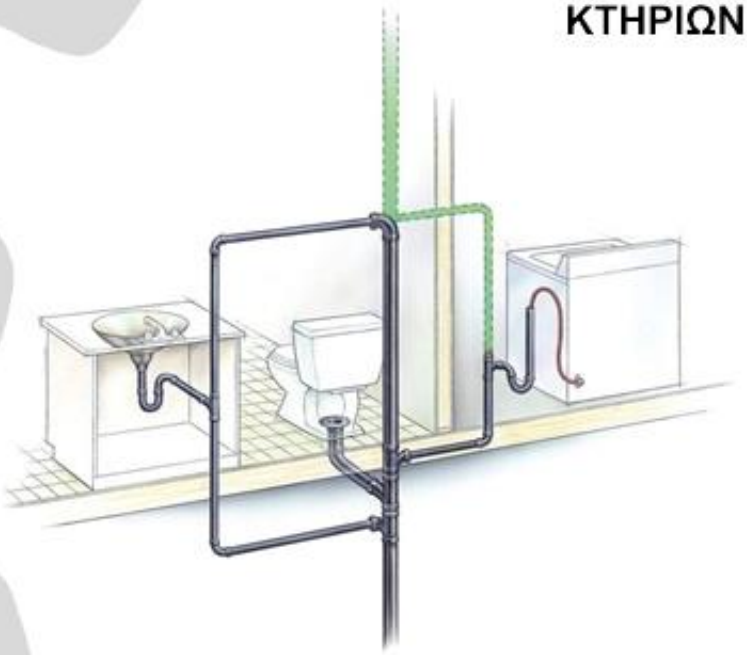
ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ ΠΑΡΑΓΩΓΗΣ & ΔΙΟΙΚΗΣΗΣ
ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΚΟΠΗΣ & ΚΑΤΑΣΚΕΥΑΣΤΙΚΗΣ ΠΡΟΣΟΜΟΙΩΣΗΣ

ΧΡΗΣΗ ΤΗΣ AUTOLISP ΓΙΑ ΤΗ ΣΧΕΔΙΑΣΗ ΜΕΛΕΤΩΝ ΥΔΡΕΥΣΗΣ ΚΑΙ ΑΠΟΧΕΤΕΥΣΗΣ ΚΤΗΡΙΩΝ



ΚΑΡΥΔΙΑΝΑΚΗ ΜΑΡΙΑ

ΕΠΙΒΛΕΠΩΝ: ΑΡΙΣΤΟΜΕΝΗΣ ΑΝΤΩΝΙΑΔΗΣ
ΚΑΘΗΓΗΤΗΣ

ΑΡ. ΔΙΠΛ. : 101

ΧΑΝΙΑ 2022

ΠΕΡΙΕΧΟΜΕΝΑ

1. ΕΙΣΑΓΩΓΗ

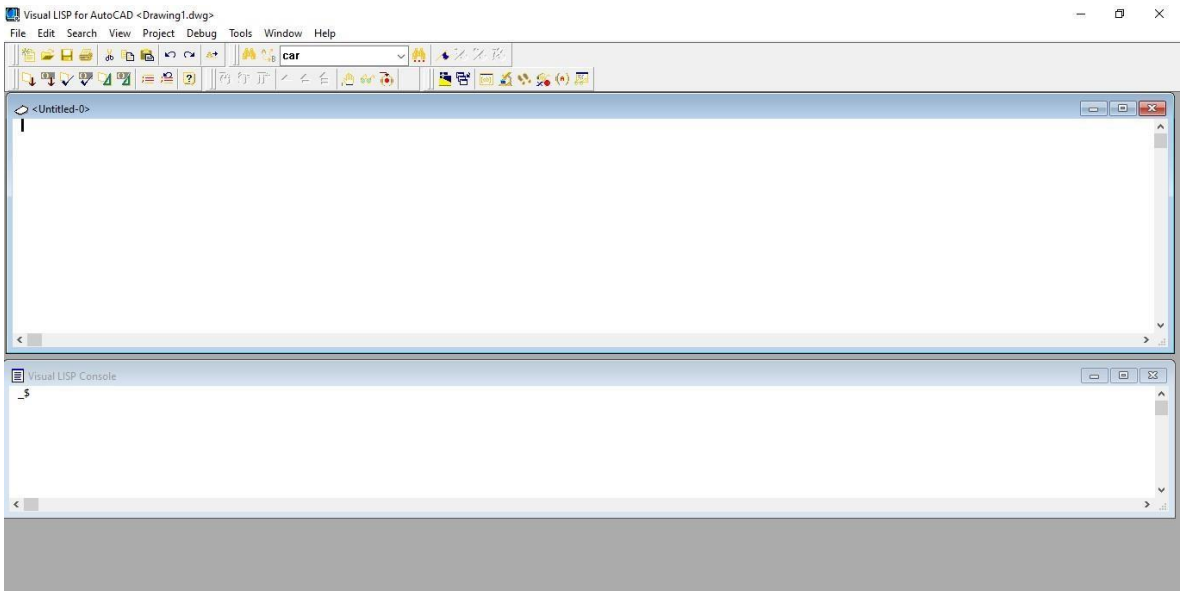
Η AutoLISP είναι μια γλώσσα προγραμματισμού Lisp που έχει δημιουργηθεί ειδικά για τη χρήση του AutoCAD και των παραγώγων του με σκοπό την επιτάχυνση και την αυτοματοποίηση αυτών. Η παρούσα διπλωματική εργασία, απευθύνεται σε χρήστες που καλό θα ήταν να έχουν ήδη έρθει σε επαφή με το σχεδιαστικό περιβάλλον του AutoCAD και έτσι διαθέτοντας τις γνώσεις πάνω στις βασικές εντολές σχεδίασης τώρα αναζητούν μέσω του προγραμματισμού χρήσιμες εντολές που θα ανέπτυσσαν κι άλλο το περιβάλλον αυτό. Κατά τη διάρκεια της ανάγνωσής σας θα συναντήσετε αρκετά παραδείγματα καθώς και τα βήματα ενός κώδικα σε απεικόνιση, με σκοπό τόσο την καλύτερη και ταχύτερη κατανόηση κάποιων βασικών αλλά και πιο εξελιγμένων εντολών και διαδικασιών, όσο και την εκτέλεσή τους μέσα στα ίδια τα παραδείγματα. Τα συμπεράσματα και η θεωρία που έχουν διατυπωθεί, προέκυψαν κατόπιν της χρήσης του AutoCAD Electrical 2020. Ωστόσο, από έκδοση σε έκδοση ή από είδος σε είδος AutoCAD (εδώ Electrical) μπορεί να έχουν προστεθεί ή αφαιρεθεί κάποιες δυνατότητες αλλάζοντας ή εξελίσσοντας κάποιες διαδικασίες. Παρόλα αυτά, όλα τα είδη των εντολών μπορούν να αναζητηθούν μέσω λέξεων κλειδιών στο πλαίσιο αναζήτησης “Type a keyword or phrase” για την κάθε έκδοση αντίστοιχα.

Ιστορικά, η AutoLISP προήλθε από μια πρώιμη έκδοση του XLISP, η οποία δημιουργήθηκε από τον David Betz και εισήχθη στην τότε έκδοση 2.18 του AutoCAD τον Ιανουάριο του 1986 η οποία εξελισσόταν και ενισχύονταν σε διαδοχικές εκδόσεις έως την 13η τον Φεβρουάριο του 1995. Αργότερα η ανάπτυξή της λίγο παραμελήθηκε λόγω της εμφάνισης πιο μοντέρνων περιβαλλόντων ανάπτυξης, παρέμεινε ωστόσο η κύρια γλώσσα προσαρμογής του χρήστη στο AutoCAD. Ο κώδικας AutoLISP δεν είναι ένα «στεγνό» προγραμματιστικό περιβάλλον, αλλά προσφέρει άμεση αλληλεπίδραση με τον χρήστη μέσω του γραφικού επεξεργαστή του AutoCAD χρησιμοποιώντας συναρτήσεις που επιτρέπουν στον χρήστη να επιλέγει σημεία, να επιλέγει αντικείμενα στην οθόνη, να εισάγει αριθμούς, κείμενα και άλλα δεδομένα δημιουργώντας κάθε φορά αυτό που τον εξυπηρετεί. Κάθε εντολή ενός προγράμματος AutoCAD έχει στηθεί από μια AutoLISP, έτσι με την πρόσβαση του ίδιου του χρήστη σε αυτή τη γλώσσα μπορεί είτε να αλλάξει τις ήδη υπάρχουσες εντολές, είτε να τις εξελίξει διευκολύνοντας και επιταχύνοντας την καθημερινή του εργασία.

2. ΒΑΣΙΚΕΣ ΕΝΝΟΙΕΣ

Πριν ξεκινήσει η ανάλυση της AutoLISP βασική προϋπόθεση είναι να έχει γίνει εγκατάσταση του προγράμματος AutoCAD ή οποιουδήποτε από τα παράγωγά του, για παράδειγμα AutoCAD Electrical. Αφού ολοκληρωθεί η εγκατάστασή του, ανοίγοντας το πρόγραμμα στο παράθυρο start, επιλέγεται στο Get Started το Start Drawing ώστε να ανοίξει ο βασικό περιβάλλον σχεδίασης. Όταν γίνει η μετάβαση στο περιβάλλον αυτό, στο ανώτατο σημείο του παραθύρου παρουσιάζονται τα εξής, μια γραμμή εργαλείων (line, polyline*, circle κλπ.), μια γραμμή περιεχομένων (home, project,schematic κλπ.) και ακόμα μία γραμμή ακριβώς από πάνω με τις επιλογές (File,Edit,View,Insert,Format, Tools κλπ.). Για να ανοίξει το βασικό περιβάλλον σύνταξης της AutoLISP,το οποίο φαίνεται και παρακάτω στο σχήμα 2.1 ακολουθούνται τα βήματα, **Tools→AutoLISP→Visual LISP**. Για να ξεκινήσει η σύνταξη ενός κομμάτι κώδικα, επιλέγεται από τη γραμμή εργαλείων το File και στη συνέχεια New File που ανοίγει τον συντάκτη, με τίτλο “<Untitled 0>”,το όνομα του οποίου θα δοθεί από το χρήστη αποθηκεύοντάς το στον υπολογιστή του, με το πάτημα του εικονιδίου της δισκέτας που βρίσκεται πάνω και αριστερά στη γραμμή εργαλείων της AutoLISP (κάτω από τη λέξη Search). Προσοχή, το όνομα με το οποίο θα αποθηκευτεί ο αρχείο να είναι της μορφής «όνομα.LISP» και όσο γίνεται πιο αναγνωριστικό του περιεχομένου του για να μην δημιουργείται σύγχυση αργότερα. Προτιμάται όλα τα αρχεία LISP να βρίσκονται σε κοινό φάκελο για τη διευκόλυνση του χρήστη. Εκτός από το συντάκτη, υπάρχει εξαρχής ανοικτό και ένα παράθυρο Visual LISP Console που αποτελεί το *μεταφραστή*, μέσα στον οποίο εκτελούνται οι εντολές, εφόσον έχουν συνταχθεί σωστά και δίνει τα αποτελέσματα αυτών, που θα αναλυθούν και στη συνέχεια μέσω παραδειγμάτων.

Σε περίπτωση που υπάρχει διαθέσιμο ήδη κάποιο αρχείο αποθηκευμένο, για να γίνει η φόρτωση/εκτέλεσή του, ακολουθούνται τα βήματα **Tools→AutoLISP→Load Applications** επιλέγεται το κατάλληλο αρχείο. Επομένως, καλώντας στο command window το όνομα αυτού του αρχείο, πλέον αναγνωρίζεται και ξεκινάει να εκτελεί τις εντολές που περιέχει.



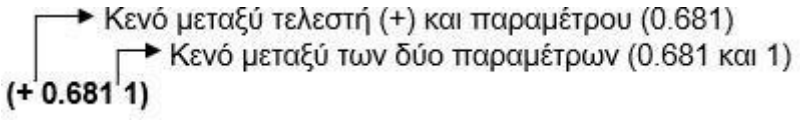
Σχήμα 2.1: Περιβάλλον σύνταξης της AutoLISP,πάνω παράθυρο συντάκτης, κάτω παράθυρο μεταφραστής.

*****Οι polylines είναι σύνθετες οντότητες οι οποίες αποτελούνται από πολλές οντότητες και μπορούν να διασπαστούν σε 3 βασικά συστατικά, κορυφές, γραμμές και τόξα.

Για τη σύνταξη ενός σχολίου μέσα στον κώδικα, γράφεται ένα ελληνικό ερωτηματικό **;** ακριβώς πριν ξεκινήσει η έκφραση που θέλουμε να μείνει ως σχόλιο και να μην διαβαστεί-εκτελεστεί. Η χρήση των σχολίων, σε συνδυασμό με τη χρήση του **prettyprint** βοηθάει στη δημιουργία ενός πιο ευανάγνωστου και οργανωμένου κώδικα. Με το prettyprint χρησιμοποιούνται κατά τη σύνταξη περιθώρια που διαχωρίζουν το κομμάτι του κώδικα οργανώνοντάς το οπτικά.
Αν οποιαδήποτε στιγμή θελήσει ο χρήστης να δει τι τιμή έχει πάρει κάποια μεταβλητή μέχρι εκείνη την ώρα, αρκεί να γράψει το όνομα της μεταβλητής με ένα θαυμαστικό μπροστά, **!όνομα_μεταβλητής** και αυτόματα θα εμφανιστεί το περιεχόμενό της μέχρι εκείνη τη στιγμή. Στο παράθυρο "Visual LISP Console" η τιμή εμφανίζεται και χωρίς το θαυμαστικό απλά γράφοντας μόνο το όνομα της μεταβλητής.
Στην AutoLISP οι δεκαδικοί αριθμοί γράφονται ως 1.5 και όχι 1,5 δηλαδή το κόμμα για τα δεκαδικά είναι η τελεία ενώ το κανονικό κόμμα χρησιμοποιείται σε λίστες όπως για παράδειγμα point1=(x1,y1).

2.1 Μια έκφραση

Στην AutoLISP κάθε έκφραση πρέπει να αποτελείται από κάποιον τελεστή δηλαδή μια οδηγία, όπως πρόσθεση(+), αφαίρεση(-), πολλαπλασιασμός(*), διαίρεση(/) που θα δώσει ένα συγκεκριμένο αποτέλεσμα και από μία ή παραπάνω παραμέτρους και συντάσσεται ως,



όπου ο τελεστής είναι η πράξη + που ζητείται να γίνει μεταξύ των παραμέτρων 0.681 και 1 και θα δώσει σαν αποτέλεσμα την πρόσθεση αυτών των δύο, δηλαδή 1.681. Να σημειωθεί πως πάντα γράφεται υποχρεωτικά 0.681 και όχι .681 αλλιώς εμφανίζεται μήνυμα λάθους. Το αποτέλεσμα αυτό θα δοθεί όταν η έκφραση (+ 0.681 1) συνταχθεί εντός του μεταφραστή Visual LISP Console, όπου προσοχή, πρέπει όσες παρενθέσεις ανοίγονται τόσες πρέπει κλείνονται κιόλας διαφορετικά θα εμφανιστεί ένα μήνυμα της μορφής « (_> » που δηλώνει ότι λείπει μια παρένθεση δεξιά για να κλείσει την αρχική παρένθεση, ή αν λείπουν δύο παρενθέσεις δεξιά θα εμφανίσει « ((_ > » και ούτω καθεξής. Ιδιαίτερη προσοχή πρέπει να δοθεί στη γενικότερη σύνταξη μιας έκφρασης καθώς είναι υποχρεωτικό να υπάρχουν κενά διαστήματα μεταξύ όλων των στοιχείων της, δηλαδή μεταξύ συνάρτησης (+) και του χαρακτήρα (0.681), καθώς μεταξύ των ίδιων των παραμέτρων, του 0.681 και του 1. Το κενό ωστόσο μεταξύ των παρενθέσεων και των παραμέτρων δεν είναι απαραίτητο αλλά διευκολύνει την αναγνωσιμότητα της έκφρασης.
Οι παράμετροι μιας έκφρασης δεν είναι απαραίτητα απλοί αριθμοί, μπορεί να είναι και ολόκληρες εκφράσεις, για παράδειγμα η παραπάνω πράξη μπορεί να χρησιμοποιηθεί εξολοκλήρου σαν μια παράμετρος εντός μιας άλλης πράξης όπως,

(/2 (+ 0.681 1))

Οι πράξεις εκτελούνται από μέσα προς τα έξω, άρα πρώτα θα εκτελεσθεί η εσωτερική παρένθεση δίνοντας το αποτέλεσμα 1.681 όπως παρουσιάστηκε παραπάνω και έτσι η έκφραση απλοποιείται σε (/ 2 1.681) δηλαδή εκτελεί τη διαίρεση μεταξύ του 2 και του 1.681και επομένως το τελικό αποτέλεσμα που δίνεται από αυτήν την έκφραση είναι ίσο με 2 / 1.681 =1.18997.

Στην AutoLISP γίνεται η χρήση μιας σειράς από διάφορα είδη δεδομένων όπως είναι, οι ακέραιοι και πραγματικοί αριθμοί, τα αλφαριθμητικά στοιχεία, οι λίστες, τα ονόματα αντικειμένων, ομάδες επιλεγμένων αντικειμένων καθώς και τα σύμβολα και οι υπορουτίνες τα οποία θα αναλυθούν παρακάτω.

2.2 Αριθμοί

Όσον αφορά τους αριθμούς, μια σημαντική παρατήρηση είναι πως η AutoLISP αποκόπτει τα δεκαδικά ψηφία, οπότε όταν μια έκφραση εκτελεί οποιαδήποτε πράξη μεταξύ αριθμών που είναι ακέραιοι τότε το αποτέλεσμα που θα δοθεί θα είναι υποχρεωτικά ένα ακέραιος αριθμός ακόμα κι αν από μαθηματικής απόψεως το αποτέλεσμα είναι δεκαδικός αριθμός. Δεκαδικά θα δοθούν εφόσον η έκφραση περιέχει μέσα πράξεις με νούμερα που περιέχουν δεκαδικό μέρος. Δηλαδή, η παραπάνω πράξη (/ 2 1.681) θα δώσει αποτέλεσμα 1.18997 ενώ αν η έκφραση είχε ως (/ 5 2) το αποτέλεσμα που δίνει είναι το 2 και όχι το 2.5 που είναι το πραγματικό αποτέλεσμα της πράξης. Για την αποφυγή αυτού του προβλήματος γράφουμε τις παραμέτρους σε μορφή πραγματικών αριθμών δηλαδή (/ 5.0 2.0) = 2.5.

2.3 Αλφαριθμητικά, Strings

Με τον όρο αλφαριθμητικά ή αλλιώς Strings, αναφερόμαστε στα κείμενα, μηνύματα και εκφράσεις που χρησιμοποιούνται μέσα στην AutoLISP, όπως για παράδειγμα

(strcat “the length is” “seven meters”)

Με την εντολή **strcat** γίνεται η συγχώνευση των δύο επιμέρους κειμένων σε ένα και έτσι επιστρέφεται η ολοκληρωμένη φράση “the length is seven meters”, ωστόσο η συγχώνευση γίνεται μεταξύ των αλφαριθμητικών και όχι μεταξύ κάποιου αλφαριθμητικού και κάποιου αριθμού. Υπάρχει επίσης και η συνάρτηση **strlen** που βρίσκει τον αριθμό των χαρακτήρων σε ένα string, (strlen string) και επιστρέφει μια ακέραια τιμή. Προσοχή στο ότι και τα κενά διαστήματα υπολογίζονται ως χαρακτήρες. Για την εισαγωγή αλφαριθμητικών από το χρήστη χρησιμοποιείται η συνάρτηση **getstring** που θα αναλυθεί παρακάτω. Εκτός από τα strings υπάρχουν και τα substrings ή αλλιώς υπό-εκφράσεις, μια ακολουθία χαρακτήρων που βρίσκονται εσωτερικά του κυρίως string. Αυτά εξάγονται από την κύρια έκφραση με τη συνάρτηση **Substr** η σύνταξη της οποίας είναι,

(substr string αρχή_του_υποstring μήκος του υποstring)

Η πρώτη παράμετρος (string) αναφέρεται στην έκφραση από την οποία θέλω να πάρω ένα κομμάτι της, η δεύτερη παράμετρος (αρχή_του_υποstring) είναι η θέση που ξεκινάει το επιμέρους αυτό κομμάτι και η τρίτη το μήκος του substring.
Για τα αλφαριθμητικά η λέξη EXAMPLE με τη λέξη example είναι δύο διαφορετικά παραδείγματα, ένα σημείο που θέλει προσοχή καθώς σε περίπτωση που χρειαστεί να αντιστοιχηθούν μεταξύ τους θα προκύπτει σφάλμα.
Σημείωση: Τα αλφαριθμητικά μηνυμάτων δεν μπορούν να περιέχουν περισσότερους από 100 χαρακτήρες και τότε πιθανόν να εμφανιστεί κάποιο μήνυμα λάθους.

Συνάρτηση Rtos

Η συνάρτηση Rtos χρησιμοποιείται για να μετατρέψει έναν αριθμό σε αλφαριθμητικό (string) και συντάσσεται,

(rtos πραγματικός/ακέραιος_αριθμός κωδικός_μοφής_μονάδων ακρίβεια)

Ο πραγματικός/ακέραιος_αριθμός είναι αυτός που θα μετατραπεί σε string. Ο κωδικός_μορφής_μονάδων ανάλογα με τον αριθμό που θα δοθεί γίνεται η αντίστοιχη μορφή στην οποία θα μετατραπεί, για παράδειγμα αν δοθεί ο κωδικός 2 θα μετατραπεί σε δεκαδική μορφή και ούτω καθεξής όπως φαίνεται στον πίνακα 2.1. Η τελευταία παράμετρος προσδιορίζει την ακρίβεια δηλαδή πόσα δεκαδικά ψηφία θα κρατηθούν για να μετατραπούν.

Κωδικός	Μορφή
1	Επιστημονική
2	Δεκαδική
3	Πόδια και δεκαδικές ίντσες
4	Πόδια και ίντσες
5	Κλασματικές μονάδες

Πίνακας 2.1: Κωδικοί Μορφών Μονάδων για την Rtos

Συνάρτηση Angtos

Η συνάρτηση Angtos μοιάζει συντακτικά με την Rtos,

(angtos αριθμός κωδικός_μορφής_μονάδων ακρίβεια)

Στον πίνακα 2.2 φαίνονται κάποιες άλλες συναρτήσεις που μετατρέπουν τα δεδομένα στους αντίστοιχους τύπους,

Συνάρτηση	Χρήσεις/λειτουργίες
(Angtos πραγματικός)	Μετατρέπει πραγματικούς αριθμούς (σε ακτίνια) σε αλφαριθμητικά
(Ascii string)	Μετατρέπει string στον αντίστοιχο κωδικό χαρακτήρα ASCII
(Atoi string)	Μετατρέπει string σε ακέραιο
(Itoa ακέραιος)	Μετατρέπει ακέραιο σε string
(Chr ακέραιος)	Μετατρέπει σε string έναν ακέραιο που συμβολίζει κάποιο κωδικό χαρακτήρα ASCII

Πίνακας 2.2: Συναρτήσεις μετατροπής δεδομένων

Για παράδειγμα, η έκφραση (itoa 20) μετατρέπει τον ακέραιο 20 σε κάποιο string.Οι κωδικοί ASCII είναι αριθμητικές τιμές που αντιπροσωπεύουν γράμματα, αριθμούς και σύμβολα, για παράδειγμα οι αριθμοί από το 65 έως το 90 αντιστοιχούν το καθένα σε ένα γράμμα της αλφαβήτου, το 65 αντιστοιχεί στο A το 66 στο B κλπ. Χρησιμοποιούνται για αναπαράσταση κειμένου στους υπολογιστές, σε συσκευές τηλεπικοινωνίας και σε άλλες συσκευές που δουλεύουν με κείμενο καθώς είναι ένα κωδικοποιημένο σύνολο χαρακτήρων του λατινικού αλφάβητου.

