Aristotle Certification Training & Assessment

Τεχνοβλαστός Αριστοτελείου Πανεπιστημίου Θεσσαλονίκης

Προγραμματισμός Ι

(βασικό επίπεδο)

Ανακαλύπτοντας τον Προγραμματισμό

Έκδοση 1

ACTA ΑΕ – Τεχνοβλαστός Αριστοτελείου Πανεπιστημίου Θεσσαλονίκης

Εγνατίας 1, 54630, Τηλ.: 2310 510 870, Fax: 2310 510 871, email: info@acta.edu.gr , www.acta.edu.gr

ΣΤΟΙΧΕΙΑ ΣΥΓΓΡΑΦΕΑ

Ονοματεπώνυμο Συγγραφέα: ΜΙΧΑΗΛ Ι. ΣΤΑΜΑΤΟΠΟΥΛΟΣ (Πληροφορικός ΠΕ19/ΠΕ86).

Είναι πτυχιούχος Πληροφορικός, του Τμήματος Πληροφορικής, της Σχολής Θετικών Επιστημών & Τεχνολογίας, Ε.Α.Π., Πατρών. Κατέχει παιδαγωγική και διδακτική επάρκεια στην εκπαίδευση της Πληροφορικής και έχει μεταπτυχιακή επιμόρφωση, σαν εκπαιδευτικός ειδικής αγωγής όπως επίσης και σαν εκπαιδευτής ενηλίκων, από το Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών (Ε.Κ.Π.Α.). Είναι επιστημονικός συνεργάτης – ερευνητής, στο Εργαστήριο Ευφυών Συστημάτων (i-Lab), του Τμήματος Πολιτισμικής Τεχνολογίας και Επικοινωνίας (Τ.Π.Τ.Ε.), της Σχολής Κοινωνικών Επιστημών, του Πανεπιστημίου Αιγαίου. Εργάζεται σαν μηχανικός λογισμικού και σαν εκπαιδευτικός Πληροφορικής για περισσότερα από 25 έτη και υπήρξε εκπρόσωπος στην Ελλάδα, κορυφαίων διεθνώς εταιρειών τεχνολογίας (FARO Technologies, CADZone, Trancite, A-T Solution).

Aristotle Certification Training & Assessment
Τεχνοβλαστός Αριστοτελείου Πανεπιστημίου Θεσσαλονίκης

Περιεχόμενα

Aristotle Certification Training & Assessment
Τεχνοβλαστός Αριστοτελείου Πανεπιστημίου Θεσσαλονίκης

ΚΕΦΑΛΑΙΟ 2^{ο}

«ΜΗΧΑΝΕΣ ΚΑΙ Η ΕΠΙΣΤΗΜΗ ΤΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ»

Aristotle Certification Training & Assessment
Τεχνοβλαστός Αριστοτελείου Πανεπιστημίου Θεσσαλονίκης

ΚΕΦΑΛΑΙΟ 3ο

«ΑΛΓΟΡΙΘΜΟΣ»

ΚΕΦΑΛΑΙΟ 4ο

«ΜΟΡΦΕΣ ΑΛΓΟΡΙΘΜΩΝ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ»

ΚΕΦΑΛΑΙΟ 5^{ο}

«ΓΛΩΣΣΕΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ»

ΚΕΦΑΛΑΙΟ 6ο

«ΑΛΓΟΡΙΘΜΟΙ ΚΑΙ ΓΛΩΣΣΕΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ»

Aristotle Certification Training & Assessment
Τεχνοβλαστός Αριστοτελείου Πανεπιστημίου Θεσσαλονίκης

ΚΕΦΑΛΑΙΟ 7^{ο}

«ΑΠΟ ΤΑ ΔΙΑΓΡΑΜΜΑΤΑ ΚΑΙ ΤΗΝ ΨΕΥΔΟΓΛΩΣΣΑ, ΣΤΗΝ ΓΛΩΣΣΑ C»

Aristotle Certification Training & Assessment
Τεχνοβλαστός Αριστοτελείου Πανεπιστημίου Θεσσαλονίκης

Γλωσσάριο σημαντικών όρων

Βιβλιογραφία

Οδηγίες για περαιτέρω μελέτη

ΕΙΣΑΓΩΓΗ

Ο θαυμαστός καινούργιος κόσμος των τεχνολογιών, ο ψηφιακός μετασχηματισμός και η κοινωνία των πληροφοριών αλλάζουν τις συνήθειες των ανθρώπων και μας οδηγούν ήδη, στην πορεία προς την τέταρτη βιομηχανική επανάσταση. Το διαδίκτυο των πραγμάτων, οι σύγχρονες επικοινωνίες, η τεχνητή νοημοσύνη, η φορητή υπολογιστική ισχύ, η κυβερνητική, η ευρεία χρήση ισχυρών υπολογιστών, οι μεγάλες συλλογές δεδομένων, δημιουργούν την νέα πραγματικότητα της ψηφιακής οικονομίας. Η επιστήμη της *Πληροφορικής* καθίσταται ο βασικότερος πυλώνας εξελίξεων για τις κοινωνίες, προσφέροντας επωφελείς λύσεις σε πολλούς κρίσιμους τομείς. Η ανάπτυξη και η διείσδυση των νέων αυτών τεχνολογιών, στο σύγχρονο κοινωνικό και οικονομικό περιβάλλον, συγκεντρώνει το έντονο ενδιαφέρον των περισσοτέρων ανθρώπων και δημιουργεί τις νέες «ψηφιακές γενιές» ανθρώπων.

Το σημερινό εκπαιδευτικό τοπίο οφείλει να προετοιμάζει κατάλληλα τους αυριανούς χρήστες της κοινωνίας των πληροφοριών και να ενισχύει τις ψηφιακές τους δεξιότητες, προκειμένου αυτοί, να μπορούν να αντιμετωπίζουν τις τεχνολογικές προκλήσεις και να αξιοποιούν τις ευκαιρίες της εποχής.

Στα πλαίσια αυτά, το σύγγραμμα , *Προγραμματισμός Ι (βασικό επίπεδο), Ανακαλύπτοντας τον Προγραμματισμό,* παρουσιάζει με συστηματικό τρόπο όλες τις απαραίτητες γνώσεις, προκειμένου να μπορεί κάποιος να κατανοεί τις αρχές του προγραμματισμού των ηλεκτρονικών υπολογιστών.

Ειδικότερα, εξετάζονται, όλες οι βασικές έννοιες του προγραμματισμού και με παραδείγματα κλιμακούμενης δυσκολίας, εφαρμόζονται τεχνικές ανάπτυξης αλγορίθμων με διαγράμματα και ψευδοκώδικα. Το σύγγραμμα πλαισιώνεται με την εισαγωγή στην γλώσσα προγραμματισμού C, την πιο δημοφιλή και σύγχρονη γλώσσα προγραμματισμού. Οι εκπαιδευόμενοι αποκτούν ικανότητες διερμηνείας προγραμμάτων γραμμένων σε γλώσσα C και καθοδηγούνται στο να μπορούν να μετατρέψουν έναν αλγόριθμο, σε ένα σύνολο συγκεκριμένων απλών εντολών στην γλώσσα προγραμματισμού C.

1. ΠΡΟΒΛΗΜΑ

Σκοπός και επιμέρους στόχοι

Ο σκοπός του πρώτου κεφαλαίου είναι να εισάγει τον εκπαιδευόμενο στην έννοια *πρόβλημα* και τις τεχνικές με τις οποίες θα πρέπει να αναγνωρίζει τα προβλήματα, να τα προσεγγίζει και να αναζητά όλα εκείνα τα γνωστικά στοιχεία ώστε να οδηγείται σε κατάλληλες επιλύσεις.

Προσδοκώμενα αποτελέσματα

Με την μελέτη του πρώτου κεφαλαίου οι εκπαιδευόμενοι θα μπορούν,

- να εντοπίζουν καταστάσεις οι οποίες προκαλούν προβλήματα,
- να επενδύουν στην αξία της παρατήρησης ενός προβλήματος,
- να εντοπίζουν τα συστατικά επιμέρους μέρη ενός ορατού προβλήματος,
- να διακρίνουν όλα εκείνα τα στοιχεία τα οποία θα αποτελέσουν δεδομένα για την αντιμετώπιση ενός προβλήματος,
- να αντιλαμβάνονται τη σημασία εύρεσης του ζητούμενου αποτελέσματος,
- να περιγράφουν τα στάδια αντιμετώπισης ενός προβλήματος,
- να διασπούν ένα πρόβλημα σε άλλα μικρότερα,
- να καταλαβαίνουν βασικές έννοιες που αφορούν την κατηγοριοποίηση των προβλημάτων,
- να υιοθετούν την πεποίθηση ότι μπορούν και πρέπει να προσπαθούν να επιλύουν τα προβλήματα που αντιμετωπίζουν,
- να είναι σε θέση να περιγράψουν ένα πρόβλημα με συνέπεια και πληρότητα.

Έννοιες – Λέξεις Κλειδιά

Συστηματική παρατήρηση, Κατανόηση προβλήματος, Διάκριση δεδομένων εισόδου, Καθορισμός ζητουμένων εξόδου, Αποδόμηση προβλήματος, Κατηγοριοποίηση προβλημάτων, Κατάστρωση σχεδίου επίλυση προβλήματος.

Εισαγωγικές Παρατηρήσεις

Με τον όρο, *πρόβλημα* εννοούμε οποιαδήποτε κατάσταση η οποία, εμποδίζει την πορεία προς έναν συγκεκριμένο και προκαθορισμένο στόχο. Μια κατάσταση χαρακτηρίζεται σαν πρόβλημα την στιγμή που αναγνωρίζεται συνειδητά από κάποιον ζωντανό οργανισμό, ότι αποτελεί εμπόδιο. Μια κατάσταση γίνεται πρόβλημα όταν προκαλεί σκέψεις και οδηγεί σε ενέργειες για επίλυση. Οι καταστάσεις που οδηγούνται στο να χαρακτηρίζονται προβλήματα, απαιτούν λύσεις και οι λύσεις αυτές δεν είναι άμεσα ορατές.

Ο όρος *πρόβλημα*, δεν αφορά αποκλειστικά και μόνο την ανθρώπινη καθημερινότητα. Τα προβλήματα απασχολούν επάξια και κάθε άλλον οργανισμό στο ζωικό, αλλά και το φυτικό βασίλειο. Όλοι οι ζωντανοί οργανισμοί του κόσμου που μας περιβάλει, αντιμετωπίζουν και καλούνται να επιλύουν προβλήματα. Είναι πολύ εύκολο να παρατηρήσουμε γύρω μας, τις προσπάθειες που κάνουν οι ζωντανοί οργανισμοί για να επιλύουν τα δικά τους μικρά καθημερινά προβλήματα. Ένα πρόβλημα το οποίο θα απασχολήσει έναν εκπρόσωπο ενός είδους του ζωικού μας βασιλείου, μπορεί να επιλυθεί στιγμιαία σε ατομικό επίπεδο από τον ίδιο τον εκπρόσωπο, μπορεί να επιλυθεί στιγμιαία σε ομαδικό επίπεδο από μια ομάδα εκπροσώπων του ίδιου είδους ή μπορεί ακόμα και να επιλυθεί σταδιακά από ολόκληρο το είδος με την αλλαγή της γενικής συμπεριφοράς του είδους, στην πορεία τους μέσα στον χρόνο.

Πολλά επιλυμένα προβλήματα είναι δύσκολο να εντοπιστούν καθώς η αναγνώριση τους και μόνο, είναι μια περίπλοκη και στοχαστική διαδικασία. Για παράδειγμα, είναι ερώτημα, εάν τα

αγκάθια επάνω στους κορμούς των φυτών και των δέντρων, αποτελούν τυχαία γεγονότα ή είναι μέρος της επίλυσης ενός σύνθετου προβλήματος από μια συνειδητή διαδικασία. Καταστάσεις που δημιουργούν εμπόδια και επιλύονται με κάποιον τρόπο, εμφανίζονται και στον υλικό κόσμο αλλά οι περιπτώσεις αυτές δεν μας απασχολούν καθώς δεν υπεισέρχεται στην επίλυση τους κανένας παράγοντας, σκέψης, ικανότητας, παρατήρησης ή συνείδησης.

1.1. Άνθρωπος και Προβλήματα

Ο άνθρωπος σαν δημιούργημα, είναι μια σύνθετη κατασκευή που μοιάζει να έχει κατασκευαστεί ειδικά για να επιλύει προβλήματα. Θα μπορούσε ίσως ακόμα καλύτερα να διατυπωθεί η άποψη ότι, ο άνθρωπος εξελίχθηκε σταδιακά, στην καλύτερη γνωστή «μηχανή» που μπορεί συνειδητά να αναλύει και να επιλύει εξαιρετικά πολύπλοκα προβλήματα, τα οποία αντιμετωπίζει συνεχώς και καθημερινά στην ζωή του. Τα προβλήματα, μικρά ή μεγάλα, σύνθετα ή απλά, τα επιλύουμε ατομικά, ομαδικά, στιγμιαία, σε μικρά ή σε μεγάλα χρονικά διαστήματα, αλλά και εξελικτικά προσαρμόζοντας την συμπεριφορά μας, σαν είδος, στην πορεία μας μέσα στον χρόνο.

Τόσο, ο εντοπισμός, όσο και η αντιμετώπιση, αλλά και η διατύπωση ενός αναγνωρίσιμου προβλήματος, αποτελούν διαδικασίες που απαιτούν ορθολογική σκέψη, ιδιαίτερες αναλυτικές και συνθετικές ικανότητες και δυνατότητα γραπτής ή προφορικής έκφρασης. Από την στιγμή που αναγνωρίζουμε ένα πρόβλημα, κάθε προσπάθεια επίλυσης του είναι καταδικασμένη σε

αποτυχία, αν προηγουμένως δεν έχει γίνει πλήρως κατανοητή η αιτία που προκαλεί το πρόβλημα. Επιπλέον, από την στιγμή που ένα πρόβλημα μας γίνεται γνωστό μέσα από περιγραφές ενός άλλου παρατηρητή, η διατύπωση του προβλήματος προς εμάς θα πρέπει να είναι απολύτως σαφής και ορθή. Σε διαφορετική περίπτωση, η επίλυση καθίσταται δυσχερής και είναι καταδικασμένη επίσης σε αποτυχία, αφού δεν γίνεται πραγματικά αντιληπτός ο λόγος που προκαλεί το πρόβλημα, ενώ ενυπάρχει και ο κίνδυνος να προταθεί λύση για ένα εντελώς διαφορετικό πρόβλημα.

Είναι θεμιτό να υιοθετούμε ως εκπαιδευόμενοι την αντίληψη, πως κάθε πρόβλημα μπορεί να έχει την λύση του. Είναι επίσης θεμιτή η αντίληψη, πως εμείς από μόνοι μας μπορούμε να επιλύσουμε κάθε πρόβλημα. Για να αυξήσουμε την ικανότητα μας να αναλύουμε και να επιλύουμε ένα πρόβλημα θα πρέπει να μάθουμε να το αποδομούμε.

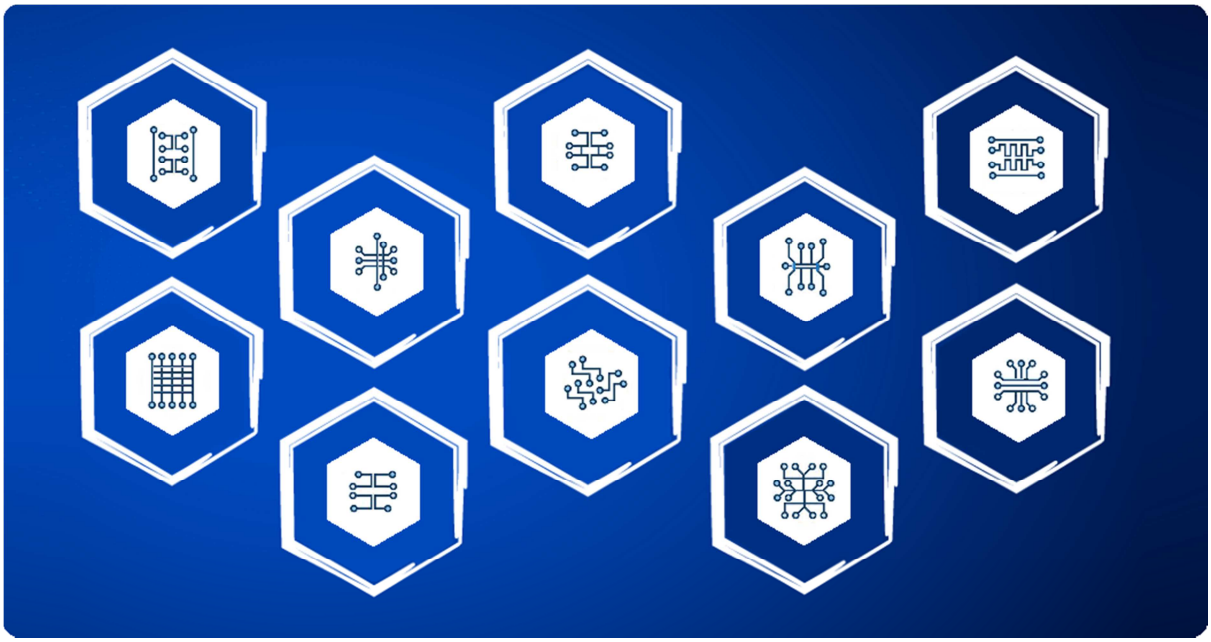
1.2. Αποδόμηση Προβλημάτων

Με το όρο *αποδόμηση προβλήματος*, εννοούμε την αναγνώριση των επιμέρους συστατικών μερών που συνθέτουν ένα πρόβλημα αλλά επίσης και στον τρόπο που αυτά τα μέρη είναι διασυνδεδεμένα μεταξύ τους. Η δυσκολία αντιμετώπισης ενός προβλήματος μειώνεται δραστικά όσο σταδιακά προχωράμε σε διάσπαση του αρχικού προβλήματος σε μικρότερα κομμάτια. Ο κατακερματισμός ενός προβλήματος σε άλλα μικρότερα προβλήματα είναι μια διαδικασία που ενεργοποιεί το ανθρώπινο μυαλό και υποβοηθά τις αναλυτικές ανθρώπινες ικανότητες. Ένα σύνθετο πρόβλημα εάν διασπαστεί σωστά μπορεί να λυθεί, ευκολότερα και σε λιγότερο χρόνο. Κανένα σύνθετο πρόβλημα δεν επιλύεται εύκολα με μια ολιστική προσπάθεια προσέγγισης. Οι άνθρωποι που μπορούν και λύνουν δύσκολα προβλήματα έχουν μάθει να τα λύνουν σταδιακά, αποδομώντας τα σε μικρότερα κομμάτια («*διαίρει και βασίλευε*»). Οι επιλύσεις των προβλημάτων μπορούν να γίνουν και με διάφορες άλλες παράλληλες τεχνικές επίλυσης όπως, η αφαιρετικότητα, η επίλυση από πάνω προς τα κάτω, η επίλυση από κάτω προς τα επάνω, κ.λπ. Το πρώτο και κυριότερο στοιχείο που μας απασχολεί κατά την προσπάθεια επίλυσης ενός προβλήματος, είναι η καλύτερη αναγνώριση των διαθέσιμων δεδομένων (πληροφορίες). Αυτό σημαίνει, ότι πρέπει να παρατηρήσουμε και να αναγνωρίσουμε κάθε πιθανό στοιχείο το οποίο θα μας προσφέρει και την παραμικρή

πληροφορία. Το δεύτερο κύριο στοιχείο που μας απασχολεί κατά την επίλυση ενός προβλήματος είναι να αντιληφτούμε ποιο είναι το ζητούμενο (το αποτέλεσμα) που καλούμαστε να επιτύχουμε για να θεωρήσουμε ότι λύσαμε το πρόβλημα. Αυτό σημαίνει, ότι θα πρέπει να βεβαιωθούμε ότι μας γίνεται πλήρως αντιληπτό το ποιο είναι πραγματικά το τελικό ζητούμενο αποτέλεσμα. Είναι συνηθισμένη κατάσταση να προτείνονται λύσεις για διαφορετικά προβλήματα από αυτά που πραγματικά προκαλούν το εμπόδιο.

1.3. Κατηγορίες Προβλημάτων

Ο τρόπος με τον οποίο οι άνθρωποι μελετούν και αντιμετωπίζουν τα προβλήματα, οδήγησε στην ανάγκη ταξινόμησης των προβλημάτων σε κατηγορίες.



Τα προβλήματα, ταξινομούνται με βάση: α) την **δυνατότητα επίλυσης** τους, β) την **δομική μορφή επίλυσης** που εφαρμόζεται σε αυτά και γ) του **τρόπου επίλυσης** που επιλέγει.

Με βάση τη **δυνατότητα επίλυσης** τους, τα προβλήματα διαχωρίζονται σε,

- **επιλύσημα προβλήματα,** με την λύση τους να είναι ήδη γνωστή και η επίλυση τους να είναι βεβαία,

- **ανοικτά προβλήματα,** με την λύση τους να είναι άγνωστη αλλά να υπάρχουν πιθανότητες ότι υπάρχει κάποιος τρόπος να λυθούν,
- **άλυτα προβλήματα,** που δεν έχουν καμία δυνατή γνωστή λύση.

Με βάση τη **δομική μορφή επίλυσης** τους, (τα επιλύσημα προβλήματα), διαχωρίζονται σε,

- **δομημένα,** με την επίλυσή τους να γίνεται μέσα από μια γνωστή, προκαθορισμένη και τυποποιημένη διαδικασία,
- **ημι-δομημένα,** με την επίλυσή τους να γίνεται με την επιλογή μιας κατάλληλης λύσης μέσα από μια γνωστή, προκαθορισμένη λίστα με τυποποιημένες και γνωστές μορφές λύσεων,
- **αδόμητα,** με την επίλυσή τους να γίνεται με την επιλογή μιας κατάλληλης λύσης μέσα από μια αδόμητη -μη τυποποιημένη- λίστα μορφών λύσεων.

Βάση του **τρόπου επίλυσης** τους (τα επιλύσημα προβλήματα), διαχωρίζονται σε,

- **επίλυσης με απόφαση,** αφού απλά και μόνο η λύση τους είναι η απόφαση σε μια ερώτηση που απαντάται καταφατικά ή αρνητικά,
- **επίλυσης με υπολογισμό,** αφού η επίλυση τους απαιτεί την εκτέλεση ενός υπολογισμού. σύνθετου ή απλού,
- **επίλυσης με επιλογή του καλύτερου,** αφού η επίλυση τους απαιτεί την διερεύνηση μιας ιδανικής λύσης μέσα από μια κατάλληλη λίστα επιλογών λύσεων.

1.4. Ακούω – Κατανοώ – Επιλύω

Καθώς η αρχική παρατήρηση είναι κυρίαρχο στοιχείο κατά την προσπάθεια επίλυσης ενός προβλήματος, είναι σημαντικό να καταλάβουμε ότι σε κάθε περίπτωση προσπάθειας επίλυσης θα πρέπει να δίνουμε ιδιαίτερη προσοχή στην αρχική μελέτη, την παρατήρηση, το άκουσμα και την ανάγνωση του προβλήματος. Μέσα από την συστηματική μελέτη και τον εντοπισμό των διαθέσιμων στοιχείων που αναγνωρίζουμε, θα οδηγηθούμε στην διάκριση όλων των πολύτιμων δεδομένων εισόδου, με στόχο, να επιτύχουμε την ορθή και πραγματική κατανόηση του προβλήματος.

Σαν δεδομένα εισόδου, χαρακτηρίζονται όλα τα διαθέσιμα στοιχεία τα οποία θα συγκεντρώσουμε ώστε να οδηγηθούμε στην λύση. Δεν θεωρείται πάντα εύκολο να διακρίνουμε ποια είναι τα δεδομένα εισόδου. Πολλές φορές τα δεδομένα εισόδου υποκρύπτονται και δεν είναι προφανή και τότε θα πρέπει να τα ανακαλύψουμε με ενδελεχή παρατήρηση των αποτελεσμάτων που προκαλεί το πρόβλημα. Το ίδιο συμβαίνει και με τα αναζητούμενα αποτελέσματα. Πολλές φορές δεν είναι προφανές, ποιό ή ποιά θα πρέπει να είναι ακριβώς τα αποτελέσματα, έτσι ώστε από αυτά τα αποτελέσματα να θεωρηθεί ότι το πρόβλημα λύθηκε.

Η διαδικασία της συγκέντρωσης των δεδομένων εισόδου αλλά και ο εντοπισμός των ζητούμενων αποτελεσμάτων, απαιτεί συγκέντρωση, σκέψη, επιμονή, εμπειρία και αναλυτικές ικανότητες. Οτιδήποτε μοιάζει, έστω και αμυδρά, να είναι δεδομένο εισόδου, θα πρέπει να θεωρηθεί ότι είναι ένα δεδομένο εισόδου. Οτιδήποτε μοιάζει, έστω και αμυδρά, να είναι ζητούμενο εξόδου, μπορεί να θεωρηθεί ως ζητούμενο εξόδου. Για τον εντοπισμό όλων των δεδομένων εισόδου αλλά και την κατανόηση όλων των αναγκαίων ζητούμενων αποτελεσμάτων, θα πρέπει να θέτονται ερωτήματα προς το πρόβλημα. Για να γίνει αυτό δυνατό, θα μπορούσαμε σαν εκπαιδευόμενοι να φανταστούμε ότι συμμετέχουμε, σαν ηθοποιοί, σε μια «θεατρική παράσταση», στην οποία υποδυόμαστε ταυτόχρονα δύο ρόλους. Ο ένας μας ρόλος είναι ένας υποθετικός εξωτερικός παρατηρητής και ο άλλος μας ρόλος είναι το ίδιο το πρόβλημα (υποδυόμαστε δηλαδή τον μηχανισμό που προκαλεί το πρόβλημα). Η

πλοκή του σεναρίου, είναι απλή και αφορά μια συνεχόμενη εναλλαγή ρόλων. Υποβάλουμε διαδοχικά ερωτήματα προς τον εαυτό μας και συγκεντρώνουμε τις απαντήσεις μας σε αυτά, σαν παρατηρητές. Συνεπώς, κάθε πρόβλημα το οποίο υποβάλλεται στην διαδικασία να αναλυθεί για να επιλυθεί, θα πρέπει υποχρεωτικά να περάσει τα στάδια, ΑΚΟΥΩ – ΚΑΤΑΝΟΩ – ΕΠΙΛΥΩ.

Σύνοψη κεφαλαίου

Στο κεφάλαιο αυτό παρουσιάστηκε η έννοια *πρόβλημα* και παρατέθηκαν παραδείγματα προβλημάτων από την καθημερινότητα. Περιγράφηκε το θέμα της ικανότητας των ανθρώπων να επιλύουν προβλήματα και αναλύθηκε η σημασία, της σε βάθος κατανόησης, κάθε προβλήματος πριν από την προσπάθεια επίλυσης του. Στην συνέχεια, αναφέρθηκε η αναγκαιότητα εντοπισμού όλων των διαθέσιμων δεδομένων εισόδου καθώς και του ζητούμενου αποτελέσματος. Τέλος, ταξινομήθηκαν τα προβλήματα σε κατηγορίες και αναλύθηκαν τα οφέλη της μεθοδολογίας επίλυσης, με τη διάσπαση τους σε μικρότερα κομμάτια.

Ερωτήσεις αξιολόγησης

Ερώτηση-1: Τι σημαίνει ο όρος *πρόβλημα*;

Ερώτηση-2: Τι σημαίνει αποδόμηση προβλημάτων;

Ερώτηση-3: Τι σημαίνει επιλύσημο πρόβλημα;

Ερώτηση-4: Ποιες τεχνικές επίλυσης προβλημάτων γνωρίζετε;

Ερώτηση-5: Πόσες κατηγορίες προβλημάτων γνωρίζετε με βάση την δυνατότητα επίλυσης;

Ερώτηση-6: Πόσες κατηγορίες προβλημάτων γνωρίζετε βάση του τρόπου επίλυσης;

Ερώτηση-7: Τι σημαίνει επίλυση προβλήματος με απόφαση;

Ερώτηση-8: Πότε χαρακτηρίζουμε ένα πρόβλημα ανοικτό;

Ερώτηση-9: Περιγράψτε τι γνωρίζετε για την διαδικασία συγκέντρωσης δεδομένων εισόδου;

Ερώτηση-10: Πως μπορούμε να εντοπίσουμε όλα τα δεδομένα εισόδου σε ένα πρόβλημα;

Απαντήσεις ερωτήσεων αξιολόγησης

Απάντηση-1: Συμβουλευτείτε τις εισαγωγικές παρατηρήσεις του κεφαλαίου.

Απάντηση-2: Συμβουλευτείτε την ενότητα 1.2. του κεφαλαίου.

Απάντηση-3: Συμβουλευτείτε την ενότητα 1.3. του κεφαλαίου.

Απάντηση-4: Συμβουλευτείτε την ενότητα 1.2. του κεφαλαίου.

Απάντηση-5: Συμβουλευτείτε την ενότητα 1.3. του κεφαλαίου.

Απάντηση-6: Συμβουλευτείτε την ενότητα 1.3. του κεφαλαίου.

Απάντηση-7: Συμβουλευτείτε την ενότητα 1.3. του κεφαλαίου.

Απάντηση-8: Συμβουλευτείτε την ενότητα 1.3. του κεφαλαίου.

Απάντηση-9: Συμβουλευτείτε την ενότητα 1.4. του κεφαλαίου.

Απάντηση-10: Συμβουλευτείτε την ενότητα 1.4. του κεφαλαίου.

2. ΜΗΧΑΝΕΣ ΚΑΙ Η ΕΠΙΣΤΗΜΗ ΤΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

Σκοπός και επιμέρους στόχοι

Στο δεύτερο κεφάλαιο οι εκπαιδευόμενοι θα αναγνωρίσουν τη σημασία της επιστήμης της *Πληροφορικής* και της καταλυτικής παρέμβασης της, στην πρόοδο όλων των άλλων επιστημών και των σύγχρονων τεχνολογιών. Επεξηγούνται οι δυνατότητες ανάπτυξης προγραμμάτων σε ηλεκτρονικούς υπολογιστές και περιγράφονται τα χαρακτηριστικά που τους κάνουν ιδανικούς για την επίλυση σύνθετων προβλημάτων. Αναλύονται ορισμένες από τις βασικότερες ιδέες της πληροφορικής, όπως οι ορισμοί των εννοιών δεδομένα, υπολογιστική επεξεργασία, πληροφορίες και ανατροφοδότηση δεδομένων. Επιπλέον, επεξηγείται εισαγωγικά ο όρος, *λογισμικό* και πως μέσα από αυτό τον όρο, είναι δυνατό να διατυπωθούν διαδικασίες σε μορφή προγράμματος ηλεκτρονικού υπολογιστή, που να μπορούν να εκτελεστούν από μια μηχανή και να επιλύσουν υπολογιστικά προβλήματα.

Προσδοκώμενα αποτελέσματα

Από την μελέτη του δεύτερου κεφαλαίου οι εκπαιδευόμενοι θα μπορούν,

- να περιγράψουν την λειτουργία ενός υπολογιστή ως ικανού συστήματος επεξεργασίας,
- να ορίσουν τον όρο, *λογισμικό*,
- να καθορίσουν τα στοιχεία από τα οποία αποτελείται ένα λογισμικό,
- να ορίσουν, τι σημαίνει, πρόγραμμα ηλεκτρονικού υπολογιστή,
- να ορίσουν, τι σημαίνει, προγραμματισμός ηλεκτρονικών υπολογιστών,
- να περιγράψουν τι σημαίνουν οι όροι, *δεδομένα* και *πληροφορία*,
- να διακρίνουν τις διαφορές ανάμεσα στα δεδομένα και τις πληροφορίες,
- να εκφράσουν ένα ολοκληρωμένο παράδειγμα μετασχηματισμού δεδομένων σε πληροφορία,
- να αναγνωρίζουν την σπουδαιότητα των δεδομένων για την επίλυση ενός προβλήματος,
- να αντιλαμβάνονται την αδιάσπαστη σχέση προγραμμάτων υπολογιστών και δεδομένων,

- να αντιλαμβάνονται τα λογισμικά και τα προγράμματα, ως άυλα τεχνικά κατασκευάσματα με τα οποία μπορούν να κατευθύνουν τις μηχανές τους,
- να συνειδητοποιούν τα οφέλη που προκύπτουν από την αξιοποίηση των λογισμικών.

Έννοιες – Λέξεις Κλειδιά

Η επιστήμη της *Πληροφορικής*, Λογισμικό, Πρόγραμμα ηλεκτρονικού υπολογιστή, Προγραμματισμός υπολογιστών, Προγραμματιστές, Δεδομένα, Επεξεργασία δεδομένων, Πληροφορίες, Ανατροφοδότηση, Μετα-δεδομένα.

Εισαγωγικές Παρατηρήσεις

Οι ανθρώπινες δραστηριότητες που κινούνται γύρω από επιλύσεις προβλημάτων, δεν βασίζονται αποκλειστικά και μόνο στις ικανότητες των ίδιων των ανθρώπων. Οι άνθρωποι σήμερα δεν επιλύουν μόνοι τα προβλήματα τους αλλά έχουν αποκτήσει την ικανότητα, να κατασκευάζουν οι ίδιοι, μηχανές που το κάνουν για λογαριασμό τους.

Οι μηχανές, ως επιλυτές προβλημάτων, συνοδεύουν τις ανθρώπινες προσπάθειες επίλυσης προβλημάτων ήδη από την αρχαιότητα. Ο *Υπολογιστής των Αντικυθήρων*, το μοναδικό και ανεκτίμητο αυτό αρχαιολογικό αντικείμενο, η επιτομή της αρχαίας ελληνικής σκέψης, το οποίο φυλάσσεται στο Εθνικό Αρχαιολογικό Μουσείο των Αθηνών (κωδικός μουσειακού καταλόγου, Χ-15087), προέρχεται από την αρχαιότητα (150–100 π.Χ.) και είναι μια μηχανή που κατασκευάστηκε για να λύνει δύσκολα προβλήματα.

