Einführung in AutoCAD für Archäologen

Christoph Rinne

01. Juni 2021

Inhaltsverzeichnis

1 Automatisierte Prozesse und LISP

Es gibt einige Tools und Tricks (sog. *little mingnons*), die das Leben leichter machen, z.B. das Zeichen zahlreicher Punkte oder einer Linie anhand einer Punkteliste. Das lässt sich verfeinern, bleibt aber rudimentär. An irgend einem Punkt findet man dann ergänzende Programme oder Funktionen, die in LISP geschrieben sind. Die in diesem Kapitel verwendeten LISP-Dateien, AutoCAD-Blöcke, Beispieldateien und ergänzende Informationen finden Sie auf ISAAK/ArchJobCAD.

Ziel ist eine ganz kurze Einführung in LISP bzw. AutoLISP, um die von mir aus einer Notwendigkeit heraus geschriebenen *Helferlein* verständlich zu machen und Ihnen die Möglichkeit auf die Nachnutzung von Code aus dem Internet oder das Verständnis der zahlreichen Forenbeiträge zu erleichtern. Es ist mehr das Aufstoßen eines Fensters als der Einstieg in ein ganz eigenes Thema der Programmierung. Von zentraler Bedeutung bei meinem Einstieg in LISP waren die Websites von Axel Strube-Zettler (Strube-Zettler, o. J.a, o. J.b). Auch wenn aus Kapitelüberschriften wie "Brot, Eier, Käse" der Inhalt nicht unmittelbar erkennbar ist, es liest sich amüsant, erleichtert das Lernen und machte mich seinerzeit neugierig auf weitere Inhalte. Eine gute Ergänzung dazu bieten die Tutorials auf AfraLISP (Watson, o. J.).

LISP ist eine sehr alte Programmiersprache, nach Fortran die zweitälteste noch lebende Sprache (Wikipedia-de 31.05.2020). Wie bei anderen Sprachen gibt es zahlreiche Dialekte, u.a. AutoLISP und Visual LISP, die bei AutoCAD genutzt werden. LISP steht übrigens für "**Lis**t **P**rocessing" (Listen-Verarbeitung) und nicht - wie gerne verballhornt - für *lost in stupid parenthesis*. Beides unterstreicht für den Einstieg zwei Aspekte: Listen wie '(x y z) sind ein zentrales Element und Klammern bestimmen die Reihenfolge der Evaluation.

1.1 Automatisierte Prozesse

1.1.1 Mehrere Punkte

Starten Sie mit einer neuen Datei. Verändern Sie als erstes unter "Format -> Punkstil" den Punktstil von einem "unsichtbaren" Punkt zu einem gut sichtbaren Symbol. Danach zeichnen wir mit "Zeichnen -> Punkt -> Mehrere Punkte" mehrere Punkt. Anstatt diese aber mit der Maus zu klicken kopieren Sie die nachfolgenden Reihe von x,y,z-Werten als ganzes und fügend diese als Block in die Kommandozeile des aktiven Befehls ein. Die Koordinaten des letzten Punktes müssen Sie noch mit <enter> bestätigen und den aktiven Befehl mit <esc> beenden. Danach auf die Grenzen der Zeichnung zoomen `zoom g` und mit `regen` die Darstellung regenerieren, um die Symbolgröße der Punkte an die Darstellung anzupassen. Beachten Sie, <Leerzeichen> oder <enter> wird gleichermaßen als Trennzeichen behandelt.

```
4,1,6 2,8,7 7,0,2
6,6,4 9,1,6 0,3,10
```

1.1.2 3D-PolyLinie

Da die Punktkoordinaten noch in der Zwischenablage von Windows liegen, machen wir mit einer 3DP weiter, Sie können das aber analog auf PL übertragen. Also `3DP` <enter>, Zwischenablage einfügen, <enter> für den letzten Punkt und <enter> für das Beenden des Befehls. Das animiert doch gleich zu einem weiteren Versuch, diesmal ohne '3dp' vorab aufzurufen:

```
3dp 4,6,1 3,8,4 2,3,10 s

3dp 4,1,8 6,4,3 2,6,9 s
```

