ΠΑΡΑΔΕΙΓΜΑΤΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ AUTOCAD ΣΕ AUTOLISP ΚΑΙ FORTRAN

Αθανάσιος Στάμος
Ιωάννης Τζουβαδάκης

Ε.Μ.Π.
Σχολή Πολιτικών Μηχανικών
Αθήνα 2008

Εισαγωγή

Η πρώτη γενιά των προγραμμάτων CAD, στην οποία ανήκει και το AutoCad, αναλώθηκε στην υποκατάσταση όλων των φυσικών εργαλείων του σχεδιαστή με ηλεκτρονικά αντίστοιχα. Έτσι το σχεδιαστήριο έγινε η επιφάνεια εργασίας στην οθόνη του υπολογιστή, ο παράλληλος αντικαταστάθηκε με τη δέσμευση σχεδίασης οριζόντιων και κατακόρυφων γραμμών, η σχεδίαση με συντεταγμένες καρτεσιανές και πολικές, απόλυτες και σχετικές αντικατέστησε το κλιμακόμετρο και τα τρίγωνα, κλπ. Το πιο σημαντικό πλεονέκτημα του CAD είναι η επεξεργασία/διόρθωση του σχεδίου που γίνεται εύκολα, γρήγορα και χωρίς ατέλειες.

Παρόλα τα πλεονεκτήματα, παραμένουν πολλές επαναληπτικές και χρονοβόρες διαδικασίες στη σχεδίαση με CAD. Για παράδειγμα μία σκάλα μπορεί να σχεδιαστεί με τις βασικές εντολές του CAD σχεδιάζοντας μία μία την κάθε γραμμή από την οποία αποτελείται. Και ενώ αυτό είναι καλύτερο από ότι αν γινόταν στο χαρτί με ραπιτογράφο, είναι εξίσου χρονοβόρο και κουραστικό. Στα, αναπόφευκτα εξειδικευμένα, CAD δεύτερης γενιάς η σχεδίαση της σκάλας γίνεται ορίζοντας το αρχικό σημείο, το τελικό σημείο και τον τύπο της, και τις κουραστικές και επαναλαμβανόμενες διαδικασίες αναλαμβάνει ο υπολογιστής.

Το AutoCad, παρόλο που είναι CAD γενικού σκοπού, έχει τη δυνατότητα να αυτοματοποιήσει επαναλαμβανόμενες διαδικασίες μέσω της δυνατότητα προγραμματισμού του. Το AutoCad σήμερα διαθέτει αρκετές γλώσσες προγραμματισμού όπως τις AutoLisp, Visual Basic, C/ C++, Python. Η πρώτη γλώσσα που διέθεσε το AutoCad στους χρήστες ήταν η AutoLisp, με αποτέλεσμα σήμερα να υπάρχουν αμέτρητα προγράμματα (ή scripts) σε αυτή τη γλώσσα. Η AutoLisp είναι μία έκδοση της γλώσσα γενικού σκοπού Lisp (LISt Processing) ειδικά προσαρμοσμένη στο AutoCad. Η Lisp αποτελεί γλώσσα “υψηλού επιπέδου” (high level), πράγμα που σημαίνει ότι δίνει πολλά έτοιμα εργαλεία στο χρήστη που σε άλλες γλώσσες προγραμματισμού (πχ C) θα έπρεπε να φτιαχτούν από την αρχή. Το αποτέλεσμα είναι ότι για μια συγκεκριμένη εργασία χρειάζεται πολύ λιγότερος κώδικας που σημαίνει ταχύτερη ανάπτυξη, λιγότερα σφάλματα, πιο ευανάγνωστος κώδικας και γενικότερα πολύ μικρότερη απαίτηση σε συντήρηση (maintenance).

Η χρήση της AutoLisp, ή κάποιας άλλης γλώσσας προσαρμοσμένης στο AutoCad, για την αυτοματοποίηση διαδικασιών, ενέχει ένα μειονέκτημα το οποίο δεν είναι προφανές. Ο χρήστης εγκλωβίζεται στο συγκεκριμένο πρόγραμμα (AutoCad) και στο συγκεκριμένο λειτουργικό σύστημα (WINDOWS), αφού ότι προγράμματα αναπτύξει δεν πρόκειται να τρέξουν πουθενά αλλού. Έτσι όσα πιο πολλά προγράμματα αναπτύξει τόσο περισσότερο περιορίζεται στο συγκεκριμένο πρόγραμμα και λειτουργικό σύστημα.

Η λύση είναι η χρήση των αρχείων dxf. Η μορφή των αρχείων dxf αρχικά καθορίστηκε για το AutoCad, αλλά έχει γίνει εκ των πραγμάτων (de facto) παγκόσμιο μέσο για την ανταλλαγή σχεδίων μεταξύ διάφορων προγραμμάτων. Όλα τα σύγχρονα σχεδιαστικά (CAD) προγράμματα

μπορούν να εισάγουν και να εξάγουν dxf. Και επειδή η μορφή του dxf είναι απλό κείμενο με απλές λέξεις ως εντολές, όλα τα μη σχεδιαστικά προγράμματα μπορούν εύκολα να γράψουν αρχεία dxf. Η δημιουργία ενός σχεδίου με μορφή .dxf μπορεί να γίνει με πρόγραμμα γραμμένο σε οποιαδήποτε γλώσσα προγραμματισμού (πχ Fortran) και σε οποιοδήποτε λειτουργικό σύστημα (πχ Linux), είτε με απλέςα εντολές εκτύπωσης, είτε χρησιμοποιώντας έτοιμες βιβλιοθήκες. Επιπλέον η επεξεργασία του αρχείου dxf μπορεί να γίνει από οποιοδήποτε πρόγραμμα CAD (πχ AllPlan).

Μειονέκτημα των αρχείων dxf είναι ότι η χρήση τους δεν είναι διαδραστική (interactive), δηλαδή δεν γίνεται τη στιγμή που ο χρήστης αλληλεπιδρά με το CAD, αλλά τα αρχεία dxf δημιουργούνται εκ των προτέρων και στη συνέχεια εισάγονται στο CAD. Αρκετές φορές το μειονέκτημα αυτό είναι αμελητέο, αλλά όχι πάντα.

Παρακάτω δίνονται δύο παραδείγματα προγραμματισμού με τη χρήση των αρχείων dxf. Η γλώσσα προγραμματισμού είναι η Fortran, γλώσσα η οποία διδάσκεται στους φοιτητές της Σχολής Πολιτικών Μηχανικών του ΕΜΠ, μετά από πρόταση του ΕΠΕΑΕΚ.