Σήμερα, όλες οι επιστήμες αλλά ειδικότερα η επιστήμη της *Πληροφορικής* έχει αναδειχθεί ως η ιδανικότερη «τεχνητή μηχανή» επίλυσης προβλημάτων. Η πληροφορική και η πρόσφατη επανάσταση στην εξέλιξη της βιομηχανίας των υπολογιστών, καταφέρνουν να επιλύουν δύσκολα προβλήματα και να συνεισφέρουν με εντυπωσιακό και καταλυτικό τρόπο στην

εξέλιξη όλων των άλλων επιστημών. Οι υπολογιστές ως μηχανές μπορούν να εκτελούν ταυτόχρονα πολλούς υπολογισμούς με πολύ υψηλή ταχύτητα και επαναλαμβανόμενο τρόπο σε μεγάλους όγκους δεδομένων. Στην πραγματικότητα όμως, η ανθρώπινη σκέψη είναι το ειδικό στοιχείο που αποδίδει «λογική συμπεριφορά» στους ηλεκτρονικούς υπολογιστές και τους καθιστά χρήσιμους βοηθούς, στην επίλυση των προβλημάτων μας.

2.1. Λογισμικό

Η δυνατότητα των μηχανών να διαθέτουν ικανότητα σκέψης ορίζεται στην πληροφορική, με τον όρο, *λογισμικό.* Ο όρος μπορεί να αποδοθεί σε αυτό, που ονομάζουμε σκέψη ή ακόμα καλύτερα, σε αυτό που περιέχει λογική και λογισμό. Το λογισμικό θεωρείται αποτέλεσμα νόησης και αποκτά υπόσταση κατά τη χρήση του από ένα ηλεκτρονικό υπολογιστή με τη πραγματοποίηση υπολογισμών. Ο άνθρωπος αξιοποιεί τον ηλεκτρονικό υπολογιστή μέσω του λογισμικού το οποίο δεν είναι άμεσα ορατό και μόνο τα αποτελέσματα από την χρήση του γίνονται αντιληπτά. Το λογισμικό είναι μια σύνθετη τεχνική άυλη ανθρώπινη κατασκευή και στην θεωρητική του υπόσταση οριοθετείται από 3 στοιχεία: α) τα δεδομένα, β) τα διαγράμματα και γ) τα προγράμματα. Το λογισμικό οι άνθρωποι το δημιουργούν, συγγράφοντας συλλογές από οδηγίες με τις οποίες κατευθύνουν (εκπαιδεύουν) τις μηχανές τους (ηλεκτρονικοί υπολογιστές) να εφαρμόζουν μεθόδους και να επιλύουν προβλήματα.

2.2. Προγραμματισμός Ηλεκτρονικών Υπολογιστών

Στην πληροφορική, μια συγκεκριμένη συλλογή από οδηγίες, ονομάζεται *Πρόγραμμα Ηλεκτρονικού Υπολογιστή.* Η διαδικασία της δημιουργίας συλλογών από οδηγίες για τις μηχανές ονομάζεται, *Προγραμματισμός Ηλεκτρονικών Υπολογιστών,* ενώ οι άνθρωποι οι οποίοι διαθέτουν τις τεχνικές γνώσεις για να συντάσσουν συλλογές από οδηγίες προς τους υπολογιστές ονομάζονται, *Προγραμματιστές Ηλεκτρονικών Υπολογιστών.* Το σύνολο των οδηγιών οι οποίες αναγνωρίζονται από έναν ηλεκτρονικό υπολογιστή ονομάζεται, *Γλώσσα Προγραμματισμού Ηλεκτρονικού Υπολογιστή.*

Η ικανότητα των ανθρώπων να λύνουν προβλήματα, το μέλλον και η πορεία του ανθρώπινου είδους μέσα στον χρόνο, μοιάζουν λαμπρότερα, καθώς υποβοηθούνται και υποστηρίζονται από την πληροφορική, την ικανή αυτή μηχανή.

2.3. Δεδομένα – Επεξεργασία – Πληροφορίες

Σε όλες τις διαδικασίες αντιμετώπισης και επίλυσης προβλημάτων που περιγράψαμε, αλλά και ειδικότερα στις διαδικασίες που κινούνται γύρω από το φάσμα της πληροφορικής, υπάρχουν κάποιοι σημαντικοί όροι οι οποίοι χρησιμοποιούνται, πρέπει να αναλυθούν και να γίνουν κατανοητοί.

2.4. Δεδομένα (Data)

Δεδομένα, ονομάζουμε κάθε στοιχείο (κάθε παρατήρηση, κάθε μέτρηση, κάθε ποσότητα, κάθε θερμοκρασία, κάθε ήχο, κ.λπ.) το οποίο θα συλλέξουμε και θα χρησιμοποιούμε στην συνέχεια για επεξεργασία ώστε να οδηγηθούμε σε κάποιο αποτέλεσμα και σε κάποια νέα γνώση.

2.5. Επεξεργασία Δεδομένων (Data Processing)

Επεξεργασία δεδομένων είναι κάθε μηχανισμός, ο οποίος δέχεται δεδομένα, τα επεξεργάζεται και τα μετασχηματίζει σε κάποιο αποτέλεσμα. Η επεξεργασία των δεδομένων έχει ποικίλες μορφές και δεν είναι πάντοτε απαραίτητη η εφαρμογή αριθμητικών πράξεων για να λάβουμε αποτέλεσμα. Ειδικά για την πληροφορική, την επεξεργασία των δεδομένων την επιτυγχάνουμε μέσα από την εφαρμογή, επάνω σε κάποια δεδομένα, ενός κατάλληλα σχεδιασμένου, προγράμματος ηλεκτρονικού υπολογιστή.

2.6. Πληροφορία (Information)

Πληροφορία, είναι το αποτέλεσμα που παίρνουμε από την επεξεργασία των δεδομένων και την παραγωγή επιπρόσθετης γνώσης. Οποιοδήποτε νέο γνωσιακό στοιχείο, (γνώση που παράγεται) προέρχεται από επεξεργασία δεδομένων.

Για παράδειγμα, εάν σκεφτούμε ότι έχουμε στην διάθεση μας 24 μετρήσεις θερμοκρασίας, τις οποίες καταγράψαμε ανά ώρα, μέσα σε μια συγκεκριμένη ημέρα και από αυτές τις διαθέσιμες θερμοκρασίες θέλουμε να υπολογίσουμε την μέση θερμοκρασία της ημέρας, ώστε να αποφασίσουμε από τον μέσο όρο, εάν η ημέρα μας ήταν ζεστή ή ψυχρή, τότε θα πρέπει να μπορούμε να διακρίνουμε ότι,

α) οι 24 θερμοκρασίες (16, 20, 22, 24, 25, 23, 20, 18, 16, 13, 12, 14, 10, 9, 8, 9, 11, 12, 10, 11, 9, 11, 12, 14) είναι τα δεδομένα,

β) η πράξη, να αθροίσουμε τις 24 θερμοκρασίες και να διαιρέσουμε το άθροισμα με το 24, είναι ο κατάλληλα σχεδιασμένος αλγόριθμος για την επεξεργασία των δεδομένων,

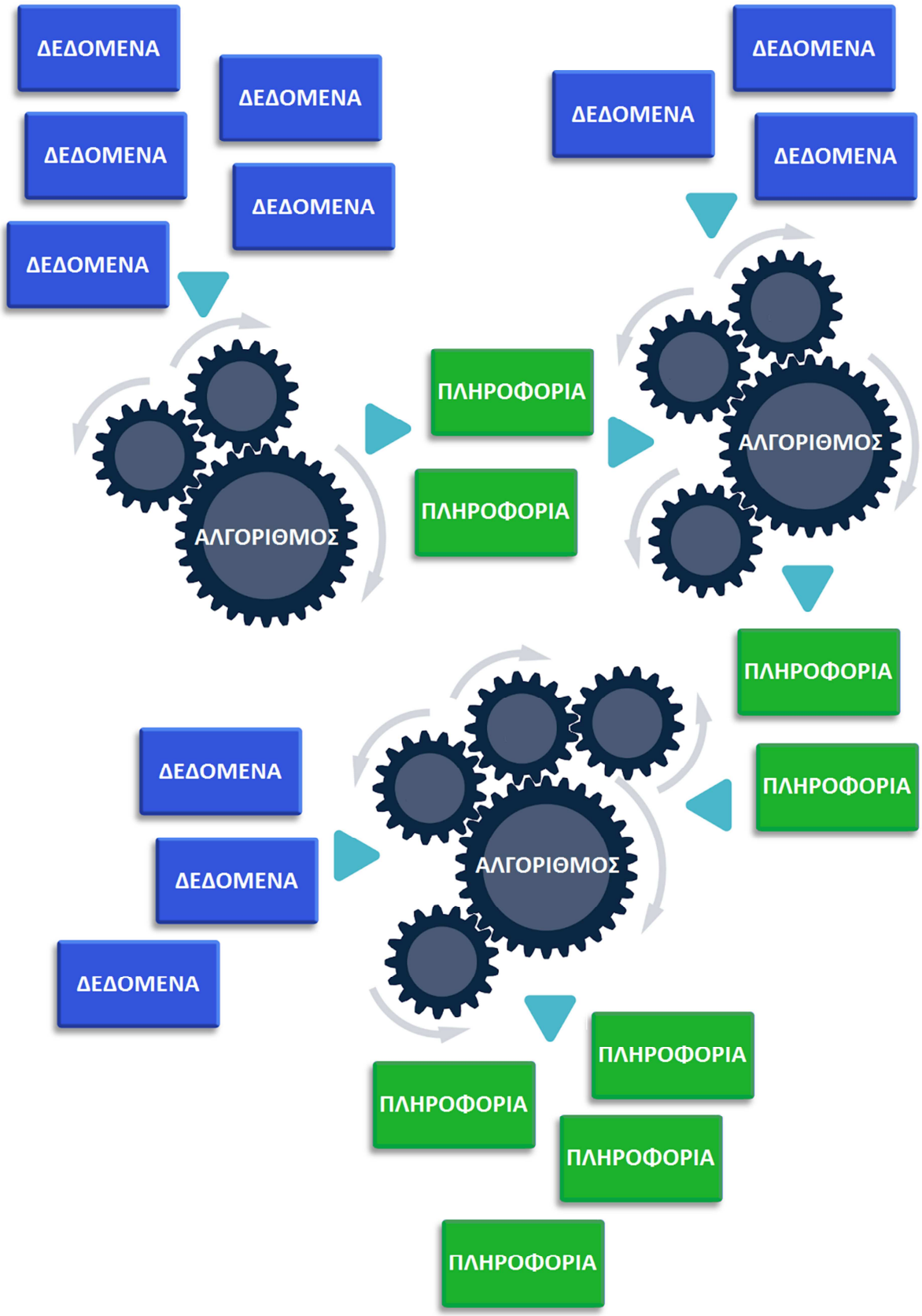
γ) το αποτέλεσμα (349/24=14) είναι η μέση θερμοκρασία (ο μέσος όρος),

δ) η μέση θερμοκρασία που υπολογίσαμε, είναι η πληροφορία ότι, η ημέρα ήταν ψυχρή.

2.7. Ανατροφοδότηση (Feedback)

Ανατροφοδότηση, ονομάζουμε την διαδικασία κατά την οποία κάποιες πληροφορίες που έχουν προκύψει ως αποτέλεσμα μιας πρώτης επεξεργασίας επαναχρησιμοποιούνται (οι ίδιες) μαζί ή και σε συνδυασμό με άλλες πρόσθετες πληροφορίες ή δεδομένα ώστε να παράγουν επιπλέον νεότερες πληροφορίες. Η διαδικασία αυτή, στην οποία οι πληροφορίες αποτελούν για την συνέχεια, δεδομένα σε μια άλλη νέα επεξεργασία, χαρακτηρίζεται στην πληροφορική, ως κύκλος επεξεργασίας των δεδομένων.

Για παράδειγμα, εάν σκεφτούμε (και σε συνέχεια του παραδείγματος με τις θερμοκρασίες), ότι έχουμε συγκεντρώσει για τις 30 τελευταίες ημέρες, τις ημέρες που ήταν ζεστές και τις ημέρες που ήταν ψυχρές, δεν θα μπορούσαμε να ανατροφοδοτήσουμε αυτές τις 30 μέσες θερμοκρασίες σε έναν νέο μηχανισμό υπολογισμού, ο οποίος θα έκανε την πράξη (άθροισμα των 30 μέσων θερμοκρασιών / 30) και συνεπώς θα μπορούσε να υπολογίσει εκ νέου εάν συνολικά ο μήνας μας ήταν ψυχρός ή ζεστός. Αυτή η διαδικασία στην πληροφορική χαρακτηρίζεται ως κύκλος επεξεργασίας των δεδομένων ή ανατροφοδότηση πληροφοριών.


ΔΕΔΟΜΕΝΑ
ΔΕΔΟΜΕΝΑ
ΔΕΔΟΜΕΝΑ
ΔΕΔΟΜΕΝΑ
ΔΕΔΟΜΕΝΑ
ΔΕΔΟΜΕΝΑ
ΔΕΔΟΜΕΝΑ
ΔΕΔΟΜΕΝΑ
ΔΕΔΟΜΕΝΑ
ΑΛΓΟΡΙΘΜΟΣ
ΠΛΗΡΟΦΟΡΙΑ
ΠΛΗΡΟΦΟΡΙΑ
ΑΛΓΟΡΙΘΜΟΣ
ΠΛΗΡΟΦΟΡΙΑ
ΠΛΗΡΟΦΟΡΙΑ
ΑΛΓΟΡΙΘΜΟΣ
ΠΛΗΡΟΦΟΡΙΑ
ΠΛΗΡΟΦΟΡΙΑ
ΠΛΗΡΟΦΟΡΙΑ
ΠΛΗΡΟΦΟΡΙΑ
ΠΛΗΡΟΦΟΡΙΑ

2.8. Επεξεργασία Δεδομένων με Υπολογιστές

Στην βάση των συλλογισμών επεξεργασίας των δεδομένων με τις θερμοκρασίες, θα μπορούσαμε να συνεχίζουμε να υπολογίζουμε τις μέσες θερμοκρασίες για κάθε μήνα, με στόχο να βρούμε την μέση θερμοκρασία ολόκληρου του έτους ή ακόμα καλύτερα να καταλήξουμε στην μέση θερμοκρασία ολόκληρης της δεκαετίας. Όμως αυτή η επαναλαμβανόμενη διαδικασία, γίνεται εύκολα αντιληπτό ότι δεν είναι λειτουργική, καθώς απαιτεί πολύ χρόνο και κόπο για να υλοποιηθεί. Συνεπώς, θα ήταν επιθυμητό να διαθέταμε κάποιον μηχανισμό ο οποίος θα μπορούσε να εκτελεί αυτόματα όλους αυτούς τους υπολογισμούς με γρήγορο και αποτελεσματικό τρόπο. Αυτό ακριβώς, κάνουν οι ηλεκτρονικοί υπολογιστές και τα προγράμματα των ηλεκτρονικών υπολογιστών. Χρησιμοποιούνται για να εκτελούν σε σύντομο χρονικό διάστημα επαναλαμβανόμενους υπολογισμούς επάνω σε μεγάλους όγκους δεδομένων, πολλά από τα οποία δεδομένα μετασχηματίζονται, από κάποια ενδιάμεση επεξεργασία σε μετα-δεδομένα, που και αυτά με την σειρά τους μετασχηματίζονται, από κάποια άλλη επεξεργασία σε μετα-μετα-δεδομένα.

Ένα πρόγραμμα ηλεκτρονικού υπολογιστή είναι ένα σύνολο από εντολές, σε μια μορφή που γίνεται κατανοητή από τον υπολογιστή και περιγράφει βήμα προς βήμα τις ενέργειες που πρέπει να εκτελέσει η μηχανή, ώστε να επιλύσει ένα πρόβλημα. Κάθε υπολογιστής έχει ένα συγκεκριμένο ρεπερτόριο από εντολές τις οποίες αναγνωρίζει και με τις οποίες μπορεί να καθοδηγηθεί από έναν προγραμματιστή, για το τι πρέπει να κάνει και πως να το κάνει. Ένας προγραμματιστής ηλεκτρονικών υπολογιστών γνωρίζει αυτές τις εντολές και άρα διαθέτει την ικανότητα να προγραμματίσει έναν ηλεκτρονικό υπολογιστή. Ένα πρόγραμμα, το οποίο γράφεται σαν ένα σύνολο από εντολές, δεν είναι απλά και μόνο η υλοποίηση μιας μεθοδολογίας επίλυσης, αλλά είναι κυρίως η περιγραφή ενός συγκεκριμένου τρόπου χειρισμού δεδομένων. Αυτή η διαδικασία του προγραμματισμού των υπολογιστών είναι που δίνει την εντύπωση ότι οι υπολογιστές είναι έξυπνες μηχανές που μπορούν και επιλύουν αυτόματα πολύπλοκα προβλήματα. Η εντύπωση όμως αυτή είναι εικονική και αυτό που συμβαίνει στην πραγματικότητα είναι η καταγραφή της μεθοδολογίας επίλυσης, σε έναν αλγόριθμο, που έχει περιγραφεί από ένα ανθρώπινο μυαλό.

Σύνοψη κεφαλαίου

Στο δεύτερο κεφάλαιο περιγράφεται η σημασία της συμβολής των μηχανών και των σύγχρονων τεχνολογιών της πληροφορικής στην επίλυση σύνθετων υπολογιστικών προβλημάτων. Επεξηγείται, η έννοια *λογισμικό* και μεταφέρονται γνωσιακά στον αναγνώστη οι όροι, *προγραμματισμός ηλεκτρονικού υπολογιστή*, *προγράμματα υπολογιστών*, *δεδομένα*, *επεξεργασία δεδομένων με υπολογιστές*, *ανατροφοδότηση* και *πληροφορίες*. Επιπλέον, χρησιμοποιείται κατάλληλο παράδειγμα ώστε να υποβοηθηθεί η ικανότητα των εκπαιδευομένων να αντιλαμβάνονται την διαφορά των δεδομένων από την πληροφορία.

Ερωτήσεις αξιολόγησης

Ερώτηση-1: Περιγράψτε τι είναι το λογισμικό;

Ερώτηση-2: Από πόσα και ποια στοιχεία αποτελείται ένα λογισμικό;

Ερώτηση-3: Είναι σωστό, ότι το λογισμικό μπορεί να είναι ορατό;

Ερώτηση-4: Η διαδικασία της δημιουργίας συλλογών από οδηγίες για τον έλεγχο των μηχανών, πως ονομάζεται;

Ερώτηση-5: Τι είναι τα δεδομένα, δώστε ένα παράδειγμα;

Ερώτηση-6: Τι είναι οι πληροφορίες, δώστε ένα παράδειγμα;

Ερώτηση-7: Περιγράψτε, τι είναι η επεξεργασία δεδομένων;

Ερώτηση-8: Το σημαίνει ανατροφοδότηση;

Ερώτηση-9: Για ποιους λόγους οι υπολογιστές θεωρούνται ιδανικές μηχανές για την επεξεργασία πολλών δεδομένων;

Ερώτηση-10: Τι είναι τα μετα-δεδομένα, δώστε ένα παράδειγμα;

Απαντήσεις ερωτήσεων αξιολόγησης

Απάντηση-1: Συμβουλευτείτε την ενότητα 2.1. του κεφαλαίου.

Απάντηση-2: Συμβουλευτείτε την ενότητα 2.1. του κεφαλαίου.

Απάντηση-3: Όχι, είναι λάθος.

Απάντηση-4: Συμβουλευτείτε την ενότητα 2.2. του κεφαλαίου.

Απάντηση-5: Συμβουλευτείτε την ενότητα 2.4. του κεφαλαίου.

Απάντηση-6: Συμβουλευτείτε την ενότητα 2.6. του κεφαλαίου.

Απάντηση-7: Συμβουλευτείτε την ενότητα 2.5. του κεφαλαίου.

Απάντηση-8: Συμβουλευτείτε την ενότητα 2.7. του κεφαλαίου.

Απάντηση-9: Συμβουλευτείτε την ενότητα 2.8. του κεφαλαίου.

Απάντηση-10: Συμβουλευτείτε την ενότητα 2.8. του κεφαλαίου.

3. ΑΛΓΟΡΙΘΜΟΣ

Σκοπός και επιμέρους στόχοι

Το τρίτο κεφάλαιο του συγγράμματος περιγράφει την έννοια των αλγορίθμων, τις ιδιότητες που τους χαρακτηρίζουν και τα απαραίτητα συστατικά τους στοιχεία. Επεξηγείται ο λόγος που οι αλγόριθμοι αποτελούν για την πληροφορική, την πρώτη ύλη για την είσοδο στο αντικείμενο της γνώσης των γλωσσών προγραμματισμού. Επιπλέον, περιγράφονται τρόποι με τους οποίους αξιολογούνται οι αλγόριθμοι, για την αποτελεσματικότητα τους και καταγράφονται οι βασικές κατηγορίες στις οποίες κατατάσσονται.

Προσδοκώμενα αποτελέσματα

Με την μελέτη του τρίτου κεφαλαίου οι εκπαιδευόμενοι θα μπορούν,

- να περιγράφουν, τον όρο, *αλγόριθμος*,
- να αντιλαμβάνονται την σημασία των αλγορίθμων για την πληροφορική,
- να εντοπίζουν τα συστατικά στοιχεία τα οποία οριοθετούν την ύπαρξη ενός αλγορίθμου,
- να αντιλαμβάνονται την ανάγκη της αλγοριθμικής προσέγγισης των προβλημάτων,
- να αξιολογήσουν έναν αλγόριθμό για την αποτελεσματικότητα και την ικανότητα του,
- να καταλαβαίνουν πως οι αλγόριθμοι έχουν την ικανότητα να επενεργούν επάνω σε δεδομένα,
- να περιγράφουν τους βασικούς τύπους των αλγορίθμων.

Έννοιες – Λέξεις Κλειδιά

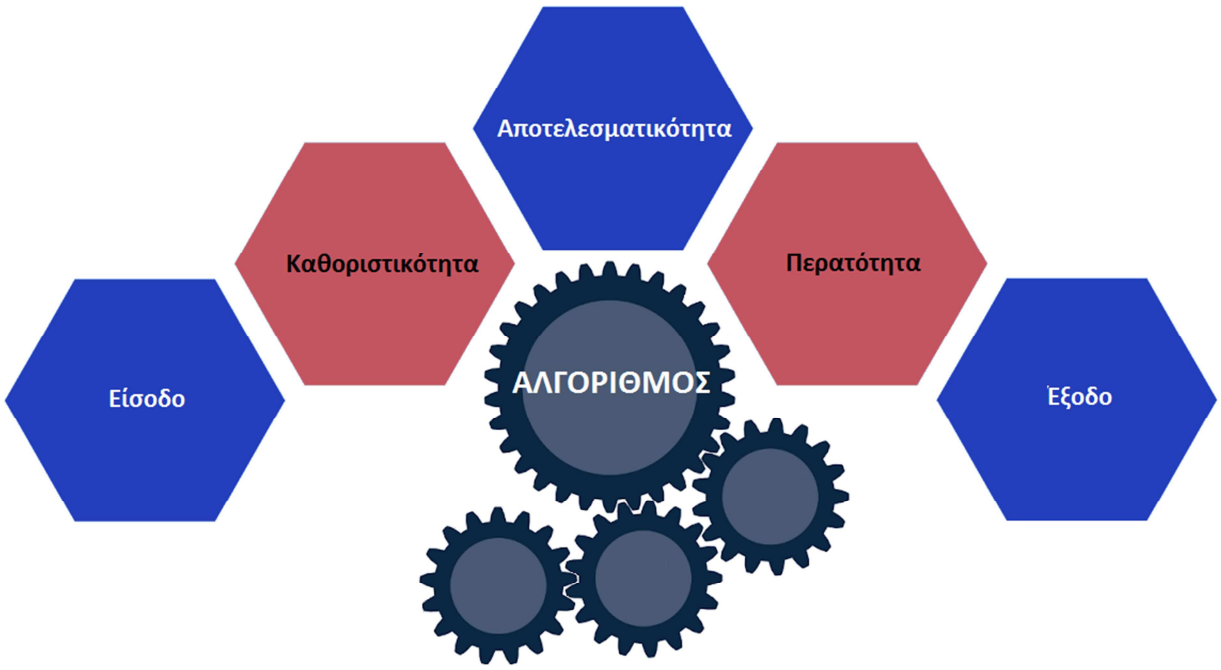
Ακολουθία ενεργειών, Είσοδος αλγορίθμου, Καθοριστικότητα, Αποτελεσματικότητα, Περατότητα, Έξοδος αλγορίθμου, Αξιολόγηση αλγορίθμων, Αποδοτικότητα αλγορίθμου, Χρονική πολυπλοκότητα, Χωρική πολυπλοκότητα, Βέλτιστοι αλγόριθμοι, Προσεγγιστικοί αλγόριθμοι.

Εισαγωγικές Παρατηρήσεις

Η επίλυση ενός προβλήματος με ένα ηλεκτρονικό υπολογιστή, όπως έχει ήδη αναφερθεί, περιλαμβάνει την ανάπτυξη και την διατύπωση ενός κατάλληλου αλγορίθμου. Ο όρος *αλγόριθμος* θα μπορούσε να σημαίνει την τυποποιημένη, συστηματική διαδικασία εκτέλεσης λογικών και αριθμητικών πράξεων. Φορμαλιστικά, ο ορισμός αλγόριθμος, είναι μια ακολουθία συγκεκριμένων ενεργειών, αυστηρά προκαθορισμένων, εκτελέσιμων σε ικανοποιητικό χρονικό διάστημα, οι οποίες αν εφαρμοστούν επιτυγχάνουν το επιθυμητό αποτέλεσμα. Οι αλγόριθμοι θεωρούνται η βάση της επιστήμης της *Πληροφορικής*. Πολλές άλλες επιστήμες βασίζονται επίσης στους αλγορίθμους. Στην επιστήμη της *Πληροφορικής*, ο όρος *αλγόριθμος* (algorithm) χρησιμοποιείται, για να περιγράψει μια πεπερασμένη (finite), αιτιοκρατική (deterministic) και αποτελεσματική (affective) μέθοδο επίλυσης ενός προβλήματος, κατάλληλη για την υλοποίηση σε ένα πρόγραμμα ηλεκτρονικού υπολογιστή.

3.1. Χαρακτηριστικά Στοιχεία Αλγορίθμων

Ένας αλγόριθμος σε ένα πρόγραμμα ηλεκτρονικού υπολογιστή συνήθως έχει,



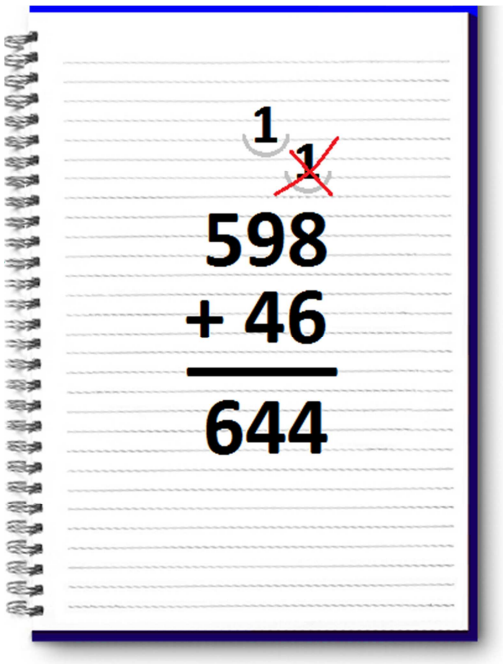
- **είσοδο** (input): με ένα ή πολλά δεδομένα ή πληροφορίες εισόδου,
- **καθοριστικότητα** (definiteness): οι ενέργειες που τον αποτελούν καθορίζονται ρητά (επακριβώς),
- **αποτελεσματικότητα** (effectiveness): οι εντολές που τον αποτελούν λειτουργούν σωστά,
- **περατότητα** (finiteness): τελειώνει μετά από συγκεκριμένα βήματα,
- **έξοδο** (output): μπορεί να παράγει ένα ή πολλά αποτελέσματα.

3.2. Αξιολόγηση Αλγορίθμων

Ένα πρόβλημα, το οποίο τίθεται για να επιλυθεί, δεν σημαίνει ότι μπορεί να επιλυθεί από έναν και μόνο αλγόριθμο. Για ένα πρόβλημα είναι δυνατό να προταθούν πολλές και διαφορετικές λύσεις. Συνεπώς, ένας αλγόριθμος ο οποίος λύνει ένα συγκεκριμένο πρόβλημα μπορεί να είναι «καλύτερος» ή «χειρότερος» από έναν άλλο αλγόριθμο, ο οποίος λύνει με έναν διαφορετικό τρόπο το ίδιο ακριβώς πρόβλημα. Δεδομένου λοιπόν, ότι ένα πρόβλημα μπορεί να επιλύεται από πολλούς και διαφορετικούς αλγορίθμους είναι σημαντικό να μπορούμε να αξιολογούμε την αποτελεσματικότητα του κάθε μηχανισμού επίλυσης και να αποφασίζουμε ποια από τις προτεινόμενες λύσεις θα θεωρήσουμε ως την καλύτερη.

Τα βασικά κριτήρια αξιολόγησης των αλγορίθμων είναι: α) ο αναγκαίος χρόνος εκτέλεσης και β) τα αναγκαία αποθηκευτικά μέσα. Σαν αναγκαίο χρόνο εκτέλεσης, εννοούμε τον χρόνο που χρειάζεται ο μηχανισμός για να επιτελέσει το έργο του και αναγκαία αποθηκευτικά μέσα εννοούμε τα μέσα μνήμης που χρειάζεται ο μηχανισμός για να ολοκληρώσει το έργο του. Για να γίνει ακόμα καλύτερα κατανοητή, η έκφραση «αναγκαία αποθηκευτικά μέσα» ή η έκφραση «μέσα μνήμης», μπορούμε να φανταστούμε τις ενέργειες που κάνουμε με την πράξη της πρόσθεσης, χρησιμοποιώντας από τα παιδικά μας χρόνια ένα «καπελάκι» στο οποίο αποθηκεύουμε (κρατάμε στην μνήμη μας για να μην τα χάσουμε) προσωρινά τα κρατούμενα ώστε να συνεχίσουμε στα επόμενα ψηφιά. Ακριβώς το ίδιο χρειάζεται και ο ηλεκτρονικός υπολογιστής για να επιτελέσει έναν αλγόριθμο, προσωρινές θέσεις μνήμης, οι οποίες όμως είναι περιορισμένες σε έναν υπολογιστή και άρα μας ενδιαφέρει ένας αλγόριθμος που χρειάζεται τον ελάχιστο δυνατό χώρο μνήμης, όπως ακριβώς και στην πρόσθεση με το χέρι.

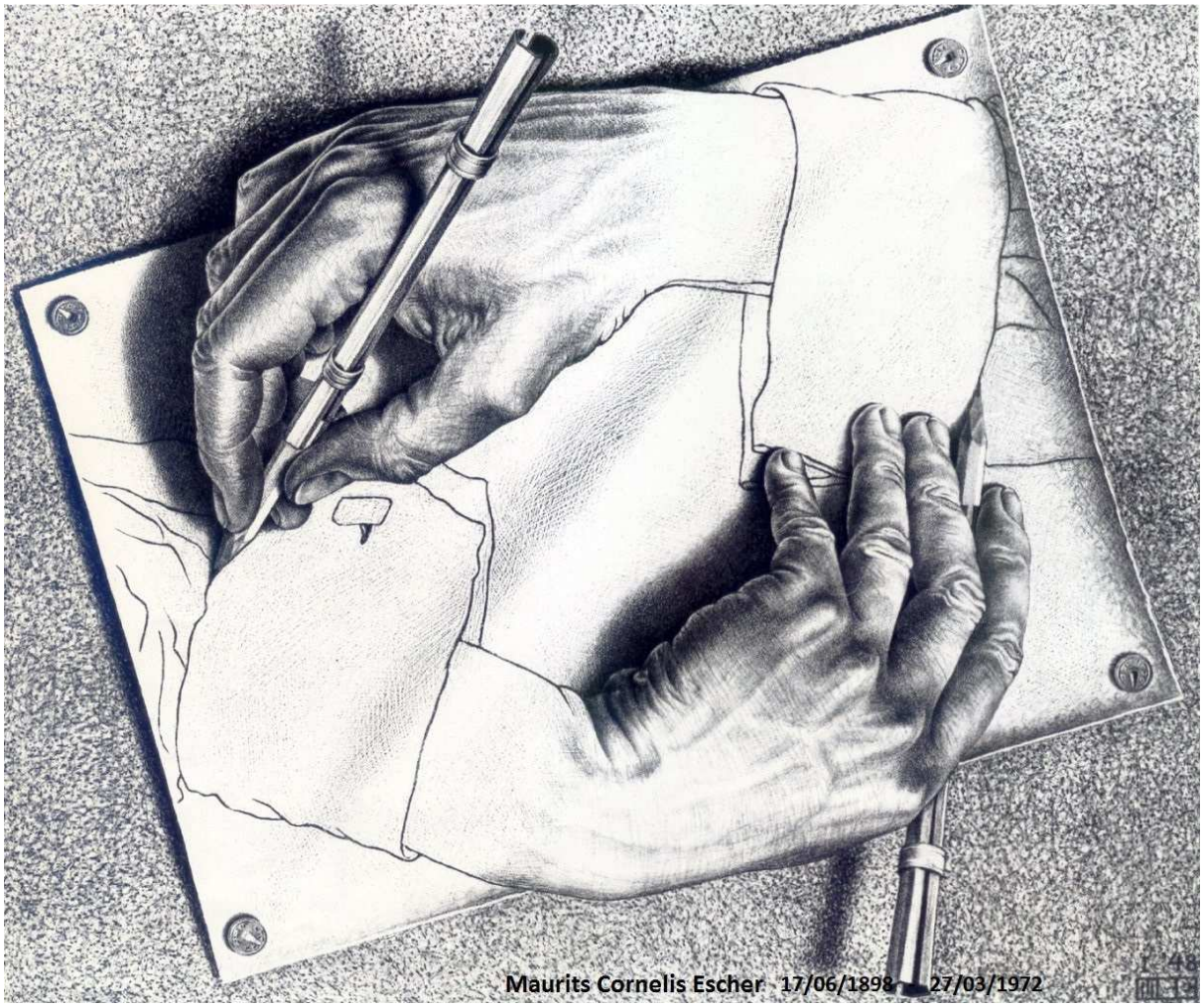
Συνεπώς, για ποιο λόγο να κρατήσουμε κάπου στο χαρτί μας, περισσότερα ψηφία αφού στην πραγματικότητα χρειαζόμαστε μόνο ένα.



