Εφαρμογές Πληροφορικής στην Τοπογραφία

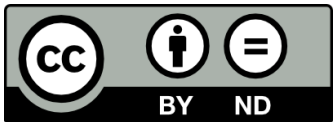
12η Ενότητα - Προγραμματισμός στην AutoLISP

Τσιούκας Βασίλειος, Αναπληρωτής Καθηγητής
Τμήμα Αγρονόμων Τοπογράφων Μηχανικών

Θεσσαλονίκη, Φεβρουάριος 2014

Άδειες Χρήσης

Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons. Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.



Χρηματοδότηση

Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα. Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.

Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Περιεχόμενα

Πίνακας Εικόνων

1. Σκοποί ενότητας

Η παρούσα ενότητα εισάγει το χρήστη του AutoCAD στην γλώσσα AutoLISP. Ο χρήστης θα πρέπει να διαθέτει βασικές γνώσεις προγραμματισμού. Η εκμάθηση γίνεται με την βήμα-βήμα δημιουργία ενός προγράμματος που εισάγει δεδομένων μετρήσεων ή υπολογισμένων συντεταγμένων σημείων τοπογραφικής αποτύπωσης στο περιβάλλον του AutoCAD.

2. Περιεχόμενα ενότητας

Προγραμματισμός στην AutoLISP

3. Εισαγωγή

Σκοπός του παρόντος εγχειριδίου είναι η γνωριμία με τη γλώσσα προγραμματισμού AutoLISP που χρησιμοποιείται από τους έμπειρους χειριστές του λογισμικού AutoCAD για την υλοποίηση σύνθετων εντολών που δεν βρίσκονται ήδη υλοποιημένες στα μενού των εντολών και εργαλειοθηκών του AutoCAD. Η εκμάθηση των εντολών γίνεται μέσα από τις απαιτήσεις για τη σύνταξη ενός μικρού προγράμματος σε εντολές πηγαίου κώδικα σε LISP που είναι ιδιαίτερα χρήσιμο για τους Αγρονόμους Τοπογράφους Μηχανικούς και αφορά την εισαγωγή μετρήσεων από ένα αρχείο κειμένου στο γραφικό περιβάλλον στο AutoCAD.

4. Η γλώσσα Lisp στο AutoCAD

Το λογισμικό AutoCAD συνοδεύεται από μία εξαιρετικά υψηλού επιπέδου γλώσσα προγραμματισμού την AutoLISP που είναι μια από τις διαλέκτους της LISP.
Η LISP έχει δημιουργηθεί το 1958 και βασίστηκε στο λογισμό λάμδα του Alonzo Church ενώ έχει χρησιμοποιηθεί στο παρελθόν για την ανάπτυξη εφαρμογών τεχνητής νοημοσύνης (http://el.wikipedia.org/wiki/Lisp).
Η διάλεκτος της, που χρησιμοποιείται στο περιβάλλον του AutoCAD δε διαφέρει σημαντικά στη σύνταξη και τη γραμματική από την κοινή (common) LISP την πλέον διαδεδομένη έκδοση που χρησιμοποιείται στα περισσότερα λειτουργικά συστήματα. Η ονομασία της (LISt Processing) προέρχεται από τη μορφή των εντολών που δημιουργούνται σε προγράμματα πηγαίου κώδικα στη LISP, που είναι λίστες με πρώτο μέλος πάντα μια συνάρτηση βιβλιοθήκης ή μια συνάρτηση που έχει δημιουργήσει ο χρήστης. Επίσης οι βασικότερες δομές δεδομένων που μπορεί να επεξεργάζονται από το περιβάλλον μεταφραστή (interpreter) με χρήση των συναρτήσεων της LISP, είναι και αυτές, λίστες.
Για την πληρέστερη κατανόηση της μορφής και του τρόπο εκτέλεσης των εντολών δίνεται ένα παράδειγμα μιας απλής εντολής εκτύπωσης κειμένου στη AutoLISP.

```
(princ "Hello World")
```