Επίσης δίνονται και δύο παραδείγματα σε AutoLisp, για την αυτοματοποίηση διαδικασιών που πρέπει από τη φύση τους να είναι διαδραστικές (interactive).

Παράδειγμα 1

Να γίνει σχέδιο σύνθετης διατομής δοκού που αποτελείται από επιμέρους ορθογωνικές διατομές. Κάθε επιμέρους διατομή ορίζεται από τις συντεταγμένες $x_{1,}y_1$ (cm) του κάτω αριστερού της σημείου και τις διαστάσεις της $b\times h$ (cm) . Στο σχέδιο να φαίνονται και οι κεντροβαρικοί άξονες. Το αρχείο δεδομένων έχει κατάληξη ".dia" και περιέχει:
1η σειρά: Συντεταγμένες κέντρου βάρους x_c, y_c της σύνθετης διατομής με format 2f15.3
κάθε μία από τις επόμενες σειρές: συντεταγμένες και διαστάσεις επί μέρους διατομής $x_{\backslash,}y_{\backslash,}b, h$ με format 4f15.3

1.1 Επιλογή μορφής σχεδίασης

Εφόσον η άσκηση δεν εξειδικεύει για τον τρόπο με τον οποίο θα γίνει η σχεδίαση, είναι ελεύθερος ο προγραμματιστής να διαλέξει ανάμεσα από τις πολλές έτοιμες "βιβλιοθήκες" (libraries), δηλαδή συλλογές από έτοιμα υποπρογράμματα (subroutines, functions), που κάνουν σχεδίαση με κάποια έννοια (πχ σε οθόνη, εκτυπωτή, σχεδιαστή (plotter), αρχείο κειμένου κλπ). Τα κριτήρια επιλογής είναι:
α. Λειτουργικότητα (functionality). Αν η βιβλιοθήκη δεν μπορεί να κάνει αυτό που ζητείται, είναι ακατάλληλη.
β. Σταθερότητα (robustness). Η βιβλιοθήκη δεν πρέπει να περιέχει λάθη (bugs) που να οδηγούν σε κατάρρευση (crash), πρέπει να κάνει αυτό που περιγράφεται το εγχειρίδιο οδηγιών, και αν ανιχνεύσει λάθος στα δεδομένα πρέπει να τυπώνει τα κατάλληλα μηνύματα λάθους. Ειδικά δεν πρέπει άλλοτε να λειτουργεί και άλλοτε να καταρρέει μυστηριωδώς χωρίς μηνύματα λάθους.
γ. Τεκμηρίωση. Πρέπει να υπάρχει εγχειρίδιο οδηγιών που να περιγράφει τι κάνει κάθε υποπρόγραμμα.
δ. Συντήρηση, επεκτασιμότητα – ελεύθερο λογισμικό (free software). Μία πραγματική βιβλιοθήκη χρειάζεται διορθώσεις, προσαρμογές, νέες δυνατότητες που μπορούν να γίνουν αν υπάρχει ενεργή υποστήριξη από τον κατασκευαστή της βιβλιοθήκης. Αν ο κώδικας της βιβλιοθήκης είναι ελεύθερο λογισμικό, ο οποιοσδήποτε μπορεί να αναλάβει ως συντηρητής. Ας σημειωθεί ότι προγράμματα ανοικτού κώδικα (open source) δεν αποτελούν απαραίτητα ελεύθερο λογισμικό, διότι αρκετές φορές ανοικτό λογισμικό σημαίνει ότι μπορεί κάποιος να μελετήσει τον κώδικα αλλά όχι και να τον αλλάξει.
ε. Κόστος. Όσο μικρότερο είναι το κόστος της βιβλιοθήκης τόσο καλύτερα. Ιδανικό είναι το μηδενικό κόστος, είτε σε μορφή χρημάτων, είτε σε μορφή χρόνου μάθησης, είτε σε δυσχρηστία, είτε όλα μαζί.
στ. Ανεξαρτησία από πλατφόρμες (platform independence). Αν η βιβλιοθήκη λειτουργεί μόνο σε μία πλατφόρμα (πχ Windows, HP-UX κλπ.), ή ακόμα χειρότερα αν λειτουργεί μόνο με ένα μεταγλωττιστή σε μία πλατφόρμα, είναι απορριπτέα. Πρώτον οδηγεί σε περαιτέρω έξοδα για την αγορά πλατφόρμας (πχ Windows ή συγκεκριμένη μάρκα υπολογιστή) και για την αγορά του μεταγλωττιστή. Δεύτερον, επειδή χρήσιμα προγράμματα έχουν απρόσμενα μεγάλο χρόνο ζωής (πολλά έτη), και επειδή οι πλατφόρμες αλλάζουν με την πάροδο του χρόνου (πχ Windows 98 με Windows XP), είναι πιθανό ένα χρήσιμο πρόγραμμα να μην μπορεί να λειτουργήσει στις νέες πλατφόρμες, επειδή η βιβλιοθήκη στην οποία στηρίζεται λειτουργεί μόνο σε παλαιές πλατφόρμες. Ας σημειωθεί ότι το ελεύθερο λογισμικό είναι πολλές φορές, αλλά όχι πάντα, ανεξάρτητο από πλατφόρμες.
ζ. Γλώσσα προγραμματισμού. Αν η γλώσσα στην οποία είναι γραμμένη η βιβλιοθήκη είναι διαφορετική από την Fortran77, υπάρχει πάντα κίνδυνος από λάθη (bugs) που οφείλονται στην ανταλλαγή δεδομένων (interface) μεταξύ των δύο γλωσσών. Επίσης υπάρχουν ασυμβατότητες που οδηγούν σε απαράδεκτους συμβιβασμούς, ή σε τεχνικές που παραβιάζουν το πρότυπο της Fortran77 και οδηγούν σε παρερμηνείες, λάθη και δυσκολοσυντήρητα προγράμματα. Εξαίρεση σε αυτόν τον κανόνα είναι η Fortran2003 που έχει σχεδιαστεί για να συνεργάζεται αρμονικά με τις γλώσσες προγραμματισμού C/C++.