2.4 Λίστες

Όταν τα δεδομένα εμπεριέχονται μεταξύ παρενθέσεων, τότε προκύπτει μία λίστα ακόμα και το () είναι μία λίστα με μηδενικό αριθμό στοιχείων. Υπάρχουν δύο κατηγορίες, αυτές που προορίζονται για αποτίμηση από κάποιο πρόγραμμα της AutoLISP (οι εκφράσεις ή φόρμες) και αυτές που λειτουργούν σαν χώροι αποθήκευσης δεδομένων (πχ λίστες συντεταγμένων). Μια λίστα μπορεί να περιλαμβάνει οποιονδήποτε αριθμό είτε ακέραιο είτε πραγματικό, κάποιο αλφαριθμητικό ή ακόμα και κάποια άλλη λίστα και μπορεί να χρησιμοποιηθεί ως

μέσο αποθήκευσης κάποιων δεδομένων όπως οι συντεταγμένες στις δύο διαστάσεις και είναι της μορφής, (1.5 3) ή πιο γενικευμένα της μορφής (A B) με το 1.5 να αναφέρεται στον x και το 3 να αναφέρεται στον y άξονα. Τα x και y ονομάζονται άτομα καθώς μπορούν να αποσυντεθούν από τη λίστα και το καθένα να αποτελέσει ένα ξεχωριστό δεδομένο και οι ίδιες οι τιμές αυτών, δηλαδή τα νούμερα 1.5 και 3 *δεν* θεωρούνται άτομα καθώς δεν μπορούν να διαχωριστούν περεταίρω.

Λίστες χαρακτηριστικών

Οι λίστες χαρακτηριστικών αποτελούνται από άλλες λίστες των οποίων το πρώτο στοιχείο είναι ένας ακέραιος κωδικός που αντιπροσωπεύει ένα συγκεκριμένο χαρακτηριστικό όπως ((“osmode” 0)(“orthomode”1) όπου τα osmode και orthomode είναι τύποι των Osnaps που αναλύονται παρακάτω στο κεφάλαιο 4. Ο πίνακας 2.3 δείχνει τους κωδικούς αυτούς αλλά και τη σημασία τους,

Κωδικός	Σημασία
1	Όνομα οντότητας
0	Είδος οντότητας (“Text”,”Line”…κλπ)
7	Στυλ γραφής
8	Layer
10	Insertion point(σημείο εισαγωγής)
11	Κεντρικό σημείο στοίχισης (για αντικείμενο κεντραρισμένο)
21	Δεξιό σημείο στοίχισης (για στοιχισμένο κείμενο στα δεξιά)
31	Δεύτερο σημείο στοίχισης
40	Ύψος κειμένου
41	Χ συντελεστής κλίμακας
50	Γωνία περιστροφής
51	Γωνία κλίσης
71	Κωδικός ανεστραμμένου κειμένου
72	Κωδικός ευθυγράμμισης κειμένου
210	Μέγεθος ανύψωσης στην x,y,ή z διεύθυνση

Πίνακας 2.3: Κωδικοί λιστών χαρακτηριστικών

Η συνάρτηση setq

Με τη χρήση της συνάρτησης setq αντιστοιχίζονται τιμές στις μεταβλητές που χρησιμοποιούμε, για παράδειγμα με τη γραμμή εντολής **(setq x 1.5)** γίνεται η αντιστοίχιση της τιμής 1.5 στη μεταβλητή x. Τα βήματα για τη σύνταξη και την αναγνώριση μιας εντολής όπως η setq έχουν ως εξής,

- **Βήμα 1:** γράφω την εντολή (setq x 1.5) στον συντάκτη, δηλαδή το αρχείο που άνοιξε μέσω του File→New File. Αφού αποθηκευτεί στον υπολογιστή από το χρήστη με το κατάλληλο όνομα ακολουθεί το επόμενο βήμα.
- **Βήμα 2:** Με το ποντίκι μαρκάρεται από το χρήστη όλη η εντολή και επιλέγει από την γραμμή εργαλείων την ενέργεια “Load selection”(δεύτερο εικονίδιο από αριστερά στη δεύτερη εργαλείων).

Με το βήμα αυτό, γίνεται φόρτωση της συνάρτησης setq και εκτελεί την ενέργεια που της ζητήσαμε. Σε αυτό το στάδιο αν είχαμε κάποιο συντακτικό λάθος, θα έβγαζε μήνυμα λάθους ακριβώς από κάτω στο παράθυρο του μεταφραστή, οπότε είναι και ένας άμεσος και εύκολος

τρόπος για να γίνει έλεγχος για τυχόν συντακτικά λάθη ενώ ταυτόχρονα φορτώνονται και εκτελούνται οι εντολές.

- **Βήμα 3:** Σε αυτό το βήμα, δεδομένου ότι πλέον έχει εκτελεσθεί η setq, γράφοντας στο παράθυρο του μεταφραστή την τιμή x και πατώντας enter θα εμφανιστεί η τιμή που μόλις δόθηκε σε αυτό, δηλαδή 1,5.

Να σημειωθεί πως το x κάθε φορά θα κρατάει την τελευταία τιμή που θα του δοθεί. Δηλαδή αν γραφτούν διαδοχικά δύο setq και στην πρώτη του δοθεί η τιμή 1.5 ενώ στην αμέσως επόμενη setq του δοθεί η τιμή 5, αυτή που τελικά θα κρατήσει θα είναι το 5.

Οι συναρτήσεις Setvar,Quote

Η συνάρτηση **Setvar**, αλλάζει τις τιμές των μεταβλητών που βρίσκονται στο σύστημα δηλαδή, αν η εντολή ήταν

(setvar "snapunit" (15 15))

τότε δέχεται σαν παράμετρο τη μεταβλητή του συστήματος που θέλει να αλλάξει, εδώ το snapunit, και προσπαθεί να το αλλάξει σε (15 15) να δώσει δηλαδή στο snap τις αποστάσεις 15x15. Όμως επειδή η AutoLISP έχει την τάση προτού περάσει μια παράμετρο σε κάποια συνάρτηση, να προσπαθεί να βρει την τιμή της, με την εντολή αυτή εκτός από την ανάθεση προσπαθεί ταυτόχρονα να προσδιορίσει και την τιμή του (1515). Στην παρούσα κατάσταση όμως αυτό που ζητείται είναι να αλλάξει την παράμετρο αντικαθιστώντας σε αυτήν όλη τη λίστα μαζί, δηλαδή όλο το (15,15). Για την αποφυγή αυτού του προβλήματος, χρησιμοποιείται τη συνάρτηση **Quote,** προσθέτοντας για συντομία μια απόστροφο από τοστοιχείο που ζητείται να μην προσπαθήσει να αλλάξει η AutoLISP και η εντολή γίνεται,

(setvar "snapunit" '(15 15))

και έτσι τώρα θα περάσει ολόκληρη η λίστα μέσα στο snapunit.

Βασικές συναρτήσεις για διαχείριση λιστών

Ο πίνακας 2.4. περιέχει μερικές από τις βασικές συναρτήσεις που είναι απαραίτητο να γνωρίζουμε ώστε να μπορούμε να επεξεργαστούμε-διαχειριστούμε μια λίστα.

Συνάρτηση	Χρήση
(setq y (list 5 6 7 8 9 10))	Αναθέτει στη μεταβλητή y το περιεχόμενο της λίστας (5 6 7 8 9 10)
(car λίστα)	Δίνει το πρώτο στοιχείο της λίστας (εδώ για παράδειγμα της y) δηλαδή το 5
(cadr λίστα)	Δίνει το δεύτερο στοιχείο της λίστας, δηλαδή το 6
(cdr λίστα)	Δίνει όλα τα στοιχεία της λίστας εκτός από το πρώτο, δηλαδή (6,7,8,9,10)
(mapcar συνάρτηση λίστα1 λίστα2….λίσταn)	Εφαρμόζει τα στοιχεία των n λιστών σαν παραμέτρους σε κάποια συνάρτηση. Κάθε στοιχείο σε κάθε λίστα επεξεργάζεται μέχρι το τέλος και της τελευταίας λίστας.
(apply συνάρτηση λίστα)	Εφαρμόζει όλο το περιεχόμενο μίας λίστας σε μία συνάρτηση.
(foreach σύμβολο λίστα έκφραση)	Διαβάζει κάθε στοιχείο της λίστας και το αντιστοιχεί στο σύμβολο. Η foreach μετά αποτιμά μια έκφραση που περιέχει αυτό το

	σύμβολο, δηλαδή αποτιμά μια έκφραση μέχρι το τέλος της λίστας.
(reverse λίστα)	Αντιστρέφει τη σειρά των στοιχείων της λίστας.
(cons στοιχείο λίστα)	Προσθέτει σε μία λίστα ένα στοιχείο οποιουδήποτε τύπου δεδομένου.
(append λίστα1 λίστα2…..λίσταn)	Δέχεται έναν οποιοδήποτε αριθμό n λιστών και συνθέτει τα στοιχεία τους σε μία λίστα
(last λίστα)	Δίνει το τελευταίο στοιχείο μίας λίστας
(length λίστα)	Δίνει τον αριθμό των στοιχείων μίας λίστας
(member *στοιχείο* λίστα)	Βρίσκει το κομμάτι της λίστας (υπό-λίστα) που αρχίσει με το «*στοιχείο*» και φτάνει μέχρι το τέλος της λίστας.
(nth ακέραιος λίστα)	Βρίσκει ένα στοιχείο της λίστας που να είναι ακέραιος αριθμός. Το πρώτο αντικείμενο σε μία λίστα έχει τον αριθμό 0.
(setq macro '(command))	Η συνάρτηση Macro είναι μια μεταβλητή λίστας που χρησιμοποιείται για να αποθηκεύσει ότι πληκτρολογείται από τον χρήστη, δηλαδή οι πληκτρολογήσεις θα προστεθούν στο τέλος αυτής της λίστας και στο τέλος θα έχει δημιουργηθεί μια έκφραση εντολής

Πίνακας 2.4: Βασικές συναρτήσεις διαχείρισης λιστών

Παράδειγμα της foreach που γενικά συνηθίζεται να χρησιμοποιείται για να ελέγχει τα στοιχεία μιας λίστας,

```
(foreach n
'((4.00 1.00) (10.09 1.01) (11.96 6.80) (7.03 10.38) (2.11 6.79) (4.00 1.00))
(if (>2 (cadr n)2)(Setq nealista (append nealista (list n))))
)
```

Σχήμα 2.2: Παράδειγμα foreach

Αυτό που κάνει είναι να ελέγχει όλα τα δεύτερα στοιχεία της κάθε λίστας (επειδή χρησιμοποιεί την cadr), δηλαδή κάθε συντεταγμένη y, εάν είναι μεγαλύτερα από το 2. Αν ισχύει αυτή η συνθήκη, τότε προστίθενται σε μια nealista που θα περιέχει μόνο αυτές τις y συντεταγμένες.

Προσοχή, οι συναρτήσεις θα πρέπει να χρησιμοποιούνται σε μορφές **quoted** (με απόστροφο) δηλαδή, (cdr '(d k m))→ (k m)

διαφορετικά η AutoLISP και πάλι θα μπει σε διαδικασία να αποτιμήσει τη λίστα (d k m) γιατί το πρώτο στοιχείο της λίστας δεν είναι κάποια συνάρτηση αλλά μία μεταβλητή.

Λίστες «ζεύγος τελείας»

Εάν η γενική μορφή μιας λίστας είναι (A B) τότε η λίστα «ζεύγος τελείας» είναι της μορφής (A . B) και δημιουργείται χρησιμοποιώντας τη συνάρτηση **Cons**, (cons ' A ' (B)) όπου η πρώτη παράμετρος το A είναι το στοιχείο που θα προστεθεί στην αρχή μίας λίστας και το δεύτερο στοιχείο το B είναι η λίστα αυτή στην οποία θα προστεθεί. Αν η δεύτερη παράμετρος δεν ήταν λίστα δηλαδή (cons 'A 'B) τότε επιστρέφεται το ζεύγος τελείας (A . B).

Άλλες επιπλέον συναρτήσεις,

- **Η συνάρτηση Append**

Ένα παράδειγμα σύνταξης αυτής της συνάρτησης έχει ως εξής,

(setq macro (append macro(list str1)))

και ο ρόλος της είναι να παίρνει μια λίστα και να προσθέτει το περιεχόμενό της στο τέλος μια άλλης λίστας. Στο συγκεκριμένο παράδειγμα ότι περιέχει η λίστα str1 προστίθεται στο τέλος της λίστας macro με τη χρήση της append. Έτσι η νέα macro που διαμορφώνεται αντικαθιστά την παλιά με την “setq macro”.

- **Η συνάρτηση Mapcar**

Η Mapcar εφαρμόζει περισσότερες από μια λίστες σε μία συνάρτηση όπως,

```
(mapcar 'setvar
'("metavlhth1" "metavlhth2" "metavlhth3" "metavlhth4")
'(0 1 2 3)
)
```

Σχήμα 2.3 : Παράδειγμα mapcar

Η mapcar παίρνει τα στοιχεία της λίστας (0 1 2 3) και τα αναθέτει μέσω της setvar στις matavlhth1, metavlhth2, metavlhth3 και metavlhth4 αντίστοιχα.

- **Η συνάρτηση Apply**

Η apply παρέχει ολόκληρο το περιεχόμενο μια λίστας σαν μια παράμετρο. Αυτό είναι αρκετά χρήσιμο όταν θέλουμε για παράδειγμα κάποια στιγμή να εκτυπώσουμε το περιεχόμενό της, (princ (apply ‘+ list)) όπου εκτυπώνει το περιεχόμενο της list που συμπληρώνεται δυναμικά μέσα στο πρόγραμμα.

3. ΕΙΣΑΓΩΓΗ ΔΕΔΟΜΕΝΩΝ

Στο παρόν κεφάλαιο θα γίνει η ανάλυση των διαδικασιών καθώς και των βασικών συναρτήσεων, μέσω των οποίων γίνεται η συλλογή των πληροφοριών-δεδομένων μέσω της επικοινωνίας με τον ίδιο το χρήστη. Η επαφή αυτή δίνει στο χρήστη τη δυνατότητα να εκτελεί σύνθετα προγράμματα με γρήγορο ρυθμό απλά δίνοντας τα δεδομένα που τον ικανοποιούν σε κάθε περίπτωση.

3.1 Συναρτήσεις για εισαγωγή δεδομένων

Οι συναρτήσεις που ξεκινάνε με το πρόθεμα Get χρησιμοποιούνται στο πρόγραμμα για την εισαγωγή δεδομένων.

- Με τη **getpoint** επιτρέπεται η εισαγωγή συντεταγμένων σημείων από το χρήστη μέσω του πληκτρολογίου ή με το σταυρόνημα/ποντίκι.
-

```
(setq syntetagmenes_shmeiou (getpoint))
```

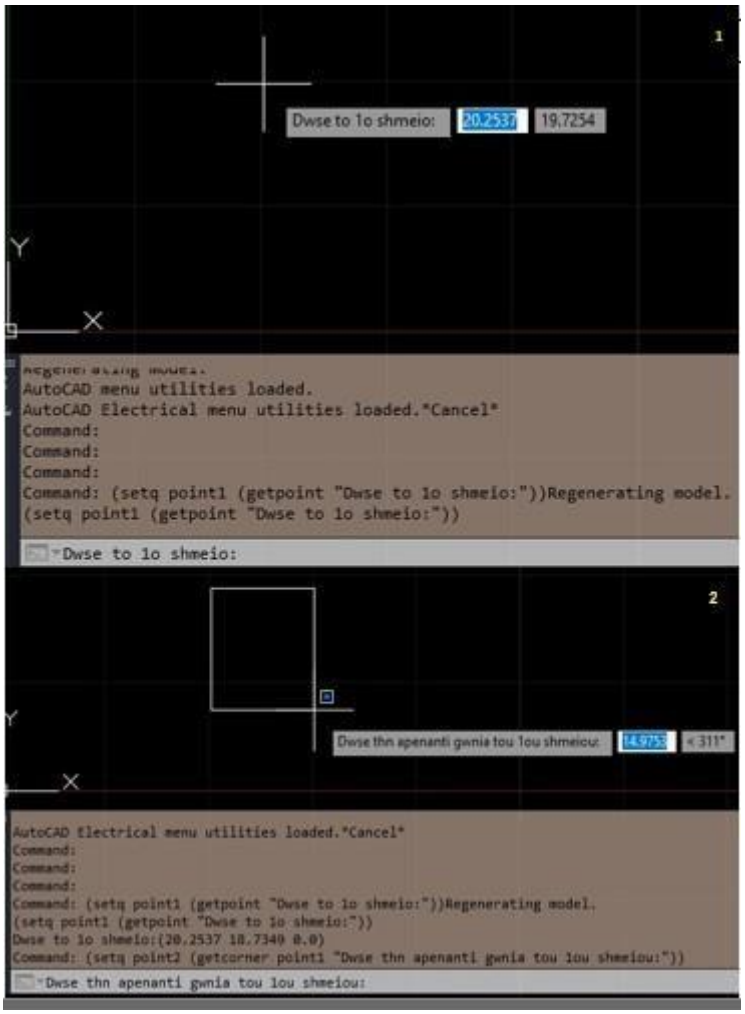
Η παραπάνω έκφραση αναθέτει με τη χρήση της setq στη μεταβλητή “syntetagmenes_shmeiou” την τιμή της getpoint, δηλαδή μια λίστα που θα δοθεί από το χρήστη, είτε πληκτρολογώντας την είτε επιλέγοντας το σημείο πάνω στο χώρο της σχεδίασης, και θα αντιπροσωπεύει ένα σύστημα συντεταγμένων. Επιπλέον κάθε φορά που ζητείται από το χρήστη να δώσει ένα σημείο, μπορεί να του ζητείται πρώτα μέσω κάποιου μηνύματος όπως,

```
(setq syntetagmenes (getpoint “Pick the first point:”))
```

- Εφόσον έχει γίνει η επιλογή των συντεταγμένων με τη getpoint εμφανίζεται ένα παράθυρο στο περιβάλλον σχεδίασης το οποίο αυξομειώνεται όσο κινείται το σταυρόνημα και η τελική του διάσταση θα οριστικοποιηθεί όταν επιλεγεί και ένα δεύτερο σημείο με τη συνάρτηση **getcorner** ,

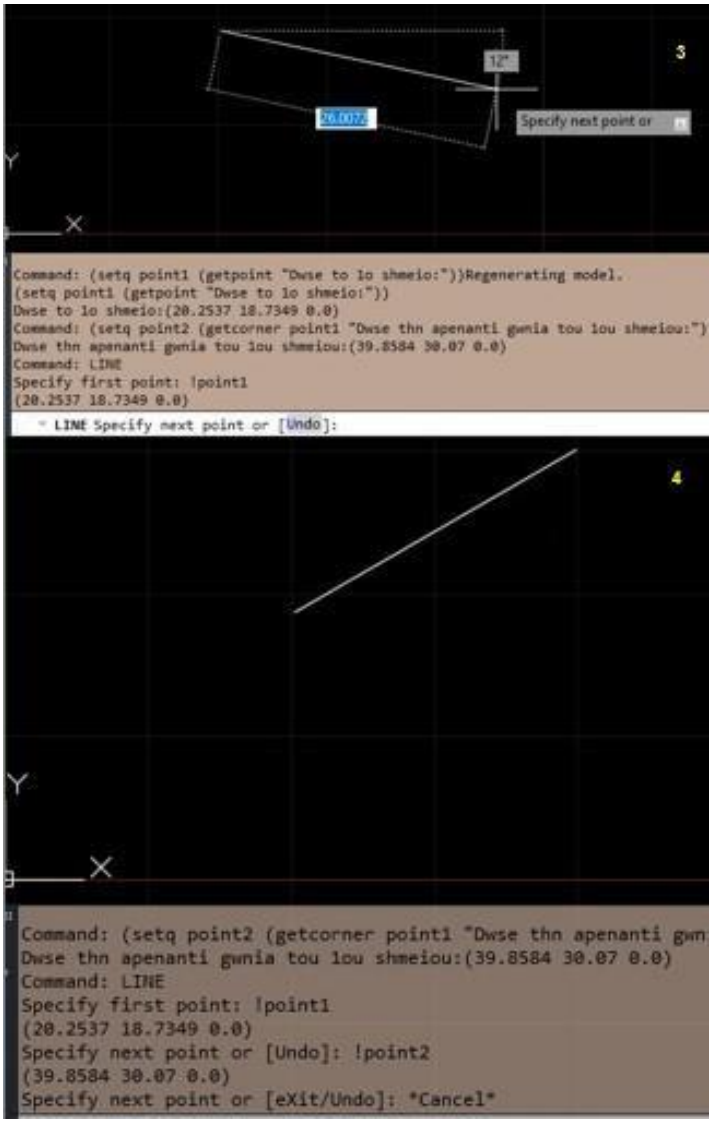
```
(setq antidiagwnies_syntetagmenes (getcorner  syntetagmenes))
```

Το σημείο που επιλέγει η getcorner θα είναι αντιδιαγώνιο αυτού που επιλέχθηκε από τη getpoint, με σκοπό τη σχεδίαση ενός παραλληλόγραμμου-τετράγωνου.
Έστω το παρακάτω παράδειγμα,



Όπως φαίνεται στο σχήμα 3.1 στην εικόνα 1, έχει ήδη τρέξει στο command η εντολή, (setq point1 (getpoint “Dwse to 1o shmeio:”)) και έτσι ζητείται από το χρήστη να δώσει το 1ο σημείο με την εμφάνιση του μηνύματος, «Dwse to 1o shmeio». Το σταυρόνημα έχει ενεργοποιηθεί και περιμένει και έστω ότι στη συνέχεια ο χρήστης διαλέγει το σημείο με συντεταγμένες (20.2537 18.7348 0.0) όπως φαίνεται και στο command window της εικόνας 2. Στη συνέχεια έχει τρέξει και η εντολή (setq point2 (getcorner point1 “Dwse thn apenanti gwnia tou 1ou shmeiou:”)) και τώρα το πρόγραμμα περιμένει ο χρήστης να του δώσει τη γωνία που βρίσκεται απέναντι από το point1 που επέλεξε νωρίτερα με την εμφάνιση του μηνύματος “Dwse thn apenanti gwnia tou 1ou shmeiou”.