1.1.3 Text einfügen

Und weil das animiert teste ich noch etwas.

```
text 4,5,0 0.4 0 Erste Zeile
text 4,6,0 0.4 0 Zweite Zeile
```

Das funktioniert aber nicht, sondern bleibt nach der Winkelangabe in Erwartung einer Texteingabe hängen. Beachten Sie: Die Leerzeichen werden als Zeilenumbruch oder vielmehr als Ende des erwarteten Rückgabewertes aufgefasst! Kopieren Sie die beiden Zeilen in eine neue Textdatei (Editor) und speichern Sie dies mit der Endung *.scr als AutoCAD-Skript. Mit "Extras -> Skript ausführen" oder `script` können Sie dieses Skript ausführen und der Text erscheint an der jeweiligen Position, mit der angegebenen Höhe und Rotation.

1.1.4 LISP-Einzeiler

Auch ohne große LISP-Kenntnis können mit der LISP-Funktion (command) gezielt AutoCAD Befehle aufgerufen und die notwendigen Parameter übergeben werden. Mit Blick auf das vorangehende Beispiel sieht der Befehl dann so aus:

```
(command "text" "4,5,0" 0.4 0 "Erste Zeile")
(command "_text" "4,6,0" "0.4" "0" "Zweite Zeile")
```

Die Unterschiede sind eher marginal und lassen sich mit den gezeigten Varianten leicht erklären. Wie später erklärt, wird der LISP-Befehl als ganzes in eine Klammer geschrieben, das erste Element ist die Funktion und der Rest der Liste wird entsprechend der Funktion verarbeitet. Mit der Funktion (command) kann jeder Befehl aufgerufen werden, auf den dann die für diesen Befehl notwendigen Parameter folgen müssen. Das Leerzeichen trennt die einzelnen Listenelemente. Text bzw. *string* muss in Anführungszeichen stehen, bei Zahlen ist dies nicht notwendig. Der "_" vor dem Befehl verweist auf den englischen Originalbefehl unabhängig von der Installationssprache. Bei "text" ist das weniger offensichtlich bei "einfüge" bzw. "_insert" schon. Da wir bereits mit Blöcken mit Attribut gearbeitet haben noch folgendes Beispiel.

Achtung

- Für das folgende Beispiel muss ein Block "Quadr_Nr" mit genau einem Attributfeld vorliegen!
- Der Attributadialog `attdia` muss ausgeschaltet (0) sein.
- Wird im Einfügedialog für Blöcke nicht die Option "Einfügepunkt [x] Am Bildschirm bestimmen" gewählt erscheint der Attribits-Dialog in jedem Fall.

```
(command "_insert" "Quadr_Nr" "44.1,57.4,15.9" "1" "1" "0" "2")
```

Auf den Befehl zum einfügen eines Blockes folgen: Blockname, Einfügepunkt, Skalierung x, Skalierung y, Rotation und zuletzt die ggf. geforderten Attribute jeweils getrennt mit Leerzeichen. Sie erkennen unschwer, das lässt sich leicht in jeder Tabellenkalkulation erstellen (in Excel: VERKETTEN()). Lediglich die Punkt-

Koma-Problematik erschwert uns in Excel ein wenig das Leben, ist aber mit einer Kombination von FINDEN() und ERSETZEN() zu lösen. Nachfolgend ein Beispiel mit einem ersten Tabellenblatt (Name “Daten”) und einem weiteren Tabellenblatt mit der folgenden Syntax in einer Zelle je Zeile.