Για τις ανάγκες του προγράμματος θα χρησιμοποιηθεί η βιβλιοθήκη DXFDBL η οποία παράγει το σχέδιο σε μορφή αρχείου κειμένου dxf (Drawing Interchange Format). Η μορφή των αρχείων dxf αρχικά καθορίστηκε για το το γνωστό σχεδιαστικό (CAD) λογισμικό AutoCad, αλλά έχει γίνει εκ των πραγμάτων (de facto) παγκόσμιο μέσο για την ανταλλαγή σχεδίων μεταξύ διάφορων προγραμμάτων. Όλα τα σύγχρονα σχεδιαστικά (CAD) προγράμματα μπορούν να εισάγουν και να εξάγουν dxf. Και επειδή η μορφή του dxf είναι απλό κείμενο με απλές λέξεις ως εντολές, όλα τα μη σχεδιαστικά προγράμματα μπορούν εύκολα να γράψουν αρχεία dxf.

Η βιβλιοθήκη DXFDBL κάνει ακόμα πιο εύκολη και απλή την παραγωγή dxf για προγράμματα Fortran77. Η βασική ιδέα της βιβλιοθήκης είναι η ιδεατή πένα. Η πένα μπορεί να είναι "κάτω", δηλαδή να γράφει όταν μετακινείται, ή πάνω, δηλαδή να μην γράφει όταν μετακινείται. Το σχέδιο γράφεται εξορισμού στο αρχείο "data001.dxf". Μερικά βασικά υποπρογράμματα της βιβλιοθήκης είναι:

α. `call plot (x, y, ipen)`
Μετακινεί την πένα από την τρέχουσα θέση στο σημείο με συντεταγμένες x, y. Αν ipen=2 η πένα είναι κάτω και γράφει. Αν ipen=3 η πένα είναι πάνω και δεν γράφει.

β. `call plots (1)`
Ανοίγει το αρχείο dxf και αρχικοποιεί την βιβλιοθήκη. Πρέπει να κληθεί πριν από οποιοδήποτε άλλο υποπρόγραμμα της βιβλιοθήκης.

γ. `call plot (0.0d0, 0.0d0, 999)`
Εκτελεί τελευταίες ενέργειες και κλείνει το αρχείο dxf. Πρέπει να είναι η τελευταία κλήση υποπρογράμματος της βιβλιοθήκης.

Η βιβλιοθήκη DXFDBL είναι εξολοκλήρου γραμμένη σε Fortran77, και όπως φαίνεται, χρησιμοποιεί μεταβλητές διπλής ακριβείας. Είναι πλήρως ανεξάρτητη από πλατφόρμες και αποτελεί ελεύθερο λογισμικό.

1.2 Ανάλυση

Κάθε επιμέρους διατομή είναι ένα ορθογώνιο με συντεταγμένες κορυφών:
$(x_{1,i}, y_{1,i}), (x_{1,i}+b_i, y_{1,i}), (x_{1,i}+b_i, y_{1,i}+h_i), (x_{1,i}, y_{1,i}+h_i)$
Ο κεντροβαρικός άξονας x είναι μία οριζόντια γραμμή που ξεκινά από το κέντρο βάρους και φθάνει στη μέγιστη τετμημένη της σύνθετης διατομής $max\left(x_{1,i}+b_i\right)$ και λίγο παραπάνω, έστω $\frac{b_1}{4}$, για να ξεχωρίζει από τις επιμέρους διατομές. Ομοίως, ο κεντροβαρικός άξονας x είναι μία κατακόρυφη γραμμή που ξεκινά από το κέντρο βάρους και φθάνει στη μέγιστη τεταγμένη της σύνθετης διατομής $max\left(y_{1,i}+h_i\right)$ και λίγο παραπάνω, έστω $\frac{h_1}{4}$, για να ξεχωρίζει από τις επιμέρους διατομές.

1.3 Σύνθεση

Τα δεδομένα κάθε επιμέρους διατομής χρησιμοποιούνται δύο φορές, μία φορά κατά τον υπολογισμό του κέντρου βάρους και μία φορά κατά τον υπολογισμό των ροπών αδρανείας. Οι υπολογισμοί δεν μπορούν να γίνουν ταυτόχρονα διότι ο υπολογισμός των ροπών αδρανείας

εξαρτάται από το κέντρο βάρους. Έτσι τα δεδομένα κάθε επί μέρους διατομής και τα κέντρα βάρη κάθε επιμέρους διατομής θα τοποθετηθούν σε μητρώα (ή πίνακες – arrays):

Άνοιγμα αρχείων
Ανάγνωση κέντρου βάρους
Ανάγνωση δεδομένων επί μέρους διατομών
Σχεδίαση
Κλείσιμο αρχείων

Παρακάτω δίνεται καλύτερη προσέγγιση της σχεδίασης. Η ανάγνωση και η το άνοιγμα/κλείσιμο αρχείων φαίνονται απευθείας στον κώδικα του προγράμματος.

Σχεδίαση:
Αρχικοποίηση βιβλιοθήκης
Θέσε xmax=ymax=-1.0E30
Για κάθε επιμέρους διατομή:
[Υπολογισμός συντεταγμένων κορυφών
σχεδίαση ορθογωνίου
Αν x1i+bi>xmax τότε θέσε xmax=x1i+bi
Αν y1i+hi>ymax τότε θέσε ymax=y1i+hi]
Σχεδίαση γραμμής xc,yc έως xmax+b1/4,yc (άξονας x)
Σχεδίαση γραμμής xc,yc έως xc,ymax+h1/4 (άξονας y)
Κλείσιμο βιβλιοθήκης