Σχήμα 3.1: Συναρτήσεις getpoint, getcorner στη σχεδίαση



Σε αυτό το στάδιο το πρόγραμμα έχει αποθηκεύσει τις συντεταγμένες των point1 και point2 και τώρα θέλουμε να σχεδιαστεί η διαγώνιος του σχήματος 4 η οποία θα δουλέψει σαν οδηγός για να σχεδιαστεί το παραλληλόγραμμο που θέλουμε. Για να γίνει αυτό, καλείται η εντολή line και στο μήνυμα που θα δώσει “Specify first point:” απαντάμε !point1 ώστε να ξεκινήσει η γραμμή από το point1 που επιλέχθηκε νωρίτερα. Έπειτα στο μήνυμα “Specify next point:” απαντάμε !point2 και έτσι αυτόματα σχεδιάζεται μια διαγώνιος από το point1 έως το point2 πάνω στην οποία θα βασιστούμε για να σχεδιάσουμε το παραλληλόγραμμο.

Όλα τα βήματα που έχουν εκτελεσθεί φαίνονται με τη σειρά στο command window. Ωστόσο αυτό είναι ένα πολύ συγκεκριμένο παράδειγμα για την κατανόηση των getpoint και getcorner το οποίο μπορεί να γενικευθεί για να εξυπηρετεί κάθε φορά το χρήστη.

Σχήμα 3.2: Ολοκλήρωση παραδείγματος με getpoint, getcorner

Σημείωση: επειδή μπορεί οι γωνίες να δίνονται σε ακτίνια είναι απαραίτητη χρήση των τύπων για μετατροπή μονάδων,

- ✓ Τύπος για μετατροπή από ακτίνια σε μοίρες, radians*57.2958
- ✓ Τύπος για μετατροπή από μοίρες σε ακτίνια, degrees*0.0174533

- Με τη χρήση της **getdist** υπολογίζεται η απόσταση μεταξύ δύο συντεταγμένων που έχουν δοθεί από το χρήστη. Για παράδειγμα αν γραφτεί στο command , (setq distance1 (getdist “Pick the point or enter distance:”) τότε θα εμφανιστεί το μήνυμα, Pick thepoint or enter distance: , και έστω ότι επιλέγεται το σημείο 3,3. Στη συνέχεια ζητείται και ένα δεύτερο σημείο, έστω ότι δίνεται το 8,3. Με χρήση της παραπάνω γραμμής κώδικα έχει τώρα αντιστοιχηθεί στη μεταβλητή distance1 η τιμή 5 που είναι η απόσταση μεταξύ των σημείων (3,3) και (8,3), distance1=(8,3)-(3,3)=(5,0)=5.0
- Για την εισαγωγή αλφαριθμητικού κειμένου, χρησιμοποιείται η συνάρτηση **Getstring** και συντάσσεται ως,

```
(setq onomatepwnumo (getstring T «Enter a name: “))
```

Το T μετά τη getstring συμβολίζει κάτι αληθές και επιτρέπει στο χρήστη να εισάγει εκφράσεις, δηλαδή κενά μεταξύ των λέξεων. Χωρίς αυτό, δεν θα ήταν εφικτή η εισαγωγή του «Όνομα επίθετο» αλλά μόνο η μορφή «Όνομα_επίθετο» καθώς το κενό θα το λάμβανε σαν enter και θα προχωρούσε σε επόμενες εντολές στο command. Αντί για το T θα μπορούσε να μπει οποιαδήποτε έκφραση αρκεί να μην αποτιμάται ως nill.

- Για την είσοδο καθορισμένων μηνυμάτων προς απάντηση, χρησιμοποιείται η **Getword** συνδυαστικά με την **Initget** και συντάσσονται ως,

```
(initget "Yes No")
(setq apanthsh (getword "Eiste sigouroi?" <Yes/No>:")
)
```

Σχήμα 3.3: Παράδειγμα Initget, Getword

Στη συνέχεια εμφανίζεται το μήνυμα ερώτηση Eiste sigouroi?<Yes/No>:
Ο χρήστης πληκτρολογώντας ένα Yes ή No ή ακόμα και τα αρχικά τους μόνο Y,N εφόσον είναι κεφαλαία, έχει δώσει στη μεταβλητή «apanthsh» μια από τις απαντήσεις που έχει προκαθορίσει η initget. Οποιαδήποτε άλλη απάντηση δεν γίνεται δεκτή και εμφανίζεται μήνυμα λάθους.

- Για την είσοδο πραγματικών τιμών και ακεραίων χρησιμοποιούνται οι **Getreal** και **Getint** αντίστοιχα που συντάσσονται ως εξής,
-

```
(getint "Enter a number :")
(getreal "Enter a number :")
```

Αν στη θέση του ακέραιου δοθεί δεκαδικός αριθμός τότε θα ζητήσει ξανά έναν αριθμό ενώ αν δοθεί ακέραιος στην getreal τότε αυτή θα το μετατρέψει σε πραγματικό, δηλαδή αν δοθεί ο αριθμός 5 θα επιστρέψει τον πραγματικό 5.0.

Συνάρτηση	Χρήση	Σύνταξη
Getpoint	Εισαγωγή συντεταγμένων σημείων από το χρήστη μέσω του πληκτρολογίου ή με το σταυρόνημα/ποντίκι.	(setq syntetagmenes (getpoint “Pick the first point:”))
Getcorner	Το σημείο που επιλέγει η getcorner θα είναι αντιδιαγώνιο αυτού που επιλέχθηκε από τη getpoint	(setq antidiagwnies_syntetagmenes (getcorner syntetagmenes))
Getdist	Υπολογίζει την απόσταση μεταξύ δύο συντεταγμένων που έχουν δοθεί από το χρήστη	(setq distance1 (getdist “Pick the point or enter distance:”)
Getorient	Επιστρέφει τη γωνία βάσει της πρότυπης ρύθμισης, με το μηδέν να είναι στον οριζόντιο άξονα	setq gwnia1 (getorient “Dwse th gwnia apo to plhktrologio:”))

Getangle	Επιστρέφει τη γωνία με βάση την τρέχουσα ρύθμιση που ορίζει ο χρήστης με την εντολή Units	(setq gwnia1 (getangle “Dwse th gwnia apo to plhktrologio:”))
Getstring	Για την εισαγωγή αλφαριθμητικού κειμένου	(setq onomatepwnumo (getstring T «Enter a name: “))
Getword	Για την είσοδο καθορισμένων μηνυμάτων προς απάντηση, χρησιμοποιείται η Getword συνδυαστικά με την Initget	(initget “Yes No”) (setq apanthsh (getword “Eiste sigouroi?” <Yes/No>:”))
Getint	Για την είσοδο ακέραιων τιμών	(getint “Enter a number :”)
Getreal	Για την είσοδο πραγματικών τιμών	(getreal “Enter a number :”)

Πίνακας 3.1: Ανακεφαλαίωση **συναρτήσεων Get** για εισαγωγή δεδομένων

- Για να ανοίξει και να διαβαστεί ένα αρχείο κατά τη διάρκεια της εκτέλεσης ενός κώδικα χρησιμοποιείται η συνάρτηση **open**,

(open όνομα_αρχείου κωδικός_εγγραφής/ανάγνωσης)

Η πρώτη παράμετρος έχει το όνομα του αρχείου που θέλουμε να ανοίξει και να διαβαστεί και η δεύτερη παράμετρος είναι ένας από τους κωδικούς του πίνακα 3.2 που θα δείξει στην AutoLISP τι να κάνει στο αρχείο.
Προσοχή! Οι κωδικοί είναι σημαντικό να είναι με πεζά γράμματα διαφορετικά η συνάρτηση open δεν θα λειτουργήσει.
Είναι σημαντικό μετά την ανάγνωση του αρχείου ο χρήστης να το κλείσει,**(close αρχείο)** διαφορετικά το περιεχόμενό του κινδυνεύει να χαθεί.

Κωδικός εγγραφής/ανάγνωσης	Χρήση
“r”	Ανοίγει ένα αρχείο μόνο για να το διαβάσει (read). Αν το αρχείο δεν υπάρχει θα εμφανιστεί μήνυμα λάθους.
“w”	Ανοίγει ένα αρχείο για εγγραφή (writing). Αν το αρχείο υπάρχει ήδη το περιεχόμενό του θα αντικατασταθεί με το νέο που θα δώσει ο χρήστης. Αν ωστόσο το αρχείο δεν υπάρχει τότε θα δημιουργηθεί εκείνη τη στιγμή
“a”	Ανοίγει ένα αρχείο και προσθέτει κάτι στο τέλος του (add). Αν το αρχείο δεν υπάρχει θα δημιουργηθεί.

Πίνακας 3.2: Κωδικοί της συνάρτησης open

Ο πίνακας 3.3 παραθέτει μερικές από τις συναρτήσεις ανάγνωσης και εγγραφής αρχείων με μία σύντομη περιγραφή για την κάθε μια.

Συνάρτηση	Περιγραφή
(Prin1 σύμβολο/έκφραση)	Εκτυπώνει στη γραμμή εντολής οποιαδήποτε έκφραση.
(Princ σύμβολο/έκφραση)	Εκτυπώνει το ίδιο με την Prin1 αλλά επεξεργάζεται και τους χαρακτήρες ελέγχου.
(Print σύμβολο/έκφραση)	Εκτυπώνει το ίδιο με την Prin1 αλλά εκτυπώνεται μία νέα γραμμή πριν την έκφρασή της και ένα κενό μετά.
(Read-char ενδείκτης_αρχείου)	Διαβάζει από το πληκτρολόγιο έναν χαρακτήρα .
(Read-line ενδείκτης_αρχείου)	Διαβάζει ένα string από το πληκτρολόγιο ή μια γραμμή κειμένου από ένα ανοικτό αρχείο.
(Write-char ακέραιος ενδείκτης_αρχείου)	Γράφει ένα χαρακτήρα στη γραμμή εντολής της οθόνης.
(Write_line string ενδείκτης_αρχείου)	Γράφει ένα string στη γραμμή εντολής της οθόνης.

Πίνακας 3.3: Συναρτήσει ανάγνωσης και εγγραφής αρχείου

Οι συναρτήσεις Prin1, Princ και Print είναι σχεδόν όμοιες και χρησιμοποιούν την κοινή σύνταξη, (princ string/μεταβλητή_string προαιρετικός_ενδείκτης_αρχείου)

4. ΣΧΕΔΙΑΣΜΟΣ ΠΡΟΓΡΑΜΜΑΤΟΣ

Για το σχεδιασμό ενός προγράμματος είναι σημαντικό να ακολουθούνται κάποια βασικά βήματα προς αποφυγή γνωστών λαθών ή παράλειψη πληροφοριών, σύμφωνα με τα οποία,

1. Στην αρχή του προγράμματος θα πρέπει να ορίζεται το όνομά του ή το όνομα της συνάρτησης που θα χρησιμοποιηθεί.
2. Οι πληροφορίες που χρειάζονται, να συλλέγονται από το χρήστη σταδιακά ή να δίνονται αμέσως κατά τη σύνταξη του προγράμματος.
3. Η πληροφορία που δίνεται να επεξεργάζεται.
4. Οι υπολογισμοί να εκτελούνται με τη σωστή σειρά παράγοντας σωστά αποτελέσματα.

Μία σειρά από γνωστά λάθη που πρέπει να λαμβάνονται υπόψιν προς αποφυγή είναι, τα λάθη κατά τη σύνταξη και κατ' επέκταση τα ορθογραφικά, τα κενά διαστήματα και οι παρενθέσεις, τα πεζά γράμματα να συγχέονται με τα κεφαλαία καθώς και η επανάληψη στα ονόματα των συναρτήσεων. Είναι σημαντικό να δίνεται η απαραίτητη προσοχή κατά τη σύνταξη ενός προγράμματος για να αποφευχθούν στοιχειώδη λάθη για τα οποία θα ξοδευθεί επιπλέον χρόνος για την αναγνώρισή τους.

4.1 Εντολές του AutoCAD στη σύνταξη της AutoLISP

Εκτός από τις συναρτήσεις που χρησιμοποιούνται σε προγράμματα στην AutoLISP,συνηθίζεται και η χρήση εντολών που υπάρχουν ήδη μέσα στο AutoCAD για την επίτευξη μίας σχεδίασης, όπως για παράδειγμα οι εντολές **line** και **circle**,

```
(command "line" point1 point2 point3 point4 "c")
```

Η παραπάνω εντολή σχεδιάζει μία γραμμή με τη χρήση της εντολής line (οτιδήποτε ακολουθείται της συνάρτησης command και βρίσκεται σε εισαγωγικά θεωρείται είσοδος από το πληκτρολόγιο) ακολουθώντας τα σημεία point1,point2,point3 και point4. Το "c" εκτελεί την εντολή close για να ολοκληρώσει την εντολή line

4.1.1 Συνάρτηση που ορίζεται από το χρήστη

Όταν μια συνάρτηση ορίζεται από το χρήστη συντάσσεται *χωρίς το c,*δηλαδή δεν γράφεται defun c ATHROISMA….,

```
(
defun ATHROISMA_TETRAGWNWN (x y)
(+ (* x x) (* y y))
)
```

Η παραπάνω συνάρτηση υπολογίζει το άθροισμα των τετραγώνων για οποιοδήποτε ζεύγος τιμών x,y δοθεί από το χρήστη. Γράφοντας στο command (athroisma_tetragwnwn 2 3) εκτελεί τα βήματα, **(+ (* 2 2) (* 3 3)) = (+ (4) (9)) = 13** και εκτυπώνει το 13. Προσοχή, το όνομα της συνάρτησης που καλείται να είναι ακριβώς ίδιο με αυτό της defun, (χωρίς απαραίτητα να είναι κεφαλαία) δηλαδή αν γραφτεί (athroisma tetragwnwn 2 3) δεν εντοπίζει τη συνάρτηση, αφού λείπει η κάτω παύλα και εκτυπώνει μήνυμα λάθους.

4.1.2 Η δεξιά κάθετος / και οι χαρακτήρες ελέγχου

Η δεξιά κάθετος / σε μία συνάρτηση διαχωρίζει τις μεταβλητές που θα πάρουν κάποια τιμή με την εκτέλεση αυτής της συνάρτησης, με αυτές τις μεταβλητές που έχουν ήδη κάποια τιμή και θα χρησιμοποιηθούν έτοιμες. Για παράδειγμα στη συνάρτηση,

```
(
 defun tetragwno (x y / dx dy)
(setq dx (* x x))
(setq dy (* y y))
(P dx dy)
)
```

τα x,y καλούνται δεδομένου ότι έχουν ήδη κάποιες τιμές ενώ στα dx,dy η ανάθεση τιμών τους γίνεται μέσα στην ίδια τη συνάρτηση " tetragwno ". Προσοχή πριν και μετά την κάθετο να υπάρχει από ένα κενό αλλιώς θα εμφανιστεί μήνυμα λάθους.
Ο πίνακας 4.1 περιέχει μια λίστα χαρακτήρων ελέγχου που ο καθένας εκτελεί μια διαφορετική ενέργεια. Για παράδειγμα το \n στην αρχή ενός αλφαριθμητικού μηνύματος δίνει εντολή στην AutoLISP να ξεκινήσει μια νέα γραμμή.

Χαρακτήρας	Χρήση
\e	Escape
\n	New line (νέα γραμμή)
\r	Enter (Return)
\t	Tab
\007	Beep (καμπανάκι)
\nnn	Αφορά χαρακτήρες που έχουν εισαχθεί με οκταδική τους μορφή

Πίνακας 4.1: Λίστα χαρακτήρων ελέγχου

4.1.3 Η συνάρτηση Clean

Αν η συνάρτηση Clean προστεθεί στην αρχή ενός αρχείου LISP θα γίνει για δύο σκοπούς, είτε για κλείσει τυχόν αρχεία που έχουν μείνει ανοικτά, είτε για να καθαρίσει την Atomlist από οποιαδήποτε συνάρτηση που προστέθηκε μετά από αυτήν, ανακτώντας έτσι τη μνήμη. Η Atomlist είναι μια ειδική λίστα που αποθηκεύει δεδομένα, διάφορες συναρτήσεις και σύμβολα του χρήστη και ανανεώνεται κάθε φορά που προστίθεται ένα νέο σύμβολο ή μια συνάρτηση. Η σύνταξή της Clean φαίνεται στο σχήμα 4.1

CLEAN.lsp

```
; Programma diagrafhs anoiktwn arxeiwn kai katharisma ths Atomlist

(defun C:CLEAN (/ i item)
  (setq i 0)   ; to i einai metrhths
  (while (not (equal (setq item (nth i atomlist) nil))
                     (if (= (type (eval item)) 'FILE)
                            (close (eval item));an to item einai arxeio to kleinei
                            );kleinei h if
                       (setq i (1+i) )
                       )) ;kleinei h while
  (setq atomlist (member 'C:CLEAN atomlist)) ;orizei ek neou thn atomlist afou thn
 'DONE                                       ;exei katharisei
  ) ;telos programmatos
```

Σχήμα 4.1: Το πρόγραμμα Clean

Στο σχήμα 4.1 φαίνεται ακόμα μια συνάρτηση η **Type** η οποία επιστρέφει τον τύπο ενός δεδομένου, εδώ του item, άρα η συνθήκη ελέγχου κοιτάζει αν το item είναι FILE (δείκτης αρχείου) (if (= type (eval item)) 'FILE). Αν δεν χρησιμοποιούνταν συνδυαστικά με την Eval τότε θα επέστρεφε τον τύπο του δεδομένου Item και όχι το τύπο δεδομένου της τιμής του item. Στον πίνακα 4.2 φαίνονται κάποιες τιμές που επιστρέφει η Type αλλά και πως αυτές ερμηνεύονται,

Τιμές της Type	Ερμηνεία
REAL	Πραγματικός αριθμός
FILE	Ενδείκτης αρχείου
STR	String(αλφαριθμητικό)
INT	Integer (ακέραιος)
SYM	Σύμβολο
LIST	Λίστα ή συνάρτηση ορισμένη από το χρήστη
SUBR	Συνάρτηση της AutoLISP
PICKSET	Επιλεγμένη ομάδα
ENAME	Όνομα οντότητας (αντικειμένου σχεδίασης)
PAGETB	Πίνακας συνάρτησης σελιδοποίησης

Πίνακας 4.2: Λίστα τιμών που επιστρέφει η Type

4.1.4 Το πρόγραμμα BOX

Για την καλύτερη κατανόηση των παραπάνω βημάτων, θα γίνει η απεικόνισή τους με την εφαρμογή του προγράμματος σχεδίασης ενός ορθογώνιου παραλληλόγραμμου BOX.

1) Ο ορισμός του προγράμματος γίνεται στην πρώτη γραμμή με την εντολή,

```
(defun c:BOX (/ dx dy point1 point2 point3 point4)
```

στην οποία δίνεται το όνομα BOX και ορίζονται οι μεταβλητές που θα χρησιμοποιηθούν που είναι οι dx,dy,point1,point2,point3,point4. Το σύμβολο c εδώ, κάνει όλο το πρόγραμμα που θα ακολουθήσει μια εντολή, δηλαδή καλώντας στο command το όνομα του προγράμματος, το BOX, το πρόγραμμα θα εκτελέσει αμέσως όλα αυτά που θα περιέχονται μέσα σε αυτό, ενώ με την αφαίρεση του c από τη συνάρτηση επιτρέπεται το BOX να κληθεί και από άλλες συναρτήσεις.

Το όνομα BOX πρέπει να είναι με κεφαλαία και να ΜΗΝ υπάρχει ξανά στο πρόγραμμά μας, για παράδειγμα αν ονομαστεί line που υπάρχει ήδη ως εντολή του AUTOCAD, τότε η υπάρχουσα εντολή θα αντικατασταθεί με αυτό που θα συντάξει ο χρήστης.

2) Η συλλογή των πληροφοριών από το χρήστη γίνεται με τη χρήση της συνάρτησης **setq** με τις εντολές,

```
(setq point1 (getpoint "Dwse thn prwth gwnia"))
(setq point2 (getcorner "Dwse thn antidiagwnio tou shmeiou point1"))
```

Για να σχεδιαστεί το παραλληλόγραμμο χρειάζονται ακόμα δύο σημεία τα οποία προκύπτουν με την,

3) Επεξεργασία των δεδομένων που έχουν ήδη δοθεί μέσω των εντολών που αφορούν μια λίστα

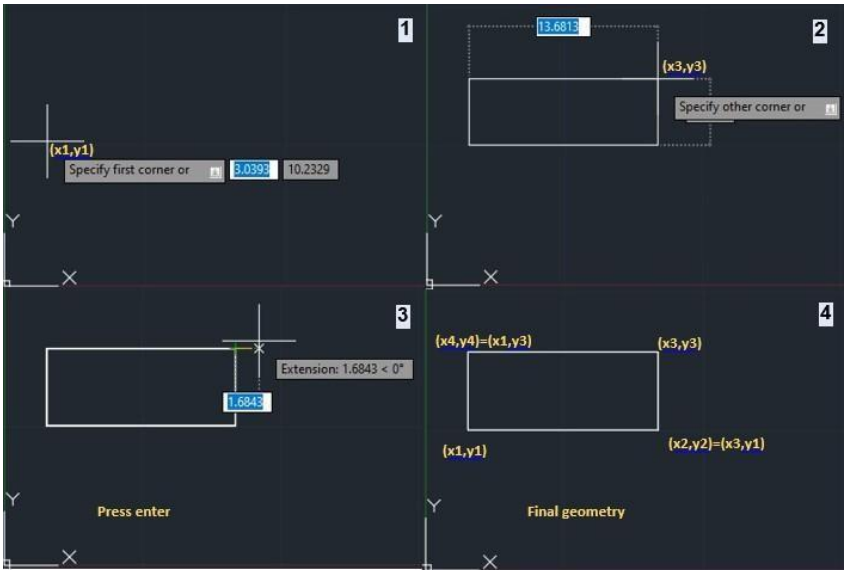
```
(
 setq point2 (list (car point3) (cadr point1)))
(setq point4 (list (car point1) (cadr point3))
)
```

4) Η παραγωγή των αποτελεσμάτων, δηλαδή η σχεδίαση του παραλληλόγραμμου γίνεται με την τελευταία εντολή με την εντολή line μεταξύ των σημείων/μεταβλητών που ορίστηκαν παραπάνω,

```
(command "line" point1 point2 point3 point4 "c")
```

Στο σχήμα 4.2 φαίνονται όλα τα παραπάνω βήματα σε απεικόνιση, με τη σειρά που εκτελούνται, όπου (x1,y1) οι συντεταγμένες του point1, (x2,y2) οι συντεταγμένες του point2 και ούτω καθεξής.

.



Σχήμα 4.2: Τα βήματα για την ανάπτυξη του προγράμματος BOX

4.2 Λήψη αποφάσεων

Μια έκφραση ελέγχου μπορεί να χρησιμοποιήσει οποιαδήποτε συνάρτηση, αλλά τις περισσότερες φορές χρησιμοποιούνται κατηγορηματικοί ή λογικοί τελεστές που είναι επίσης συναρτήσεις και επιστρέφουν αληθές (T) ή ψευδές (nill) σαν μήνυμα. Όλοι οι κατηγορηματικοί και λογικοί τελεστές εμφανίζονται ως πρώτο στοιχείο σε μία έκφραση όπως, (> 2 4) που θα επιστρέψει nill καθώς δεν ισχύει η συνθήκη 2>4.

