Programmeren in AutoLISP

Les vier

Lijsten

Construeren en Manipuleren

door

Joop F. Moelee

een gelovig volger van
“de Sacrale Kunst van Luiheid”
zijn Hoge Priester LISP en Acoliet Script

1) Inhoudsopgave

2) Verantwoording

Dit document is gebaseerd op en/of bevat delen van de artikelen geschreven door Kenny Ramage eigenaar/ beheerder van de site http://www.afraLISP.com/
Kenny heeft de schrijver dezes toestemming gegeven tot gebruik van zijn teksten.
Hiervoor heel hartelijk bedankt, Kenny.

Voor alle duidelijkheid eerst even de verklaringen van de diverse tekstsoorten.
Italic--Engelse woorden en termen die geen specifieke AutoLISP en/of AutoCAD termen zijn.
Italic en vet – AutoCAD termen, commando's en menu opdrachten.
Kleiner en vet – regels programmacode en AutoLISP termen.

3) Lijsten

Zoals reeds eerder verteld: LISP betekent *List Proces*. Deze taal is ontwikkeld om lijsten te manipuleren. En geloof het of niet maar dit is alles wat je doet als je een programma in AutoLISP schrijft.
Je creëert en manipuleert lijsten.
Wat is een lijst? Een lijst is een verzameling van een groep elementen, van welk data type dan ook, en dat is opgeslagen als een enkele variabele. Een lijst kan elk aantal gehele of decimale getallen, teksten, variabelen en zelfs andere lijsten bevatten

4) Lijsten uit elkaar halen

4.1) De car en cdr functies

Kijken we nog eens naar een van de meest voorkomende opdrachten:

(setq Startpunt (getpoint "\Kies een punt: "))

AutoLISP retourneert zoiets als dit:

(381.453 251.185 0.0)

OK, we hebben een lijst. En nu?
AutoLISP zou AutoLISP niet zijn als er niet een heleboel functies beschikbaar zou zijn om de lijsten te manipuleren.
We gaan er een aantal wat nader bekijken.

car
Het primaire commando om een lijst uit elkaar te halen is de **car** functie. Deze functie retourneert het eerste element uit een lijst, in dit geval de x-coördinaat. Bijvoorbeeld:

(setq XStartpunt (car Startpunt))

retourneerd:

(381.453)

cdr
Deze functie retourneerd het tweede element in een lijst plus de rest van de lijst. Bijvoorbeeld:

(setq YStartpunt (cdr Startpunt))

retourneerd:

(251.185 0.0)

Maar ik wil alleen het tweede element, de y- coördinaat? We zouden natuurlijk kunnen schrijven:

(setq YStartpunt (car (cdr Startpunt)))

en we zouden krijgen:

(251.185)

cadr
Maar er is een betere manier. De **cadr** functie is in principe een afkorting van een genest commando (**car** en **cdr**), dat het tweede element van een lijst retourneerd.

(setq YStartpunt (cadr Startpunt))

retourneerd:

(251.185)

caddr
En natuurlijk is er een afgekorte functie dat het derde element retourneerd:

(setq ZStartpunt (car (caddr Startpunt)))

en we krijgen dan:

(0.0)

4.2) Geneste functies

AutoLISP ondersteunt het nesten van **car** en **cdr** tot vier nivo's diep.
Voor de volledigheid een uittreksel uit de helpgids van AutoLISP. Je kunt het vinden in de index onder *list handling, returning all but first element*.

caaaar	cadaar	cdaaar	cddaar
caaadr	cadadr	cdaadr	cddadr
caaar	cadar	cdaar	cddar
caadar	caddar	cdadar	cdddar
caaddr	cadddr	cdaddr	cddddr
caadr	caddr	cdadr	cdddr
caar	cadr	cdar	cddr

Elke **a** vertegenwoordigd een oproep van **car** en elke **d** een oproep van **cdr**.
Bijvoorbeeld:

(caar x) komt overeen met **(car (car x))**
(cdar x) komt overeen met **(cdr (car x))**
(cadar x) komt overeen met **(car (cdr (car x)))**
(cadr x) komt overeen met **(car (cdr x))**
(cddr x) komt overeen met **(cdr (cdr x))**
(caddr x) komt overeen met **(car (cdr (cdr x)))**

Ik verwacht niet dat je bovenstaande geneste commando's vaak zult gebruiken, aangezien de kans dat je de verkeerde gebruikt relatief hoog is.

Echter **car**, **cdr** en **cadr** zul je heel veel gebruiken om o.a. de coördinaten van een punt in aparte variabelen op te slaan, of om het tweede element van een *dotted pair* te krijgen.

4.3) De nth functie

Behalve de **car**,**cdr** en **cadr** functies is er ook nog de **nth** functie om elementen uit een lijst te vissen. De syntax is als volgt:

(nth num list)

num = het nummer c.q. de index van het element van de lijst dat geretourneerd moet worden.

Bijvoorbeeld:

(setq Moeren '("M10" "M20" "M30" "M40"))
(setq Keuze (nth 0 Moeren))

retourneerd:

("M10")

Wat, het nulde element? En ik krijg het eerste element?
Ja en nee, voor ons simpele mensen is het eerste element in een lijst het eerste en het tweede element het tweede. Een computer is vééééél slimmer, die zegt namelijk: "Het eerste getal dat ik ken is 00000000 en het tweede 00000001."
Dus voor AutoLISP is het eerste element in een lijst het nulde element, oftewel het element met de index 0, en het tweede het element met de index 1, en om het helemaal gemakkelijk te maken geldt dit niet voor een tekst. Bij een tekst heeft het eerste karakter de index 1.
Tussen haakjes, wat is een tekst anders dan een lijst van woorden en leestekens en is elk woord niet ook een lijst van karakters? Bovendien kan men een zin ook nog beschouwen, en bewerken, als een lijst van ASCII codes.

(setq Keuze (nth 1 Moeren))

Retourneert:

("M20")

(setq Keuze (nth 3 Moeren))

Retourneert:

("M40")

5) Lijsten maken

5.1) De list functie

We weten nu hoe we elementen uit een lijst moeten halen, maar als we nu een nieuwe lijst willen maken?
Laten we eens een basis voorbeeld bekijken. We gaan een rechthoek tekenen. Hoe gaan we het nieuwe AutoCAD commando noemen? Laten we zeggen ***Rechthoek.*** Type nu eerst:

```
(defun c:Rechthoek ()
)
```

Vraag nu om de hoekpunt linksonder:

```
(defun c:Rechthoek ()
  (setq Hoek1 (getpoint "\nKies hoek linksonder: "))
)
```

Dan vragen om de hoek diagonaal er tegenover te kiezen:

```
(defun c:Rechthoek ()
  (setq Hoek1 (getpoint "\nKies hoek linksonder: "))
  (setq Hoek3 (getcorner Hoek1 "\nKies hoek rechtsboven: "))
)
```

Dan gaan we de hoek rechtsonder bepalen:

```
(defun c:Rechthoek ()
  (setq Hoek1 (getpoint "\nKies hoek linksonder: "))
  (setq Hoek3 (getcorner Hoek1 "\nKies hoek rechtsboven: "))
  (setq Hoek2 (list (car Hoek1) (cadr Hoek3)))
)
```

Wat doen we hier nu precies? We halen het eerste element (de x-coördinaat) van de hoek linksonder uit de lijst **Hoek1** en het tweede element (de y-coördinaat) uit de lijst Hoek3. Samengevoegd d.m.v. de functie **list**

```
(list (car Hoek1) (cadr Hoek3))
```

vormen ze de coördinaten van **Hoek2**. We doen nu hetzelfde voor de hoek linksboven.

```
(defun c:Rechthoek ()
  (setq Hoek1 (getpoint "\nKies hoek linksonder: "))
  (setq Hoek3 (getcorner Hoek1 "\nKies hoek rechtsboven: "))
  (setq Hoek2 (list (car Hoek1) (cadr Hoek3)))
```

```
  (setq Hoek4 (list (car Hoek3) (cadr Hoek1)))
)
```

In plaats van

```
(list (car Hoek3) (cadr Hoek1))
```

kan men ook schrijven:

```
'((car Hoek3) (cadr Hoek1))
```

maar ook:

```
(quote((car Hoek3) (cadr Hoek1)))
```

Naar alle waarschijnlijkheid zul je de tweede schrijfwijze, de apostrofe (‘), het meest gebruiken, want die vergt minder typen.
We hebben nu alle punten die we nodig hebben om de rechthoek te tekenen. Dat doen we met het AutoCAD commando ***line***.

(defun c:Rechthoek ()

```
  (setq Hoek1 (getpoint "\nKies hoek linksonder: "))
  (setq Hoek3 (getcorner Hoek1 "\nKies hoek rechtsboven: "))
  (setq Hoek2 (list (car Hoek1) (cadr Hoek3)))
  (setq Hoek4 (list (car Hoek3) (cadr Hoek1)))
  (command "line" Hoek1 Hoek2 Hoek3 Hoek4 "c")
)
```

Wij hebben bovenstaande rechthoek getekend door twee lijsten (**Hoek1** en **Hoek3**) te manipuleren en daaruit twee nieuwe lijsten te construeren (**Hoek2** en **Hoek4**).
Om het programma nu netjes af te werken voegen we de volgende code toe:

```
(defun c:Rechthoek (/ Hoek1 Hoek2 Hoek3 Hoek4)
  (setq Hoek1 (getpoint "\nKies hoek linksonder: "))
  (setq Hoek3 (getcorner Hoek1 "\nKies hoek rechtsboven: "))
  (setq Hoek2 (list (car Hoek1) (cadr Hoek3)))
  (setq Hoek4 (list (car Hoek3) (cadr Hoek1)))
  (command "Line" Hoek1 Hoek2 Hoek3 Hoek4 "c")
  (princ)
)
```

(princ) is om het programma stilletjes af te sluiten, dus geen afsluitend commentaar op de ***command line***.

5.2) De cons functie

De cons functie is de *basic list constructor* oftewel de meest simpele lijsten maker.
Je neemt een element en nog een element en met **cons** maak je er een lijst van.

```
(cons 100 208)
```

geeft als resultaat:

```
(100 . 208)
```

Dit is een ***dotted pair.*** Men noemt deze lijst ook wel een *asoiciation list*. Deze vorm van lijst wordt gebruikt in de database van de tekening. Zonder er nu diep op in te gaan kan men zeggen dat het eerste element aangeeft **wat** het is en het tweede element de **waarde**. Bijvoorbeeld: (0 . “LWPOLYLINE”), waarbij de 0 aangeeft dat het om een elementnaam gaat en “LWPOLYLINE” de naam van het element is.

```
(setq Element “LWPOLYLINE”)
(cons 0 Element)
```

heeft als resultaat:

```
(0 . "LWPOLYLINE")
```

Maar als je een element neemt en als tweede element een lijst dan wordt het nieuwe element aan het begin van de lijst gezet.

(setq Maanden '("Jan" "Feb" "Mrt" "Apr" "Mei" "Jun" "Jul " "Aug" "Sep" "Okt" "Nov"))
(setq Maanden (cons "Dec" Maanden))

geeft:

("Dec" "Jan" "Feb" Mrt" " Apr" " Mei" " Jun" " Jul " "Aug" " Sep" " Okt" " Nov")

en dit is een gewone lijst alleen staat **Dec** op de verkeerde plaats.

6) Lijsten manipuleren

6.1) De append functie

De append functie neemt een of meer lijsten en voegt ze samen tot één lijst in de volgorde waarin de lijsten aangeleverd worden.

(setq Maanden '("Jan" "Feb" "Mrt" "Apr" "Mei" "Jun" "Jul " "Aug" "Sep" "Okt" "Nov"))
(setq Maanden (append (cons Maanden "Dec")))

geeft als resultaat:

("Jan" "Feb" "Mrt" "Apr" "Mei" "Jun" "Jul " "Aug" "Sep" "Okt" "Nov" "Dec")

Nog een voorbeeld:

(setq x 10)
(setq y 20)
(setq z 0)
(setq Hoek1 (append x y z))

Het resultaat is:

(10.0 20.0 0.0)

6.2) De acad_strlsort functie

De functie **acad_strlsort** sorteert lijsten van tekst elementen.

(setq Maanden '("Jan" "Feb" "Mrt" "Apr" "Mei" "Jun" "Jul " "Aug" "Sep" "Okt" "Nov"))
(setq Maanden (cons "Dec" Maanden))
(setq Maanden (acad_strlsort Maanden))

Dit levert op:

("Apr" "Aug" "Dec" "Feb" "Jan" "Jul " "Jun" "Mei" "Mrt" "Nov" "Okt" "Sep")

6.3) De last functie

De **last** functie retourneert het laatste element in een lijst.

```
(setq  Maanden '("Jan" "Feb" "Mrt" "Apr" "Mei" "Jun" "Jul " "Aug" "Sep" "Okt"
"Nov"))
(setq Maanden (append (cons Maanden "Dec")))
(setq Laatste (last Maanden))
```

Heeft als resultaat:

("Dec")

6.4) De length functie

De **length** functie wordt gebruikt het uitvoeren van de bewerkingen op of met alle elementen in de lijst.
Om de lengte van een lijst te bepalen :

```
(setq  Maanden '("Jan" "Feb" "Mrt" "Apr" "Mei" "Jun" "Jul " "Aug" "Sep" "Okt"
"Nov"))
(setq Maanden (append (cons Maanden "Dec")))
(setq Aantal Maanden (length Maanden))
```

en dat geeft:

12

6.5) De member functie

Met de **member** functie controleer je of een element deel uitmaakt van een lijst. Dit is hoofdletter gevoelig.

```
(setq  Maanden '("Jan" "Feb" "Mrt" "Apr" "Mei" "Jun" "Jul " "Aug" "Sep" "Okt"
"Nov"))
(setq Maanden (append (cons Maanden "Dec")))
(member "Mei" Maanden)
```

retourneert:

("Mei" "Jun" "Jul " "Aug" "Sep" "Okt" "Nov" "Dec")

maar:

```
(setq  Maanden '("Jan" "Feb" "Mrt" "Apr" "Mei" "Jun" "Jul " "Aug" "Sep" "Okt"
"Nov"))
(setq Maanden (append (cons Maanden "Dec")))
(member "mei" Maanden)
```

retourneert:

nil

6.6) De listp functie

Met de listp functie controleer je of een item een lijst is.

(listp Maanden)

retourneerd :

T

(listp setq)

geeft:

nil

6.7) De assoc functie

Met de **assoc** functie haal je bepaalde waardes, zoals laagnaam, uit de *association list* van een element. In een *association list* staan alle eigenschappen van een element, met uitzondering van de default waarden. Later meer hierover. Type in de ***command window*** van AutoCAD:

(setq Lijn (entget (car (entsel)))

selecteer een lijn in de tekening en je krijgt een *association list* zoals:

((-1 . <Entity name: 40085050>)
(0 . "LINE") (330 . <Entity name: 4008dcf8>)
(5 . "11F2") (100 . "AcDbEntity")
(67 . 0)
(410 . "Model")
(8 . "RO_water_Cl2")
(100 . "AcDbLine")
(10 117.0 335.0 0.0)
(11 129.0 335.0 0.0)
(210 0.0 0.0 1.0))

Om nu bijvoorbeeld de laagnaam op te vragen type je:

(setq Laag (cdr (assoc 8 Lijn)))

en je krijgt:

("RO_water_Cl2")

6.8) De subst functie

Met de subst functie vervang je alle elementen met dezelfde naam in een lijst met een ander element. De vergelijking is context gevoelig.

(setq Lijst (10 20 30 “Jan” “Piet” “Wim” 10 20 30))
(subst 40 10)
(subst “Kees” “Piet”)

Dit retourneerd de lijst:

(40 20 30 “Jan” “Kees” “Wim” 40 20 30)

7) Tot slot

In deze les hebben we enkele functies gezien die lijsten maken en manipuleren.
Als je denkt dat was het dan vergis je je toch heel erg.
Er is nog veel meer mogelijk. Maar dat behandelen we in een andere les.
De volgende les gaat over selecteren en selectie sets.

Tot dan,

Joop