1.4 Κώδικας

Ο κώδικας που υλοποιεί το πρόγραμμα δίνεται πιο κάτω.

```
      program ropAdr
      include 'ropadr.inc'
      call openFi
      call rdDia
      call sxed
      call closeF
      stop
      end

      subroutine sxed
c     "Σχεδίαση σύνθετης διατομής και κεντροβαρικών αξόνων."
      include 'ropadr.inc'
      integer i
      double precision xmax, ymax, xb, yb
      call plots (1)
      xmax = -1.0d30
      ymax = -1.0d30

      do 100 i=1, nDia
          xb = x1(i) + b(i)
          yb = y1(i) + h(i)
          call plot (x1(i), y1(i), 3)
          call plot (xb,    y1(i), 2)
          call plot (xb,    yb,    2)
          call plot (x1(i), yb,    2)
```

```
          call plot (x1(i), y1(i), 2)
          if (xb .gt. xmax) xmax = xb
          if (yb .gt. ymax) ymax = yb
100   continue

      call plot (xc,              yc,              3)
      call plot (xmax+b(1)/4.0d0, yc,              2)
      call plot (xc,              yc,              3)
      call plot (xc,              ymax+h(1)/4.0d0, 2)

      call plot (0.0d0, 0.0d0, 999)
      return
      end

      subroutine rdDia
c     "Ανάγνωση συντεταγμένων/διαστάσεων επιμέρους διατομών."
      include 'ropadr.inc'
      integer i, linDia, ierr
      nDia = 0
      linDia = 0
      read (iDia, 10, iostat=ierr) xc, yc
      if (ierr .lt. 0) then
          if (nDia .gt. 0) return
          print 20, linDia, 'Το αρχείο δεδομένων είναι κενό'
          stop
      else if (ierr .gt. 0) then
          print 20, linDia, 'Συντακτικό λάθος στα δεδομένα'
          stop
      end if
100   continue
          i = nDia + 1
          linDia = linDia + 1
          read (iDia, 10, iostat=ierr) x1(i), y1(i), b(i), h(i)
10        format (4f15.0)
          if (ierr .lt. 0) then
              if (nDia .gt. 0) return
              print 20, linDia, 'Το αρχείο δεδομένων είναι κενό'
              stop
          else if (ierr .gt. 0) then
              print 20, linDia, 'Συντακτικό λάθος στα δεδομένα'
20            format ('Λάθος στη γραμμή', i5, ' του αρχείου δεδομένων:',
     &        /a)
              stop
          else if (nDia .gt. MDIA) then
              print 20, linDia, 'Πάρα πολλές επιμέρους διατομές'
              stop
          else if (b(i) .le. 0.0d0 .or. b(i) .ge. bhMax .or.
     &             h(i) .le. 0.0d0 .or. h(i) .ge. bhMax) then
              print 20, linDia, 'Η διάσταση b ή h είναι εκτός ορίων'
              stop
          end if
          nDia = i
      go to 100
      end

      subroutine openFi
c     "Άνοιγμα αρχείων."
      include 'ropadr.inc'
```

```
      integer ierr

      open (iDia, file='koil.dia', status='old', iostat=ierr)
      if (ierr .ne. 0) then
          print 10, 'koil.dia'
10        format ('Το αρχείο δεδομένων ', a, ' δεν υπάρχει ',
     &    'ή δεν μπορεί να  προσπελαστεί')
          stop
      end if
      return
      end

      subroutine closeF
c     "Κλείσιμο αρχείων."
      include 'ropadr.inc'
      close (iDia)
      return
      end
```

όπου το αρχείο 'ropadr.inc' περιέχει:

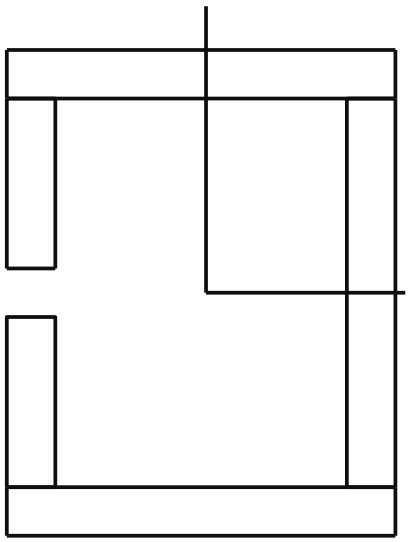
```
      implicit none
      integer iDia
      parameter (iDia=11)

      integer MDIA
      parameter (MDIA=10)
      double precision bhMax
      parameter (bhMax=10000.0d0)

      integer nDia
      double precision x1(MDIA+1), y1(MDIA+1), b(MDIA+1), h(MDIA+1),
     &xc1(MDIA), yc1(MDIA), xc, yc

      common /cadran/ x1, y1, b, h, xc1, yc1,
     &xc, yc, nDia
```

1.5 Έλεγχος ορθότητας προγράμματος - Παραδείγματα δεδομένων και αποτελεσμάτων

Εδώ θα ελεγχθεί μόνο η σχεδίαση. Δεδομένα παραδείγματος (αρχείο 'koil.dia'):

```
        81.129         83.333
        40.000         88.333         10.000         35.000
        40.000        123.333         80.000         10.000
       110.000         43.333         10.000         80.000
        40.000         33.333         80.000         10.000
        40.000         43.333         10.000         35.000
```

Αποτελέσματα παραδείγματος (αρχείο 'data001.dxf')



Παράδειγμα 2

Να αυτοματοποιηθεί η σχεδίαση ευθύγραμμης σκάλα σε κάτοψη. Το μεικτό ύψος ορόφου $h\ (m)$ και το πλάτος της σκάλας $b\ (m)$, και η κλίμακα s (δηλαδή κλίμακα $1:s$) θα εισάγεται από το πληκτρολόγιο.

2.1 Ανάλυση

Από οποιοδήποτε βιβλίο οικοδομικής, το ύψος (ρίχτι) της βαθμίδας (σκαλοπάτι) είναι 16-18cm και το πλάτος (πάτημα) είναι 30cm. Το πλήθος των υψών είναι:

$$n=\left[\frac{h}{0.165}+0.5\right]$$

όπου οι αγκύλες σημαίνουν ακέραιο μέρος. Το ύψος μιας βαθμίδας θα είναι:

$$h_i=\frac{h}{n}$$

Το πλήθος των πατημάτων είναι $n-1$ και το μήκος της σκάλας σε κάτοψη είναι:

$$L_k=(n-1)\cdot 0.30$$

Τα παρακάτω είναι λογικά όρια για τα δεδομένα:

$2.20\text{m}\le h\le 6\text{m}\ ,\ 0.70\,m\le b\le 20\text{m}\ ,\ 10\le s\le 1000$

Ως προκαθορισμένες τιμές, δηλαδή τιμές που λαμβάνονται αν ο χρήστης δεν δώσει κάποια τιμή, μπορούν να δοθούν οι παρακάτω συνηθισμένες τιμές:

$h_{\text{def}}=3\text{m}\ ,\ b_{\text{def}}=1.10\text{m}\ ,\ s_{\text{def}}=50$

Η σκάλα σε κάτοψη είναι στην ουσία ένα σύνολο από ευθύγραμμα τμήματα και κείμενα. Αυτά μπορούν να σχεδιαστούν σε αυθαίρετη θέση και διεύθυνση, για παράδειγμα στην αρχή των αξόνων με διεύθυνση νότο προς βορρά, και σε μορφή dxf, που αποτελεί εκ των πραγμάτων πρότυπο (de facto standard) για όλα τα προγράμματα CAD. Στη συνέχεια το αρχείο dxf εισάγεται στο CAD, και με το CAD η σκάλα μπορεί να μετακινηθεί/περιστραφεί ως ενιαίο αντικείμενο και να τοποθετηθεί στη σωστή της θέση.