Η συγκεκριμένη εντολή μπορεί να εκτελεστεί απ' ευθείας στην προτροπή εκτέλεσης εντολών (Command) του AutoCAD και τυπώνει το κείμενο που σημειώνεται εντός των εισαγωγικών στο παράθυρο.
Η εκτέλεση της παραπάνω εντολής οδηγεί σε ένα αποτέλεσμα λιγάκι διαφορετικό από αυτό περιμέναμε καθώς εμφανίζεται κάτω από την εκτέλεση της εντολής δύο φορές (μία φορά εκτός εισαγωγικών και μία εντός) το κείμενο **Hello World**. Αυτό οφείλεται στο γεγονός ότι οι εντολές της LISP είναι στην ουσία συναρτήσεις που "επιστρέφουν" πάντα ένα αποτέλεσμα και στην προκειμένη περίπτωση αυτό το αποτέλεσμα είναι το αλφαριθμητικό **"Hello World"** (εντός εισαγωγικών) ενώ νωρίτερα πριν την εμφάνιση της τιμής αυτής εκτυπώνεται και το αλφαριθμητικό **Hello World** (χωρίς τα εισαγωγικά).
Αν μια εντολή της LISP δεν εκτελεστεί σωστά όπως αναμενόταν (πχ το άνοιγμα ενός αρχείου δεδομένων αποτυγχάνει ή έχει αποτύχει η ανάγνωση μιας γραμμής κειμένου από ένα ανοιχτό αρχείο κειμένου γιατί έχουμε εξαντλήσει τις γραμμές κειμένου προς ανάγνωση και έχουμε φτάσει στο τέλος του αρχείου) η τιμή που επιστρέφει μια συνάρτηση είναι **nil** (δηλαδή κάτι αντίστοιχο με την λογική τιμή FALSE κατά την εκτέλεση των λογικών εκφράσεων σε άλλες γλώσσες προγραμματισμού).
Σημαντικό ρόλο παίζουν οι παρενθέσεις στη LISP. Κάθε αριστερή παρένθεση συνοδεύεται υποχρεωτικά και από μία δεξιά διαφορετικά ο μεταφραστής εμφανίζει σφάλμα (**πχ error: extra right paren on input**) ή αναμένει με το συμβολισμό **->** να συνεχίσουμε με μια νέα σειρά εντολών και σταθερών ή μεταβλητών παραμέτρων για να ολοκληρώσουμε τη δημιουργία ή εκτέλεση της εντολής μας.
Για τη δημιουργία προγραμμάτων από τους χρήστες, το AutoCAD παρέχει ένα πλήρες περιβάλλον μεταφραστή που εξασφαλίζει υψηλή λειτουργικότητα, εργαλεία αποσφαλμάτωσης,

παρακολούθησης μεταβλητών και άλλα βοηθήματα, Ο μεταφραστής Visual Lisp ενεργοποιείται μέσω της εντολής VLISP ή VLIDE ή μέσω του μενού των εντολών.
Ο χρήστης μπορεί να δημιουργήσει τις δικές του εντολές ή ακόμα και ολοκληρωμένα προγράμματα με μενού και πλαίσια διαλόγων.

5. Δημιουργία εντολών - συναρτήσεων

Για τη δημιουργία μιας νέας εντολής σε ένα αρχείο πηγαίου κώδικα στη LISP μπορεί να χρησιμοποιηθεί ένας απλός κειμενογράφος (NOTEPAD) ή το ολοκληρωμένο περιβάλλον ανάπτυξης (VLIDE). Τα αρχεία πηγαίου κώδικα της Visual LISP έχουν επέκταση LSP και αποτελούνται από λίστες εντολών που δημιουργούν νέες συναρτήσεις της γλώσσας προγραμματισμού.

Η εντολή δημιουργίας μιας νέας συνάρτησης της AutoLISP είναι η **defun**. Με την εντολή **defun** δίνουμε μια νέα συνάρτηση με όνομα το στοιχείο της λίστας που έπεται του ονόματος της εντολής **defun**.

Ένα παράδειγμα για την δημιουργία μιας εντολής hello_world που είδαμε προηγουμένως και εκτυπώνει στην οθόνη το κείμενο **"Hello World"** είναι το παρακάτω:

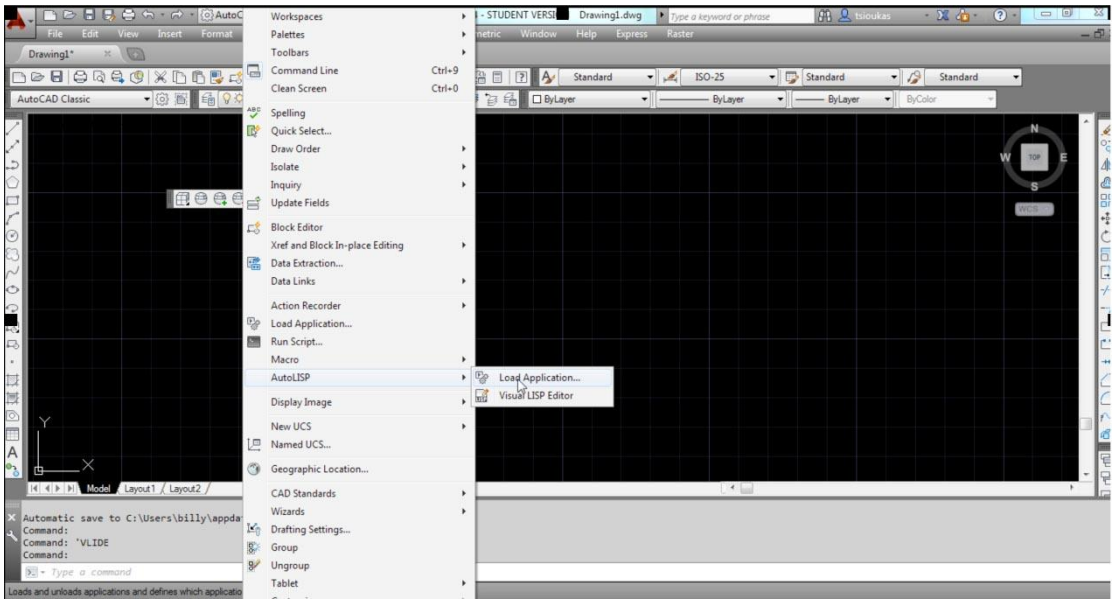
```
(defun hello_world()
        (princ "Hello World") ; εκτύπωση του κειμένου Hello World
)
```

Οι παρενθέσεις που έπονται του ονόματος της νέας εντολής ορίζουν τις παραμέτρους εισόδου για την εκτέλεση των εντολών που ομαδοποιούνται κάτω από το όνομα της νέας εντολής. Αμέσως μετά ακολουθούν λίστες εντολών που είτε ανήκουν στη βασική βιβλιοθήκη των εντολών της Visual Lisp ή είναι εντολές χρήστη που έχουν νωρίτερα δηλωθεί στο αρχείο του πηγαίου κώδικα της LISP.