4.2.1 Κατηγορηματικοί τελεστές

Για τον παρακάτω πίνακα με τους κατηγορηματικούς τελεστές το p στο τέλος του ονόματος κάποιων τελεστών προκύπτει από το predicate (κατηγορηματικός) και το «αντικείμενο» σημαίνει οποιαδήποτε λίστα ή άτομο που περιέχουν αριθμούς και σύμβολα.

Κατηγορηματικοί τελεστές	Ελέγχουν αν,
<	αριθμητική τιμή μικρότερη κάποιας άλλης
>	αριθμητική τιμή μεγαλύτερη κάποιας άλλης
<=	αριθμητική τιμή μικρότερη ή ίση κάποιας άλλης
>=	αριθμητική τιμή μεγαλύτερη ή ίση κάποιας άλλης
=	δύο αριθμητικές τιμές ή δύο αλφαριθμητικά ίσα
/=	δύο αριθμητικές τιμές ή δύο αλφαριθμητικά άνισα

eq	δύο τιμές είναι *ακριβώς ίδιες*
equal	δύο τιμές είναι ίδιες
atom	ένα αντικείμενο δεν είναι λίστα αλλά άτομο
boundp	ένα σύμβολο έχει κάποια τιμή
listp	ένα αντικείμενο είναι λίστα
minusp	αρνητική αριθμητική τιμή
numberp	ένα αντικείμενο είναι πραγματικός ή ακέραιος αριθμός
zerop	ένα αντικείμενο είναι μηδέν

Πίνακας 4.3: Κατηγορηματικοί τελεστές και τι ελέγχουν

4.2.2 Λογικοί τελεστές

Οι συναρτήσεις των λογικών τελεστών χρησιμοποιούνται περισσότερο για να ελέγξουν τους κατηγορηματικούς τελεστές, για παράδειγμα

```
(setq x 1.5)
(zerop x)
```

Επιστρέφει nill καθώς στο x έχει τεθεί η τιμή 1,5 και δεν ισούται με μηδέν που λέει η εντολή zerop x.

Λογικοί τελεστές	
and	όλα τα άτομα ή οι εκφράσεις επιστρέφουν μη-nill
not	ένα σύμβολο είναι nill
nul	μία λίστα είναι nill
or	μία ή μερικές εκφράσεις ή μερικά άτομα επιστρέφουν μη-nill

Πίνακας 4.4: Λογικοί τελεστές

4.2.3 Η συνάρτηση if

Η συνάρτηση που χρησιμοποιείται για τον έλεγχο των συνθηκών είναι η **if** η οποία προτού εκτελέσει κάποια ενέργεια ή κάνει κάποια επιλογή. Ελέγχει εάν εκπληρώνονται κάποιες καθορισμένες συνθήκες και η γενικευμένη της σύνταξη είναι,

```
(
if (h sunthiki pou thelei o xrhsths na plhreitai)
(ektelese mia sugkekrimenh energeia)
)
```

Ένα πιο συγκεκριμένο παράδειγμα για την καλύτερη κατανόηση της συνάρτησης είναι πάλι από το πρόγραμμα BOX,

```
(if (not C:BOX) (load "box") (princ "to Box exei hdh fortothei."))
```

→ Εάν η έκφραση (not C:BOX) είναι αληθής τότε εκτελείται η πρώτη παρένθεση (load “box”) δηλαδή φορτώνει το πρόγραμμα BOX.

→Εάν η έκφραση (not C:BOX) είναι ψευδής τότε εκτελείται η δεύτερη παρένθεση (princ “to Box exei hdh fortothei.”) που εκτυπώνει το μήνυμα “to Box exei hdh fortothei.”

4.2.4 Η συνάρτηση progn

Σε συνέχεια της συνάρτησης if έρχεται η συνάρτηση progn με τη χρήση της οποίας κάνουμε πολλές εκφράσεις να αποτιμώνται σαν να ήταν μία. Συνδυαστικά με τη συνάρτηση if ανάλογα ποια συνθήκη επιλέγεται εκτελείται και το αντίστοιχο πρόγραμμα όπως φαίνεται στο σχήμα 4.3.

MAINBOX.lsp

```
(defun C:MAINBOX (/ point1 point2 point3 point4  choose)
  (setq choose (getstring "\n Thelete sxediash tou box? <Y=yes/Rturn=no>"))
  (if (or (equal choose "y") (equal choose "Y") (equal choose "Yes") (equal choose "yes") (equal choose "YES"))
    (progn   ;An apanthsh einai yes tote mpainei edw kai sxediazei to box
      (setq point1 (getpoint "Dwse thn prwth gwnia: "))
      (setq point3 (getcorner point1 "Dwse thn antidiagwnio tou shmeiou point1: "))
      (setq point2 (list (car point3) (cadr point1))) ; point2 = (x3,y1)
      (setq point4 (list (car point1) (cadr point3))) ; point4= (x1,y3)
      (command "line" point1 point2 point3 point4 "c")
      ); kleinei h progn

    (progn ;An h apanthsh den einai Yes
      (princ "Telos programmatos!")
      );kleinei h progn
    );kleinei h if
  ); kleinei h defun
```

Σχήμα 4.3: Παράδειγμα συνάρτησης progn

4.2.5 Η συνάρτηση Cond

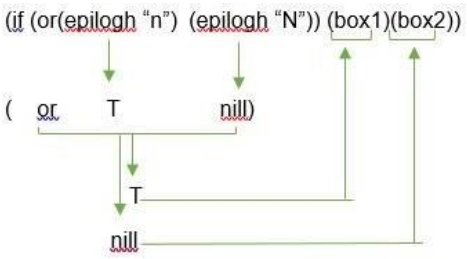
Η συνάρτηση Cond μπορεί να χρησιμοποιηθεί οπουδήποτε αντικαθιστώντας τη συνάρτηση If όπως στο παραπάνω παράδειγμα το οποίο θα γινόταν πιο απλοϊκό με τη συνάρτηση Cond όπως φαίνεται στο σχήμα 4.4.

```
(defun C:COND (/ point1 point2 point3 point4  choose)
  (setq choose (getstring "\n Thelete sxediash tou box? <Y=yes/Rturn=no>"))
  (cond
    ( (or (equal choose "y") (equal choose "Y") (equal choose "Yes") (equal choose "yes") (equal choose "YES"))
      (setq point1 (getpoint "Dwse thn prwth gwnia: "))
      (setq point3 (getcorner point1 "Dwse thn antidiagwnio tou shmeiou point1: "))
      (setq point2 (list (car point3) (cadr point1))) ; point2 = (x3,y1)
      (setq point4 (list (car point1) (cadr point3))) ; point4= (x1,y3)
      (command "line" point1 point2 point3 point4 "c")
      )
    ( (or ;se opoiadhpote allh apanthsh tipose thn parakatw ekfrash
          (princ "Telos programmatos!"))
      )
    )
  )
```

Σχήμα 4.4: Συνάρτηση Cond

Η Cond εκτελεί ελέγχους μέχρι να φθάσει εκείνον που επιστρέφει T και επιπλέον με τη χρήση της δεν χρειάζεται ούτε η συνάρτηση progn.

4.2.6 Ο λογικός τελεστής Or



Σχήμα 4.5: Παράδειγμα του τελεστή Or

Στην or η μια παράμετρος ελέγχει αν η μεταβλητή epilogh ισούται με το πεζό γράμμα n και η άλλη ελέγχει αν ισούται με το κεφαλαίο N. Όταν ο χρήστης απαντήσει με μια από τις δύο επιλογές, δηλαδή n ή N τότε η or θα επιστρέψει αληθές μήνυμα T καθώς αναγνωρίζει αυτά τα δύο ορίσματα. Σε οποιαδήποτε άλλη απάντηση από τον χρήστη θα επιστραφεί μήνυμα λάθους nill. Όταν η or επιστρέφει T (δηλαδή η epilogh ισούται με "n"), η if αποτιμά τη δεύτερη παράμετρο box1, ενώ όταν επιστρέφει nill αποτιμά την παράμετρο box2.

4.2.7 Η συνάρτηση While

Η σύνταξη της While σε γενικευμένη μορφή έχει ως εξής,

```
(
 while (ekfrash_elegxou)
(ekfrash1)(ekfrash2)……
)
```

Η "ekfrash_elegxou" είναι μια έκφραση που αποτελεί μια συνθήκη. Όσο αυτή η συνθήκη τηρείται, δηλαδή επιστρέφεται αποτέλεσμα μη-nill, τόσο εκτελούνται όλες οι εκφράσεις που υπάρχουν εσωτερικά της while όπως οι, ekfrash1, ekfrash2 κλπ.
Παράδειγμα από κομμάτι κώδικα,

```
(while (/=str1 "/")
(setq str1 (getstring "/n Dwse to macro or / gia eksodo: "))
.
.
);end while
```

Όσο λοιπόν η συνθήκη της while είναι αληθής, δηλαδή το str1 δεν ορίζεται ως / τότε να εκτελέσει την setq εντολή της επόμενης γραμμής. Στη συνέχεια εκτελεί ότι άλλο μπορεί να υπάρχει εσωτερικά της μέχρι να φτάσει στο τερματικό end while.

Σημείωση: το κομμάτι ;end while αποτελεί σχόλιο για τον κώδικα καθώς προηγείται το ελληνικό ερωτηματικό και χρησιμοποιείται για διευκόλυνση θυμίζοντάς μας ότι αυτή η παρένθεση κλείνει την while.

4.2.8 Η συνάρτηση Repeat

Η Repeat όπως και η While αποτελούν συναρτήσεις επανάληψης. Η βασική τους διαφορά είναι πως η Repeat δεν εκτελείται όσο κάποια συνθήκη είναι αληθής αλλά εκτελείται για n φορές που θα έχει ορίσει ο χρήστης.

```
(
 repeat n
(ekfrash1)(ekfrash2).......(ekfrashn)
)
```

Το n μπορεί να είναι ένα ακέραιος αριθμός είτε κάποιο σύμβολο που αντιπροσωπεύει αυτόν.

5. Γεωμετρία στην AutoLISP

Σε αυτό το κεφάλαιο θα αναφερθούν τρόποι υπολογισμού γωνιών και αποστάσεων τόσο για σημεία που είναι προϋπάρχοντα στο χώρο όσο και για σημεία που θα εξυπηρετούν το χρήστη γενικότερα. Ακόμα θα αναλυθεί ένα πολύ σημαντικό κεφάλαιο, αυτό των Osnaps που με την κατανόηση και τη χρήση τους δημιουργείται ένα ευκολότερο και ακριβέστερο αποτέλεσμα σχεδίασης ενώ τέλος ακολουθούν μερικές από τις βασικές τριγωνομετρικές συναρτήσεις.

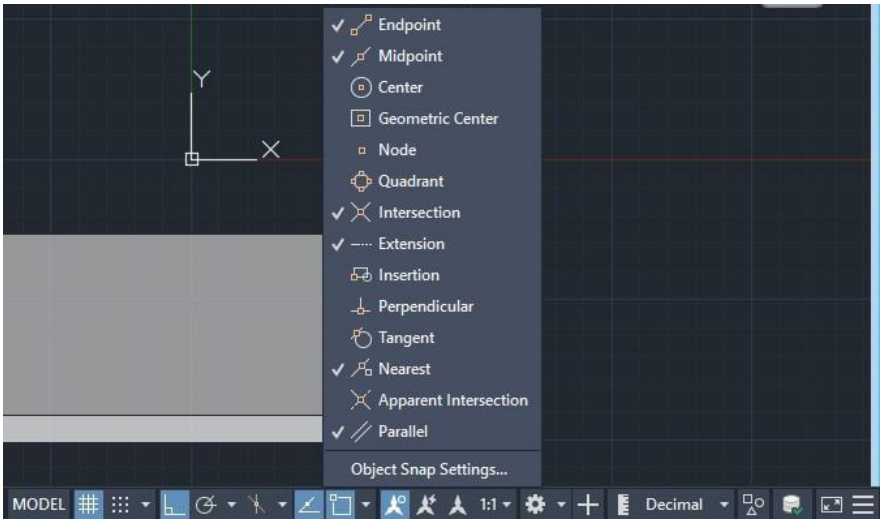
5.1 Osnaps, γωνίες και αποστάσεις

Στο κεφάλαιο 2 που αφορούσε την εισαγωγή δεδομένων είχαν αναλυθεί συναρτήσεις που υπολογίζουν γωνίες και αποστάσεις οι getangle και getdist αντίστοιχα, όμως τα σημεία για τα οποία γινόντουσαν αυτοί οι υπολογισμοί δίνονταν από το χρήστη. Τι συμβαίνει όμως όταν θέλουμε να υπολογιστούν γωνίες και αποστάσεις για ήδη υπάρχοντα σημεία στο χώρο;

Πρώτα μια **Υπενθύμιση:**

Επειδή μπορεί οι γωνίες να δίνονται σε ακτίνια είναι απαραίτητη χρήση των τύπων για μετατροπή μονάδων,

- Τύπος για μετατροπή από ακτίνια σε μοίρες, *radians*57.2958*
- Τύπος για μετατροπή από μοίρες σε ακτίνια, *degrees*0.0174533*



Σχήμα 5.1: Τα Osnaps στο χώρο σχεδίασης

Αρχικά είναι απαραίτητη η αναφορά στα Osnaps που διαθέτει το AutoCAD. Τα Osnaps (object snaps) είναι απαραίτητα καθώς προσφέρουν ακριβέστερη και ευκολότερη σχεδίαση για το χρήστη. Υπάρχουν πολλά είδη snaps που μπορούν να ενεργοποιηθούν κατά τη σχεδίαση όπως για παράδειγμα το midpoint που με την ενεργοποίηση του εντοπίζει αυτόματα σε ένα αντικείμενο το μέσο του, ακουμπώντας ο χρήστης απλά τον κέρσορα κοντά στο κέντρο αυτού.

Σχήμα 5.2: Παράδειγμα midpoint

Έστω για παράδειγμα ότι έχει σχεδιαστεί ήδη μια οριζόντια γραμμή και θέλω να σχεδιάσω μία κάθετη που να περνάει από το μέσω της. Επιλέγοντας την εντολή line και απλά τοποθετώντας τον κέρσορα περίπου προς το κέντρο, με την ενεργοποίηση του snap θα εμφανιστεί που είναι η ακριβής τοποθεσία του μέσου του αντικειμένου (midpoint) χωρίς επιπλέον υπολογισμούς. Με την ίδια λογική λειτουργούν όλα τα είδη των snaps προσφέροντας το καθένα τη δικιά του διευκόλυνση όπως φαίνεται στον παρακάτω πίνακα.

Είδη των Osnaps	**Τι προκαλούν/δίνουν**
ENDppoint	το πλησιέστερο τελικό σημείο ή γωνία ενός γεωμετρικού αντικειμένου
MIDpoint	το μέσο σημείο ενός γεωμετρικού αντικειμένου
Center	το κέντρο ενός τόξου, κύκλου, έλλειψης ή ελλειπτικού τόξου
Geometric Center	το κεντροειδές σε κάθε κλειστή καμπύλη και splines
Node	το σημείο ενός αντικειμένου, ένα σημείο που καθορίζει μια διάσταση ή τη διάσταση ενός κειμένου
Quadrant	ένα τεταρτημόριο ενός τόξου, κύκλου, έλλειψης ή ελλειπτικού τόξου
Intersection	τη διατομή μεταξύ γεωμετρικών αντικειμένων
Extension	δημιουργεί μια προσωρινή/νοητή γραμμή επέκτασης ή τόξο που εμφανίζεται όταν περνάει ο κέρσορας πάνω από το τελικό σημείο των αντικειμένων, ώστε να μπορεί ο χρήστης να καθορίσει σημεία στην επέκταση.
Insertion	το σημείο εισαγωγής αντικειμένων όπως ένα block ή ένα κείμενο
Perpendicular	δίνει για ένα επιλεγμένο γεωμετρικό αντικείμενο την κάθετό του
Tangent	την εφαπτομένη ενός τόξου, κύκλου, έλλειψης, ελλειπτικού τόξου, τόξου ή στίγματος
Nearest	δίνει το πλησιέστερο σημείο σε ένα αντικείμενο, όπως τόξο, κύκλο, έλλειψη, ελλειπτικό τόξο, γραμμή, σημείο, ακτίνα
Apparent Intersection	πατάει στην φαινομενική τομή δύο αντικειμένων που δεν τέμνονται στον τρισδιάστατο χώρο αλλά μπορεί να φαίνεται να τέμνονται στην τρέχουσα προβολή.
Parallel	Περιορίζει ένα νέο τμήμα γραμμής, τμήμα polyline , ή ακτίνα ώστε να είναι παράλληλο με ένα υπάρχον γραμμικό αντικείμενο που αναγνωρίζεται τοποθετώντας τον κέρσορα

None	Απενεργοποιεί τη λειτουργία snap

Πίνακας 5.1: Τα είδη των Osnaps και ο ρόλος τους

Αυτές είναι οι λειτουργίες των snaps όταν ενεργοποιηθούν από το χώρο σχεδίασης όπως φαίνεται στο σχήμα 5.1. Για να αυτοματοποιηθεί η ενεργοποίηση των snaps μέσω του κώδικα της AutoLISP χρησιμοποιείται η μεταβλητή **Osmode** η οποία συνδυαστικά με κάποιους αριθμούς που αντιστοιχούν στα snaps ενεργοποιούνται. Για παράδειγμα η εντολή (setvar “osmode” 128) ενεργοποιεί το snap Perpendicular που αντιστοιχεί στον αριθμό 128. Στον παρακάτω πίνακα φαίνονται οι αντιστοιχίες των αριθμών με το κάθε snap.

Κωδικός	**Αντιστοιχία σε Osnap**
0	None
1	Endpoint
2	Midpoint
4	Center
8	Node
16	Quadrant
32	Intersection
64	Insertion
128	Perpendicular
256	Targent
512	Nearest
1024	Quick

Πίνακας 5.2: Αντιστοιχία αριθμών με λειτουργία Osnap

Ένα παράδειγμα φαίνεται στο σχήμα 5.3 όπου ενεργοποιείται αυτόματα το είδος Nearest του osnap με την εκτέλεση του Osmode συνδυαστικά με τον αριθμό 512 όπως φαίνεται στον Πίνακα 5.1.2.

```
(setvar "osmode" 512) ;energopoiei automata to Osnap Nearest

(setq point1 (getpoint "\Epilekse to 1o shmeio point1:"))
(setq point2 (getpoint point1 "\Epilekse to 2o shmeio point2:"))
```

Σχήμα 5.3: Εφαρμογή osmode σε κώδικα

5.2 Τριγωνομετρία και χρήσιμες συναρτήσεις

Τα περισσότερα προβλήματα μπορούν να λυθούν με τις βασικές τριγωνομετρικές συναρτήσεις, όμως η AutoLISP προσφέρει κάποιες επιπλέον συναρτήσεις που μπορούν να λύσουν και πιο σύνθετη τριγωνομετρία.

5.2.1 Συνάρτηση Trans

Η Trans επιστρέφει μία συντεταγμένη ή μια λίστα μετατόπισης και η σύνταξή της είναι,

(trans λίστα_συντεταγμένων κώδικας_UCS κώδικας_UCS προαιρετικά_T/nill)

Η πρώτη παράμετρος είναι ένα σημείο αναφοράς, η δεύτερη παράμετρος είναι ένας κώδικας UCS που δείχνει σε ποιο σύστημα συντεταγμένων είναι εκφρασμένο αυτό το σημείο, ο επόμενος κώδικας UCS δείχνει σε ποιο σύστημα θα μεταφερθεί αυτό το σημείο και τέλος προστίθεται προαιρετικά μια παράμετρος T ή nill. Αν αυτή η παράμετρος αποτιμηθεί σε T τότε η πρώτη παράμετρος αντιμετωπίζεται σαν μια μετατόπιση και όχι σαν μια τιμή ενός σημείου. Οι κώδικες UCS που υπάρχουν είναι,

Κώδικας	Σύστημα Συντεταγμένων
0	Παγκόσμιο Σύστημα
1	Τρέχον Σύστημα
2	Σύστημα για το τρέχον επίπεδο απεικόνισης

Πίνακας 5.3: Κώδικες για Συστήματα Συντεταγμένων

5.2.2 Συνάρτηση Atan

Η σύνταξη της Atan έχει σύνταξη,

(atan αριθμός 2ος_αριθμός_προαιρετικά)

και επιστρέφει την αντίστροφη εφαπτομένη της 1ης παραμέτρου σε ακτίνια. Αν δίνονται δύο παράμετροι τότε επιστρέφει την αντίστροφη εφαπτομένη της πρώτης παραμέτρου διαιρεμένη με την δεύτερη παράμετρο.

5.2.3 Συνάρτηση Inters

Η Inters επιστρέφει τα σημεία στα οποία τέμνονται δύο διανύσματα,

(Inters σημείο1_δ1 σημείο2_δ1 σημείο1_δ2 σημείο2_δ2 προαιρετικά_T/nill)

Τα δύο πρώτα σημεία είναι τα άκρα το διανύσματος 1 δ1, και τα άλλα δύο σημεία του διανύσματος 2 δ2. Αν η τελευταία παράμετρος αποτιμηθεί σε nill τότε η Inters συνεχίζει να ψάχνει για το σημείο αλληλοτομίας των δύο διανυσμάτων ανεξάρτητα αν το σημείο είναι μεταξύ των δεδομένων των δεδομένων διανυσμάτων ή όχι.

5.2.4 Συναρτήσεις Sin και Cos

Η συνάρτηση Sin επιστρέφει το ημίτονο Sin μια γωνίας σε πραγματικό αριθμό. Η γωνία που θα δοθεί στον τύπο (sin γωνίας) πρέπει να είναι σε ακτίνια.
Η Cos επιστρέφει σε πραγματικό αριθμό το συνημίτονο cos μιας γωνίας που θα δοθεί σε ακτίνια (cos γωνίας).

6. ΔΙΕΥΚΟΛΥΝΣΗ ΤΟΥ ΧΡΗΣΤΗ

Ένα πρόγραμμα γίνεται πιο εύκολο και βοηθητικό για το χρήστη όταν εκτελεί κάποια κομμάτια με τη δική του πρωτοβουλία. Όταν για παράδειγμα εντοπίζει και κάνει γνωστά κάποια λάθη, όταν δίνονται και παίρνονται εύκολα δεδομένα και πληροφορίες και όταν τα προγράμματα εκτελούνται γρήγορα με μία ανθεκτικότητα προς τα λάθη. Μην ξεχνάμε επίσης ότι δύο απλά αλλά ταυτόχρονα πολύ βοηθητικά βήμα, ειδικά για μεγάλα κομμάτια κώδικα είναι η προσθήκη σχολίων και η αποθήκευση των αρχείων με κατατοπιστικό όνομα όπως έχει ήδη προαναφερθεί. Είναι πολύ πιθανό να χρειαστεί να κληθεί ένα αρχείο μέσα σε ένα άλλο, μέσα από ένα πλήθος προγραμμάτων που όμως αν έχουν ονομαστεί κατάλληλα θα γλιτώνουν σίγουρα χρόνο και υπομονή από το χρήστη. Τα σχόλια επίσης βοηθάνε για υπενθύμιση της λειτουργίας των εντολών ενώ ταυτόχρονα μπορούν οπτικά να διαχωρίσουν τον κώδικα σε κομμάτια.

6.1 Ανάγνωση θέσης του σταυρονήματος δυναμικά

Το σταυρόνημα καθώς κινείται στο χώρο, δίνει τη θέση του σε ένα σύστημα συντεταγμένων ανά πάσα στιγμή, ως προς ένα σημείο αναφοράς. Αυτό αποτελεί μια δυναμική διαδικασία καθώς όσο κινείται δίνει αριθμό συντεταγμένων μια δυνατότητα που δίνεται στην AutoLISP με τη χρήση της συνάρτησης **Grread**, **(grread προαιρετική_παράμετρος)**. Η συνάρτηση μπορεί να διαβάσει δεδομένα από διάφορες συσκευές (πληκτρολόγιο, ποντίκι, σταυρόνημα κλπ.) επιστρέφοντας ένα αποτέλεσμα της μορφής (ακέραιος ακέραιος/λίστα_συντεταγμένων). Ο πρώτος ακέραιος είναι ένας κωδικός εισαγωγής και αντιπροσωπεύει το είδος των δεδομένων που δέχθηκε, αλλά και μέσω ποιας συσκευής τα δέχτηκε, μέσα από μια λίστα αριθμών όπως φαίνεται στον πίνακα 6.1. Το δεύτερο στοιχείο θα είναι είτε ένας ακέραιος είτε μια λίστα συνταγμένων ανάλογα με το αν δόθηκε μια λίστα ή ένας αριθμός αντίστοιχα.

Σημείωση για τον Πίνακα 6.1: Ο κάθε κωδικός ακολουθείται από το δικό του στοιχείο, ο κωδικός 3 για παράδειγμα θα ακολουθείται από έναν κωδικό ASCII και επιπλέον ο κάθε αριθμός αντιπροσωπεύει τον τρόπο με τον οποίο δόθηκαν τα δεδομένα(πληκτρολόγιο, σταυρόνημα, δυναμική ανάγνωση κ.ο.κ.). Έστω το (3 συντεταγμένες), το νούμερο 3 δίνει την πληροφορία ότι επιλέχθηκε η θέση του σταυρονήματος με τη χρήση του ποντικιού, και το δεύτερο στοιχείο «συντεταγμένες» θα περιλαμβάνουν τη λίστα των συντεταγμένων αυτών.

Κωδικός Εισαγωγής	Σημασία του
(2 κωδικός_ASCII)	Πιέσθηκε ένα πλήκτρο του πληκτρολογίου(το καταλαβαίνουμε από το 2) και το δεύτερο στοιχείο αντιπροσωπεύει τον χαρακτήρα που δόθηκε από το πληκτρολόγιο, στο σύστημα ASCII.
(3 συντεταγμένες)	Επιλέχθηκε θέση του σταυρονήματος με το ποντίκι επιστρέφοντας μια λίστα συντεταγμένων.
(4 αριθμός_θέσης_μενού)	Έγινε επιλογή από το μενού και ο αριθμός αντιπροσωπεύει τον αριθμό της θέσης του μενού, μετρώντας από πάνω προς τα κάτω.
(5 συντεταγμένες)	Δυναμική ανάγνωση του σταυρονήματος όσο αυτό μετακινείται πάνω στην οθόνη. Επιστρέφει πίσω κάθε φορά τις συντεταγμένες του σημείου που βρίσκεται, σε μορφή λίστας (ο κωδικός επιστρέφεται

	μόνο όταν δοθεί μια αληθής παράμετρος στην Grread)
(6 αριθμός_πλήκτρου)	Πιέσθηκε πλήκτρο στο ποντίκι, ο αριθμός του οποίου αντιπροσωπεύεται από το δεύτερο στοιχείο «αριθμός_πλήκτρου»
(7-10 αριθμός_ορθογώνιου)	Επιλέχθηκε κάτι από το μενού της ταμπλέτας, το 7 αντιπροσωπεύει την πρώτη ομάδα του μενού, το 8 τη δεύτερη και ούτω καθεξής. Ο αριθμός του ορθογωνίου αναφέρεται στον αριθμό του ορθογωνίου της ταμπλέτας.
(11 αριθμός_ορθογωνίου)	Επιλέχθηκε εντολή από τη γραμμή εργαλείων στο πάνω μέρος του παραθύρου σχεδίασης (home,project,file,edit κλπ.)
(12 συντεταγμένες)	Αν έχουν χρησιμοποιηθεί δύο Grread η μια μετά την άλλη και η πρώτη έχει επιστρέψει κωδικό 6 τότε η δεύτερη θα επιστρέψει κωδικό 12 και το δεύτερο στοιχείο θα είναι οι συντεταγμένες τη στιγμή που πιέσθηκε το πλήκτρο.
(13 αριθμός_θέσης_μενού)	Επιλογή εντολής από το μενού όχι χρησιμοποιώντας το ποντίκι αλλά τα βέλη του πληκτρολογίου. Η θέση του μενού θα αποθηκευτεί στο «αριθμός_θέσης_μενού».

Πίνακας 6.1: Κωδικοί εισαγωγής της Grread

Πως λειτουργεί,

Αν γράψουμε στο command την εντολή (grread) αυτόματα την ενεργοποιούμε. Στη συνέχεια έστω ότι επιλέγουμε ένα σημείο τυχαία με το σταυρόνημα και πατάμε πάνω σε αυτό με το ποντίκι. Αυτόματα θα βγάλει ένα αποτέλεσμα του τύπου (3 (5.0 7.0 0.0)) που σημαίνει ότι βρισκόμαστε στο σημείο με συντεταγμένες (5,7,0) και το έχουμε επιλέξει στο χώρο με το σταυρόνημα «κλικάροντας» το ποντίκι στο σημείο αυτό.

6.2 Έλεγχος λαθών

Η AutoLISP βοηθάει στον εντοπισμό των λαθών τυπώνοντας κάποια μηνύματα λάθους. Αυτό όμως δεν είναι πάντα αρκετό γι' αυτό συνίσταται ανά διαστήματα μέσα στο κομμάτι του κώδικα με τη χρήση της princ σε «στρατηγικές θέσεις» να εκτυπώνονται κάποιες μεταβλητές επιβεβαιώνοντας ότι έχουν πάρει τις τιμές που μας εξυπηρετούν χωρίς να έχει ολοκληρωθεί απαραίτητα όλο το πρόγραμμα προλαβαίνοντας αλυσιδωτά λάθη.

6.2.1 Συνάρτηση χειρισμού λαθών

Ένα σύνηθες πρόβλημα που συναντάται, είναι η διακοπή ενός προγράμματος πριν την ολοκλήρωσή του. Αν όμως ένα πρόγραμμα διακοπεί και οι ρυθμίσεις του AutoCAD παραμείνουν στην κατάσταση που θα εξυπηρετούσε εκείνες τις γραμμές κώδικα, τότε θα δημιουργηθούν άλλα προβλήματα. Για παράδειγμα, αν ένα Osnap είχε ενεργοποιηθεί στη λειτουργία Nearest για την εκτέλεση του προγράμματος, πρέπει οπωσδήποτε με την ολοκλήρωση ή τη διακοπή του, το Osnap να επανέλθει στην αρχική του κατάσταση None ή γενικότερα στη μορφή που είχε ακριβώς πριν ξεκινήσει να εκτελείται το παρών πρόγραμμα. Πως επιτυγχάνεται αυτό; Με τη χρήση της συνάρτησης **error** της AutoLISP η οποία μπορεί

να γραφτεί στην αρχή του κώδικα προτού ξεκινήσει η σύνταξη της βασικής συνάρτησης. Για παράδειγμα, πριν αρχίσει ο κώδικας του σχήματος 5.3 παραπάνω, προστίθενται οι εξής γραμμές κώδικα,

```
(defun *error* (msg)
(setvar "osmode" 0)
(princ msg)
(princ)
)
```

Σχήμα 6.1: Πρόσθετο κομμάτι error για το Σχήμα 5.3

Με αυτό τον τρόπο εμφανίζεται ένα μήνυμα λάθους msg που παίρνει σαν όρισμα την error αλλά την ίδια στιγμή επαναφέρει και το osnap στην κατάσταση None με τη χρήση της setvar. Αν θέλουμε να γενικεύσουμε την error, μπορούμε να αντικαταστήσουμε τη δεύτερη γραμμή με την (setvar “osmode” *osnap) η οποία επαναφέρει το osnap στην κατηγορία που είχε ενεργοποιηθεί μέχρι και πριν αρχίσει η εκτέλεση του τρέχον προγράμματος. Δηλαδή, αν το osnap πριν την εκτέλεση αυτού του κώδικα ήταν σε λειτουργία Nearest κι εμείς για τις ανάγκες του προγράμματος το ενεργοποιήσαμε σε Perpendicular τότε με την εκτέλεση της error και την εμφάνιση του μηνύματος λάθους θα επανέλθει στη λειτουργία Nearest.

6.2.2 Τα συχνότερα λάθη

Τα περισσότερα λάθη προκύπτουν συνήθως από έλλειψη παρενθέσεων, ορθογραφικά και προβλήματα χαρακτήρων όπως φαίνονται ανακεφαλαιωτικά στον πίνακα 6.2 . Ένα άλλο συχνό λάθος είναι το μπέρδεμα μεταξύ αριθμών και γραμμάτων όπως το αγγλικό l με τον αριθμό 1 ή το 0 με το O όπως επίσης και ένα ακόμα είναι η ανάθεση ενός αριθμού σε ένα αλφαριθμητικό (setq string1 “5”).

Μήνυμα λάθους	Ερμηνεία μηνύματος
error: malformed list	*λάθος: κακώς σχηματισμένη λίστα* Λόγω λανθασμένου αριθμού παρενθέσεων ή λάθος τοποθέτησής τους
error: extra right paren	*λάθος: επιπλέον δεξιά παρένθεση*
error: Insufficient string space	*λάθος: ανεπαρκής χώρος string* πιθανόν να μην έχουν κλείσει τα εισαγωγικά σε ένα αλφαριθμητικό, πχ (setq “timh=1)
Error: bad argument type	*Λάθος: λάθος τύπος παραμέτρου* Εάν νωρίτερα έχει τεθεί ένα αριθμός σε ένα αλφαριθμητικό όπως (setq string1 “5”) και μετά προσπαθήσουμε να κάνουμε την πράξη (1+ string1)

Πίνακας 6.2: Τα συχνότερα μηνύματα λάθους και η ερμηνεία τους

6.3 Επιλογή και επεξεργασία ομάδας αντικειμένων

Η διαδικασία της επιλογής των αντικειμένων σε ομάδες με τη χρήση της **ssget** (Selection Set get) αποτελεί μία από τις βασικές διευκολύνσεις για το χρήστη καθώς με αυτόν τον τρόπο μπορεί να επεξεργαστεί γρήγορα και εύκολα ένα πλήθος αντικειμένων και όχι το καθένα μεμονωμένα. Η ssget επιλέγει είτε αντικείμενα που βασίζονται σε κάποια φίλτρα είτε δίνοντας τη δυνατότητα στο χρήστη να επιλέξει αυτά που θέλει τυπώνοντας το μήνυμα “select objects” είτε να ορίσει εξ ’αρχής τις συντεταγμένες του αντικειμένου όπως θα δούμε

παρακάτω. Έστω ότι έχει προηγηθεί ο ορισμός τεσσάρων σημείων από το χρήστη με τη χρήση της εντολής,

```
(setq pt1 '(0.0 0.0 0.0) pt2 '(5.0 5.0 0.0) pt3 '(4.0 1.0 0.0) pt4 '(2.0 6.0 0.0))
```

τότε στον πίνακα 6.3 με τη χρήση των παραπάνω σημείων βλέπουμε κάποια παραδείγματα εφαρμογής της ssget. Οι συνδυασμοί που μπορούν να γίνει είναι πάρα πολλοί ανάλογα πάντα πως εξυπηρετείται ο χρήστης και ποια λίστα επιλογής θα τον διευκόλυνε περισσότερο.

Κατηγορίες σύνταξης ssget	**Λειτουργία της**
(setq A (ssget))	Ζητάει από το χρήστη μία γενική επιλογή αντικειμένων και τα τοποθετεί σε set επιλογής A
(setq A (ssget pt2))	Δημιουργεί μια λίστα επιλογής A η οποία θα περιέχει ένα αντικείμενο με συντεταγμένες αυτές του pt2
(setq A (ssget "**P**"))	Δημιουργεί μια λίστα επιλογής A η οποία θα πάρει τα ορίσματα που έχουν αποθηκευτεί στην πιο πρόσφατη λίστα επιλογής που δημιουργήθηκε
(setq A (ssget "P'"((0 . "LINE"))))	Μια πιο στοχευμένη εφαρμογή της δυνατότητας “P” είναι αυτό το παράδειγμα όπου επιλέγει συγκεκριμένα τις γραμμές-LINE που έχουν αποθηκευτεί στην πιο πρόσφατη λίστα επιλογής που δημιουργήθηκε.
(setq A (ssget "**L**"))	Αποθηκεύει στη λίστα A την τελευταία οντότητα που σχεδιάστηκε
(setq A (ssget "**W**" pt1 pt2))	Αποθηκεύει στην A οντότητες μέσα από ένα απλό παράθυρο **W**indow του οποίου οι γωνίες βρίσκονται στις συντεταγμένες των σημείων pt1, pt2 αντίστοιχα. Εδώ θα μπορούσα να συντάσσεται και ως **(ssget “W” ‘(0 0) ‘(5 5))** αν θέλαμε να ορίσουμε μέσα στην ίδια την ssget τα σημεία
(setq A (ssget "W" pt1 pt2 '((8 . "FLOOR9"))))	Ένα πιο συγκεκριμένο παράδειγμα εφαρμογής της επιλογής W είναι το παρόν παράδειγμα όπου αποθηκεύει στην A οντότητες μέσα από ένα απλό παράθυρο Window που βρίσκονται όμως ταυτόχρονα και στο layer “FLOOR9”
(setq A (ssget "**F**" (list pt1 pt2 pt3)))	Αποθηκεύει στην A ότι αντικείμενο βρει διασχίζοντας τα σημεία/διαδρομή (**F**ence) με συντεταγμένες pt1, pt2 και pt3.
(setq A (ssget "**X**"))	Αποθηκεύει στη λίστα επιλογής A όλα τα αντικείμενα που βρίσκονται σε μία βάση δεδομένων X, για παράδειγμα, **(setq A(ssget “X” lay))** χρησιμοποιεί το X για να πει στην ssget να χρησιμοποιήσει τη

	λίστα lay με σκοπό να δημιουργήσει την επιλεγμένη ομάδα A.
(setq A(ssget "X" '((0 . "CIRCLE"))))	Ένα παράδειγμα εφαρμογής της δυνατότητας “X” όπου θέτει στη λίστα A όλα τα αντικείμενα της βάσης δεδομένων “CIRCLE” δηλαδή που είναι κυκλικά.
(setq A (ssget '((0 . "TEXT"))))	Αποθηκεύει στη λίστα A μόνο αντικείμενα κειμένου.

Πίνακας 6.3: Κατηγορίες σύνταξης της συνάρτησης ssget

Ένα παράδειγμα εφαρμογής της ssget είναι η ίδια η συνάρτηση **Getlayer** της οποίας ο κώδικας φαίνεται στο σχήμα 6.2 και ο ρόλος της είναι να επιλέγει όλο το περιεχόμενο ενός layer που θα έχει καθορίσει ο χρήστης και να γίνει σε αυτό η αντίστοιχη επεξεργασία, για παράδειγμα να διαγραφούν όλα τα αντικείμενα ενός layer.

```
(defun GETLAYER (/ lay)
(setq lay (list (cons 8
(strcase (getsting "\Dwse onoma layer:"))))
(ssget "X" lay)  )
```

Σχήμα 6.2: Συνάρτηση Getlayer

Πρώτα ζητείται από το χρήστη ένα layer, το όνομα του οποίου χρησιμοποιείται για να δημιουργηθεί το ζεύγος τελείας ((8. “onoma layer”)) το οποίο περιλαμβάνεται σε μία λίστα με την εντολή list και στη συνέχεια αντιστοιχίζεται στην μεταβλητή lay. Αυτή χρησιμοποιείται στη συνέχεια στην ssget η οποία χρησιμοποιώντας την παράμετρο “X” δίνει ουσιαστικά την εντολή να δημιουργήσει μια επιλεγμένη ομάδα χρησιμοποιώντας τη λίστα φίλτρου lay. Ο αριθμός 8 αναφέρεται σε μία ομάδα που αποτελείται από οντότητες του ίδιου layer, αν ήταν οντότητες του ίδιου block θα είχαμε τον αριθμό 2 και ούτε καθεξής όπως φαίνεται στον πίνακα 6.4.
Έστω λοιπόν ότι θέλουμε να διαγράψουμε (Erase) όλα τα ανοίγματα του layer “anoigmata” σε μία κάτοψη που έχουμε σχεδιάσει. Πρώτα φορτώνεται το αρχείο με τη συνάρτηση Getlayer και στη συνέχει επιλέγεται το Erase. Στο μήνυμα “Erase select objects:” πληκτρολογήστε (getlayer) και τότε δίνεται το μήνυμα “ Dwse onoma layer:” εκεί πληκτρολογήστε το όνομα του layer δηλαδή anoigmata (δεν έχει σημασία αν το όνομα είναι στα ελληνικά ή οποιαδήποτε άλλη γλώσσα αρκεί να είναι ακριβώς ίδιο με εκείνο του layer) και με διπλό enter διαγράφονται όλα τα ανοίγματα της κάτοψης που βρίσκονταν σε αυτό το layer εύκολα και γρήγορα. Αυτό είναι ένα μοτίβο της λειτουργίας του, ωστόσο μπορεί να χρησιμοποιηθεί για οποιαδήποτε εντολή όπως περιστροφή των αντικειμένων (Rotate), όπως την αντιγραφή των αντικειμένων (copy) ότι εξυπηρετεί κάθε φορά το χρήστη.

Κωδικοί Ομάδων	Χαρακτηριστικά
0	Τύπος οντότητας
2	Όνομα Block
6	Όνομα τύπου γραμμής
7	Όνομα τύπου γραφής
8	Όνομα layer
38	Ανύψωση
39	Πάχος

62	Αριθμός χρώματος
66	Block που περιέχει attributes
210	3D διάνυσμα εξώθησης

Πίνακας 6.4: Κωδικοί ομάδων αποδεκτοί από την ssget

7. ΠΑΡΑΔΕΙΓΜΑ ΕΦΑΡΜΟΓΗΣ AUTOLISP ΣΤΗ ΣΧΕΔΙΑΣΗ ΑΠΟΧΕΤΕΥΣΗΣ

Η παρούσα διπλωματική εργασία, κλείνει με το έβδομο και τελευταίο κεφάλαιο το οποίο σε συνδυασμό με τη θεωρία των προηγούμενων κεφαλαίων, έρχεται με ένα αναλυτικό παράδειγμα εφαρμογής της AutoLISP να ολοκληρώσει την εικόνα αυτού του προγραμματιστικού περιβάλλοντος. Η εφαρμογή έγινε πάνω σε ένα χαρακτηριστικό και σύνηθες πρόβλημα που έχει να αντιμετωπίσει ένας χρήστης του AutoCAD που δεν είναι άλλο από τη σχεδίαση των συνδέσεων ενός αποχετευτικού συστήματος για μια δεδομένη κάτοψη. Σκοπός της εφαρμογής είναι αφενός η καλύτερη κατανόηση της γλώσσας και αφετέρου η επίδειξη της αυτοματοποίησης που προσφέρει διευκολύνοντας και εξυπηρετώντας το χρήστη στην καθημερινή του εργασία.

7.1 Η αποχέτευση θεωρητικά

Ως αποχέτευση ορίζεται μια υποδομή-εγκατάσταση, με βασική λειτουργία την απομάκρυνση των λυμάτων, αστικών ή βιομηχανικών και των όμβριων υδάτων από κατοικημένες περιοχές. Οι εγκαταστάσεις της αποχέτευσης είναι το ίδιο σημαντικές με εκείνες της ύδρευσης και της θέρμανσης για ένα κτήριο, ίσως και σημαντικότερες. Οι κίνδυνοι που μπορεί να προέλθουν λόγω τυχόν σφαλμάτων σε αυτές, φέρουν σε κίνδυνο ολόκληρες περιοχές, μιας και αποτελούν φανερή εστία μικροβίων αλλά και κίνδυνο για τη δημόσια υγεία. Σε περιοχές χωρίς εγκαταστάσεις δικτύων αποχέτευσης λυμάτων παρατηρούνται μολύνσεις τόσο επιφανειακά όσο και υπόγεια του εδάφους καθώς και εμφάνιση οσμών, ενώ η απουσία αποχέτευσης όμβριων υδάτων μετατρέπει τους δρόμους σε ρέματα με ότι κινδύνους αυτό συνεπάγεται. Ένα σύστημα αποτελείται από ένα σύνολο εγκατεστημένων στοιχείων όπως οι σωλήνες, τα φρεάτια, οι υπόνομοι, οι συσκευές κλπ. που συνδυαστικά μεταφέρουν αποβαλλόμενα ύδατα και στερεά από τις ανθρώπινες δραστηριότητες ή από τις βροχές. Η αποχέτευση διαφέρει από το σύστημα απορροής όμβριων υδάτων, όμως παρόλα αυτά είναι συχνό το φαινόμενο όπου αυτά τα δύο συγχέονται μεταξύ τους ανάλογα τι εξυπηρετεί κάθε φορά καλύτερα το δίκτυο.

Τα συστήματα αποχέτευσης χωρίζονται σε μεικτά και σε διπλά ή χωριστά. Στα μεικτά, αποχετεύονται μαζί τα λύματα από τις λεκάνες των W.C.,τα νερά από τους υπόλοιπους υδραυλικούς υποδοχείς αλλά και τα βρόχινα ύδατα. Τα διπλά ή χωριστά, διαχωρίζουν τα λύματα των W.C. από τα λύματα των υπόλοιπων υδραυλικών υποδοχών και των βρόχινων. Το χωριστό σύστημα που είναι και πιο σπάνιο, διαθέτει δύο υπονόμους για τα ακάθαρτα και τα βρόχινα νερά αντίστοιχα. Αυτή η μορφή εγκατάστασης εξουδετερώνει κάποια προβλήματα που θα προκαλούνταν από ένα μεικτό σύστημα, όπως είναι οι πλημμύρες στους δρόμους καθώς τα νερά φεύγουν καθυστερημένα. Επιπλέον στο μεικτό, τα ύδατα επιτρέπεται να έχουν έως μέγιστη θερμοκρασία τους 35^{o}C και ακόμα πιο σημαντικό είναι το μειονέκτημα όπου επιτρέπεται τα αποχετευτικά νερά να διοχετευτούν ακαθάριστα αρκεί να ανακατεύονται με πέντε μέρη νερού. Επιπλέον πλεονέκτημα για το χωριστό σύστημα, είναι σχετικά με τα βρόχινα νερά όπου δεν μεταφέρονται καθόλου στις εγκαταστάσεις καθαρισμού ενώ η μέγιστη ποσότητα που μπορεί να δεχθεί το κάθε δίκτυο υπολογίζεται χωριστά.

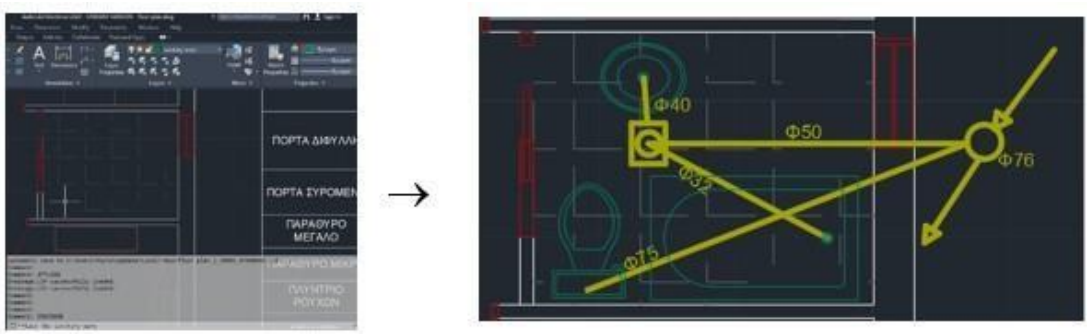
Στην αποχέτευση μπορούμε να μιλήσουμε για αρκετά είδη αγωγών που έχουμε συναντήσει στο πέρασμα του χρόνου με κάποια από αυτά να έχουν πλέον καταργηθεί και κάποια άλλα να χρησιμοποιούνται κατά κόρον. Οι αγωγοί διαχωρίζονται βάσει της διαμέτρους τους σε μικρούς και μεγάλους, ως προς τη μορφή εξυπηρέτησής τους, όμβριων και αερισμού και ανάλογα το υλικό κατασκευής τους σε χαλύβδινοι σωλήνες, P.V.C κλπ. Μικροί σωλήνες θεωρούνται εκείνοι με διάμετρο μικρότερη των 3 ιντσών ενώ μεγάλοι εκείνοι με μέγεθος

διαμέτρου μεγαλύτερης εκείνης των 3 ιντσών. Να σημειωθεί ωστόσο, ότι οι μικροί σωλήνες πρέπει να τοποθετούνται με κλίση περίπου 1/50 και οι μεγάλοι με μεγαλύτερη των 2 f/p.s.

Οι σωλήνες που χρησιμοποιούνται ως επί το πλείστων σήμερα, είναι οι πλαστικοί P.V.C. καθώς ταυτόχρονα προσφέρουν στεγανότητα, χαμηλό βάρος και αντοχή σε χημικές διαβρώσεις με μοναδικό θα λέγαμε μειονέκτημα, τη χαμηλή αντοχή τους στις μηχανικές καταπονήσεις. Πηγαίνοντας κάποια χρόνια προς τα πίσω, σαν υλικό κατασκευής σωλήνων αποχέτευσης χρησιμοποιούνταν ο πηλός για την παρασκευή των πηλοσωλήνων, που χαρακτηρίζονταν από χημική, μηχανική αλλά και φυσική αντοχή στο πέρασμα του χρόνου, όμως παρόλα αυτά μιλάμε για ακριβά και εύθραυστα υλικά, γι' αυτό και καταργήθηκαν αργότερα. Για περισσότερο από εκατό χρόνια, χρησιμοποιούνταν και οι σωλήνες από σκυρόδεμα οπλισμένου και μη, όμως μειονεκτούν στο βάρος τους, στην αντοχή τους σε συγκεκριμένες διαβρωτικές ουσίες και στη στεγανότητά τους. Όσον αφορά τις δεκαετίες '70 και '80, η Ελλάδα χρησιμοποιούσε σωλήνες τσιμέντου με προσμίξεις ινών αμιάντου προσδίδοντας επιπλέον αντοχή εφελκυσμού. Μια τελευταία κατηγορία σωλήνων είναι οι μεταλλικοί οι οποίοι έχουν πλέον καταργηθεί λόγω της προφανής ευαισθησίας τους στις διαβρώσεις με συνέπεια να αποτελούν κίνδυνο για τη δημόσια υγεία.

7.2 Αναλυτικό παράδειγμα κώδικα AutoLISP

Ο κώδικας AutoLISP πάνω στην αποχέτευση, σχεδιάστηκε με σκοπό να διευκολύνει την εργασία του χρήστη ο οποίος θα φορτώσει μια φορά αρχικά το πρόγραμμα μέσα στο πρόγραμμα AutoCAD και στη συνέχεια απλά θα καλέσει στο command τη φράση "drainage" όπως θα καλούσε την εντολή "line" για να σχεδιάσει μια γραμμή. Ο κώδικας εξυπηρετεί το χρήστη σε τρείς βασικές διαδικασίες, **1)** την εισαγωγή όλων των Blocks που αφορούν τη σχεδίαση μιας αποχέτευσης (είδη υγιεινής, ηλεκτρικές συσκευές, είδη αποχέτευσης) **2)** το σχεδιασμό των συνδέσεων μεταξύ των ειδών υγιεινής και των ειδών αποχέτευσης, υπολογίζοντας ταυτόχρονα την απαραίτητη διατομή ανάλογα το είδος της σύνδεσης, που στη συνέχεια εμφανίζεται σε μορφή κειμένου πάνω από κάθε σύνδεση και **3)**τον υπολογισμό της συνολικής διατομής του συλλέκτη στον οποίον θα καταλήξουν τα λύματα από τα είδη που ο χρήστης θα επιλέξει. Όσον αφορά το συλλέκτη, η διατομή του και πάλι υπολογίζεται αυτόματα ανάλογα με τον αριθμό και την κατηγορία των ειδών των οποίων τα λύματα θα καταλήξουν σε αυτόν και επιπλέον το αποτέλεσμα του υπολογισμού αυτού θα φανεί σε μορφή κειμένου στην κάτοψη του χρήστη. Ο κώδικας, που αποτυπώνεται εξολοκλήρου στη συνέχεια του κεφαλαίου, μπορεί να επικολληθεί ολόκληρος στο παράθυρο σύνταξης της AutoLISP του AutoCAD και με την εκτέλεσή του να πάρετε τα αποτελέσματα που αναλύθηκαν. Ακολουθεί η ανάλυση των γραμμών των κώδικα αλλά και η ανάλυση των αποτελεσμάτων που δίνει κατά την εκτέλεσή του, φτάνοντας σταδιακά όπως φαίνεται παρακάτω, από την αριστερή στη δεξιά εικόνα.



Σχήμα 7.1: Κάτοψη πριν και μετά την εφαρμογή του κώδικα

Το πρόγραμμα ξεκινάει με μία συνάρτηση που ονομάζεται “iscompatible”, και δεν είναι της μορφής (defun c:iscompatible (sunduasmos /)) που σημαίνει ότι δεν θα την καλέσει ο χρήστης κάποια στιγμή στο command window , αλλά θα κληθεί εσωτερικά κάποιας άλλης συνάρτησης για να την εξυπηρετήσει με τρόπο που θα δούμε και παρακάτω. Άρα, προτού ξεκινήσει η ανάλυση, μένουμε στο ότι, όλες οι συναρτήσεις της μορφής (defun….) χρησιμοποιούνται μέσα σε άλλες συναρτήσεις.

```
(defun iscompatible (sunduasmos / )

  (setq compatibleitems '(("floor sink" "washing machine")
                          ("washing machine" "floor sink")
                          ("sewer collector" "sink")
                          ("sink" "sewer collector")
                          ("sewer collector" "toilet")
                          ("toilet" "sewer collector")
                          ("sewer collector" "floor sink")
                          ("floor sink" "sewer collector")
                          ("bathroom sink" "floor sink")
                          ("floor sink" "bathroom sink")
                          ("bathtub" "floor sink")
                          ("floor sink" "bathtub" )
                          ("shower tray" "floor sink")
                          ("floor sink" "shower tray")

                          ))
  (member sunduasmos compatibleitems)
)
```

Σχήμα 7.2: Συνάρτηση iscompatible

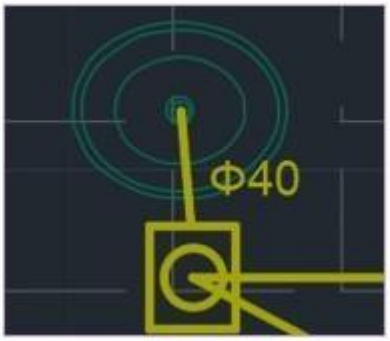
Σκοπός της iscompatible, είναι να επικυρώσει έναν έγκυρο συνδυασμό από αντικείμενα που θα επιλέξει ο χρήστης για να δημιουργήσει μια σύνδεση αποχέτευσης. Για παράδειγμα,
σε πραγματικές συνθήκες μέσα σε μια κάτοψη δεν συναντάει κανείς μια σύνδεση μεταξύ ενός νιπτήρα και μιας μπανιέρας, έτσι δημιουργήθηκε η λίστα “compatibleitems” που δεδομένου ότι υφίστανται σε ένα πραγματικό σχέδιο, θα επιτρέψει στη συνέχεια στο πρόγραμμα να κάνει μόνο τις αποδεκτές συνδέσεις. Η εντολή (member sunduasmos compatibleitems) λειτουργεί σαν δείκετης βάζοντας τη μεταβλητή “sunduasmos” σε μια διαδικασία αναζήτησης, αν ανήκει στη λίστα compatibleitems. Έτσι διασταυρώνεται ότι οποιοσδήποτε “sunduasmos” δοθεί θα είναι κομμάτι της λίστας “compatibleitems”, δηλαδή είναι έγκυρος και σε πραγματικές συνθήκες μελέτης.

```
(defun calculatePhi (sunduasmos / )
  (setq q40 '(("washing machine" "floor sink")
              ("floor sink" "washing machine")
              ("bathtub" "floor sink")
              ("floor sink" "bathtub" )
              ("shower tray" "floor sink")
              ("floor sink" "shower tray")
              ("bathroom sink" "floor sink")
              ("floor sink" "bathroom sink") ))

   (setq q100 '(("toilet" "sewer collector")
                ("sewer collector" "floor sink") ))

   (setq q50 '(("sewer collector" "floor sink")
               ("floor sink" "sewer collector") ))

   (setq q63 '( ("sink" "sewer collector")
                ("sewer collector" "toilet")  ))

  (cond
    ((member sunduasmos q40) "40")
    ((member sunduasmos q100)"100")
    ((member sunduasmos q50)"50")
    ((member sunduasmos q63) "63")
    (t nil))
  )
```

Σχήμα 7.3: Κώδικας συνάρτησης calculatePhi

Η επόμενη συνάρτηση calculatePhi υπολογίζει τη διατομή Φ του κάθε σωλήνα, ανάλογα με το είδος που συνδέει. Δεδομένου ότι θα της δοθεί ένας “sunduasmos”, ανάλογα ποιος θα είναι αυτός, θα αντιστοιχηθεί και σε μια τιμή διατομής. Έστω ότι ο χρήστης θέλει να συνδέσει έναν νιπτήρα με το σιφόνι του μπάνιου, δίνεται δηλαδή ο συνδυασμός (“floor sink” “bathroom sink”), τότε αυτός θα αντιστοιχηθεί στη μεταβλητή q40. Έπειτα, χρησιμοποιώντας την εντολή member όπως παραπάνω, έχει επιτευχθεί η αντικατάσταση του συνδυασμού (“floor sink” “bathroom sink”) με το “40” και έτσι δίνοντας έναν συνδυασμό παίρνουμε τελικά πίσω μια τιμή. Αυτός είναι και ο αριθμός που αναλογεί στη διατομή του σωλήνα της συγκεκριμένης σύνδεσης και που στη συνέχεια θα τυπωθεί σε μορφή κειμένου πάνω από το σωλήνα σύνδεσης, μεταξύ του “floor sink” και του“bathroom sink”.

```
(defun phiText (point phi / )
  (entmake
    (list
      (cons 0 "MTEXT")
      (cons 100 "AcDbEntity")
      (cons 100 "AcDbMText")
      (cons 1 (strcat "Ø" phi))
      (cons 10 point)
      (cons 40 0.10)
    )
  )
)
```



Σχήμα 7.4: Κώδικας και εφαρμογή συνάρτησης phiText

Η συνάρτηση Phitext, που ακολουθεί είναι υπεύθυνη για την εμφάνιση του κειμένου πάνω από τη γραμμή σύνδεσης, για παράδειγμα Φ40 δηλαδή, διατομή σωλήνα 40. Σαν ορίσματα παίρνει ένα σημείο point που θα τοποθετήσει το κείμενο και την τιμή phi που είναι η διατομή που αντιστοιχεί σε κάθε σύνδεση και υπολογίζεται μέσω συνάρτησης. Όταν παρακάτω χρειάζεται να υπολογιστεί το phi η εντολή είναι κάπως έτσι, (setq phi (calculatePhi combination)). Δηλαδή, εφόσον πληρούνται οι προϋποθέσεις της calculatephi που αναφέρθηκαν παραπάνω, για έναν “combination” που θα της δοθεί σαν όρισμα, το phi τελικά θα πάρει κάποια από τις τιμές 40,100,50,63. Αυτό το νούμερο μαζί με το γράμμα Φ τοποθετούνται σε ένα point και συγκεκριμένα στο mdp, δηλαδή το midpoint της γραμμής σύνδεσης που υπολογίζεται από τις παρακάτω εντολές.

(setq phi (calculatePhi combination))

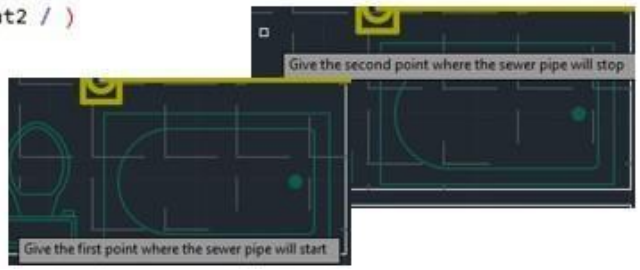
(drainageLine point1 point2)

(setq mdp (polar point1 (+ (angle point1 point2)0.5) (/ (distance point1 point2) 2.0)))

(phiText mdp phi))

(Το σύμβολο δίπλα από την εντολή strcat είναι το ελληνικό γράμμα Φ της διατομής. Επειδή όμως δεν αναγνωρίζεται σαν ελληνικό γράμμα στο παράθυρο τη σύνταξης του κώδικα, εμείς βλέπουμε ένα «περίεργο» σύμβολο. Με την εκτέλεση του κώδικα ωστόσο αποτυπώνεται στην κάτοψη κανονικά το Φ40.)

```
(defun drainageLine (point1 point2 / )
  (entmake
    (list
      (cons 0 "LINE")
      (cons 10 point1)
      (cons 11 point2)
      )
    )
  )
```



Σχήμα 7.5: Κώδικας και εφαρμογή συνάρτησης drainageLine

Η drainageLine είναι εκείνη που μέσω της εντολής Line θα σχεδιάσει τη γραμμή σύνδεσης μεταξύ των αντικειμένων που θα επιλέξει ο χρήστης. Ως μεταβλητές εισόδου παίρνει τα point1 και point2 που δίνονται από το χρήστη μεταξύ των οποίων θα σχεδιαστεί η γραμμή. Έτσι ξεκινάει ζητώντας από το χρήστη το point1 μέσω του μηνύματος “Give the first point the sewer pipe will start” και στη συνέχεια του ζητάει το point2 με την εντολή “Give the second point where the sewer collector will stop”. Τα δύο σημεία point1 και point2 ζητούνται άμεσα από το χρήστη και η εντολή που είναι υπεύθυνη γι’ αυτό, φαίνεται παρακάτω στη συνάρτηση drainage. Θα εμφανιστούν αρκετές φορές συναρτήσεις της μορφής,

(defun ονομα_συνάρτησης (δεδομένα εισόδου/)

(entmake

(list

(cons 0 “κύρια εντολή”)

(cons ότι_μας_εξυπηρετεί)

.

.

όσες (cons…) χρειαζόμαστε

)))

Εδώ έχουμε μια λίστα που περιλαμβάνει πολλά διαφορετικά conditions “cons”. Το (cons 0) θα αναφέρεται πάντα στη κύρια εντολή που πρόκειται να εκτελεστεί όπως “LINE”. Ο κάθε αριθμός ανάλογα, μέσα από μια λίστα αριθμών, αντιπροσωπεύει και μια διαφορετική ενέργεια, το 10 για παράδειγμα δέχεται σαν όρισμα το starting point της line και το 11 το last point της Line κ.ο.κ. Έτσι, κάθε συνάρτηση ανάλογα των αναγκών της σε ορίσματα που έχει, κωδικοποιείται από άλλους αριθμούς για να καλύψει τις ανάγκες της προκειμένου να εκτελεστεί.

```
(defun insertBlock (insertionPoint blockName /)
  (entmake
    (list
      (cons 0 "INSERT")
      (cons 2 blockName)
      (cons 10 insertionPoint)
      (cons 42 1)
      (cons 43 1)
      )
    )
  )

(defun insertUserBlock (point/)
  (setq name (getstring t "\nEnter one of \n'washing machine',\n'bathtub'\n'sink' \n'bathroom sink' \n'shower tray' \n'toilet'"))
  (setq point (getpoint "\nGive the Block insertion point"))


  (insertBlock point name)


  (setq ent (entget (entlast)))
  (setq kind (cdr (nth 9 ent)))
  (list kind point)
  )

(defun insertDrainageBlock (/)
  (setq ui nil)
  (while (/= (cadr ui) 13)

    (setq ui nil)
    (while (= nil ui) (setq ui (grread)))
    (if (= (car ui) 3)
        (progn
          (setq ip (cadr ui))

     (insertUserBlock ip))
 )))
```

Σχήμα 7.6: Κώδικας συναρτήσεων insertBlock, insertUserBlock, insertDrainageBlock

Όλο το παραπάνω κομμάτι του κώδικα αφορά την αυτοματοποιημένη εισαγωγή των Blocks που ο χρήστης θα επιλέξει να εισάγει και συγκεκριμένα όσον αφορά τα είδη υγιεινής ενός μπάνιου. Η insertBlock δέχεται σαν όρισμα το όνομα του Block, blockname και το σημείο όπου θα εισαχθεί insertionPoint και με τη χρήση της εντολής insert γίνεται η τοποθέτησή τους. Η αλληλουχία έχει ως εξής, η insertUserBlock καλεί την insertBlock για να εκτελεστεί και αφού ολοκληρωθεί, έρχεται η insertDrainageBlock που καλεί με τη σειρά της την insertUserBlock. Η insertUserBlock δημιουργεί την επαφή μεταξύ του χρήστη και του κώδικα, καθώς του ζητά να δώσει το είδος που θέλει να εισάγει και το σημείο που θέλει να το τοποθετήσει. Αφού λάβει αυτά τα δεδομένα τα «στέλνει» στην insertBlock η οποία χρησιμοποιώντας τα, εκτελεί την εντολή insert. Τέλος η insertDrainageBlock είναι η συνάρτηση που δημιουργεί ουσιαστικά τη λούπα για όλες εκείνες τις φορές που ο χρήστης θα θελήσει να εισάγει κάτι και όσο εκείνος δίνει δεδομένα τόσο εκείνη τοποθετεί τα αντικείμενα. Πιο αναλυτικά…..

```
(defun insertBlock (insertionPoint blockName /)
  (entmake
    (list
      (cons 0 "INSERT")
      (cons 2 blockName)
      (cons 10 insertionPoint)
      (cons 42 1)
      (cons 43 1)
      )
    )
  )
```

Σχήμα 7.7: Κώδικας συνάρτησης InsertBlock

Η συνάρτηση insertBlock, δεδομένου ότι είναι συνάρτηση της μορφής (defun όνομα (ορίσματα /)) σημαίνει ότι κι εκείνη έχει συνταχθεί με σκοπό να εξυπηρετήσει τη συνάρτηση μέσα στην οποία θα κληθεί. Αποτελείται από μια λίστα με κύρια την εντολή (cons 0 “INSERT”) καθώς με τον αριθμό 0 δείχνει ότι θα ακολουθήσει ένα αλφαριθμητικό που υποδεικνύει την ενέργεια, εδώ “insert”. Κάθε αριθμός που έχει επιλεχθεί και εδώ είναι χαρακτηριστικός και έχει επιλεχθεί από μια λίστα «συμβολικών κωδικών», εδώ για την εντολή insert (Insert DXF) ανάλογα με αυτό που εξυπηρετεί τη συνάρτηση. Από τις εντολές (cons 42 1) και (cons 43 1) για παράδειγμα το 42 είναι υπεύθυνο για “Y scale factor” και το 43 για “Z scale factor” αντίστοιχα και παίρνουν fixed τιμές το 1.

```
(defun insertUserBlock (point/)
  (setq name (getstring t "\nEnter one of \n'washing machine',\n'bathtub'\n'sink' \n'bathroom sink' \n'shower tray' \n'toilet'"))
  (setq point (getpoint "\nGive the Block insertion point"))


  (insertBlock point name)


  (setq ent (entget (entlast)))
  (setq kind (cdr (nth 9 ent)))
  (list kind point)
  )
```

Σχήμα 7.8: Κώδικας συνάρτησης InsertUserBlock

Η isertUserBlock καλεί εσωτερικά την insertBlock η οποία θα εκτελεστεί για ένα δεδομένο συνδυασμό (point name) που θα πληκτρολογήσει ο χρήστης. Η συνάρτηση entget, δίνοντας της ως είσοδο μια οντότητα, επιστρέφει μια λίστα με όλα τα χαρακτηριστικά αυτής, όπως για παράδειγμα σε ποιο layer ανήκει, σε ποιο Block κλπ. Η entlast έχει το ρόλο μεταβλητής προσωρινής μνήμης, κρατάει δηλαδή προσωρινά το όνομα της τελευταίας μεταβλητής που είτε δόθηκε στο πρόγραμμα γραπτώς είτε επιλέχθηκε. Εδώ βάσει της σειράς εντολών, τελευταίο όνομα θα είναι το όρισμα του “name” δηλαδή ότι έδωσε ο χρήστης, άρα με την (entget (entlast)) παίρνουμε μέσα στην “ent” μια λίστα χαρακτηριστικών του αλφαριθμητικού που θα επιλέξει ο χρήστης παραπάνω. Επειδή για το παρόν πρόβλημα μια λίστα με όλα αυτά τα χαρακτηριστικά μας είναι περιττή και μας ενδιαφέρει να κρατήσουμε μόνο το όνομα της οντότητας, επιλέγεται το (nth 9 ent) δηλαδή το δέκατο στοιχείο της ent (αρίθμηση από το 0) που θα είναι μια παρένθεση της μορφής (2. “bathtub”). Για τη λήψη ακόμα πιο «καθαρού» ονόματος επιλέγεται το δεύτερο στοιχείο αυτής της λίστα με την εντολή (cdr) και τελικά kind=”bathtub” που στη συνέχεια το τοποθετείται σε μια κοινή λίστα με το point.

```
(defun insertDrainageBlock (/)
  (setq ui nil)
  (while (/= (cadr ui) 13)

    (setq ui nil)
    (while (= nil ui) (setq ui (grread)))
    (if (= (car ui) 3)
        (progn
          (setq ip (cadr ui))

     (insertUserBlock ip))
 )))
```