Η σχεδίαση θα γίνει σε m, και στη συνέχεια με το CAD μπορεί τυπωθεί σε οποιαδήποτε κλίμακα. Όμως το μέγεθος των κειμένων (πχ. 0.18cm στο χαρτί) δεν πρέπει να αλλάξει με την κλίμακα. Για να πετύχουμε το απαιτούμενο μέγεθος, το μέγεθος πρέπει να οριστεί σε m ως

$$0.18\cdot\frac{s}{100}$$

έτσι ώστε με την επιβολή της κλίμακας από το CAD το τελικό μέγεθος να είναι 0.18

Για τη σχεδίαση θα χρησιμοποιηθεί η βιβλιοθήκη DXFDBL που χρησιμοποιήθηκε και στο παράδειγμα 1. Εκτός των υποπρογραμμάτων που χρησιμοποιήθηκαν στο παράδειγμα 1, θα χρησιμοποιηθούν και τα παρακάτω:

α. `call number (x, y, h, dn, theta, ndec)`

Σχεδιάζει (γράφει) τον πραγματικό αριθμό dn στο σημείο x,y με μέγεθος γραμμάτων h, και με ndec δεκαδικά στοιχεία, κατά μήκος μίας φανταστικής γραμμής που σχηματίζει γωνία theta (δεκαδικές μοίρες) με τον άξονα x. Αν ndec=-1, δεν γράφει κανένα δεκαδικό στοιχείο, ούτε υποδιαστολή. Ας σημειωθεί ότι x,y είναι οι συντεταγμένες του κάτω αριστερά σημείου ενός φανταστικού ορθογωνίου που περιγράφει όλο τα ψηφία του αριθμού.

β. `call layer (lname)`

Καθορίζει ότι όλα τα γεωμετρικά σχήματα που θα σχεδιαστούν από εδώ και πέρα, θα αποθηκεύονται στο layer lname.

2.2 Σύνθεση

Μία πρώτη προσέγγιση είναι:

```
Ανάγνωση δεδομένων
Υπολογισμών στοιχείων σκάλας
Αρχικοποίηση σχεδίασης
Σχεδίαση βαθμίδων
Σχεδίαση φοράς
Σχεδίαση αριθμών πατημάτων
Τέλος σχεδίασης
```

Η σχεδίαση των υψών είναι σχεδίαση n ευθυγράμμων τμημάτων σε απόσταση 0.30 m το ένα από το άλλο, και η σχεδίαση περιγράμματος:

```
Σχεδίαση βαθμίδων:
Θέσε layer "SKALA"
Θέσε y=-0.30
Για i=1 εώς n:
     (Θέσε y=y+0.30
      Σχεδίασε Ευθ.τμήμα από 0,y έως b,y)
Σχεδίαση περιγράμματος από 0,0 έως 0,y
Σχεδίαση περιγράμματος από b,0 έως b,y
Τέλος υποπρογράμματος
```

Η σχεδίαση της φοράς είναι μία ευθεία στη μέση της σκάλας με ένα βέλος στην κορυφή. Το βέλος θα έχει μέγεθος 0.25 cm στο χαρτί. Για να σχεδιαστεί το βέλος με μέγεθος 0.25cm σε κλίμακα 1:s το μέγεθός του σε m πρέπει να είναι $۰.۲۵\cdot\frac{s}{100}$

```
Σχεδίαση φοράς:
Θέσε layer "SKALA_FORA"
Θέσε hs=0.25*s/100
Θέσε y=(n-1)*0.30
Σχεδίασε άξονα από b/2, 0 έως b/2, y
Σχεδίασε βέλος από b/2, y έως b/2-hs/2, y-hs
Σχεδίασε βέλος από b/2, y έως b/2+hs/2, y-hs
Τέλος υποπρογράμματος
```

Τέλος η αρίθμηση των πατημάτων θα είναι κεντραρισμένη στο μέσο του κάθε πατήματος κατά τον άξονα y και λίγο πιο δεξια από το μέσο του πατήματος κατά τον άξονα x. Ας σημειωθεί ότι αν ένας αριθμός με k ψηφία σχεδιαστεί στο σημείο x_a, y_a (κάτω αριστερό σημείο του φανταστικού περιγεγραμμένου ορθογωνίου) με μέγεθος hs , τότε το κέντρο του αριθμού (ή του περιγεγραμμένου ορθογωνίου) είναι $x_a+\frac{k\cdot hs}{۲}$, $y_a+\frac{hs}{2}$

```
Σχεδίαση αρίθμησης:
Θέσε hs=0.18*s/100
```

```
Θέσε dy=-0.30
Για i=1 εώς n-1:
      (Θέσε y=y+dy
       Αν i < 10 -> θέσε k=1 αλλιώς θέσε k=2
       Θέσε xa=b/2+hs,  ya=y+0.30/2-hs/2
       Σχεδίασε αριθμό i στη θέση xa,ya με μεγεθος hs, θ=0
Τέλος υποπρογράμματος
```

Για την ανάγνωση των δεδομένων από το χρήστη χρειάζεται ένα υποπρόγραμμα που να τυπώνει προτροπή στην οθόνη (δηλαδή επεξήγηση για το τι πρέπει να δοθεί), να ελέγχει τα δεδομένα για τυχόν λάθη και να τα ξαναζητάει αν χρειαστεί, και να δίνει την προκαθορισμένη τιμή αν ο χρήστης δεν πληκτρολογήσει τίποτα.

```
Εισαγωγή δεδομένου από πληκτρολόγιο:
dmin, dmax είναι τα όρια, ddef η προκαθορισμένη τιμή

Αρχή:
Τύπωσε μήνυμα στην οθόνη
Διάβασε κείμενο dline από το χρήστη
Αν dline = "" -> επέστρεψε το ddef
Αν το dline δεν είναι αριθμός ->τύπωσε λάθος και πήγαινε στην αρχή
Θέσε d τον αριθμο στο dline
Αν dmin <= d <= dmax -> επέστρεψε το d
Τύπωσε λάθος (εκτός ορίων)
Πήγαινε στην αρχή
Τέλος υποπρογράμματος
```