Τα χρώματα στο κείμενο που δίνονται αυτόματα από το IDE (Integrated Development Environment) της Visual Lisp έχουν ιδιαίτερη σημασία και βοηθούν στον έλεγχο ορθής σύνταξης των εντολών από τον προγραμματιστή. Έτσι οι λέξεις κλειδιά ή συναρτήσεις που ανήκουν στη βασική βιβλιοθήκη των εντολών της Visual Lisp χρωματίζονται **μπλε**, οι παρενθέσεις εμφανίζονται **κόκκινες**, οι ακέραιοι αριθμοί με **πράσινο**, οι πραγματικοί αριθμοί με **λαδί** και το κείμενο **μωβ**. Οι παράμετροι μεταβλητών και οι νέες συναρτήσεις εμφανίζονται με μαύρη γραφή. Επεξηγηματικά σχόλια μπορούν να μπουν οπουδήποτε στο κείμενο με το ελληνικό ερωτηματικό να προηγείται, και εμφανίζονται με χρώμα **σκούρο μωβ** χρώμα και σκίαση.

Για την ενσωμάτωση της νέας εντολής hello_world που μπορεί να εκτελεστεί από την Command προτροπή του AutoCAD θα πρέπει να φορτωθεί το αρχείο πηγαίου κώδικα που την περιέχει, μέσω της εντολής *Tools>Load Application* ή *Tools>AutoLISP>Load Application* (**εικόνα 1**) ή μπορεί με μια εντολή AutoLISP στην command προτροπή να γίνει η εκτέλεση της φόρτωσης του αρχείου LSP και συγκεκριμένα με την εντολή (**load "C:\\AutoLISP\\helloworld.lsp"**)[1].

[1] Οι διπλοί χαρακτήρες της καθέτου είναι απαραίτητοι για να ορίσουν σωστά το μονοπάτι του αρχείου που αποθηκεύει το αρχείο πηγαίου κώδικα της AutoLISP.



Εικόνα 1. Το πρόγραμμα AutoLISP μπορεί να φορτωθεί μέσω του μενού Tools>Load Applications ή Tools>AutoLISP>Load Application

Για τη δημιουργία μιας εντολής AutoLISP που εκτελείται όπως κάθε άλλη εντολή στην προτροπή εντολής στο AutoCAD (δηλαδή χωρίς τη χρήση παρενθέσεων με τη μορφή λίστας) θα πρέπει η εντολή (πχ. hello_world) να δημιουργηθεί από την **defun** με την παρακάτω μορφή:

```
(defun c:hello_world()
        (princ "Hello World") ; εκτύπωση του κειμένου Hello World
)
```

προστίθενται δηλαδή τα σύμβολα **c:** στο όνομα της νέας συνάρτησης. Εφόσον γίνει η φόρτωση της εντολής θα μπορεί να εκτελείται με την κλήση μόνο του ονόματός της (δηλαδή: **hello_world**).

6. Δεδομένα

Τα είδη των δεδομένων που επεξεργάζονται από την LISP δε διαφέρουν από αυτά που υπάρχουν σε άλλες γλώσσες προγραμματισμού. Έτσι υπάρχουν οι βασικές μορφές δεδομένων:

- **Αριθμοί ακέραιοι**: πχ 1 2 34 7878
- **Αριθμοί πραγματικοί**: πχ 1.23 34.23 -56.7837
- **Αλφαριθμητικά (string):** κείμενο μέσα σε εισαγωγικά πχ **"S1"**
- **Μεταβλητές (άτομα) και σύμβολα.** Ως άτομα αναφέρονται συνδυασμοί λατινικών αλφαβητικών και αριθμητικών χαρακτήρων (πχ a1) που μπορεί να αποτελέσουν μια μεταβλητή (τη θέση μνήμης που αποθηκεύονται δεδομένα της LISP). Τα άτομα που δεν αποθηκεύουν μια τιμή που έχει ανατεθεί με την εντολή ανάθεσης (δείτε παρακάτω την εντολή **setq**) δεν είναι αλφαριθμητικά.
- **Μια δομή δεδομένων** που πραγματικά κάνει τη διαφορά της LISP σε σχέση με τις παραδοσιακές γλώσσες προγραμματισμού είναι η λίστα. Ως λίστα θεωρείται μια σειρά από

αριθμούς και μεταβλητές. Πχ η λίστα **("S1" 100.0 100. 100.0)** μπορεί να χρησιμοποιηθεί για τον ορισμό της θέσης μιας στάσης με κωδικό S1 και συντεταγμένες X=100.0, Y=100.0, Z=100.0.

- **Μια άλλη μορφή λίστας είναι η συνδεδεμένη λίστα.** Στις συνδεδεμένες λίστες μπορούν να αποθηκευτούν πολύπλοκα δεδομένα που αφορούν οντότητες CAD.

Η εντολή απόδοσης μιας τιμής (πχ ακέραιας) σε μία μεταβλητή είναι **setq**. Για την απόδοση της ακέραιας τιμής 12 στη μεταβλητή a θα χρησιμοποιηθεί η εντολή **(setq a 12)**. Στις μεταβλητές μπορεί να αποδοθεί κάθε είδους δομής δεδομένων όπως για παράδειγμα μια λίστα. Για την αποθήκευσης του κωδικού S1 στη μεταβλητή **s1** με συντεταγμένες **100.0 100. 100.0** θα πρέπει να χρησιμοποιηθεί η εντολή **list**. Η **list** ομαδοποιεί τις τιμές που την ακολουθούν σε μια λίστα. Άρα θα εκτελεστεί ως εξής για την απόδοση του κωδικού και της θέσης της στάσης **s1**:

```
( setq s1 ( list "S1" 100.0 100. 100.0 ) )
```

Τέλος για την προβολή στην οθόνη του περιεχομένου μιας μεταβλητής μπορεί να χρησιμοποιηθεί η εντολή **princ** αλλά σε περιβάλλον προτροπής εντολής (command) στο AutoCAD η εντολή ! (θαυμαστικό) είναι περισσότερο εύχρηστη.

7. Αριθμητικές πράξεις στην AutoLISP

Οι πράξεις στην AutoLISP έχουν την ίδια λογική με τις συναρτήσεις. Για την εκτέλεση (πχ της πρόσθεσης ανάμεσα σε δύο αριθμούς) χρειάζεται η εκτέλεση μιας εντολής λίστας με πρώτο όρισμα το σύμβολο της πράξης (πχ της πρόσθεσης) και ακολουθούν οι προσθετέοι (ως μεταβλητές ή ως καθαροί αριθμοί). Έτσι για να εκτελέσουμε την πράξη 1+2 και να εμφανιστεί στην οθόνη στην προτροπή command το αποτέλεσμα αρκεί να δοθεί η λίστα της εντολής:

```
( + 1 2 )
```

Αν επιπλέον θέλουμε να αποθηκευτεί το αποτέλεσμα της πράξης θα πρέπει να εμφωλευθεί η πράξη της πρόσθεσης στην πράξη της ανάθεσης της τιμής σε μια μεταβλητή. Έτσι για την απόδοση του αποτελέσματος της πρόσθεσης 1+2 στη μεταβλητή **a** θα πρέπει να εκτελεστεί η εντολή:

```
( setq a ( + 1 2 ) )
```

Στην ουσία όλες οι εντολές που ορίζουν μια νέα συνάρτηση χρήστη στην AutoLISP αποτελούνται από εμφωλιασμούς στη λίστα της εντολής **defun** που είδαμε προηγουμένως.

8. Δομές ελέγχου ροής του προγράμματος και επανάληψης εντολών

Όπως σε άλλες γλώσσες προγραμματισμού έτσι και στη LISP απαιτείται ο έλεγχος της ροής του προγράμματος και η ανακατεύθυνση εκτέλεσης των εντολών ανάλογα με τα αποτελέσματα που παράγονται κατά τη διάρκεια του προγράμματος.

Επίσης σε πολλές περιπτώσεις απαιτείται η επανάληψη εντολής ή εντολών εφόσον μια συνθήκη είναι αληθής ή ψευδής. Πχ. χρειάζεται να διαβάσουμε τα περιεχόμενα ενός αρχείου κειμένου που περιέχουν σε κάθε ξεχωριστεί γραμμή τον κωδικό ενός σημείου και τις συντεταγμένες του όπως έχουν μετρηθεί από ένα γεωδαιτικό σταθμό.

Η πρώτη περίπτωση καλύπτεται με τη χρήση της εντολής **if**. Η πιο απλή σύνταξη της **if** είναι:

```
( if thisIsTrue thenDoThis )
```

όπου **thisIsTrue** και **thenDoThis** είναι είτε εντολή χρήστη ή εντολή βιβλιοθήκης.
Στη θέση της εντολής **thisIsTrue** τοποθετείται μια εντολή λίστας που παράγει ως αποτέλεσμα μια λογική τιμή αλήθειας ή ψεύδους. Πχ η εντολή **(= x 1)** παράγει ως αποτέλεσμα την τιμή της αλήθειας (**t**) ή του ψεύδους (**nil**) αν προηγουμένως μέσω κάποιας εντολής **setq** έχει αποθηκευτεί στη μεταβλητή **x** η αριθμητική τιμή **1**. Αν η έκφραση είναι αληθής τότε εκτελείται η εντολή **thenDoThis** και συνεχίζεται η εκτέλεση της εντολής που έπεται. Σε περίπτωση που θέλουμε να εκτελεστεί μια άλλη εντολή όταν η συνθήκη **thisIsTrue** είναι ψευδής θα πρέπει να εκτελεστεί μια εντολή **if** με τη σύνταξη:

(if thisIsTrue thenDoThis elseDoThat)

Σε περίπτωση που απαιτείται η εκτέλεση ομάδας εντολών είτε αν η συνθήκη είναι αληθής ή αν είναι ψευδής η εκτέλεση της εντολής if θα αλλάξει ως εξής:

(if thisIsTrue (progn thenDoAllOfThis) (progn elseDoAllOfThis))

όπου **thenDoAllOfThis** και **elseDoAllOfThis** είναι ομάδες εντελών που ομαδοποιούνται με χρήση της εντολής λίστας. Γενικότερα και μια ομάδα εντολών που εκτελούνται η μία μετά την άλλη θα πρέπει να ενοποιηθούν με τη χρήση της **progn** ώστε να αντιμετωπιστούν από το μεταφραστή της LISP ως μια ξεχωριστή συνάρτηση χρησιμοποιώντας όμως χωρίς κανένα πρόβλημα τις τοπικές μεταβλητές που έχουν μέχρι τώρα αρχικοποιηθεί μέσα στη συγκεκριμένη εντολή στο πλαίσιο της οποίας εκτελείται η εντολή **if**.
Πολλές φορές μια ομάδα εντολών απαιτείται να εκτελεστεί και μάλιστα για μεγάλο αριθμό επαναλήψεων όταν εφαρμόζεται ένας συγκεκριμένος αλγόριθμος κατά τον ίδιο τρόπο σε μεγάλο αριθμό δεδομένων. Πχ. η ανάγνωση από ένα αρχείο κειμένου της θέσης (X,Y,Z) σημείων που έχουν υπολογιστεί με χρήση του 1ου θεμελιώδους προβλήματος της τοπογραφίας και η τοποθέτησή τους στο περιβάλλον του AutoCAD ως σημεία με κείμενο σχολιασμού τον κωδικό του κάθε σημείου μπορεί να γίνει μέσω της εντολής **while** ή **repeat**.
Η σύνταξη της εντολής **while** είναι:

(while expression dothis)

Στη θέση της **dothis** μπορεί να οριστεί μια ομάδα εντολών και δεν απαιτείται η χρήση της εντολής **progn**.
Με παρόμοιο τρόπο μπορεί να επαναληφθεί μια σειρά από εντολές με χρήση της εντολής **repeat**:

(repeat number dothis)

Η επανάληψη των εντολών γίνεται τόσες φορές όσες ο ακέραιος αριθμός που ακολουθεί το όνομα της εντολής **repeat**.

9. Είσοδος - Έξοδος δεδομένων

Η χρήση εντολών ανάγνωσης και εγγραφής δεδομένων από σειριακά αρχεία είναι μια πάγια πρακτική που εφαρμόζεται σε όλες τις γλώσσες προγραμματισμού. Ειδικά η AutoLISP, με τη χρήση των εντολών ανάγνωσης και εγγραφής δεδομένων από αρχεία, μπορεί με απλό και εύκολο τρόπο να μετατρέψει (αριθμητικά-διανυσματικά) δεδομένα από μορφή απλών αρχείων κειμένου σε γραφικές οντότητες του AutoCAD.
Για να μπορέσει η AutoLISP να διαβάσει ή να αποθηκεύσει δεδομένα σε σειριακά αρχεία κειμένου θα πρέπει αρχικά να ανοίξει (**open**) το αρχείο και όταν τελειώσει η μεταφορά των δεδομένων να το κλείσει (**close**). Για την ανάγνωση αλλά και για την εγγραφή δεδομένων χρησιμοποιούνται οι ίδιες

εντολές ανοίγματος και κλεισίματος. Για το άνοιγμα ενός αρχείου θα πρέπει να εκτελεστεί η εντολή **open**:

(open filename mode)

όπου **filename** και **mode** είναι κείμενο ή μεταβλητές που αποθηκεύουν αλφαριθμητικά και δηλώνουν το πλήρες μονοπάτι του αρχείου κειμένου που θα ανοιχτεί και ο τρόπος επεξεργασίας του αρχείου (ανάγνωσης **"r"**, εγγραφής **"w"** και προσθήκης νέων δεδομένων **"a"**). Π.χ για να ανοιχτεί ένα αρχείο ως ανάγνωσης, θα πρέπει να εκτελεστεί η εντολή:

(open "C:\\AutoLISP\\points.dat" "r")

Είναι σημαντικό το αποτέλεσμα της εντολής **open** να αποδοθεί σε μία μεταβλητή ώστε να είναι αποθηκευμένη η διεύθυνση-κανάλι μεταφοράς δεδομένων (stream) που χρησιμοποιείται για την πρόσβαση στα δεδομένα του αρχείου. Έτσι για την αποθήκευση της μεταβλητής ενός αρχείου (**fileVariable**) θα χρησιμοποιηθεί κατάλληλη εντολή **setq** που αναφέρεται στο αρχείο που είναι ανοιχτό:

(setq fileVariable (open "C:\\AutoLISP\\points.dat" "r"))

Η εντολή κλεισίματος αρχείου θα πρέπει να εκτελεστεί εφόσον έχει τελειώσει η επεξεργασία και μεταφορά δεδομένων. Διαφορετικά μπορεί τα περιεχόμενα του αρχείου να μην είναι προσβάσιμα από άλλες εφαρμογές ή να είναι ατελής η μεταφορά των δεδομένων και να υπάρξει απώλεια δεδομένων. Για το κλείσιμο του αρχείου θα πρέπει να έχει ήδη αποθηκευτεί σε κάποια μεταβλητή αρχείου το stream των δεδομένων του αρχείου και η εντολή **close** εκτελείται ανάλογα:

(close fileVariable)

Εφόσον έχει ανοιχτεί το αρχείο θα χρησιμοποιηθούν οι εντολές **read-line** και **write-line** για την ανάγνωση και εγγραφή κάθε μίας γραμμής κειμένου που αποθηκεύεται στο αρχείο ανάγνωσης ή εγγραφής.

Για την ανάγνωση μιας γραμμής κειμένου χρησιμοποιείται η εντολή read-line:

(setq line (read-line fileVariable))

Συγκεκριμένα η εντολή διαβάζει μια σειρά χαρακτήρων που παρεμβάλλονται μεταξύ της τρέχουσας θέσης για την ανάγνωση των δεδομένων του αρχείου και το χαρακτήρα ελέγχου (newline με κωδικό ASCII 13) που εισάγεται σε αρχεία κειμένου με το πάτημα του πλήκτρου Enter.
Η μεταβλητή που χρησιμοποιείται για την αποθήκευση των δεδομένων που διαβάζονται από μια γραμμή κειμένου στη LISP με την εντολή **read-line** είναι αλφαριθμητικό.Ο μεταφραστής της LISP δεν είναι δυνατόν να καταλάβει τις διαφορετικές (αριθμητικές) τιμές που ενδεχόμενα είναι αποθηκεμένες σε μία γραμμή κειμένου.
Έτσι αν σε ένα αρχείο κειμένου έχουμε αποθηκεύσει σε κάθε ξεχωριστή γραμμή τις συντεταγμένες και τον κωδικό σημείου που έχουν προέλθει από την εφαρμογή των θεμελιωδών προβλημάτων της Τοπογραφίας ή έχουν "κατέβει" από Γεωδαιτικό Σταθμό, η ανάγνωση των δεδομένων είναι προβληματική.

Πχ αν σε ένα αρχείο κειμένου υπάρχει η παρακάτω αλληλουχία γραμμών:

S1 100.000 100.000 100.000
S2 100.000 123.672 99.560
S3 112.561 138.900 100.34
.
.

αυτό που αντιλαμβάνεται ο Η/Υ μέσω της LISP είναι ένα αντικείμενο (αλφαριθμητικό) ενώ κανονικά θα έπρεπε να ορίζεται ένα διάνυσμα που ορίζει ένα σημείο στο χώρο με ονομασία ίδιο με των κωδικό του κάθε σημείου που έχει υπολογιστεί ή μετρηθεί. Η θέση του στο χώρο θα πρέπει να δίνεται από την τριάδα των καρτεσιανών του συντεταγμένων (X Y Z) που ακολουθούν τον κωδικό σε κάθε γραμμή κειμένου του αρχείου.
Για να μπορέσουμε να μετατρέψουμε το κείμενο που αντιλαμβάνεται ο ΗΥ με την εντολή **read-line** η LISP θα πρέπει να κάνει χρήση της εντολής **read** που διαβάζει το όρισμα της εντολής που ακολουθεί στη λίστα της εντολής και δημιουργεί μια λίστα δεδομένων.
Για να δημιουργηθεί η λίστα των δεδομένων με πρώτο στοιχείο τον κωδικό του σημείου, 2ο στοιχείο τη θέση του κατά Χ, κτλ. η εκτέλεση της εντολής **read** θα έπρεπε να είναι:

```
( setq dataline ( read "( list S1 100.000 100.000 100.000 )" ) )
```

που επιπλέον αποδίδει στη μεταβλητή **dataline** τη λίστα του πρώτου σημείου που διαβάζεται από την πρώτη γραμμή του αρχείου κειμένου.
Υπάρχει ωστόσο το πρόβλημα ότι η γραμμή κειμένου δεν μπορεί να μεταφερθεί απευθείας ως όρισμα στην εκτέλεση της εντολής **read** για να εκτελεστεί σωστά και να δημιουργηθεί η λίστα με τον κωδικό και τη θέση-διάνυσμα του κάθε σημείου. Λείπει από την αρχή της γραμμής κειμένου η αριστερή παρένθεση και η εντολή **list** και από το τέλος της η δεξιά παρένθεση. Για να συνθέσουμε σωστά το όρισμα της εντολής δημιουργίας της λίστας για κάθε γραμμή κειμένου που διαβάζεται από το αρχείο χρησιμοποιούμε επιπλέον την εντολή **strcat**. Η εντολή δέχεται ως ορίσματα αλφαριθμητικά τα συνθέτει σε μία ενότητα που την επιστρέφει για αποθήκευση με την εντολή **setq** σε μία μεταβλητή.

Έτσι η εκτέλεση των εντολών για το πέρασμα των δεδομένων της θέσης ενός σημείων στη LISP θα πρέπει να γίνει:

```
( setq line ( read-line fileVariable ) )

( setq dataline (read ( strcat "( list " line " ) " ) ) )
```

Στη μεταβλητή λοιπόν **dataline** θα αποθηκευτεί μία λίστα με πρώτο στοιχείο την εντολή δημιουργίας λίστας (**list**), δεύτερο στοιχείο τον κωδικό **S1** (άτομο) και ακολουθούν οι συντεταγμένες (πραγματικοί αριθμοί) του σημείου.

Τέλος για να διαβαστεί ολόκληρο το αρχείο των σημείων θα πρέπει να επαναληφθούν οι εντολές **read-line** και **read** μέχρι να εξαντληθούν όλες οι γραμμές κειμένου και να συναντηθεί το τέλος του αρχείου. Αυτό πραγματοποιείται με την εντολή **while** ως εξής:

```
(while ( setq line ( read-line fileVariable ) )
        ( setq dataline ( read ( strcat " ( list " line " )") ) )
)
```

10. Διαχείριση δεδομένων λίστας και οντότητες του AutoCAD

Τα δεδομένα λίστας αποτελούν βολικές δομές δεδομένων ειδικά για την αποθήκευση και διαχείριση των οντοτήτων του AutoCAD. Πριν μιλήσουμε για τις οντότητες του AutoCAD και τον τρόπο με τον οποίο γίνεται η χρήση συντεταγμένων σημείων σε λίστες για την εκτέλεση εντολών σχεδίασης γραμμών, σημείων, κύκλων και άλλων σχεδιαστικών αντικειμένων στο AutoCAD θα πρέπει να δούμε τις εντολές διαχείρισης των στοιχείων μιας λίστας.
Όπως ήδη αναφέραμε η εντολή **list** δημιουργεί τη δομή μιας λίστας. Αρκεί να εκτελεστεί ως:

```
( setq s1 ( list "S1" 100.0 100. 100.0 ) )
```

Πολλές φορές χρειάζεται να απομονωθούν μερικά στοιχεία της λίστας. Πχ για να ανακτήσουμε σε μία μεταβλητή μόνο το πρώτο στοιχείο της λίστας πρέπει να εκτελεστεί η εντολή **car**:

```
( setq head ( car ( list "S1" 100.0 100. 100.0 ) ) )➔ head="S1"
```

που αποθηκεύει την τιμή "S1" στη μεταβλητή **head**. Η συμπληρωματική εντολή της **car** είναι η **cdr** και αποδίδει το υπόλοιπο κομμάτι πλην του πρώτου:

```
( setq tail ( cdr ( list "S1" 100.0 100. 100.0 ) ) ) ➔ tail=( 100.0 100.0 100.0 )
```

Η διαφορά των δύο εντολών είναι ότι η πρώτη (**car**) επιστρέφει το απλό στοιχείο (αριθμός αλφαριθμητικό, άτομο) που είναι αποθηκευμένο στη λίστα ενώ η δεύτερη (**cdr**) επιστρέφει μια λίστα. Υπάρχει βέβαια περίπτωση στη λίστα που εφαρμόζεται ή **car** το πρώτο στοιχείο να είναι κι αυτό λίστα οπότε σε αυτή την περίπτωση η **car** επιστρέφει ως αποτέλεσμα μια λίστα.
Παραλλαγές της **car** και **cdr** είναι οι **cadr caddr cddr caar cddr** που παράγουν τα παρακάτω αποτελέσματα:

cadr: επιστρέφει το δεύτερο στοιχείο της λίστας.
cddr: επιστρέφει λίστα αφαιρώντας τα δύο πρώτα στοιχεία της.
caddr: επιστρέφει το τρίτο στοιχείο της λίστας.
caar: Αν το πρώτο στοιχείο της λίστας είναι και αυτό λίστα το αποτέλεσμα είναι το πρώτο του στοιχείο.

Μια πολύ χρήσιμη εντολή είναι η **nth** που επιστρέφει το στοιχείο της λίστας που ορίζεται ως πρώτο όρισμα στην εντολή. Πχ η εντολή:

```
( nth 1 ( list 1 2 3 ) )
```

επιστρέφει το δεύτερο στοιχείο της λίστας. Η αρίθμηση των στοιχείων της λίστας ξεκινά από το 0 να είναι το πρώτο στοιχείο της.

Υπάρχουν εντολές της LISP που χρησιμοποιούνται με σκοπό να ανακτήσουμε δεδομένα σχεδιαστικών αντικειμένων του AutoCAD.
Η εντολή **getpoint** αλληλεπιδρά με το περιβάλλον σχεδίασης του AutoCAD και εμφανίζει ένα μήνυμα που δίνεται ως όρισμα της εντολής και παράλληλα ζητείται από το χρήστη να δοθεί ένα σημείο (μια θέση) στο γραφικό περιβάλλον του AutoCAD. Το αποτέλεσμα της συνάρτησης είναι μία λίστα τριών στοιχείων που είναι οι συντεταγμένες του δοσμένου σημείου πχ: (100.000 123.672 99.560).
Ακόμα για να εκτελεστεί μια εντολή του AutoCAD πχ για την τοποθέτηση μέσω της εντολής POINT ενός σημείου η αντίστοιχη εντολή LISP είναι η **command** με ορίσματα πρώτα την εντολή του AutoCAD ως αλφαριθμητικό δηλαδή "point" ενώ ακολουθεί μια λίστα (ως διάνυσμα) των συντεταγγμένων του σημείου που ζητούμε την εισαγωγή του στο γραφικό περιβάλλον του AutoCAD.

Αν λοιπόν θέλουμε μετά την ανάγνωση των στοιχείων από το αρχείο κειμένου των σημείων να εισάγουμε ακριβώς στη θέση που ορίζεται από την τριάδα των συντεταγμένων του ένα σημείο θα πρέπει κατ' αρχήν να απομονώσουμε από τη λίστα των δεδομένων τα τελευταία του 3 στοιχεία (όλα πλην του κωδικού) και να εκτελεστεί η εντολή **command "point"**. Η αλληλουχία των εντολών θα πρέπει να είναι η εξής:

```
(while ( setq line( read-line fileVariable ) )
        ( setq dataline ( read ( strcat "( list " line ")" ) ) )
        ( setq point ( cddr dataline ) )
        ( command "point" point )
)
```

Στη μεταβλητή **point** αποθηκεύονται με τη βοήθεια της εντολής **cddr** ως λίστα, οι συντεταγμένες του κάθε σημείου που διαβάζεται από το αρχείο των δεδομένων και γίνεται η εισαγωγή του σημείου με χρήση της εντολής **command "point"**.
Η χρήση των λιστών με τουλάχιστον δύο στοιχεία πραγματικούς ή ακέραιους αριθμούς μπορεί να γίνει όχι μόνο για την τοποθέτηση σχεδιαστικών αντικειμένων σημείων (POINT) στο AutoCAD αλλά και σε όλες τις εντολές που πρέπει να οριστεί ένα σημείο είτε ως αρχή της σχεδίασης (πχ γραμμής) ή τοποθέτησης άλλου σχεδιαστικού αντικειμένου (πχ κειμένου) ή ακόμα και για τον ορισμό μιας θέσης που θα χρησιμεύσει σε μια εντολή μεταβολής σχεδιαστικού αντικειμένου (εν μέσω μιας εντολής copy, move, κτλ.).

Έτσι για την εισαγωγή κειμένου ακριβώς στη θέση που ορίζεται το σημείο μιας μέτρησης που διαβάζεται από ένα αρχείο κειμένου θα χρησιμοποιηθεί η εντολή εντολής **command "text"** ως εξής:

```
command "text" point 100 text_string [2]
```

Η μεταβλητή **text_string** πρέπει να αποθηκεύσει το κείμενο (αλφαριθμητικό) του κωδικού του σημείου που αποθηκεύεται ως πρώτο στοιχείο της λίστας **dataline**. Το στοιχείο αυτό αν είναι ακέραιος αριθμός ή ένα άτομο δεν είναι της κατάλληλης μορφής (αλφαριθμητικό) και θα πρέπει να χρησιμοποιηθεί κατάλληλη εντολή για την μετατροπή της. Η εντολή αυτή είναι η **vl-prin1-to-string**.
Οι εντολές που ολοκληρώνουν την ομάδα εντολών για την εισαγωγή των σημείων από αρχείο κειμένου σε αρχείο σχεδίασης του AutoCAD και τοποθετούν και κατάλληλο κείμενο δίπλα στο εισαγόμενο σημείο είναι:

```
( setq fileVariable ( open "C:\\AutoLISP\\points.dat" "r" ) )
( while ( setq line ( read-line fileVariable ) )
        ( setq dataline ( read ( strcat "( list " line " ) " ) ) )
        ( setq point ( cddr dataline ) )
        ( command "point" point )
        ( setq text_string ( vl-prin1-to-string ( card dataline ) ) )
        ( command "text" point 100 text_string )
)
( close fileVariable )
```

Τέλος όλες οι εντολές θα πρέπει να οριστούν ως συνάρτηση (**raport**) που θα μπορεί να εκτελείται από την προτροπή εντολών του AutoCAD. Θα δημιουργηθεί επομένως ένα αρχείο LSP και θα εισαχθεί με χρήση της εντολής **defun** ο ορισμός της νέας εντολής χρήστης.

Για να είναι περισσότερο ευέλικτο το πρόγραμμά μας θα ορίσουμε ως μια μεταβλητή εισόδου το αλφαριθμητικό που αποθηκεύει το όνομα του αρχείου κειμένου των σημείων-μετρήσεων που θα πρέπει να δίνεται με την κλήση της εντολής **raport** στην προτροπή εντολών στο AutoCAD.

Η τελική μορφή του προγράμματος που θα πρέπει να φορτωθεί στο περιβάλλον του IDE θα είναι:

```
( defun raport ( FileName )
        ( setq fileVariable ( open FileName "r" ) )
        ( while ( setq line( read-line fileVariable ) )
                ( setq dataline ( read ( strcat "( list " line ")" ) ) )
                ( setq point ( cddr dataline ) )
                ( command "point" point )
                ( setq text_string ( vl-prin1-to-string ( cadr dataline ) ) )
                ( command "text" point 100 text_string )
```

[2] 100 είναι η γωνία προσανατολισμού για την τοποθέτηση του κειμένου που είναι 100 βαθμοί

```
        )
        ( close fileVariable )
)
```

και η εκτέλεση της εντολής στην προτροπή εντολών του AutoCAD θα γίνει:
(raport "C:\\AutoLISP\\points.dat")

11. Βιβλιογραφία

Jeffery Sanders , The Ultimate AutoLISP Tutorial (http://www.jefferypsanders.com/autolisp.html)

Visual Lisp Help Topics, AutoCAD 2014, Help Files