Σχήμα 7.9: Κώδικας συνάρτησης insertDrianageBlock

Η insertDrainageBlock αναλύεται και παρακάτω στη συνάρτηση sumphi και σκοπός της είναι να δημιουργήσει μια δομή επανάληψης. Όταν ο χρήστης τοποθετεί τα είδη υγιεινής απλά χρησιμοποιεί το σταυρόνημα κλικάροντας πάνω στην κάτοψη και σταδιακά γεμίζει το σχέδιό του. Για την αποφυγή οποιασδήποτε πιο χρονοβόρας διαδικασίας χρησιμοποιείται αυτή η μορφή της λούπας επαναλήψεων που σε συνδυασμό με την grread αναμένει τα ορίσματα από το χρήστη.

```
(defun insertUserDrainageItems (point/)
  (setq name (getstring t "\nEnter one of \n'floor sink'\n'sewer collector'"))
  (setq point (getpoint "\nGive the Block insertion point"))
  (insertBlock point name)
  (setq ent (entget (entlast)))
  (setq kind (cdr (nth 9 ent)))
  (list kind point)
  )

(defun insertDrainageItems (/)
 (setq ui nil)
  (while (/= (cadr ui) 13)

    (setq ui nil)
    (while (= nil ui) (setq ui (grread)))
    (if (= (car ui) 3)
        (progn
          (setq ip (cadr ui))
     (insertUserDrainageItems ip)
   )
  )))
```

Σχήμα 7.10: Κώδικας συναρτήσεων InsertUserDrainageItems, insertDrainageItems

Οι συναρτήσεις inserUserDrianageItems και insertDrainageItems κάνουν ακριβώς ότι και οι insertUserBlock, insertDrainageBlock αντίστοιχα, με τη μόνη διαφορά ότι αντί να εισάγει είδη υγιεινής όπως πριν, τώρα εισάγει τα είδη αποχέτευσης.

```
(defun insertUserElectricalDevices (point/)
  (setq name (getstring t "\nEnter one of \n'washing machine'"))
  (setq point (getpoint "\nGive the Block insertion point"))
  (insertBlock point name)
  (setq ent (entget (entlast)))
  (setq kind (cdr (nth 9 ent)))
  (list kind point)
  )

(defun insertElectricaDevices (/)
  (setq ui nil)
  (while (/= (cadr ui) 13)

    (setq ui nil)
    (while (= nil ui) (setq ui (grread)))
    (if (= (car ui) 3)
        (progn
          (setq ip (cadr ui))
     ( insertUserElectricalDevices ip)
   )
  )))
```

Σχήμα 7.11: Κώδικας συναρτήσεων insertUserElectricalDevices, insertElectricalDevices

Όμοια και για τις συναρτήσεις insertUserElectricalDevices και insertElectricalDevices οι οποίες βοηθούν το χρήστη να εισάγει τις ηλεκτρικές συσκευές.

```
(defun connectDrainage (entName1 point1 entName2 point2 /)
  (setq combination (cons entName1 (cons entName2 '())))

  (if (iscompatible combination)
    (progn
      (setq phi (calculatePhi combination))
      (drainageLine point1 point2)
   (setq mdp (polar point1  (+ (angle point1 point2)0.5) (/ (distance point1 point2) 2.0)))
      (phiText mdp  phi))
    (print "Incompatible, please create only acceptable connections")))
```

Σχήμα 7.12: Κώδικας συνάρτησης connectDrainage

Η connectDrianage, είναι η υπεύθυνη συνάρτηση για τη σχεδίαση των συνδέσεων μεταξύ δύο ειδών. Δέχεται σαν ορίσματα τα ονόματα των δύο οντοτήτων που πρόκειται να συνδέσει

entName1 και entName2 και τα δύο σημεία point1,point2, εκεί που θα ξεκινάει και εκεί που θα τελειώνει η γραμμή σύνδεσης αντίστοιχα. Η μεταβλητή combination κάνει τα δύο ονόματα μια λίστα φέροντας τα στη μορφή των ορισμάτων της compatibleitems που έχει η συνάρτηση iscompatible που αναφέρθηκε νωρίτερα. Με αυτόν τον τρόπο αργότερα, διασταυρώνεται ότι όντως αυτός ο συνδυασμός των οντοτήτων ανήκει σε αυτή τη λίστα ώστε να γίνει αποδεκτή η σύνδεση, ή αντίστοιχα όταν δεν εγκρίνεται και εμφανίζεται ένα μήνυμα λάθους προς το χρήστης για να δώσει μόνο αποδεκτούς συνδυασμούς αντικειμένων. Όταν είναι αποδεκτός ωστόσο, υπολογίζεται το Φ (phi) τη σύνδεσης, σχεδιάζεται η γραμμή σύνδεσης και εμφανίζεται σε μορφή κειμένου στο σημείο mdp το Φ σε συνδυασμό με τον αριθμό της διατομής για την παρούσα σύνδεση.

```
(defun c:drainage (/)
  (setvar 'clayer "sanitary ware")
  (princ "Place the sanitary ware")
  (insertDrainageBlock)
  (setvar 'clayer "drainage")
  (princ "Place the drainage items")
  (insertDrainageItems)
  (setvar 'clayer "electrical_devices")
  (princ "Place the Electrical Devices")
  (insertElectricaDevices)
  (setvar 'clayer "drainage")
  (while (eq "y" (getstring nil "\nDo you want to draw a connection?? [y/n]"))
  (setq object1 (entsel "\nGive the first point where the sewer pipe will start \n"))
  (setq object2 (entsel "\nGive the second point where the sewer pipe will stop"))
  (setq point1 (cadr object1))
  (setq point2 (cadr object2))
  (setq entity1 (entget (car object1))) ;(2."bathroom")
  (setq entity2 (entget (car object2)))
  (setq kind1 (cdr (nth 9 entity1))) ;("bathroom")
  (setq kind2 (cdr (nth 9 entity2)))
  (connectDrainage kind1 point1 kind2 point2))

  (sumphi)
)
```

Σχήμα 7.13: Κώδικας συνάρτησης drainage

Η συνάρτηση drainage έρχεται για να ενώσει όλες τις παραπάνω συναρτήσεις. Είναι συνάρτηση της μορφής (defun c: όνομα συνάρτησης) δηλαδή ο χρήστης απλά γράφοντας στο command window το όνομα της συνάρτησης (drainage) λειτουργεί σαν να έγραφε την εντολή line και έτσι αυτοματοποιούνται οι διαδικασίες,

1) Εισαγωγή των Blocks (είδη υγιεινής, είδη αποχέτευσης, ηλεκτρικές συσκευές)

2) Σχεδίαση γραμμής σωλήνα μεταξύ των ειδών που θα επιλέξει ο χρήστης.

3) Υπολογισμός και αποτύπωση της διατομής του σωλήνα πάνω από τη γραμμή σύνδεσης, ανάλογα ποια είδη συνδέει

4) Υπολογισμός της διατομής του κεντρικού σωλήνα με την κλήση της συνάρτησης sumphi.

```
(defun sumphi (/)
(setq sumAWS 0)
(setq collectionPoint (cadr (entsel "\nGive the collection point")))
(setq ui nil ap 0.05)  ; User Input,selection APerture size
;; This ap sets the size of the crossing box that gets the block
(princ "\n Give whatever ends up in the sewer collector")
(while (/= (cadr ui) 13) ; Escapes on RETURN (ascii 13)
    (setq ui nil)
    (while (= nil ui) (setq ui (grread))) ; Wait for input
    (if (= (car ui) 3)                    ; Selection loop
        (progn
        (setq ip (cadr ui))   ; Extract Input Point
        (setq c1 (list (- (car ip) (/ ap 2.0)) (- (cadr ip) (/ ap 2.0)))) ; Crossing box pt1
        (setq c2 (list (+ (car ip) (/ ap 2.0)) (+ (cadr ip) (/ ap 2.0)))) ; Crossing box pt2

        (setq ss (ssget "c" c1 c2 '((8 . "sanitary ware,drainage,electrical_devices") (0 . "insert"))))
        ;crossing box from c1 and c2 for ssget
        (if (/= ss nil)
            (progn
            (setq antikeimeno (ssname ss 0))

            (setq eidos (cdr (nth 9 (entget antikeimeno))))
            (princ (strcat "\n" eidos " added (RETURN to end selection)"))

             (setq sumAWS (+ (calculateAWS eidos) sumAWS))
            )))
  (phiText collectionPoint (itoa (calculatephiTELIKO sumAWS)))
(princ "\nFinished!")

)
  (princ)
```

Σχήμα 7.14: Κώδικας συνάρτησης sumphi

Η sumphi είναι η συνάρτηση που χρησιμοποιείται για την ολοκλήρωση της σχεδίασης της αποχέτευσης του χώρου, υπολογίζοντας τη διατομή του συλλέκτη από όλα τα είδη του μπάνιου που έχει τοποθετήσει ο χρήστης έως τώρα. Όπως και σε μια πραγματική μελέτη, ο υπολογισμός του συλλέκτη εξαρτάται άμεσα από το τι θα καταλήξει σε αυτόν, γι' αυτό και ξεκινάει ζητώντας από το χρήστη πρώτα να δώσει το σημείο συλλογής των λυμάτων και στη συνέχεια όλα τα είδη που πρόκειται να καταλήξουν εκεί. Ακολουθεί μια δομή επανάληψης while η οποία εσωτερικά περιέχει μια δεύτερη while για την εισαγωγή των ειδών αποχέτευσης. Περιμένει από τον χρήστη να εισάγει τα δεδομένα του για ένα επιλεγμένο μέγεθος λούπας (εδώ 3), και στη συνέχεια υπολογίζει τα σημεία c1, c2 που είναι δύο αντικριστά σημεία ενός box. Δημιουργείται αυτό το crossing box, γιατί η ssget συνάρτηση που ακολουθεί, δεν λειτουργεί δίνοντάς της ένα σημείο αντικειμένου, αλλά δίνοντάς της ένα μέγεθος «κουτιού» που περικλείει αυτά τα αντικείμενα, δηλαδή για εμάς τα blocks που θέλουμε να λάβει υπόψιν της και συγκεκριμένα αυτά που θα καταλήγουν στον συλλέκτη. Με τη γραμμή *(setq ss (ssget "c" c1 c2 '((8 . "sanitary ware,drainage,electrical_devices") (0 . "insert"))))* εξασφαλίζεται ότι τα blocks που θα επιλεχθούν είναι μόνο εκείνα που ανήκουν στα layers "sanitary ware,drainage,electrical_devices" είναι δηλαδή ηλεκτρική συσκευή (για την περίπτωση του πλυντηρίου που συνδέονται με την αποχέτευση) ή είναι είδος υγιεινής ή είναι είδος αποχέτευσης. Η If που ακολουθεί μεταφράζεται ως, όταν η ssget δεν έχει κάποιο όνομα συνόλου, τότε να εξάγει από το selection set (ss) μόνο το πρώτο αντικείμενο, δηλαδή το όνομα του αντικειμένου. Γενικευμένα για την (ssname ss index), ss= A selection set και 0=index= the first element of set. Στην ουσία δημιουργείται μια «προσωρινή» ανανεώσιμη λίστα των αντικειμένων που συμβάλουν στον υπολογισμό της συνολικής διατομής του συλλέκτη. Όσο αυτή η λίστα ανανεώνεται, επιλέγοντας ο χρήστης τα αντικείμενα που θέλει, κρατείται το όνομα αυτού του αντικειμένου και προστίθεται στην μεταβλητή sumAWS το AWS του αντικειμένου που προστέθηκε. Η μπανιέρα έχει AWS=3, αν επιλεχθεί η μπανιέρα αυτόματα στο SumAWS μπαίνει ο αριθμός 3, αν στη συνέχεια επιλεχθεί και ο νιπτήρας που έχει AWS=1 αμέσως γίνεται η πράξη sumAWS=1+3+4 κ.ο.κ. Η συνάρτηση itoa εσωτερικά της phitext χρησιμοποιείται για να επιστρέψει τη μετατροπή ενός ακεραίου σε συμβολοσειρά, δηλαδή το όρισμα phi της (defun phiText (point phi /) που χρειάζεται για να εκτελεστεί και να αποτυπώσει τη διατομή του συλλέκτη.

```
(defun calculateAWS (onoma / )
  (cond
  ((= onoma "bathroom sink") 1)
  ((= onoma "toilet") 3)
  ((= onoma "bathtub") 3)
  ((= onoma "shower tray") 3)
  ((= onoma "sink") 3)
  ((= onoma "washing machine") 3)
  ((= onoma "floor sink") 1.3)
  )
)
```

Σχήμα 7.15: Κώδικας συνάρτησης calculateAWS

Αναφορικά των δύο συναρτήσεων calculateAWS και calculatephiTELIKO που καλούνται μέσα στην sumphi, όπως και σε μια πραγματική μελέτη, ο υπολογισμός του συλλέκτη εξαρτάται άμεσα από το τι θα καταλήξει σε αυτόν, Για παράδειγμα η μπανιέρα, αντιστοιχεί στον συντελεστή 3 (AWS=3) ,ο νιπτήρας σε AWS=1 κ.ο.κ. Σαν μεταβλητές εισόδου δέχεται μόνο το όνομα του block που θα καταλήξει στο συλλέκτη και ανάλογα αντιστοιχεί το κάθε είδος στο συντελεστή του.

```
(defun calculatePhiTELIKO (sumAWS / )
  (setq slope (getreal "\nGive 0.01 for slope 1:100 \n0.02 for slope 2:100 \nand 0.04 for slope 4:100 : "))

  (cond
    ((= slope 0.01)
    (cond
      ((<= sumAWS 1) (setq minPhiTELIKO 32))
      ((<= sumAWS 2) (setq minPhiTELIKO 38))
      ((<= sumAWS 5) (setq minPhiTELIKO 50))
      ((<= sumAWS 15)(setq minPhiTELIKO 76))
      ((<= sumAWS 84)(setq minPhiTELIKO 101))
      ((<= sumAWS 162)(setq minPhiTELIKO 127))
      ((<= sumAWS 300)(setq minPhiTELIKO 152))
      ((<= sumAWS 990)(setq minPhiTELIKO 177))
      ((<= sumAWS 1800)(setq minPhiTELIKO 254))
      ((<= sumAWS 3084)(setq minPhiTELIKO 305))
    )
    )
     ((= slope 0.02)
     (cond
       ((<= sumAWS 1) (setq minPhiTELIKO 32))
       ((<= sumAWS 2) (setq minPhiTELIKO 38))
       ((<= sumAWS 6) (setq minPhiTELIKO 50))
       ((<= sumAWS 18)(setq minPhiTELIKO 76))
       ((<= sumAWS 96)(setq minPhiTELIKO 101))
       ((<= sumAWS 261)(setq minPhiTELIKO 127))
       ((<= sumAWS 450)(setq minPhiTELIKO 152))
       ((<= sumAWS 1392)(setq minPhiTELIKO 177))
       ((<= sumAWS 2520)(setq minPhiTELIKO 254))
       ((<= sumAWS 4320)(setq minPhiTELIKO 305))
     ))
      ((= slope 0.04)
       (cond
       ((<= sumAWS 1) (setq minPhiTELIKO 32))
       ((<= sumAWS 3) (setq minPhiTELIKO 38))
       ((<= sumAWS 8) (setq minPhiTELIKO 50))
       ((<= sumAWS 21)(setq minPhiTELIKO 76))
       ((<= sumAWS 114)(setq minPhiTELIKO 101))
       ((<= sumAWS 264)(setq minPhiTELIKO 127))
       ((<= sumAWS 600)(setq minPhiTELIKO 152))
       ((<= sumAWS 2220)(setq minPhiTELIKO 177))
       ((<= sumAWS 3900)(setq minPhiTELIKO 254))
       ((<= sumAWS 6012)(setq minPhiTELIKO 305))
     ))
     ))
```

Διάμετρος σωλήνων (ίντσες)	Μέγιστος αριθμός μονάδων υποδοχέων		
	Κλίση 1:100	Κλίση 2:100	Κλίση 4:100
1 1/4	1	1	1
1 1/2	2	2	3
2	5	6	8
3	15	18	21
4	84	96	114
5	162	261	264
6	300	450	600
7	990	1392	2220
10	1800	2520	3900
12	3084	4320	6012

Σχήμα 7.16: Κώδικας συνάρτησης calculatephiTeliko και πίνακας επιλογής τιμών

Η συνάρτηση calculatePhiTELIKO υπολογίζει τη διατομή Φ του συλλέκτη ανάλογα με την κλίση του δαπέδου που θα ορίσει ο χρήστης και ανάλογα το όριο στο οποίο κυμαίνεται το άθροισμα όλων των μεταβλητών AWS που έχουν επιλεγεί έως εκείνη τη στιγμή. Έστω ότι διαλέγει να δώσει κλίση 0.01 (slope=0.01), αμέσως θα ψάξει να τοποθετήσει το sumAWS σε ένα εύρος τιμών, είτε sumAWS<=1, είτε sumAWS<=2 κλπ και ανάλογα θα θέσει μια τιμή minPhiTELIKO δηλαδή την ελάχιστη διατομή που θα πρέπει να έχει ο συλλέκτης. Οι τιμές αυτές δεν είναι τυχαίες και έχουν επιλεχθεί βάσει ενός ορισμένου πίνακα,(πίνακας 24 κεφάλαιο 2 βιβλίο «μηχανολογικές εγκαταστάσεις κτιρίων για τους μηχανολόγους μηχανικούς, Παναγιώτης Γ. Χαρώνης») όπου για γενικούς αποχετευτικούς αγωγούς υπό την προϋπόθεση ότι σε αυτούς δεν καταλήγουν τα όμβρια ύδατα, η διάμετρος τους προσδιορίζεται βάσει τον ολικό αριθμό των μονάδων και την κλίση του δαπέδου.

Με τη sumphi ολοκληρώνεται ο κώδικας για τη σχεδίαση ενός συστήματος αποχέτευσης, από «το μηδέν» σε ένα άδειο δωμάτιο, έως και το συλλέκτη από όλο το μπάνιο. Στόχος, η ανάδειξη της ευκολίας που μπορεί να προσφέρει κάθε μορφή αυτοματοποίησης που θα δημιουργούνταν για τους σκοπούς της σχεδίασης. Με την εκτέλεση του κώδικα, ο χρήστης άμεσα αντιλαμβάνεται την εξοικονόμηση σε κινήσεις και κατά συνέπεια σε χρόνο, που θα χρειαζόταν σε σχέση με την περίπτωση όπου σχεδίαζε την ίδια κάτοψη χωρίς καμία επιπλέον εφαρμογή της γλώσσας AutoLISP. Παρακάτω φαίνεται και όλος ο κώδικας αναλυτικά, σε μορφή που να μπορεί να αντιγραφεί εύκολα από τον αναγνώστη. Μόλις το αντιγράψει το αποθηκεύει στον υπολογιστή του με ένα αντιπροσωπευτικό όνομα, για παράδειγμα drainage, και με την εντολή appload στο command window φορτώνει τον κώδικα μέσα στο AutoCAD. Εφόσον η μόνη συνάρτηση που είναι της μορφής “defun c:” είναι η drainage, ο χρήστης πληκτρολογεί drainage στο command window κάνει Load και το πρόγραμμα είναι έτοιμο να τρέξει. Εκτελεί τα βήματα που έχουν γραφτεί αναλυτικά και σε περίπτωση που θέλει να παρακάμψει κάποιο βήμα (για να παράδειγμα να μην τοποθετήσει ηλεκτρικές συσκευές) πιέζει enter. Για να τοποθετήσει τα αντικείμενα που θέλει πάνω στην κάτοψή του πατάει το αριστερό κουμπί του ποντικιού, του ζητείται το είδος του block, το σημείο εισαγωγής κλπ. εισάγει όποιες και όσες πληροφορίες τον ενδιαφέρουν και βλέπει άμεσα το αποτέλεσμα.

```
(defun iscompatible (sunduasmos / )
(setq compatibleitems '(("floor sink" "washing machine")("washing machine" "floor sink")
("sewer collector" "sink")("sink" "sewer collector")("sewer collector" "toilet")
("toilet" "sewer collector")("sewer collector" "floor sink")("floor sink" "sewer collector")
("bathroom sink" "floor sink")("floor sink" "bathroom sink")("bathtub" "floor sink")
("floor sink" "bathtub" )("shower tray" "floor sink")("floor sink" "shower tray")))
(member sunduasmos compatibleitems))
(defun calculatePhi (sunduasmos / )
(setq q40 '(("washing machine" "floor sink")("floor sink" "washing machine") ("bathtub" "floor sink")("floor sink" "bathtub" )("shower tray" "floor sink")("floor sink" "shower tray" )("bathroom sink" "floor sink")("floor sink" "bathroom sink") )) (setq q100 '(("toilet" "sewer collector") ("sewer collector" "floor sink") ))
(setq q50 '(("sewer collector" "floor sink") ("floor sink" "sewer collector") ))
(setq q63 '( ("sink" "sewer collector")("sewer collector" "toilet")  ))
(cond
((member sunduasmos q40) "40") ;cond ((predicate1) (then do something 1))
((member sunduasmos q100)"100") ((member sunduasmos q50)"50")
((member sunduasmos q63) "63") (t nil)))
(defun phiText (point phi / )
(entmake
(list
(cons 0 "MTEXT")
(cons 100 "AcDbEntity")
(cons 100 "AcDbMText")
(cons 1 (strcat "F" phi))
(cons 10 point)
(cons 40 0.10) ) ))
(defun drainageLine (point1 point2 / )
(entmake
(list
(cons 0 "LINE")
(cons 10 point1)
```

```
(cons 11 point2))))
(defun insertBlock (insertionPoint blockName /)
(entmake
(list
(cons 0 "INSERT")
(cons 2 blockName)
(cons 10 insertionPoint)
(cons 42 1)
(cons 43 1) )) )
(defun insertUserBlock (point/)
(setq name (getstring t "\nEnter one of \n'washing machine',\n'bathtub'\n'sink' \n'bathroom sink' \n'shower tray' \n'toilet'"))
(setq point (getpoint "\nGive the Block insertion point"))
(insertBlock point name)
(setq ent (entget (entlast)))
(setq kind (cdr (nth 9 ent)))
(list kind point))
(defun insertDrainageBlock (/)
(setq ui nil)
(while (/= (cadr ui) 13)
(setq ui nil)
(while (= nil ui) (setq ui (grread)))
(if (= (car ui) 3)
(progn (setq ip (cadr ui))
(insertUserBlock ip)))))

(defun insertUserDrainageItems (point/)
(setq name (getstring t "\nEnter one of \n'floor sink'\n'sewer collector'"))
(setq point (getpoint "\nGive the Block insertion point"))
(insertBlock point name)
(setq ent (entget (entlast)))
(setq kind (cdr (nth 9 ent)))
(list kind point))
(defun insertDrainageItems (/)
(setq ui nil)
(while (/= (cadr ui) 13)
(setq ui nil)
(while (= nil ui) (setq ui (grread)))
(if (= (car ui) 3)
(progn (setq ip (cadr ui))
(insertUserDrainageItems ip)) )))
(defun insertUserElectricalDevices (point/)
(setq name (getstring t "\nEnter one of \n'washing machine'"))
(setq point (getpoint "\nGive the Block insertion point"))
(insertBlock point name)
(setq ent (entget (entlast)))
(setq kind (cdr (nth 9 ent)))
(list kind point))
```

```
(defun insertElectricaDevices (/)
(setq ui nil)
(while (/= (cadr ui) 13)
(setq ui nil)
(while (= nil ui) (setq ui (grread)))
(if (= (car ui) 3)
(progn (setq ip (cadr ui))
( insertUserElectricalDevices ip)))))
(defun connectDrainage (entName1 point1 entName2 point2 /)
(setq combination (cons entName1 (cons entName2 '())))
(if (iscompatible combination)
(progn
(setq phi (calculatePhi combination))
(drainageLine point1 point2)
(setq mdp (polar point1 (angle point1 point2) (/ (distance point1 point2) 2.0)))
(phiText mdp phi))
(print "Not compatible, please connect manually using DRAINAGE")))
(defun c:drainage (/)
(setvar 'clayer "sanitary ware")
(princ "Place the sanitary ware")
(insertDrainageBlock)
(setvar 'clayer "drainage")
(princ "Place the drainage items")
(insertDrainageItems)
(setvar 'clayer "electrical_devices")
(princ "Place the Electrical Devices")
(insertElectricaDevices)
(setvar 'clayer "drainage")
(while (eq "y" (getstring nil "\nDo you want to draw a connection? [y/n]"))
(setq object1 (entsel "\nGive the first point where the sewer pipe will start \n"))
(setq object2 (entsel "\nGive the second point where the sewer pipe will stop"))
(setq point1 (cadr object1))
(setq point2 (cadr object2))
(setq entity1 (entget (car object1))) ;(2."bathroom")
(setq entity2 (entget (car object2)))
(setq kind1 (cdr (nth 9 entity1))) ;("bathroom")
(setq kind2 (cdr (nth 9 entity2)))
(connectDrainage kind1 point1 kind2 point2))
(sumphi))
(defun calculateAWS (onoma / )
(cond
((= onoma "bathroom sink") 1)
((= onoma "toilet") 3)
((= onoma "bathtub") 3)
((= onoma "shower tray") 3)
((= onoma "sink") 3)
```

```
((= onoma "washing machine") 3)
((= onoma "floor sink") 1.3)))
(defun calculatePhiTELIKO (sumAWS / )
(setq slope (getreal "\nGive 0.01 for slope 1:100 \n0.02 for slope 2:100 \nand 0.04 for slope 4:100 : "))
(cond
((= slope 0.01)
(cond
((<= sumAWS 1) (setq minPhiTELIKO 32))
((<= sumAWS 2) (setq minPhiTELIKO 38))
((<= sumAWS 5) (setq minPhiTELIKO 50))
((<= sumAWS 15)(setq minPhiTELIKO 76))
((<= sumAWS 84)(setq minPhiTELIKO 101))
((<= sumAWS 162)(setq minPhiTELIKO 127))
((<= sumAWS 300)(setq minPhiTELIKO 152))
((<= sumAWS 990)(setq minPhiTELIKO 177))
((<= sumAWS 1800)(setq minPhiTELIKO 254))
((<= sumAWS 3084)(setq minPhiTELIKO 305))))
((= slope 0.02)
(cond
((<= sumAWS 1) (setq minPhiTELIKO 32))
((<= sumAWS 2) (setq minPhiTELIKO 38))
((<= sumAWS 6) (setq minPhiTELIKO 50))
((<= sumAWS 18)(setq minPhiTELIKO 76))
((<= sumAWS 96)(setq minPhiTELIKO 101))
((<= sumAWS 261)(setq minPhiTELIKO 127))
((<= sumAWS 450)(setq minPhiTELIKO 152))
((<= sumAWS 1392)(setq minPhiTELIKO 177))
((<= sumAWS 2520)(setq minPhiTELIKO 254))
((<= sumAWS 4320)(setq minPhiTELIKO 305))))
((=slope 0.04)
(cond
((<= sumAWS 1) (setq minPhiTELIKO 32))
((<= sumAWS 3) (setq minPhiTELIKO 38))
((<= sumAWS 8) (setq minPhiTELIKO 50))
((<= sumAWS 21)(setq minPhiTELIKO 76))
((<= sumAWS 114)(setq minPhiTELIKO 101))
((<= sumAWS 264)(setq minPhiTELIKO 127))
((<= sumAWS 600)(setq minPhiTELIKO 152))
((<= sumAWS 2220)(setq minPhiTELIKO 177))
((<= sumAWS 3900)(setq minPhiTELIKO 254))
((<= sumAWS 6012)(setq minPhiTELIKO 305))))))
(defun sumphi (/)
(setq sumAWS 0)
(setq collectionPoint (cadr (entsel "\nGive the collection point")))
(setq ui nil ap 0.05)
(princ "\nDWSE ANTIKEIMENO GIA TO TELIKO")
```

```
(while (/= (cadr ui) 13)
(setq ui nil)
(while (= nil ui) (setq ui (grread)))
(if (=(car ui) 3)
(progn
(setq ip (cadr ui))
(setq c1 (list (- (car ip) (/ ap 2.0)) (- (cadr ip) (/ ap 2.0))))
(setq c2 (list (+ (car ip) (/ ap 2.0)) (+ (cadr ip) (/ ap 2.0))))
(setq ss (ssget "c" c1 c2 '((8 . "sanitary ware,drainage,electrical_devices") (0 . "insert"))))
(if (/= ss nil)
(progn
(setq antikeimeno (ssname ss 0))
(setq eidos (cdr (nth 9 (entget antikeimeno))))
(princ (strcat "\n" eidos " added (RETURN to end selection)"))
(setq sumAWS (+ (calculateAWS eidos) sumAWS)))) )))
(phiText collectionPoint (itoa (calculatephiTELIKO sumAWS)))
(princ "\nFinished!")) (princ)
```

Προετοιμασία για τη σωστή εκτέλεση του κώδικα……

Για να εκτελεστεί ο κώδικας ορθά και να αποφευχθούν προβλήματα αντιστοιχίας πληροφοριών θα πρέπει ο χρήστης,

α) Όλα τα είδη υγιεινής να τα έχει τοποθετήσει σε ένα layer με όνομα sanitary ware

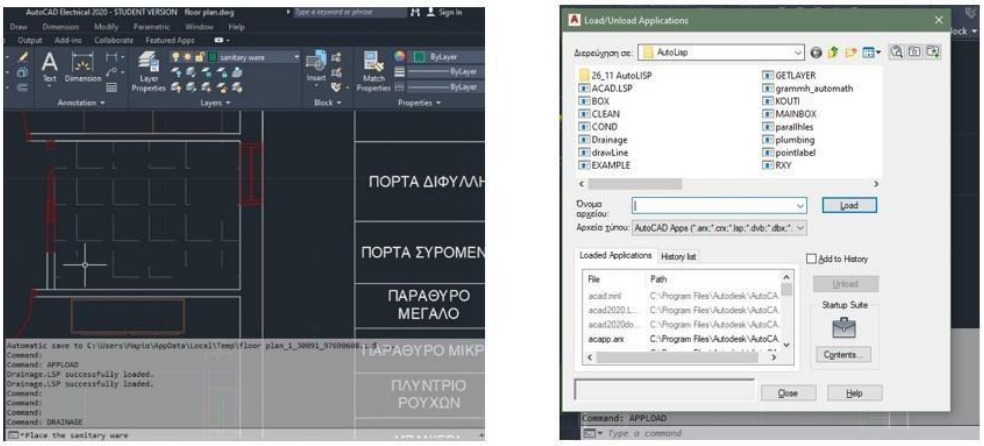
β) Όλα τα είδη αποχέτευσης να τα έχει τοποθετήσει σε ένα layer με όνομα drainage

γ) Τα ηλεκτρικά ήδη, κυρίως αυτά που θα συμβάλουν στη σχεδίαση της αποχέτευσης, όπως πλυντήριο ρούχων, να έχουν τοποθετηθεί σε ένα layer με όνομα electrical_devices.

δ) Τα ήδη που θα χρησιμοποιηθούν να έχουν οριστεί ως blocks τα ονόματα των οποίων θα φέρουν αντιστοιχία με τα ονόματα του κώδικα.

Για παράδειγμα το αντικείμενο που θα αντιπροσωπεύει μια μπανιέρα, θα πρέπει να μπει στο layer sanitary ware ως ένα Block με όνομα bathtub και όχι με όνομα μπανιέρα ή οτιδήποτε άλλο. Τα παραπάνω βήματα αφορούν την προεργασία για την εκτέλεση αποκλειστικά του παρόντος κώδικα. Ο χρήστης φυσικά μπορεί σε δεύτερο χρόνο να τροποποιήσει ότι τον εξυπηρετεί, αρκεί να έχει υπόψιν του ότι η AutoLISP εκτελείται πάνω στο AutoCAD και τα ορίσματα της πρέπει να τα εντοπίζει στη βάση του AutoCAD, στα layers του τα blocks του κλπ.

Ας δούμε αναλυτικά τι θα συναντήσει ο χρήστης κατά την εκτέλεση του κώδικα και πως θα πρέπει να το χειριστεί,



Σχήμα 7.17: Βήμα appload και load για τη φόρτωση του αρχείου .LSP

Αρχικά, η ανάγνωση του command window είναι πολύ βοηθητική για έναν χρήστη καθώς ανά πάσα στιγμή είναι σε θέση να θυμηθεί-κατανοήσει τι έχει εκτελέσει μέχρι εκείνη την ώρα το πρόγραμμα. Επίσης σε οποιαδήποτε κατάσταση αν έχει απορία για το τι περιέχει μια μεταβλητή, μπορεί να την καλέσει στο command γράφοντας (το όνομα της μεταβλητής) για παράδειγμα (kind) και εφόσον έχει πεπεραστεί ο κώδικας και την έχει ορίσει, αυτόματα θα δούμε το περιεχόμενό της. Βλέπουμε λοιπόν ότι ξεκινάει εκτελώντας την appload (έχει ήδη εκτελεστεί η εντολή appload όπως φαίνεται στην παρακάτω εικόνα), η οποία ανοίγει έναν φάκελο αρχείων του υπολογιστή για την επιλογή του αρχείου AutoLISP που θέλει να φορτώσει ο χρήστης. Γι' αυτόν ακριβώς το λόγο, μια οργανωμένη αποθήκευση και μια χαρακτηριστική ονομασία των αρχείων είναι πάντα βοηθητική, και αφού εντοπιστεί το αρχείο που μτον ενδιαφέρει επιλέγει load 1-2 φορές μέχρι να δοθεί στο command window η εντολή "όνομα_αρχείου.LSP successfully loaded.".

Σχήμα 7.18: Κλήση συνάρτησης drainage

Αφού γίνει η φόρτωση, καλείται στο command το όνομα τη συνάρτησης (defun c:.....), σε αυτό το παράδειγμα η μόνη συνάρτηση της μορφής (defun c) ήταν η (defun c:drainage) άρα πληκτρολογείται drainage και enter, και αν έχουν πάει όλα καλά, το AutoCAD πλέον αντιμετωπίζει τον κώδικά σαν μια εντολή που απλά θα την εκτελέσει. Ουσιαστικά αυτό που θα συμβεί, είναι σταδιακά να εκτελούνται τα βήματα της συνάρτησης drainage, η οποία για να φτάσει να ολοκληρωθεί περιέχει μια σειρά από συναρτήσεις, αυτές που αναλύθηκαν παραπάνω, όπου καλώντας η μια εσωτερικά της την άλλη, έχει δημιουργηθεί μια αλυσίδα ενεργειών που βήμα-βήμα θα γεμίσει την κάτοψη. Ακόμα, θα υπολογίσει και τις αποχετεύσεις του γι' αυτό και ξεκινάει ζητώντας από το χρήστη να τοποθετήσει τα είδη υγιεινής.

Σχήμα 7.19: Εναλλαγή στα layers

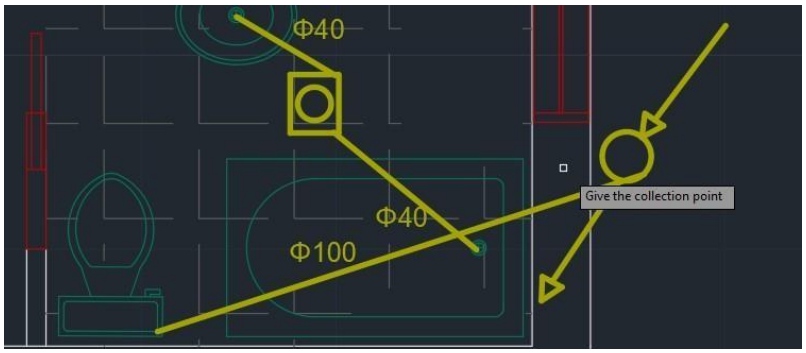
Παρατηρήστε ότι κάθε φορά που ζητάει από το χρήστη να τοποθετήσει αντικείμενα διαφορετικού Layer, γίνεται αυτόματη αλλαγή στην ομάδα των layer. Για παράδειγμα όταν ζητάει είδη υγιεινής στο πρόγραμμα αυτόματα ενεργοποιείται το sanitary ware όταν ζητάει είδη αποχέτευσης το drainage και ούτω καθεξής.

Έστω ότι θέλει να τοποθετήσει λοιπόν τα είδη υγιεινής. Στη εντολή place the sanitary ware πατάει με το σταυρόνημα κάπου πάνω στην κάτοψη και στην εντολή Enter one of 'washing machine','bathtub''sink''bathroom sink''shower tray''toilet' πληκτρολογεί αυτό που θέλει και επιλέγοντας το σημείο που θέλει να το τοποθετήσει απαντάει και στο ερώτημα Give the Block insertion point. Αν θέλει να συνεχίσει να τοποθετεί ήδη υγιεινής επιλέγει και πάλι ένα σημείο πάνω στην κάτοψη και εκτελεί την ίδια διαδικασία μέχρι να τοποθετήσει όσα θέλει. Αν θέλει να προχωρήσει σε είδη αποχέτευσης πατάει enter και εκτελεί την ίδια διαδικασία για να τοποθετήσει και αυτά τα είδη και αν θέλει να προχωρήσει τέλος στις ηλεκτρικές συσκευές πατάει και πάλι enter και προχωράει. Γενικά το enter λειτουργεί σαν κουμπί παράκαμψης ενώ αν δεν θέλει να παρακάμψει τη διαδικασία επιλέγει με το σταυρόνημα ένα σημείο πάνω στην κάτοψη και εκτελεί τις διαδικασίες. Αν ωστόσο μετανιώσει που παράκαμψε κάποιο βήμα ή θέλει να συμπληρώσει κάτι σε ένα βήμα πριν (π.χ. ο χρήστης βρίσκεται στο βήμα εισαγωγής ειδών αποχέτευσης και θέλει να τοποθετήσει κάτι στα είδη υγιεινής που ήταν ένα βήμα πριν) πατάει esc βγαίνει από το πρόγραμμα και πατάει ξανά στο command drainage. Το πρόγραμμα εκτελείται από την αρχή και πατώντας enter φτάνει στο βήμα που ήθελε να αναπληρώσει κάτι. Λειτουργεί σαν προσωρινή παύση για το πρόγραμμα και εκτελείται χωρίς πρόβλημα ξανά όταν ξανακληθεί. Αφού λοιπόν τοποθετήσει όλα τα είδη που θέλει για την κάτοψη του πατάει esc για να βγει προσωρινά για λίγο από το πρόγραμμα και ξεκινάει να οργανώσει λίγο τα αντικείμενα στο χώρο όπως τον βολεύει στο σχέδιο. Αφού ολοκληρώσει και αυτό το βήμα καλεί και πάλι drainage και πατώντας enter φτάνει στο βήμα για να αρχίσει να δημιουργεί τις συνδέσεις σωλήνων μεταξύ των ειδών. Θα γίνει αντιληπτό ότι έφτασε σε αυτό το βήμα όταν στο command εμφανιστεί το μήνυμα Do you want to draw a connection? [y n]. Όσο ο χρήσης απαντάει y (όπου y=yes συντομογραφία για διευκόλυνση) τόσο το πρόγραμμα θα τρέχει κανονικά. Έστω ότι η απάντηση είναι y,


```
Give the collection point*Cancel*
; error: Function cancelled
Command:
Command: drainage
Place the sanitary warePlace the drainage itemsPlace the Electrical Devices
Do you want to draw a connection?? [y/n]y
Give the first point where the sewer pipe will start
Give the second point where the sewer pipe will stop
"Incompatible, please create only acceptable connections"
Do you want to draw a connection?? [y n]
```

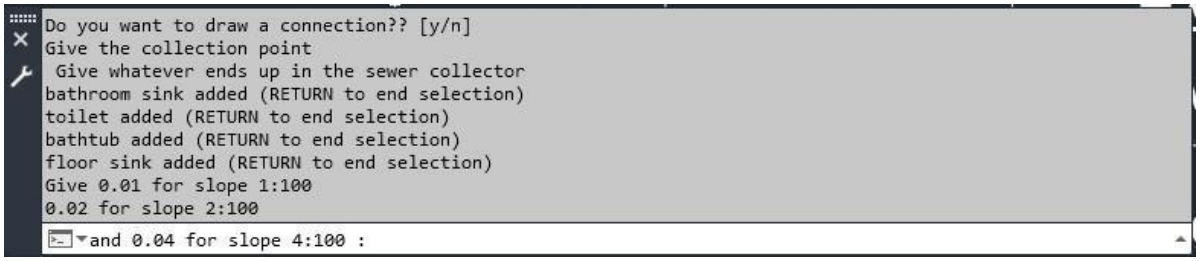
Σχήμα 7.20: Βήματα δημιουργίας σύνδεσης

αμέσως ζητείται το σημείο που θα ξεκινάει η σύνδεση και το σημείο που θα τελειώσει και ακολουθεί εκ νέου η ερώτηση αν θέλει να πραγματοποιήσει νέα σύνδεση. Αν απαντήσει n=no, δηλαδή δεν έχει να πραγματοποιήσει κάποια σύνδεση μεταξύ των ειδών περνάει το πρόγραμμα αμέσως στην ερώτηση, Give the collection point,



Σχήμα 7.21: Βήματα της sumphi

Στη συνέχεια ζητείται να δοθούν όλα τα είδη που καταλήγουν στο συλλέκτη (είδη αποχέτευσης, είδη υγιεινής, ηλεκτρικά είδη) και ο χρήστης επιλέγει τα αντικείμενα χρησιμοποιώντας το σταυρόνημα ενώ αμέσως μετά του ζητείται και η κλίση του δαπέδου. Επιβεβαιώστε ότι έχετε επιλέξει όλα τα αντικείμενα διαβάζοντας τι επιστρέφει το command window, όταν επιλεχθεί παραδείγματος χάρη ο νιπτήρας, το command window ενημερώνει το χρήστη επιστρέφοντας “bathroom sink added”.



Σχήμα 7.22: Εισαγωγή κλίσης

Τέλος δίνοντας την κλήση και πατώντας enter, αυτόματα αποτυπώνεται στο σημείο του συλλέκτη το γράμμα Φ της διατομής και δίπλα το νούμερο που έχει υπολογιστεί να καλύψει τις ανάγκες για τα είδη του μπάνιου.

Σημείωση: Στο στάδιο σύνδεσης των ειδών σε περίπτωση που επιλεχθεί μια σύνδεση που δεν είναι εφικτή, βάσει πραγματικών συνθηκών, επιστρέφει το μήνυμα "Incompatible, please create only acceptable connections" και ο χρήστης ερωτάται ξανά για να σχεδιάσει μια σύνδεση.

Σχήμα 7.23: Μήνυμα ολοκλήρωσης προγράμματος

Με την εμφάνιση του μηνύματος Finished και επίσημα, έχουν ολοκληρωθεί όλα τα στάδια του προγράμματος μας.

ΣΥΝΟΨΗ

Με την ολοκλήρωση αυτής της διπλωματικής εργασίας, έχουν παρουσιαστεί κάποιες βασικές εντολές που πρώτον διευκολύνουν το χρήστη και δεύτερον αυτοματοποιούν το περιβάλλον σχεδίασης AutoCAD. Οι λόγοι για τους οποίους θα έμπαινε κάποιος τόσο εις βάθος στο AutoCAD όσο την ίδια τη γλώσσα που το προγραμματίζει, θα ήταν πιθανώς δύο. Πρώτα απ' όλα για να κατανοήσει τον τρόπο με τον οποίο έχει "στηθεί" το AutoCAD αλλά και τη φιλοσοφία του αποδομώντας το και δεύτερον δημιουργώντας μια «βιβλιοθήκη» από δικά του προγράμματα, που με την κλήση τους κάθε φορά θα διαφοροποιούν το AutoCAD από ένα «συμβατό» πρόγραμμα που θα διέθετε ο καθένας ευκολότερα. Είναι δηλαδή, διαφορετικό να χρησιμοποιείται το κοινό AutoCAD και διαφορετικό να χρησιμοποιείται ένα "πειραγμένο" από την AutoLISP με προσαρμοσμένες επιπλέον δυνατότητες. Ειδικότερα για όσους έρχονται συχνά σε επαφή με αυτό το περιβάλλον όπως αρχιτέκτονες, μηχανικοί και ενδιαφερόμενοι παρόμοιου κλάδου, η κατανόηση και η εφαρμογή της AutoLISP αποτελεί «επένδυση» για το AutoCAD. Κατανοώντας και συντάσσοντας μία φορά ένα πρόγραμμα, αυτό θα διευκολύνει το χρήστη κάθε φορά που θα το καλεί και θα το εντάσσει στη σχεδίασή του. Αναφέρθηκαν κάποιες βασικές δυνατότητες και εφαρμογές της γλώσσας αλλά και πως μπορούν οι ίδιες να συνδυαστούν. Γίνεται αντιληπτό πως όσο πιο πολλά στοιχεία συνδυάζονται ταυτόχρονα τόσο πιο σύνθετο γίνεται το πρόβλημα προγραμματιστικά δημιουργώντας μια πληθώρα από προγράμματα το καθένα με άλλη εφαρμογή και άλλη πολυπλοκότητα. Ο χρήστης θα εφαρμόσει κάθε φορά αυτό που εκείνον θα εξυπηρετήσει καλύτερα, διευκολύνοντάς τον μέσω της αυτοματοποίησης που θα του προσφέρει.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] ΕΙΣΑΓΩΓΗ ΣΤΗΝ AUTOLISP, Εκδότης Μ. Γκιούρδας
[2] ΠΑΝΑΓΙΩΤΗΣ Γ.ΧΑΡΩΝΗΣ ,ΜΗΧΑΝΟΛΟΓΙΚΕΣ ΕΓΚΑΤΑΣΤΑΣΕΙΣ ΚΤΙΡΙΩΝ ΓΙΑ ΤΟΥΣ ΜΗΧΑΝΟΛΟΓΟΥΣ ΜΗΧΑΝΙΚΟΥΣ, Διπλ. Μηχανολόγος-Ηλεκτρολόγος Ε.Μ.Π., καθηγητής Τ.Ε.Ι. Πειραιά.