2.3 Κώδικας

Παρακάτω δίνεται ο κώδικας του προγράμματος που αλλάζει σε σχέση με την προηγούμενη άσκηση

```
      program skala
      include 'skala.inc'
      call rdDed
      call sxSkal
      call sxFora
      call sxArit
      stop
      end

      subroutine sxSkal
c     "Σχεδίαση σκάλας."
      include 'skala.inc'
      integer i
      double precision y

      nyps = horof / HRIXTI + 0.5d0
      call plots (1)
      call layer ('SKALA')
```

```
      y = -BPAT
      do 100 i=1, nyps
          y = y + BPAT
          call plot (0.0d0, y, 3)
          call plot (bskal, y, 2)
100   continue

      call plot (0.0d0, 0.0d0, 3)
      call plot (0.0d0, y,     2)
      call plot (bskal, 0.0d0, 3)
      call plot (bskal, y,     2)
      return
      end

      subroutine sxFora
c     "Σχεδίαση φοράς σκάλας."
      include 'skala.inc'
      double precision y, b2, hs

      call layer ('SKALA_FORA')
      hs = HARROW * scale / 100.0d0
      y = dfloat(nyps-1) * BPAT
      b2 = bskal * 0.50d0

      call plot (b2,          0.0d0, 3)
      call plot (b2,          y+hs*0.5d0,  2)
      call plot (b2-hs*0.5d0, y,     2)
      call plot (b2+hs*0.5d0, y,     2)
      call plot (b2,          y+hs*0.5d0,  2)
      return
      end

      subroutine sxArit
c     "Σχεδίαση Αρίθμησης."
      include 'skala.inc'
      integer i
      double precision y, hs, ak, xa, ya

      hs = HNUM * scale / 100.0d0
      y = -BPAT
      do 100 i=1, nyps-1
          y = y + BPAT
          ak = 1.0d0
          if (i .ge. 10) ak = 2.0d0
          xa = bskal*0.5d0 + hs
          ya = y + BPAT*0.5d0 - hs*0.5d0
          call number (xa, ya, hs, dfloat(i), 0.0d0, -1)
100   continue

      call plot (0.0d0, 0.0d0, 999)
      return
      end


      subroutine rdDed
c     "Ανάγνωση ύψους ορόψου, πλάτους σκάλας, κλίμακας."
      include 'skala.inc'
```

```
      call inpDou('Δώστε ύψος ορόψου σε m      (enter=3) : ',
     &2.20d0, 6.0d0, 3.0d0, horof)

      call inpDou('Δώστε πλάτος σκάλας σε m (enter=1.10) : ',
     &0.70d0, 20.0d0, 1.10d0, bskal)

      call inpDou('Δώστε κλίμακα s (1:s)      (enter=50) : ',
     &10.0d0, 1000.0d0, 50.0d0, scale)
      return
      end

      subroutine inpDou (mes, dmin, dmax, ddef, dnum)
c-----"Διαβάζει από πληκτρολόγιο διπλής ακριβείας με όρια και προκαθορισμένο."
      implicit none
      double precision dmin, dmax, ddef, dnum
      character*(*) mes
      integer ierr
      character*20 dline

100   continue
          print *, mes
          read 10, dline
10        format (a)
          dnum = ddef
          if (dline .eq. ' ') return
          read (dline, 20, iostat=ierr) dnum
20        format (f20.0)
          if (ierr .ne. 0) then
              print *, 'Συντακτικό λάθος. Προσπαθείστε πάλι.'
              go to 100
          end if
          if (dnum .ge. dmin .and. dnum .le. dmax) return
          print *, 'Ο αριθμός είναι εκτός ορίων. Προσπαθείστε πάλι.'
      go to 100
      end
```

όπου το αρχείο 'skala.inc' περιέχει:

```
      implicit none
      double precision HRIXTI, BPAT, HARROW, HNUM
      parameter (HRIXTI=0.165d0, BPAT=0.30d0,
     &HARROW=0.25d0, HNUM=0.18d0)

      integer nyps
      double precision horof, bskal, scale
      common /cskala/ horof, bskal, scale, nyps
```

2.4 Έλεγχος ορθότητας προγράμματος - Παραδείγματα δεδομένων και αποτελεσμάτων

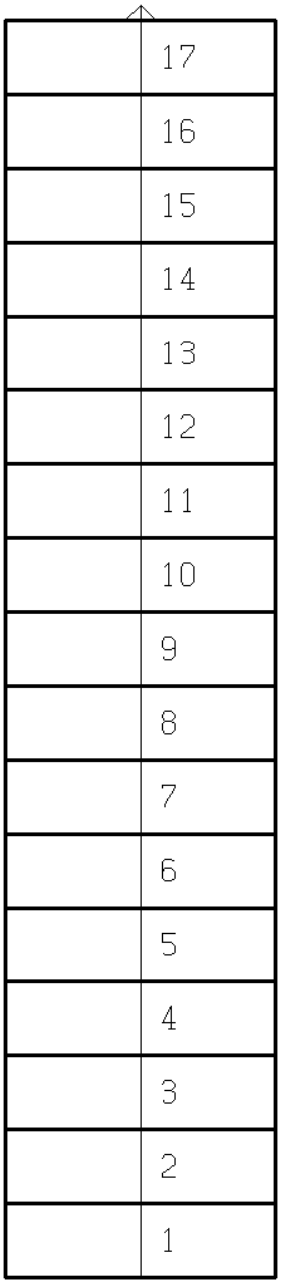
Το παρόν πρόγραμμα κάνει μία συγκεκριμένη εργασία και έχει πολλούς ελέγχους (δεδομένα εντός ορίων), η επιβεβαίωση των οποίων δίνεται ως άσκηση στον αναγνώστη. Εδώ θα δοθεί παράδειγμα για του υπολογισμούς.

Δεδομένα παραδείγματος (πληκτρολόγιο/οθόνη):

```
 Δώστε ύψος ορόψου σε m      (enter=3) :
```

```
2.80
Δώστε πλάτος σκάλας σε m (enter=1.10) :
1.20
Δώστε κλίμακα s (1:s)      (enter=50) :
50
```

Αποτελέσματα παραδείγματος (αρχείο 'data001.dxf')



Παράδειγμα 3

Να αυτοματοποιηθεί η εισαγωγή τρισδιάστατων σημείων (points) στο AutoCad. Οι συντεταγμένες x, y των σημείων δίνονται με γραφικό τρόπο (ποντίκι – mouse), ενώ το υψόμετρο z δίνεται αριθμητικά.