Nummer	x-Wert	y-Wert	z-Wert	Blockname	x-Faktor	y-Faktor	Rotation
1	4433863,19	5743121,48	154,978	Quadr_Nr	1	1	0
2	4433861,19	5743122,58	154,945	Quadr_Nr	1	1	0
3	4433865,19	5743120,36	154,265	Quadr_Nr	1	1	0

```
=VERKETTEN("(command ""_insert"" """&Daten!E3&""" """&
            ERSETZEN(Daten!B3;FINDEN(",";Daten!B3);1;".")&", "&
            ERSETZEN(Daten!C3;FINDEN(",";Daten!C3);1;".")&", "&
            ERSETZEN(Daten!D3;FINDEN(",";Daten!D3);1;".")&""" """&
            Daten!F3&""" """&Daten!G3&""" """&Daten!H3&""" """&Daten!A3&""")")
```

In der vorangenden Syntax ist folgendes zu beachten:

- " (Gänsefüßchen) leiten einen Text ein und beendet diesen, so dass Excel diesen Text nicht weiter evaluiert. Kommt in diesem Text ein " vor muss es mit einen " auskommentiert werden, also "". Steht "" am Ende des Textes werden es drei """. Sieht verwirrend aus und ist es auch beim Schreiben.
- Die Funktion ‘Ersetzen’ ersetzt nicht ein Zeichenkette, z.B. ‘,’, sondern eine Position und Zeichenanzahl innerhalb einer Zeichenkette. Diese Position muss mit der Funktion ‘Finden’ bestimmt werden. In Libre Office Calc können Sie das Punkt-Komma-Problem durch die Formatzuweisung “1.1” nur in der Darstellung umgehen, durch die Verkettung wird auch hier ein Komma gesetzt.

1.2 LISP

1.2.1 Grudlegendes

Die nachfolgenden Übungen orientieren sich an dem “Kochbuch AutoLISP” (Strube-Zettler und Schönwald, 2007), dieses beruht auf den überarbeiteten Tutorials von Axel Strube-Zettler, die frei im Netz zur Verfügung stehen (Strube-Zettler, o. J.b, o. J.a). Die Übungsaufgaben mit Lösungen und den Code zu diesem Buch finden Sie auf der Website des Hansa Verlag. Alle LISP-Funktionen finden Sie in der Online-Hilfe von AutoCAD. Und nochmals, Ziel dieser wenigen Zeilen ist es, ein Fenster aufzustoßen, so dass ein Einstieg für Sie möglich wird. Funktionierenden, guten und schönen Code zu schreiben ist 1. Dreierlei und 2. lernt es sich nicht mal eben.

Mit “Extras -> AutoLISP -> Visual LISP Editor” starten Sie den bei AutoCAD integrierten Editor. Schreiben Sie die nachfolgenden Anweisungen in die Konsole des Editors.

1.2.2 Klammern und Präfixnotation

```
(+ 1 2 3)
```

Die Klammern verhindern das Ausführen nach dem Leerzeichen (s.o.), das erste Element in der Klammer ist die Funktion, hier Addition, und evaluiert alle nachfolgenden Elemente. Die Präfixnotation spart Schreibarbeit zu der uns geläufigen Form: 1 + 2 + 3.

```
(/ (+ 3 4) (+ 1 1))
```

Analog zum Vorangehenden erkennen Sie hier zwei Additionen mit den Summen 7 und 2 auf die dann eine Division angewendet wird. Die Evaluation von Klammern erfolgt von links nach rechts und von innen nach außen. Also erst links vor rechts, und dann, wenn geschachtelt, von innen nach außen. Nur das Ergebnis “3” stört. Wiederholen Sie den Befehl und ergänzen aber bei irgend einer Zahl “.0” oder (float <zahl>). Hierdurch dimensionieren Sie den Wertebereich der gesamten nachfolgenden Rechnung zu Gleitkommazahl.
Warum funktioniert das aber nicht bei: `(* (/ 5 3) (* 4.0 2))`

Es gibt viele weitere Funktionen: (sqrt 4), (abs -2), (sin 90). Letztere liefert nicht 1, für mich ebenfalls ein unerwartetes Ergebnis. AutoCAD arbeitet mit RAD, wir müssen die Grad also ableiten aus (/ pi 180): (sin (* 90 (/ pi 180))).

1.2.3 Variablen Werte zuweisen

Die Funktion, die Variablen einen Wert zuweist ist *setq*, das steht für *set quoted* und verhindert die Evaluation des unmittelbar folgenden Elementes, da dies der Name der Variable wird. Und natürlich geht das auch anders.

```
(setq var1 12)
(set (quote var3) 4)
(set 'var2 5)
```

Bei der Namensgebung gibt es zu meidende Sonderzeichen mit speziellen Bedeutungen (“.”, “;”, “(”, “)”, "‘“,”"") und Namen von vorhandenen, vor allem wichtigen Funktionen sollten ebenfalls gemieden werden. Die Namen sind nicht *case sensitive*: VAR1 ist identisch mit var1. Zudem sind gute Variablennamen verständlich, z.B. ’layer_name’.

1.2.4 Listen

Im Grunde ist alles in () eine Liste und () eine leere Liste.

```
(setq var1 ())
(setq farben '("rot" "grün" "blau"))
(setq zahlen '(1 2 3 4 5))
(setq linie '((1 2 3) (4 5 6)))
```

(setq) weist einer Variablen einen Wert zu, danach folgt mit vorangestelltem *quote* die Liste in Klammern. Warum das *quote*? Wie im Beispiel `(set (quote var3) 4)` würde die Liste von innen nach außen und links nach rechts evaluiert. Dann müsste “rot” eine Funktion sein wie (quote) ist es aber nicht.

Speziell für Listen gibt es zahlreiche Funktionen, die Rückgabewerte sind erwartbar `(length farben)` oder auch eher nicht `(member "grün" farben)`, letzteres gibt den Rest der Liste ab dem gefundenen Element zurück. Wenn Sie unbedingt die Position brauchen könne Sie ja `(- (length farben) (length (member "grün" farben)) -1)` schreiben, das -1 da “grün” in der zweiten Liste enthalten ist: 3-2=1. Statt noch eine Klammer für (+ 1) zu setzen ist (- -1) etwas eleganter.

Listen können natürlich geschachtelt sein, so eine Linie (Startpunkt Endpunkt), wobei jede Koordinate für sich eine Liste ist (x y z). Damit zu “zwei” weiteren Funktionen: **(car)** und **(cdr)**. Das erste Element von linie erhalten wir mit `(car linie)` und mit (cdr linie) den Rest nach dem ersten Element. Das lässt sich bis zu 4 x kombinieren (caaaar, cadadr ... cddddr). Testen Sie einfach die folgenden Befehle und spielen Sie etwas mit eigenen Variationen: (caar linie), (cadr linie), (cadar linie). Achten Sie auf die Unterschied zwischen (cdr linie), dem Rest der Liste und (cadr linie), dem ersten Element vom Rest der Liste. Bekommen Sie ausschließlich den z-Wert des 2. Punktes? `(cddadr linie)` Das ist aber noch eine Liste und da (caddadr linie) nicht mehr geht bleibt aber noch `(car(cddadr linie))`. Zugestanden, das war verwirrend und wäre auch einfacher gegangen `(last(last linie))`.

1.2.5 Visual Lisp Editor

Machen wir eine Pause und betrachten mal kursorisch den Visual LISP Editor von AutoCAD. Eine etwas ausführlichere Darstellungen finden Sie in einem PDF-Dokument im Internet (Bosse, o. J.). Neben der Konsole, in der wir bisher gearbeitet haben wurde die Ablaufverfolgung automatisch geöffnet, es dient der Information. Mit “Datei -> Neue Datei” öffnen Sie ein neues Fenster zum erstellen eines Code-Skriptes. Öffnen Sie mal die Datei “LayerErstellen.lsp” oder eine andere/neue LSP-Datei. Sie können “Fenster -> Untereinander oder Nebeneinander” anordnen lassen und ergänzend die Größe verändern. In der Abbildung ist noch das “Überwachungsfenster” geöffnet. Markieren Sie dafür eine Variable, z.B. LayerListe, und wählen im Kontextmenü “Überwachung hinzufügen”.

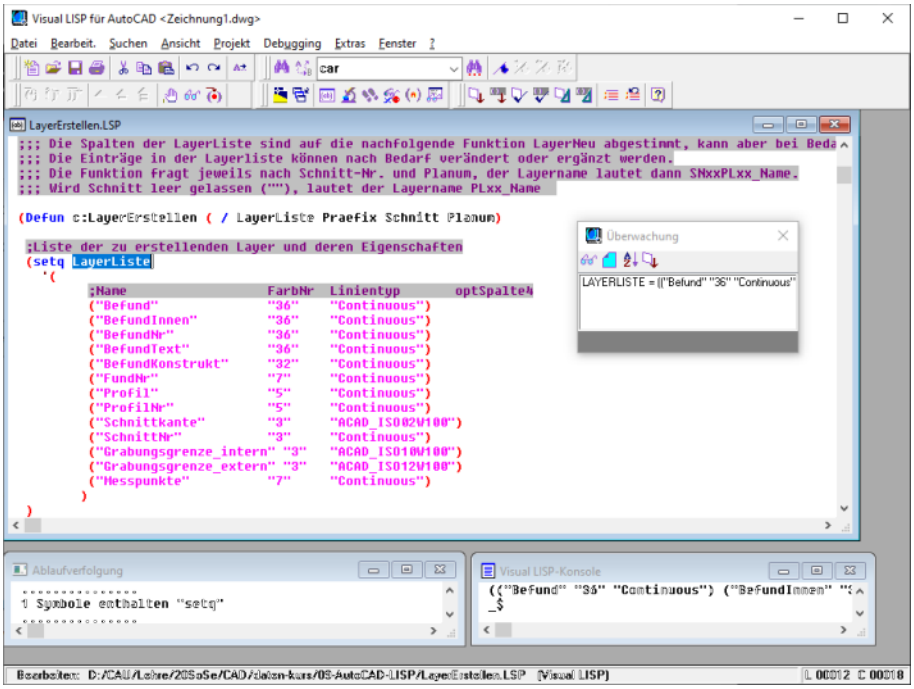

Abbildung 1: AutoCAD LISP-Editor

Wie zuvor erläutert umschließen Klammern stets einen Ausdruck. Die einzelnen Abschnitte sind hier durch Einrückungen gut sichtbar gegliedert, dies dient aber nur der Lesefreundlichkeit. Ganz allgemein, Kommentare werden mit einem “;” eingeleitet.

Code kann im Editor auf Syntaxfehler geprüft werden “Extras -> Text im Editor prüfen”, alternativ geht dies auch mit dem Icon mit dem blauen Haken, wahlweise für den gesamten Code oder nur den markierten Bereich. Wenn Sie bei der Definition der Variable LayerListe das Hochkomme entfernen, also bei (setq LayerListe ’(. . .)), diesen Abschnitt markieren und prüfen sollten Sie eine Fehlermeldung erhalten:

```
; Fehler: Fehlerhafte Funktion in Ausdruck: ("Befund" "36" "Continuous")
; Prüfung durchgeführt.
```

Direkt daneben sind ähnliche Icon mit einem roten Pfeil, hiermit können Sie den gesmmten Code oder den markierten Abschnitt laden. Die Fehlersuche wird durch schrittweises Ausführen oder durch Haltepunkte und in Kombination mit der Überwachung von Variablen deutlich erleichtert. Setzen Sie **Haltepunkte** stets vor die öffnende Klammer, gehen Sie dafür mit dem Cursor an die entsprechende Position und wählen Sie das Hand-Icon oder <F9>. Auf die gleiche Weise entfernen Sie auch den Haltepunkt am Cursor. Um alle Haltepunkte zu löschen <strg>+<shift>+<F9>. Laden Sie die Datei mit gesetzten Haltepunkten, wechseln Sie zu AutoCAD und starten Sie dort die Funktion “layererstellen”. Automatisch wird zum Editor gewechselt und die Ausführung des Codes am ersten Haltepunkt unterbrochen. Von hieraus können Sie mit den drei Icon mit “()” oder “))” 1. Schrittweise weitergehen, 2. den gesamten Schritt der Klammer nach dem Haltepunkt ausführen oder 3. bis zum Ende der gesamten Prozedur durchlaufen lassen. Parallel dazu wird für die überwachten Ausdrücke / Variablen der jeweils aktuelle Wert angezeigt.

1.2.6 Funktion Layererstellen

Das kleine Programm ist mein ältester Gehversuch in LISP und müsste eigentlich überarbeitet werden. Es ist ein gutes Beispiel für unschön vermischten Code. Die Funktion soll ein ganzes Paket an vorab definierten Layern für Befunde, Befundnummern, Profile, Funde etc. auf einen Schlag erstellen und das für einen spezifischen Schnitt und Planum (konsistente Layernamen für Gruppenfilter).

Mit einem Doppelklick auf die erste Klammer "(Defun" wird der gesamte Ausdruck dieser Klammer markiert. Es ist die Funktion insgesamt, danach folgt noch die Ausgabe einer Information an die Befehlszeile.

```
(Defun c:LayerErstellen ( / LayerListe Praefix Schnitt Planum)
; Hier steht viel anderer Kram.
)
```

Defun definiert eine Funktion, das erste Element der Liste ist der Name der Funktion, danach folgen in Klammern die übergebenen Parameter vor und nach dem "/" die innerhalb der Funktion verwendeten Parameter. Da ich die Funktion in AutoCAD aufrufen will muss ich vor den Namen ein "c:" setzen.

```
(setq LayerListe
   '(
     ;Name          FarbNr  Linientyp   optSpalte4
     ("Befund"      "36"    "Continuous")
     ("BefundInnen"     "36"    "Continuous")
     ("BefundNr"        "36"    "Continuous")
     ; ... noch mehr Layer...
    )
 )
```

Als erstes werden die zu erstellenden Layer mit ihre Eigenschaften in eine Liste geschrieben und der variable Layerliste zugewiesen. Jeder Layer der Liste ist mit einer eigenen Sub-Liste vertreten, in der Name, Farbzahl und Linientyp angegeben ist. Mit der Kommentarzeile ist eine Art Spaltenüberschrift eingeschoben und da man mit "cadddr" an das vierte Element einer Liste kommt wird auf eine optional vierte Spalte hingewiesen.

Danach folgt eine interne Funktion, die anhand der zuvor definierten Sub-Listen mit Layer-Attributen jeweils einen Layer erstellt und nach Vorgaben verändert. Innerhalb dieser Funktion wird eine Variable "Praefix" benötigt, die weiter unter erstellt wird, ich springe deshalb im Code nach unten. (Genau dieses Geschachtel und Gehüpfe ist unsauber, sollte zwar in dieser Datei aber außerhalb der Funktion c:LayerErstellen und jeweils als kleine eigene Funktion vorliegen.) Das Präfix des Layernames soll sich aus Schnittnummer (SNxx) und Planum (PLxx) zusammensetzen.

```
;Praefix für die Layernamen erstellen
 (setq Schnitt(getstring "\nSchnittnummer eingeben oder leer lassen: "))
 (if (/= Schnitt "")
   (if (= (strlen Schnitt) 1)
     (setq Schnitt (strcat "0" Schnitt))
   )
 )

 (setq Planum (getstring "\nNummer des Planums eingeben:"))
  (if (= (strlen Planum) 1)
    (setq Planum (strcat "0" Planum))
 )

 (if (/= Schnitt "")
    (setq Praefix (strcat "SN" Schnitt "PL" Planum "_"))
    (setq Praefix (strcat "PL" Planum "_"))
 )
```

Als erstes wird die Schnittnummer in eine Variable geschrieben (setq), für den Inhalt wird mit der Funktion (getstring) der Nutzer gefragt. Die Anfrage an den Nutzer muss natürlich ausformuliert werden und steht als Text in "". Das "\n" steht für *new line*, so dass die Anfrage in einer neuen und leeren Zeile landet. Es folgen zwei geschachtelte if-Funktionen, die erste Bedingung prüft, ob die Variable Schnitt nicht leer ist und überhaupt gehandelt werden muss. Die zweite Bedingung prüft, ob die Länge der angegebenen Schnittnummer "1" ist und definiert Schnitt dann neu mit einer vorangestellten "0" unter Verwendung der Funktion (strcat)

also *string concatenate*. Das ganze ist ziemlich simpel und geht von einem wohlwollenden und einsichtigen Nutzer aus.

Es folgt die Zuweisung der Nummer des Planums, die bei der Länge 1 ebenfalls um eine "0" erweitert wird.

Das dritte Element verbindet die Variablen zu Schnitt-Nr. und Planum-Nr. mit dem jeweilige Etikett ("SN", "PL") zur Variable Präfix, aber nur wenn (if) Schnitt nicht leer ist, ansonsten wird nur das Planum an Praefix übergeben. Als erkennbares Trennzeichen wird jeweils ein " *" beigefügt. Das ganze ist Schrittweise ausgeführt und leicht lesbar (einfache Sprache) also nicht zwingend elegant oder kompakt. Das Ergebnis ist das Praefix, z.B. "SN01PL01"* " und geht zurück zur internen Funktion LayerNeu.

```
;Funktion zum erstellen der Layer
 (Defun LayerNeu (LayerProperties)
  (command "_layer"
       "Neu"
       (strcat Praefix (car LayerProperties))
       "FArbe"
       (cadr LayerProperties)
       (strcat Praefix (car LayerProperties))
       "Ltyp"
       (caddr LayerProperties)
       (strcat Praefix (car LayerProperties))
       ;(cadddr LayerProperties) ;für die optSpalte4
       ;(strcat Praefix (car LayerProperties))  ;für die optSpalte4
       ;mehr als 4 Spalten geht nur mit (cadr(cadddr LayerProperties))
       ;oder (last LayerProperties)
       ""
  )
  (princ)
 )
```

Also `Defun` kennen Sie schon, danach folgt ohne und damit implizit vor dem "/" in Klammern der übergebene Parameter LayerProperties. Der ist erst leer und kann auch anders heißen, er muss aber beim Funktionsaufruf mit einer Liste gefüllt werden.
Innerhalb der Funktion bediene ich mich mit `(command)` direkt des AutoCAD-Befehles `_layer`. Ist dieser Befehl aktiv kann ich die weiteren Sub-Befehle nutzen und liefer dann die dafür notwendigen Eingaben als Elemente aus der übergebenen Liste LayerProperties. Das sind:

- Befehl Neu: Praefix und der Name als 1. Element der Liste (car LayerProperties)
- Befehl FArbe: Welche Farbe? Das 2. Element der Liste (cadr). Für welchen Layer? Das 1. Element der Liste.
- Befehl Linientyp: ...

Am Ende wird mit (princ) eine leere Zeile geschrieben. Bis jetzt ist außer der Liste mit den Eigenschaften für die Layer noch nichts Konkretes geschehen. Erst die letzte Zeile meines Programmes sorgt für Aktivität. In meiner Funktion c:LayerErstellen müsste eigentlich nur diese letzte Zeile stehen, nebst einer Fehlerbehandlung die immernoch fehlt.

```
;Ruft Funktion LayerNeu auf für jedes Element der Liste.
 (mapcar 'LayerNeu LayerListe)
```

Die Funktion (mapcar) wendet eine Funktion auf eine Liste an, sie benötigt damit mindestens diese beiden Elemente den Ausdruck der die Funktion repräsentiert (nicht evaluiert und deshalb ein ' vorweg) und die Liste auf die sie angewendet werden soll. Hier geht es also zur Sache: Funktion LayerNeu vornehmen, erste Sub-Liste als übergebener Parameter holen, das zuvor gebaute Präfix ergänzen und den Layer erstellen. Funktion LayerNeu aufrufen, nächste Subliste usw.

Ein weiteres kleines LISP-Programm für die effiziente Arbeit in AutoCAD finden Sie bei ISAAK. Die

notwendigen Grundlagen, um sich den Code von ArchCAD.lsp oder auch steigung.lsp zu erarbeiten haben Sie nun.

1.3 Ergänzende Literatur

1.3.1 Text & Buch

- Omura 1997: G. Omura, ABC's of AutoLISP: Contents at Glance - Copyright © 1997 George Omura. 1997 <http://people.fsv.cvut.cz/www/chourpav/Lisp/Contents.htm>(17.04.2012).
- Strube-Zettler/Schönwald 2007: A. Strube-Zettler/T. Schönwald, Kochbuch AutoLISP: AutoCAD programmieren mit LISP (München 2007).
- Bosse : J. Bosse, Visual-LISP Editor ab AutoCAD 2000, (o. J.) <http://www.bosse-engineering.com/downloads/090501_VisualLISP-Editor.pdf>(24.03.2012).

1.3.2 Foren, Lernen, Tools

In alphabetischer Ordnung:

- AfraLISP : AutoLISP | AfraLISP. AfraLISP() Learn AutoLISP for AutoCAD productivity <http://www.afralisp.net/autolisp/>(08.04.2012).
- AUGCE: Autodesk User Group Central Europe. <http://www.augce.de/de.shtml>(15.04.2012).
- Battin : G. Battin, AutoCAD Tips | helpful tips for everyday users. <http://autocadtips.wordpress.com/>(13.04.2012).
- CAD.DE : CAD.DE,CAD.de – Die CAD-CAM-CAE COMMUNITY, (o. J.) <https://ww3.cad.de/>(31.05.2020).
- cadstudio : cadstudio, CAD Forum - tips, tricks, utilities, discussion | AutoCAD, Inventor, Revit, Civil 3D, Autodesk. <http://www.cadforum.cz/cadforum_en/default.asp>(15.04.2012).
- CADTutor : CADTutor, AutoCAD Tutorials, Articles & Forums | CADTutor. <https://www.cadtutor.net/>(31.05.2020).
- Drese : M. Drese, CADwiesel. <http://www.cadwiesel.de/>(12.12.2017).
- Karlsson 2014: L. Karlsson, Glamsen: CadTools. 2014 <https://www.glamsen.se/CadTools.htm>(06.07.2016).
- Strube-Zettler : A. Strube-Zettler, Ausgewählte Themen zu AutoLisp. <http://www.advanced.autolisp.info/>(09.05.2017).
- Strube-Zettler : A. Strube-Zettler, Das deutschsprachige VisualLisp-Tutorial im Internet. <http://www.tutorial.autolisp.info/>(31.05.2020).

Bosse, J., o. J. Visual-LISP Editor Ab AutoCAD 2000.

Strube-Zettler, A., o. J.a. Ausgewählte Themen Zu AutoLisp [WWW Document]. URL http://www.advanced.autolisp.info/ (zugegriffen 5.9.2017).

Strube-Zettler, A., o. J.b. Das Deutschsprachige VisualLisp-Tutorial Im Internet [WWW Document]. URL http://www.tutorial.autolisp.info/ (zugegriffen 5.31.2020).

Strube-Zettler, A., Schönwald, T., 2007. Kochbuch AutoLISP: AutoCAD Programmieren Mit LISP. Carl Hanser Verlag, München.

Watson, D., o. J. AutoLISP | AfraLISP [WWW Document]. AfraLISP() Learn AutoLISP for AutoCAD productivity. URL http://www.afralisp.net/autolisp/ (zugegriffen 4.8.2012).