3.3. Ο Καλύτερος Αλγόριθμος

Προκειμένου, να εκτιμήσουμε έναν αλγόριθμο, πρέπει να τον αξιολογήσουμε με βάση την επίδοση και την αποδοτικότητα του (performance & efficiency). Καλύτερος αλγόριθμος είναι αυτός που έχει την καλύτερη χρονική πολυπλοκότητα (time complexity) και την καλύτερη χωρική πολυπλοκότητα (space complexity), αυτός δηλαδή που εκτελείται στον λιγότερο δυνατό χρόνο και στον μικρότερο δυνατό χώρο. Επιπλέον όμως ένας αλγόριθμος πρέπει να είναι και εύκολος στην κατασκευή του. Ποια είναι η αξία ενός αλγορίθμου εάν τον καταλαβαίνει μόνο αυτός που τον έφτιαξε. Ένας αλγόριθμος όπως και κάθε άλλο πνευματικό έργο, για να έχει αξία, πρέπει να κατασκευάζεται με τρόπο ώστε να το καταβαίνουν πολλοί. Στην πληροφορική ένας αλγόριθμος θεωρείται ο καλύτερος, μέχρι να βρεθεί κάποιος άλλος καλύτερος ο οποίος λύνει το ίδιο πρόβλημα σε λιγότερο χρόνο και σε μικρότερο χώρο. Στους κύκλους πολλών επιστημών, οι σημαντικοί αλγόριθμοι «επικηρύσσονται» με υψηλά χρηματικά ποσά, ώστε οι επιστήμονες και τα επιστημονικά εργαστήρια να δελεάζονται με οικονομική ενίσχυση και να αναζητούν συνεχώς καλύτερες επιλύσεις από τις γνωστές υπάρχουσες.

Η απόδειξη στην πληροφορική ότι, ένας αλγόριθμος θα λειτουργεί για πάντα σωστά, δεν πρέπει να θεωρείται αυταπόδεικτο. Για να γίνει δεκτό ότι ένας αλγόριθμος θα λειτουργεί για πάντα σωστά, θα πρέπει ο αλγόριθμος να λειτουργήσει για όλες τις πιθανές καταστάσεις στις οποίες μπορεί να βρεθεί. Ειδικά όμως για τους μεγάλους και πολύπλοκους αλγορίθμους, αυτό είναι πολύ δύσκολο και συνεπώς δεν μπορεί να υπάρξει απόλυτη ασφάλεια στην εκτίμηση για πιθανά κενά.



3.4. Τύποι Αλγορίθμων

Η ανάπτυξη των αλγορίθμων απασχολεί πλήθος επιστημόνων και για το λόγο αυτό, έχουν αναπτυχθεί πολλά είδη και κατηγορίες αλγορίθμων. Οι αλγόριθμοι ταξινομούνται με βάση τις επιδόσεις τους. Ένας αλγόριθμος ονομάζεται *βέλτιστος* (optimal) αν δεν μπορεί να αποδειχθεί ότι μπορεί να κατασκευασθεί κάποιος καλύτερος. Αλγόριθμοι που ονομάζονται *προσεγγιστικοί* (approximate) εφαρμόζονται σε σύνθετα προβλήματα και απλά επιτυγχάνουν μια αποδεκτή λύση σε λογικό χρόνο. Οι προσεγγιστικοί αλγόριθμοι, προσεγγίζουν και επιλύουν δύσκολα πρόβλημα με μια απάντηση που πλησιάζει την επιθυμητή λύση χωρίς όμως να είναι η καλύτερη δυνατή λύση. Καθώς η βελτιστοποίηση ενός αλγορίθμου είναι σημαντική, διερευνούνται συνεχώς οι δυνατότητες βελτίωσης τους. Νέοι αλγόριθμοι αναπτύσσονται και άλλοι παλαιότεροι αποσύρονται. Η πληροφορική γενικότερα, διερευνά και αναλύει την πολυπλοκότητα των αλγορίθμων και τους ταξινομεί ανάλογα με την απόδοση τους. Υπάρχουν, οι *πολυωνυμικοί* (polynomial) αλγόριθμοι, οι *εκθετικοί* (exponential) αλγόριθμοι, αλλά και οι *ευριστικοί* (heuristic), αλγόριθμοι. Αυτοί οι τελευταίοι τύποι των αλγορίθμων προσεγγίζουν την επίλυση των προβλημάτων στην βάση μαθηματικών συλλογισμών και για το λόγο αυτό έχουν ονόματα μαθηματικών όρων.

Σύνοψη κεφαλαίου

Το τρίτο κεφάλαιο προσεγγίζει την έννοια *αλγόριθμος*, επικεντρώνοντας στην μεταφορά της αντίληψης, ότι αλγόριθμος είναι η περιγραφή μιας ικανής και συγκεκριμένης διαδικασίας επίλυσης ενός προβλήματος. Επιπλέον, οριοθετούνται τα χαρακτηριστικά συστατικά στοιχεία των αλγορίθμων στον κόσμο της πληροφορικής, ταξινομούνται οι αλγόριθμοι σε κατηγορίες και περιγράφεται η μεθοδολογία με την οποία αξιολογούνται ως προς την αποδοτικότητα τους.

Ερωτήσεις αξιολόγησης

Ερώτηση-1: Τι σημαίνει αλγόριθμος;

Ερώτηση-2: Είναι σωστό, ότι ένα πρόβλημα μπορεί να λυθεί μόνο από έναν αλγόριθμο;

Ερώτηση-3: Πόσα και ποια είναι τα χαρακτηριστικά στοιχεία ενός αλγορίθμου;

Ερώτηση-4: Τι σημαίνει είσοδος σε έναν αλγόριθμο;

Ερώτηση-5: Τι σημαίνει έξοδος σε έναν αλγόριθμο;

Ερώτηση-6: Τι σημαίνει περατότητα ενός αλγορίθμου;

Ερώτηση-7: Για ποιον λόγο χρειαζόμαστε χώρους μνήμης για έναν αλγόριθμο;

Ερώτηση-8: Ποια είναι τα βασικά κριτήρια με τα οποία μπορούμε να αξιολογήσουμε την αποτελεσματικότητα ενός αλγορίθμου;

Ερώτηση-9: Ένας αλγόριθμος είναι δεδομένο ότι θα λειτουργεί για πάντα σωστά;

Ερώτηση-10: Δώστε 2 διαφορετικούς τύπους αλγορίθμων;

Απαντήσεις ερωτήσεων αξιολόγησης

Απάντηση-1: Συμβουλευτείτε τις εισαγωγικές παρατηρήσεις του κεφαλαίου.

Απάντηση-2: Όχι, είναι λάθος.

Απάντηση-3: Συμβουλευτείτε την ενότητα 3.1. του κεφαλαίου.

Απάντηση-4: Συμβουλευτείτε την ενότητα 3.1. του κεφαλαίου.

Απάντηση-5: Συμβουλευτείτε την ενότητα 3.1. του κεφαλαίου.

Απάντηση-6: Συμβουλευτείτε την ενότητα 3.1. του κεφαλαίου.

Απάντηση-7: Συμβουλευτείτε την ενότητα 3.2. του κεφαλαίου.

Απάντηση-8: Συμβουλευτείτε την ενότητα 3.2. του κεφαλαίου.

Απάντηση-9: Συμβουλευτείτε την ενότητα 3.3. του κεφαλαίου.

Απάντηση-10: Συμβουλευτείτε την ενότητα 3.4. του κεφαλαίου.

4. ΜΟΡΦΕΣ ΑΛΓΟΡΙΘΜΩΝ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ

Σκοπός και επιμέρους στόχοι

Στο τέταρτο κεφάλαιο μεταφέρονται στον εκπαιδευόμενο οι εικόνες των μορφών αναπαράστασης αλγορίθμων, που χρησιμοποιούνται στον προγραμματισμό των ηλεκτρονικών υπολογιστών. Συγκεκριμένα, παρουσιάζεται η αναπαράσταση σε ελεύθερο κείμενο, η αναπαράσταση με λογικά διαγράμματα ροής και η κωδικοποίηση με κανόνες ψευδογλώσσας. Ειδικά για τα λογικά διαγράμματα, το κεφάλαιο περιγράφει τα πρώτα απαραίτητα γεωμετρικά σύμβολα και επεξηγεί με ολοκληρωμένο παράδειγμα την μεθοδολογία ανάπτυξης και χρήσης αυτών των συμβολισμών. Ακολούθως, το κεφάλαιο αφιερώνεται στην παρουσίαση μιας ψευδογλώσσας, με συγκεκριμένους κανόνες ψευδοκώδικα και στην βάση αυτών των κανόνων καταστρώνονται 3 διαφορετικοί αλγόριθμοι, η δομή μιας απλής ακολουθίας, η δομή επιλογής και η δομή επανάληψης. Στο τέλος του κεφαλαίου παρουσιάζεται η αλγοριθμική προσέγγιση για την επίλυση ενός σύνθετου προβλήματος ενώ καταγράφονται και οι προγραμματιστικοί κανόνες για την συγγραφή δομημένων προγραμμάτων με τάξη και κανόνες.

Προσδοκώμενα αποτελέσματα

Με την μελέτη του κεφαλαίου οι εκπαιδευόμενοι θα μπορούν,

- να αναλύσουν μορφές σχεδίασης αλγορίθμων στον προγραμματισμό,
- να περιγράφουν τις βασικότερες προσεγγίσεις σχεδίασης αλγορίθμων,
- να εφαρμόζουν συγκεκριμένα τεχνικά βήματα για να καταστρώνουν μια αλγοριθμική διαδικασία επίλυσης,
- να αναγνωρίζουν πως ένας αλγόριθμος μπορεί να επηρεαστεί από εξωγενείς παράγοντες,
- να αντιλαμβάνονται πως ένας αλγόριθμος είναι δυνατό να υλοποιηθεί σε διαφορετικά επίπεδα λεπτομέρειας,
- να τεκμηριώνουν την αναγκαιότητα διάσπασης των προβλημάτων που αντιμετωπίζουν σε κομμάτια και να σχεδιάζουν τους κατάλληλους αλγόριθμους για αυτά,

- να καταγράφουν την απαραίτητη ακολουθία βημάτων για την ανάλυση ενός αλγορίθμου,
- να αναλύουν ένα βασικό διάγραμμα ροής και να κατανοούν την λειτουργία του,
- να αναλύουν έναν βασικό ψευδοκώδικα και να κατανοούν την λειτουργικότητα του,
- να αντιπαραβάλλουν αλγόριθμους με βάση τυποποιημένους κανόνες και να συγκρίνουν την ικανότητα τους,
- να ελέγχουν την ορθότητα των αλγορίθμων.

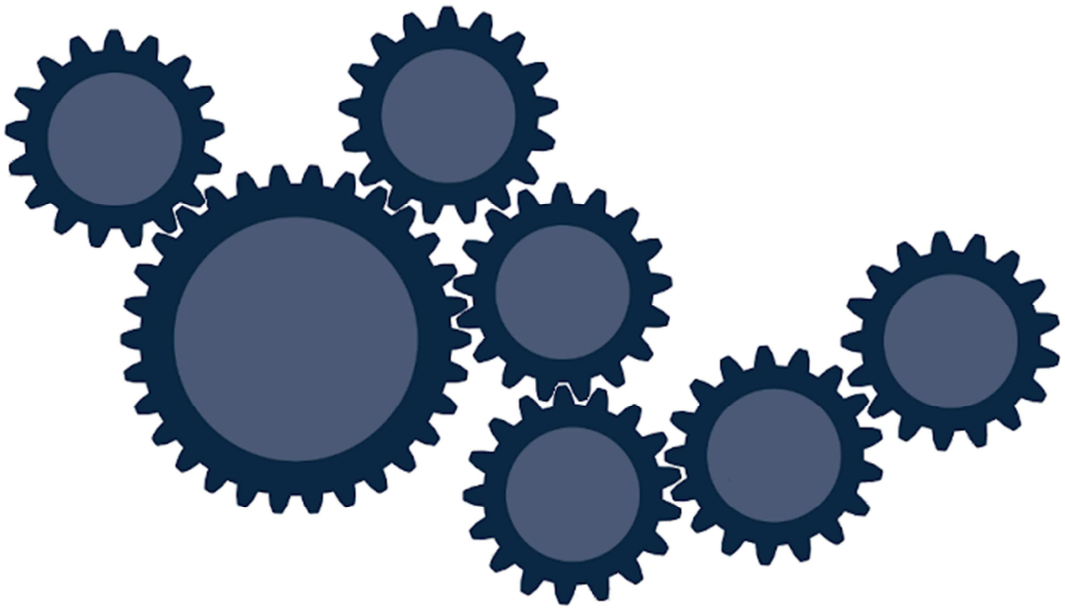
Έννοιες – Λέξεις Κλειδιά

Διάγραμμα ροής, Ψευδοκώδικας, Κώδικας σε γλώσσα προγραμματισμού, Πρότυπο Ε.C.M.A., Μεταβλητή, Περιοχή μνήμης, Απόφαση σε συνθήκη, Περιγραφικά ονόματα μεταβλητών, Λεξιλόγιο γλώσσας, Συντακτική δομή, Απλή ακολουθία, Δομή επιλογής, Δομή επανάληψης υπό συνθήκη, Δομημένος προγραμματισμός.

Εισαγωγικές Παρατηρήσεις

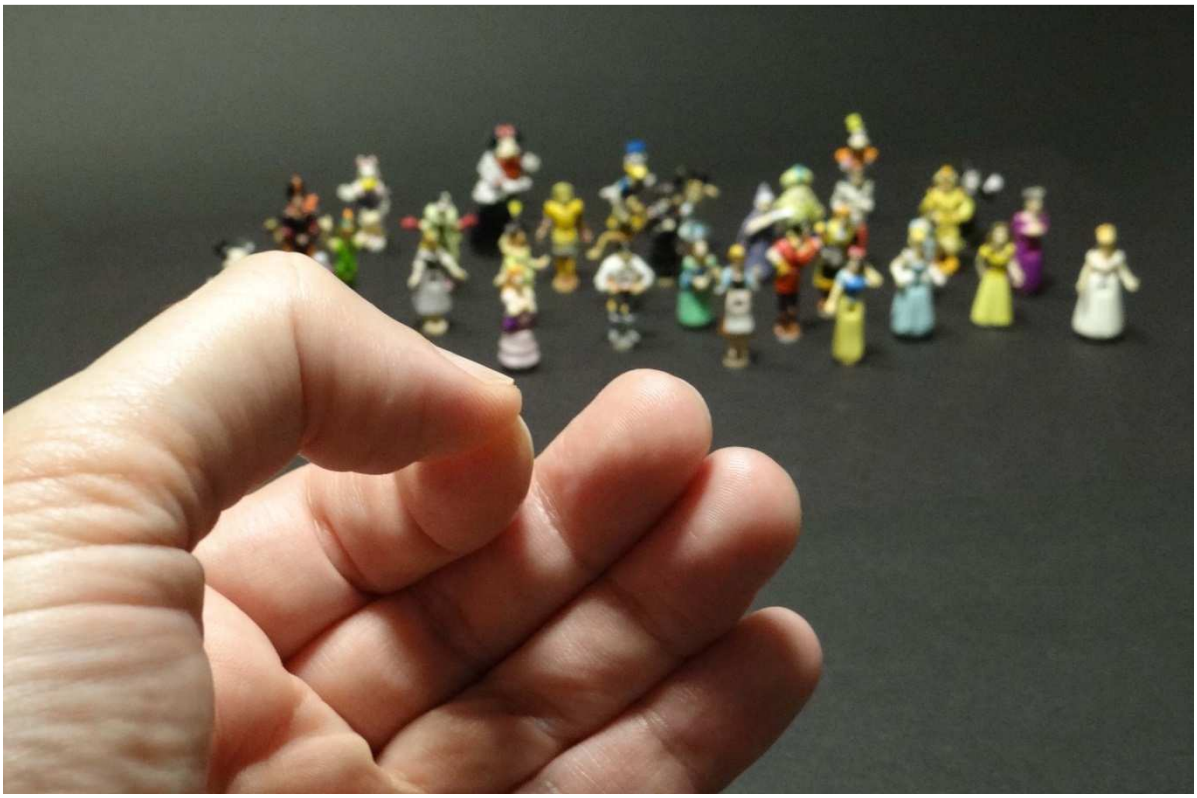
Η μελέτη των αλγορίθμων είναι ένα σημαντικό θέμα στον προγραμματισμό των ηλεκτρονικών υπολογιστών. Οι προγραμματιστές των ηλεκτρονικών υπολογιστών εξαντλούν την καθημερινότητα τους, στην μελέτη και την κατασκευή αλγορίθμων. Ένας αλγόριθμος στην καθημερινότητα των προγραμματιστών μπορεί να εμφανιστεί σε μια ή περισσότερες από τις παρακάτω μορφές,

- **Φυσική γλώσσα**: μια περιγραφή με μορφή διάλεξης, όλων των τεχνικών βημάτων που θα πρέπει να ακολουθήσει ο αλγόριθμος για να οδηγήσει στην λύση,
- **Ελεύθερο Κείμενο**: μια καταγραφή σε μορφή γραπτής έκθεσης, όλων των τεχνικών βημάτων που θα πρέπει να ακολουθήσει ο αλγόριθμος για να οδηγήσει στην λύση,
- **Διάγραμμα ροής**: ένα σύνολο απλών σχημάτων, που το καθένα δηλώνει κάποια συγκεκριμένη ενέργεια,
- **Ψευδοκώδικας**: μια δομημένη συγγραφή βημάτων γραμμένη σε μορφή ψευδο-γλώσσας (στα Ελληνικά ή τα Αγγλικά), που στηρίζεται σε συμφωνημένες τυποποιημένες δηλώσεις που στην συνέχεια μπορούν να μεταφερθούν σε οποιαδήποτε γλώσσα προγραμματισμού,
- **Πρόγραμμα**: ένα σύνολο συγκεκριμένων εντολών και ενεργειών σε μια συγκεκριμένη γλώσσα προγραμματισμού που μπορούν να αναγνωριστούν από έναν υπολογιστή.



4.1. Αλγόριθμος σε μορφή, Φυσικής Γλώσσας

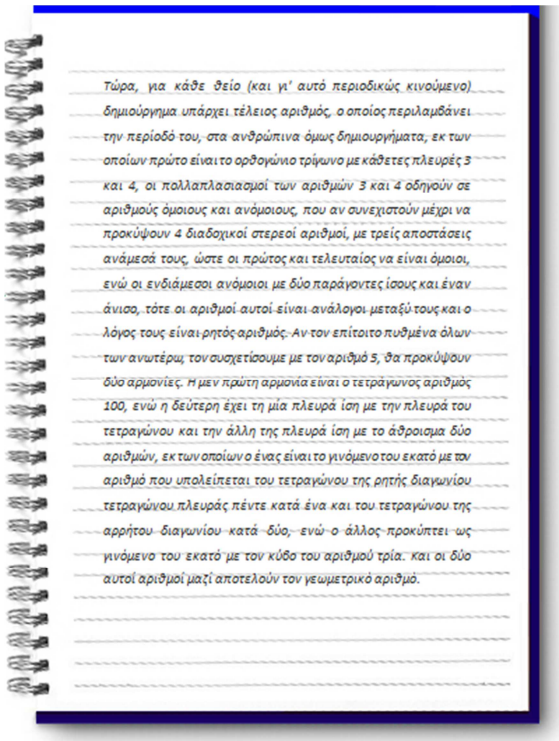
Οι άνθρωποι είναι όντα ομιλητικά και έχουν επίσης την ικανότητα να μιλούν πολλές διαφορετικές γλώσσες. Η χρήση της φυσικής γλώσσας είναι μια συνηθισμένη πρακτική για κάθε άνθρωπο. Ένας συνηθισμένος άνθρωπος ξεκινά να μιλά από το πρωί και συνεχίζει μέχρι να κοιμηθεί. Οι φυσικές γλώσσες βασίζονται σε συγκεκριμένους κανόνες που αποτελούνται από γράμματα, λεξιλόγια, γραμματικές και συντακτικό. Η χρήση της φυσικής γλώσσας από τους προγραμματιστές, είναι μια συνηθισμένη πρακτική για την περιγραφή της λειτουργικότητας ενός αλγορίθμου.

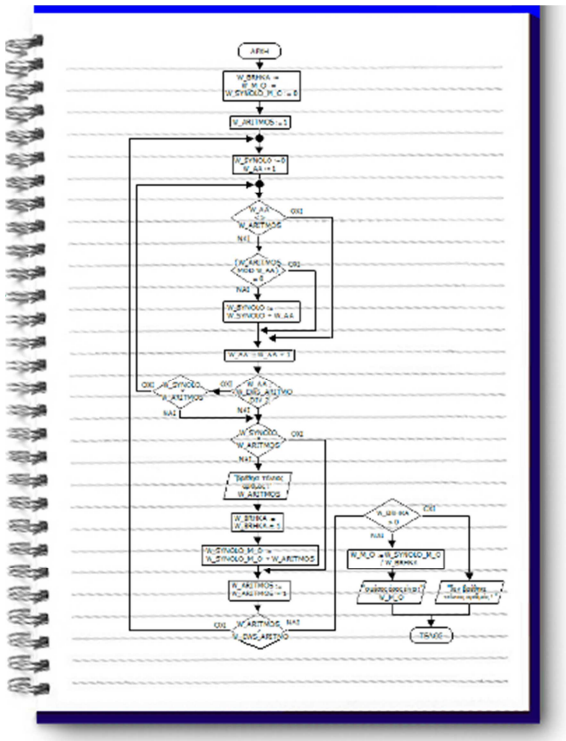


4.2. Αλγόριθμος σε μορφή, Ελεύθερου Κειμένου

Ο *αριθμός του Πλάτωνος* είναι ένας ιδανικός αριθμός που μνημονεύεται από τον Πλάτωνα στον διάλογό του, «Πολιτεία» (ΠΛΑΤΩΝΟΣ, Πολιτεία, 546b, 546c.) και αφορά την περιγραφή ενός αλγορίθμου σε ελεύθερο γραπτό κείμενο. Είναι γνωστός και σαν *γεωμετρικός αριθμός.*

«*Τώρα, για κάθε θείο (και γι' αυτό περιοδικώς κινούμενο) δημιούργημα υπάρχει τέλειος αριθμός, ο οποίος περιλαμβάνει την περίοδό του, στα ανθρώπινα όμως δημιουργήματα, εκ των οποίων πρώτο είναι το ορθογώνιο τρίγωνο με κάθετες πλευρές 3 και 4, οι πολλαπλασιασμοί των αριθμών 3 και 4 οδηγούν σε αριθμούς όμοιους και ανόμοιους, που αν συνεχιστούν μέχρι να προκύψουν 4 διαδοχικοί στερεοί αριθμοί, με τρείς αποστάσεις ανάμεσά τους, ώστε οι πρώτος και τελευταίος να είναι όμοιοι, ενώ οι ενδιάμεσοι ανόμοιοι με δύο παράγοντες ίσους και έναν άνισο, τότε οι αριθμοί αυτοί είναι ανάλογοι μεταξύ τους και ο λόγος τους είναι ρητός αριθμός. Αν τον επίτριτο πυθμένα όλων των ανωτέρω, τον συσχετίσουμε με τον αριθμό 5, θα προκύψουν δύο αρμονίες. Η μεν πρώτη αρμονία είναι ο τετράγωνος αριθμός 100, ενώ η δεύτερη έχει τη μία πλευρά ίση με την πλευρά του τετραγώνου και την άλλη της πλευρά ίση με το άθροισμα δύο αριθμών, εκ των οποίων ο ένας είναι το γινόμενο του εκατό με τον κύβο του αριθμού τρία. Και οι δύο αυτοί αριθμοί μαζί αποτελούν τον γεωμετρικό αριθμό.*»

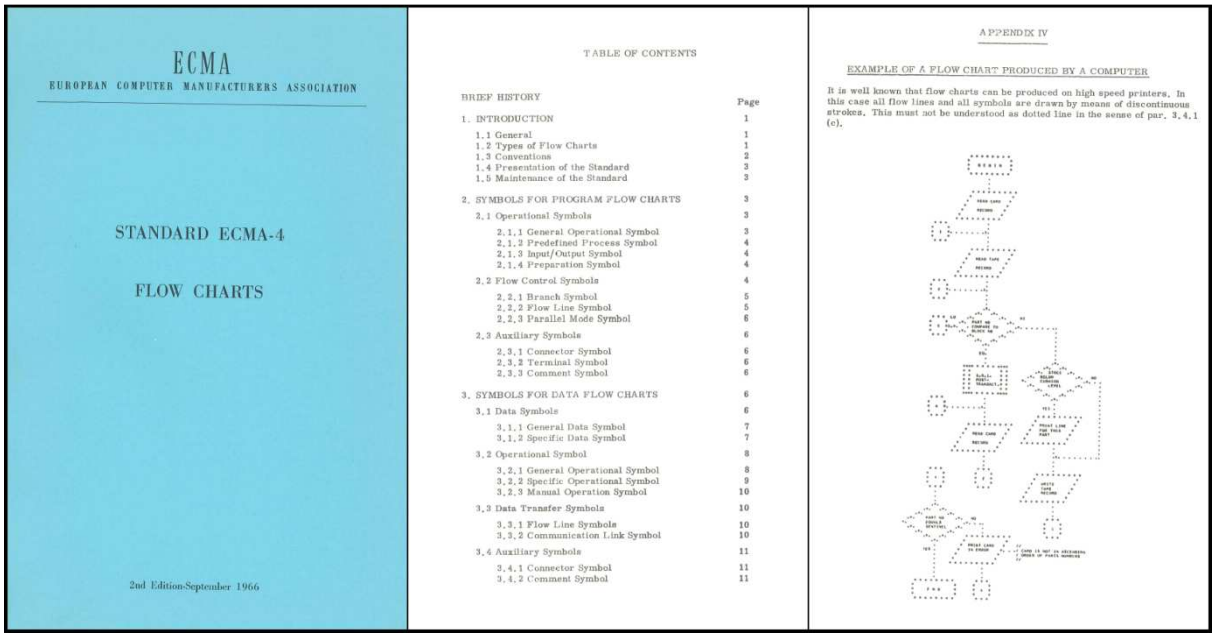




Η χρήση του γραπτού λόγου και η περιγραφή της λειτουργικότητας των αλγορίθμων σε μορφή ελεύθερου κειμένου είναι επίσης μια συνηθισμένη πρακτική στους κύκλους των προγραμματιστών ηλεκτρονικών υπολογιστών. Εάν επιλέξουμε να περιγράψουμε έναν αλγόριθμο σε ελεύθερο κείμενο θα πρέπει να ακολουθήσουμε τους συντακτικούς και γραμματικούς κανόνες αλλά και το αποδεκτό λεξιλόγιο της γλώσσας συγγραφής του κειμένου μας. Πολλές σημαντικές ιδέες ξεκίνησαν από ένα λευκό κομμάτι χαρτί.

4.3. Αλγόριθμος σε μορφή, Διαγράμματος

Η κατασκευή ενός αλγορίθμου σε μορφή σχεδιαγράμματος είναι μια πολύ συνηθισμένη πρακτική για τους προγραμματιστές, καθώς προσφέρει την ευκολία της οπτικής απεικόνισης ενός αλγορίθμου. Τα σχέδια αυτά είναι διαγράμματα τα οποία μπορούν να απεικονίσουν την ροή εκτέλεσης, για μια μικρή διαδικασία, ένα ολόκληρο σύστημα ή έναν σύνθετο αλγόριθμο. Χρησιμοποιούνται ευρέως για την ανάλυση, το σχεδιασμό, την τεκμηρίωση, τη μελέτη και τη βελτίωση περίπλοκων διαδικασιών, με απλά και κατανοητά δυσδιάστατα σχέδια. Τα διαγράμματα αυτά, ονομάζονται *λογικά διαγράμματα* ή *διαγράμματα ροής* προγράμματος (program flow diagrams ή program flowcharts). Κάθε καινούργιος προγραμματιστής, από κάθε σχολή προγραμματιστών, περνά από την διαδικασία της κατασκευής λογικών διαγραμμάτων. Τα λογικά διαγράμματα βασίζονται σε ένα παγκόσμιο πρότυπο το οποίο σχεδιάστηκε αρχικά το 1964, από την ευρωπαϊκή ένωση κατασκευαστών υπολογιστών E.C.M.A. (European Computer Manufacturers Association).



ECMA

EUROPEAN COMPUTER MANUFACTURERS ASSOCIATION

STANDARD ECMA-4

FLOW CHARTS

2nd Edition-September 1966

TABLE OF CONTENTS

APPENDIX IV

EXAMPLE OF A FLOW CHART PRODUCED BY A COMPUTER

It is well known that flow charts can be produced on high speed printers. In this case all flow lines and all symbols are drawn by means of discontinuous strokes. This must not be understood as dotted line in the sense of par. 3.4.1 (c).

Τα λογικά διαγράμματα εκτός από τους προγραμματιστές των ηλεκτρονικών υπολογιστών χρησιμοποιούνται και από χιλιάδες άλλους ανθρώπους σε πολλούς τομείς σε όλο τον κόσμο. Τα λογικά διαγράμματα μπορούν να δημιουργηθούν με το χέρι και ένα μολύβι, με την χρήση ενός ειδικού χάρακα (template) ή ακόμα και με ειδικά σχεδιαστικά προγράμματα σε υπολογιστή για να απεικονίζουν σύνθετους αλγορίθμους. Για την δημιουργία ενός λογικού διαγράμματος χρησιμοποιούνται συγκεκριμένα τυποποιημένα γεωμετρικά σύμβολα τα οποία περιγράφουν όλες τις απαραίτητες ενέργειες. Για παράδειγμα, ένα τετράγωνο σημαίνει την επεξεργασία και ένας ρόμβος σημαίνει ότι πρέπει να παρθεί μια απόφαση. Στο εσωτερικό κάθε συμβόλου ο προγραμματιστής μπορεί να αποτυπώσει με σημείωση την ενέργεια που εκτελεί αλλά και τα αποτελέσματα της. Επιπλέον, τα λογικά διαγράμματα εμπεριέχουν την έννοια του χρόνου καθώς περιγράφουν με ακολουθιακό τρόπο (βήμα προς βήμα) την πορεία της εξέλιξης του αλγορίθμου. Αυτό επιτυγχάνεται με την χρήση τόξων, συνδετικών γραμμών και συμβόλων ανακατεύθυνσης, της ροής εκτέλεσης, από και προς συγκεκριμένα σημεία (εσωτερικά ή εξωτερικά). Με αυτά τα σύμβολα κατεύθυνσης ροής, ο προγραμματιστής μπορεί να καταδείξει την πορεία της σκέψης του.

Σύμβολα λογικού διαγράμματος, κατά το πρότυπο E.C.M.A.

Τα λογικά διαγράμματα διαφοροποιούνται πολλές φορές ελαφρά, σε σχέση με την εποχή, την εκπαιδευτική βαθμίδα, τον τόπο αλλά κυρίως το τεχνικό περιβάλλον ανάπτυξης, όμως είναι πάντα επιθυμητό να ακολουθούμε τα βασικά πρότυπα.

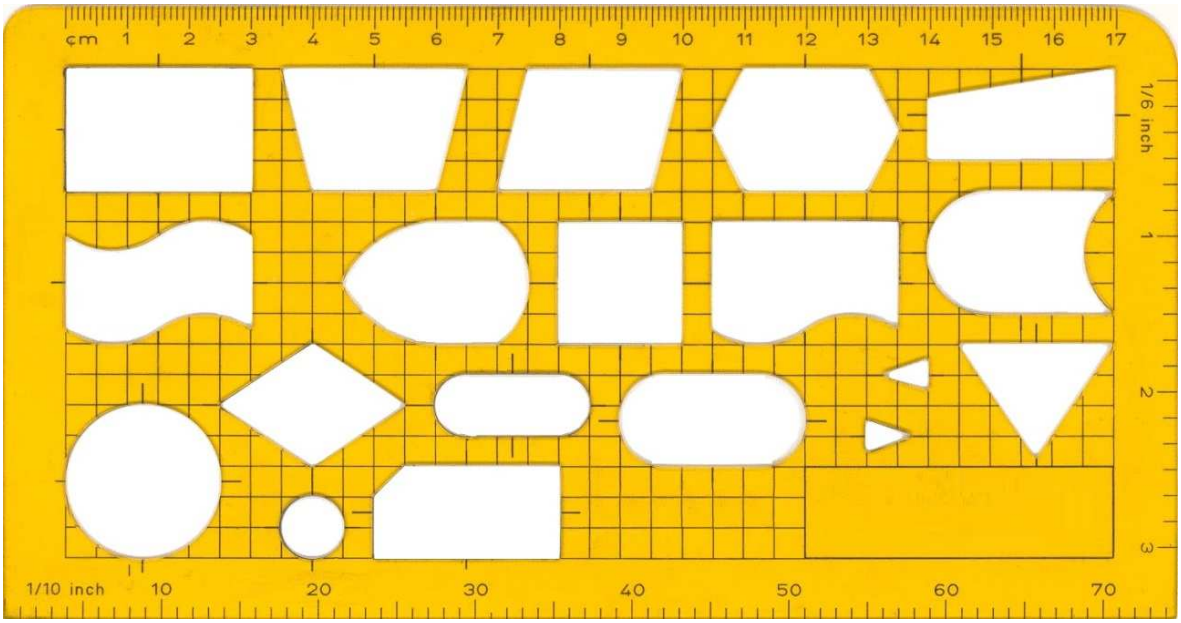
Ένα λογικό διάγραμμα μας δίνει την δυνατότητα,

- να παρουσιάσουμε μια οπτική αναπαράσταση του αλγορίθμου που καταστρώσαμε,
- να μελετήσουμε καλά ένα πρόβλημα και να εμβαθύνουμε σε αυτό με μεγαλύτερη ευκολία από μια αφηγηματική περιγραφή,
- να διασπάσουμε το πρόβλημα σε μικρότερες αυτόνομες οργανωτικές μονάδες από πολλά διαφορετικά λογικά διαγράμματα,

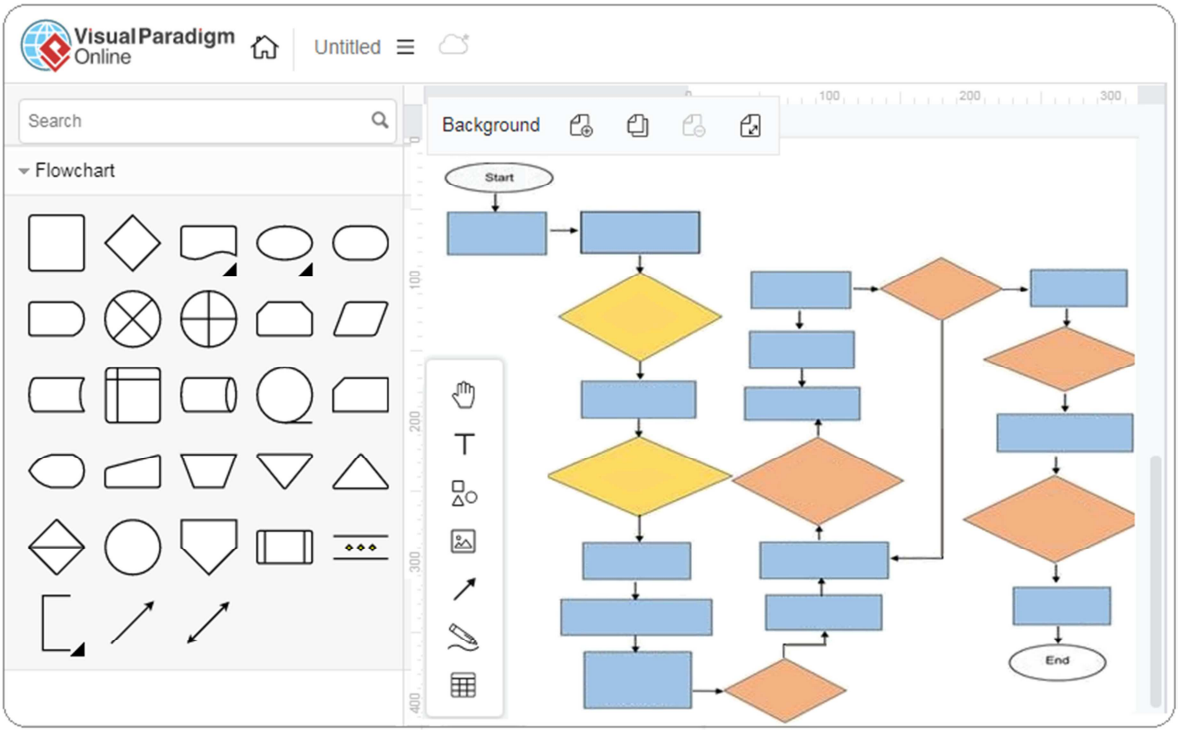
- να αποκτήσουμε μια συνολική εποπτική εικόνα πριν προχωρήσουμε στην συγγραφή του πραγματικού προγράμματος σε γλώσσα προγραμματισμού,
- να διαπιστώσουμε εγκαίρως κενά στον αλγόριθμο μας και να τα αποκαταστήσουμε,
- να επιδείξουμε με εύκολο τρόπο τις σκέψεις μας και να συνεργαστούμε με άλλους προγραμματιστές,
- να τεκμηριώσουμε το έργο μας.

Για να κατασκευάσουμε ένα λογικό διάγραμμα θα πρέπει,

- αρχικά να δώσουμε στο διάγραμμα μια ονομασία,
- να αποφασίσουμε πότε και πως θα ξεκινά η διαδικασία,
- να αποφασίσουμε πότε και πώς θα τελειώνει η διαδικασία,
- να καθορίσουμε το επίπεδο της λεπτομέρειας το οποίο θα παρουσιάσουμε,
- να συγκεντρώσουμε τα δεδομένα εισόδου που θα χρειαστεί να αποκτήσουμε και να καθορίσουμε τους τρόπους με τους οποίους θα τα αποκτήσουμε,
- να υπολογίσουμε τα αποτελέσματα και να σχεδιάσουμε τις πληροφορίες που θα κατασκευάσουμε και θα εξάγουμε,
- να συγκεντρώσουμε όλες τις επιμέρους δραστηριότητες που θα λαμβάνουν χώρα κατά την εκτέλεση του αλγορίθμου και να τις οριοθετήσουμε με το κατάλληλο γεωμετρικό σχήμα,
- να σημειώσουμε μέσα σε κάθε σύμβολο με κατάλληλο κείμενο, την ενέργεια που αποτυπώνει,
- να αποφασίσουμε σε ποια σημεία θα πρέπει να τοποθετήσουμε σχήματα ρόμβων, που υποδεικνύουν ερωτήσεις στην διαδικασία και από τις σχετικές απαντήσεις (ΝΑΙ ή ΟΧΙ) να ανακατευθύνουμε την ροή της διαδικασίας σε κατάλληλα σημεία,
- να τακτοποιήσουμε αυτά τα γεωμετρικά σχήματα με τις δραστηριότητες σε μια ορθή χωρική κατανομή,
- να ενώσουμε τα γεωμετρικά σχήματα με τις κατάλληλες ενώσεις και τα κατάλληλα βέλη ώστε να καθορίσουμε με ορθό τρόπο την ροή της διαδικασίας.



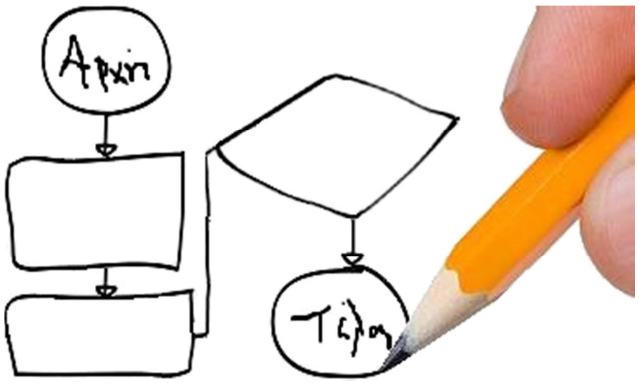
Ειδικός χάρακας (template), με τα σύμβολα του λογικού διαγράμματος.



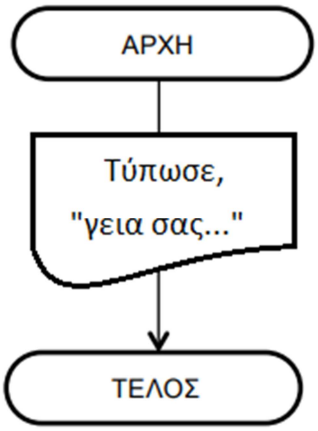
Λογισμικό δημιουργίας λογικών διαγραμμάτων (Visual Paradigm).

4.4. Το πρώτο Λογικό Διάγραμμα, κάθε Προγραμματιστή

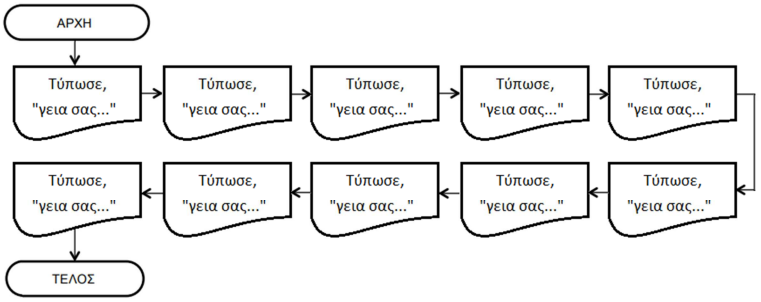
Ο πρώτος αλγόριθμος, που παραδοσιακά κάθε νέος προγραμματιστής καλείται να σχεδιάσει σε λογικό διάγραμμα ή να συντάξει σε μια γλώσσα προγραμματισμού, είναι ένας απλούστατος μηχανισμός ο οποίος, το μόνο που κάνει είναι να τυπώνει την φράση «*γεια σας…*» (ή στα αγγλικά «*Hello World…»).* Ο λόγος ύπαρξης αυτής της συνήθειας στον κόσμο των προγραμματιστών, είναι η ανάγκη ικανοποίησης των νέων ανθρώπων που έρχονται για πρώτη φορά σε επαφή με μια γλώσσα προγραμματισμού και ανυπομονούν να την κάνουν να λειτουργήσει για λογαριασμό τους. Σε όλες λοιπόν τις γλώσσες προγραμματισμού, σε όλα τα βιβλία, σε όλες τις εποχές και όλες τις σχολές, έχει καθιερωθεί εδώ και δεκαετίες να περιγράφεται η διαδικασία συγγραφής αυτού του μικρού αλγορίθμου.



Το επόμενο λογικό διάγραμμα περιγράφει αυτόν ακριβώς τον μηχανισμό και διαθέτει 3 γεωμετρικά σύμβολα από το πρότυπο Ε.C.M.A. Ένα σύμβολο που αποτυπώνει την έναρξη, ένα σύμβολο που αποτυπώνει την εκτύπωση και ένα σύμβολο που αποτυπώνει την έξοδο.

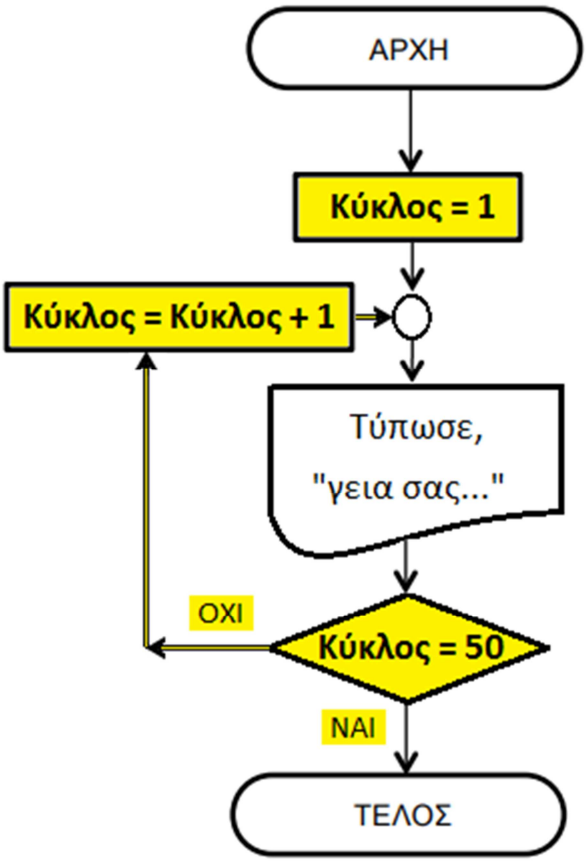


Ομοίως, εάν θέλαμε να κατασκευάσουμε μια μηχανή η οποία θα τύπωνε την ίδια φράση για 10 φορές, θα έπρεπε να καταστρώσουμε το επόμενο λογικό διάγραμμα,



και επιπλέον, εάν θέλαμε να σχεδιάσουμε μια νέα μηχανή η οποία θα τύπωνε την ίδια φράση για 50 φορές, θα έπρεπε να καταστρώσουμε το επόμενο λογικό διάγραμμα.

Επειδή όμως ο προγραμματισμός των ηλεκτρονικών υπολογιστών, εκτός από ενδιαφέρουσα, είναι και εθιστική απασχόληση, πολλοί νέοι και φιλόδοξοι προγραμματιστές θα αναρωτιόντουσαν ήδη, πως θα ήταν δυνατό να συγγράψουν κάποιον εξυπνότερο κώδικα και όχι να αναγκάζονται να επαναλαμβάνουν συνεχώς τα ίδια σύμβολα. Το παρακάτω λογικό διάγραμμα περιγράφει ένα επόμενο στάδιο προγραμματιστικής αντίληψης με έναν «εξυπνότερο» αλγόριθμο, ο οποίος εκτυπώνει επαναλαμβανόμενα 50 φορές, την φράση «*γεια σας…*», με εμφανώς λιγότερα γεωμετρικά σύμβολα και προσπάθεια.

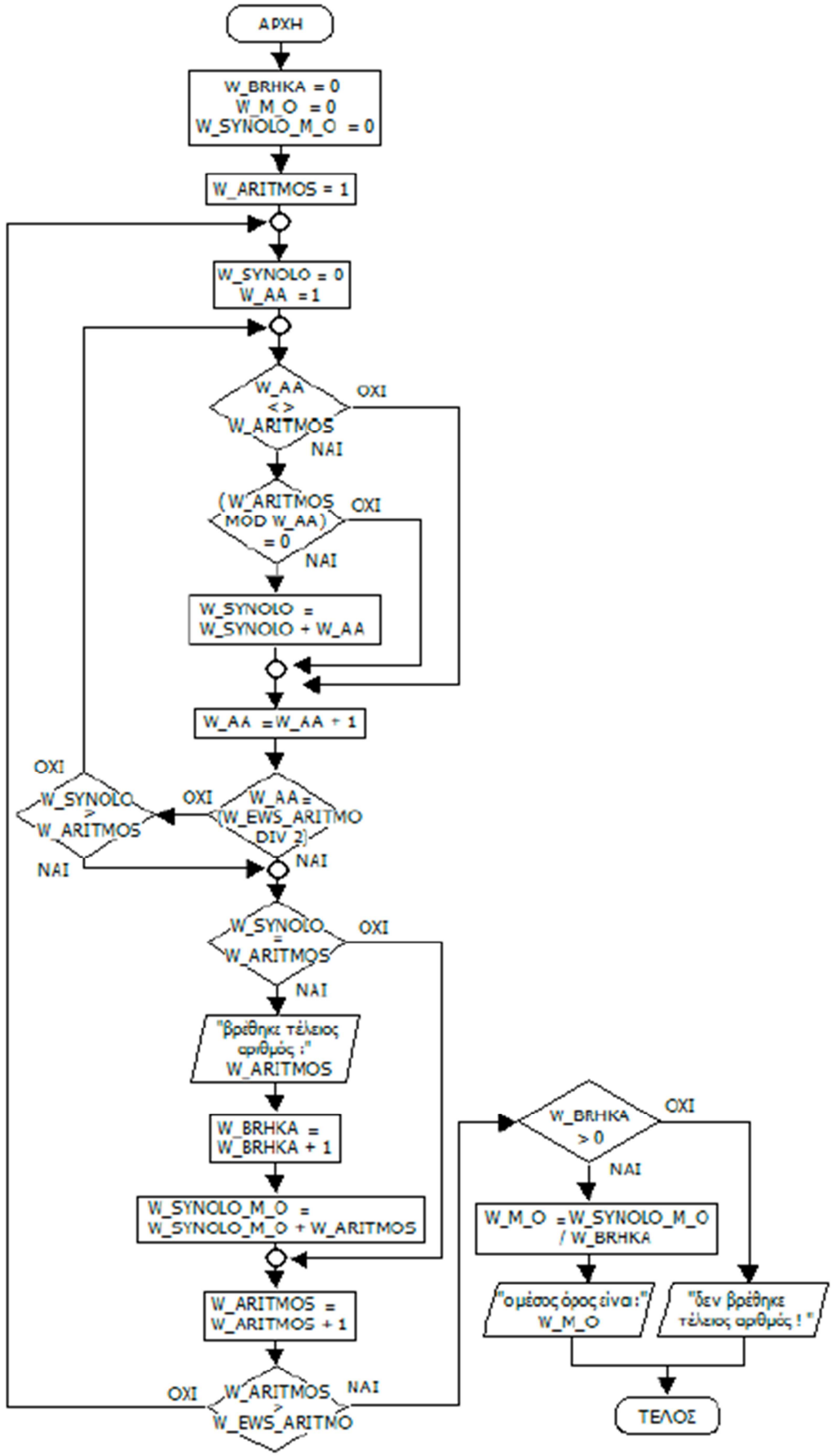


Είναι προφανές ότι το διάγραμμα μοιάζει στο αρχικό σχέδιο με τα 3 γεωμετρικά σύμβολα (την έναρξη, την εκτύπωση και την έξοδο) με την διαφορά όμως ότι έχει εμπλουτιστεί και με άλλα 3 νέα σύμβολα τα οποία χρωματίζονται (για να γίνουν εύκολα αντιληπτά) με κίτρινο χρώμα. Τα 3 αυτά νέα κίτρινα σύμβολα αποτυπώνουν την ισχυρότερη έννοια στην τεχνολογία της κατασκευής προγραμμάτων ηλεκτρονικών υπολογιστών. Την έννοια της επανάληψης υπό συνθήκη. Όλες οι γλώσσες προγραμματισμού, όλων των εποχών και όλων των τεχνολογιών βασίζονται με μικρές διαφοροποιήσεις σε αυτή ακριβώς την έννοια.

Η επανάληψη υπό συνθήκη, σε ελεύθερο κείμενο, σημαίνει για τον υπολογιστή, συνέχιζε να ανακυκλώνεις, δηλαδή να επαναλαμβάνεις ένα συγκεκριμένο κομμάτι κώδικα, μέχρι να ξεπεραστεί ένα προκαθορισμένο όριο επιθυμητών επαναλήψεων. Το όριο αυτών των επαναλήψεων καθορίστηκε από τον προγραμματιστή με την χρήση μιας περιοχής προσωρινής αποθήκευσης στην οποία της δόθηκε (για λόγους ευκολίας και απομνημόνευσης) το όνομα «Κύκλος». Ακριβώς όπως, το «καπελάκι» στο οποίο αποθηκεύονται τα κρατούμενα στην πράξη της πρόσθεσης με το χέρι. Στην προσωρινή αυτή περιοχή αποθήκευσης, που στις γλώσσες προγραμματισμού ονομάζεται *μνήμη*, αποθηκεύεται προοδευτικά ο αριθμός των περασμάτων, (η τρέχουσα επανάληψη) που εκτελεί ο αλγόριθμος (1, 2, 3, 4, … 50). Συνεπώς, γίνεται αντιληπτό ότι ο μηχανισμός ανακύκλωσης, βασίζεται στην μεταβλητή «Κύκλος», που με την έναρξη λειτουργίας του μηχανισμού, λαμβάνει την αρχική τιμή 1, (Κύκλος=1) και καθώς η ροή προχωρά, θα τυπώσει για πρώτη φορά την φράση «*γεια σας…*», ενώ στην συνέχεια θα φτάσει στο γεωμετρικό σύμβολο του ρόμβου που αναγράφει την ερώτηση, «*η μεταβλητή «Κύκλος», είναι ίση με τον αριθμό 50?),* (Κύκλος = 50) και επειδή η απόφαση θα είναι αρνητική, (ΟΧΙ) η ροή του αλγορίθμου θα ανακατευθυνθεί και πάλι προς τα επάνω, περνώντας όμως από ένα νέο τετράγωνο επεξεργασίας στο οποίο η «Κύκλος», θα αυξήσει το περιεχόμενο της κατά 1 (Κύκλος=Κύκλος+1). Για την συνέχεια, η ροή του αλγορίθμου θα συνεχίζει να τυπώνει επαναλαμβανόμενα την φράση, «*γεια σας…*», μέσα από συνεχιζόμενες ανακυκλώσεις μέχρι η μεταβλητή «Κύκλος», να λάβει την τιμή 50 και άρα όταν θα φτάσει και πάλι στον σχετικό ρόμβο, η απόφαση στην νέα ερώτηση (την $50^{η}$ ερώτηση) θα είναι θετική, (ΝΑΙ) με αποτέλεσμα η ροή του αλγορίθμου να οδηγηθεί τελικά σε έξοδο, αφού θα έχει τυπώσει 50 συνολικά «*γεια σας…*».

4.5. Σύνθετα Λογικά Διαγράμματα

Τα διαγράμματα ροής χρησιμοποιούνται ευρέως για την περιγραφή μεγάλων και σύνθετων αλγορίθμων. Το επόμενο παράδειγμα αφορά έναν μικρό διάγραμμα το οποίο αναπαριστά έναν σύνθετο αλγόριθμο ο οποίος προσπαθεί να υπολογίσει και να τυπώσει όλους τους τέλειους αριθμούς. «*Τέλειος*» λέγεται ένας ακέραιος αριθμός όταν το άθροισμα των θετικών διαιρετών του, εκτός του ίδιου του αριθμού, είναι ίσο με τον αριθμό αυτό. Η εύρεση ενός τέλειου αριθμού δεν είναι μια εύκολη υπόθεση και μέχρι το 2018 είχαν βρεθεί συνολικά μόνο 50 τέλειοι αριθμοί. Οι πρώτοι 4 τέλειοι αριθμοί (το 6, το 28, το 496 και το 8128) ανακαλύφθηκαν από τον Ευκλείδη τον 4^{ο} αιώνα π.Χ. Ο 49^{ος} τέλειος αριθμός ανακαλύφθηκε το 2016 και είναι ένα απίθανα τεράστιο νούμερο, ο ακέραιος αριθμός 451120062706…557930315776. Οι τέλειοι αριθμοί χρησιμοποιούνται στην επίλυση μαθηματικών εξισώσεων και στην κρυπτογραφία. Παρόλο που το λογικό διάγραμμα που ακολουθεί περιγράφει σωστά τον αλγόριθμο και λειτουργεί ομαλά, εάν μετατραπεί σε ένα πρόγραμμα σε μια γλώσσα προγραμματισμού, είναι άγνωστος ο χρόνος και η μνήμη που θα χρειαστεί από έναν σημερινό οικιακό ηλεκτρονικό υπολογιστή, για να υπολογιστεί και να τυπωθεί ένα ικανό πλήθος τέλειων αριθμών. Το πρόγραμμα δηλαδή που θα κατασκευαστεί θα λειτουργεί κανονικά, αλλά είναι άγνωστο εάν θα τελειώσει ποτέ, εξαιτίας των τεράστιων σε όγκο αριθμητικών δεδομένων που θα πρέπει να προσπελαστούν. Εάν όμως κάποιος προγραμματιστής καταφέρει να το αφήσει να τρέξει, σε έναν πολύ ισχυρό υπολογιστή, τότε θα έχει βρει τον 51^{ο} τέλειο αριθμό και θα έχει πετύχει να αφήσει το όνομα του, στην ιστορία.


ΑΡΧΗ
W_BRHKA = 0
W_M_O = 0
W_SYNOLO_M_O = 0
W_ARITMOS = 1
W_SYNOLO = 0
W_AA = 1
W_AA <> W_ARITMOS
ΟΧΙ
ΝΑΙ
(W_ARITMOS MOD W_AA) = 0
ΟΧΙ
ΝΑΙ
W_SYNOLO = W_SYNOLO + W_AA
W_AA = W_AA + 1
W_AA = (W_EWS_ARITMO DIV 2)
ΟΧΙ
ΝΑΙ
W_SYNOLO > W_ARITMOS
ΟΧΙ
ΝΑΙ
W_SYNOLO = W_ARITMOS
ΟΧΙ
ΝΑΙ
"βρέθηκε τέλειος αριθμός :" W_ARITMOS
W_BRHKA = W_BRHKA + 1
W_SYNOLO_M_O = W_SYNOLO_M_O + W_ARITMOS
W_ARITMOS = W_ARITMOS + 1
W_ARITMOS > W_EWS_ARITMO
ΟΧΙ
ΝΑΙ
W_BRHKA > 0
ΟΧΙ
ΝΑΙ
W_M_O = W_SYNOLO_M_O / W_BRHKA
"ο μέσος όρος είναι:" W_M_O
"δεν βρέθηκε τέλειος αριθμός ! "
ΤΕΛΟΣ

4.6. Αλγόριθμος σε μορφή, Ψευδοκώδικα

Η κατασκευή ενός αλγορίθμου σε μορφή ελεύθερου κειμένου ή σε μορφή λογικού διαγράμματος είναι μια εύκολη διαδικασία καθώς βασίζεται σε μια προϋπάρχουσα αντίληψη και ικανότητα την οποία διαθέτουν σχεδόν όλοι οι άνθρωποι. Η ικανότητα από όλους μας, για γραπτή έκφραση και για κατάστρωση απλών σχεδίων, υποστηρίζει τις δυο αυτές μορφές. Ένας αλγόριθμος όμως γραμμένος σε αυτές τις δυο μορφές δεν μπορεί να εκτελεστεί από μια μηχανή, διότι απλά δεν μπορεί να αναγνωριστεί από την μηχανή.

Η μεταφορά ενός λογικού διαγράμματος σε μια γλώσσα προγραμματισμού, την οποία θα κατανοεί ένας υπολογιστής, είναι μια δύσκολη διαδικασία καθώς απαιτεί την γνώση και την απομνημόνευση ενός μεγάλου και σύνθετου λεξιλογίου τεχνικών όρων ενώ απαιτεί και την εφαρμογή αυστηρών συντακτικών κανόνων. Για την υποβοήθηση του έργου των νέων προγραμματιστών, στην μετάβαση σε αληθινές γλώσσες προγραμματισμού ηλεκτρονικών υπολογιστών, χρησιμοποιείται εκπαιδευτικά η ενδιάμεση μέθοδος της δημιουργίας προγραμμάτων σε ψευδοκώδικα ή ψευδογλώσσα. Η ψευδογλώσσα είναι μια προσομοίωση μιας υποθετικής γλώσσας προγραμματισμού, με κάποιους βασικούς κανόνες δανεισμένους από μια φυσική γλώσσα (τα Ελληνικά ή τα Αγγλικά) και χωρίς αυστηρούς συντακτικούς περιορισμούς. Επιπλέον μια ψευδογλώσσα μπορεί να την κατασκευάσει οποιοσδήποτε αρκεί να την συμφωνήσει με κάποιον άλλον, για να μπορεί εκείνος να διαβάζει και να ερμηνεύει τις ιδέες του. Είναι σύνηθες φαινόμενο κάθε εταιρία ανάπτυξης προγραμμάτων ή κάθε σχολή προγραμματιστών να διαθέτει, ο καθένας για λογαριασμό του, μια δική του σχεδόν όμοια ψευδογλώσσα. Ακόμα όμως και εάν ένας εκπαιδευόμενος σε μια εξεταστική διαδικασία αλλάξει την μεταξύ τους (με τον εκπαιδευτή) συμφωνημένη λέξη ΕΚΤΕΛΕΣΕ, με την λέξη ΕΠΕΞΕΡΓΑΣΟΥ, ο εκπαιδευτής θα πρέπει να την θεωρήσει ορθή καθώς είναι συνώνυμη και αποδίδει την ίδια ακριβώς έννοια. Αυτό, ο ηλεκτρονικός υπολογιστής (η μηχανή), δεν μπορεί να το καταλάβει και εάν η συμφωνημένη (μεταξύ της εταιρίας που έφτιαξε την γλώσσα προγραμματισμού του υπολογιστή και των προγραμματιστών που την έμαθαν) λέξη COMPUTE, δοθεί στον υπολογιστή, σαν λέξη RUN, ή με ορθογραφικό λάθος, σαν λέξη CAMPUTE, τότε απλά η μηχανή δεν θα μπορεί να την αναγνωρίσει και δεν θα μπορεί να συνεχίζει να εκτελεί τον αλγόριθμο. Ενώ ο εκπαιδευτής μπορεί να κάνει τα «στραβά μάτια». Αυτός είναι ο δεύτερος λόγος εκμάθησης της ψευδογλώσσας. Προσπαθούμε δηλαδή να

κάνουμε τους ανθρώπους που ενδιαφέρονται να γίνουν προγραμματιστές να αποκτήσουν την κουλτούρα του προγραμματιστή.

Στις ψευδογλώσσες ενσωματώνονται όλες οι συντακτικές δομές που ανευρίσκονται και στις κανονικές γλώσσες προγραμματισμού, ενώ η αυστηρότητα του λεξιλογίου και του τρόπου σύνταξης υπολείπεται. Οι αλγόριθμοι που αναπτύσσονται σε αυτό το σύγγραμμα και είναι γραμμένοι σε ψευδογλώσσα, βασίζονται στους βασικούς κανόνες (τις συμφωνημένες λέξεις) που σημειώνονται στον επόμενο πίνακα και θεωρούνται συμφωνημένοι μεταξύ: α) του συγγράμματος , β) του εκπαιδευτή και γ) του αναγνώστη εκπαιδευομένου.

ΔΙΑΔΙΚΑΣΙΑ ...	ΕΑΝ ... ΤΟΤΕ ... ΑΛΛΙΩΣ ... ΕΑΝ-ΤΕΛΟΣ	ΑΡΧΗ ... ΤΕΛΟΣ-ΔΙΑΔΙΚΑΣΙΑΣ
ΔΙΕΠΑΦΗ ΕΙΣΟΔΟΣ ... ΕΞΟΔΟΣ ...		ΤΥΠΩΣΕ ...
ΔΕΔΟΜΕΝΑ ...	ΕΠΑΝΕΛΑΒΕ ... ΜΕΧΡΙ ... ΕΠΑΝΕΛΑΒΕ-ΤΕΛΟΣ	ΥΠΟΛΟΓΙΣΕ ...
ΑΠΟΘΗΚΕΥΣΕ ...		ΠΡΟΒΑΛΕ ...
ΚΑΛΕΣΕ ...		ΔΙΑΒΑΣΕ ...

Ο χρωματισμός των συμφωνημένων **λέξεων** υποβοηθά την προσπάθεια ανάγνωσης κάθε εντολής και την κατανόηση των σκέψεων λειτουργίας σε κάθε αλγόριθμο. Όλες οι υπόλοιπες μη χρωματισμένες λέξεις είναι περιγραφές ενεργειών, σχόλια και ονόματα μεταβλητών που δόθηκαν από τον προγραμματιστή (σαν ανεξάρτητα «καπελάκια) τα οποία και χρησιμοποιούνται σε κάθε αντίστοιχο κώδικα για να πραγματοποιηθούν οι υπολογισμοί. Συμφωνείται επίσης ότι, στο τέλος της κάθε εντολής (ή στην τελευταία λέξη κάθε σύνθετης συντακτικής δομής) θα παρουσιάζεται το σύμβολο του ελληνικού ερωτηματικού **;** (semicolon) το οποίο θα υποδεικνύει το τέλος της εντολής. Το σύμβολο αυτό θα χρησιμοποιείται σε όλα τα

παραδείγματα ψευδοκώδικα, στο τέλος κάθε εντολής, όπως το σύμβολο της τελείας **.** που τοποθετείται στο γραπτό κείμενο και υποδεικνύει το τέλος της κάθε πρότασης. Τα παραδείγματα που ακολουθούν, αναπαριστούν τρείς απλούς αλγορίθμους σε ψευδογλώσσα, με τον καθένα από αυτούς να περιγράφει αντίστοιχα, α) μια απλή ακολουθία ενεργειών, β) μια δομή επιλογής ενεργειών και γ) μια δομή επανάληψης με κάποια συνθήκη.

ΑΠΛΗ ΑΚΟΛΟΥΘΙΑ	ΔΟΜΗ ΕΠΙΛΟΓΗΣ	ΔΟΜΗ ΕΠΑΝΑΛΗΨΗΣ
ΔΙΑΔΙΚΑΣΙΑ *Το 1ο μου πρόγραμμα* ; **ΔΕΔΟΜΕΝΑ** «*Το καπελάκι*» **ΑΚΕΡΑΙΟΣ** ; **ΑΡΧΗ** **ΔΙΑΒΑΣΕ** Από το πληκτρολόγιο ; **ΑΠΟΘΗΚΕΥΣΕ** Στην περιοχή μνήμης με το όνομα «*Το καπελάκι*», αυτό που διάβασες από το πληκτρολόγιο ; **ΤΥΠΩΣΕ** Το περιεχόμενο μέσα από την μνήμη, με το όνομα	**ΔΙΑΔΙΚΑΣΙΑ** *Το 2ο μου πρόγραμμα* ; **ΔΕΔΟΜΕΝΑ** «*Θερμοκρασία*» **ΑΚΕΡΑΙΟΣ** ; **ΑΡΧΗ** **ΔΙΑΒΑΣΕ** Από το πληκτρολόγιο ; **ΑΠΟΘΗΚΕΥΣΕ** Στην περιοχή μνήμης με το όνομα «*Θερμοκρασία*», αυτό που διάβασες από το πληκτρολόγιο ; **ΕΑΝ** «*Θερμοκρασία*» **>** 35 **ΤΟΤΕ**	**ΔΙΑΔΙΚΑΣΙΑ** *Το 3ο μου πρόγραμμα* ; **ΔΕΔΟΜΕΝΑ** «*Ο μετρητής* » **ΑΚΕΡΑΙΟΣ** ; **ΑΡΧΗ** **ΥΠΟΛΟΓΙΣΕ** «*Ο μετρητής*» **=** 1 ; **ΕΠΑΝΕΛΑΒΕ** **ΤΥΠΩΣΕ** «Αγαπώ την εκπαίδευση» ; **ΤΥΠΩΣΕ** «Πλουτίζω σε γνώσεις» ; **ΥΠΟΛΟΓΙΣΕ**

«*Το καπελάκι*» ; **ΤΕΛΟΣ-ΔΙΑΔΙΚΑΣΙΑΣ** ;	**ΤΥΠΩΣΕ** «*Είναι πάνω από 35*» ; **ΤΥΠΩΣΕ** «*Κάνει πολύ ζέστη*!» ; **ΑΛΛΙΩΣ** **ΤΥΠΩΣΕ** «*Είναι κάτω από 35*» ; **ΕΑΝ-ΤΕΛΟΣ** ; **ΤΕΛΟΣ-ΔΙΑΔΙΚΑΣΙΑΣ** ;	«*Ο μετρητής*» **=** «*Ο μετρητής*» **+ 1** ; **ΜΕΧΡΙ** «*Ο μετρητής*» **<** 11 **ΕΠΑΝΕΛΑΒΕ-ΤΕΛΟΣ** ; **ΤΥΠΩΣΕ** «Έμαθα, να μαθαίνω» ; **ΤΕΛΟΣ-ΔΙΑΔΙΚΑΣΙΑΣ** ;

Στο παράδειγμα «*Το 1ο μου πρόγραμμα*», ο ψευδοκώδικας εκτελεί μια απλή ανάγνωση από το πληκτρολόγιο, τοποθετεί την πληκτρολόγηση (με την λέξη **ΑΠΟΘΗΚΕΥΣΕ**), σε μια προσωρινή περιοχή μνήμης (με το όνομα «*Το καπελάκι*») και ακολούθως τυπώνει (με την λέξη **ΤΥΠΩΣΕ**), το περιεχόμενο της μεταβλητής «*Το καπελάκι*» και τελικά τερματίζει. Ο αλγόριθμος είναι μια απλή ακολουθιακή διαδικασία ροής που εκκινεί από την λέξη **ΑΡΧΗ**, προχωρά σταδιακά προς τα κάτω (σαν μια «μπίλια» που «πέφτει») και τελικά καταλήγει στην λέξη **ΤΕΛΟΣ-ΔΙΑΔΙΚΑΣΙΑΣ**.

Με μια όμοια διαδικασία στο παράδειγμα «*Το 2ο μου πρόγραμμα*», μια «αόρατη μπίλια» προχωρά σταδιακά προς τα κάτω, γεμίζοντας μια προσωρινή περιοχή μνήμης (με το όνομα «*Θερμοκρασία*») και συναντά στην πορεία της την λέξη **ΕΑΝ** που λειτουργεί σαν τον ρόμβο, από τα λογικά διαγράμματα (flowcharts) που υποδηλώνει ερώτηση, με την απόφαση (οι λέξεις **ΤΟΤΕ** και **ΑΛΛΙΩΣ)** να κατευθύνει την ροή του αλγορίθμου, στην εκτύπωση των φράσεων, «*Είναι πάνω από 35*», «*Κάνει πολύ ζέστη*!» ή στην εκτύπωση της φράσης «*Είναι κάτω από 35*». Ο αλγόριθμος και η δομή της ερώτησης που αποτυπώνεται με τη συλλογή των εντολών (**ΕΑΝ**

… **ΤΟΤΕ** … **ΑΛΛΙΩΣ** … **ΕΑΝ-ΤΕΛΟΣ**), ονομάζεται, στον κόσμο του προγραμματισμού, *δομή επιλογής*.

Στο επόμενο παράδειγμα «*Το 3ο μου πρόγραμμα*», η «αόρατη μπίλια» προχωρά σταδιακά προς τα κάτω, συναντώντας στην πορεία της, την λέξη **ΕΠΑΝΕΛΑΒΕ**, που λειτουργεί επίσης σαν ρόμβος, αλλά έχει επιπλέον την δυνατότητα να παγιδεύει την «αόρατη μπίλια» (τη ροή του αλγορίθμου) σε μια ατέρμονη διαδικασία μέχρι να συμβεί ένα συγκεκριμένο γεγονός. Εδώ η «μπίλια» εγκλωβίζεται ανάμεσα στη συλλογή των εντολών (**ΕΠΑΝΕΛΑΒΕ** … **ΜΕΧΡΙ** … **ΕΠΑΝΕΛΑΒΕ-ΤΕΛΟΣ**) μέχρι η ερώτηση («*Ο μετρητής*» < 11), να παράγει αρνητική απόφαση. Την ικανότητα αρνητικής απόφασης, προκαλεί η εντολή **ΥΠΟΛΟΓΙΣΕ**, η οποία είναι τοποθετημένη μέσα στην δομή εγκλωβισμού και αυξάνει κατά +1 μια προσωρινή περιοχή μνήμης (με το όνομα «*Ο μετρητής*»). Η περιοχή αυτή λειτουργεί σαν ένας απλός καταμετρητής «καπελάκι». Όταν αυτός ο βοηθητικός καταμετρητής φτάσει τον αριθμό 11, τότε ο εγκλωβισμός της «μπίλιας» θα ελευθερωθεί και θα της επιτρέψει να «πέσει» προς την επόμενη **ΤΥΠΩΣΕ** εντολή για να προκαλέσει την εκτύπωση της φράσης, «*Έμαθα, να μαθαίνω*». Ο αλγόριθμος και η δομή αυτών των εντολών στον κόσμο των προγραμματιστών, ονομάζεται *δομή επανάληψης*.

4.7. Σύνθετοι Ψευδοκώδικες

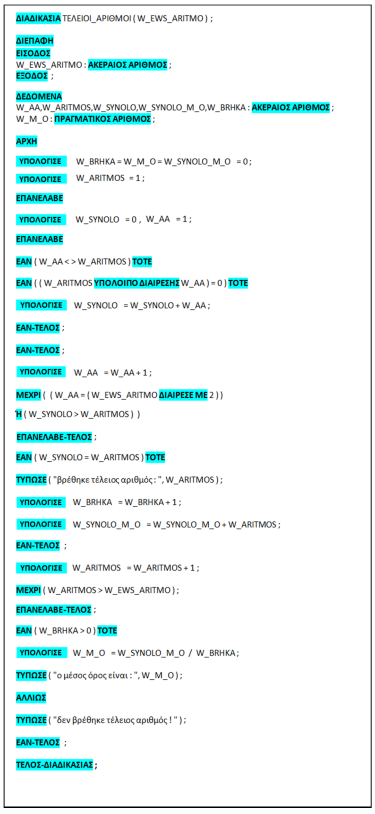
Το επόμενο παράδειγμα αναπαριστά με ψευδοκώδικα τον αλγόριθμο που υπολογίζει και τυπώνει όλους τους τέλειους αριθμούς, όπως αυτός περιγράφηκε με λογικό διάγραμμα στο προηγούμενο κεφάλαιο.

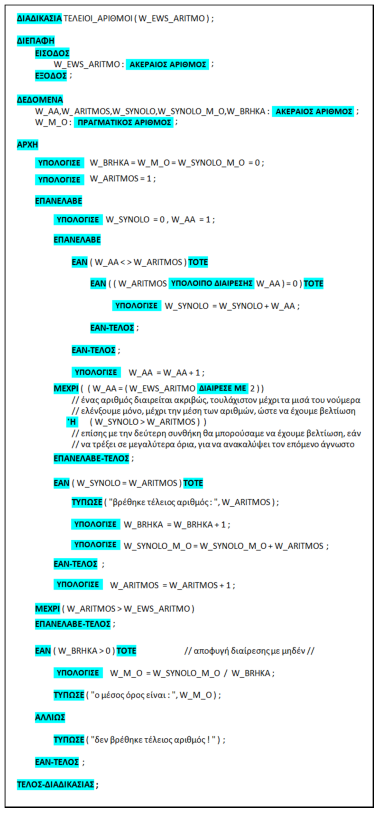
```
ΔΙΑΔΙΚΑΣΙΑ ΤΕΛΕΙΟΙ_ΑΡΙΘΜΟΙ ( W_EWS_ARITMO ) ;

ΔΙΕΠΑΦΗ
    ΕΙΣΟΔΟΣ
        W_EWS_ARITMO : ΑΚΕΡΑΙΟΣ ΑΡΙΘΜΟΣ ;
    ΕΞΟΔΟΣ ;

ΔΕΔΟΜΕΝΑ
    W_AA,W_ ARITMOS,W_SYNOLO,W_SYNOLO_M_O,W_BRHKA : ΑΚΕΡΑΙΟΣ ΑΡΙΘΜΟΣ ;
    W_M_O : ΠΡΑΓΜΑΤΙΚΟΣ ΑΡΙΘΜΟΣ ;

ΑΡΧΗ
    ΥΠΟΛΟΓΙΣΕ  W_BRHKA = W_M_O = W_SYNOLO_M_O  = 0 ;
    ΥΠΟΛΟΓΙΣΕ  W_ARITMOS = 1 ;

    ΕΠΑΝΕΛΑΒΕ

        ΥΠΟΛΟΓΙΣΕ  W_SYNOLO  = 0 , W_AA  = 1 ;

        ΕΠΑΝΕΛΑΒΕ

            ΕΑΝ ( W_AA < > W_ARITMOS ) ΤΟΤΕ

                ΕΑΝ ( ( W_ARITMOS ΥΠΟΛΟΙΠΟ ΔΙΑΙΡΕΣΗΣ W_AA ) = 0 ) ΤΟΤΕ

                    ΥΠΟΛΟΓΙΣΕ  W_SYNOLO  = W_SYNOLO + W_AA ;

                ΕΑΝ-ΤΕΛΟΣ ;

            ΕΑΝ-ΤΕΛΟΣ ;

            ΥΠΟΛΟΓΙΣΕ  W_AA  = W_AA + 1 ;

        ΜΕΧΡΙ (  ( W_AA = ( W_EWS_ARITMO ΔΙΑΙΡΕΣΕ ΜΕ 2 ) )
            // ένας αριθμός διαιρείται ακριβώς, τουλάχιστον μέχρι τα μισά του νούμερα, άρα είναι αρκετό να τον //
            // ελένξουμε μόνο, μέχρι την μέση των αριθμών, ώστε να έχουμε βελτίωση του χρόνου εκτέλεσης,     //
           'Η     ( W_SYNOLO > W_ARITMOS ) )
            // επίσης με την δεύτερη συνθήκη θα μπορούσαμε να έχουμε βελτίωση, εάν αφήναμε το πρόγραμμα  //
            // να τρέξει σε μεγαλύτερα όρια, για να ανακαλύψει τον επόμενο άγνωστο τέλειο αριθμό          //
        ΕΠΑΝΕΛΑΒΕ-ΤΕΛΟΣ ;

        ΕΑΝ ( W_SYNOLO = W_ARITMOS ) ΤΟΤΕ

            ΤΥΠΩΣΕ ( "βρέθηκε τέλειος αριθμός : ", W_ARITMOS ) ;

            ΥΠΟΛΟΓΙΣΕ  W_BRHKA  = W_BRHKA + 1 ;

            ΥΠΟΛΟΓΙΣΕ  W_SYNOLO_M_O = W_SYNOLO_M_O + W_ARITMOS ;

        ΕΑΝ-ΤΕΛΟΣ  ;

        ΥΠΟΛΟΓΙΣΕ  W_ARITMOS  = W_ARITMOS + 1 ;

    ΜΕΧΡΙ ( W_ARITMOS > W_EWS_ARITMO )
    ΕΠΑΝΕΛΑΒΕ-ΤΕΛΟΣ ;

    ΕΑΝ ( W_BRHKA > 0 ) ΤΟΤΕ                    // αποφυγή διαίρεσης με μηδέν //

        ΥΠΟΛΟΓΙΣΕ  W_M_O  = W_SYNOLO_M_O  /  W_BRHKA ;

        ΤΥΠΩΣΕ ( "ο μέσος όρος είναι : ", W_M_O ) ;

    ΑΛΛΙΩΣ

        ΤΥΠΩΣΕ ( "δεν βρέθηκε τέλειος αριθμός ! " ) ;

    ΕΑΝ-ΤΕΛΟΣ ;

ΤΕΛΟΣ-ΔΙΑΔΙΚΑΣΙΑΣ ;
```

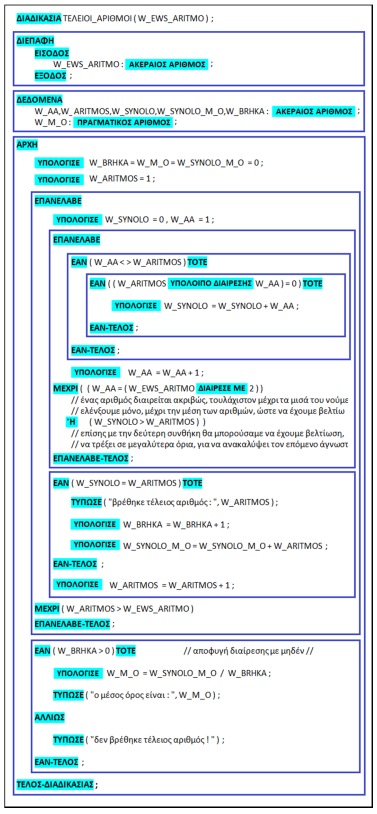
```
ΔΙΑΔΙΚΑΣΙΑ ΤΕΛΕΙΟΙ_ΑΡΙΘΜΟΙ ( W_EWS_ARITMO ) ;

ΔΙΕΠΑΦΗ
    ΕΙΣΟΔΟΣ
        W_EWS_ARITMO : ΑΚΕΡΑΙΟΣ ΑΡΙΘΜΟΣ ;
    ΕΞΟΔΟΣ ;

ΔΕΔΟΜΕΝΑ
    W_AA,W_ ARITMOS,W_SYNOLO,W_SYNOLO_M_O,W_BRHKA : ΑΚΕΡΑΙΟΣ ΑΡΙΘΜΟΣ ;
    W_M_O : ΠΡΑΓΜΑΤΙΚΟΣ ΑΡΙΘΜΟΣ ;

ΑΡΧΗ
    ΥΠΟΛΟΓΙΣΕ  W_BRHKA = W_M_O = W_SYNOLO_M_O  = 0 ;
    ΥΠΟΛΟΓΙΣΕ  W_ARITMOS = 1 ;
    ΕΠΑΝΕΛΑΒΕ
        ΥΠΟΛΟΓΙΣΕ  W_SYNOLO  = 0 , W_AA  = 1 ;
        ΕΠΑΝΕΛΑΒΕ
            ΕΑΝ ( W_AA < > W_ARITMOS ) ΤΟΤΕ
                ΕΑΝ ( ( W_ARITMOS ΥΠΟΛΟΙΠΟ ΔΙΑΙΡΕΣΗΣ W_AA ) = 0 ) ΤΟΤΕ
                    ΥΠΟΛΟΓΙΣΕ  W_SYNOLO  = W_SYNOLO + W_AA ;
                ΕΑΝ-ΤΕΛΟΣ ;
            ΕΑΝ-ΤΕΛΟΣ ;
            ΥΠΟΛΟΓΙΣΕ  W_AA  = W_AA + 1 ;
        ΜΕΧΡΙ (  ( W_AA = ( W_EWS_ARITMO ΔΙΑΙΡΕΣΕ ΜΕ 2 ) )
            // ένας αριθμός διαιρείται ακριβώς, τουλάχιστον μέχρι τα μισά του νούμερα, άρα είναι αρκετό να τον //
            // ελένξουμε μόνο, μέχρι την μέση των αριθμών, ώστε να έχουμε βελτίωση του χρόνου εκτέλεσης,     //
            'Η    ( W_SYNOLO > W_ARITMOS ) )
            // επίσης με την δεύτερη συνθήκη θα μπορούσαμε να έχουμε βελτίωση, εάν αφήναμε το πρόγραμμα //
            // να τρέξει σε μεγαλύτερα όρια, για να ανακαλύψει τον επόμενο άγνωστο τέλειο αριθμό          //
        ΕΠΑΝΕΛΑΒΕ-ΤΕΛΟΣ ;
        ΕΑΝ ( W_SYNOLO = W_ARITMOS ) ΤΟΤΕ
            ΤΥΠΩΣΕ ( "βρέθηκε τέλειος αριθμός : ", W_ARITMOS ) ;
            ΥΠΟΛΟΓΙΣΕ  W_BRHKA  = W_BRHKA + 1 ;
            ΥΠΟΛΟΓΙΣΕ  W_SYNOLO_M_O = W_SYNOLO_M_O + W_ARITMOS ;
        ΕΑΝ-ΤΕΛΟΣ  ;
        ΥΠΟΛΟΓΙΣΕ  W_ARITMOS  = W_ARITMOS + 1 ;
    ΜΕΧΡΙ ( W_ARITMOS > W_EWS_ARITMO )
    ΕΠΑΝΕΛΑΒΕ-ΤΕΛΟΣ ;
    ΕΑΝ ( W_BRHKA > 0 ) ΤΟΤΕ                    // αποφυγή διαίρεσης με μηδέν //
        ΥΠΟΛΟΓΙΣΕ  W_M_O  = W_SYNOLO_M_O  /  W_BRHKA ;
        ΤΥΠΩΣΕ ( "ο μέσος όρος είναι : ", W_M_O ) ;
    ΑΛΛΙΩΣ
        ΤΥΠΩΣΕ ( "δεν βρέθηκε τέλειος αριθμός ! " ) ;
    ΕΑΝ-ΤΕΛΟΣ ;
ΤΕΛΟΣ-ΔΙΑΔΙΚΑΣΙΑΣ ;
```

Ο ψευδοκώδικας για τους τέλειους αριθμούς εμφανίζεται δυο φορές, μια όπως θα έπρεπε να ήταν κανονικά και μια δεύτερη φορά, με μια γραφική οριοθέτηση των ενεργειών που προκαλεί η κάθε εντολή. Συγκεκριμένα, όταν μια εντολή βρίσκεται εσωτερικά μέσα σε κάποια άλλη, οριοθετείται η εμβέλεια της με ένα παραλληλόγραμμο ενώ ολόκληρο το σώμα των στοιχείων που την αποτελούν, τοποθετείται και λίγο δεξιότερα (συνήθως, 4 χαρακτήρες δεξιότερα). Επιπλέον, χρησιμοποιούνται ιδιαίτερα περιγραφικά ονόματα για όλα τα απαραίτητα «καπελάκια» που χρησιμοποιούνται στους υπολογισμούς και τοποθετούνται και σχολιασμοί.

Για να γίνει αντιληπτή η σημασία της συγγραφής του κώδικα με κανόνες, αλλά και ο λόγος της νοητής οριοθέτησης των εντολών που πρέπει να αναζητά ο προγραμματιστής σε κάθε πρόγραμμα, ακολουθεί μια συγκριτική παρουσίαση του ψευδοκώδικα για τους τέλειους αριθμούς.







Στην αριστερή πλευρά, ο κώδικας εμφανίζεται με τις εντολές να ξεκινούν, όλες χωρίς απολύτως καμία διευθέτηση, από την αριστερότερη άκρη κάθε γραμμής. Στην μεσαία πλευρά

ο ίδιος κώδικας παρουσιάζεται βελτιωμένος, καθώς το σώμα της κάθε εντολής (δηλαδή, τα στοιχεία που την αποτελούν) έχουν μετακινηθεί δεξιότερα και μόνο στην περίπτωση που αυτές φιλοξενούνται «μέσα» σε κάποια άλλη εντολή, ενώ έχουν προστεθεί και σχόλια. Τέλος, στην δεξιά πλευρά εμφανίζεται πάλι ο ίδιος κώδικας μαζί με τις μετακινήσεις, αλλά επιπλέον οι εντολές, έχουν οριοθετηθεί νοητά μέσα σε κατάλληλα παραλληλόγραμμα με το ένα περικλείεται μέσα σε κάθε άλλο και έτσι να αναδεικνύονται τα όρια λειτουργίας της κάθε εντολής και ο κώδικας να είναι κατά πολύ περισσότερο ευανάγνωστος.

Είναι προφανές ότι στην δεξιά αποτύπωση η ανάγνωση του κώδικα είναι ευκολότερη και καλούνται οι εκπαιδευόμενοι να υιοθετήσουν αυτήν την κρίσιμη προγραμματιστική συμπεριφορά. Οριοθετείτε στο μυαλό σας τα προγράμματα σας, σε κομμάτια κώδικα, που θα τα φαντάζεστε να βρίσκονται μέσα σε νοητά παραλληλόγραμμα. Μετακινείτε τα σώματα των εντολών σας δεξιότερα στην περίπτωση που αυτές φιλοξενούνται μέσα σε άλλες εντολές. Υιοθετείτε περιγραφικές ονομασίες στα «καπελάκια» σας (τα ονόματα των μεταβλητών σας) και ενσωματώνετε κατάλληλα σχόλια.

Αυτή η κουλτούρα, της συγγραφής προγραμμάτων, δομημένων, με τάξη, κανόνες, περιγραφικά ονόματα και κατάλληλους σχολιασμούς ονομάζεται, *δομημένος προγραμματισμός*.

Σύνοψη κεφαλαίου

Στο τέταρτο κεφάλαιο περιγράφονται αναλυτικά οι βασικότερες μορφές με τις οποίες μπορεί να εμφανιστεί στον κόσμο της πληροφορικής και του προγραμματισμού, ένας αλγόριθμος. Δίνεται έμφαση στην αναπαράσταση αλγορίθμων με ολοκληρωμένα παραδείγματα με χρήση ψευδογλώσσας και χρήση λογικών διαγραμμάτων. Ειδικότερα, παρουσιάζεται βήμα προς βήμα, η κατασκευή ενός αλγορίθμου (για την δομή επανάληψης υπό συνθήκη), για κάθε μια από τις δυο χρησιμοποιούμενες μορφές αποτύπωσης και επισυνάπτονται επιπλέον σύνθετα παραδείγματα για περεταίρω μελέτη.

Ερωτήσεις αξιολόγησης

Ερώτηση-1: Στον προγραμματισμό, ένας αλγόριθμός με πόσες μορφές μπορεί να εμφανιστεί;

Ερώτηση-2: Περιγράψτε 3 βασικά σύμβολα στα λογικά διαγράμματα;

Ερώτηση-3: Με ποιο σύμβολο στα λογικά διαγράμματα δηλώνουμε την απόφαση;

Ερώτηση-4: Περιγράψτε μια απλή δομή ακολουθίας και να δοθεί το διάγραμμα ροής της;

Ερώτηση-5: Η έννοια, «επανάληψη υπό συνθήκη» στον προγραμματισμό, τι ακριβώς σημαίνει, περιγράψτε το;

Ερώτηση-6: Τι είναι ο ψευδοκώδικας;

Ερώτηση-7: Περιγράψτε, τι κάνει σε ψευδογλώσσα, η δομή, **ΕΑΝ**… **ΤΟΤΕ**… **ΑΛΛΙΩΣ**… **ΕΑΝ-ΤΕΛΟΣ**;

Ερώτηση-8: Περιγράψτε, τι κάνει σε ψευδογλώσσα, η δομή, **ΕΠΑΝΕΛΑΒΕ**… **ΜΕΧΡΙ**… **ΕΠΑΝΕΛΑΒΕ-ΤΕΛΟΣ**;

Ερώτηση-9: Για ποιον λόγο πρέπει να χρησιμοποιούμε περιγραφικά ονόματα στις μεταβλητές μας;

Ερώτηση-10: Για ποιον λόγο πρέπει να διευθετούμε δεξιότερα την δομή μιας εντολής, που βρίσκεται να ενσωματώνεται («φωλιάζει») μέσα σε μια άλλη εντολή;

Απαντήσεις ερωτήσεων αξιολόγησης

Απάντηση-1: Συμβουλευτείτε τις εισαγωγικές παρατηρήσεις του κεφαλαίου.

Απάντηση-2: Συμβουλευτείτε την ενότητα 4.3. του κεφαλαίου.

Απάντηση-3: Συμβουλευτείτε την ενότητα 4.3. του κεφαλαίου.

Απάντηση-4: Συμβουλευτείτε την ενότητα 4.4. του κεφαλαίου.

Απάντηση-5: Συμβουλευτείτε την ενότητα 4.4. του κεφαλαίου.

Απάντηση-6: Συμβουλευτείτε την ενότητα 4.6. του κεφαλαίου.

Απάντηση-7: Συμβουλευτείτε την ενότητα 4.6. του κεφαλαίου.

Απάντηση-8: Συμβουλευτείτε την ενότητα 4.6. του κεφαλαίου.

Απάντηση-9: Συμβουλευτείτε την ενότητα 4.7. του κεφαλαίου.

Απάντηση-10: Συμβουλευτείτε την ενότητα 4.7. του κεφαλαίου.

5. ΓΛΩΣΣΕΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Σκοπός και επιμέρους στόχοι

Με το πέμπτο κεφάλαιο ο εκπαιδευόμενος αντιλαμβάνεται τους λόγους για τους οποίους χρειαζόμαστε μια γλώσσα προγραμματισμού για να επιτύχουμε τον έλεγχο μιας μηχανής ή ενός σύγχρονου υπολογιστικού συστήματος. Στο κεφάλαιο περιγράφεται η έννοια του προγραμματισμού σε γλώσσες χαμηλού επιπέδου και σε γλώσσες υψηλού επιπέδου με παραδείγματα και κατάλληλες επεξηγήσεις.

Προσδοκώμενα αποτελέσματα

Με την μελέτη του πέμπτου κεφαλαίου οι εκπαιδευόμενοι θα μπορούν,

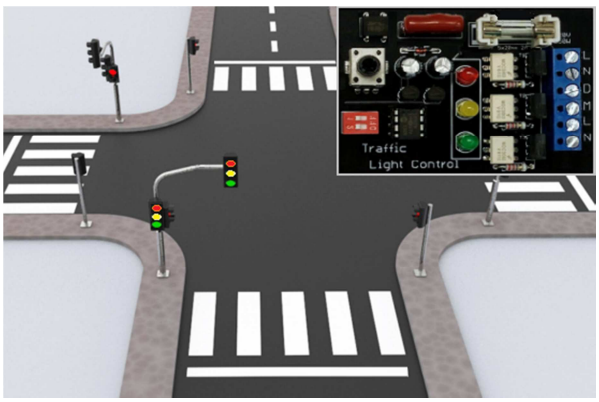
- να διατυπώνουν τι είναι μια γλώσσα προγραμματισμού,
- να αναφέρουν γιατί χρειάζονται οι γλώσσες προγραμματισμού και τι επιτυγχάνουμε με αυτές,
- να περιγράφουν τι είναι μια γλώσσα προγραμματισμού χαμηλού επιπέδου,
- να περιγράφουν τι είναι μια γλώσσα προγραμματισμού υψηλού επιπέδου,
- να αντιλαμβάνονται ποιες είναι οι διαφορές μεταξύ γλωσσών, χαμηλού επιπέδου και γλωσσών, υψηλού επιπέδου,
- να περιγράφουν τα πλεονεκτήματα και τα μειονεκτήματα της κάθε κατηγορίας,
- να συνειδητοποιούν τα οφέλη που προκύπτουν από την μαζική χρήση των γλωσσών προγραμματισμού υψηλού επιπέδου.

Έννοιες – Λέξεις Κλειδιά

Γλώσσες προσανατολισμένες προς την μηχανή, Γλώσσες προσανατολισμένες προς τον άνθρωπο, Λεξιλόγια γλωσσών, Γλώσσες Assembly, Γλώσσες χαμηλού επιπέδου, Γλώσσες υψηλού επιπέδου, Συμβολικές γλώσσες, Μεταφερσιμότητα προγραμμάτων, Διαδικασιακές γλώσσες, Αντικειμενοστραφείς γλώσσες, Συναρτησιακές γλώσσες.

Εισαγωγικές Παρατηρήσεις

Γλώσσα προγραμματισμού ονομάζουμε, κάθε σύνολο άυλων κανόνων και εντολών που μπορούν να χρησιμοποιηθούν για τον έλεγχο λειτουργίας μιας μηχανής. Όλες οι μηχανές που μας περιβάλουν διαθέτουν τρόπους με τους οποίους προγραμματίζουμε την συμπεριφορά τους. Οι περισσότεροι άνθρωποι μετατρέπονται σε προγραμματιστές κάθε βράδυ, όταν προσπαθούν να προγραμματίσουν το ρολόι τους για να χτυπήσει σωστά το επόμενο πρωινό. Υπάρχουν μηχανές, όπως τα ρολόγια, οι οποίες μπορούν να προγραμματιστούν εύκολα με μερικές απλές οδηγίες και μηχανές οι οποίες προγραμματίζονται με σύνθετες εντολές. Οι γλώσσες προγραμματισμού των μηχανών, έχουν επινοηθεί σε διάφορες παραλλαγές, με ρεπερτόρια τα οποία μπορούν να περιλαμβάνουν, από κάποια πλήκτρα μέχρι σύνθετους συνδυασμούς από σύμβολα, αριθμούς, γράμματα, λέξεις, εντολές, φράσεις και προτάσεις. Οι σύγχρονοι ηλεκτρονικοί υπολογιστές προγραμματίζονται με σύνθετες γλώσσες προγραμματισμού.



5.1. Γλώσσες Προγραμματισμού Ηλεκτρονικών Υπολογιστών

Η επικοινωνία ανθρώπων και υπολογιστών είναι ένα ανοικτό πρόβλημα το οποίο μπορεί να λυθεί με δύο τρόπους, ή ο άνθρωπος θα μάθει τη γλώσσα του υπολογιστή ή ο υπολογιστής θα μάθει την ανθρώπινη γλώσσα. Σήμερα η τεχνολογία και η επιστήμη έχει επιτρέψει σε πολλές μηχανές να μπορούν να προσεγγίζουν οι ίδιες ακόμα και τον ανθρώπινο λόγο και να αντιλαμβάνονται οδηγίες που τους δίνονται στην φυσική γλώσσα. Οι ηλεκτρονικοί υπολογιστές δεν έχουν όλοι τους, μια κοινή γλώσσα προγραμματισμού. Υπάρχουν πολλές διαφορετικές γλώσσες και επίσης πολλές διαφορετικές κατηγορίες γλωσσών. Όμως όλες οι

γλώσσες προγραμματισμού έχουν κοινά σημεία και η καλή γνώση μιας και μόνο γλώσσας δίνει την δυνατότητα σε κάθε φιλόδοξο προγραμματιστή να μάθει εύκολα και οποιαδήποτε άλλη γλώσσα προγραμματισμού. Γενικότερα, μπορούμε να θεωρήσουμε ότι υπάρχουν δύο γενικές κατηγορίες γλωσσών προγραμματισμού, οι γλώσσες που είναι προσανατολισμένες προς την μηχανή και οι γλώσσες που είναι προσανατολισμένες προς τον άνθρωπο.

5.2. Γλώσσες Προγραμματισμού Η/Υ, προσανατολισμένες προς την Μηχανή

Οι γλώσσες που ανήκουν στη κατηγορία αυτή, διαθέτουν λεξιλόγια με εντολές, συνήθως περιορισμένες σε αριθμό και ειδικές για κάθε τύπο μηχανής. Στις περιπτώσεις αυτές ο προγραμματιστής είναι αυτός που προσεγγίζει τον τρόπο λειτουργίας της μηχανής, καθώς μαθαίνει τους δικούς της ιδιωματισμούς. Για παράδειγμα, η ρύθμιση μιας διασταύρωσης οδικής κυκλοφορίας είναι μια περίπτωση προγραμματισμού μιας μηχανής σε μια γλώσσα χαμηλού επιπέδου ειδικά προσανατολισμένη προς την συγκεκριμένη μηχανή. Ο τεχνικός δηλαδή, που προγραμματίζει την διασταύρωση, εισάγει στην μηχανισμό ειδικές εντολές οι οποίες δεν μοιάζουν να ανήκουν σε μια γλώσσα κοινής ανθρώπινης επικοινωνίας. Για παράδειγμα οι δηλώσεις, STA, PUSH, POP, JMP και HLT είναι εντολές που ανήκουν σε μια γλώσσα που ονομάζεται *Assembly* και είναι προσανατολισμένη προς τις μηχανές. Γλώσσα Assembly, μπορεί να διαθέτει μια διασταύρωση οδικής κυκλοφορίας αλλά και ο πυρήνας (ο επεξεργαστής) ενός κινητού τηλεφώνου ή ενός σύγχρονου ηλεκτρονικού υπολογιστή. Οι γλώσσες Assembly χρησιμοποιούνται και από πολύ μεγαλύτερες μηχανές αλλά και πάλι το ρεπερτόριο των εντολών που διαθέτουν, είναι συγκεκριμένο και προσανατολισμένο ειδικά για τους πόρους και το κάθε στοιχειώδες υλικό στοιχείο (Hardware) της συγκεκριμένης μηχανής. Οι γλώσσες Assembly είναι δύσκολες γλώσσες με ειδικές εντολές, διαφορετικές για κάθε μηχανή, που δεν μοιάζουν με τις κοινές γλώσσες ανθρώπινης επικοινωνίας, απαιτούν την περιγραφή και του παραμικρού βήματος, για να δηλωθεί πλήρως μια διεργασία και γενικά ο προγραμματιστής πρέπει να φροντίζει για το οτιδήποτε. Παρόλα αυτά οι γλώσσες αυτές παράγουν πολύ γρήγορα και πολύ συμπαγή προγράμματα. Η εκμάθηση μιας γλώσσας αυτής της κατηγορίας συνδέεται άμεσα με την εκμάθηση της εσωτερικής αρχιτεκτονικής της κάθε μηχανής και των ιδιαίτερων χαρακτηριστικών της. Για όλους αυτούς τους λόγους, οι γλώσσες

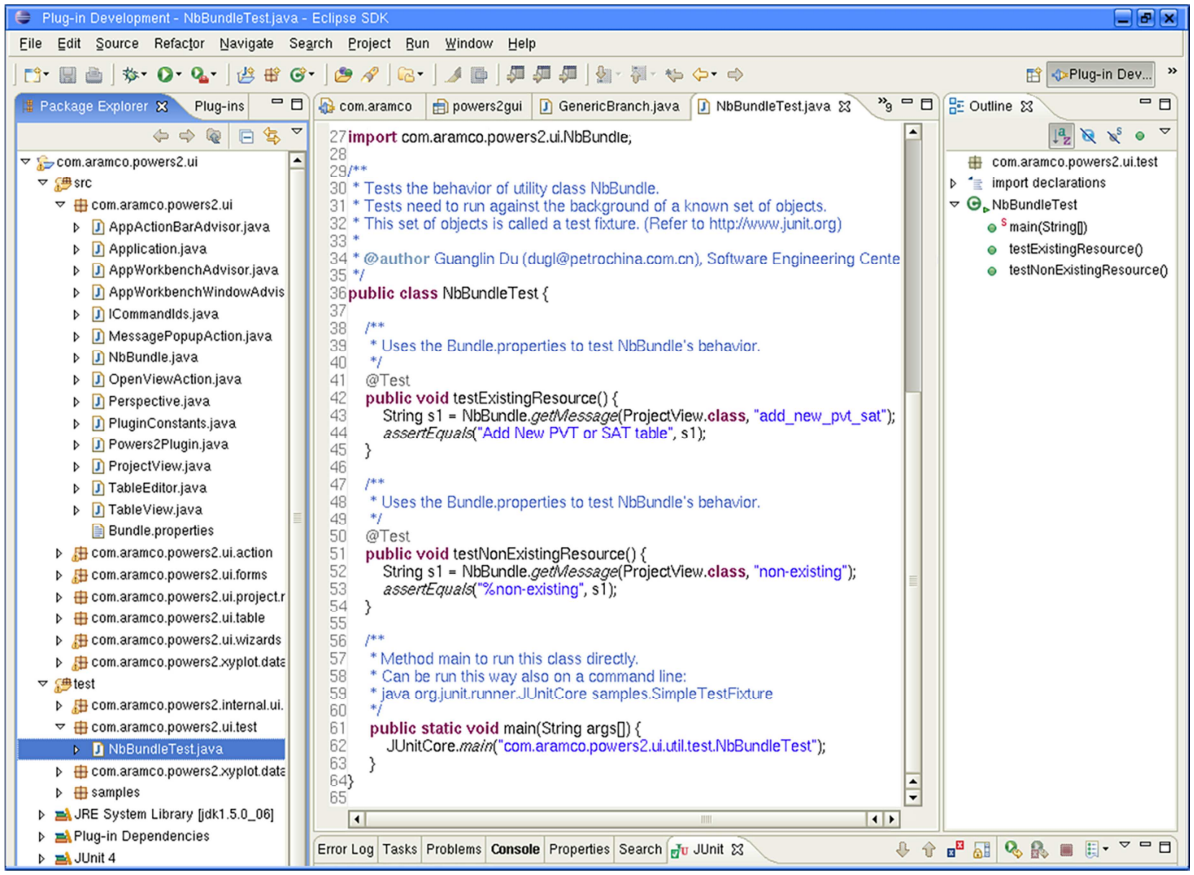
αυτές ονομάζονται γλώσσες «*χαμηλού επιπέδου*», «*συμβολικές*» ή γλώσσες προσανατολισμένες προς τις μηχανές.

5.3. Γλώσσες Προγραμματισμού Η/Υ, προσανατολισμένες προς τον Άνθρωπο

Οι γλώσσες, οι προσανατολισμένες προς τον άνθρωπο, έχουν δημιουργηθεί με βασικό στόχο να έρθουν κοντά στην ανθρώπινη φύση και την ανθρώπινη λογική. Διαθέτουν πλούσια λεξιλόγια με εντολές δανεισμένες από το αγγλικό λεξιλόγιο. Στις περιπτώσεις των συγκεκριμένων γλωσσών, ο προγραμματιστής χρησιμοποιεί λέξεις της καθημερινής ομιλίας και η μηχανή είναι αυτή που από μόνη της αναλαμβάνει να τις μετατρέψει σε εντολές που θα καθορίσουν τη συμπεριφορά της. Ο προγραμματιστής έτσι εστιάζει στην περιγραφή της λύσης του πραγματικού προβλήματος και δεν ασχολείται με τους πόρους και τα στοιχειώδη υλικά στοιχεία (Hardware) της μηχανής. Αυτό το αναλαμβάνει η ίδια η γλώσσα προγραμματισμού. Επιπλέον, προσφέρουν μεταφερσιμότητα καθώς, εάν ο ίδιος προγραμματιστικός κώδικας τοποθετηθεί σε μια άλλη παρόμοια μηχανή θα λειτουργήσει χωρίς καμία απολύτως αλλαγή.

Συνολικά, οι γλώσσες της κατηγορίας έχουν ελαττώσει τον χρόνο και το κόστος παραγωγής νέων προγραμμάτων και συστημάτων, αφού πολλοί προγραμματιστές μπορούν εύκολα να εκπαιδευτούν μέσα σε σύντομο χρονικό διάστημα, ώστε να αναπτύσσουν άμεσα πολλά προγράμματα που θα χρησιμοποιηθούν από περισσότερες υπολογιστικές μηχανές. Υπάρχουν πολλές γλώσσες προγραμματισμού, προσανατολισμένες προς τον άνθρωπο, άλλες κατάλληλες για ειδικές εφαρμογές, άλλες για την ανάπτυξη εμπορικών εφαρμογών, άλλες για την εκπαίδευση, για επιστημονικούς σκοπούς και άλλες για γενική χρήση.

Από την δεκαετία του 60 που άρχισαν να παράγονται οι γλώσσες προγραμματισμού, έχουν δημιουργηθεί πολλές εκατοντάδες διαφορετικές γλώσσες με σημαντικότερες από αυτές να θεωρούνται οι, ADA, ALGOL, ALICE, B, BASIC, C, C++, C#, COBOL, DART, JAVA, JAVASCRIPT, LISP, LOGO, PASCAL, PERL, PHP, PL/1, PROLOG, PYHTON, RUBY, SCRATCH, η SQL, αλλά και πολλές άλλες. Η μεγάλη πλειοψηφία όλων αυτών των γλωσσών ονομάζονται *ακολουθιακές*, *διαδικασιακές* ή *διαδικαστικές* (procedural), καθώς είναι σχεδιασμένες ειδικά για την υλοποίηση αλγοριθμικών κατασκευών. Οι γλώσσες αυτές μπορούν επίσης, δευτερευόντως να ταξινομηθούν, σε *αντικειμενοστραφείς γλώσσες* (object-oriented languages), *συναρτησιακές γλώσσες* (functional languages), *μη διαδικασιακές γλώσσες* (nonprocedural languages) και *γλώσσες ερωτήσεων/απαντήσεων* (query languages). Όλες οι γλώσσες αυτής της κατηγορίας, ονομάζονται γλώσσες «*υψηλού επιπέδου*», ή γλώσσες προσανατολισμένες προς τον άνθρωπο.



Ολοκληρωμένο περιβάλλον ανάπτυξης (I.D.E.) γλώσσας προγραμματισμού.

Σύνοψη κεφαλαίου

Στο κεφάλαιο πέντε γίνεται μια συνοπτική παρουσίαση των μεθόδων που χρησιμοποιούνται για τον έλεγχο των μηχανών και ειδικά των υπολογιστικών μηχανών και ονομάζονται *γλώσσες προγραμματισμού υπολογιστών*. Επεξηγούνται οι λόγοι για τους οποίους οι γλώσσες προγραμματισμού είναι τα απαραίτητα στοιχεία ελέγχου και επικοινωνίας των ανθρώπων και των μηχανών για τον έλεγχο και την καθοδήγηση τους. Παρουσιάζεται μια ποικιλία γλωσσών προγραμματισμού και αναλύεται ειδικότερα η διαφορά των γλωσσών χαμηλού επιπέδου και των γλωσσών υψηλού επιπέδου.

Ερωτήσεις αξιολόγησης

Ερώτηση-1: Περιγράψτε, τι ονομάζουμε *γλώσσα προγραμματισμού υπολογιστών*;

Ερώτηση-2: Ποιες είναι οι δυο βασικές κατηγορίες γλωσσών προγραμματισμού υπολογιστών;

Ερώτηση-3: Περιγράψτε, τι είναι οι γλώσσες Assembly;

Ερώτηση-4: Ποιες γλώσσες προγραμματισμού ονομάζονται *γλώσσες χαμηλού επιπέδου* και γιατί;

Ερώτηση-5: Ποιες γλώσσες προγραμματισμού ονομάζονται *γλώσσες υψηλού επιπέδου* και γιατί;

Ερώτηση-6: Περιγράψτε γιατί χρειαζόμαστε τις γλώσσες προγραμματισμού, τι επιτυγχάνουμε με αυτές;

Ερώτηση-7: Είναι σωστό ότι, οι γλώσσες προγραμματισμού υψηλού επιπέδου είναι προσανατολισμένες ειδικά για συγκεκριμένες μηχανές;

Ερώτηση-8: Είναι σωστό ότι, οι γλώσσες προγραμματισμού χαμηλού επιπέδου απαιτούν την εκμάθηση εντολών από την εσωτερική αρχιτεκτονική των μηχανών που απευθύνονται;

Ερώτηση-9: Είναι σωστό ότι, οι γλώσσες προγραμματισμού υψηλού επιπέδου προσφέρουν μεταφερσιμότητα μεταξύ μηχανών;

Ερώτηση-10: Περιγράψτε 3 πλεονεκτήματα που προσφέρουν οι γλώσσες προγραμματισμού υψηλού επιπέδου;

Απαντήσεις ερωτήσεων αξιολόγησης

Απάντηση-1: Συμβουλευτείτε τις εισαγωγικές παρατηρήσεις του κεφαλαίου.

Απάντηση-2: Συμβουλευτείτε την ενότητα 5.1. του κεφαλαίου.

Απάντηση-3: Συμβουλευτείτε την ενότητα 5.2. του κεφαλαίου.

Απάντηση-4: Συμβουλευτείτε την ενότητα 5.2. του κεφαλαίου.

Απάντηση-5: Συμβουλευτείτε την ενότητα 5.3. του κεφαλαίου.

Απάντηση-6: Συμβουλευτείτε την ενότητα 5.3. του κεφαλαίου.

Απάντηση-7: "Όχι, είναι λάθος.

Απάντηση-8: Ναι, είναι σωστό.

Απάντηση-9: Ναι, είναι σωστό.

Απάντηση-10: Συμβουλευτείτε την ενότητα 5.3. του κεφαλαίου.

6. ΑΛΓΟΡΙΘΜΟΙ ΚΑΙ ΓΛΩΣΣΕΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Σκοπός και επιμέρους στόχοι

Με το έκτο κεφάλαιο οι εκπαιδευόμενοι μεταφέρονται στον κόσμο των πραγματικών γλωσσών προγραμματισμού. Αρχικά περιγράφονται, όλα τα απαραίτητα δομικά στοιχεία που πρέπει να αναζητά κάθε νέος προγραμματιστής για να καθίσταται ικανός να αντιλαμβάνεται και να αφομοιώνει, τα βασικά γνωστικά αντικείμενα μιας γλώσσας. Στην συνέχεια, μελετώνται οι βασικές αλγοριθμικές δομές που χρησιμοποιούνται από όλες τις γλώσσες προγραμματισμού. Το κεφάλαιο καθοδηγεί τον εκπαιδευόμενο, αξιοποιώντας την προηγούμενη αποκτηθείσα γνώση του, από τα λογικά διαγράμματα και την ψευδογλώσσα.

Προσδοκώμενα αποτελέσματα

Με την μελέτη του έκτου κεφαλαίου οι εκπαιδευόμενοι θα μπορούν,

- να ορίζουν τα βασικά συστατικά στοιχεία μιας γλώσσας προγραμματισμού,
- να περιγράψουν τι είναι το αλφάβητο και τι είναι το λεξιλόγιο μιας γλώσσας,
- να αντιλαμβάνονται την αναγκαιότητα της ορθής σύνταξης των εντολών της γλώσσας,
- να διακρίνουν σε μια γλώσσα προγραμματισμού την διαφορά της σημασιολογίας από το συντακτικό της,
- να αναγνωρίσουν τους τελεστές και την σημασία τους,
- να διαχωρίζουν τους διαφορετικούς τύπους των δεδομένων,
- να καταλαβαίνουν ποια περιεχόμενα μπορούν να αποθηκευτούν σε κάθε διαφορετικό τύπο δεδομένων,
- να σκεφτούν ότι μπορούν να συνθέσουν την ίδια εντολή με διαφορετικούς τρόπους επιτυγχάνοντας διαφορετικά αποτελέσματα,
- να εντοπίζουν εντολές, με παρόμοια αποτελεσματικότητα μεταξύ διαφορετικών γλωσσών προγραμματισμού,
- να περιγράψουν ποιες είναι οι βασικές αλγοριθμικές δομές στις γλώσσες προγραμματισμού,

- να καταστρώσουν, με διάγραμμα ή σε ψευδογλώσσα, κάθε μια από τις βασικές αλγοριθμικές δομές,
- να αντιλαμβάνονται πως λειτουργεί κάθε αλγοριθμική δομή και ποια είναι τα απαραίτητα στοιχεία που αποτελούν την δομή της,
- να περιγράφουν μεθοδολογίες για την δημιουργία κομψών και ευανάγνωστων προγραμμάτων.

Έννοιες – Λέξεις Κλειδιά

Αλφάβητο γλώσσας, Γραμματική και συντακτικό γλώσσας, Σημασιολογία γλώσσας, Λέξεις κλειδιά (keywords), Δεσμευμένες λέξεις (reserved words), Αριθμητικοί τελεστές, Συγκριτικοί τελεστές, Λογικοί τελεστές, Προτεραιότητες τελεστών, Μεταβλητή (variable), Όνομα μεταβλητής, Περιεχόμενο μεταβλητής, Μέγεθος μεταβλητής, Τύπος μεταβλητής, Σταθερές (constants), Αλγοριθμικές δομές επιλογής, Αλγοριθμικές δομές επανάληψης.

Εισαγωγικές Παρατηρήσεις

Στα προηγούμενα κεφάλαια η μελέτη των μεθόδων υλοποίησης αλγορίθμων, περιελάμβανε τα λογικά διαγράμματα και τις ψευδογλώσσες. Όμως τα λογικά διαγράμματα και οι ψευδοκώδικες δεν αναγνωρίζονται από τις μηχανές και χρησιμοποιούνται κυρίως για την αποτύπωση, την περιγραφή και την κατανόηση των προβλημάτων μεταξύ των ανθρώπων.

Οι τεχνολογίες των γλωσσών προγραμματισμού είναι αυτές οι οποίες μπορούν πραγματικά να ορίζουν την συμπεριφορά μιας μηχανής. Οι γλώσσες προγραμματισμού, είναι τεχνητές γλώσσες, οι οποίες στα όρια βασικών γλωσσολογικών εννοιών, μπορούν εύκολα να αφομοιωθούν από τους ανθρώπους προγραμματιστές και να χρησιμοποιηθούν για να δημιουργηθούν συλλογές εντολών που θα γίνονται κατανοητές από μια μηχανή. Ένα πρόγραμμα δηλαδή, είναι ένα σύνολο από συγκεκριμένες τεχνικές οδηγίες που δεν επιδέχονται παρερμηνεία, δεν απαιτούν «σκέψη» ή «κρίση» από την μηχανή και θα εκτελεστούν εντελώς μηχανικά από τον ηλεκτρονικό υπολογιστή. Οι γλώσσες προγραμματισμού όπως και οι φυσικές γλώσσες εξελίσσονται συνεχώς, με νέους κανόνες και δυνατότητες.

Οι κατασκευαστές των γλωσσών προγραμματισμού φροντίζουν να εξελίσσουν με νέες εκδόσεις τις κατασκευές τους, διορθώνοντας αδυναμίες και προσθέτοντας ικανότητες. Ποιος όμως θα ήταν ένας ιδανικός τρόπος, με τον οποίο κάθε νέος προγραμματιστής θα μπορούσε να προσεγγίζει, το φάσμα της ικανής γνώσης μιας νέας γλώσσας προγραμματισμού. Η γνώση μιας σύγχρονης γλώσσας προγραμματισμού μπορεί να οριοθετηθεί ανάμεσα σε πολλά διαφορετικά γνωστικά πεδία, τα επόμενα όμως στοιχεία πρέπει να θεωρούνται απολύτως απαραίτητα γνωστικά αντικείμενα και να προσεγγίζονται οπωσδήποτε κατά την αρχική προσπάθεια εκμάθησης μιας γλώσσας προγραμματισμού.

- το αλφάβητο,
- το λεξιλόγιο (οι δεσμευμένες λέξεις),
- η γραμματική,
- το συντακτικό,
- η σημασιολογία,
- οι τελεστές,
- οι μεταβλητές,
- οι τύποι των δεδομένων,
- οι βασικές αλγοριθμικές δομές.

6.1. Το Αλφάβητο

Αλφάβητο μιας γλώσσας είναι το σύνολο των στοιχείων που χρησιμοποιείται από την γλώσσα. Για παράδειγμα η ελληνική γλώσσα διαθέτει 24 βασικά στοιχεία. Όμως τα στοιχεία αυτά είναι πολύ περισσότερα καθώς θα πρέπει να λάβουμε υπόψη και τα πεζά αλλά και όλους τους ειδικούς χαρακτήρες (+ - * / = ^ : ; % # κ.λπ.). Έτσι και το αλφάβητο μιας γλώσσας προγραμματισμού αποτελείται συνήθως, από τα γράμματα του λατινικού και του ελληνικού αλφαβήτου, τα αριθμητικά ψηφία αλλά και τα ειδικά σύμβολα.

- κεφαλαία γράμματα ελληνικού αλφαβήτου (Α-Ω),
- πεζά γράμματα ελληνικού αλφαβήτου (α-ω),

- πεζά γράμματα ελληνικού αλφαβήτου με τόνους (α-ω),
- κεφαλαία γράμματα λατινικού αλφαβήτου (A-Z),
- πεζά γράμματα λατινικού αλφαβήτου (a-z),
- αριθμητικά ψηφία (0-9),
- ειδικά σύμβολα (+ - * / = ^ # . , ‘ % “ ! &),
- τον κενό χαρακτήρα.

6.2. Το Λεξιλόγιο (οι Δεσμευμένες Λέξεις)

Οι γλώσσες προγραμματισμού των υπολογιστών, όπως άλλωστε και οι ανθρώπινες γλώσσες, ορίζονται από ένα συγκεκριμένο σύνολο λέξεων, οι οποίες δημιουργούνται σαν υποσύνολα συγκεκριμένων ακολουθιών από τα αποδεκτά γράμματα του αλφαβήτου της γλώσσας. Για παράδειγμα, στην αγγλική γλώσσα, η λέξη ADDRESS είναι αποδεκτή ενώ η λέξη ARDDDESSS δεν είναι. Αντίστοιχα, κάθε γλώσσα προγραμματισμού, διαθέτει συγκριμένες λέξεις κλειδιά τις οποίες αναγνωρίζει και ονομάζονται, *keywords*, «*εργοστασιακές λέξεις*», *δεσμευμένες λέξεις,* ή *reserved words*. Για παράδειγμα, σε πολλές γλώσσες προγραμματισμού, η λέξη SWITCH είναι αποδεκτή, ενώ η λέξη SWWIITTT δεν είναι. Οι κοινές δεσμευμένες λέξεις ανάμεσα σε διαφορετικές γλώσσες προγραμματισμού είναι μια συνηθισμένη κατάσταση, ενώ σε άλλες περιπτώσεις η ίδια λειτουργία επιτυγχάνεται από συνώνυμες λέξεις (για παράδειγμα SWITCH και EVALUATE).

Aristotle Certification Training & Assessment
Τεχνοβλαστός Αριστοτελείου Πανεπιστημίου Θεσσαλονίκης

ενδεικτικό λεξιλόγιο γλώσσας C

auto	long
break	register
case	reurn
char	short
const	signed
continue	sizeof
default	static
do	struct
double	
else	
enum	
extern	
float	
for	
goto	
if	
int	

ενδεικτικό λεξιλόγιο γλώσσας C++

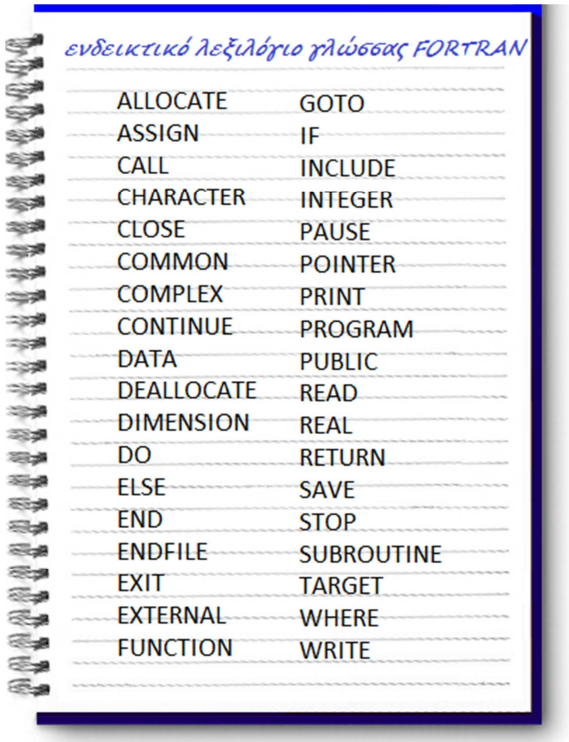
alignof	goto
auto	if
bool	int
break	long
case	new
char	operator
class	private
const	protected
continue	public
default	register
delete	return
do	static
double	switch
else	this
export	true
float	using
for	while

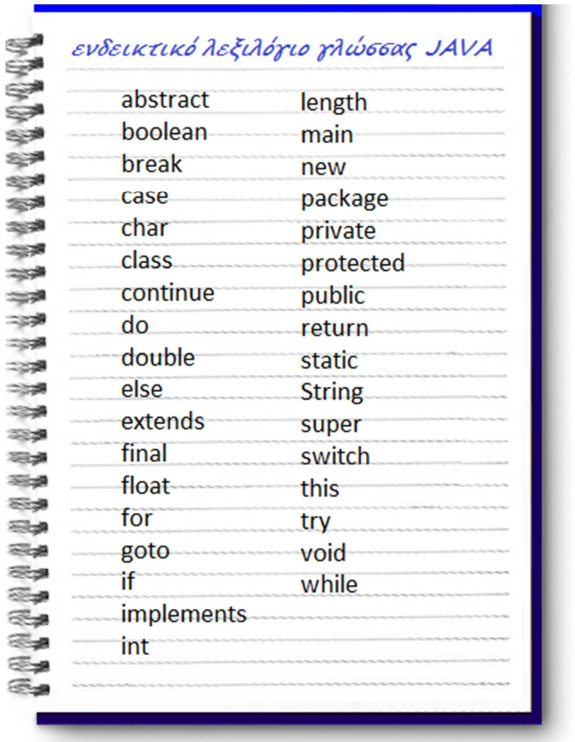
ενδεικτικό λεξιλόγιο γλώσσας C#

abstract	operator
break	private
case	protected
class	public
const	return
continue	short
decimal	static
dafault	string
do	struct
dynamic	switch
else	this
false	true
for	typeof
goto	using
if	void
int	while
lock	
new	

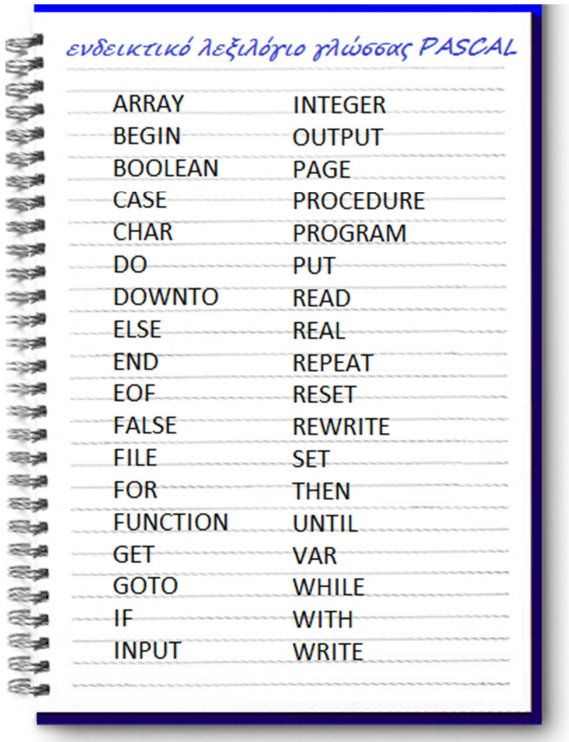
ενδεικτικό λεξιλόγιο γλώσσας COBOL

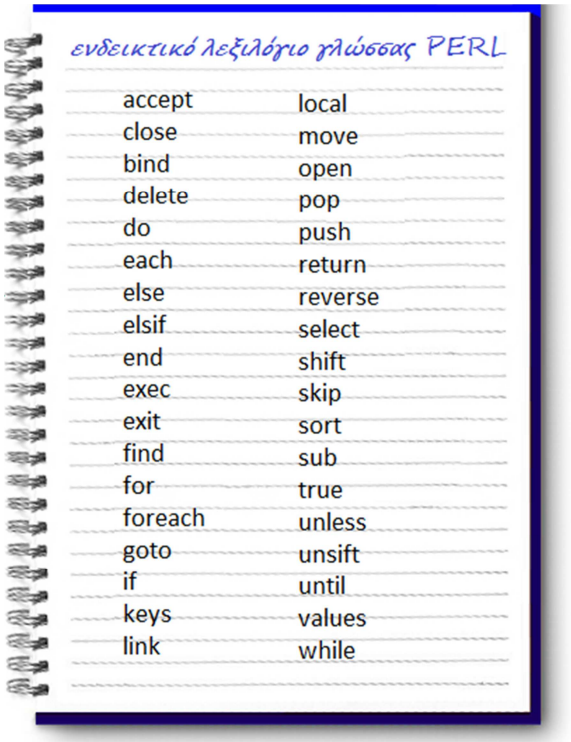
ACCEPT	MOVE
AFTER	PERFORM
ALL	PIC
BEFORE	READ
BLOCK	REWRITE
CALL	RUN
CANCEL	SEARCH
CLOSE	SECTION
COMPUTE	SORT
CONTINUE	START
COPY	STOP
DISPLAY	STRING
END	SUBTRACT
EVALUATE	THEN
EXIT	THRU
FOR	WHEN
GOTO	WITH
IF	WRITE

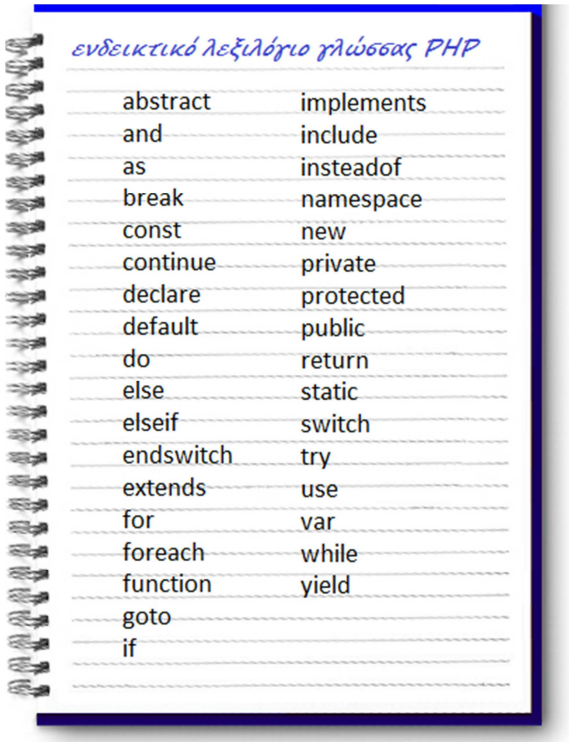


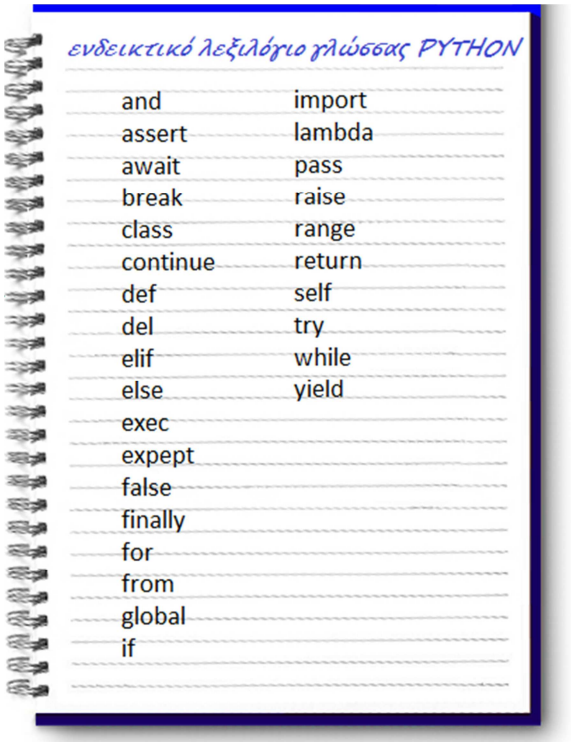


Τα λεξιλόγια των γλωσσών προγραμματισμού εξελίσσονται και αναβαθμίζονται συνεχώς. Οι ίδιοι οι κατασκευαστές ωθούν τις τεχνολογίες των γλωσσών προς νεώτερες και ισχυρότερες εκδόσεις. Τα νέα λεξιλόγια περιλαμβάνουν βελτιστοποιήσεις, διορθώσεις σφαλμάτων, περισσότερες αυτοματοποιήσεις και ενσωμάτωση μεταβολών που οφείλονται στις εξελίξεις των λειτουργικών συστημάτων (Windows, Mac, Android, κ.λπ.) και των υλικών μέσων (επεξεργαστές, μνήμες, δίσκοι, οθόνες, εκτυπωτές, κ.λπ.).









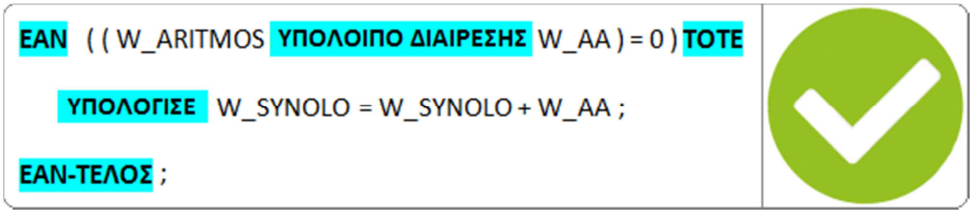
6.3. Η Γραμματική

Η γραμματική αφορά το τυπικό σύνολο των κανόνων που ορίζει τις μορφές με τις οποίες μια δεσμευμένη λέξη είναι αποδεκτή και αναγνωρίσιμη από μια γλώσσα προγραμματισμού. Για

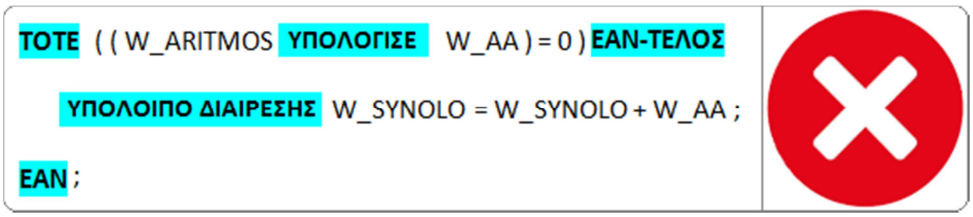
παράδειγμα, η λέξη THROUGH είναι όμοια με την λέξη THRU, η λέξη COLUMN είναι όμοια με την COL, η λέξη CORRESPONDING είναι όμοια με την CORR, η φράση NOT AT END είναι όμοια με την φράση NOT END, κ.λπ. Στην πληροφορική, η έννοια της γραμματικής για τις γλώσσες προγραμματισμού, μελετάται στην βάση ενός θεωρητικού μαθηματικού υποβάθρου, το οποίο όμως δεν αποτελεί στόχο του συγγράμματος.

6.4. Το Συντακτικό

Ένα από τα σημαντικότερα στοιχεία για την κατανόηση μιας γλώσσας προγραμματισμού είναι η γνώση των κανόνων διάταξης και διασύνδεσης των αποδεκτών λέξεων της γλώσσας με στόχο, την δημιουργία σύνθετων και ταυτόχρονα, ορθών προγραμματιστικών προτάσεων. Αυτό, για να γίνει κατανοητό, αρκεί να μελετηθεί ένα μικρό κομμάτι από ένα σημείο από το κέντρο του ψευδοκώδικα που μελετήθηκε σε προηγούμενο κεφάλαιο και αφορούσε τον αλγόριθμο για τους τέλειους αριθμούς.



Σε αυτό το μικρό κομμάτι κώδικα που αποτελεί το σώμα της εντολής **ΕΑΝ**…, εάν επανατοποθετήσουμε με διαφορετικό τρόπο τις συμφωνημένες λέξεις, σε σχέση με το πώς είχαν αρχικά καταστρωθεί, προφανώς παύει και η ικανότητα ορθής λειτουργίας του αλγορίθμου.



Συνεπώς, η ολοκληρωμένη γνώση από τους προγραμματιστές, των συντακτικών κανόνων και των επιλογών ορθής συμπλοκής των αποδεκτών λέξεων της γλώσσας, δημιουργεί τις σωστές προτάσεις που θα μπορούν να ερμηνεύονται σωστά από την μηχανή.

6.5. Η Σημασιολογία

Η σημασιολογία στις γλώσσες προγραμματισμού είναι το σύνολο των κανόνων, που περιγράφουν την σωστή χρήση των λέξεων και των κανόνων σύνταξης και όχι καθαυτή η σωστή τους σύνταξη. Ειδικότερα, σημασιολογία είναι το σύνολο των κανόνων που καθορίζουν το νόημα των λέξεων με βάση το πώς τοποθετούνται επάνω σε μια προγραμματιστική πρόταση και κατά επέκταση των συνεπειών που προκαλούν όταν αναγνωρίζονται από αυτή την γλώσσα. Σε πολλές γλώσσες το σώμα μιας συγκεκριμένης εντολής μπορεί να γραφεί με πολλούς διαφορετικούς τρόπους και αυτό να δώσει τελικά μια εντελώς διαφορετική σημασία στην λειτουργία που θα επιτελέσει η γλώσσα. Αυτό αποτελεί την σημασιολογία.

- WHILE { και εδώ να κάνει κάποια ενέργεια }
- { και εδώ να κάνει κάποια ενέργεια } WHILE

Για παράδειγμα, εάν η δεσμευμένη λέξη WHILE, σε μια γλώσσα προγραμματισμού χρησιμοποιηθεί με έναν από τους ανωτέρω τρόπους, η αποτελεσματικότητα είναι εντελώς διαφορετική για κάθε περίπτωση.

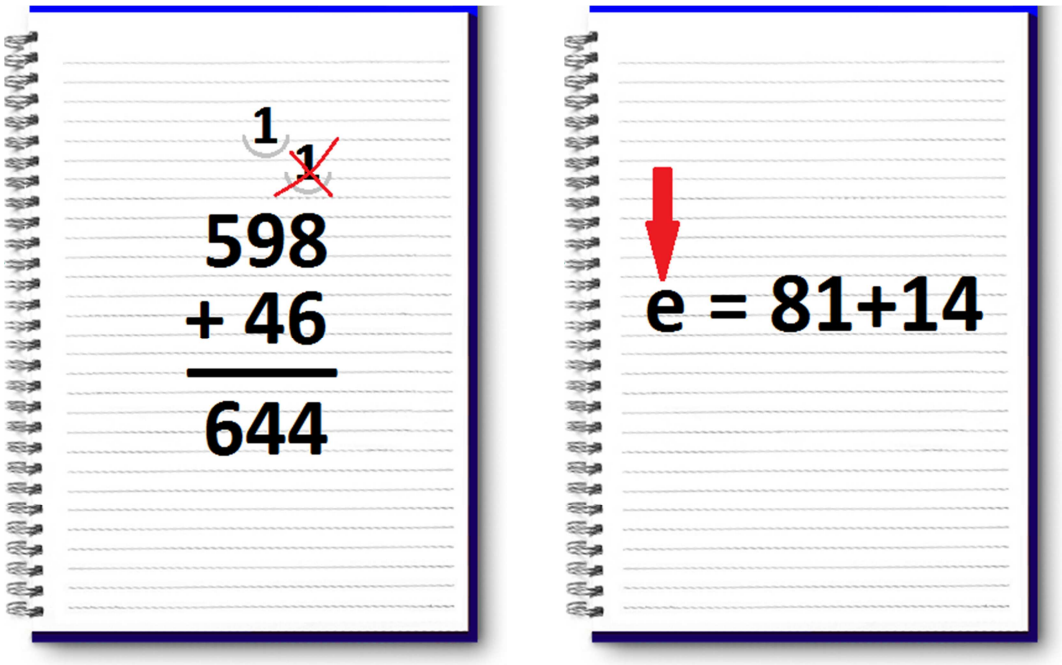
6.6. Οι Τελεστές

Μέσα στο αλφάβητο μιας γλώσσας προγραμματισμού περιλαμβάνονται και πολλοί ειδικοί χαρακτήρες (+ - * / = ^ : ; % # κ.λπ.). Κάποια όμως από τα σύμβολα αυτά έχουν μια επιπλέον σημαντική σημασία για τις γλώσσες. Οι συμβολισμοί που έχουν αυτή την ειδική σημασία, ονομάζονται *τελεστές*. Οι τελεστές είναι τα γνωστά σύμβολα που χρησιμοποιούμε στις απλές αριθμητικές πράξεις και στις συγκρίσεις. Εκτός από τους αριθμητικούς και τους συγκριτικούς τελεστές, οι γλώσσες προγραμματισμού διαθέτουν και λογικούς τελεστές. Η μορφή και η σημασία ενός τελεστή, μπορεί να αλλάζει ανάλογα την γλώσσα προγραμματισμού. Οι τελεστές εκτός από τις γλώσσες προγραμματισμού, χρησιμοποιούνται και στα λογικά διαγράμματα και στις ψευδογλώσσες. Οι τελεστές επιδρούν ανάμεσα σε αντικείμενα που ονομάζονται *τελεστέοι*. Για παράδειγμα στην έκφραση 81 + 14, το σύμβολο + είναι ο τελεστής και οι αριθμοί 81 και 14 είναι οι τελεσταίοι. Οι τελεστές επιτυγχάνουν διαφορετική λειτουργία επάνω στην ίδια έκφραση ανάλογα με την θέση που θα

τοποθετηθούν σε σχέση με τους τελεστέους. Οι τελεστές ταξινομούνται σε κατηγορίες, ακόμα και από τον αριθμό των τελεστέων που επηρεάζουν. Επίσης, είναι πολύ σημαντικό να αναγνωρίζεται από τους προγραμματιστές, ότι οι τελεστές διαθέτουν επίπεδα προτεραιότητας, με τους τελεστές υψηλότερης προτεραιότητας να επιδρούν πριν από τους τελεστές χαμηλότερης προτεραιότητας.

6.7. Οι Μεταβλητές

Η έννοια της μεταβλητής (variable) είναι γνωστή από την αριθμητική (με το «καπελάκι» της πρόσθεσης) αλλά και από τα μαθηματικά. Για παράδειγμα στον αριθμητικό τύπο, e=81+14 η μεταβλητή είναι το e. Μια μεταβλητή, στον κόσμο του προγραμματισμού παριστάνει μια ποσότητα μνήμης που το περιεχόμενο της μπορεί να μεταβάλλεται και για το λόγο αυτό, ονομάζεται *μεταβλητή*.



Οι μεταβλητές που χρησιμοποιούνται σε ένα πρόγραμμα, αντιστοιχούν σε συγκεκριμένες θέσεις μνήμης μέσα στον υπολογιστή. Κάθε μεταβλητή δηλαδή έχει δική της αποκλειστική θέση μνήμης μέσα στον υπολογιστή. Η τιμή της μεταβλητής είναι το περιεχόμενο που βρίσκεται στην αντίστοιχη θέση μνήμης και όπως αναφέρθηκε μπορεί να μεταβάλλεται κατά τη διάρκεια της εκτέλεσης του προγράμματος. Μπορούμε να φανταστούμε τη μεταβλητή και την αντίστοιχη θέση μνήμης στον υπολογιστή σαν ένα «προσωρινό κουτί», το οποίο εξωτερικά

έχει για όνομα (το όνομα της μεταβλητής) και ως περιεχόμενο, εσωτερικά, την τιμή που έχει εκείνη τη συγκεκριμένη στιγμή η μεταβλητή. Ενώ η τιμή της μεταβλητής μπορεί να αλλάζει κατά την εκτέλεση του προγράμματος, αυτό που μένει υποχρεωτικά αναλλοίωτο είναι ο τύπος και το όνομα της μεταβλητής. Οι γλώσσες προγραμματισμού επιτρέπουν συνήθως τη χρήση κάποιων βασικών τύπων μεταβλητών (ακεραίων αριθμών, πραγματικών αριθμών, ημερομηνιών, χαρακτήρων και λογικών μεταβλητών) τους οποίους θα αναλύσουμε στις επόμενες ενότητες. Οι μεταβλητές χρησιμοποιούνται και στα λογικά διαγράμματα και στις ψευδογλώσσες.

Τα ονόματα των μεταβλητών τα αποφασίζουν οι προγραμματιστές και μπορούν να αποτελούνται από γράμματα, πεζά ή κεφαλαία, του λατινικού αλφαβήτου, ψηφία (0-9) και συνήθως τους χαρακτήρες, κάτω παύλα (underscore) (_) ή κανονική παύλα (-). Στα όρια του δομημένου προγραμματισμού, είναι καλό να ορίζουμε περιγραφικά ονόματα για τις μεταβλητές μας (W-MY-COUNT-VALUE, W-O-METRHTHS-MOY, κ.λπ.).

Αναγνωρίζοντας στο μυαλό του, ένας προγραμματιστής, την ανάγκη για χρήση μιας νέας μεταβλητής για το πρόγραμμα του, χρειάζεται στην πραγματικότητα μια ποσότητα μνήμης, την οποία ζητά να δεσμεύει από τους διαθέσιμους πόρους ενός υπολογιστή.

Για να γίνει αυτό δυνατό θα πρέπει,

- να την δηλώσει, πριν την χρησιμοποιήσει, δίνοντας της ένα μοναδικό όνομα,
- να καθορίσει τον τύπο της (εάν δηλαδή σκοπεύει να την γεμίζει με γράμματα ή με αριθμούς),
- να την καθαρίσει, μήπως είχε μέσα «σκουπίδια» από προηγούμενη χρήση (απροσδιοριστία),
- να την χρησιμοποιήσει ακολούθως και καταλλήλως, μέσα στο πρόγραμμα του.

Στις γλώσσες προγραμματισμού εκτός από τις απλές μεταβλητές υπάρχουν και κάποιες ειδικές μεταβλητές οι οποίες ονομάζονται *σταθερές* (constants). Οι σταθερές, λαμβάνουν μια αρχική τιμή (π.χ. το 3.14, την λέξη «ΝΑΙ», κ.λπ.) και την διατηρούν σε όλη την διάρκεια εκτέλεσης του προγράμματος. Οι σταθερές χρησιμοποιούνται επίσης, στα λογικά διαγράμματα, στις ψευδογλώσσες και μπορούν να είναι διαφόρων τύπων όπως και οι απλές μεταβλητές (ακέραιοι, πραγματικοί, ημερομηνίες, χαρακτήρες και λογικές). Στις νεότερες γλώσσες η αναγνώριση του τύπου της κάθε μεταβλητής γίνεται αυτόματα.

6.8. Τύποι Μεταβλητών - Τύποι Δεδομένων

Κατά την συγγραφή ενός νέου προγράμματος σε μια γλώσσα προγραμματισμού είναι λογικό να υποθέσουμε ότι θα χρειαστούμε πολλές διαφορετικές μεταβλητές. Επιπλέον είναι λογικό να υποθέσουμε ότι οι μεταβλητές αυτές δεν θα είναι όλες του ίδιου αλλά διαφορετικού τύπου. Δηλαδή, μπορεί να χρειαστεί να ορίσουμε μια μεταβλητή για να τοποθετήσουμε αριθμούς, μια άλλη μεταβλητή για να τοποθετήσουμε το όνομα ενός δρόμου και μια άλλη για να αποθηκεύσουμε ημερομηνίες. Συνεπώς, ο αλγόριθμος μας χρειάζεται 3 διαφορετικούς τύπους δεδομένων, μια για αριθμούς, μια για γράμματα και μια για ημερομηνίες. Οι τύποι των δεδομένων (οι τύποι των μεταβλητών δηλαδή) που υποστηρίζει μια γλώσσα προγραμματισμού είναι πολλοί και διαφορετικοί, αλλά συνήθως σε όλες τις γλώσσες αφορούν, τους ακέραιους αριθμούς, τους πραγματικούς αριθμούς, τους χαρακτήρες και έναν ειδικό τύπο που ονομάζεται *λογικός τύπος*.

6.8.1. Ακέραιος τύπος

Ο τύπος αυτός αφορά τους ακέραιους αριθμούς που είναι γνωστοί από τα μαθηματικά. Οι ακέραιοι μπορούν να είναι θετικοί, αρνητικοί ή το μηδέν (π.χ. 1, 67419, 0, -12980). Οι μεταβλητές αυτού του τύπου μπορούν να γεμίσουν με ακέραιες ποσότητες χωρίς κλασματικό μέρος.

6.8.2. Πραγματικός τύπος

Ο τύπος αυτός αφορά τους πραγματικούς αριθμούς που γνωρίζουμε από τα μαθηματικά και μπορεί να περιγράφει θετικούς ή αρνητικούς αριθμούς, με επιπλέον όμως την ικανότητα να αναπαριστά και κλασματικό μέρος (π.χ. 33.156159, 0.00, 22.743528, -452.45, 0.445).

6.8.3. Τύπος Χαρακτήρων

Ο τύπος αυτός αναφέρεται σαν περιεχόμενο, τόσο σε έναν χαρακτήρα όσο και μια σειρά χαρακτήρων. Τα δεδομένα αυτού του τύπου ονομάζονται *αλφαριθμητικά* και μπορούν να περιέχουν έναν οποιονδήποτε ή πολλούς οποιουσδήποτε χαρακτήρες που παράγονται από το πληκτρολόγιο (π.χ. «Κ», «$», «Κώστας», «σήμερα είναι καλά», «Τα οχήματα είναι 2», «το υπόλοιπο είναι > από 3.14», «/», κ.λπ.). Ο χειρισμός αυτού του τύπου των μεταβλητών υποχρεώνει τους προγραμματιστές να τοποθετούν τα περιεχόμενα ανάμεσα σε εισαγωγικά.

6.8.4. Τύπος Λογικός

Ο ειδικός αυτός τύπος χρησιμοποιείται σαν ένας εικονικός διακόπτης ο οποίος μπορεί να δέχεται μόνο δύο καταστάσεις, να είναι αληθές ή ψευδές. Χρησιμοποιείται σαν λογικός διακόπτης.

Ολοκληρώνοντας την περιγραφή για τους τύπους των δεδομένων, θα πρέπει να λάβουμε επίσης υπόψη ότι στις γλώσσες προγραμματισμού είμαστε υποχρεωμένοι να αποφασίζουμε και να δηλώνουμε και το μέγεθος το οποίο χρειαζόμαστε για κάθε «καπελάκι». Για παράδειγμα, στην μεταβλητή τύπου χαρακτήρων, για το όνομα ενός δρόμου θα πρέπει να αποφασίσουμε πόσο χώρο θα χρειαστούμε, καθώς με 20 κελιά μνήμης θα μπορούμε να

χειριστούμε διευθύνσεις δρόμων για μέχρι και 20 χαρακτήρες, ενώ για μεγαλύτερα ονόματα δρόμων θα πρέπει να δηλώσουμε μεταβλητή με μεγαλύτερο μέγεθος. Στις γλώσσες προγραμματισμού η δήλωση του τύπου δεδομένων γίνεται με συγκεκριμένες «εργοστασιακές λέξεις». Για παράδειγμα, για τους ακεραίους θα μπορούσαμε να χρησιμοποιούμε την λέξη numeric, για τους πραγματικούς την λέξη float, για τους χαρακτήρες την λέξη character και για τους λογικούς την λέξη boolean. Εάν ορίσουμε δηλαδή σε μια γλώσσα, μια μεταβλητή με την δήλωση numeric(3), θα μπορούμε να καλύψουμε τα όρια των τιμών από το -999 έως το +999 ενώ εάν ορίσουμε την ίδια μεταβλητή με την δήλωση numeric(6), θα μπορούμε να καλύψουμε τα όρια των τιμών από το -999999 έως το +999999. Τα μεγέθη των μεταβλητών στα λογικά διαγράμματα και στις ψευδογλώσσες, δεν χρησιμοποιούνται.

6.9. Οι Βασικές Αλγοριθμικές Δομές

Στις γλώσσες προγραμματισμού, οι μεθοδολογίες επίλυσης, για πολλά προβλήματα, μπορούν να περιορίζονται σε μια απλή μαθηματική πράξη ή σε έναν έλεγχο με μια θετική και αρνητική απόφαση. Για παράδειγμα, η εντολή **ΕΑΝ**… **ΤΟΤΕ**… **ΑΛΛΙΩΣ**… υλοποιεί τη δομή της επιλογής με βάση μια απόφαση σε μια συνθήκη. Σε πολλές όμως άλλες περιπτώσεις αυτό δεν είναι αρκετό καθώς τα προβλήματα που πρέπει να μελετηθούν είναι πολύ δυσκολότερα και απαιτούν διαφορετικούς τρόπους επίλυσης. Για τους λόγους αυτούς οι γλώσσες προγραμματισμού διαθέτουν ικανότερους μηχανισμούς επίλυσης.

Οι μηχανισμοί αυτοί ονομάζονται *αλγοριθμικές δομές* και διαχωρίζονται σε δυο βασικές κατηγορίες: α) στις δομές επιλογής και β) στις δομές επανάληψης. Κάθε μια από αυτές τις δύο βασικές κατηγορίες, διαθέτει επιπλέον 3 διαφορετικούς τύπους διαμόρφωσης,

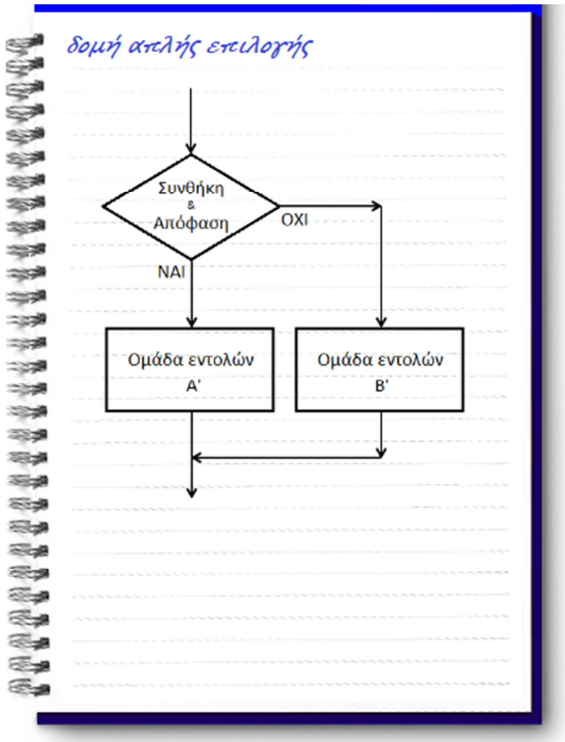
α) για τις αλγοριθμικές δομές επιλογής,

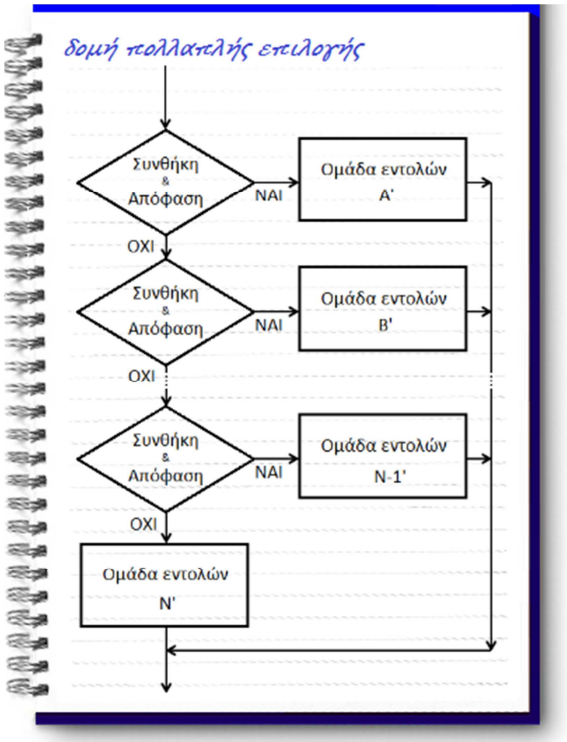
- α.1) την απλής επιλογής,
- α.2) την πολλαπλής επιλογής,
- α.3) την επιλογής με πολλές εξόδους.

β) για τις αλγοριθμικές δομές επανάληψης,

- β.1) με την συνθήκη στην αρχή,
- β.2) με την συνθήκη στο τέλος,
- β.3) με συνθήκη και βηματισμό.

Για κάθε προγραμματιστή, άπειρο ή έμπειρο, η εκμάθηση μιας γλώσσας προγραμματισμού περιλαμβάνει την ολοκληρωμένη κατανόηση των τρόπων με τους οποίους υλοποιεί η γλώσσα αυτές τις δομές. Οι 6 αλγοριθμικές δομές που ακολουθούν και παρουσιάζονται για εκπαιδευτικούς λόγους, σε μορφή λογικών διαγραμμάτων και ψευδογλώσσας, θεωρούνται απολύτως απαραίτητες δομές και ενυπάρχουν σε όλες τις γλώσσες προγραμματισμού με μικρές συντακτικές διαφοροποιήσεις για κάθε γλώσσα. Στις περισσότερες όμως περιπτώσεις τα στοιχεία της κάθε δομής είναι ακριβώς τα ίδια. Προτρέπονται οι εκπαιδευόμενοι να δώσουν μεγάλη βαρύτητα στη μελέτη και την κατανόηση της πορείας που επιτελεί αυτή η «αόρατη μπίλια» (ροή αλγορίθμου), μέσα σε κάθε ένα από τα λογικά διαγράμματα και να αντιληφθούν στην συνέχεια τον τρόπο με τον οποίο το κάθε λογικό διάγραμμα μετασχηματίζεται σε αντίστοιχες εντολές ψευδοκώδικα.

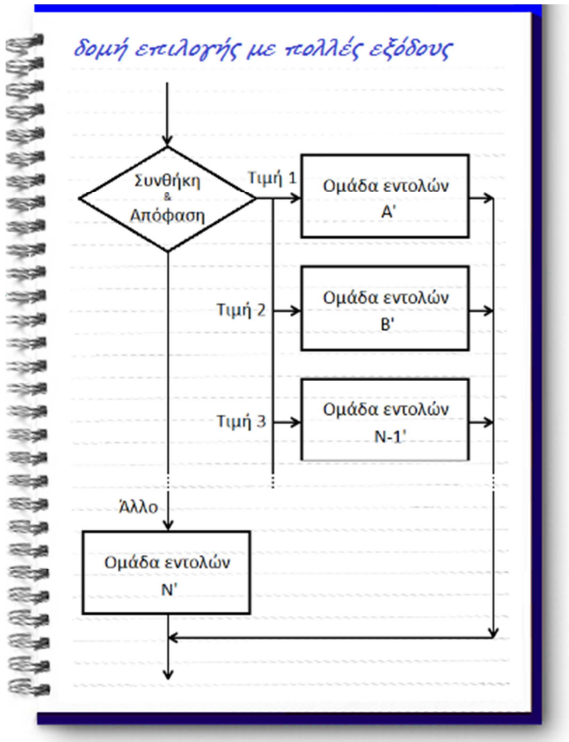


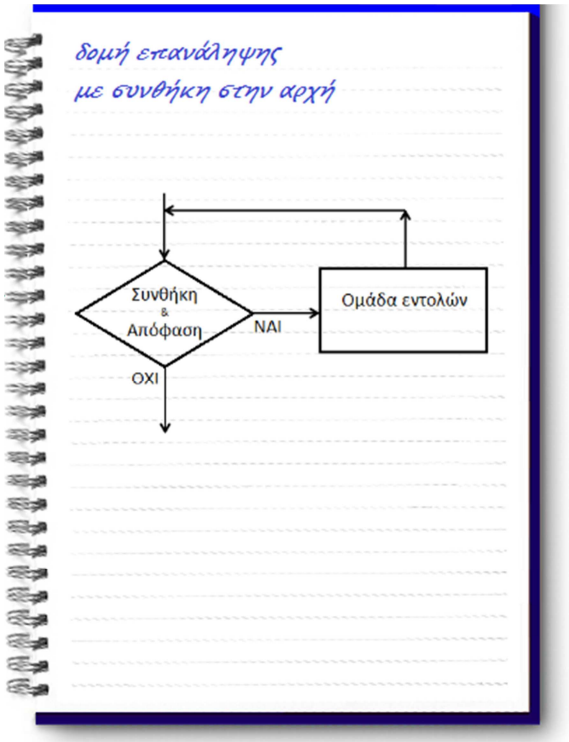



```
ΕΑΝ ... ΤΟΤΕ
    ...
ΑΛΛΙΩΣ
    ...
ΕΑΝ-ΤΕΛΟΣ
```



```
ΕΑΝ ... ΤΟΤΕ
    ...
ΑΛΛΙΩΣ
        ΕΑΝ ... ΤΟΤΕ
            ...
        ΑΛΛΙΩΣ
            ⋮
            ΕΑΝ ... ΤΟΤΕ
                ...
            ΑΛΛΙΩΣ
                ...
            ΕΑΝ-ΤΕΛΟΣ
        ΕΑΝ-ΤΕΛΟΣ
ΕΑΝ-ΤΕΛΟΣ
```





ΕΛΕΓΞΕ ...

ΣΤΗΝ ΠΕΡΙΠΤΩΣΗ ...

...

ΣΤΗΝ ΠΕΡΙΠΤΩΣΗ ...

...

ΣΤΗΝ ΠΕΡΙΠΤΩΣΗ ...

...

⋮

ΔΙΑΦΟΡΕΤΙΚΑ

...

ΕΛΕΓΞΕ-ΤΕΛΟΣ

Ψευδογλώσσα

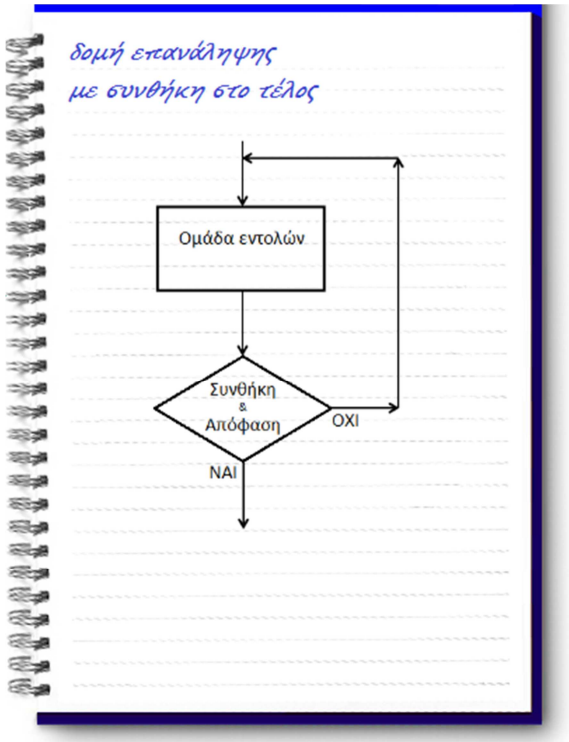
ΜΕΧΡΙ ...

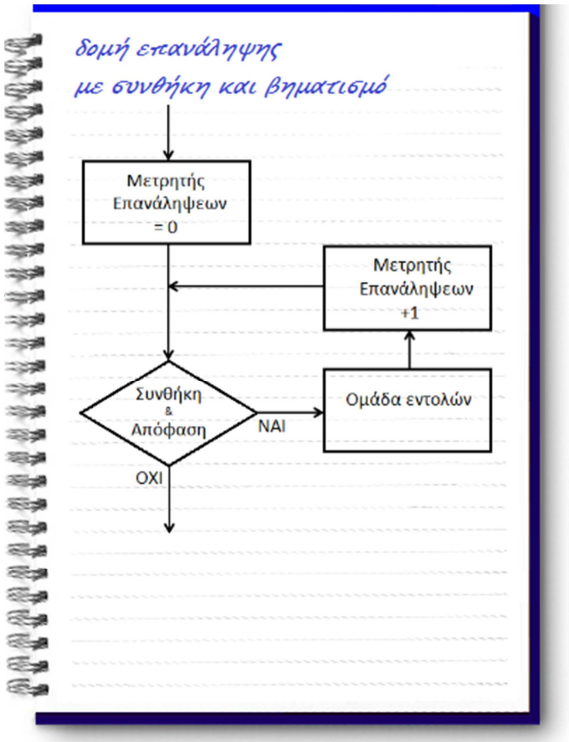
ΕΠΑΝΕΛΑΒΕ

...

ΕΠΑΝΕΛΑΒΕ-ΤΕΛΟΣ

Ψευδογλώσσα





ΕΠΑΝΕΛΑΒΕ

...

ΜΕΧΡΙ ...

ΕΠΑΝΕΛΑΒΕ-ΤΕΛΟΣ

Ψευδογλώσσα

ΕΠΑΝΕΛΑΒΕ

ΑΠΟ ... **ΜΕΧΡΙ** ... **ΜΕ ΒΗΜΑ** ...

...

ΕΠΑΝΕΛΑΒΕ-ΤΕΛΟΣ

Ψευδογλώσσα

Σύνοψη κεφαλαίου

Στο κεφάλαιο αυτό παρουσιάστηκαν με αναλυτικό τρόπο τα βασικά χαρακτηριστικά μιας γλώσσας προγραμματισμού, το αλφάβητο, το λεξιλόγιο, η γραμματική και το συντακτικό της, καθώς και οι κανόνες σημασιολογίας που συνήθως χρησιμοποιούνται από όλες τις γλώσσες προγραμματισμού. Επιπλέον, περιγράφηκαν οι έννοιες, μεταβλητή, σταθερά και τελεστές, ενώ παρουσιάστηκαν και όλοι οι βασικοί τύποι δεδομένων. Ακολούθως, στο κεφάλαιο περιγράφηκαν οι βασικές αλγοριθμικές δομές που υποστηρίζονται από όλες της γλώσσες κατασκευής προγραμμάτων και δόθηκαν αναλυτικά παραδείγματα σε μορφή λογικών διαγραμμάτων και ψευδοκώδικα.

Ερωτήσεις αξιολόγησης

Ερώτηση-1: Περιγράψτε, ποια είναι τα πρώτα γνωστικά αντικείμενα που πρέπει να αναζητούνται κατά την εκμάθηση μιας γλώσσας προγραμματισμού υπολογιστών;

Ερώτηση-2: Περιγράψτε, τι είναι το λεξιλόγιο μιας γλώσσας προγραμματισμού;

Ερώτηση-3: Μια αναβάθμιση μιας γλώσσας προγραμματισμού, τι περιλαμβάνει;

Ερώτηση-4: Περιγράψτε, τι είναι η γραμματική μιας γλώσσας προγραμματισμού;

Ερώτηση-5: Περιγράψτε, τι είναι το συντακτικό μιας γλώσσας προγραμματισμού;

Ερώτηση-6: Ποιους τελεστές γνωρίζετε στις γλώσσες προγραμματισμού υπολογιστών;

Ερώτηση-7: Τι πρέπει να κάνει ένας προγραμματιστής, από την στιγμή που θα αποφασίσει ότι χρειάζεται σε ένα πρόγραμμα, μια μεταβλητή;

Ερώτηση-8: Περιγράψτε, τι είναι οι τύποι δεδομένων, δώστε μερικά παραδείγματα;

Ερώτηση-9: Ποιες είναι, οι βασικές αλγοριθμικές δομές επιλογής;

Ερώτηση-10: Ποιες είναι, οι βασικές αλγοριθμικές δομές επανάληψης;

Ερώτηση-11: Προσπαθήστε σαν δραστηριότητα, να μετασχηματίσετε, για κάθε μια από τις έξι αλγοριθμικές δομές, από το λογικό διάγραμμα σε ψευδοκώδικα (με χαρτί και με μολύβι και χωρίς να βλέπετε τον έτοιμο ψευδοκώδικα, της ενότητας 6.9.);

Ερώτηση-12: Προσπαθήστε σαν δραστηριότητα, να μετασχηματίσετε, για κάθε μια από τις έξι αλγοριθμικές δομές, από τον ψευδοκώδικα σε λογικό διάγραμμα (με χαρτί και με μολύβι και χωρίς να βλέπετε το έτοιμο λογικό διάγραμμα, της ενότητας 6.9.);

Απαντήσεις ερωτήσεων αξιολόγησης

Απάντηση-1: Συμβουλευτείτε τις εισαγωγικές παρατηρήσεις του κεφαλαίου.

Απάντηση-2: Συμβουλευτείτε την ενότητα 6.2. του κεφαλαίου.

Απάντηση-3: Συμβουλευτείτε την ενότητα 6.2. του κεφαλαίου.

Απάντηση-4: Συμβουλευτείτε την ενότητα 6.3. του κεφαλαίου.

Απάντηση-5: Συμβουλευτείτε την ενότητα 6.4. του κεφαλαίου.

Απάντηση-6: Συμβουλευτείτε την ενότητα 6.6. του κεφαλαίου.

Απάντηση-7: Συμβουλευτείτε την ενότητα 6.7. του κεφαλαίου.

Απάντηση-8: Συμβουλευτείτε την ενότητα 6.8. του κεφαλαίου.

Απάντηση-9: Συμβουλευτείτε την ενότητα 6.9. του κεφαλαίου.

Απάντηση-10: Συμβουλευτείτε την ενότητα 6.9. του κεφαλαίου.

Απάντηση-11: Συμβουλευτείτε την ενότητα 6.9. του κεφαλαίου.

Απάντηση-12: Συμβουλευτείτε την ενότητα 6.9. του κεφαλαίου.

7. ΑΠΟ ΤΑ ΔΙΑΓΡΑΜΜΑΤΑ ΚΑΙ ΤΗΝ ΨΕΥΔΟΓΛΩΣΣΑ, ΣΤΗΝ ΓΛΩΣΣΑ C

Σκοπός και επιμέρους στόχοι

Στο έβδομο και τελευταίο κεφάλαιο οι εκπαιδευόμενοι μελετούν προγράμματα γραμμένα σε γλώσσα προγραμματισμού C. Ειδικότερα, χρησιμοποιούνται από προηγούμενα κεφάλαια, αλγόριθμοι οι οποίοι κατασκευάστηκαν με ψευδογλώσσα και μετατρέπονται βήμα προς βήμα σε προγράμματα C. Οι εκπαιδευόμενοι αποκτούν ικανότητες συγγραφής και διερμηνείας κώδικα στην γλώσσα προγραμματισμού C.

Προσδοκώμενα αποτελέσματα

Με την μελέτη του έβδομου κεφαλαίου οι εκπαιδευόμενοι θα μπορούν,

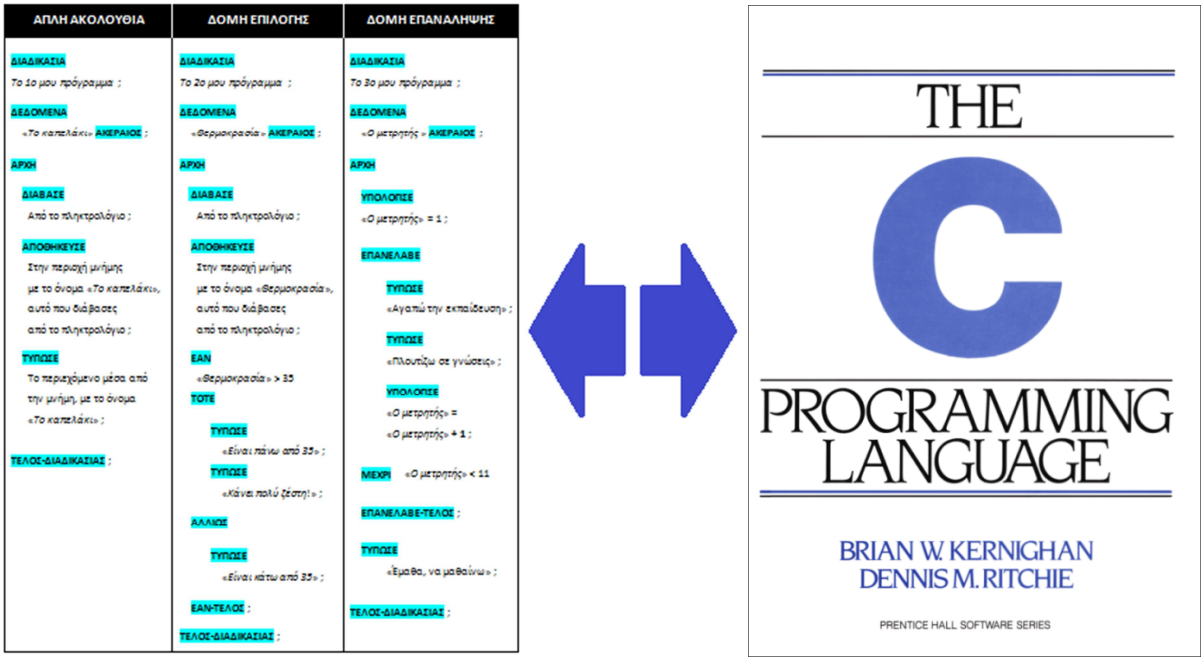
- να αναγνωρίζουν τον βασικό κορμό των απαραίτητων εντολών που χρειάζεται για να λειτουργήσει ένα πρόγραμμα σε C,
- να μετατρέπουν έναν αλγόριθμο από ψευδογλώσσα σε ένα σύνολο συγκεκριμένων εντολών σε γλώσσα προγραμματισμού C,
- να διακρίνουν σε ένα πρόγραμμα C, ποιες είναι οι λέξεις κλειδιά,
- να καταλαβαίνουν γιατί σε ένα πρόγραμμα C, είναι αναγκαίο πρέπει να γράφουν με δομημένο τρόπο,
- να αντιλαμβάνονται την αναγκαιότητα να επιλέγουν περιγραφικά ονόματα για τις μεταβλητές τους,
- να διερμηνεύουν μια σειρά από εντολές βασικών αλγοριθμικών δομών για την εκτέλεση μιας εργασίας στην γλώσσα προγραμματισμού C.

Έννοιες – Λέξεις Κλειδιά

Επεξεργαστής κειμένου (text editor), Πηγαίος κώδικας (source code), Εκτελέσιμος κώδικας (executable code), Μετάφραση (compilation), Μεταγλώττιση, Μεταφραστές (compilers), Διερμηνευτές (interpreters), Ολοκληρωμένα περιβάλλοντα ανάπτυξης (Integrated Development Environment).

Εισαγωγικές Παρατηρήσεις

Η κατανόηση του διαδικαστικού προγραμματισμού και η υιοθέτηση των αρχών του δομημένου προγραμματισμού, προσφέρουν στους νέους προγραμματιστές τα πρώτα κατάλληλα εφόδια για να εμβαθύνουν στην συνέχεια, σε σύνθετα προβλήματα και να μπορούν να επινοούν κατάλληλους αλγορίθμους επίλυσης. Για την μύηση σε αυτές στις προγραμματιστικές τακτικές, παρουσιάστηκαν αναλυτικά, με λογικά διαγράμματα και ψευδοκώδικες, αλγόριθμοι, όπως η απλή ακολουθία, η δομή επιλογής και μια δομή επανάληψης. Καθώς όμως ο απώτερος εκπαιδευτικός στόχος είναι η μετάβαση σε μια πραγματική γλώσσα προγραμματισμού, οι αλγόριθμοι, της απλής ακολουθίας, της δομής επιλογής και της δομής επανάληψης, παρουσιάζονται και πάλι, γραμμένοι όμως αυτή τη φορά στην γλώσσα προγραμματισμού C, με όλους του απαραίτητους συντακτικούς κανόνες και τους περιορισμούς της γλώσσας.



7.1. Η γλώσσα Προγραμματισμού C

Η γλώσσα C, είναι μια ικανή πλατφόρμα προγραμματισμού με ισχυρά χαρακτηριστικά, κατάλληλη για την ανάπτυξη πολλών διαφορετικών τύπων εφαρμογών. Η γλώσσα C, είναι ένα περιβάλλον στο οποίο εκπαιδεύονται και έχουν εκπαιδευτεί, όλοι σχεδόν οι μηχανικοί λογισμικού και οι προγραμματιστές, όλων των πρόσφατων εποχών. Οι μηχανισμοί λειτουργίας

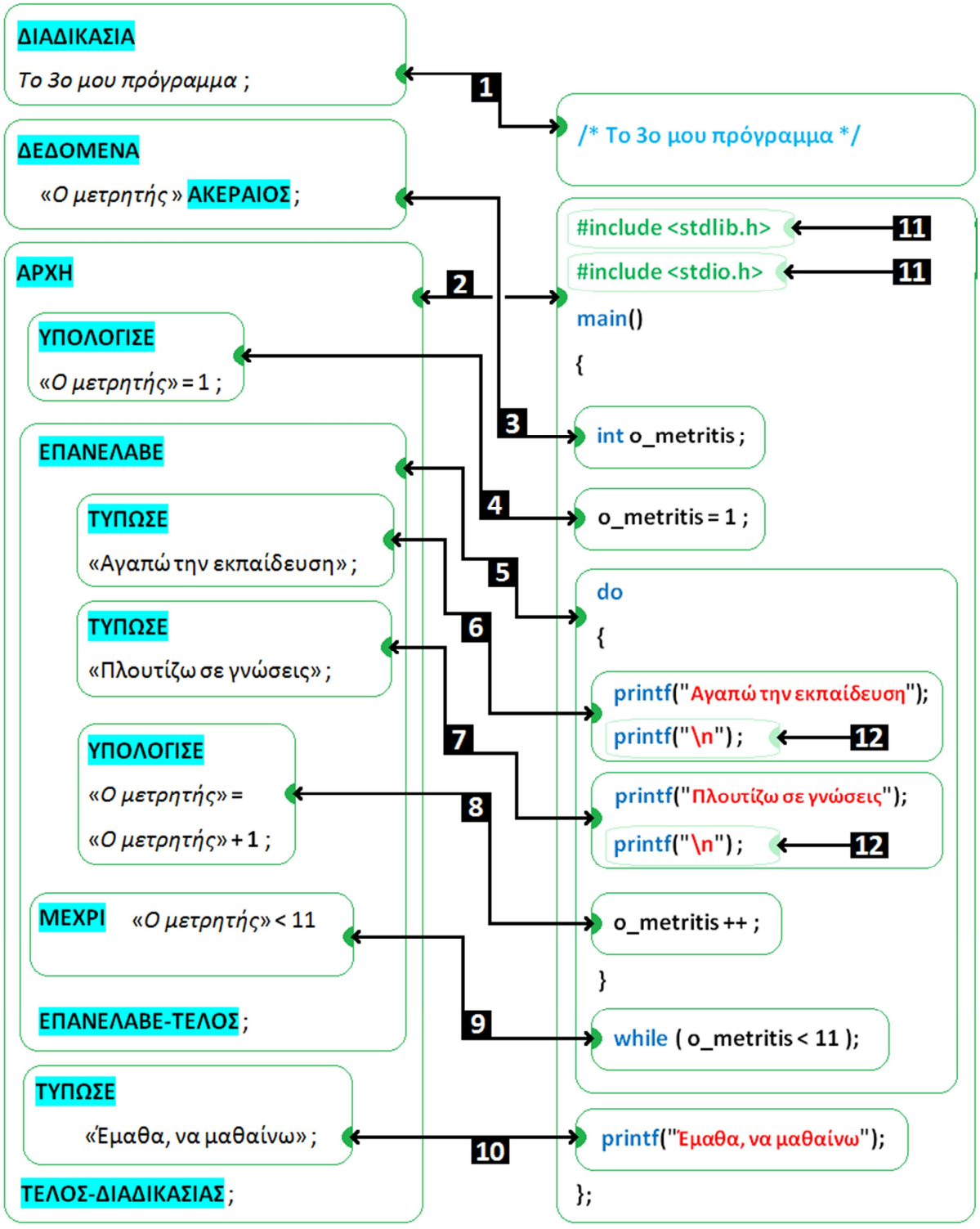
της, έχουν αφομοιωθεί από πολλές άλλες γλώσσες και δεν είναι υπερβολικό να ειπωθεί ότι οι περισσότερες σύγχρονες γλώσσες προγραμματισμού είναι C-like γλώσσες (ομοιάζουσες). Σήμερα, η γλώσσα C χρησιμοποιείται περισσότερο με τις εξελιγμένες εκδόσεις αντικειμενοστραφούς προγραμματισμού (Object Oriented Programming), όπως η C++ και η C#.

ΑΠΛΗ ΑΚΟΛΟΥΘΙΑ	ΔΟΜΗ ΕΠΙΛΟΓΗΣ	ΔΟΜΗ ΕΠΑΝΑΛΗΨΗΣ

/* Το 1ο μου πρόγραμμα */
#include <stdlib.h>
#include <stdio.h>
main()
{
int to_kapelaki ;
scanf("%d", &to_kapelaki) ;
printf("%d\n", to_kapelaki) ;
 	
} ;
	/* Το 2ο μου πρόγραμμα */
	#include <stdlib.h>
	#include <stdio.h>
	main()
	{
	int termokrasia ;
	scanf("%d", &termokrasia) ;
	if (termokrasia > 35)
	{
	printf("Είναι πάνω από 35") ;
	printf("\n") ;
	printf("Κάνει πολύ ζέστη!") ;
	}
	else
	{
	printf("Είναι κάτω από 35") ;
	} ;
	
	} ;
		/* Το 3ο μου πρόγραμμα */
		#include <stdlib.h>
		#include <stdio.h>
		main()
		{
		int o_metritis ;
		o_metritis = 1 ;
		do
		{
		printf("Αγαπώ την εκπαίδευση");
		printf("\n") ;
		printf("Πλουτίζω σε γνώσεις") ;
		printf("\n") ;
		
		o_metritis ++ ;
		
		}
		while (o_metritis < 11) ;
		
		printf("Εμαθα, να μαθαίνω") ;
		} ;

7.2. Συγκρίνοντας και Επεξηγώντας

Για την υποστήριξη της ικανότητας μετατροπής αλγορίθμων ψευδοκώδικα, στην γλώσσα προγραμματισμού C, ο αλγόριθμος της δομής επανάληψης, επεξηγείται και περιγράφεται συγκριτικά σε μια βηματική παρουσίαση, όλων των εντολών υλοποίησης του.

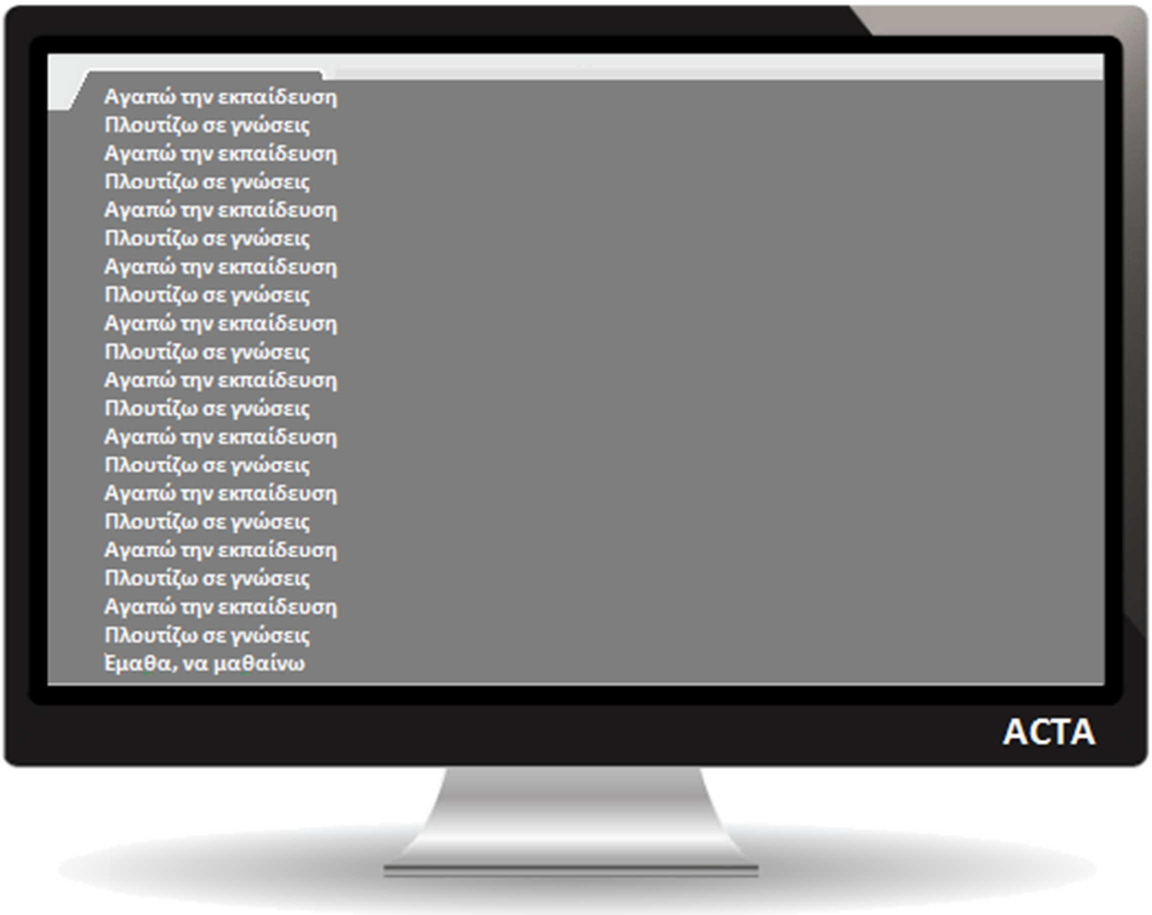


Ο αλγόριθμος είναι αριστερά, σε μορφή ψευδογλώσσας και δεξιά, σε μορφή εντολών γλώσσας προγραμματισμού C++. Επιπλέον, κάθε εντολή ψευδοκώδικα διασυνδέεται με κατάλληλο βέλος προς την αντίστοιχη εντολή C, ενώ κάθε βέλος αριθμείται με έναν μοναδικό αριθμό ο οποίος χρησιμοποιείται ως επεξηγηματική παραπομπή με σχολιασμό στον επόμενο πίνακα.

1	Δηλώνεται μέσα στους χαρακτήρες **/*** και ***/** η έκφραση «Το 3ο μου πρόγραμμα». Η γλώσσα C, όταν «δει» κείμενο ανάμεσα στους χαρακτήρες, **/*** και ***/** θεωρεί ότι είναι απλά σχόλια και δεν προσπαθεί να τα ερμηνεύσει για να τα εκτελέσει.
2	Η δήλωση έναρξης του προγράμματος με την δήλωση **main() { … }**. Τα όρια στα οποία αναπτύσσεται ο κώδικας δηλώνεται από τους χαρακτήρες **{** και **}**. Η πραγματική έναρξη (η αρχή) του κύριου κώδικα υποδεικνύεται από την εντολή **main()**.
3	Η δήλωση της μεταβλητής με το όνομα «*o_metritis*» και η απαραίτητη δήλωση του τύπου της μεταβλητής με το **int** για να μπορεί να δεχθεί ακέραιους αριθμούς.
4	Η αρχικοποίηση της μεταβλητής «*o_metritis*», με την τιμή 1.
5	Η εντολή **do{ … } while …** υποδηλώνει την δομή επανάληψης, η οποία θα παγιδεύει την «αόρατη μπίλια» (τη ροή του αλγορίθμου) μέσα σε μια ατέρμονη διαδικασία μέχρι να συμβεί το γεγονός, από τη δήλωση **while …**
6	Η εντολή **print(…)**, θα τυπώσει στην οθόνη την πρόταση «Αγαπώ την εκπαίδευση». Το κείμενο πρέπει να βρίσκεται ανάμεσα σε «».
7	Η εντολή **print(…)**, θα τυπώσει στην οθόνη την πρόταση «Πλουτίζω σε γνώσεις». Το κείμενο πρέπει να βρίσκεται ανάμεσα σε «».
8	Η αύξηση του περιεχομένου, της αξίας της μεταβλητής «*o_metritis*», κατά +1. Η μεταβλητή λειτουργεί σαν ένας καταμετρητής για να είναι δυνατή, στην συνέχεια, η διακοπή της δομής επανάληψης (στην περίπτωση αυτή, το 11).
9	Η δήλωση **while …** που συνοδεύει την εντολή **do{ … }**, θα απελευθερώσει, τη ροή της επανάληψης όταν η μεταβλητή «*o_metritis*» φτάσει τον αριθμό 11 (και θα έχουν ήδη ολοκληρωθεί 10 ανακυκλώσεις).
10	Η εντολή **print(…)**, θα τυπώσει στην οθόνη την πρόταση «Έμαθα, να μαθαίνω». Το κείμενο πρέπει να βρίσκεται ανάμεσα σε «».
11	Για να λειτουργήσει ένα πρόγραμμα, εκτός από τον καθαυτό «καθαρό» κώδικα του προγραμματιστή, χρειάζεται συνήθως και κάποιες βοηθητικές βιβλιοθήκες οι οποίες περιέχουν μέσα τους κάποιες αναγκαίες πρόσθετες εντολές. Οι εντολές **#include** ενσωματώνουν δυο απαραίτητες βιβλιοθήκες

	με χρήσιμες συναρτήσεις, για αυτόν τον κώδικα.
12	Η εντολή **print(…)**, σε αυτή την μορφή, με τη χρήση της ειδικής δήλωσης **\n** προκαλεί μια αλλαγή γραμμής στα αποτελέσματα. Είναι μια απλή δήλωση κομψότητας.

Η συγγραφή και η επιτυχής εκτέλεση του ανωτέρω προγραμματιστικού κώδικα στην γλώσσα προγραμματισμού C++, θα έχει για αποτελεσματικότητα την εμφάνιση στην οθόνη της επόμενης εικόνας.

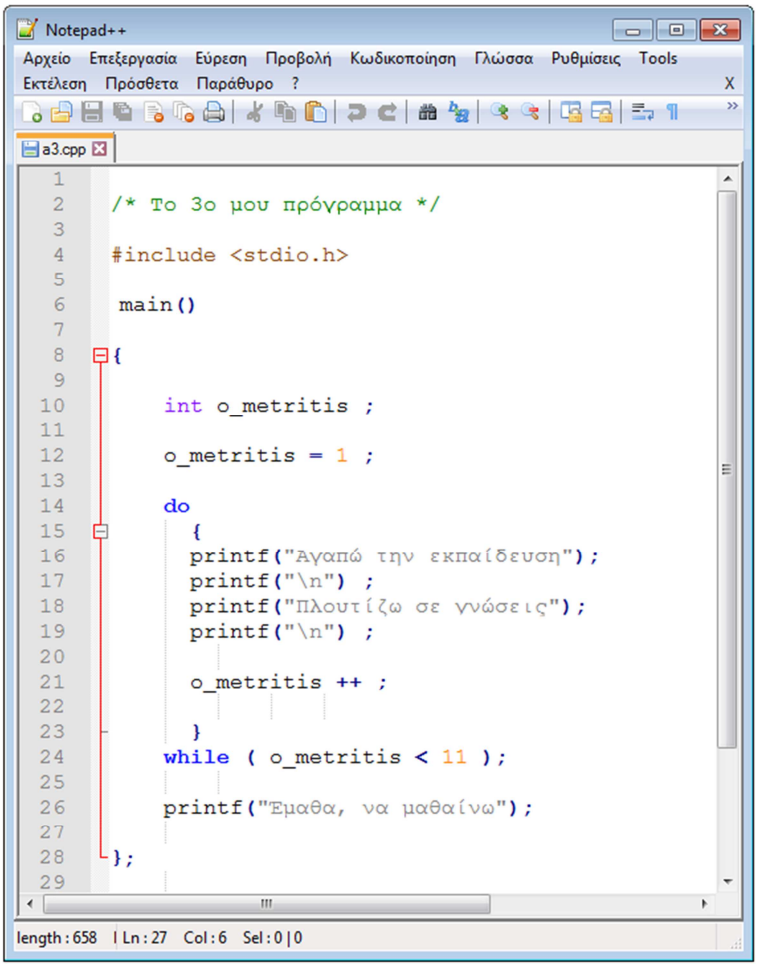


7.3. Γράφοντας και Εκτελώντας ένα Πρόγραμμα σε Γλώσσα C

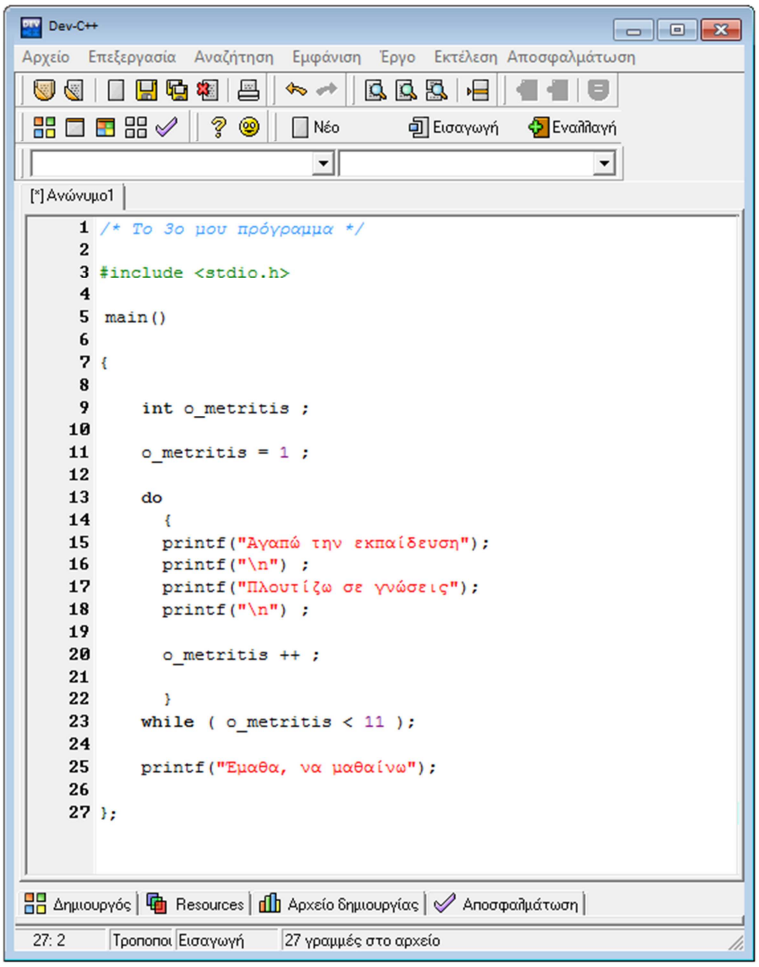
Για να δημιουργήσουμε ένα πρόγραμμα σε C, θα πρέπει αρχικά να συγγράψουμε τις εντολές σε μορφή κειμένου με κάποιον απλό επεξεργαστή κειμένου (text editor). Απλός επεξεργαστής κειμένου θεωρείται για παράδειγμα το εργαλείο «Σημειωματάριο» («Notepad») του λειτουργικού Windows και είναι κάτι πολύ απλούστερο από το Word. Υπάρχουν πολλοί

διαφορετικοί text editors, με διαφορετικά χαρακτηριστικά ο καθένας και είναι στην επιλογή του κάθε προγραμματιστή ποιον θα χρησιμοποιήσει. Οι σύγχρονες γλώσσες προγραμματισμού διαθέτουν από μόνες τους κατάλληλα ολοκληρωμένα περιβάλλοντα για την υποβοήθηση της ανάπτυξης προγραμμάτων (I.D.E.), τα οποία περιέχουν και επεξεργαστή κειμένου.

Στο κόσμο των προγραμματιστών χρησιμοποιούνται εκτός των άλλων, οι σημαντικές έννοιες, *πηγαίος κώδικας* (source code) και *εκτελέσιμος κώδικας* (executable code). Πηγαίος κώδικας, είναι ο κώδικας που καταστρώνει ο προγραμματιστής με το μυαλό και τα χέρια του, με την θεώρηση ότι η πηγή είναι το ανθρώπινο μυαλό (η πηγή των ιδεών). Εκτελέσιμος κώδικας, είναι ο κώδικας που παράγεται μέσα από μια διαδικασία μετατροπής που έχουν όλες οι γλώσσες προγραμματισμού για να μετασχηματίσουν αυτά που γράφει ο άνθρωπος (με τους συγκεκριμένους κανόνες της γλώσσας), σε κάτι που θα το καταλαβαίνει ο υπολογιστής. Η μετατροπή ενός πηγαίου προγράμματος (source code) σε εκτελέσιμο κώδικα (executable code), ονομάζεται *compilation* (μετάφραση ή μεταγλώττιση). Υπάρχουν δύο ειδών μεταφραστικά προγράμματα: α) οι μεταφραστές (compilers) και β) οι διερμηνευτές (interpreters). Η διαφορά τους είναι κυρίως στον τρόπο με τον οποίο μεταφράζουν το πηγαίο πρόγραμμα σε γλώσσα μηχανής, ολόκληρο ή κατά γραμμή.



Επεξεργαστής κειμένου (text editor).



Ολοκληρωμένο περιβάλλον ανάπτυξης(I.D.E.), γλώσσας C.

Ειδικότερα, οι μεταφραστές (compilers) δημιουργούν ολόκληρο το εκτελέσιμο πρόγραμμα το αποθηκεύουν, απαλλαγμένο από λάθη και δεν χρειάζεται να επαναληφθεί η διαδικασία μετάφρασης εάν πρέπει να εκτελεστεί και πάλι το ίδιο πρόγραμμα. Απλά το καλούν πάλι για εκτέλεση, από την αποθηκευμένη μορφή σε γλώσσα μηχανής και για αυτό θεωρούνται πολύ γρήγορα εργαλεία. Αντίθετα, οι διερμηνευτές (interpreters) μεταφράζουν την κάθε γραμμή του προγράμματος ξεχωριστά και εάν δεν ανιχνεύσουν λάθη στην γραμμή, την δίνουν στον υπολογιστή για να την εκτελέσει χωρίς όμως να εκτελούν κάπου, κάποια αποθήκευση. Ακολούθως, συνεχίζουν με την επόμενη γραμμή του πηγαίου προγράμματος και επαναλαμβάνουν την ίδια διαδικασία της διερμηνείας για κάθε γραμμή, μέχρι το τέλος του προγράμματος. Εάν το πρόγραμμα πρέπει να εκτελεστεί πάλι, θα πρέπει η ίδια διαδικασία να επαναληφθεί πάλι από την αρχή και αυτό θα γίνεται για κάθε εκτέλεση του προγράμματος. Για τον λόγο αυτό οι interpreters είναι αργά εργαλεία στις γλώσσες προγραμματισμού.

Σύνοψη κεφαλαίου

Στο τελευταίο κεφάλαιο του συγγράμματος χρησιμοποιείται ένα από τα προγράμματα σε ψευδογλώσσα των προηγούμενων ενοτήτων, στο οποίο αναπτύσσεται μια αλγοριθμική δομή επανάληψης υπό συνθήκη και επιδεικνύεται, βήμα προς βήμα, εντολή προς εντολή, η διαδικασία μετατροπής του ψευδοκώδικα με ρητές εντολές στη γλώσσα προγραμματισμού C. Ακολούθως επεξηγείται και η σειρά των διαδικασιών που εκτελεί ένας προγραμματιστής για να φτάσει στο σημείο, ο πηγαίος του κώδικας (ο αλγόριθμός) να μετατραπεί σε ένα αναγνωρίσιμο και εκτελέσιμο πρόγραμμα από έναν υπολογιστή.

Ερωτήσεις αξιολόγησης

Ερώτηση-1: Περιγράψτε, για ποιον λόγο χρειαζόμαστε έναν απλό επεξεργαστή κειμένου στην δημιουργία ενός προγράμματος;

Ερώτηση-2: Περιγράψτε, τι σημαίνει πηγαίος κώδικας ενός προγράμματος;

Ερώτηση-3: Περιγράψτε, τι σημαίνει εκτελέσιμος κώδικας ενός προγράμματος;

Ερώτηση-4: Περιγράψτε, πως μπορούμε από τον πηγαίο κώδικα να πάμε στον εκτελέσιμο κώδικα σε μια γλώσσα προγραμματισμού;

Ερώτηση-5: Περιγράψτε, τι είναι οι μεταφραστές (compilers);

Ερώτηση-6: Περιγράψτε, τι είναι οι διερμηνευτές (interpreters);

Ερώτηση-7: Περιγράψτε, τι σημαίνει compilation;

Ερώτηση-8: Στην γλώσσα προγραμματισμού C, ποια είναι η σημασία της εντολής **int main()**;

Ερώτηση-9: Στην γλώσσα προγραμματισμού C, με ποια εντολή υλοποιούμε την δομή επανάληψης, συντάξτε την;

Ερώτηση-10: Στην γλώσσα προγραμματισμού C, για ποιον λόγο χρησιμοποιούμε την εντολή **#include**;

Απαντήσεις ερωτήσεων αξιολόγησης

Απάντηση-1: Συμβουλευτείτε την ενότητα 7.3. του κεφαλαίου.

Απάντηση-2: Συμβουλευτείτε την ενότητα 7.3. του κεφαλαίου.

Απάντηση-3: Συμβουλευτείτε την ενότητα 7.3. του κεφαλαίου.

Απάντηση-4: Συμβουλευτείτε την ενότητα 7.3. του κεφαλαίου.

Απάντηση-5: Συμβουλευτείτε την ενότητα 7.3. του κεφαλαίου.

Απάντηση-6: Συμβουλευτείτε την ενότητα 7.3. του κεφαλαίου.

Απάντηση-7: Συμβουλευτείτε την ενότητα 7.3. του κεφαλαίου.

Απάντηση-8: Συμβουλευτείτε την ενότητα 7.2. του κεφαλαίου.

Απάντηση-9: Συμβουλευτείτε την ενότητα 7.2. του κεφαλαίου.

Απάντηση-10: Συμβουλευτείτε την ενότητα 7.2. του κεφαλαίου.

Γλωσσάριο σημαντικών όρων

Ακέραιος

Ακολουθία ενεργειών

Αλγοριθμικές δομές επανάληψης

Αλγοριθμικές δομές επιλογής

Αλφάβητο γλώσσας

Αλφαριθμητικό

Αμυντικός προγραμματισμός

Ανάλυση

Αναζήτηση

Ανατροφοδότηση

Αντικειμενοστραφής προγραμματισμός

Αντικειμενοστραφείς γλώσσες

Αξιολόγηση αλγορίθμων

Απλή ακολουθία

Αποδοτικότητα αλγορίθμου

Αποδόμηση προβλήματος

Αποτελεσματικότητα

Απόφαση σε συνθήκη

Αριθμητικοί τελεστές

Αρχεία

Ατέρμων βρόχος

Βέλτιστοι αλγόριθμοι

Γεωμετρικός αριθμός

Γλώσσες Assembly

Γλώσσες προσανατολισμένες προς την μηχανή

Γλώσσες προσανατολισμένες προς τον άνθρωπο

Γλώσσες υψηλού επιπέδου

Γλώσσες χαμηλού επιπέδου

Γραμματική γλώσσας

Έκδοση

Έξοδος αλγορίθμου

Δεδομένα

Δεσμευμένες λέξεις

Διάγραμμα

Διάγραμμα ροής

Διάκριση δεδομένων εισόδου

Διαδικασία ανάπτυξης λογισμικού

Διαδικασία επίλυσης προβλημάτων

Διαδικασιακές γλώσσες

Διαδικαστικός προγραμματισμός

Διαχείριση δεδομένων

Διερμηνέας

Διερμηνευτές

Δομή

Δομή επανάληψης υπό συνθήκη

Δομή επιλογής

Δομημένος προγραμματισμός

Είσοδος αλγορίθμου

Εκτελέσιμος κώδικας

Εντολή

Επίδοση

Επεξεργασία δεδομένων

Επεξεργαστής κειμένου

Εργαλεία ανάπτυξης λογισμικού

Η επιστήμη της Πληροφορικής

Ηλεκτρονικός υπολογιστής Η/Υ

Καθορισμός ζητουμένων εξόδου

Καθοριστικότητα

Κατάστρωση σχεδίου επίλυσης προβλήματος

Κατανόηση προβλήματος

Κατηγοριοποίηση προβλημάτων

Κλασματικό μέρος

Κοινωνία της πληροφορίας

Κώδικας σε γλώσσα προγραμματισμού

Λέξεις κλειδιά

Λειτουργικό σύστημα

Λεξιλόγιο γλώσσας

Λογικοί τελεστές

Λογικό διάγραμμα

Λογισμικό

Μέγεθος μεταβλητής

Μεθοδολογία ανάπτυξης λογισμικού

Μετα-δεδομένα

Μεταβλητές

Μεταγλωττιστής

Μεταγλώττιση

Μεταφερσιμότητα

Μεταφραστής

Ολοκληρωμένα περιβάλλοντα ανάπτυξης

Όνομα μεταβλητής

Πεδίο

Περατότητα

Περιγραφικά ονόματα μεταβλητών

Περιεχόμενο μεταβλητής

Περιοχή μνήμης

Πηγαίος κώδικας

Πληροφορίες

Πραγματικός αριθμός

Προγραμματισμός υπολογιστών

Προγραμματιστές

Προδιαγραφή

Προσεγγιστικοί αλγόριθμοι

Προτεραιότητα

Προτεραιότητες τελεστών

Πρόγραμμα ηλεκτρονικού υπολογιστή

Πρότυπο Ε.C.M.A.

Ροή εισόδου

Ροή εξόδου

Σημασιολογία γλώσσας

Σταθερές

Συγκριτικοί τελεστές

Συμβολικές γλώσσες

Συναρτησιακές γλώσσες

Συντακτικά σφάλματα

Συντακτική δομή

Συντακτικό γλώσσας

Συστηματική παρατήρηση

Σχέδια προγράμματος

Σχεδίαση

Σχόλια

Τελεστής

Τελεσταίος

Τύπος μεταβλητής

Υλικό

Φυσική γλώσσα

Χαρακτήρας

Χρονική πολυπλοκότητα

Χρόνος εκτέλεσης

Χωρική πολυπλοκότητα

Ψευδογλώσσα

Ψευδοκώδικας

Βιβλιογραφία

ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ, Λουκάς Γεωργιάδης, Σταύρος Δ. Νικολόπουλος, Λεωνίδας Παληός, Σ.Ε.Α.Β., 2015.

Διαδικαστικός Προγραμματισμός (η γλώσσα C), Πάρις Μαστοροκώστας, Σ.Ε.Α.Β., 2015.

Εισαγωγή στην Επιστήμη των Υπολογιστών & Επικοινωνιών, Δημήτρης Δρόσος, Δημοσθένης Βουγιούκας, Εμμανουήλ Καλλίγερος, Σπυρίδων Κοκολάκης, Χαράλαμπος Σκιάνης, Σ.Ε.Α.Β., 2015.

Στοιχεία Τεχνολογίας Λογισμικού, Βασίλειος Βεσκούκης, Σ.Ε.Α.Β., 2015.

Σχεδίαση και Ανάλυση Αλγορίθμων, Κωνσταντίνος Τσίχλας, Ιωάννης Μανωλόπουλος, Αναστάσιος Γούναρης, Σ.Ε.Α.Β., 2015.

Introduction to Algorithms, Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, Massachusetts Institute of Technology, 2009.

STANDARD ECMA-4 FLOW CHARTS, E.C.M.A., 1966.

STANDARD FOR SOFTWARE QUALITY ASSURANCE PLANS, A.N.S.I./I.E.E.E., 1989.

STANDARD GLOSSARY OF SOFTWARE ENGINEERING TERMINOLOGY, A.N.S.I./I.E.E.E., 1990.

The Art of Computer Programming, Donald Knuth, Addison-Wesley, 1968.

THE C PROGRAMMING LANGUAGE, Brian W. Kernighan, Dennis M. Ritchie, Prentice-Hall International, Bell Labs, 1978.

Οδηγίες για περαιτέρω μελέτη

Ο προγραμματισμός των ηλεκτρονικών υπολογιστών είναι μια απαιτητική δραστηριότητα, η οποία μπορεί να αποκτηθεί μόνο με συστηματική μελέτη, επιμονή και συνεχή ενασχόληση. Επιπλέον όμως και για τους επαγγελματίες εκπαιδευτικούς της επιστήμης της *Πληροφορικής*, η δημιουργία των νέων και ικανών προγραμματιστών είναι επίσης μια απαιτητική και σύνθετη διαδικασία. Η μετάδοση των γνώσεων προγραμματισμού και η μεταλαμπάδευση των απαραίτητων εμπειριών και αντιλήψεων, είναι ένα πολυδιάστατο και δύσκολο έργο που απαιτεί κόπο αλλά κυρίως, τρόπο. Παρόλο τον μεγάλο αριθμό των σχολών, όλων των βαθμίδων, οι οποίες παράγουν τεχνικούς λογισμικού και προγραμματιστές, είναι κοινή η αντίληψη, στις κοινότητες των εταιριών διαχείρισης ανθρωπίνου δυναμικού σε ευρωπαϊκό επίπεδο, για το κενό που υπάρχει στον αριθμό των προγραμματιστών που έχουν τις απαραίτητες ικανότητες για να αναλάβουν και να φέρουν σε πέρας απαιτητικές κατασκευές λογισμικού.

Στα πλαίσια των προσπαθειών για περαιτέρω ενασχόληση, καλούνται οι αναγνώστες εκπαιδευόμενοι, νέοι προγραμματιστές, μετά την ολοκλήρωση της μελέτης του συγγράμματος,

- *Προγραμματισμός Ι (βασικό επίπεδο), Ανακαλύπτοντας τον Προγραμματισμό,*

να συνεχίσουν και με τα επόμενα συγγράμματα της σειράς, *Προγραμματισμός Ηλεκτρονικών Υπολογιστών & Μηχανών, της ΑΣΤΑ,*

- *Προγραμματισμός ΙΙ (ενδιάμεσο επίπεδο), Κατανοώντας τον Προγραμματισμό,*
- *Προγραμματισμός ΙΙΙ (προχωρημένο επίπεδο), Αξιοποιώντας τον Προγραμματισμό.*