3.1 Επιχειρηματολογία

Για να γίνει τοπογραφικό διάγραμμα περιοχής τεχνικού έργου, χρειάζεται να μετρηθούν τοπογραφικά σημεία. Κατά τα αρχικά στάδια της μελέτης του έργου, μπορούν να χρησιμοποιηθούν μετρημένα σημεία σε έτοιμους χάρτες μικρής κλίμακας (1:5000). Οι χάρτες αυτοί σαρώνονται (scanning) και τοποθετούνται στο AutoCad, ως φωτογραφίες (raster). Στη συνέχεια χρησιμοποιείται η εντολή POINT του AutoCad και ο χρήστης τοποθετεί τον κέρσορα πάνω από το τοπογραφικό σημείο του χάρτη και κλικάρει για να δημιουργήσει ένα σημείο στο AutoCad με τις συντεταγμένες x, y του κέρσορα (και συνεπώς τις συντεταγμένες του τοπογραφικού σημείου πάνω στο χάρτη).

Δυστυχώς το υψόμετρο που είναι γραμμένο στο χάρτη δίπλα στο σημείο δεν μπορεί να δοθεί έτσι. Θα πρέπει πρώτα ο χρήστης να δώσει το υψόμετρο χρησιμοποιώντας την εντολή ELEV του AutoCad, και στη συνέχεια να χρησιμοποιήσει την εντολή POINT. Επίσης, για να αποφευχθούν παρενέργειες, μετά την εντολή POINT ο χρήστης πρέπει θέσει το υψόμετρο ίσο με μηδέν, πάλι με την εντολή ELEV. Συνοπτικά πρέπει να γίνουν τα εξής:

α. command: *ELEV <enter>*
β. Specify new default elevation: *<Ο χρήστης πληκτρολογεί υψόμετρο><enter>*
γ. Specify new default thickness: *0 <enter>*
δ. command: *POINT <enter>*
ε. Specify a point: *<Ο χρήστης κλικάρει πάνω στο τοπογραφικό σημείο>*
στ. command: *ELEV <enter>*
ζ. Specify new default elevation: *0 <enter>*
η. Specify new default thickness: *0 <enter>*

Είναι προφανές ότι η παραπάνω διαδικασία είναι επίπονη και επιρρεπής σε λάθη. Αυτό που χρειάζεται είναι να κλικάρει ο χρήστης σε τοπογραφικό σημείο και στη συνέχεια να πληκτρολογεί το υψόμετρο, χωρίς να χρειάζεται να γράφει καμμία άλλη εντολή.

3.2 Εντολές AutoLisp

Μία εντολή της AutoLisp αποτελείται από αυθαίρετο πλήθος λέξεων ή αριθμών χωρισμένων με κενά που βρίσκονται εντός παρενθέσεων. Για παράδειγμα η επόμενη εντολή τυπώνει έναν αριθμό στην οθόνη:

```
(princ 1.25)
```

Η πρώτη λέξη κάθε εντολή καθορίζει τι θα γίνει στης επόμενες λέξεις. Για παράδειγμα στην παραπάνω εντολή η πρώτη λέξη `princ` καθορίζει ότι η επόμενη λέξη θα τυπωθεί στην οθόνη. Ομοίως η επόμενη εντολή προσθέτει δύο αριθμούς:

```
(+ 2 3.1415)
```

Δύο εντολές μπορούν να συνδυαστούν έτσι ώστε το αποτέλεσμα της μίας να είναι μία λέξη για την άλλη:

```
(princ (+ 2 3.1415))
```

Ένα πρόγραμμα σε AutoLisp είναι ένα σύνολο από εντολές που περικλείονται από την εντολή `defun`. Η εντολή αυτή καθορίζει και το όνομα με το οποίο χρήστης εκτελεί το πρόγραμμα. Για

παραειγμα το επόμενο πρόγραμμα τυπώνει ένα άθροισμα, και εκτελείται πληκτρολογώντας την εντολή ektyp στο AutoCad:

```
(defun c:ektyp()
     (princ “Athroisma=”)
     (princ (+ 2 3.1415))
)
```

Μία μεταβλητή παίρνει τιμή με την εντολή setq. Επίσης η μεταβλητή πρέπει να οριστεί στην εντολή defun:

```
(defun c:ektyp(/ txt sm)
     (setq txt “Athroisma=”)
     (setq sm (+ 2 3.1415))
     (princ txt sm)
)
```

Η εντολή getpoint τυπώνει προτροπή προς το χρήστη και περιμένει να κλικάρει ο χρήστης ένα σημείο. Για παράδειγμα η εντολή:

```
(setq pt (getpoint “Select point: “)
```

κάνει τα εξής:
α. Τυπώνει στην οθόνη: Select point:
β. Περιμένει από το χρήστη να κλικάρει ένα σημείο
γ. Θέτει και τις 3 συντεταγμένες του σημείου στη μεταβλητή pt. Δηλαδή η μεταβλητή pt είναι ένας Οι εντολές getreal και getstring κάνουν την ίδια εργασία για πραγματικούς αριθμούς και κείμενο αντίστοιχα.

Οι εντολές car και cadr παίρνουν το πρώτο και δεύτερο στοιχείο μίας λίστας αντίστοιχα. Στις παρακάτω εντολές:

```
(setq x (car pt))
(setq y (cadr pt))
```

οι μεταβλητές x και y λαμβάνουν το πρώτο και δεύτερο στοιχείο της λίστας pt. Σε Fortran οι αντίστοιχες εντολές θα ήταν:

```
x = pt(1)
y = pt(2)
```

Η εντολή list δημιουργεί μία λίστα από τις επόμενες λέξεις στην ίδια γραμμή:

```
(setq pnt (list x y 108.0))
```

Η παραπάνω εντολή θέτει στη μεταβλητή pnt μία λίστα με 3 στοιχεία (ότι περιέχει η μεταβλητή x, οι περιέχει η μεταβλητή y και τον αριθμό 108.0).

Τέλος η εντολή command εκετελί μία εντολή του AutoCad. Για παράδειγμα η εντολή:

```
(command LINE (list 10.1 20.2 0.0) 5.0)
```

σχεδιάζει έναν κύκλο με συντεταγμένες κέντρου (10.1, 20.3) και ακτίνα 5.0.

3.3 Σύνθεση

Το πρόγραμμα ακολουθεί την παρακάτω πορεία υπολογισμών:

Ανάγνωση ενός σημείου από το χρήστη
Ανάγνωση υψομέτρου από το χρήστη
Δημιουργία λίστας pt με x,y απί σημείο και z από το υψόμετρο

Δημιουργία σημείου AutoCad στις συντεταγμένες pt
Τέλος σχεδίασης

3.4 Κώδικας

Παρακάτω δίνεται ο κώδικας του προγράμματος:

```
(defun c:yps (/ pt x y h)
  (setvar "CMDECHO" 0)
  (setq pt (getpoint "Select point: "))
  (setq h  (getreal  "Height      : "))
  (setq x (car pt))
  (setq y (cadr pt))
  (setq pt (list x y h))
  (command "point" pt)
  (setvar "CMDECHO" 1)
)
```

Η δεύτερη εντολή θέτει την μεταβλητή περιβάλλοντος (environmental variable) του AutoCad σε 0, έτσι ώστε το AutoCad να μην επαναλαμβάνει τα ονόματα των εντολών που εκτελεί, διότι αυτό μπορεί να δημιουργήσει παρερμηνείες. Η τελευταία εντολή ακυρώνει αυτή την ενέργεια.

