφορετικές τιμές του μετρητή χωρίς την τελευταία που ευθύνεται για το γεγονός ότι φεύγουμε από την επανάληψη (ψευδοτιμή).Όταν η αρχική τιμή του μετρητή είναι ένα και το βήμα είναι ένα τότε η τελική τιμή του μετρητή μου δείχνει και το πλήθος των επαναλήψεων. Η τελευταία είναι και η πιο συνηθισμένη περίπτωση χρησιμοποίησης της επαναληπτικής διαδικασίας «**Για**».



Για i **από** 1 **μέχρι** 3
εντολές
Τέλος_επανάληψης

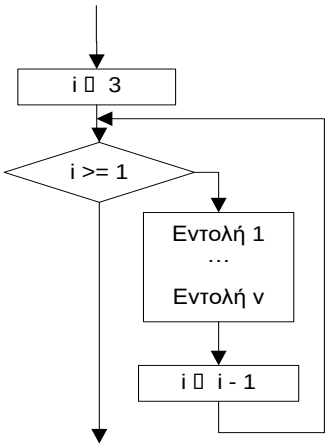
6.3-1 Λογικό διάγραμμα

Παρακάτω θα δούμε κάποιες περιπτώσεις που μπορεί να συναντήσουμε:
Για i **από** 1 **μέχρι** 3 i=1, 2, 3 η ομάδα των εντολών γίνεται 3 φορές.
Για i **από** 1 **μέχρι** 5 **με_βήμα** 2 i=1, 3, 5 η ομάδα των εντολών γίνεται 3 φορές.
Για i **από** 2 **μέχρι** 9 **με_βήμα** 3 i=2, 5, 8 η ομάδα των εντολών γίνεται 3 φορές.
Για i **από** 3 **μέχρι** 4 **με_βήμα** 0.5 i = 3, 3.5, 4 η ομάδα των εντολών θα εκτελεστεί 3 φορές.

Το βήμα β δεν μπορεί να είναι μηδέν διότι τότε η ομάδα εντολών εκτελείται επ' άπειρον.
Π.Χ. Για i **από** 1 **μέχρι** 3 **με_βήμα** 0

Στην περίπτωση κατά την οποία ΑΤ=ΤΤ ο βρόχος εκτελείται ακριβώς μία φορά.
Π.Χ. Για i **από** 3 **μέχρι** 3 **με_βήμα** 1 i=3 η ομάδα των εντολών γίνεται 1 φορά.
Στην περίπτωση κατά την οποία ΑΤ>ΤΤ με β>0 ο βρόχος δεν εκτελείται καμία φορά.

Π.Χ. Για i **από** 3 **μέχρι** 1
Είναι δυνατόν το βήμα β να είναι αρνητικός ακέραιος αρκεί να ισχύει ότι ΑΤ>ΤΤ.
Π.Χ. Για i **από** 3 **μέχρι** 1 **με_βήμα** -1 i=3,2,1 η ομάδα των εντολών γίνεται 3 φορές.



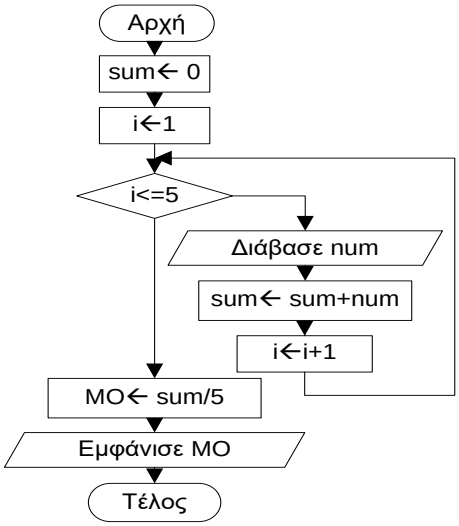
6.3-2 Λογικό διάγραμμα

6.3.1. Πρόβλημα Η1

Να φτιαχτεί αλγόριθμος που να διαβάζει πέντε αριθμούς από το πληκτρολόγιο. Να βρεθεί και να εμφανιστεί ο μέσος όρος τους.

Λύση: Εδώ η επαναληπτική διαδικασία που θα χρησιμοποιηθεί είναι η Για καθώς γνωρίζουμε από την αρχή το πλήθος των δεδομένων, άρα και το πλήθος των επαναλήψεων. Ο μετρητής δεν χρειάζεται για την εύρεση του μέσου όρου καθώς ξέρουμε από την αρχή ότι οι αριθμοί είναι πέντε. Αν η αρχική τιμή του μετρητή i είναι ένα και το βήμα είναι ένα τότε η τελική τιμή που είναι πέντε, εκφράζει και το πλήθος των επαναλήψεων.

```
Αλγόριθμος αριθμοί
sum←0
Για i από 1 μέχρι 5
    Διάβασε num
    sum←sum+num
Τέλος_επανάληψης
ΜΟ←sum/5
Εμφάνισε  ΜΟ
Τέλος αριθμοί
```



6.3.1-1 Λογικό διάγραμμα

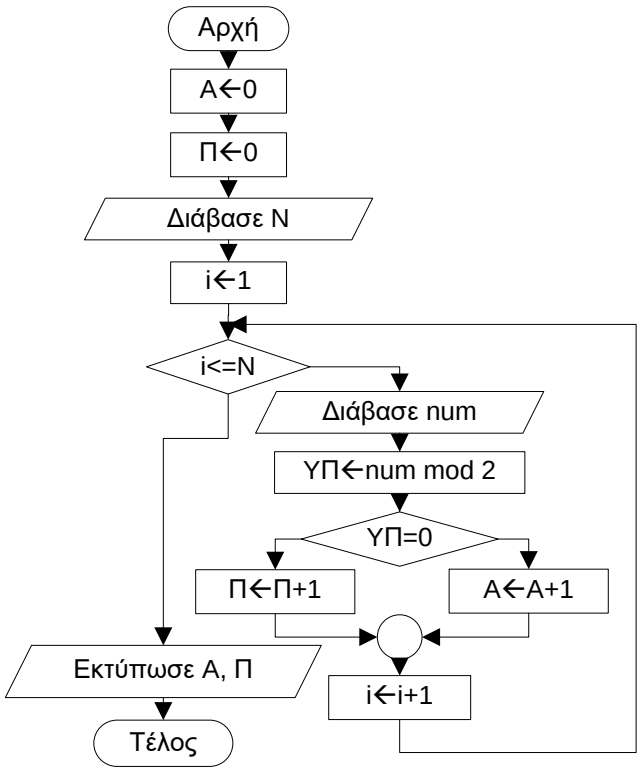
6.3.2. Πρόβλημα Η2

Να διαβαστούν Ν αριθμοί. Να βρεθεί και να εμφανιστεί πόσοι από αυτούς είναι άρτιοι και πόσοι περιττοί.

N	i	num	ΥΠ	Α	Π	Οθόνη
5				0	0	
	1	5	1		1	
	2	4	0	1		
	3	6	0	2		
	4	9	1		2	
	5	7	1		3	
						2,3

6.3.2-1 Πίνακας τιμών

Αλγόριθμος
Άρτιοι_Περιττοί
Α←0
Π←0
Διάβασε Ν
Για i **από** 1 **μέχρι** Ν
  Διάβασε num
  ΥΠ←num mod 2
  Αν ΥΠ=0 **τότε**
    Α←Α+1
  αλλιώς
    Π←Π+1
  Τέλος_αν
Τέλος_επανάληψης
Εμφάνισε Α, Π
Τέλος Άρτιοι_Περιττοί



6.3.2-2 Λογικό διάγραμμα

Παραπάνω φαίνεται ο κώδικας, το λογικό διάγραμμα και ο πίνακας τιμών, έστω για 5 αριθμούς, δηλαδή για num= 5,4,6,9,7.

6.4. «Για» σε δεδομένα με αρχή, τέλος και συγκεκριμένο βήμα.

Κάποιες φορές τα δεδομένα δεν είναι τυχαία, αλλά έχουν μία αρχή, ένα τέλος και ένα βήμα. Τότε δεν χρειάζεται να διαβάσουμε δεδομένα από το πληκτρολόγιο με την εντολή διάβασε όπως κάνουμε συνήθως, αλλά τα δεδομένα προκύπτουν από μαθηματική σκέψη με μία επαναληπτική διαδικασία «Για». Το i εδώ εκτός από το γεγονός ότι ελέγχει τις επαναλήψεις, μπαίνει και μέσα στην ομάδα των εντολών παίζοντας τον ρόλο των δεδομένων. Η αρχική τιμή του i και του βήματος ενδέχεται να είναι διαφοροποιημένες από την μονάδα, ανάλογα με τα δεδομένα του προβλήματος.

6.4.1. Πρόβλημα Θ1

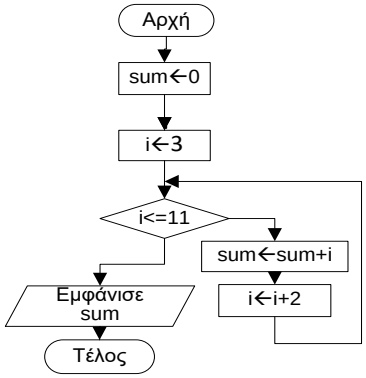
Να Εμφανιστεί το άθροισμα S= 3 + 5 + 7 + 9 + 11

Λύση: Εδώ δεν πρόκειται για το άθροισμα 5 τυχαίων αριθμών αλλά για το άθροισμα συγκεκριμένης σειράς αριθμών με αρχική τιμή τρία, τελική τιμή έντεκα, ενώ ο επόμενος αριθμός είναι πάντα ο προηγούμενος αριθμός αυξημένος κατά δύο. Έτσι

οι εντολές Διάβασε x και sum←sum+x αντικαθίστανται από την εντολή sum←sum+i , καθώς το i παίρνει τιμές από 3 μέχρι 11 με_βήμα 2.

Αλγόριθμος Άθροισμα
sum←0
Για i **από** 3 **μέχρι** 11 **με_βήμα** 2
    sum ← sum + i
Τέλος_επανάληψης
Εμφάνισε sum
Τέλος Άθροισμα

i	sum
	0
3	3
5	8
7	15
9	24
11	35



6.4.1-1 Λογικό διάγραμμα

6.4.2. Πρόβλημα Θ2

Να γίνει πρόγραμμα που να υπολογίζει και να εμφανίζει για όλες τις ακέραιες γωνίες από 0 μέχρι 360 μοίρες, σε πόσα ακτίνια αντιστοιχούν. Ο τύπος μετατροπής είναι: ακτίνια /π=μοίρες/180

Λύση: Ότι θα κάναμε για την μετατροπή μιας γωνίας από μοίρες σε ακτίνια τώρα πρέπει γίνει για 361 γωνίες. Η πρώτη γωνία είναι 0 μοίρες, η τελευταία γωνία είναι 360 μοίρες, και όλες οι ενδιάμεσες γωνίες αυξάνονται κατά μία μοίρα κάθε φορά. Άρα δεν χρειάζεται να διαβάζω τις μοίρες από το πληκτρολόγιο κάθε φορά με την εντολή διάβασε, αφού μπορούν οι μοίρες να δημιουργηθούν από μια επαναληπτική διαδικασία «Για» ξεκινώντας με αρχική τιμή 0, καταλήγοντας σε τελική τιμή 360 και προχωρώντας με βήμα 1.

ΠΡΟΓΡΑΜΜΑ RAD
ΣΤΑΘΕΡΕΣ
    Π=3.14
ΜΕΤΑΒΛΗΤΕΣ
    ΠΡΑΓΜΑΤΙΚΕΣ: ακτίνια
ΑΡΧΗ
ΓΙΑ i **ΑΠΟ** 0 **ΜΕΧΡΙ** 360
    ακτίνια←i / 180 * Π
    ΓΡΑΨΕ ‘Η ακέραια γωνία με’, i, ‘ μοίρες είναι ’, ακτίνια, ‘ακτίνια’
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ

6.5.Μετατροπή της «Για» σε «Όσο» και «Μέχρις_ότου»

Η «**Για»** είναι ουσιαστικά μία «**Όσο»** με έναν ενσωματωμένο μετρητή που παίρνει αρχική τιμή και αυξάνεται αυτόματα κατά ένα συγκεκριμένο βήμα. Αρά αντίστροφα θα μπορούσαμε να πούμε ότι η «**Όσο»** είναι μια «**Για»** χωρίς ενσωματωμένο μετρητή που πρέπει εμείς να του δώσουμε και αρχική τιμή αλλά και το βήμα.

Μια δομή «**Για»** μετατρέπεται πάντα σε «**Όσο»** και σε «**Μέχρις_ότου»**. Το αντίστροφο συμβαίνει μόνο όταν η «**Όσο»** ή η «**Μέχρις_ότου»** δουλεύουν σε γνωστό αριθμό επαναλήψεων. Για να γίνει μετατροπή από «**Για»** σε «**Όσο»**, πρέπει να απομονωθούν:

α) η αρχική τιμή της μεταβλητής ελέγχου (τιμή μετά το «**από»**),

β) η συνθήκη τερματισμού (τιμή μετά το «**μέχρι»**)

γ) το βήμα της (τιμή μετά το «**με_βήμα»**)

Επειδή η «**Όσο»** και η «**Μέχρις_ότου»** μπορούν να λάβουν μόνο την συνθήκη τερματισμού, αναγκαστικά στην μετατροπή της «**Για»** σε «**Όσο»** και «**Μεχρις_ότου**» υπάρχουν δύο ακόμα εντολές στον κώδικα:

Α) Η αρχική τιμή της μεταβλητής ελέγχου πριν την επαναληπτική διαδικασία (εδώ i←1)

Β) Η αύξηση της μεταβλητής ελέγχου κατά το βήμα, ως τελευταία εντολή μέσα στην επαναληπτική διαδικασία (εδώ i←i+1)

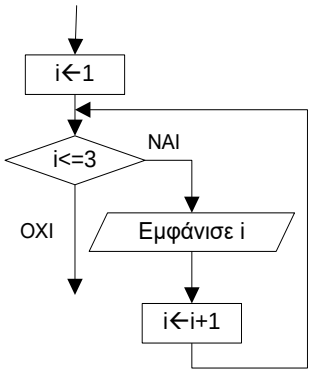
6.5.1. Πρόβλημα Ι1

Να εμφανιστούν στη οθόνη του υπολογιστή οι τιμές 1,2,3.

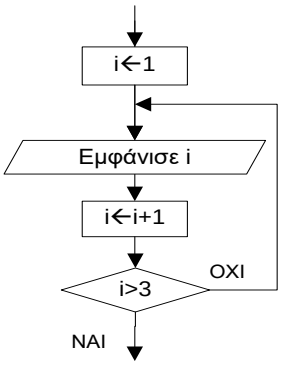
```
Για i από 1 μέχρι 3
    Εμφάνισε i
Τέλος_επανάληψης
```

```
i←1
Όσο i<=3 επανάλαβε
    Εμφάνισε i
    i←i+1
Τέλος_επανάληψης
```

```
i←1
Αρχή_επανάληψης
    Εμφάνισε i
    i←i+1
Μέχρις_ότου i>3
```



6.5.1-1 Όσο - Για



6.5.1-2 Μέχρις_ότου

6.5.2. Πρόβλημα Ι2

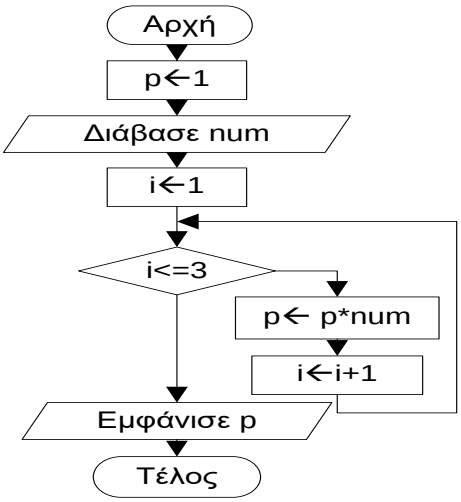
Να φτιαχτεί αλγόριθμος που να διαβάζει έναν αριθμό από το πληκτρολόγιο και να βρίσκει και να εμφανίζει την τρίτη δύναμη του αριθμού αυτού και με τις τρεις επαναληπτικές διαδικασίες.

Λύση: Γνωρίζουμε ότι η τρίτη δύναμη ενός αριθμού προκύπτει από το γινόμενο 5*5*5, δηλαδή από τον πολλαπλασιασμό του αριθμού με τον εαυτό του τρείς φορές. Έξω από την επαναληπτική διαδικασία, δίνουμε αρχική τιμή 1 σε έναν πολλαπλασιαστή και διαβάζουμε μία φορά τον αριθμό που θέλουμε να υψώσουμε. Μέσα στην επαναληπτική διαδικασία πολλαπλασιάζουμε τον αριθμό αυτό με τον εαυτό του τρείς φορές. Το λογικό διάγραμμα για την Όσο και την Για είναι ίδιο.

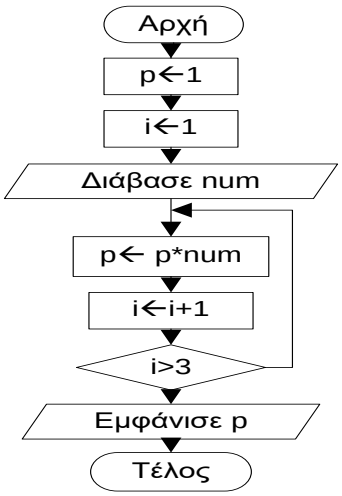
```
Αλγόριθμος δύναμη
p←1
Διάβασε num
Για i από 1 μέχρι 3
    p←p*num
Τέλος_επανάληψης
Εμφάνισε p
Τέλος δύναμη
```

```
Αλγόριθμος δύναμη
p←1
Διάβασε num
i←1
Όσο num<=3 Επανάλαβε
        p←p*num
        i← i +1
Τέλος_επανάληψης
Εμφάνισε p
Τέλος δύναμη
```

```
Αλγόριθμος δύναμη
p←1
Διάβασε num
i←1
Αρχή_επανάληψης
        p←p*num
        i← i +1
Μέχρις_ότου i>3
Εμφάνισε p
Τέλος δύναμη
```



6.5.2-1 Τρίτη δύναμη Όσο-Για



6.5.2-2 Τρίτη δύναμη Μέχρις_ότου

6.6.Μετατροπή «Μέχρις_ότου» σε «Όσο»

Οι εντολές στη δομή επανάληψης ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ … ΜΕΧΡΙΣ_ΟΤΟΥ, εκτελούνται όσο η <συνθήκη> μετά το Μέχρις_ότου είναι Ψευδής, ενώ στην δομή επανάληψης ΟΣΟ…ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ, εκτελούνται όσο η <συνθήκη> ανάμεσα στο ΟΣΟ και το ΕΠΑΝΑΛΑΒΕ είναι Αληθής. Γι' αυτό, κατά την μετατροπή από την μια δομή επανάληψης στην άλλη, αρκεί να γράφουμε την άρνηση της <συνθήκη> της πρώτης στη δεύτερη ή να προτάξουμε τον τελεστή ΟΧΙ στην συνθήκη. Επίσης, η ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ … ΜΕΧΡΙΣ_ΟΤΟΥ εκτελεί τουλάχιστον μια φορά όλες τις εντολές της, γι' αυτό όταν την μετατρέπουμε στην ΟΣΟ…ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ πρέπει, όλες τις εντολές που περιέχει, να τις γράψουμε μια φορά πριν την ΟΣΟ…ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ και άλλη μια φορά μέσα σ' αυτήν.

Αρχή_επανάληψης	<εντολές>
<εντολές>	**ΟΣΟ Όχι** <συνθήκη> **ΕΠΑΝΑΛΑΒΕ**
Μέχρις_ότου <συνθήκη>	<εντολές>
	ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

6.7.Μετατροπή «Όσο» σε «Μέχρις_ότου»

Η δομή επανάληψης ΟΣΟ … ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ εκτελείται όσο η <συνθήκη> ανάμεσα στο ΟΣΟ και το ΕΠΑΝΑΛΑΒΕ είναι Αληθής, ενώ η δομή επανάληψης ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ … ΜΕΧΡΙΣ_ΟΤΟΥ εκτελείται όσο η συνθήκη είναι μετά το Μέχρις_ότου Ψευδής. Γι' αυτό κατά την μετατροπή από την μια δομή στην άλλη αρκεί να γράψουμε την άρνηση της <συνθήκης> της πρώτης στη δεύτερη ή να προτάξουμε τον τελεστή ΟΧΙ στην συνθήκη. Επίσης, η ΟΣΟ…ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ μπορεί να μην εκτελεστεί καμία φορά σε αντίθεση με την ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ…ΜΕΧΡΙΣ_ΟΤΟΥ που θα εκτελεστεί τουλάχιστον μια φορά, γι' αυτό πριν την δομή ΑΡΧΗ_ ΕΠΑΝΑΛΗΨΗΣ…ΜΕΧΡΙΣ_ΟΤΟΥ θα χρησιμοποιηθεί μια εντολή ελέγχου, που αν ισχύει η <συνθήκη> θα εκτελεστεί η ΑΡΧΗ_ ΕΠΑΝΑΛΗΨΗΣ…ΜΕΧΡΙΣ_ΟΤΟΥ.

ΟΣΟ <συνθήκη> **ΕΠΑΝΑΛΑΒΕ** <εντολές> **ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ**	**ΑΝ** <συνθήκη> **ΤΟΤΕ** **ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ** <εντολές> **ΜΕΧΡΙΣ_ΟΤΟΥ ΟΧΙ** <συνθήκη> **ΤΕΛΟΣ_ΑΝ**

*Ως μη βέλτιστη λύση (η συνθήκη ελέγχεται δύο φορές), για την ίδια μετατροπή μπορεί να δοθεί και η παρακάτω:

ΟΣΟ <συνθήκη> **ΕΠΑΝΑΛΑΒΕ** <εντολές> **ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ**	**ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ** **ΑΝ** <συνθήκη> **ΤΟΤΕ** <εντολές> **ΤΕΛΟΣ_ΑΝ** **ΜΕΧΡΙΣ_ΟΤΟΥ ΟΧΙ** <συνθήκη>

6.8.Εύρεση μεγαλύτερου και μικρότερου στοιχείου

6.8.1. 1ος Τρόπος σκέψης

Θεωρώ τον πρώτο αριθμό ως τον μεγαλύτερο και τον τοποθετώ σε μια μεταβλητή (συνήθως την ονομάζω max). Συγκρίνω κάθε έναν από τους επόμενους

αριθμούς με τον μεγαλύτερο εκείνης της στιγμής (δηλ. με την τρέχουσα τιμή της μεταβλητής max) και αν βρω τον αριθμό αυτό μεγαλύτερο, αντικαθιστώ την τρέχουσα τιμή της μεταβλητής max με τον αριθμό αυτό. Το ίδιο ισχύει και για την εύρεση μικρότερου με μικρές παραλλαγές.

Παράδειγμα: Εύρεση μεγαλύτερου σε μια σειρά αριθμών

Έστω ότι οι τιμές του Χ είναι οκτώ και είναι οι εξής: **X = 3, 5, 4, 7, 2, 9, 1, 8**
Συνολικά οι αριθμοί είναι οκτώ. Ο τρόπος σκέψης εύρεσης του max με αναγκάζει να σπάσω τα δεδομένα στον πρώτο και στα επόμενα ως εξής:
Θεωρώ τον πρώτο ως τον μεγαλύτερο και τον τοποθετώ σε μια μεταβλητή που συνήθως την ονομάζω max (από το maximum που σημαίνει μέγιστη τιμή).
Άρα έστω **max=3**

Συγκρίνω κάθε έναν από τους επόμενους αριθμούς με την τιμή του max εκείνης της στιγμής και σε περίπτωση που τον βρίσκω μεγαλύτερο τότε το υποθετικό max αλλάζει και δέχεται την νέα τιμή. Σε περίπτωση που το βρίσκω μικρότερο τότε η τιμή του max είναι σωστή για τους αριθμούς που έχω ελέγξει μέχρι εκείνη την στιγμή.

Άρα για τους επόμενους επτά αριθμούς κάνω την παραπάνω διαδικασία που είναι σαφώς μια επαναληπτική διαδικασία που γίνεται επτά φορές για το συγκεκριμένο παράδειγμα, ή Ν-1 φορές γενικότερα, αφού Ν-1 είναι οι επόμενοι αριθμοί εκτός του πρώτου.

1. Είναι το 5> max; δηλαδή είναι το 5>3; Ναι. Άρα αλλάζω την τιμή του max με τον αριθμό που βρήκα μεγαλύτερο και έτσι το max γίνεται **5**.
2. Είναι το 4> max ; δηλαδή είναι το 4>5; Όχι. Άρα το max μέχρι αυτή την στιγμή είναι σωστό για τους αριθμούς που έχω ελέγξει και δεν του αλλάζω τιμή.
3. Είναι το 7>max; δηλαδή είναι το 7>5; Ναι. Άρα αλλάζω την τιμή του max με τον αριθμό που βρήκα μεγαλύτερο και έτσι το max γίνεται **7**
4. Είναι το 2> max; δηλαδή είναι το 7>7; Όχι. Άρα το max μέχρι αυτή την στιγμή είναι σωστό για τους αριθμούς που έχω ελέγξει και δεν του αλλάζω τιμή.
5. Είναι το 9> max; δηλαδή είναι το 9>7; Ναι. Άρα αλλάζω την τιμή του max με τον αριθμό που βρήκα μεγαλύτερο και έτσι το max γίνεται **9**
6. Είναι το 1>max; δηλαδή είναι το 1>9; Όχι. Άρα το max μέχρι αυτή την στιγμή είναι σωστό για τους αριθμούς που έχω ελέγξει και δεν του αλλάζω τιμή.

7. Είναι 8 > max; δηλαδή είναι το 8>9; Όχι. Άρα το max μέχρι αυτή την στιγμή είναι σωστό για τους αριθμούς που έχω ελέγξει και δεν του αλλάζω τιμή.

Η μεταβλητή max άλλαξε πολλές τιμές που δεν αντιπροσωπεύουν τον μεγαλύτερο αριθμό, αλλά η τελευταία είναι και η σωστή. Όλες οι προηγούμενες τιμές είναι μπορεί να μην είναι σωστές αλλά είναι και απαραίτητες για να έχουμε στο τέλος το σωστό αποτέλεσμα.

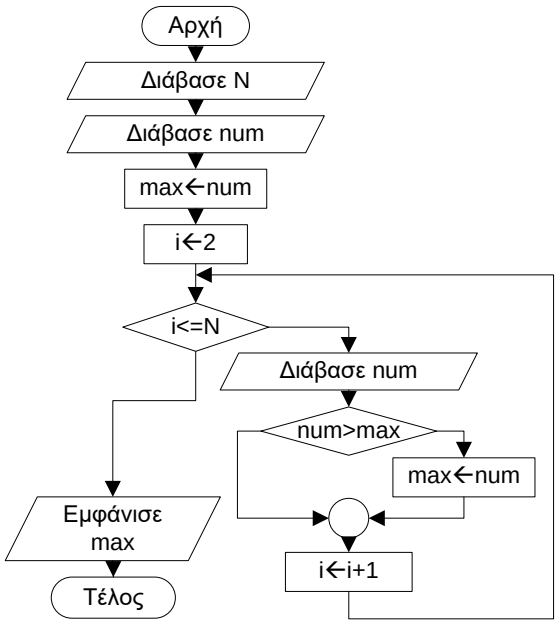
Η επαναληπτική διαδικασία γίνεται 7 φορές, για όλους τους αριθμούς εκτός από τον πρώτο. Το i δηλαδή θα έπρεπε να πάρει τιμές από το 1-7. Όμως προτιμάμε το i να ξεκινάει από δύο και να πάρει τελικά τιμές από το 2-8 γιατί μας βοηθάει, ή συχνά είναι απαραίτητο το i να δείχνει τον αύξοντα αριθμό του δεδομένου. Έτσι όταν το i είναι δύο θα δείχνει τον δεύτερο αριθμό που θα ασχοληθώ μαζί του, όταν το i είναι τρία τον τρίτο αριθμό κ.ο.κ

6.8.1.1.Πρόβλημα Κ1

Να φτιαχτεί αλγόριθμος που να διαβάζει Ν αριθμούς και να βρίσκει και να εμφανίζει τον μεγαλύτερο.

Λύση: Ο τρόπος σκέψης με αναγκάζει να σπάσω τα δεδομένα στο πρώτο και στα επόμενα Ν-1 δεδομένα. Έξω από την επανάληψη διαβάζω τον πρώτο αριθμό και υποθέτω ότι είναι ο μεγαλύτερος τοποθετώντας τον ως τιμή στην μεταβλητή max. Μέσα στην επαναληπτική διαδικασία, διαβάζω κάθε έναν από τους επόμενους αριθμούς, δηλ τον δεύτερο, τον τρίτο κ.ο.κ και συγκρίνω τους αριθμούς αυτούς με τον max εκείνης της στιγμής. Ο μετρητής της επανάληψης ξεκινάει από την τιμή δύο και καταλήγει στην τιμή Ν για να είναι όμοιος με την αύξουσα σειρά των αριθμών που διαβάζω. Όταν το i είναι δύο τότε διαβάζω τον δεύτερο αριθμό, όταν είναι τρία διαβάζω τον τρίτο αριθμό κ.ο.κ.

Αλγόριθμος μεγαλύτερος
Διάβασε N
Διάβασε num
max←num
Για i **από** 2 **μέχρι** N
    Διάβασε num
    Αν num>max **τότε**
        max←num
    Τέλος_αν
Τέλος_επανάληψης
Εμφάνισε max
Τέλος μεγαλύτερος



6.8.1.1-1 Λογικό διάγραμμα

Σημείωση: Πολλές φορές μου είναι αδύνατον, πολύ δύσκολο ή απλά με βολεύει να μην ακολουθήσω τον κλασικό τρόπο σκέψης και σπάσω τα δεδομένα στο πρώτο και στα επόμενα.

Τότε διαβάζω όλα τα δεδομένα μέσα στην επαναληπτική διαδικασία και ακολουθώ κάποιο τέχνασμα, κάποιον έξυπνο τρόπο σκέψης ώστε να συμπεριφερθώ διαφορετικά για το πρώτο δεδομένο και διαφορετικά για καθένα από τα υπόλοιπα.

Ας μην ξεχνάμε ότι για το πρώτο δεδομένο τοποθετώ αδιαμφισβήτητα την τιμή του στο max και στο min, ενώ για κάθε ένα από τα υπόλοιπα δεδομένα πρώτα ελέγχω αν το δεδομένο είναι μεγαλύτερο από το max εκείνης της στιγμής, ή μικρότερο από το min εκείνης της στιγμής και μετά τοποθετώ την τιμή του στο max και στο min.

Έτσι υπάρχουν άλλοι δύο τρόποι εύρεσης μεγαλύτερου και μικρότερου.

6.8.2. 2ος Τρόπος σκέψης

Όταν έχω όρια στα δεδομένα μου (π.χ. βαθμούς σε κλίμακα 0-20) βάζω στην μεταβλητή max έναν πολύ μικρό αριθμό εκτός ορίων δεδομένων (π.χ. το -1) ή στην μεταβλητή min έναν πολύ μεγάλο αριθμό εκτός ορίων δεδομένων (π.χ. το 21) έξω από την επαναληπτική διαδικασία. Στην επαναληπτική διαδικασία συγκρίνω όλους τους αριθμούς της σειράς (συμπεριλαμβάνεται και ο πρώτος) με τον μεγαλύτερο ή

τον μικρότερο εκείνης της στιγμής και ακολουθώ την διαδικασία του πρώτου τρόπου.

Την πρώτη φορά ωστόσο ο βαθμός που διαβάζω θα είναι σίγουρα μεγαλύτερος του max, αφού του έχουμε δώσει μια τιμή εκτός ορίων πολύ χαμηλή. Ακόμα και μηδέν να έχει γράψει ο μαθητής, το μηδέν είναι μεγαλύτερο του -1. Επίσης ο πρώτος βαθμός που διαβάζω θα είναι σίγουρα μικρότερος από το min καθώς του έχουμε δώσει μια τιμή εκτός ορίων πολύ υψηλή. Ακόμα και είκοσι να έχει γράψει ο μαθητής, το είκοσι είναι μικρότερο του 21. Έτσι για τον πρώτο βαθμό ισχύει ότι ακριβώς και στον πρώτο τρόπο εύρεσης μεγαλύτερου μικρότερου, δηλαδή αφού τον διαβάσω τον τοποθετώ σίγουρα ως πρώτη τιμή του max και του min, καθώς η σύγκριση αν ο βαθμός αυτός είναι μεγαλύτερος του max και μικρότερος του min, ισχύει πάντα.

6.8.2.1. Πρόβλημα Κ2

Να φτιαχτεί αλγόριθμος που να διαβάζει Ν βαθμούς (0-20) και να βρίσκει και να εμφανίζει τον μεγαλύτερο και τον μικρότερο από αυτούς.

Λύση: Τοποθετώντας ως max το -1 είναι βέβαιο ότι την πρώτη φορά όποιος και αν είναι ο βαθμός που θα διαβάσω θα είναι μεγαλύτερος από το max και έτσι θα τοποθετηθεί ως τιμή στην μεταβλητή max. Τοποθετώντας ως min το 21 είναι βέβαιο ότι την πρώτη φορά όποιος και αν είναι ο βαθμός που θα διαβάσω θα είναι μικρότερος από το min και έτσι θα τοποθετηθεί ως τιμή στην μεταβλητή min. Δηλαδή με το τέχνασμα αυτό τοποθετώ αφού διαβάσω τον πρώτο βαθμό αδιαμφισβήτητα στο min και στο max όπως ακριβώς και στον πρώτο τρόπο. Είναι σαν αν μην υπάρχει καθόλου η συνθήκη αν βαθμός >max και αν βαθμός < min αντίστοιχα για τον πρώτο βαθμό. Για τους επόμενους βαθμούς ισχύουν οι παραπάνω συγκρίσεις και ενδέχεται να αλλάξει η τιμή του max και του min, όπως στον πρώτο τρόπο.

```
Αλγόριθμος Βαθμοί
Διάβασε Ν
max←-1
min←21
Για i από 1 μέχρι Ν
    Διάβασε βαθμός
    Αν βαθμός>max τότε
```

```
            max←βαθμός
        Τέλος_αν
        Αν βαθμός<min τότε
            min←βαθμός
        Τέλος_αν
    Τέλος_επανάληψης
    Εμφάνισε max, min
    Τέλος Βαθμοί
```

6.8.3. 3ος Τρόπος σκέψης

Όταν δεν έχω όρια στα δεδομένα μου αλλά δεν θέλω να τα σπάσω στο πρώτο και στα επόμενα τότε κάνω τα εξής: Διαβάζω όλα τα δεδομένα μέσα στην επαναληπτική διαδικασία και απλά διαφοροποιώ τις εντολές όταν το i = 1 και όταν το i <> 1. Όταν το i = 1 αναφέρομαι στον πρώτο δεδομένο και άρα το τοποθετώ αδιαμφισβήτητα στο max και στο min ενώ σε περίπτωση όπου i <> 1 αναφέρομαι σε κάθε ένα από τα επόμενα δεδομένα, οπότε και συγκρίνω κάθε έναν από αυτά με τον μεγαλύτερο και τον μικρότερο εκείνης της στιγμής (δηλ. με την τρέχουσα τιμή της μεταβλητής max, min) και αν βρω το δεδομένο αυτό μεγαλύτερο-μικρότερο, αντικαθιστώ την τρέχουσα τιμή της μεταβλητής max-min με το δεδομένο αυτό.

Σε περίπτωση που δεν έχω επαναληπτική διαδικασία **Για**, και άρα δεν υπάρχει ένας ενσωματωμένος μετρητής, αλλά μία **Όσο** ή μια **μέχρις_ότου** πρέπει να χρησιμοποιηθεί ένας μετρητής, ακόμα και αν δεν χρειάζεται από τα δεδομένα του προβλήματος, που θα αυξάνεται κάθε φορά που θα διαβάζω έναν δεδομένο. Όταν ο μετρητής αυτός είναι ένα τότε αναφέρομαι στο πρώτο δεδομένο, ενώ αν είναι διαφορετικός από ένα, αναφέρομαι σε κάθε ένα από τα επόμενα δεδομένα.

6.8.3.1. Πρόβλημα Κ3

Να φτιαχτεί αλγόριθμος που να διαβάζει Ν αριθμούς και να βρίσκει και να εμφανίζει τον μεγαλύτερο

```
Αλγόριθμος max_min
Διάβασε N
Για i από 1 μέχρι N
    Διάβασε x
```

```
Αν i=1 τότε
    max←x
    min←x
αλλιώς
    Αν x>max τότε
        max←x
    Τέλος_αν
    Αν x<min τότε
        min←x
    Τέλος_αν
Τέλος_αν
Τέλος_επανάληψης
Εκτύπωσε max
Τέλος max_min
```

6.9.Έλεγχος τιμών δεδομένων:

Πολλές φορές θέλουμε να κάνουμε έλεγχο τιμών δεδομένων. Αυτό δεν θα μπορούσε να γίνει μόνο με μια δομή επιλογής, γιατί τότε δεν θα δίναμε την δυνατότητα στον χρήστη να ξαναδώσει την σωστή τιμή. Αναγκαστικά χρειαζόμαστε μια επαναληπτική διαδικασία που δεν θα μπορούσε να είναι η Για, γιατί δεν γνωρίζουμε πόσες φορές ο χρήστης θα κάνει λάθος και ποια φορά θα δώσει την σωστή τιμή. Για αυτό τον λόγο χρησιμοποιούμε ή **όσο** ή **μέχρις_ότου**, αντικαθιστώντας την εντολή διάβασε με ένα σετ από εντολές όπως φαίνεται παρακάτω:

```
Αρχή_επανάληψης
Διάβασε x
Αν x εκτός ορίων τότε
    Εμφάνισε “Λάθος Δώσε ξανά”
Τέλος_αν
Μέχρις_ότου x εντός ορίων
```

```
Διάβασε x
Όσο x εκτός ορίων επανάλαβε
    Εμφάνισε “Λάθος Δώσε ξανά”
    Διάβασε x
Τέλος_επανάληψης
```

6.9.1. Πρόβλημα Λ1

Να διαβαστεί μία μέτρηση που θα πρέπει να είναι υποχρεωτικά θετική ή μηδέν

```
Αρχή_επανάληψης
    Διάβασε x
    Αν x<0 τότε
    Εμφάνισε "Λάθος. Δώσε ξανά"
    Τέλος_αν
Μέχρις_ότου x>=0
```

```
Διάβασε x
Όσο x<0 επανάλαβε
    Εμφάνισε "Λάθος. Δώσε ξανά"
    Διάβασε x
Τέλος_επανάληψης
```

6.9.2. Πρόβλημα Λ2

Να διαβαστεί ένας βαθμός τετράμηνου που θα πρέπει να είναι υποχρεωτικά μεταξύ μηδέν και είκοσι

```
Αρχή_επανάληψης
Διάβασε β
Αν (β<0) ή (β>20) τότε
    Εμφάνισε "Λάθος. Δώσε ξανά"
Τέλος_αν
Μέχρις_ότου (β>=0) και (β<=20)
```

```
Διάβασε β
Όσο (β<0) ή (β>20) επανάλαβε
    Εμφάνισε "Λάθος. Δώσε ξανά"
    Διάβασε β
Τέλος_επανάληψης
```

6.9.3. Πρόβλημα Λ3

Να διαβαστεί μια απάντηση που θα πρέπει να είναι υποχρεωτικά "Σ" για σωστό ή "Λ" για λάθος.

```
Αρχή_επανάληψης
    Διάβασε απ
    Αν (απ<>"Σ" ) και (απ<>"Λ") τότε
        Εμφάνισε "Λάθος. Δώσε ξανά"
    Τέλος_αν
Μέχρις_ότου (απ="Σ") ή (απ="Λ")
```

```
Διάβασε απ
Όσο (απ<>"Σ" ) και (απ<>"Λ") επανάλαβε
    Εμφάνισε "Λάθος. Δώσε ξανά"
    Διάβασε β
Τέλος_επανάληψης
```

6.10. Έλεγχος επαναλήψεων με ερώτηση στον χρήστη

Η τιμή φρουρός δεν είναι πάντα ο καλύτερος τρόπος για να διακόψουμε μια επαναληπτική διαδικασία. Είναι πιο όμορφο να ρωτάμε τον χρήστη αν θέλει να δώσει ακόμα έναν βαθμό παρά να τον αναγκάζουμε να δώσει μία τιμή εκτός ορίων (π.χ. το -1) για να φύγει από την επανάληψη. Υπάρχει μία τριάδα εντολών που

ακολουθεί την ομάδα των εντολών και ελέγχει τις επαναλήψεις με την παρακάτω μορφή:

Αρχή_επανάληψης
Εντολή 1
Εντολή 2
.
.
.
Εντολή Ν
Εμφάνισε "Θέλεις να συνεχίσεις; (ναι/όχι)"
Διάβασε απ
Μέχρις_ότου απ="όχι"

(Εναλλακτικά θα μπορούσε να μπει **Μέχρις_ότου** απ<>"ναι")

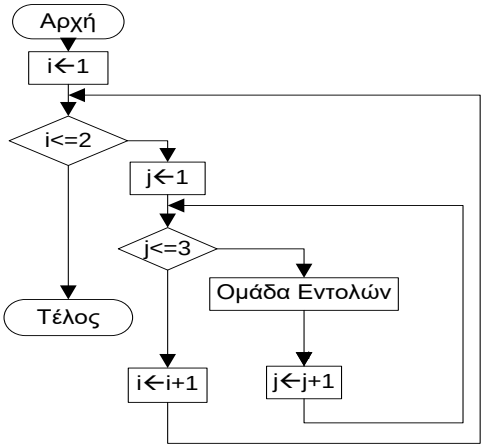
6.10.1. Πρόβλημα Μ1

Να γίνει αλγόριθμος ο οποίος θα μετατρέπει και εμφανίζει ένα χρηματικό ποσό από δραχμές σε ευρώ (€), όταν η ισοτιμία είναι 340,75 δραχμές για 1 ευρώ. θέλουμε ο αλγόριθμος να εκτελείται επαναληπτικά τόσες φορές όσες ο χρήστης το επιθυμεί. Συγκεκριμένα μετά από κάθε μετατροπή θα εμφανίζεται μήνυμα που θα ρωτάει τον χρήστη αν θέλει να μετατρέψει και κάποιο καινούριο ποσό. Αν η απάντηση είναι ΝΑΙ τότε η διαδικασία να επαναλαμβάνεται. Σε περίπτωση που η απάντηση είναι ΟΧΙ ο αλγόριθμος να τερματίζει.

Αλγόριθμος μετατροπή
Αρχή_επανάληψης
Διάβασε ποσό_δρχ
Ευρώ←ποσό_δρχ/340.75
Εμφάνισε ευρώ
Εμφάνισε "Θέλεις να μετατρέψεις νέο ποσό; (ΝΑΙ/ΟΧΙ)"
Διάβασε απ
Μέχρις_ότου απ<>"ΝΑΙ"
Τέλος μετατροπή

6.11. Διπλή Επαναληπτική Διαδικασία

Εδώ βλέπουμε το λογικό διάγραμμα και τον κώδικα για μια διπλή επαναληπτική διαδικασία με δύο δείκτες **i** και **j**.



Για i **από** 1 **μέχρι** 2
  Για j **από** 1 **μέχρι** 3
    Ομάδα Εντολών
  Τέλος_επανάληψης
Τέλος_επανάληψης

6.11-1 Διπλή επαναληπτική διαδικασία

i	j	Εκτέλεση ομάδας εντολών
1	1	1
	2	2
	3	3
2	1	4
	2	5
	3	6

Για την τιμή του i = 1 ο δείκτης j θα πάρει όλες τις τιμές του, δηλαδή 1,2,3 και για την τιμή του i = 2 ο δείκτης j θα πάρει όλες τις τιμές του, δηλαδή 1,2,3. Άρα η ομάδα των εντολών που βρίσκεται μέσα στην διπλή επανάληψη θα εκτελεστεί έξι φορές. Αυτό προκύπτει εύκολα από την πράξη 2 * 3 = 6.

Γενικότερα Στην διπλή επαναληπτική διαδικασία έχω δύο δείκτες, τον εξωτερικό δείκτη (συνήθως i) και τον εσωτερικό δείκτη (συνήθως j). Ο εσωτερικός δείκτης μεταβάλλεται γρηγορότερα από τον εξωτερικό και μάλιστα είναι αναγκασμένος να πάρει όλες τις διαφορετικές τιμές του για κάθε τιμή του εξωτερικού δείκτη. Το πλήθος των επαναλήψεων της ομάδας εντολών που βρίσκεται μέσα σε μια διπλή επαναληπτική διαδικασία είναι ίσο με το γινόμενο των μέγιστων τιμών των δύο δεικτών. (Αν i=3 και j=4 η ομάδα εντολών θα εκτελεστεί 12 φορές).

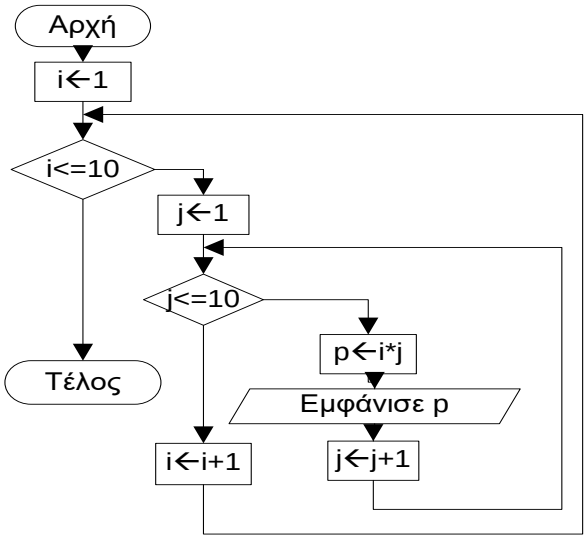
6.11.1. Πρόβλημα Ν1

Να γίνει αλγόριθμος που να βρίσκει και να εμφανίζει την προπαίδεια.

Λύση: Τα αποτελέσματα της προπαίδειας είναι 1*1, 1*2, …,1*10. Μπορώ να εμφανίσω αυτά τα αποτελέσματα λοιπόν χρησιμοποιώντας δύο δείκτες. Ο πρώτος

θα παραμένει σταθερά ένα και ο δεύτερος θα παίρνει τιμές από ένα ως δέκα. Στην συνέχεια ο πρώτος θα παραμένει σταθερά δύο και ο δεύτερος θα παίρνει τιμές από ένα ως δέκα δημιουργώντας τα αποτελέσματα 2*1, 2*2, …,2*10, κ.ο.κ. Σύμφωνα λοιπόν με τα παραπάνω η διαδικασία του πολλαπλασιασμού των δύο αριθμών μπορεί να προκύψει από μια διπλή επανάληψη, όπου τον ρόλο του πρώτου αριθμού θα παίξει το i και τον ρόλο του δεύτερου αριθμού το j. Και οι δύο δείκτες θα παίρνουν τιμές από ένα έως δέκα.

```
Αλγόριθμος προπαίδεια
Για i από 1 μέχρι 10
    Για j από 1 μέχρι 10
        p← i*j
        Εμφάνισε i, 'x', j, '=', p
    Τέλος_επανάληψης
Τέλος_επανάληψης
Τέλος προπαίδεια
```



6.11.1-1 Προπαίδεια

ΚΕΦΑΛΑΙΟ 7

7. Πίνακες

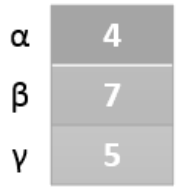
7.1.Τρόποι εισαγωγής δεδομένων

Να φτιαχτεί αλγόριθμος που να διαβάζει τρεις αριθμούς. Να υπολογιστεί το άθροισμά τους.

7.1.1. 1ος τρόπος - Χωρίς επανάληψη

Διάβασε α, β, γ
S ← α+β+γ

Εδώ το καλό είναι ότι τα δεδομένα καταλαμβάνουν διαφορετικές θέσεις μνήμης και παραμένουν στις μεταβλητές μέχρι το τέλος του προγράμματος. Όταν έχω πολλά δεδομένα τότε πρέπει να έχω διαφορετικές ονομασίες για κάθε ένα από αυτά. Πχ αν είχα δέκα δεδομένα θα έπρεπε ο παραπάνω κώδικας να γίνει:

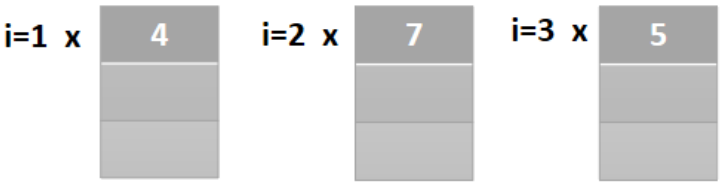


Διάβασε α1, α2, α3, α4, α5, α6, α7, α8, α9, α10
S←α1+α2+α3+α4+α5+α6+α7+α8+α9+α10

Φανταστείτε τι θα γινότανε αν είχα εκατό δεδομένα! Το αρνητικό λοιπόν αυτού του τρόπου είναι ότι όταν έχω πολλά δεδομένα τότε έχω και αντίστοιχα μεγάλο κώδικα.

7.1.2. 2ος τρόπος - Με επανάληψη

S←0
Για i **από** 1 **μέχρι** 3
    Διάβασε x
    S←S+x
Τέλος_Επανάληψης



Εδώ τα δεδομένα καταλαμβάνουν μία θέση μνήμης (μια μεταβλητή) και οι τιμές των δεδομένων διαδέχονται η μία την άλλη μέχρι στην μεταβλητή να αποθηκευτεί η τελευταία τιμή.

Για i=1 το x θα πάρει την πρώτη τιμή του δηλαδή x=4.

Για i=2 το x θα αλλάξει τιμή και θα πάρει την δεύτερη τιμή του δηλαδή x=7.

Για i=3 το x θα αλλάξει τιμή και θα πάρει την τρίτη τιμή του δηλαδή x=5.

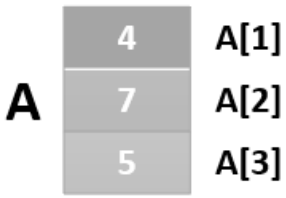
Θα χαθούν λοιπόν όλες οι τιμές των δεδομένων εκτός από την τελευταία. Αυτό με αναγκάζει να εκτελώ όλα τα ζητούμενα του προβλήματος για κάθε διαφορετική τιμή του x πριν αυτό αλλάξει τιμή και πάρει την επόμενη τιμή του.

Επίσης σε αυτές τις ασκήσεις **όλα γίνονται στην ίδια επανάληψη, για κάθε συγκεκριμένη χρονική στιγμή που το x έχει μια συγκεκριμένη τιμή**. Αν χρησιμοποιούσα περισσότερες επαναλήψεις, τότε θα συνέβαινε το εξής παράδοξο για x=4, 7, 5 : Στην πρώτη επανάληψη θα διάβαζα όλες τις τιμές του x και θα κράταγα μόνο την τελευταία. Στην δεύτερη επανάληψη ο αθροιστής S θα άθροιζε τρείς φορές την τελευταία τιμή του x, δηλαδή την τιμή 5 και το αποτέλεσμα θα ήταν 5+5+5=15 και όχι 4+7+5=16.

Το καλό είναι ότι το μέγεθος του κώδικα δεν μεταβάλλεται αν έχω περισσότερα δεδομένα με ανάλογα περισσότερες επαναλήψεις. Είτε έχω τρία δεδομένα είτε 1003, ο κώδικας είναι ίδιος στο μέγεθος.

7.1.3. 3ος τρόπος - Με πίνακες

```
Για i από 1 μέχρι 3
    Διάβασε A[i]
Τέλος_Επανάληψης
S←0
Για i από 1 μέχρι 3
    S←S+A[i]
Τέλος_Επανάληψης
```



Εδώ όλα τα δεδομένα τα τοποθετώ σε μια ολική θέση μνήμης που την ονομάζω A. Αν θέλω όμως να αναφερθώ σε ένα συγκεκριμένο στοιχείο της μνήμης αυτής πως θα μπορούσα να το κάνω; Σκέφτομαι λοιπόν να βάλω δείκτες. Έτσι θα ονομάσω την πρώτη θέσης της μνήμης αυτής A1, την δεύτερη θέσης της, A2 και την τρίτη θέση της, A3. Επειδή όμως δεν μπορώ να χρησιμοποιήσω ούτε δείκτες ούτε εκθέτες, για αυτό χρησιμοποιώ τους αντίστοιχους αριθμούς των δεικτών μέσα σε αγκύλες. Έτσι αν θέλω να αναφερθώ στην πρώτη θέση της ολικής μνήμης A τότε θα γράψω A[1], στην δεύτερη θέση της A[2] και στην τρίτη θέση της A[3].

Με αυτό τον τρόπο έχω λύσει το πρόβλημα της ονοματολογίας πολλών μεταβλητών χρησιμοποιώντας έναν πίνακα που ουσιαστικά είναι μια μεταβλητή πολλαπλών δεδομένων που έχουν κοινό όνομα , αλλά που αναφέρομαι σε κάθε ένα από αυτά τα δεδομένα με τους δείκτες. Εδώ και τα δεδομένα καταλαμβάνουν

διαφορετικές θέσεις μνήμης, άρα οποιαδήποτε στιγμή χρειάζεται μπορώ να ανακαλέσω την τιμή τους, αλλά και ο κώδικας είναι μικρός σε περίπτωση που έχω πολλά δεδομένα. Μειονέκτημα των πινάκων η κατασπατάληση θέσεων μνήμης, που κάποιες φορές μπορεί να οδηγήσει και σε αδυναμία εκτέλεσης του προγράμματος.

Τώρα μπορώ να διαβάσω τα στοιχεία του πίνακα σε άλλη επανάληψη και να αθροίσω τα στοιχεία σε νέα επανάληψη.

Για i=1 το Διάβασε A[i] γίνεται Διάβασε A[1] και οτιδήποτε διαβάζω τοποθετείται ως πρώτο στοιχείο του πίνακα.

Για i=2 το Διάβασε A[i] γίνεται Διάβασε A[2] και οτιδήποτε διαβάζω τοποθετείται ως δεύτερο στοιχείο του πίνακα.

Για i=3 το Διάβασε A[i] γίνεται Διάβασε A[3] και οτιδήποτε διαβάζω τοποθετείται ως τρίτο στοιχείο του πίνακα.

Μηδενίζω έναν αθροιστή εκτός επανάληψης.

Για i=1 το S←S+A[i] γίνεται S←S+A[1] άρα ανακαλώ την πρώτη τιμή του πίνακα Α και τοποθετώ στον αθροιστή το πρώτο στοιχείο του πίνακα, δηλαδή το 4.

Για i=2 το S←S+A[i] γίνεται S←S+A[2] άρα ανακαλώ την δεύτερη τιμή του πίνακα Α και τοποθετώ στον αθροιστή το δεύτερο στοιχείο του πίνακα, δηλαδή το 7.

Για i=3 το S←S+A[i] γίνεται S←S+A[3] άρα ανακαλώ την τρίτη τιμή του πίνακα Α και τοποθετώ στον αθροιστή το τρίτο στοιχείο του πίνακα, δηλαδή το 5.

7.2. Μονοδιάστατος πίνακας Α[Ν]

A[1]	**A[2]**	**A[3]**	**….**	**A[N]**

Οτιδήποτε κάνουμε σε μονοδιάστατο πίνακα, χρειάζεται αναγκαστικά απλή επαναληπτική διαδικασία, καθώς τα στοιχεία του πίνακα είναι πάντα περισσότερα του ενός. Τις περισσότερες φορές, χρειάζεται να προσπελάσουμε (περάσουμε από) όλα τα στοιχεία του πίνακα. Παρακάτω έχω τριά παραδείγματα διαβάσματος, αθροίσματος και εμφάνισης όλων των στοιχείων ενός πίνακα Α.

	S←0	
Για i **από** 1 **μέχρι** N	**Για** i **από** 1 **μέχρι** N	**Για** i **από** 1 **μέχρι** N
Διάβασε A[i]	S←S+A[i]	**Εμφάνισε** A[i]
Τέλος_επανάληψης	**Τέλος_επανάληψης**	**Τέλος_επανάληψης**

Όταν τα στοιχεία του πίνακα σημαίνουν κάτι τότε και οι θέσεις των στοιχείων του πίνακα σημαίνουν εξίσου κάτι. Σε ασκήσεις που τα δεδομένα έχουν δύο χαρακτηριστικά χρησιμοποιούνται μονοδιάστατοι πίνακες. Το ένα χαρακτηριστικό αντιστοιχεί στα στοιχεία του πίνακα και το άλλο αντιστοιχίζεται στην μοναδική διάσταση του πίνακα.

7.2.1. Πρόβλημα Ξ1

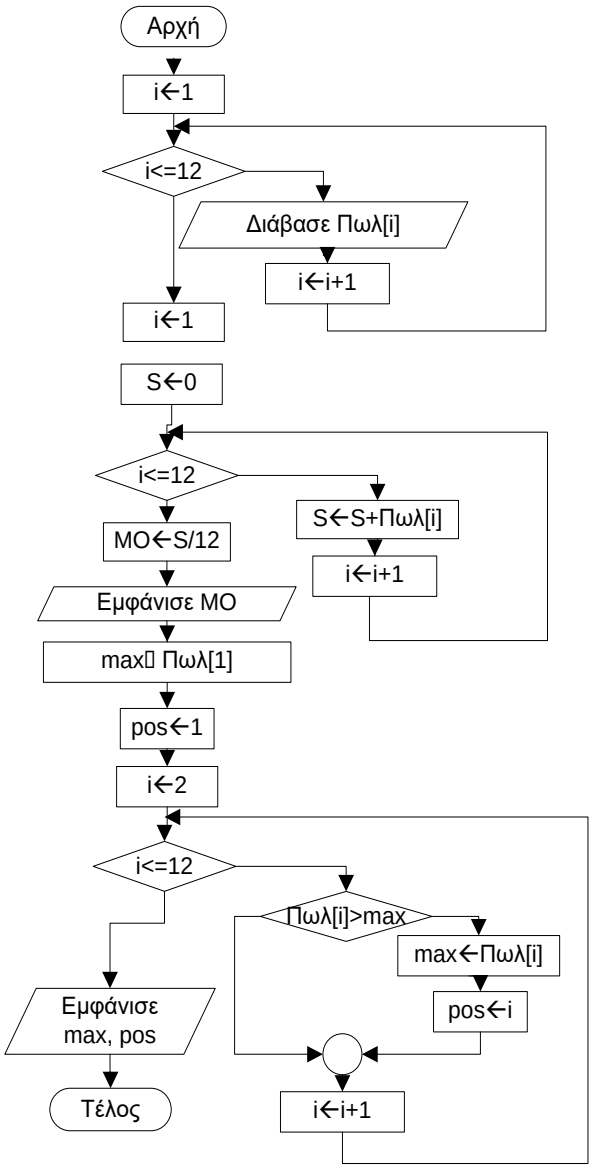
Μια εταιρεία είχε πωλήσεις για τον προηγούμενο χρόνο σύμφωνα με τον παρακάτω πίνακα:

1	2	3	4	5	6	7	8	9	10	11	12
100	120	110	90	178	95	130	110	200	70	50	155

1. Να εμφανιστεί ή μέση τιμή της πώλησης των αυτοκινήτων για το προηγούμενο έτος.
2. Να εμφανιστεί ποια ήταν η μεγαλύτερη πώληση αυτοκινήτων και για ποιο μήνα

Λύση: Εδώ έχω δύο χαρακτηριστικά. Το ένα χαρακτηριστικό είναι οι πωλήσεις των αυτοκινήτων που αντιστοιχούν στα 12 στοιχεία του πίνακα. Το άλλο χαρακτηριστικό είναι οι μήνες του χρόνου που αντιστοιχούν στον αύξοντα αριθμό των στοιχείων του πίνακα. Έτσι το τρίτο στοιχείο του πίνακα σημαίνει ότι πουλήθηκαν 110 αυτοκίνητα τον τρίτο μήνα του προηγούμενου έτους.

Αλγόριθμος Πωλήσεις
Για i **από** 1 **μέχρι** 12
Διάβασε Πωλ[i]
Τέλος_επανάληψης
S←0
Για i **από** 1 **μέχρι** 12
S←S+Πωλ[i]
Τέλος_επανάληψης
ΜΟ←S/12
Εμφάνισε ΜΟ
max←Πωλ[1]
pos←1
Για i **από** 2 **μέχρι** 12
Αν Πωλ[i] >max **τότε**
max←Πωλ[i]
pos←i
Τέλος_αν
Τέλος_επανάληψης
Εμφάνισε max,pos
Τέλος Πωλήσεις



7.2.1-1 Λογικό διάγραμμα

7.3.Δισδιάστατος πίνακας Α[Μ,Ν]

A[1,1]	**A[1,2]**	**A[1,3]**	**….**	**A[1,N]**
A[2,1	**A[2,2]**	**A[2,3]**	**….**	**A[2,N]**
…..	**….**	**….**	**….**	**….**
A[M,1]	**A[M,2]**	**A[M,3]**	**….**	**A[M,N]**

Οτιδήποτε κάνουμε σε δισδιάστατο πίνακα χρειάζεται αναγκαστικά διπλή επαναληπτική διαδικασία. Τις περισσότερες φορές, χρειάζεται να προσπελάσουμε (περάσουμε από) όλα τα στοιχεία του πίνακα. Ο δείκτης i δηλώνει συνήθως την γραμμή, ενώ ο δείκτης j δηλώνει συνήθως την στήλη.

Όταν τα στοιχεία του πίνακα σημαίνουν κάτι τότε και οι θέσεις των στοιχείων του πίνακα σημαίνουν εξίσου κάτι. Σε ασκήσεις που τα δεδομένα έχουν τρία χαρακτηριστικά χρησιμοποιούνται δισδιάστατοι πίνακες. Το ένα χαρακτηριστικό αντιστοιχεί στα στοιχεία του πίνακα και τα άλλα δυο αντιστοιχίζονται στις δύο διαστάσεις του πίνακα. Το ένα στις γραμμές και το άλλο στις στήλες του πίνακα.

7.3.1. Πρόβλημα Ο1

Να φτιαχτεί αλγόριθμος που να διαβάζει τις πωλήσεις των αυτοκινήτων που πραγματοποιήθηκαν από μία εταιρεία σε 5 πόλεις τους προηγούμενους 6 μήνες. Να βρεθεί και να εμφανιστεί η μέση τιμή των πωλήσεων των αυτοκινήτων για το προηγούμενο εξάμηνο. Επίσης να εμφανιστεί ποια πόλη πούλησε τα περισσότερα αυτοκίνητα και σε ποιο μήνα;

Λύση: Εδώ έχω τρία χαρακτηριστικά. Οι πωλήσεις των αυτοκινήτων είναι τα στοιχεία του πίνακα. Οι πόλεις αντιστοιχούν στις γραμμές του πίνακα, άρα θα έχω πέντε γραμμές, όσες και οι πόλεις. Οι μήνες αντιστοιχούν στις στήλες του πίνακα, άρα θα έχω 6 στήλες, όσοι και οι μήνες. Έτσι δημιουργώ έναν πίνακα 5Χ6 γιατί έτσι επέλεξα. Θα μπορούσα να δημιουργήσω έναν πίνακα 6Χ5 αν στις γραμμές έβαζα τους μήνες και στις στήλες τις πόλεις.

```
Αλγόριθμος Πωλήσεις
    Για i από 1 μέχρι 5
        Για j από 1 μέχρι 6
                Διάβασε Πωλ[i,j]
        Τέλος_επανάληψης
    Τέλος_επανάληψης
    S←0
    Για i από 1 μέχρι 5
        Για j από 1 μέχρι 6
                S←S+Πωλ[i,j]
        Τέλος_επανάληψης
    Τέλος_επανάληψης
    ΜΟ←S/30
    Εμφάνισε ΜΟ
    max←Πωλ[1,1]
    posi←1
```

```
        posj←1
        Για i από 1 μέχρι 5
            Για j από 1 μέχρι 6
                    Αν Πωλ[i,j] >max τότε
                            max←Πωλ[i,j]
                            posi←i
                            posj←j
                    Τέλος_αν
            Τέλος_επανάληψης
        Τέλος_επανάληψης
        Εμφάνισε max, posi, posj
Τέλος Πωλήσεις
```

7.4.Σάρωση κατά γραμμές ή στήλες:

Όταν ο δείκτης i είναι από έξω και ο δείκτης j από μέσα τότε η σάρωση των στοιχείων του πίνακα είναι κατά γραμμές, δηλαδή από το στοιχείο A[1,1] πάμε στο στοιχείο A[1,2] κτλ**.** Όταν ο δείκτης j είναι από έξω και ο δείκτης i από μέσα τότε η σάρωση των στοιχείων του πίνακα είναι κατά στήλες, δηλαδή από το στοιχείο A[1,1] πάμε στο στοιχείο A[2,1] κτλ.

7.5.Χαρακτηριστικά πινάκων

- Κάθε στοιχείο του πίνακα στην ονομασία του αποτελείται από δύο μέρη. Το πρώτο μέρος είναι κοινό για όλα τα στοιχεία του πίνακα και αποτελεί το όνομα του πίνακα. Το δεύτερο μέρος είναι ο δείκτης ή οι δείκτες που καθορίζουν την συγκεκριμένη θέση του στοιχείου μέσα στον πίνακα.
- Τα στοιχεία σε ένα πίνακα πρέπει να είναι όμοιου τύπου
- Τα στοιχεία ενός πίνακα καταλαμβάνουν διαδοχικές θέσεις μνήμης.

Συνηθίζουμε να χρησιμοποιούμε κεφαλαία γράμματα για τα ονόματα των πινάκων (χωρίς να είναι υποχρεωτικό), ώστε να ξεχωρίζουν από τις απλές μεταβλητές. Δεν πρέπει να ξεχνάμε πάντα να τοποθετούμε αγκύλες που θα περιέχουν το δείκτη του πίνακα που θέλουμε να επεξεργαστούμε. Πρέπει να προσέχουμε ώστε να μην ξεπερνάμε τα όρια του πίνακα, ούτε να αναφερόμαστε στη μηδενική θέση του πίνακα. Τέλος, οι δομές των πινάκων πρέπει να

χρησιμοποιούνται στις ασκήσεις όταν είναι απαραίτητο, με μέτρο, αφού υπάρχουν και μειονεκτήματα από τη χρήση τους όπως η άσκοπη χρήση μνήμης.

7.6.Παράλληλοι πίνακες

Τους χρησιμοποιούμε αναγκαστικά όταν πρέπει να διαχειριστούμε δεδομένα διαφορετικού τύπου καθώς οι πίνακες ως στατικές δομές δεδομένων απαγορεύουν κάτι τέτοιο. Στους πίνακες αυτούς υπάρχει μια αντιστοιχία, δηλ. το πρώτο στοιχείο του πρώτου πίνακα βρίσκεται σε πλήρη αντιστοίχιση με το πρώτο στοιχείο του δεύτερου πίνακα κ.ο.κ. Αν λοιπόν σε κάποιο σχολείο έχουμε να διαχειριστούμε βαθμούς και ονόματα μαθητών, επειδή οι βαθμοί είναι αριθμοί και τα ονόματα είναι χαρακτήρες, αναγκαστικά οι βαθμοί θα μπουν σε ένα πίνακα και τα ονόματα σε άλλο πίνακα. Επίσης παράλληλους πίνακες χρησιμοποιούμε προαιρετικά όταν έχουμε στοιχεία ίδιου τύπου, αλλά διαφορετικού είδους, καθώς συχνά αυτό διευκολύνει την επίλυση των ασκήσεων. Παραλληλία μπορεί να υπάρχει και μεταξύ μονοδιάστατου και δισδιάστατου πίνακα ως προς την μία διάστασή του βέβαια.

Βαθμός	Όνομα
Βαθ[1]	Ον[1]
Βαθ[2]	Ον[2]
Βαθ[3]	Ον[3]
…	…
Βαθ[ν]	Ον[ν]

7.6.1. Πρόβλημα Π1

Έχουμε δυο πίνακες, ο ένας με τα μοντέλα των υπολογιστών και ο δεύτερος με τις τιμές τους. Να γράψετε τις εντολές σε κώδικα που βρίσκουν και εμφανίζουν το φθηνότερο μοντέλο καθώς και το ακριβότερο.

Λύση: Οι παραπάνω πίνακες είναι παράλληλοι πίνακες. Ο πρώτος υπολογιστής έχει όνομα=μοντέλο [1] και τιμή=τιμή [1], ο δεύτερος υπολογιστής έχει όνομα=μοντέλο[2] και τιμή=τιμή[2] κ.ο.κ. Αν δεν είχα πίνακα με τα μοντέλα των υπολογιστών τότε για το ακριβότερο και το φθηνότερο μοντέλο θα μου αρκούσε η θέση των υπολογιστών, π.χ. ο πέμπτος υπολογιστής είναι ο ακριβότερος και ο τρίτος ο φθηνότερος. Δηλαδή θα μου αρκούσε να εμφανίσω τα pos1, pos2. Τώρα όμως που έχω και παράλληλο πίνακα με τα ονόματα πρέπει να εμφανίσω το

αντίστοιχο στοιχείο του πίνακα των ονομάτων που βρίσκεται στις θέσεις pos1 και pos2 δηλαδή τα στοιχεία μοντέλο[pos1] και μοντέλο[pos2].

Αλγόριθμος υπολογιστές
Διάβασε N
Για i **από** 1 **μέχρι** N
    Διάβασε μοντέλο[i] **,**τιμή[i]
Τέλος_επανάληψης
max←τιμή[1]
pos1←1
min←τιμή[1]
pos2←1
Για i **από** 2 **μέχρι** N
    Αν τιμή[i]>max **τότε**
        max← τιμή[I]
        pos1←i
    Τέλος_αν
    Αν τιμή[i]<min **τότε**
        min← τιμή[I]
        pos2←i
    Τέλος_αν
Τέλος_επανάληψης
Εμφάνισε ‘Το φτηνότερο μοντέλο είναι ‘, μοντέλο[pos2]
Εμφάνισε ‘Το ακριβότερο μοντέλο είναι ‘, μοντέλο[pos1]
Τέλος υπολογιστές

7.7.Σημαία

Είναι μια λογική μεταβλητή που την χρησιμοποιώ, όταν έχω αβεβαιότητα αν υπάρχει μία συγκεκριμένη ιδιότητα που ψάχνω να βρω, ανάμεσα σε κάποια στοιχεία έστω και μία φορά. Αρχικά της δίνω την τιμή **ψευδής**, κάνοντας την υπόθεση ότι η ιδιότητα που ψάχνω δεν υπάρχει σε κανένα από τα στοιχεία. Μέσα σε μια επαναληπτική διαδικασία ρωτάω με δομή επιλογής ένα προς ένα τα στοιχεία, αν κάποιο από αυτά έχει την ιδιότητα που θέλω και σε περίπτωση που αυτό συμβεί έστω και μία φορά, τότε αλλάζω την κατάσταση της σημαίας σε αληθής,

γιατί όντως βρέθηκε αυτό που έψαχνα. Σε περίπτωση που βρω περισσότερες από μία φορές την ιδιότητα που ψάχνω, η σημαία παραμένει στην κατάσταση αληθής.

Στο τέλος της επανάληψης έχω μια πληροφορία. Αν η σημαία έχει παραμείνει στην κατάσταση ψευδής τότε δεν υπάρχει η ιδιότητα που ψάχνω ανάμεσα στα στοιχεία, ενώ σε περίπτωση που είναι αληθής σημαίνει ότι η ιδιότητα αυτή βρέθηκε τουλάχιστον μία φορά.

Σε περίπτωση που είμαι βέβαιος ότι θα συναντήσω μια ιδιότητα ανάμεσα σε κάποια στοιχεία, είναι περιττό να χρησιμοποιήσω την λογική μεταβλητή της σημαίας.

flag←ψευδής
Για i **από** 1 **μέχρι** N
Αν *υπάρχει στα στοιχεία η ιδιότητα που ψάχνω* **τότε**
flag←αληθής
Τέλος_αν
Τέλος_επανάληψης
Αν flag=ψευδής **τότε**
Δεν βρέθηκε καμία φορά η ιδιότητα που ψάχνω
αλλιώς
Η ιδιότητα που ψάχνω υπάρχει ανάμεσα στα στοιχεία τουλάχιστον μία φορά
Τέλος_αν

7.7.1. Σημαία σε αναζήτηση

Όταν αναζητώ **αν ένα τουλάχιστον από τα στοιχεία του πίνακα έχει μία συγκεκριμένη ιδιότητα,** τότε χρησιμοποιώ την **σημαία** και ελέγχω στο τέλος μετά την επαναληπτική διαδικασία αν η σημαία έχει αλλάξει τιμή. Αν όντως έχει αλλάξει τιμή τότε αυτό σημαίνει ότι βρέθηκε τουλάχιστον μία φορά η ιδιότητα που ψάχνω στα στοιχεία του πίνακα.

7.8.Σειριακή Αναζήτηση:

Με την στενή έννοια σειριακή αναζήτηση έχω όταν ψάχνω να βρω αν ένα δεδομένο (συνήθως το βάζω σε μία μεταβλητή που την ονομάζω Key) είναι ίσο με κάποιο από τα στοιχεία του πίνακα.

Έστω λοιπόν ότι έχω τον Πίνακα Α με αριθμητικές τιμές Α [4,9,5,9,7]

Έχω τρείς περιπτώσεις:

1. Το στοιχείο αυτό να βρεθεί μια φορά ανάμεσα στα στοιχεία του πίνακα (Key=5). Στην περίπτωση αυτή με ενδιαφέρει να επιστρέψω την θέση του στοιχείου που είναι ίσο με το Key. Στην συγκεκριμένη περίπτωση επιστρέφω την τιμή 3, αφού το 5 βρίσκεται στην Τρίτη θέσης του πίνακα.
2. Το στοιχείο αυτό να μην βρεθεί καμία φορά ανάμεσα στα στοιχεία του πίνακα (Key=3). Σε αυτή την περίπτωση εμφανίζω ότι δεν βρέθηκε στοιχείο όμοιο με το Key.
3. Το στοιχείο αυτό να βρεθεί περισσότερες από μία φορές ανάμεσα στα στοιχεία του πίνακα. (Key=9). Στην περίπτωση αυτή με ενδιαφέρει να επιστρέψω τις θέσεις των στοιχείων που είναι ίσα με το Key. Συγκεκριμένα εδώ να επιστρέψω τις τιμές 2,4.

Η αναζήτηση του στοιχείου Key ανάμεσα στα στοιχεία του πίνακα μπορεί να πραγματοποιηθεί και στις τρείς παραπάνω περιπτώσεις με επαναληπτική διαδικασία για, με πλήθος αναζητήσεων σταθερό, ανεξάρτητα από την θέση του key στον πίνακα.

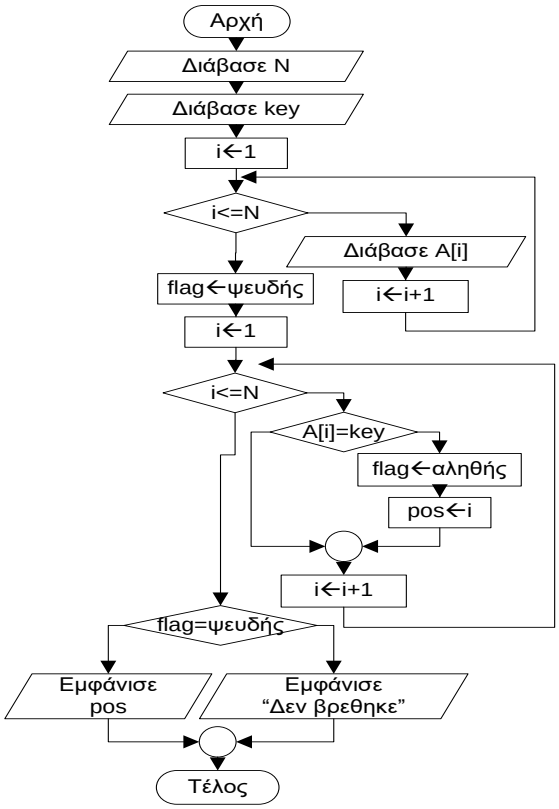
7.8.1. Πρόβλημα P1

7.8.1.1. 1ος τρόπος: Σειριακή Αναζήτηση με Για

Δίνεται πίνακας Α, Ν στοιχείων και αριθμός key. Να βρεθεί και να εμφανιστεί αν βρίσκεται ο αριθμός μέσα στον πίνακα ή όχι και σε ποια θέση. Σε περίπτωση που δεν βρίσκεται να εμφανίζεται το αντίστοιχο μήνυμα.

Αλγόριθμος σειριακή αναζήτηση
Διάβασε N, key
Για i **από** 1 **μέχρι** N
Διάβασε A[i]
Τέλος_επανάληψης
flag←ψευδής

Για i **από** 1 **μέχρι** N
    Αν A[i]= key **τότε**
        flag←αληθής
        pos←i (εμφάνισε i)
    Τέλος_αν
Τέλος_επανάληψης
Αν flag=ψευδής **τότε**
    Εμφάνισε “Δεν βρέθηκε ο αριθμός”
αλλιώς
    Εμφάνισε pos
Τέλος_αν
Τέλος σειριακή αναζήτηση



7.8.1.1-1 Σειριακή αναζήτηση - Για

Αρχικά διαβάζω το πλήθος των στοιχείων του πίνακα και την μεταβλητή Key, δηλαδή το δεδομένο που θέλω να βρω ανάμεσα στα στοιχεία του πίνακα. Μετά διαβάζω τα στοιχεία του πίνακα. Έπειτα χρησιμοποιώ μια λογική μεταβλητή flag, στην οποία αρχικά δίνω την κατάσταση ψευδής. Με αυτό τον τρόπο δηλώνω ότι έστω ότι είναι ψέμα ότι θα βρω αυτό το στοιχείο μέσα στον πίνακα.

Στην συνέχεια ελέγχω κάθε στοιχείο του πίνακα αν είναι όμοιο με το Key και στην περίπτωση αυτή αλλάζω κατάσταση στην μεταβλητή flag και της δίνω την τιμή αληθής, γιατί αλήθεια βρέθηκε το στοιχείο. Επίσης κρατάω την τρέχουσα τιμή της μεταβλητής i σε μία μεταβλητή θέσης που την ονομάζω pos.

Ο αλγόριθμος αυτός ισχύει σε περίπτωση που έχω ένα ή κανένα στοιχείο του πίνακα ίσο με το key. Σε περίπτωση που βρω περισσότερα του ενός στοιχεία να είναι όμοια με το key, τότε η μεταβλητή pos θα κρατήσει την θέση του τελευταίου. Για αυτό και αντικαθιστώ την εντολή pos←i με την εντολή Εμφάνισε i. Αν έχω περισσότερα από ένα στοιχεία του πίνακα όμοια με το key επειδή εμφανίζω τις θέσεις τους μέσα στην επανάληψη δεν υπάρχει τομέας **αλλιώς** στην δομή επιλογής **Αν**. Επίσης χρησιμοποιώ αναγκαστικά για την αναζήτηση επαναληπτική διαδικασία **Για**.

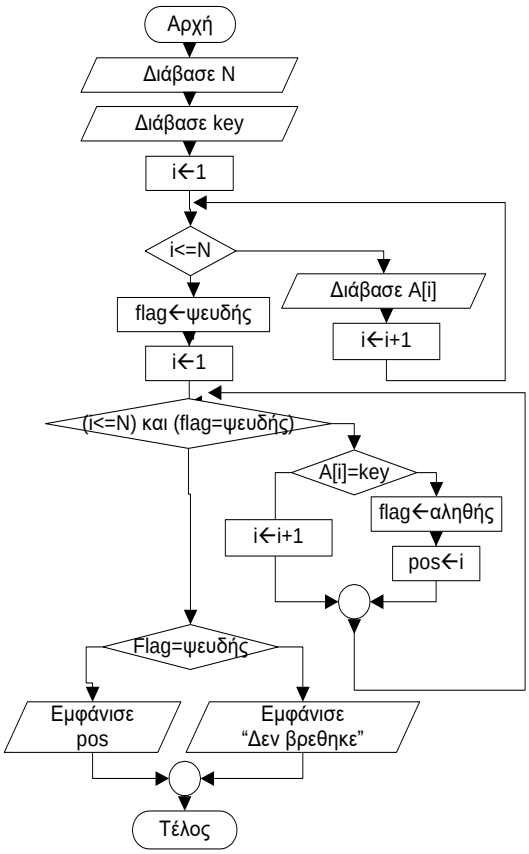
Σε περίπτωση που η σημαία (flag) παραμείνει ψευδής μετά το πέρας της επανάληψης τότε αυτό σημαίνει ότι ποτέ δεν εκτελέστηκε η συνθήκη A[i]= key**,** άρα δεν βρέθηκε κανένα στοιχείο του πίνακα όμοιο με το key, για αυτό τον λόγο εμφανίζω σχετικό μήνυμα. Σε διαφορετική περίπτωση επιστρέφω την θέση του στοιχείου που βρέθηκε. Οι επαναλήψεις εδώ θα είναι οι μέγιστες ανεξάρτητα από τη θέση του στοιχείου του πίνακα που είναι όμοιο με το key.

Σε περίπτωση που θέλω να φύγω από την επαναληπτική διαδικασία όταν βρω ένα στοιχείο του πίνακα όμοιο με το key τότε πρέπει να μετατρέψω την για σε μία όσο και να ελέγξω πότε η σημαία θα αλλάξει τιμή από ψευδής σε αληθής.

7.8.1.2. 2ος τρόπος: Σειριακή Αναζήτηση με Όσο

Οι επαναλήψεις εξαρτώνται από την θέση του στοιχείου του πίνακα που είναι όμοιο με το key. Εδώ γίνεται έλεγχος και της σημαίας.

Αλγόριθμος σειριακή αναζήτηση
Διάβασε N, key
Για i **από** 1 **μέχρι** N
Διάβασε A[i]
Τέλος_επανάληψης
flag←ψευδής
i←1
Όσο (i<=N) **και** (flag= ψευδής) **επανάλαβε**
Αν A[i]=key **τότε**
pos←i
flag←αληθής
αλλιώς
i←i+1
Τέλος_αν
Τέλος_επανάληψης
Αν flag=ψευδής **τότε**
Εμφάνισε “Δεν βρέθηκε ο αριθμός”
αλλιώς
Εμφάνισε pos
Τέλος_αν



7.8.1.2-1 Σειριακή αναζήτηση Όσο

Τέλος σειριακή αναζήτηση

Τώρα βλέπω ότι για να μείνω στην Όσο θα πρέπει να ισχύουν δύο συνθήκες ταυτόχρονα. Και τα στοιχεία του πίνακα να μην έχουν σωθεί (i<=N), αλλά και η σημαία να βρίσκεται στην κατάσταση ψευδής (δεν έχει βρεθεί ακόμα το στοιχείο). Αν η συνθήκη A[i]=key είναι αληθής, η σημαία αλλάζει τιμή και φεύγω από την επαναληπτική διαδικασία πριν να πάρει το i όλες τις τιμές του. Αυτός ό τρόπος δεν ισχύει όταν υπάρχει πολλαπλότητα εμφάνισης του στοιχείου που ψάχνω μέσα στον πίνακα.

Στις πρώτες δύο περιπτώσεις μπορώ να χρησιμοποιήσω την Όσο αλλά και την Για ως επαναληπτική διαδικασία αναζήτησης. Με την 'Οσο πάω αναγκαστικά όταν μου ζητείται να φύγω από την αναζήτηση σε περίπτωση που βρεθεί το στοιχείο. Στην τρίτη περίπτωση αναγκαστικά θα χρησιμοποιήσω την Για γιατί πρέπει η αναζήτηση να συνεχιστεί μέχρι το τέλος του πίνακα αφού το στοιχείο που ψάχνω ενδέχεται να συναντηθεί αρκετές φορές στα στοιχεία του πίνακα.

7.9.Δυαδική αναζήτηση

Ο αλγόριθμος της δυαδικής αναζήτησης (binary search) εφαρμόζεται μόνο σε πίνακες που έχουν ταξινομημένα στοιχεία. Αν τα στοιχεία δεν είναι ταξινομημένα τότε δεν μπορεί να εφαρμοστεί. Ο αλγόριθμος λειτουργεί ως εξής:

Βρίσκουμε το μεσαίο στοιχείο του ταξινομημένου πίνακα. Εάν το προς αναζήτηση στοιχείο είναι ίσο με το μεσαίο στοιχείο τότε σταματάμε την αναζήτηση αφού το στοιχείο βρέθηκε.

Εάν δεν είναι το μεσαίο, τότε ελέγχουμε αν το στοιχείο που αναζητούμε είναι μικρότερο ή μεγαλύτερο από το μεσαίο στοιχείο του πίνακα. Αν είναι μικρότερο, περιορίζουμε την αναζήτηση στο πρώτο μισό του πίνακα (με την προϋπόθεση ότι τα στοιχεία είναι διατεταγμένα κατά αύξουσα σειρά), ενώ αν είναι μεγαλύτερο περιορίζουμε την αναζήτηση στο δεύτερο μισό του πίνακα.

Η διαδικασία αυτή επαναλαμβάνεται για το κατάλληλο πρώτο ή δεύτερο μισό πίνακα, μετά για το 1/4 του πίνακα κ.ο.κ. μέχρι, είτε να βρεθεί το στοιχείο, είτε να μην είναι δυνατό να χωρισθεί ο πίνακας περαιτέρω σε δύο νέα μέρη.

ΠΡΟΓΡΑΜΜΑ δυαδική_αναζήτηση

ΜΕΤΑΒΛΗΤΕΣ

ΑΚΕΡΑΙΕΣ: A[100], Left, Right, M, k, key, i

```
    ΛΟΓΙΚΕΣ: flag
ΑΡΧΗ
ΓΡΑΨΕ 'Οι αριθμοί που θα δοθούν πρέπει να είναι ταξινομημένοι κατά αύξουσα τάξη'
ΔΙΑΒΑΣΕ N
 ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ N
     ΓΡΑΨΕ 'Δώσε το', i, ' στοιχείο του πίνακα'
     ΔΙΑΒΑΣΕ A[i]
 ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
 ΓΡΑΨΕ 'Δώσε τιμή για αναζήτηση: '
 ΔΙΑΒΑΣΕ key
 Left ← 1
 Right ←N
 flag ← ΨΕΥΔΗΣ
 ΟΣΟ (Left <= Right) ΚΑΙ (flag = ΨΕΥΔΗΣ)
 ΕΠΑΝΑΛΑΒΕ
     M ← (Left + Right) DIV 2
     ΑΝ A[M] = key ΤΟΤΕ
         pos ← M
         flag ← ΑΛΗΘΗΣ
     ΑΛΛΙΩΣ
         ΑΝ  key < A[M]  ΤΟΤΕ
             Right ← M - 1
         ΑΛΛΙΩΣ
             Left ← M + 1
         ΤΕΛΟΣ_ΑΝ
     ΤΕΛΟΣ_ΑΝ
 ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΑΝ flag = ΑΛΗΘΗΣ ΤΟΤΕ
     ΓΡΑΨΕ 'Το στοιχείο ',  key, ' υπάρχει στη θέση: ',  pos ( ή 'υπάρχει στην θέση ', M )
ΑΛΛΙΩΣ
    ΓΡΑΨΕ 'Το στοιχείο ',  key, '  δεν υπάρχει στον πίνακα'
ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

7.9-1 Πληθος Συγκρίσεων - Δυαδική αναζήτηση

Αριθμός συγκρίσεων στη δυαδική αναζήτηση	
Στοιχεία N	Συγκρίσεις
10	4
100	7
1.000	10
10.000	14
100.000	17
1.000.000	20
10.000.000	24
100.000.000	27
1.000.000.000	30

Στο πρώτο κομμάτι του κώδικα αρχικά δηλώνουμε τις μεταβλητές μας. Επειδή διαβάζω Ν στοιχεία και οι πίνακες είναι στατικές δομές δεδομένων, θα πρέπει να δηλώσω το μέγιστο πλήθος των στοιχείων του (έστω εκατό) κατά το στάδιο του προγραμματισμού. Έτσι έχω την δήλωση Α[100]. Στις λογικές βάζω την σημαία flag, γνωστή για την χρησιμοποίησή της όταν έχω αβεβαιότητα αν θα βρω το χαρακτηριστικό που ψάχνω ανάμεσα σε δεδομένα (εδώ αν θα βρω το key ανάμεσα στα στοιχεία του ταξινομημένου πίνακα). Στις μεταβλητές Left, Right θα καθορίζω το εύρος του πίνακα που κάθε φορά ψάχνω το key.

Αρχικά η μεταβλητή Left έχει την τιμή ένα και η μεταβλητή Right έχει την τιμή Ν, γιατί στην αρχή ψάχνω να βρω το στοιχείο Key σε όλον τον πίνακα. Βάζω στην μεταβλητή flag την τιμή ψευδής. Έστω ότι είναι ψέμα ότι θα βρούμε το στοιχείο Key ανάμεσα στα στοιχεία του πίνακα.

Μέσα στην επανάληψη Όσο κάνουμε τα εξής: Βρίσκω την μέση του πίνακα. Αν το μεσαίο στοιχείο του πίνακα ισούται με το key τότε κρατάω στην μεταβλητή pos την θέση του στοιχείου και αλλάζω την τιμή της σημαίας flag σε Αληθής, καθώς όντως βρέθηκε το στοιχείο. Στην Όσο η συνθήκη (flag = ΨΕΥΔΗΣ) δεν ισχύει και άρα φεύγω από την επαναληπτική διαδικασία.

Σε περίπτωση που δεν βρω το μεσαίο στοιχείο του πίνακα όμοιο με το key τότε ανάλογα, αν το key βρίσκεται στο πρώτο μισό του πίνακα μειώνω την τιμή της μεταβλητής Right έτσι ώστε να ορίζεται το πρώτο μισό του πίνακα χωρίς το μεσαίο στοιχείο του, ενώ αν το key βρίσκεται στο δεύτερο μισό του πίνακα αυξάνω την τιμή της μεταβλητής Left έτσι ώστε να ορίζεται το δεύτερο μισό του πίνακα χωρίς το μεσαίο στοιχείο του.

Η διαδικασία αυτή επαναλαμβάνεται για τις νέες τιμές των ορίων αναζήτησης Left και Right μέχρι, είτε να βρεθεί το στοιχείο, είτε να μην είναι δυνατό να χωρισθεί ο πίνακας περαιτέρω σε δύο νέα μέρη, δηλαδή όταν η μεταβλητή Left γίνει μεγαλύτερη από την μεταβλητή Right.

Αν η σημαία flag γίνει ΑΛΗΘΗΣ τότε εμφανίζω την θέση του στοιχείου που βρέθηκε. Αν η σημαία (flag) παραμείνει ψευδής μετά το πέρας της επανάληψης τότε αυτό σημαίνει ότι δεν βρέθηκε κανένα στοιχείο του πίνακα όμοιο με το key και εμφανίζω σχετικό μήνυμα.

7.10. Αναζήτηση για κάθε στοιχείο του πίνακα:

Όταν αναζητώ αν **όλα τα στοιχεία του πίνακα έχουν μια συγκεκριμένη ιδιότητα** τότε χρησιμοποιώ **μετρητή** και ελέγχω στο τέλος μετά την επαναληπτική διαδικασία αν ο μετρητής έχει πάρει την μέγιστη τιμή του. Αν όντως έχει πάρει την μέγιστη τιμή του αυτό σημαίνει ότι όλα τα στοιχεία του πίνακα έχουν αυτή την ιδιότητα που ψάχνω. Μπορώ να χρησιμοποιήσω και την σημαία αλλάζοντάς την συνθήκη στο αντίθετό της.

7.10.1. Πρόβλημα Σ1

Να φτιαχτεί αλγόριθμος που να διαβάζει δύο πίνακες ακεραίων 100 στοιχείων και να συγκρίνει όλα τα στοιχεία τους. Να εμφανίζει το μήνυμα «Ίσοι» εάν έχουν τις ίδιες τιμές και τα 100 στοιχεία τους, διαφορετικά «Άνισοι»

```
Αλγόριθμος άσκηση
  Για i από 1 μέχρι  100
      Διάβασε A[i] , B[i]
Τέλος_επανάληψης
M←0
  Για i από 1 μέχρι  100
      Αν A[i]=B[i] τότε
      M←M+1
    Τέλος_αν
Τέλος_επανάληψης
Αν M=100 τότε
    Εμφάνισε “Ίσοι”
αλλιώς
    Εμφάνισε “Άνισοι”
Τέλος_αν
Τέλος άσκηση
```

```
Αλγόριθμος άσκηση
  Για i από 1 μέχρι  100
      Διάβασε A[i] , B[i]
Τέλος_επανάληψης
flag←αληθής
  Για i από 1 μέχρι  100
      Αν A[i]<>B[i] τότε
      flag ←ψευδής
    Τέλος_αν
Τέλος_επανάληψης
Αν flag =αληθής τότε
    Εμφάνισε “Ίσοι”
αλλιώς
    Εμφάνισε “Άνισοι”
Τέλος_αν
Τέλος άσκηση
```

7.11. Αθροίσματα γραμμών και στηλών

7.11.1. Πρόβλημα Σ2

Να φτιαχτεί αλγόριθμος που να διαβάζει έναν πίνακα Α 2x3. Να βρεθούν και να εμφανιστούν τα αθροίσματα των γραμμών και των στηλών του.

Λύση: Έστω ότι έχω τον παρακάτω πίνακα 2 γραμμών και 3 στηλών. Οτιδήποτε κάνω σε μονοδιάστατο πίνακα χρειάζεται απλή επαναληπτική διαδικασία με έναν δείκτη. Οτιδήποτε κάνω σε δισδιάστατο πίνακα θέλει διπλή επαναληπτική διαδικασία με δύο δείκτες. Έτσι για να διαβάσω τα στοιχεία του πίνακα Α θα πρέπει να χρησιμοποιήσω μια διπλή επαναληπτική διαδικασία. Το i παίρνει τιμές από 1 μέχρι 2 γιατί δύο είναι οι γραμμές του πίνακα. Το j παίρνει τιμές από 1 μέχρι 3 γιατί τρείς είναι οι στήλες του πίνακα.

				row
	4	3	2	**9**
	1	7	5	**13**
col	**5**	**10**	**7**	**22**

1,1	1,2	1,3
2,1	2,2	2,3

Πολλές φορές χρειάζεται να βρω τα επιμέρους αθροίσματα της κάθε γραμμής και της κάθε στήλης. Άθροισμα σε επαναληπτική διαδικασία σημαίνει αθροιστής. Επειδή έχω περισσότερες από μία γραμμές για αυτό το λόγο έχω και περισσότερους από έναν αθροιστές. Συγκεκριμένα έχω τόσους αθροιστές όσες και οι γραμμές του πίνακα. Εδώ δηλαδή έχω δύο αθροιστές. Τους αθροιστές αυτούς τους τοποθετώ ως στοιχεία ενός πίνακα που συνήθως τον ονομάζω row. Γενικότερα κάθε στοιχείο του πίνακα row αφού είναι ένας αθροιστής μηδενίζεται.

Με την ίδια λογική δημιουργώ και έναν πίνακα col όπου κάθε στοιχείο του είναι και ένας αθροιστής της αντίστοιχης στήλης. Συγκεκριμένα εδώ έχω 3 στοιχεία για τον πίνακα col, όσες και οι στήλες του πίνακα Α. Επειδή κάθε αθροιστής αρχικά παίρνει την τιμή 0, μηδενίζω όλα τα στοιχεία του πίνακα col.

Για να βρω τον πίνακα row θα πρέπει η σάρωση των στοιχείων του πίνακα να είναι κατά γραμμές. Έτσι το i είναι εξωτερικός δείκτης και το j είναι εσωτερικός. Ο εξωτερικός δείκτης πάντα παραμένει σταθερός μέχρι ο εσωτερικός δείκτης να πάρει όλες τις διαφορετικές τιμές του. Ο πίνακας row έχει αρχικά μηδενισμένες όλες τις τιμές του. Έτσι στον αθροιστή row[1] θα αθροίσω διαδοχικά τα στοιχεία A[1,1], A[1,2], A[1,3]. Ενώ στον αθροιστή row[2] θα αθροίσω διαδοχικά τα στοιχεία A[2,1], A[2,2], A[2,3].

Για να βρω τον πίνακα col θα πρέπει η σάρωση των στοιχείων του πίνακα να είναι κατά στήλες. Έτσι το j είναι εξωτερικός δείκτης και το i είναι εσωτερικός. Ο εξωτερικός δείκτης πάντα παραμένει σταθερός μέχρι ο εσωτερικός δείκτης να πάρει όλες τις διαφορετικές τιμές του. Ο πίνακας col έχει αρχικά μηδενισμένες όλες τις τιμές του. Έτσι στον αθροιστή col[1] θα αθροίσω διαδοχικά τα στοιχεία A[1,1], A[2,1]. Στον αθροιστή col[2] θα αθροίσω διαδοχικά τα στοιχεία A[1,2], A[2,2]. Ενώ στον αθροιστή col[3] θα αθροίσω διαδοχικά τα στοιχεία A[1,3], A[2,3].

```
Αλγόριθμος Αθροισμα_γραμμών_στηλών
Για i από 1 μέχρι 2
    Για j από 1 μέχρι 3
    Διάβασε A[ι,j]
    Τέλος_επανάληψης
Τέλος_επανάληψης
Για i από 1 μέχρι 2
    row[i]←0
Τέλος_επανάληψης
Για i από 1 μέχρι 2
     Για j από 1 μέχρι 3
          row[i]← row[i]+A[i,j]
      Τέλος_επανάληψης
Τέλος_επανάληψης
Για i από 1 μέχρι 2
    Εμφάνισε row[i]
Τέλος_επανάληψης
Για j από 1 μέχρι 3
    col[j]←0
Τέλος_επανάληψης
Για j από 1 μέχρι 3
    Για i από 1 μέχρι 2
          col[j]← col[j]+A[i,j]
    Τέλος_επανάληψης
Τέλος_επανάληψης
```

Υπολογισμός και εμφάνιση των στοιχείων κάθε γραμμής με απλό αθροιστή

```
Για i από 1 μέχρι 2
   S←0
   Για j από 1 μέχρι 3
      S← S+A[i,j]
   Τέλος_επανάληψης
   Εμφάνισε S
Τέλος_επανάληψης
```

Υπολογισμός και εμφάνιση των στοιχείων κάθε γραμμής με απλό αθροιστή

```
Για j από 1 μέχρι 3
   S←0
   Για i από 1 μέχρι 2
      S← S+A[i,j]
   Τέλος_επανάληψης
   Εμφάνισε S
Τέλος_επανάληψης
```

```
 Για j από 1 μέχρι 3
     Εμφάνισε col[j]
Τέλος_επανάληψης
Για j από 1 μέχρι 3
     Για i από 1 μέχρι 2
          col[j]← col[j]+A[i,j]
     Τέλος_επανάληψης
Τέλος_επανάληψης
 Για j από 1 μέχρι 3
     Εμφάνισε col[j]
Τέλος_επανάληψης
Τέλος Αθροισμα_γραμμών_στηλών
```

7.12. Ταξινόμηση Φυσαλίδα

Ως προς το διάβασμα του μη ταξινομημένου πίνακα και ως προς την εμφάνιση του νέου ταξινομημένου πίνακα δεν έχουμε καμία παρατήρηση να κάνουμε. Είναι πράγματα γνωστά. Η φυσαλίδα χρησιμοποιείται αποκλειστικά σε μονοδιάστατους πίνακες. Έχει αποδειχθεί ότι με επαναλαμβανόμενη διαδοχική σύγκριση των στοιχείων του πίνακα από το πίσω μέρος του προς την αρχή του, ο πίνακας σιγά σιγά ταξινομείται.

Παρόλο που ο πίνακας Τ είναι μονοδιάστατος στο κέντρο του κώδικα βλέπουμε μια διπλή επαναληπτική διαδικασία.

- Την εξωτερική επαναληπτική διαδικασία με δείκτη το i
- Την εσωτερική επαναληπτική διαδικασία με τον δείκτη j.

Όμως μόνο ο δείκτης j διαχειρίζεται τα στοιχεία του πίνακα. Ο άλλος δείκτης (i) χρειάζεται για να επαναληφθεί η διαδοχική σύγκριση των στοιχείων του πίνακα από το πίσω μέρος του προς την αρχή του, περισσότερες από μία φορές.

Για να μπορέσουμε να κατανοήσουμε την φυσαλίδα θα πρέπει να εξηγήσουμε αρχικά τι κάνει η τριάδα των εντολών που βρίσκεται στο κέντρο της διπλής επαναληπτικής διαδικασίας. Έτσι θα εξηγήσουμε για την αντιμετάθεση δύο αριθμών.

7.12.1. Αντιμετάθεση δύο αριθμών

A=5	Έστω ότι έχω δύο μεταβλητές την μεταβλητή Α και την
B=8	μεταβλητή Β και θέλω να αντιμεταθέσω τις τιμές τους.
A←B A=8	Βάζω στο Α την τιμή του Β και έτσι το Α παίρνει την τιμή 8.
B←A B=8	Βάζω στο Β την τιμή του Α, που όμως από την προηγούμενη εντολή έχει χάσει την τιμή 5 και έχει πάρει την τιμή 8. Άρα έχασα μία τιμή.
A=5	Σκέφτομαι ότι μάλλον είμαι άτυχος. Ξεκινάω λοιπόν ανάποδα.
B=8	Βάζω στο Β την τιμή του Α και έτσι το Β παίρνει την τιμή 5.
B←A B=5	Βάζω στο Α την τιμή του Β που όμως από την προηγούμενη
A←B A=5	εντολή έχει χάσει την τιμή 8 και έχει πάρει την τιμή 5. Άρα έχασα πάλι μία τιμή.

Χρειάζεται αναγκαστικά λοιπόν να χρησιμοποιήσουμε μια βοηθητική μεταβλητή που την ονομάζω συνήθως temp από τον temporary που σημαίνει προσωρινή θέση μνήμης. Και έτσι έχουμε την παρακάτω τριάδα εντολών, ξεκινώντας από το Α στο Β.

temp ← A	temp=5
A ← B	A=8
B ← temp	B=5

Την τριάδα αυτή των εντολών είναι εύκολο να την θυμάμαι αν έχω τον παρακάτω μνημονικό κανόνα: Η τριάδα αυτή ξεκινάει και τελειώνει με την μεταβλητή temp. Οτιδήποτε βρίσκεται στα δεξιά της προηγούμενης εντολής εκχώρησης, βρίσκεται στα αριστερά της επόμενης εντολής εκχώρησης. Θα μπορούσα λοιπόν κάλλιστα να χρησιμοποιήσω και την αντίστοιχη τριάδα εντολών, ξεκινώντας από το Β.

temp ← B	temp=8
B ← A	B=5
A ← temp	A=8

Τώρα λοιπόν καταλαβαίνω ότι η τριάδα των εντολών που βρίσκεται στο κέντρο της διπλής επαναληπτικής διαδικασίας έχει να κάνει με αντιμετάθεση δύο στοιχείων του πίνακα. Κοιτάζοντας καλύτερα βλέπω ότι αυτά τα στοιχεία του πίνακα είναι διαδοχικά, καθώς όταν το j πάρει πχ την τιμή 5 τότε το T[j-1] θα είναι το T[4] και το T[j] θα είναι το T[5].

Πότε όμως αντιμεταθέτω τα διαδοχικά στοιχεία του πίνακα; Όταν το T[j-1] > T[j] , δηλαδή όταν το προηγούμενο στοιχείο του πίνακα είναι μεγαλύτερο από το επόμενο στοιχείο του πίνακα. Σε ένα ταξινομημένο πίνακα κάθε προηγούμενο στοιχείο του πίνακα είναι μικρότερο ή ίσο από το επόμενο στοιχείο του πίνακα. Για αυτό λοιπόν όταν βρίσκω ένα προηγούμενο στοιχείο του πίνακα να είναι μεγαλύτερο από ένα επόμενο στοιχείο του πίνακα το αντιμεταθέτω.

Ο δείκτης i ξεκινάει από την τιμή 2 για να μπορέσουμε να συγκρίνουμε το T[1] με το T[2]. Αν το i ξεκινούσε από την τιμή 1 τότε θα συγκρίναμε το T[0] με το T[1], κάτι που δεν είναι επιτρεπτό, καθώς δεν υπάρχει μηδενικό στοιχείο του πίνακα.

Για i=2 ταξινομώ στην τελική του θέση το πρώτο στοιχείο του πίνακα. Για i=3 ταξινομώ στην τελική τους θέση δύο στοιχεία του πίνακα κ.ο.κ. Άρα γενικότερα αν θέλω να ταξινομήσω κ από τα N στοιχεία του πίνακα ο δείκτης i θα πρέπει να πάρει μέχρι και την τιμή κ+1.

Το πρόσημο στην εντολή T[j-1] > T[j] είναι αυτό που καθορίζει αν η ταξινόμηση φυσαλίδα θα είναι κατά αύξουσα ή κατά φθίνουσα σειρά. Σε περίπτωση που αντιμεταθέτω τα διαδοχικά στοιχεία του πίνακα όταν το προηγούμενο στοιχείο του πίνακα είναι μικρότερο από το επόμενο (T[j-1] < T[j]) τότε η ταξινόμηση γίνεται κατά φθίνουσα σειρά δηλαδή από τις μεγάλες τιμές στις μικρές τιμές.

Σε περίπτωση ύπαρξης παράλληλων πινάκων πρέπει το κριτήριο μου να είναι ο ένας από τους πίνακες αυτούς, αλλά όταν αλλάζω την σειρά των στοιχείων του ενός πίνακα πρέπει ταυτόχρονα να αλλάζω και την σειρά των αντιστοίχων στοιχείων των άλλων παράλληλων πινάκων. Κάτω δεξιά φαίνεται ο κώδικας σε περίπτωση που θα ήθελα να ταξινομήσω δύο παράλληλους πίνακες, ο ένας με τους βαθμούς και ο άλλος με τα ονόματα των μαθητών. Το κριτήριο ταξινόμησης θα είναι οι βαθμοί των μαθητών.

7.12.2. Πρόβλημα Τ1

Να γίνει ο αλγόριθμος της ταξινόμησης πίνακα με τη μέθοδο φυσαλίδα σε έναν πίνακα που διαβάζεται από το πληκτρολόγιο. Να εμφανιστούν τα ταξινομημένα στοιχεία του πίνακα**.**

```
Αλγόριθμος Φυσαλίδα
Διάβασε N
Για i από 1 μέχρι  N
    Διάβασε T[i]
Τέλος_επανάληψης
Για i από 2 μέχρι N
    Για j από N μέχρι i με βήμα –1
        Αν T[j-1] > T[j] τότε
            temp←T[j-1]
            T[j-1]←T[j]
            T[j]←temp
        Τέλος_αν
    Τέλος_επανάληψης
Τέλος_επανάληψης
Για i από 1 μέχρι  N
    Εμφάνισε T[i]
Τέλος_επανάληψης
Τέλος Φυσαλίδα
```

```
Αλγόριθμος Φυσαλίδα
Διάβασε N
Για i από 1 μέχρι  N
    Διάβασε βαθ[i], ονομ[i]
Τέλος_επανάληψης
Για i από 2 μέχρι N
    Για j από N μέχρι i με βήμα –1
        Αν βαθ[j-1] > βαθ[j] τότε
            temp1←βαθ[j-1]
            βαθ[j-1]←βαθ[j]
            βαθ[j]←temp1
            temp2←ονομ[j-1]
            ονομ [j-1]←ονομ [j]
            ονομ [j]←temp2
        Τέλος_αν
    Τέλος_επανάληψης
Τέλος_επανάληψης
Για i από 1 μέχρι  N
    Εμφάνισε βαθ[i], ονομ[i]
Τέλος_επανάληψης
Τέλος Φυσαλίδα
```

7.12.3. Πρόβλημα Τ2

Να γίνει ο αλγόριθμος της ταξινόμησης πίνακα με τη μέθοδο φυσαλίδα σε έναν πίνακα που διαβάζεται από το πληκτρολόγιο. Να εμφανιστούν τα ταξινομημένα στοιχεία. Σε περίπτωση που ο πίνακας ταξινομηθεί οποιαδήποτε χρονική στιγμή ο αλγόριθμος να σταματάει την εκτέλεσή του.

Όταν ο πίνακας έχει πολλά δεδομένα ή όταν είναι μικρός αλλά ήδη μερικώς ταξινομημένος, τότε ενδέχεται να έχουμε άσκοπα διαδοχικές συγκρίσεις των στοιχείων

```
flag←αληθής
i←2
Όσο i <= N και flag =αληθής επανάλαβε
    flag←ψευδής
    Για j από N μέχρι i με βήμα –1
```

του πίνακα, παρόλο που αυτός είναι ήδη ταξινομημένος.

Για τον πίνακα T=[5 6 7 9 1] παρατηρούμε ότι τα στοιχεία του είναι μερικώς ταξινομημένα. Για i=2 ο πίνακας παίρνει την μορφή = [**1** 5 6 7 9] και άρα είναι ήδη ταξινομημένος και δεν χρειάζεται το i να πάρει όλες τις υπόλοιπες τιμές του (δηλ.3,4,5).

Για να το αποφύγουμε αυτό έχουμε τον αλγόριθμο της **έξυπνης φυσαλίδας** όπου και αντικαθιστούμε την εξωτερική επαναληπτική διαδικασία **για** με μια **όσο** και βέβαια ελέγχουμε μία σημαία. Η αρχική εντολή flag←αληθής χρειάζεται για να μπούμε στην όσο την πρώτη φορά. Μετατρέπουμε την για σε όσο με την γνωστή τριάδα των εντολών. Βεβαίως ελέγχουμε και την σημαία, η οποία πριν από κάθε διαδοχική σύγκριση των στοιχείων του πίνακα από το πίσω μέρος του πίνακα προς το εμπρός του, δέχεται την τιμή ψευδής. Αν δεν υπάρξει κάποια αντιμετάθεση στα στοιχεία του πίνακα τότε η σημαία δεν θα πάρει την τιμή αληθής και άρα θα διακοπεί η διαδικασία **Όσο.**

```
    Αν table[j-1] >table[j] τότε
        temp ← table[j-1]
        table[j-1] ← table[j]
        table[j] ← temp
        flag ← αληθής
    Τέλος_αν
Τέλος_επανάληψης
i←i+1
Τέλος_επανάληψης
```

7.13. Ταξινόμηση με επιλογή (selection sort)

7.13.1. Εύρεση μικρότερου στοιχείου μόνο με θέση.

Για να μπορέσουμε να κατανοήσουμε το αλγόριθμό της επιλογής θα πρέπει να ασχοληθούμε λίγο με την εύρεση μικρότερου στοιχείου μόνο με την μεταβλητή της θέσης (εδώ την ονομάζουμε pos)

```
min←A[1]
pos←1
Για  i από 2 μέχρι N
    Αν A[i] <min τότε
        min←A[i]
        pos←i
```

```
pos←1
Για  i από 2 μέχρι N
    Αν A[i] < A[pos]  τότε
        pos←i
    Τέλος_αν
Τέλος_επανάληψης
```

Τέλος_αν

Τέλος_επανάληψης

Αριστερά έχουμε τον κώδικα εύρεσης μικρότερου αριθμού με χρησιμοποίηση δύο μεταβλητών, μία μεταβλητή που κρατάει κάθε φορά τον μικρότερο (min) και μία μεταβλητή που κρατάει κάθε φορά την θέση του μικρότερου (pos).

Δεξιά έχουμε τον κώδικα εύρεσης μικρότερου αριθμού με χρησιμοποίηση μίας μόνο μεταβλητής (pos). Παρατηρούμε ότι όπου χρησιμοποιούμε την μεταβλητή min αντικαθιστούμε με την A[pos]. Πραγματικά αφού η θέση του μικρότερου βρίσκεται στην θέση pos, λογικό είναι ότι το στοιχείο του πίνακα που βρίσκεται σε αυτή την θέση (A[pos]) θα είναι το μικρότερο. Άρα min=A[pos].

7.13.2. Επεξήγηση

Η ταξινόμηση με επιλογή (selection sort), αποτελεί βασικό τρόπο ταξινόμησης, που υλοποιείται σε ένα μονοδιάστατο πίνακα σε τρία βήματα:

1. Επιλογή του ελάχιστου στοιχείου στα ενεργά στοιχεία του πίνακα.
2. Ανταλλαγή του ελάχιστου με το πρώτο στοιχείο της σειράς των ενεργών κάθε φορά αριθμών προς ταξινόμηση.
3. Επανάληψη των βημάτων 1 και 2 για τα υπόλοιπα στοιχεία του πίνακα, που σε κάθε επανάληψη θα είναι λιγότερα από την προηγούμενη φορά.

Έστω ότι έχω τον πίνακα A = [9, 4, 7, 1, 3] .

Αρχικά βρίσκω ότι στα πέντε στοιχεία που ελέγχω το μικρότερο στοιχείο είναι το **1** οπότε και το αλλάζω με το **9** που είναι το πρώτο στοιχείο του πίνακα.

A [**9**, 4, 7, **1**, 3] = [**1**, 4, 7, **9**, 3].

Τώρα αφού το πρώτο στοιχείο είναι το μικρότερο, τα στοιχεία προς ταξινόμηση είναι τέσσερα, από το δεύτερο στοιχείο του πίνακα μέχρι και το πέμπτο. Το μικρότερο στοιχείο από αυτά είναι το **3**, οπότε το αλλάζω με το **4** που είναι το πρώτο στοιχείο των αριθμών προς ταξινόμηση, αλλά ταυτόχρονα βρίσκεται και στην δεύτερη θέση του πίνακα.

A [**1**, **4**, 7, 9, **3** = [**1**, **3**, 7, 9, **4**].

Τώρα αφού τα δύο πρώτα στοιχεία του πίνακα είναι τα μικρότερα, τα στοιχεία προς ταξινόμηση είναι τρία, από το τρίτο στοιχείο του πίνακα μέχρι και το πέμπτο.

Τώρα το μικρότερο στοιχείο από αυτά είναι το 4, οπότε το αλλάζω με το 7 που είναι το πρώτο στοιχείο των αριθμών προς ταξινόμηση, αλλά ταυτόχρονα βρίσκεται και στην τρίτη θέση του πίνακα. A [**1**, **3**, **7**, 9, **4**]= [**1**, **3**, **4**, 9, **7**].

Τώρα αφού τα τρία στοιχεία του πίνακα είναι ταξινομημένα, τα στοιχεία προς ταξινόμηση είναι δύο, από το τέταρτο στοιχείο του πίνακα μέχρι και το πέμπτο. Τώρα το μικρότερο στοιχείο από αυτά είναι το **7**, οπότε το αλλάζω με το **9** που είναι το πρώτο στοιχείο των αριθμών προς ταξινόμηση, αλλά ταυτόχρονα βρίσκεται και στην τέταρτη θέση του πίνακα. A [**1**, **3**, **4**, **9**, **7**]= [**1**, **3**, **4, 7**, **9**].

Αφού τα τέσσερα από τα πέντε στοιχεία του πίνακα είναι ταξινομημένα ο πίνακας είναι ταξινομημένος. A = [**1**, **3**, **4**, **7**, 9].

Ο αλγόριθμος ταξινόμησης με επιλογή είναι παρακάτω:

```
ΠΡΟΓΡΑΜΜΑ Selection_Sort
ΜΕΤΑΒΛΗΤΕΣ
ΑΚΕΡΑΙΕΣ: A[100], pos, min, i, j ,temp
ΑΡΧΗ
ΔΙΑΒΑΣΕ N
ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ N
    ΓΡΑΨΕ 'Δώσε το', i, ' στοιχείο του πίνακα'
    ΔΙΑΒΑΣΕ A[i]
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ N-1
    min ← A[i]
    pos ← i
    ΓΙΑ j ΑΠΟ i + 1 ΜΕΧΡΙ N
        ΑΝ  A[j] < min ΤΟΤΕ
            min ← A[j]
            pos ← j
        ΤΕΛΟΣ_ΑΝ
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
    temp ← A[pos]
    A[pos] ← A[i]
    A[i] ← temp
```

A =	**9**	**4**	**7**	**1**	**3**
	1	**4**	**7**	**9**	**3**
	1	**3**	**7**	**9**	**4**
	1	**3**	**4**	**9**	**7**
	1	**3**	**4**	**7**	**9**

Εναλλακτικά μπορούν να μπουν οι δύο εντολές

A[pos] ← A[i]

A[i] ← min

Καθώς το min και το A[pos] έχουν την ίδια τιμή

```
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΓΡΑΨΕ 'Εκτύπωση με ταξινομημένα τα στοιχεία'
ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ N
    ΓΡΑΨΕ A[i]
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

7.14. Εύρεση ανά γραμμή ή ανά στήλη κάποιου χαρακτηριστικού

Όταν θέλω να βρω για κάθε γραμμή ή για κάθε στήλη ένα συγκεκριμένο χαρακτηριστικό για τα στοιχεία ενός δισδιάστατου πίνακα τότε θα πρέπει αρχικά η σάρωση των στοιχείων του πίνακα να είναι αντίστοιχα κατά γραμμές ή κατά στήλες. Πάντα τοποθετώ μία ή περισσότερες εντολές ανάμεσα στα δύο «για» και στα δύο «τέλος_επανάληψης» της διπλής επαναληπτικής διαδικασίας.

7.14.1. Πρόβλημα Υ1

Να φτιαχτεί αλγόριθμος που να βρίσκει τον μέσο όρο ανά γραμμή σε έναν δισδιάστατο πίνακα ΝΧΜ.

Λύση: Ξέρω ότι ο μέσος όρος είναι το άθροισμα δια το πλήθος. Άρα χρειάζομαι έναν αθροιστή. Ο αθροιστής αυτός μπαίνει ανάμεσα στα δύο «για». Για i=1 αρχικά μηδενίζω τον αθροιστή για την πρώτη γραμμή (S←0). Μετά μπαίνω στη εσωτερική επανάληψη του j και αθροίζω όλα τα στοιχεία του πίνακα στον αθροιστή (S←S+ table [i,j]). Όταν τελειώσει η εσωτερική επανάληψη τότε ο αθροιστής έχει το άθροισμα όλων των στοιχείων της πρώτης γραμμής του πίνακα. Τοποθετώ με μια εντολή εκχώρησης τον μέσο όρο της πρώτης γραμμής ως το πρώτο στοιχείο του πίνακα ΜΟ (Μο[1]←S/M). Όταν το i πάρει την τιμή δύο τότε μηδενίζω τον αθροιστή για την δεύτερη γραμμή και συνεχίζω με τον ίδιο τρόπο και για τις επόμενες γραμμές. Η αλλαγή τιμής του αθροιστή σε μηδέν δεν με προβληματίζει καθώς πιο πριν το άθροισμα των στοιχείων της προηγούμενης γραμμής έχει ήδη αποθηκευτεί στο πίνακα ΜΟ. Αντίστοιχα δουλεύω και για την εύρεση του μέσου όρου ανά στήλη, μόνο που εδώ η σάρωση αναγκαστικά είναι κατά στήλες. Παρακάτω φαίνονται οι κώδικες:

```
Εύρεση μέσου όρου ανά γραμμή
Για i από 1 μέχρι N
    S←0
   Για j από 1 μέχρι M
       S←S+ table [i,j]
   Τέλος_επανάληψης
    Mo[i]←S/M
    Εμφάνισε  Mo[i]
Τέλος_επανάληψης
```

```
Εύρεση μέσου όρου ανά στήλη
Για j από 1 μέχρι M
    S←0
    Για i από 1 μέχρι N
        S←S+ table [i,j]
    Τέλος_επανάληψης
     Mo[j]←S/N
     Εμφάνισε  Mo[j]
Τέλος_επανάληψης
```

7.14.2. Πρόβλημα Υ2

Να φτιαχτεί αλγόριθμος που να βρίσκει το μεγαλύτερο στοιχείο ανά γραμμή σε έναν δισδιάστατο πίνακα NXM.

Λύση: Για την πρώτη γραμμή έστω ότι το πρώτο στοιχείο της γραμμής είναι το μεγαλύτερο και το τοποθετώ στην μεταβλητή max. Έτσι το max θα πάρει την τιμή table[1,1]. Για την δεύτερη γραμμή το max θα πάρει την τιμή του πρώτου στοιχείου της, δηλαδή το στοιχείο table[2,1]. Γενικότερα για την γραμμή i το πρώτο στοιχείο είναι το table[i,1]. Εμφανίζοντας την τρέχουσα τιμή του max μετά το τέλος της εσωτερικής επανάληψης μου δίνει την δυνατότητα να του δώσω την τιμή του πρώτου στοιχείου της επόμενης γραμμής, χωρίς να νοιάζομαι ότι έχω χάσει κάποια πληροφορία. Όμοια συμβαίνει και στην εύρεση μεγαλύτερου ανά στήλη.

```
Εύρεση μεγαλύτερου ανά γραμμή
Για i από 1 μέχρι N
    max←table[i,1]
   Για j από 2 μέχρι M
       Αν table [i,j] >max τότε
        max← table [i,j]
       Τελος_αν
   Τέλος_επανάληψης
   Εμφάνισε  max
Τέλος_επανάληψης
```

```
Εύρεση μεγαλύτερου  ανά στήλη
Για j από 1 μέχρι M
    max←table[1,j]
    Για i από 2 μέχρι N
        Αν table [i,j] >max τότε
         max← table [i,j]
        Τελος_αν
    Τέλος_επανάληψης
    Εμφάνισε  max
Τέλος_επανάληψης
```

7.14.3. Πρόβλημα Υ3

Να φτιαχτεί αλγόριθμος που να ταξινομεί τα στοιχεία ενός δισδιάστατου πίνακα ΝΧΜ διαστάσεων ανά γραμμή κατά αύξουσα σειρά.

Λύση: Γνωρίζουμε ότι για να ταξινομήσουμε τα στοιχεία ενός μονοδιάστατου πίνακα με την μέθοδο της ευθείας ανταλλαγής (φυσαλίδα) χρειαζόμαστε δύο επαναληπτικές διαδικασίες. Η εσωτερική επανάληψη χρειάζεται για να συγκρίνουμε τα στοιχεία του πίνακα ανά δύο από το πίσω μέρος του πίνακα προς την αρχή του. Η εξωτερική επανάληψη χρειάζεται γιατί αυτή η σύγκρισή πρέπει να γίνει περισσότερες από μία φορές. Στον δισδιάστατο πίνακα χρειαζόμαστε πάντα διπλή επαναληπτική διαδικασία για να διαχειριστούμε τα στοιχεία του. Από τα παραπάνω προκύπτει ότι ταξινόμηση φυσαλίδα ανά γραμμή χρειάζεται τρείς επαναλήψεις με αντίστοιχους τρείς δείκτες. Ο δείκτης i διαχειρίζεται τις γραμμές, ο δείκτης j διαχειρίζεται τις στήλες και ο δείκτης κ χρειάζεται για να γίνει η διαδοχική σύγκριση των ανά δύο στοιχείων του πίνακα σε κάθε γραμμή περισσότερες από μία φορές. Όμοια ισχύουν και για την ταξινόμηση ανά στήλη ενός δισδιάστατου πίνακα.

Ταξινόμηση ανά γραμμή

```
Για i από 1 μέχρι N
    Για κ από 2 μέχρι M
        Για j από M μέχρι κ με βήμα –1
            Αν table[i,j-1] >table[i,j] τότε
                temp← table[i,j-1]
                table[i,j-1]← table[i,j]
                table[i,j]← temp
            Τέλος_αν
        Τέλος_επανάληψης
    Τέλος_επανάληψης
Τέλος_επανάληψης
```

Ταξινόμηση ανά στήλη

```
Για j από 1 μέχρι M
    Για κ από 2 μέχρι N
        Για i από N μέχρι κ με βήμα –1
            Αν table[i-1,j] >table[i,j] τότε
                temp← table[i-1,j]
                table[i-1,j]← table[i,j]
                table[i,j]← temp
            Τέλος_αν
        Τέλος_επανάληψης
    Τέλος_επανάληψης
Τέλος_επανάληψης
```

7.15. Ψευδοκώδικας – Γλώσσα – Ομοιότητες Διαφορές

1. Οι εντολές της ΓΛΩΣΣΑΣ είναι πάντα με κεφαλαία, ενώ οι μεταβλητές είναι με πεζά ή κεφαλαία. Στην ψευδογλώσσα μόνο το πρώτο γράμμα των εντολών είναι με κεφαλαία.
2. Στη ΓΛΩΣΣΑ πρέπει να υπάρχουν δηλώσεις τύπων δεδομένων των μεταβλητών που χρησιμοποιούνται. Αυτό οφείλεται στο ότι στην αρχή της

εκτέλεσης του προγράμματος, δεσμεύεται συγκεκριμένος αριθμός θέσεων (bytes) στην κύρια μνήμη για την προσωρινή αποθήκευση των μεταβλητών. Η κύρια μνήμη έχει περιορισμένη χωρητικότητα, γεγονός που πρέπει να λαμβάνεται υπόψη κατά τη δήλωση των τύπων δεδομένων των μεταβλητών. Οι δηλώσεις αυτές πρέπει να γίνονται κατά τέτοιο τρόπο ώστε να μην σπαταλάτε άσκοπα χώρος στη μνήμη. Στο επίπεδο της αλγοριθμικής ψευδογλώσσας, δεν είναι απαραίτητες οι δηλώσεις τύπων δεδομένων των μεταβλητών που χρησιμοποιούνται.

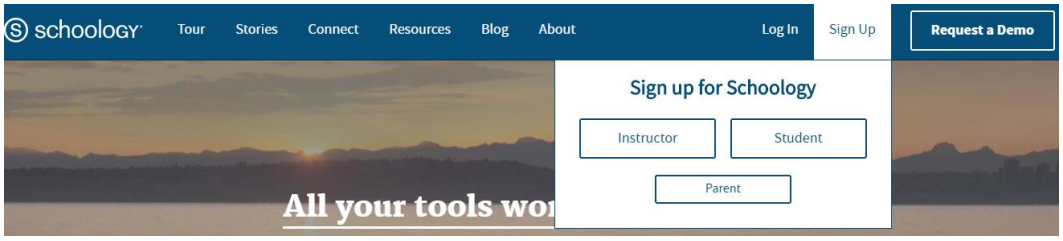
3. Κάθε πρόγραμμα αρχίζει με τη λέξη **ΠΡΟΓΡΑΜΜΑ** σε αντίθεση με τον ψευδοκώδικα που άρχιζει με τη λέξη **Αλγόριθμος**.
4. Στο πρόγραμμα, μετά τις δηλώσεις των μεταβλητών υπάρχει η λέξη **ΑΡΧΗ** για να δηλώνει την αρχή των εκτελέσιμων εντολών, κάτι που δεν υπάρχει στον ψευδοκώδικα.
5. Στο πρόγραμμα, δηλώνεται ο τερματισμός του προγράμματος με το **ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ**, ενώ στον ψευδοκώδικα με την εντολή **Τέλος** ακολουθούμενη από το όνομα του προγράμματος.
6. Στην γλώσσα κάθε εντολή θα πρέπει να γράφεται σε ξεχωριστή γραμμή. Αν μια εντολή πρέπει να συνεχιστεί και στην επόμενη γραμμή, τότε ο πρώτος χαρακτήρας αυτής της γραμμής πρέπει να είναι ο χαρακτήρας &.
7. Η ΓΛΩΣΣΑ περιέχει έτοιμες γνωστές συναρτήσεις από τα μαθηματικά οι οποίες χρησιμοποιούνται στα προγράμματα.
8. Η ΓΛΩΣΣΑ υποστηρίζει για την εμφάνιση των αποτελεσμάτων την εντολή **ΓΡΑΨΕ** ενώ στο ψευδοκώδικα χρησιμοποιούνται οι εντολές **Εμφάνισε** ή **Εκτύπωσε** ανάλογα με την εκφώνηση.

ΠΑΡΑΡΤΗΜΑ

Ανάλυση της ψηφιακής πλατφόρμας schoology.com

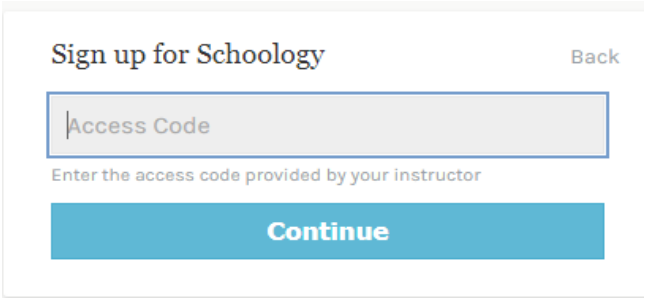
Αρχικά πληκτρολογείτε https://www.schoology.com/

Έπειτα επιλέγετε **Sign Up** και πατάτε το πλήκτρο **Student**.



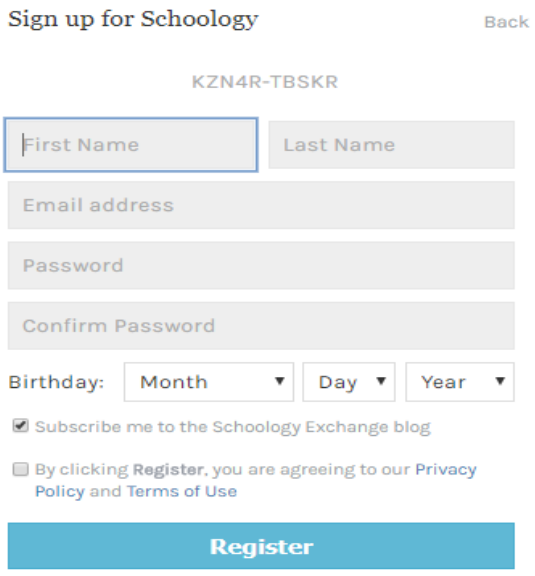
Π-1 Sign Up – Εικόνα 1

Δίνετε τον κωδικό τάξης που έχετε από τον καθηγητή σας.



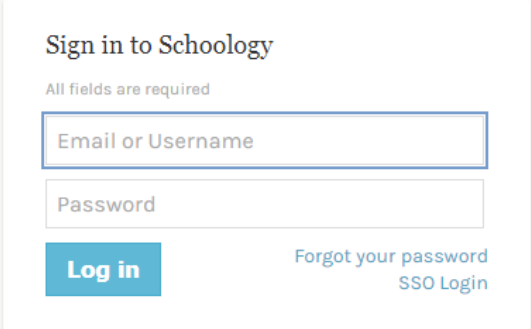
Π-2 Sign Up – Εικόνα 2

Συμπληρώνετε τα στοιχεία σας και πατάτε το κουμπί **Register**. Προσοχή να δώσετε σωστό email γιατί στέλνεται σε αυτό μήνυμα επιβεβαίωσης.



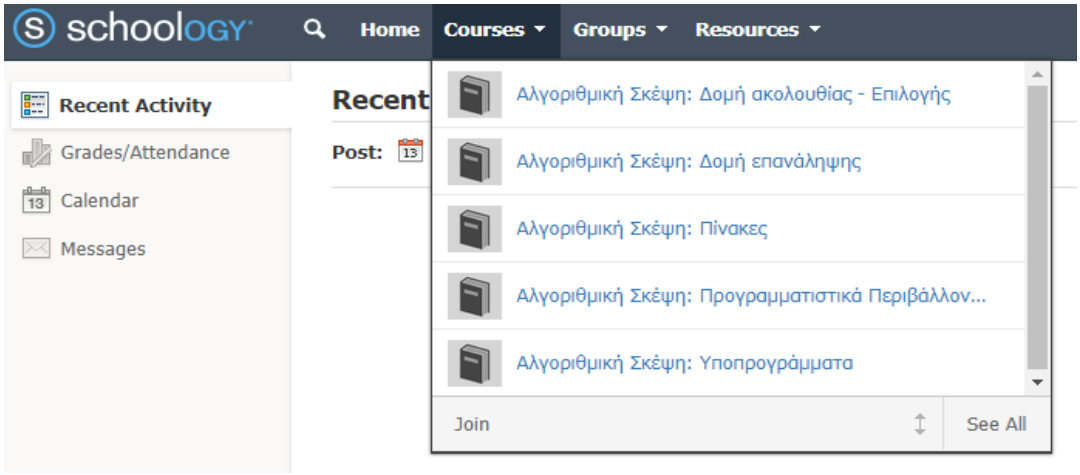
Π-3 Sign Up – Εικόνα 3

Τις επόμενες φορές απλά πατάτε το κουμπί Sign in και δίνετε το email σας και το password και μετά πατάτε Log in.



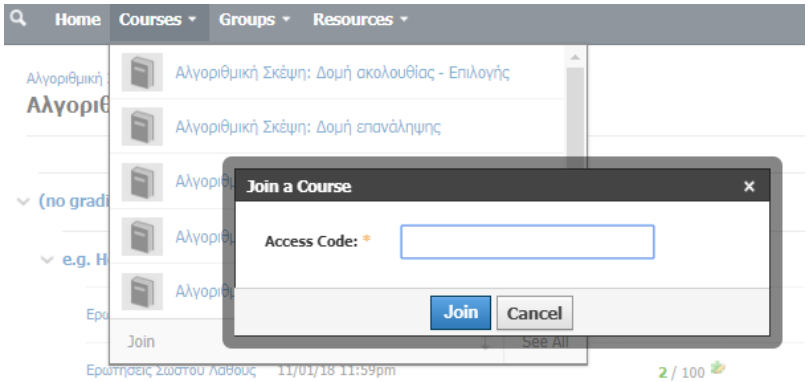
Π-4 Sign in

6. Πατάτε στο μενού **Courses** Εμφανίζονται όλα τα μαθήματα που έχετε εγγραφεί. Αρχικά μπορεί να έχετε εγγραφεί μόνο σε ένα μάθημα, πχ. σε αυτό της Δομής της Ακολουθίας – Επιλογής.



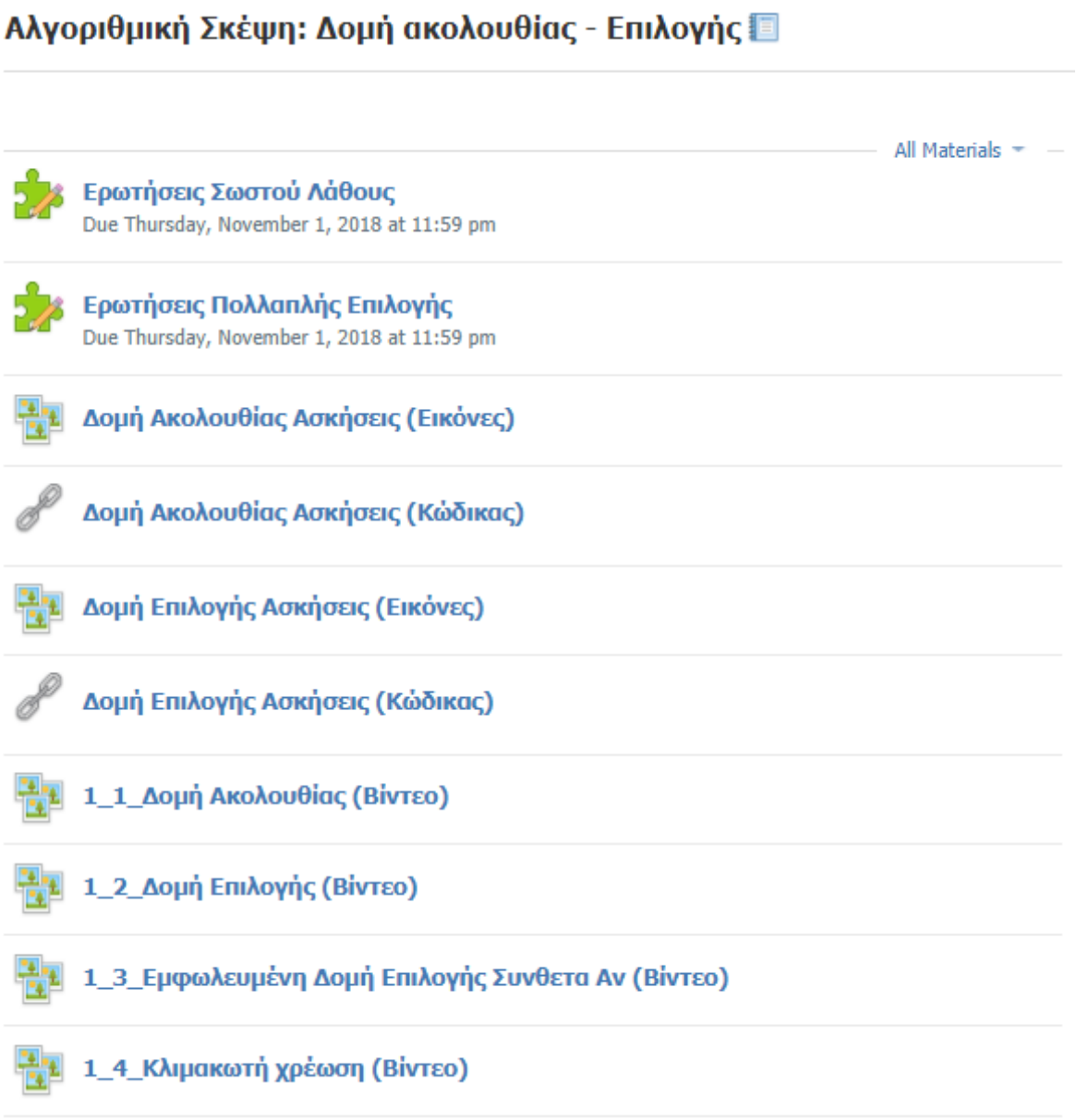
Π-5 Courses

Έχετε πάντα την δυνατότητα να πατήσετε το πλήκτρο **Join** και να εγγραφείτε σε ένα καινούργιο μάθημα



Π-6 Join

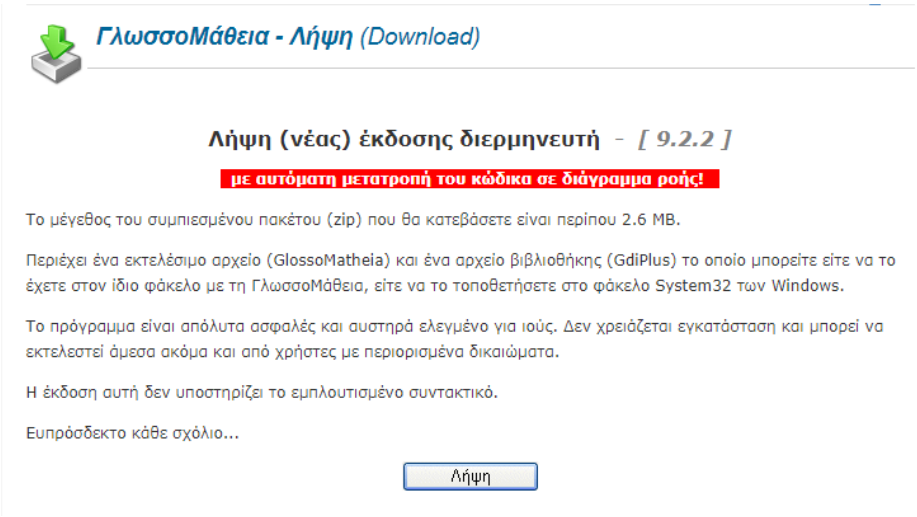
Κάνοντας κλικ πάνω στην δομή της Ακολουθίας έχετε την παρακάτω εικόνα και επιλέγετε ανάμεσα σε ερωτήσεις σωστού λάθους , πολλαπλής επιλογής, ασκήσεις σε μορφή εικόνας , ασκήσεις σε μορφή κώδικα, καθώς και βίντεο.



Π-7 Επιλογές μαθητή

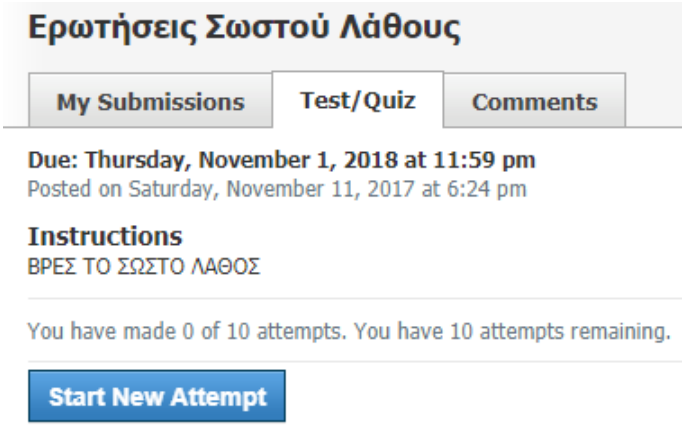
Για να τρέξετε τον κώδικα πρέπει να τον κατεβάσετε στον υπολογιστή σας και να κατεβάσετέ επίσης και το πρόγραμμα **Γλωσσομάθεια** από την παρακάτω διεύθυνση πατώντας το κουμπί **Λήψη**.

http://spinet.gr/glossomatheia/download/



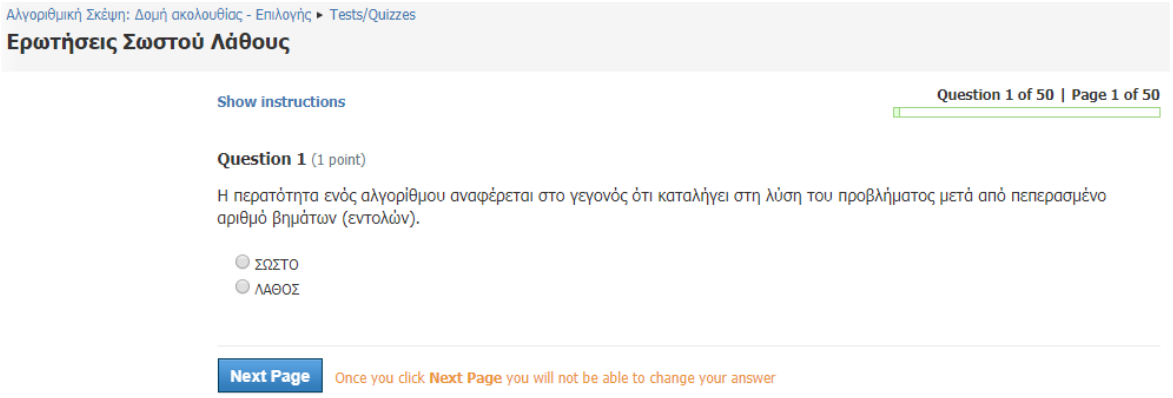
Π-8 Γλωσσομάθεια

Επιλέγοντας τις ερωτήσεις σωστού λάθους βγαίνει η παρακάτω εικόνα και πατάτε **Start New Attemp**.



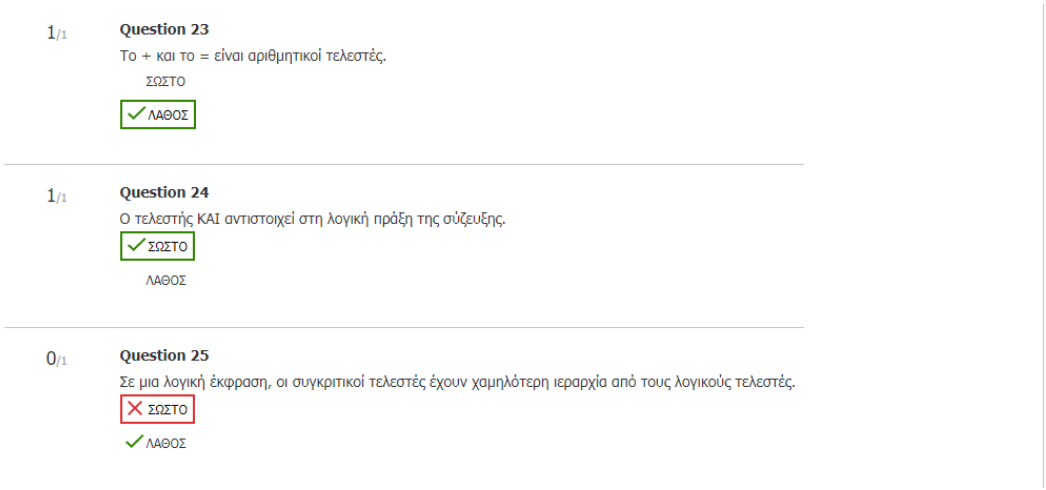
Π-9 Ερωτήσεις σωστού λάθους – Εικόνα 1

Και στην συνέχεια πατάτε **Next Page** μέχρι να ολοκληρωθεί το τεστ.



Π-10 Ερωτήσεις σωστού λάθους – Εικόνα 2

Υπάρχει η άμεση δυνατότητα βαθμολόγησης και γνωστοποίησης των αποτελεσμάτων στον εκπαιδευόμενο.



Π-11 Ερωτήσεις σωστού λάθους – Εικόνα 3

Υπάρχει επίσης δυνατότητα επανάληψης των ερωτήσεων.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Saliba, G., Rankine, L., Cortez, H. (2013) *"Fundamentals of Blended Learning"* Univescity of Western Sydney (σελίδα 4)

Horn, B., M., Staker, H. (2011) "*The Rise of K–12 Blended Learning".* Innosight Institute (σελίδες 5-7)

Bath, D., Bourke, J. (2010) Getting started with Blended Learning Griffith Institute for Higher Education (σελίδες 25-27)

Τσιωτάκης, Α.Π. (2008) "*Ανάπτυξη Εφαρμογών σε Προγραμματιστικό Περιβάλλον Γ΄ Λυκείου Τεχνολογικής Κατεύθυνσης, β΄ τεύχος".* Αθήνα: Εκδόσεις Σαββάλας.

Δημητριάδης, Σ., Καραγιαννίδης, Χ., Πομπόρτσης, Α., Τσιάτσος, Θρ. (2008). *"Ευέλικτη μάθηση με χρήση Τεχνολογιών Πληροφορίας & Επικοινωνιών".* Θεσσαλονίκη: Εκδόσεις Τζιόλα,

Παπουτσάς, Γ. (2008) "*Ανάπτυξη εφαρμογών σε προγραμματιστικό περιβάλλον - Γ΄ Τάξη ενιαίου Λυκείου Τεχνολογική Κατεύθυνση".* Αθήνα: Εκδόσεις Έναστρον.

Ντζιος, Ν.Κ., Κοψίνης, Χ.Ι. (2008) "*Ανάπτυξη Εφαρμογών σε Προγραμματιστικό Περιβάλλον Γ΄ Λυκείου".* Αθήνα: Εκδόσεις Σαββάλας.

Τσιωτάκης, Α.Π. (2006) "*Ανάπτυξη Εφαρμογών σε Προγραμματιστικό Περιβάλλον Γ΄ Λυκείου Τεχνολογικής Κατεύθυνσης, α΄ τεύχος".* Αθήνα: Εκδόσεις Σαββάλας.

Μιχαλακόπουλος, Κ., Τσιαπάλας, Σ. (2004) "*Ανάπτυξη εφαρμογών σε προγραμματιστικό περιβάλλον - Γ΄ Λυκείου (Τεχνολογικής Κατεύθυνσης)".* Αθήνα: Ελληνοεκδοτική.

Περγάντης, Ν. (2000) "*Ανάπτυξη εφαρμογών σε Προγραμματιστικό περιβάλλον".* Αθήνα: Εκδόσεις Ιανός.

ΔΙΑΔΙΚΤΥΑΚΟΙ ΤΟΠΟΙ

Ηλεκτρονική μάθηση Μοντέλα ηλεκτρονικής μάθησης

http://eprl.korinthos.uop.gr/wiki/index.php/Ηλεκτρονική_μάθηση_Μοντέλα_ηλεκτρονικής_μάθησης

Blended learning

https://en.wikipedia.org/wiki/Blended_learning

Blended Learning Advantages and Disadvantages

https://www.easy-lms.com/help/lms-knowledge-center/blended-learning-advantages/item10386

Blended Learning: An Innovative Approach

http://www.hrpub.org/download/20161230/UJER16-19508256.pdf

Blended Learning A synthesis of research findings in Victorian education 2006-2011

http://www.education.vic.gov.au/documents/about/research/blendedlearning.pdf

6 Disadvantages of Blended Learning You Have to Cope With

https://myelearningworld.com/6-disadvantages-of-blended-learning/