3.5 Έλεγχος ορθότητας προγράμματος

Το παρόν πρόγραμμα κάνει μία συγκεκριμένη εργασία που μπορεί να ελεγχθεί μόνο μέσα από το AutoCad. Για να φορτωθεί το πρόγραμμα στο AutoCad πρέπει να γίνουν τα εξής:
`Μενού Tools -> AutoLISP -> Load..`
και στη συνέχεια επιλέγουμε το αρχείο στο οποίο έχει αποθηκευτεί το πρόγραμμα και κλικάρουμε το κουμπί `Load`.

Πιο κάτω φαίνεται η στιχομυθία με το χρήστη κατά την εκτέλεση του προγράμματος.

α. command: *yps <enter>*
β. Select point: *<Ο χρήστης κλικάρει πάνω στο τοπογραφικό σημείο>*
γ. height : *<Ο χρήστης πληκτρολογεί υψόμετρο>*
δ. command:

Όπως διαπιστώνεται, το πρόγραμμα είναι μία μεγάλη βελτίωση της διαδικασίας που περιγράφεται στην παράγραφο 3.1.

Παράδειγμα 4

Να αυτοματοποιηθεί η μετατροπή polylines σε καμπύλες γραμμές, σε παλαιότερη έκδοση του AutoCad.

4.1 Ανάλυση

Μία polyline (συνεχής τεθλασμένη γραμμή) μπορεί να γίνει καμπύλη χρησιμοποιώντας την υποεντολή `SPLINE` της εντολής `PEDIT`. Σε παλαιότερες εκδόσεις του AutoCad πρέπει να πληκτρολογηθούν τόσες `PEDIT` όσες είναι και οι polylines. Συνεπώς χρειάζεται ένα πρόγραμμα που θα εκτελεί την εντολή `PEDIT` σε πολλές polylines αυτόματα.

4.2 Σύνθεση

Το πρόγραμμα ακολουθεί την παρακάτω πορεία υπολογισμών:

```
Επιλογή συνόλου αντικειμένων από το χρήστη
Επιλογή υποεντολής PEDIT που θα εκτελεστεί σε όλες τις polylines
Εφαρμογή της PEDIT/υποεντολής μόνο στις polylines
Τέλος
```

Το τρίτο βήμα χρειάζεται καλύτερη προσέγγιση:

```
Εφαρμογή PEDIT
Έστω objs τα επιλεγμένα αντικείμενα
Αν το objs δεν είναι κενό:
     Για κάθε αντικείμενο obj από τα objs:
          Αν το αντικείμενο είναι Polyline:
               Εκτέλεση PEDIT/υποεντολή στην OBJ
Τέλος
```

4.3 Εντολές AutoLisp

Για την υλοποίηση του προγράμματος σε AutoLisp χρειάζεται η γνώση ορισμένων εντολών. Η εντολή `ssget` περιμένει από το χρήστη να επιλέξει αντικείμενα. Για παράδειγμα η επόμενη εντολή θέτει τίς ονομασίες (ή κωδικούς) των αντικειμένων που επέλεξε ο χρήστης στη λίστα `objs`:

```
(setq objs (ssget))
```

Η εντολή `ssname` επιλέγει συγκεκριμένο αντικείμενο από τα επιλεγμένα (όχι το ίδιο το αντικείμενο αλλά τον κωδικό του). Για παράδειγμα το 16ο από τα αντικείμενα είναι::

```
(setq objnam (ssname objs 15))
```

Η εντολή `sslength` επιστρέφει το πλήθος των επιλεγμένων αντικειμένων.

Η εντολή `entget` επιστρέφει το αντικείμενο από τον κωδικό του:

```
(setq obj (entget objnam))
```

Η παρακάτω εντολή αποτελεί ένα ιδίωμα της AutoLisp με το οποίο μπορεί να βρεθεί το είδος του αντικειμένου, δηλαδή αν είναι Polyline, κύκλος κλπ:

```
(setq typ (cdr (assoc 0 obj)))
```

Τέλος η εντολή `while` εκτελεί τις περιεχόμενές της εντολές επαναληπτικά, η εντολή `if` εκτελεί τις περιεχόμενές της εντολές αν η συνθήκη της είναι αληθής, και η εντολή `or` είναι αληθής αν μία είτε οι δύο περιεχόμενες συνθήκες της είναι αληθείς.

4.4 Κώδικας

Παρακάτω δίνεται ο κώδικας του προγράμματος:

```
(defun c:yk (/ objs oper i objnam obj typ)
  (setvar "CMDECHO" 0)
  (princ "Select Polylines:\n")
  (setq objs (ssget))
  (setq oper (getstring "PEDIT subcommand to pass to all polylines : "))
  (setq i 0)
  (if (/= objs nil)
     (while (< i (sslength objs))
        (setq objnam (ssname objs i))
        (setq obj (entget objnam))
        (if (or (= typ "POLYLINE") (= typ "LWPOLYLINE"))
           (command "pedit" objnam oper "")
        )  ;end if
        (setq i (+ i 1))
     )  ;endwhile
  )  ;end if
  (setvar "CMDECHO" 1)
)
```

Ας σημειωθεί ότι το πρόγραμμα ελέγχει αν το είδος του αντικειμένου είναι Polyline ή LWPolyline έτσι ώστε να μπορεί να εκτελεστεί και σε νεότερες εκδόσεις του AutoCad.

4.5 Έλεγχος ορθότητας προγράμματος

Το παρόν πρόγραμμα κάνει μία συγκεκριμένη εργασία που μπορεί να ελεγχθεί μόνο μέσα από το AutoCad. Πιο κάτω φαίνεται η στιχομυθία με το χρήστη κατά την εκτέλεση του προγράμματος.

α. command: *yk <enter>*
β. Select Polylines:
   Select objects: *<Ο χρήστης επιλέγει αντικείμενα> <enter>*
γ. PEDIT subcommand to pass to all polylines : *s <enter>*
δ. command: