DERS.1 AutoLISP'

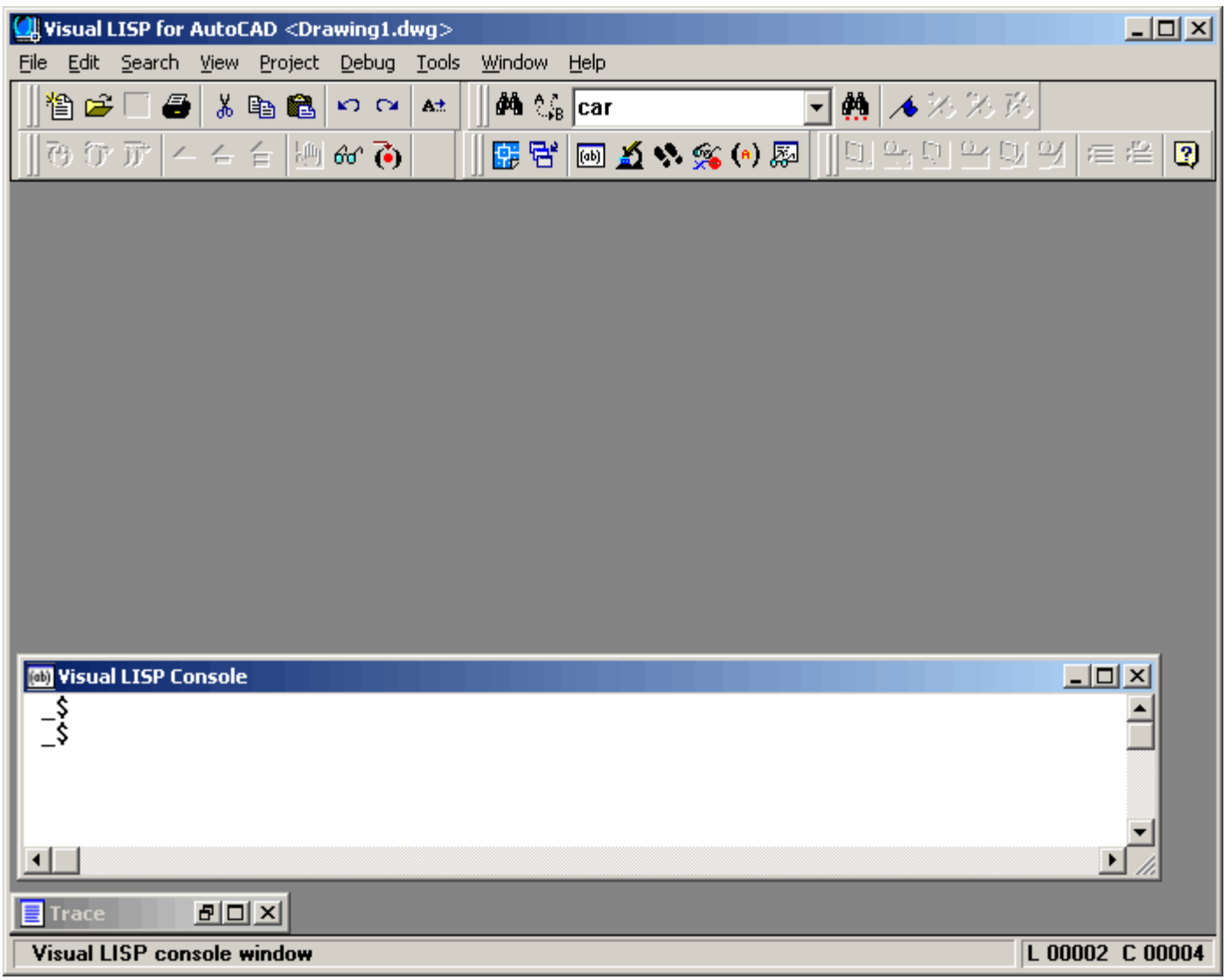
AutoLISP Nedir?

AutoLISP parantezler dili diye de bilinen LISP (List Processor) programlama dilinin, Autodesk tarafından Autocad için özel fonksiyonlar eklenmiş bir alt versiyonudur. Standart LISP fonksiyonlarının yanı sıra karmaşık Autocad işlevlerini yerine getirebilecek fonksiyonlarla güçlendirilmiştir. LISP programlama dili yapay zeka çalışmaları için geliştirilen, liste fonksiyonları güçlü bir dildir ve bir yorumlayıcıya ihtiyaç duyar. Autlisp'in yorumlayıcısı Autocad içinde gömülü olarak gelir.
AutoCAD 2.0 sürümünden beri Autocad'in özelleştirilmesi için kullanılan AutoLISP dili, başlarda profesyonel Autocad eklentileri yapmak için idealdi. Ancak Autocad 11 ile birlikte gelen yeni grafik arayüzler ve ObjectARX gibi C++ ile programlanabilme özelliği AutoLISP'i daha çok bir script dili haline getirdi.
Her şeye rağmen AutoLISP, Autocad nesnelerini, veri tablolarını ve komutlarını yönetebilen oldukça gelişmiş bir Autocad programlama aracıdır. Autocad'in içinde gömülü olarak gelen Visual LISP editörü ile, ofis içi otomasyonlarınız için ideal bir dildir.

Basit Bir LISP programı

```
(defun ilkFonksiyonum()
..(princ "Merhaba Dünya")
..(princ)
)
```

Gördüğünüz gibi her LISP fonksiyonu bir parantezle başlıyor ve parametreler girildikten sonra parantezler kapanıyor. LISP dili "kayıp parantezler" dili diye bilinir. Açtığınız parantezi kapamadığınızda yazdığınız dosyayı yüklerken mutlaka hata mesajı alırsınız. Şükür ki AutoCAD 2000'den itibaren Visual LISP editörü var ve modern IDE lerde olması gerektiği gibi dil kontrolü yapabiliyor. Madem konu açıldı nedir şu Visual LISP editörü görelim. Visual LISP editörü Autocad komut satırından VLISP komutu yazdığınızda karşınıza çıkan penceredir. **Şekil.1.1**

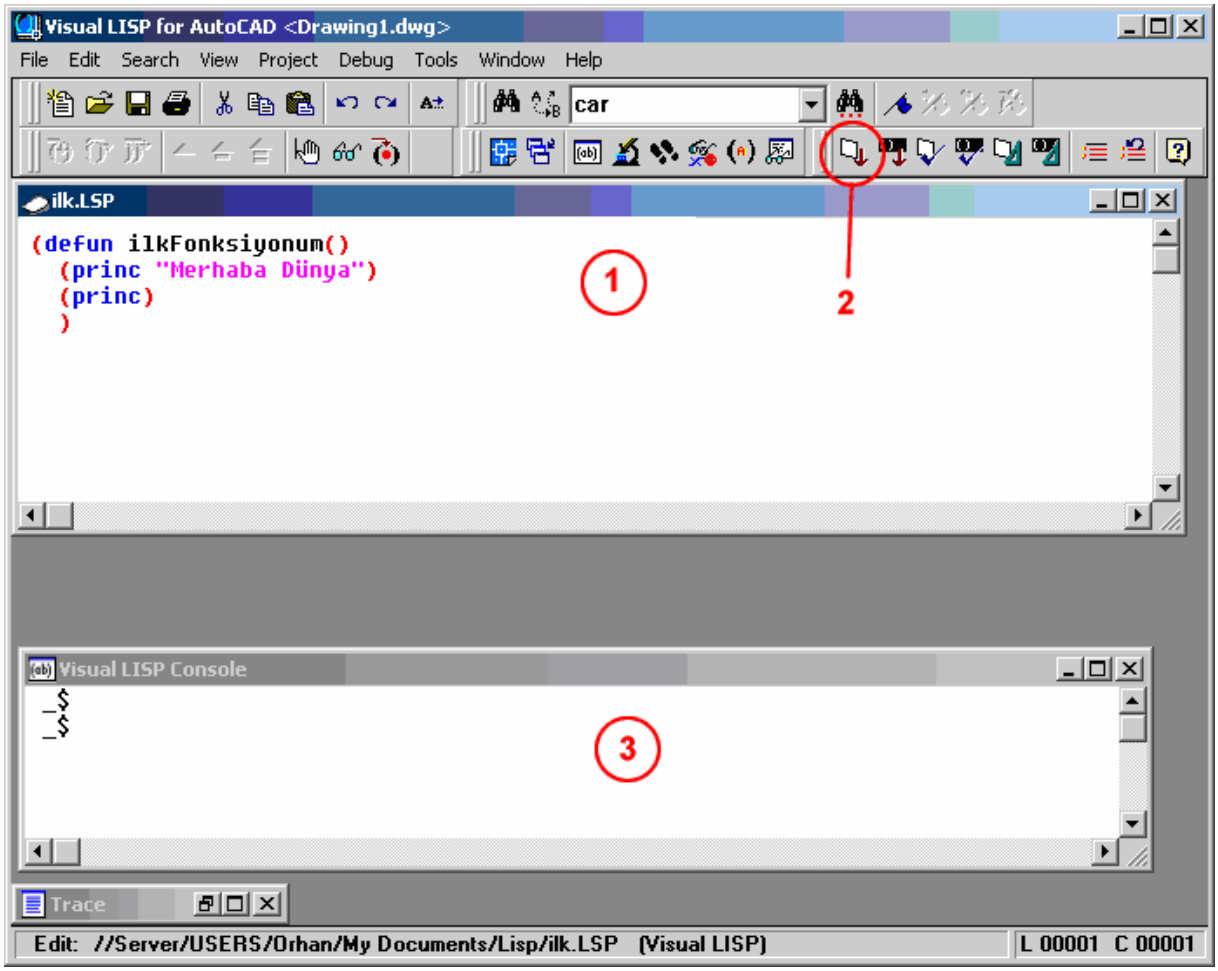


Şekil.1.1 Visual LISP IDE

İlk açıldığında karşınıza Şekil.1.1 deki manzara çıkar.

İlk LISP programım

Hiç lafı uzatmayalım ve ilk programımızı yazalım. Yeni bir dosya açmaya ihtiyacınız var ve bunu nasıl yapacağınızı biliyorsunuz. Yeni bir dosya açın ve aşağıda **Şekil 1.2.** deki kodu oluşturun ve **`ilk.lsp`** diye kaydedin. İşte ilk AutoLISP programınızı oluşturdunuz.

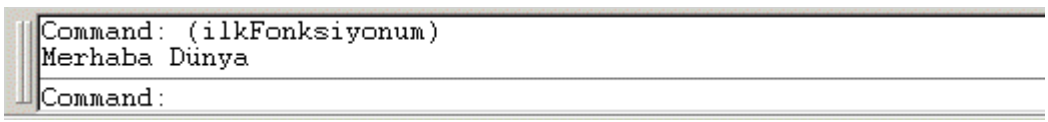


Şekil.1.2. İlk AutoLISP programınız.

Bu programı çalıştırmak için önce yüklemelisiniz. **Şekil 1.2**'de 2 numaralı okun gösterdiği düğmeye (Load Active Edit Window) komutuna basın. Eğer programı doğru kodladıysanız aşağıdaki iletiyi alırınız.

Şekil.1.3. Kod doğruysa konsolda alacağınız mesaj böyle olacaktır.

Programınızı çalıştırmak için Autocad komut satırından (ilkFonksiyonum) yazmanız yeterli olacaktır. **Şekil 1.4**



Şekil.1.4. İşte ilk programınız çalıştı.

DERS.2 Visual LISP editörü ile program geliştirmek.
Seviye: Başlangıç / Kaynak: Autodesk VisualLISP yardım dosyası

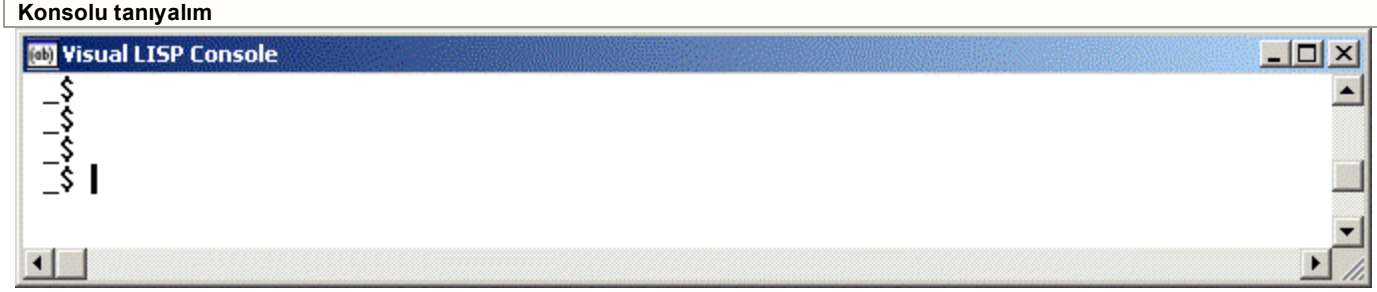
Organize olun...

Hangi programlama dili ile çalışırsanız çalışın, gerçek anlamda bir program geliştirmek istiyorsanız organize olmanız gerekir. Eğer VLISP ile bir AutoLISP programı geliştirmek istiyorsanız aşağıdaki adımları gerçekleştirmeniz gerekir.

- Programınızın yapmasını istediğiniz işlere karar verin ve bu görevlere nasıl yaklaşacağınızı planlayın.
- Programı tasarlayın.
- Kodu yazın
- Kodun okunabilir olmasını sağlayın
- Hata kontrolünü yapın
- Programı test edin ve hata ayıklayın

Bu dersimiz boyunca VLISP ortamında AutoLISP programlamanın ana hatlarını öğrenmiş olacağız. Dersin amacı VLISP programlama ortamını tanımak ve programlama alışkanlığınızı bu ortama göre standartlaştırmaktır.

Konsolu tanıyalım



Şekil.2.1 VLISP Konsolu

AutoLISP programlarınızı geliştirme esnasında VLISP konsolu en sık başvuracağınız alandır. **_$** ile belirlenmiş ayraçın sağına denemek istediğiniz fonksiyonları, anında görmek istediğiniz sonuçları yazabilirsiniz. Bu pencereyi Visual Studio'nun **immediate window**'unla aynıo işlevi görmektedir. VLISP program geliştirme süreciniz boyunca bu pencerede olup biteni saklayacaktır. Böylece istediğiniz zaman içeriği geriye doğru kaydırıp analizler yapabilirsiniz.

Konsol penceresi Autocad'in komut penceresi ile benzerdir. Bu iki pencere birbirine benzer olmasına rağmen, her zaman her iki pencerede de aynı komutları kullanamazsınız. Örneğin a değişkenine yüklü değeri görmek için konsol penceresinde a yazıp ENTER'e basmak yeterliyken Autocad komut satırında a nın değerini öğrenmek için !a yazmalısınız. Konsol penceresinde aşağıdaki işlevleri gerçekleştirebilirsiniz:

- Herhangi bir LISP koduna devam etmek için CTRL+ENTER yaparsanız kodu çalıştırmadan yazmaya devam edebilirsiniz.
- Yazdığınız kodu çalıştırmak için ENTER'e basmanız yeter.
- Eğer bir bölümü işaretleyip ENTER'e basarsanız sadece işaretlediğiniz kod çalıştırılır.

Autocad komut penceresi ile konsol arasındaki bir fark da SPACE tuşunun kullanımıdır. Autocad'de SPACE, komut onayı için kullanılırken konsol penceresinde boşluk anlamına gelir.

Birden fazla çizim için konsol penceresi

Autocad'de açtığınız çizim adedi ne olursa olsun sadece tek bir konsol penceresi vardır. Autocad'de aktif olan çizime ait komutları konsol penceresinde görürsünüz. Ya da Autocad'i etkileyecek bir komut verdiğinizde o anda hangi çizim penceresi aktifse o pencerede komut çalışır. Konsol penceresi hangi çizim aktif ise ona ait tarihçeyi ve dosyaları içerir.

Konsol Kısayollarını Kullanmak

Konsol menüde çalışırken ihtiyaç duyacağınız en yaygın komutlar farenizin sağ tuşu ile aktif olacak bir kısayol menüsüne konmuştur.

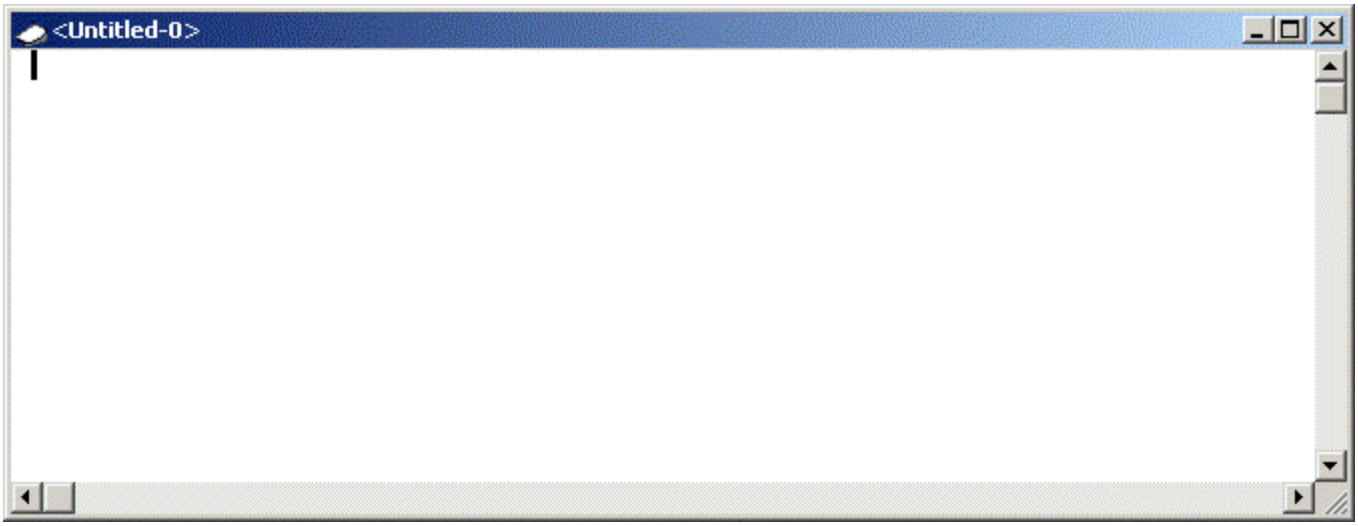
Komut	Yaptığı işlem
Cut	Kesme işlemi
Copy	Kopyalama işlemi
Paste	Yapıştırma (Clipboard içeriğini)
Clear Console Window	Konsol penceresini temizler
Inspect	Inspect diyalog kutusunu açar
Add Watch	Watch diyalog kutusunu açar
Apropos Window	Apropos penceresini açar
Symbol Service	Sembol servis penceresini açar
Undo	Geri alma işlemi
Redo	İleri alma işlemi

Autocad Mode	Konsoldaki içeriği Autocad komut satırı formatına çevirir
Toggle Console Log	Konsol penceresi çıkış içeriğini log'a kopyalar

Konsol penceresinin logunu tutmak.

İleride yapmanız gerekebilecek muhtemel analizler için konsol penceresinin aktivitelerine ait bir log dosyası tutmak isteyebilirsiniz. Bunu yapmak için konsol penceresinin içinde herhangi bir alana farenizin sağ tuşuyla klikleyin ve açılan menüden "**Toggle Console Log**" deyin. Log dosyasının yerini gösterin ve adını girin. Bundan sonra konsolunuzun tüm içeriği ve aktiviteleri bu dosyaya kaydolacaktır.

Text Editörünü Kullanmak



Şekil.2.2. VLISP Text Editörü

Eğer sadece bir kaç LISP komutunu denemek istiyorsanız konsol yeterlidir. Ancak bir program yazmak ve onu saklamak, gerektiğinde çiziminize yükleyip çalışmak istiyorsanız text editörüne ihtiyacınız vardır. Text editörü VLISP'in ana bir unsuru olup kullanımı Windows programlarına aşina olanlar için oldukça basittir. Visual LISP text editörü, parantez bulma, sözdizimi renklendirme ve keyword'leri otomatik algılama gibi kod yazımını kolaylaştıracak yeteneklerle donatılmıştır. Text editöründe yeni bir dosya açmak için **File/New File...** seçeneği yeterlidir. Yeni bir dosya açtınız bile. Tıpkı diğer ditör programlarında olduğu gibi yazım kuralları aynıdır. Konsol penceresinde olduğu gibi text editörünün de kısayol menüsü mevcuttur. Sağ tıklamayla açılan menünün içeriği şöyledir.

Komut	Yaptığı işlem
Cut	Kesme işlemi
Copy	Kopyalama işlemi
Paste	Yapıştırma (Clipboard içeriğini)
Find	Kelime arama penceresi açılır
Inspect	Inspect diyalog kutusunu açar
Toggle Breakpoint	Breakpoint imi koyar veya kaldırır
Apropos Window	Apropos penceresini açar
Add Watch	Watch diyalog kutusunu açar
Undo	Geri alma işlemi
Redo	İleri alma işlemi
Symbol Service	Sembol servis penceresini açar

Renk Kodlamaları

Color	AutoLISP language element
Blue	Lisp dili fonksiyonları ve korunmuş semboller
Magenta	Dizeler
Green	Tam sayılar
Magenta/Gray Bg	Açıklama satırları
Red	Parantezler

Kod yazımı sırasında yardım almak için, kursörünüz yardım almak istediğiniz fonksiyonun sonundayken F1 tuşuna basın.

Kodunuzu anlaşılır yazın

Yazdığınız kodun sonradan okunabilirliğini sağlamak çok önemlidir. Bu yüzden kodunuzu okunaklı yazmalısınız. **Tools > Environment Options > Visual LISP Format Options** menüsünden yazım formatınızı ayarlayabilirsiniz. Anlaşılır bir LISP kodu aşağıdaki gibi olmalıdır. **Şekil 2.3**

arpo.LSP

```
1 (defun yinyang (/ origin radius i-radius half-r origin-x origin-y)
2 (setq half-r (/ radius 2))
3 (setq origin-x (car origin))
4 (setq origin-y (cadr origin))
5 (command "_.CIRCLE"
6          origin
7          radius
8          (command "_.ARC"
9                "_C"
10               (list origin-x (+ origin-y half-r))
11               (list origin-x (+ origin-y radius))
12                origin
13         )
14         (command "_.ARC"
15               "_C"
16              (list origin-x (- origin-y half-r))
17               (list origin-x (- origin-y radius))
18               origin
19         )
20 )
```

Şekil.2.3. Düzgün kodlamaya bir örnek

DERS.3 AutoLISP Deyimleri, Değişkenleri ve Veri Tipleri

AutoLISP Deyimleri

Bir AutoLISP programı alt alta seri halinde deyimlerden oluşur. En basit anlamıyla bir LISP deyimi aşağıdaki gibidir.

```
(fonksiyon argümanlar)
```

Her LISP deyimi bir parantez açarak başlar. Bunu fonksiyon ve ona bağlı argümanlar (parametreler) izler ve parantez kapanarak biter. Her deyim kendini kapsayan bir başka deyimin kullanacağı bir dönüş değeri oluşturur. En son yorumlanan deyimin değeri kendini çağıran deyime döner. Örneğin:

```
(* 2 (+ 5 3))
16
```

Çarpma fonksiyonun birinci argümanı bir numaradır. İkinci argüman ise bir toplama işleminin sonucudur. Yorumlayıcı önce en içteki parantezlerde bulunan deyimi yani 5 + 3 işlemini yorumlar ve sonuç değerini çarpma işlemini bulunduran deyimin 2. argümanı olarak döner. Eğer kapatmadığınız eksik parantez olursa 16 sonucu yerine aşağıdaki gibi görünür.

```
(* 2 (+ 5 3)
(_>
```

Bu durumda hemen eksik kalan parantezi kapatırsanız yorumlayıcı görevine devam eder.

```
(* 2 (+ 5 3)
(_>)
16
```

AutoLISP Veri Tipleri

AutoLISP deyimleri parantezlerin içindeki veri tpilerinin sıralamasına ve niteliklerine göre işlem görür. AutoLISP'i tam anlamıyla kullanmanız için veri tiplerini iyice anlamalısınız.

Integer (Tamsayılar)

İçinde ondalık ayracı olmayan sayılardır. 32 bitlik bu sayılar -2,147,483,647 ile +2,147,483,648 değerleri arasındaki tüm tamsayılardır. Eğer bu aralığın dığında bir tamsayı girecek olursanız yorumlayıcı bunu otomatik olarak reel sayıya çevirecektir. Ancak örneğin, iki geçerli tamsayının toplamı 32 bitlik sistemde olabilecek en büyük tamsayının değerinini geçiyorsa sonuç geçersiz olacaktır. Örnek:

```
(+ 2147483646 3)
-2147483647
```

Sonuç negatif tamsayı çıkar.

Real (Reel Sayılar)

AutoLISP te reel sayılar floating point formatında olan noktadan sonra 14 hane hassasiyetli sayılardır. AutoLISP yorumlayıcısı için gerçek sayılara örnek olarak 3.1 -5.4 0.0000013 21,000,000.0 verilebilir.

String (dizi)

İçinde bir dizi alfanümerik karakter bulunduran katarlardır. "Kelime 1" "kitap" "H&B" "?*_i" gibi dizeler AutoLISP yorumlayıcısı için geçerli dizelerdir. \ karakteri AutoLISP dizelerinde kontrol karakeri girmenizi sağlar. Bu kontrol karakterlerinin listesi aşağıdaki gibidir.

Kod	Açıklama
\\	**\ karakteri için** `(princ "\\ahmet") "\ahmet"`
\"	**" karakteri** `(princ "\"Deneme\"") ""Deneme""`
\e	**Escape**

\n	**Yeni satıra geçmek için**
	`(princ "Bu ilk Satır \nBu ikinci satır")`
	`Bu ilk satır`
	`Bu ikinci satır`

\r	**ENTER**

\t	**TAB bir tab sağa geçirir**
	`(princ "Ali\tVeli\tAyşe")`
	`Ali.......Veli........Ayşe`

\ nnn	**nnn kodundaki ACCII karakteri**

Lists (Listeler)

Birebirlerinden bir boşlukla ayrılan parantez içindeki değerler silsilesine liste denir. Birbiri ile ilgili birçok sayıyı bir arada saklamanın en etkili yolu listelerdir. Örneğin bir noktayı (13.45 23.50 0.00) listesinde saklayabiliriz. İleride listelerle ilgi çok daha fazla ayrıntıyı paylaşacağız.

Selection Sets (Seçim setleri)

Bir veya birden fazla Autocad nesnesinin oluşturduğu gruplardır. AutoLISP kullanarak seçim setlerine nesne ekleyebilir ya da çıkarabilirsiniz.

Entity Name (Varlık isimleri)

Çizimdeki nesnelere Autocad tarafından verilen nümerik etikete varlık ismi denir. Varlık isimleri çizim içindeki nesnelerin veritabanına ulaşmak için kullanılabilir. Seçilen nesneler üzerinde değişik işlemler yapmak için çok çeşitli LISP rutinleri vardır. Bu konuyu ilerideki derslerimizde geniş bir biçimde ele alacağız.

AutoLISP Değişkenleri (Variables)

Her programla dilinde olduğu gibi AutoLISP dilinde de değişkenler vardır. AutoLISP dilinde bir değişkeni önceden tanımlamanız gerekmez. Değişken atandığı anda eşitlenir ve eşitlediğiniz veri tipinde yaratılır. Birkaç değişken yaratalım. AutoLISP değişkeni yaratma, tanımlama ve atama için tek bir fonksiyon kullanılır.

```
(setq isim "Haluk")
```

Yukarıdaki değim isim adında bir değişken yaratır, string olarak tanımlar ve "Haluk" değerini bu değişkene atar. AutoLISP bir yorumlayıcı olduğu için değişkenlerin tümü (fonksiyon argümanları hariç) global yani diğer fonksiyonlar tarafından tanınabilirdir. AutoLISP değişkenlerinin tümü global olduğuna göre değişken ismi verirken dikkatli olmalıyız.

```
(setq intNoktaAdedi 5)
(setq strAd "Orhan" strSoyad "Toker")
(setq lstBaslangicNoktasi (getpoint "\nBir nokta girin"))
```

Yukarıdaki tüm örneklerde değişkenlere verilecek isimlere dikkat edilmiştir. Autolisp yorumlayıcıda değişken tipleri önceden tanımlanamadığı ve tüm değişkenler global olduğu için, kodlarken değişkeninizin tipine uygun ön ekler koymaya özen gösterirseniz işinizi kolaylaştırmış olursunuz.

Bir değişkenin değerini izlemek

Herhangi bir değişkenininizin değerini o anda öğrenmek istiyorsanız konsol veya komut satırından bunu yapabilirsiniz. İkisi arasında çok az bir fark vardır.

```
$_strAd ;konsoldan öğrenme
"Orhan"

Command: !strAd ;komut satırından öğrenme
"KTBMYO"
```

Gördüğünüz gibi değerini öğrenmek istediğiniz değişkenin adının başına ! işareti koymanız yeterli.

Önceden tanımlanmış değişkenler

AutoLISP'te değişken ismi olarak kullanamayacağınız önceden tanımlı 3 değişken vardır.

```
PI Pi sabiti
T .True
PAUSE komut duraksaması
```

Forcing (zorlama)

İki tamsayının bölümünü gerçek sayı elde etmek için

```
(/ 12 5) ;Forcing yapılmadan
2

(/ 12 5.0); forcing ile
2.4
```

DERS.4 (command) fonksiyonu
Seviye: Başlangıç Kaynaklar: Autodesk, VisualLISP IDE Help file

Burada **(command)** fonksiyonunu ele alınacak. Bildiğiniz gibi AutoLISP, Autocad için bir makro dili olmaktan çok daha öte bir dil olmasına karşın temelde Autocad'e scriptler yazmaktır. Eh biz de işe temelden başladığımıza göre, Autocad'e bir şeyler yaptırmanın ek kolay yolu olan **(command)** fonksiyonunu inceleyeceğiz. **(command)** fonksiyonu Autocad'in komut satırına direk komut göndermek için kullanılır. (command) fonksiyonu, kullanacağınız Autocad komutuna bağlı olarak değişik tipte ve sayıda argüman kullanır. Açı, mesafe, nokta gibi veriler tırnak içinde verilebileceği gibi direkt sayı olarak da **(command)** fonksiyonunun argümanlarına pas edilebilir. Fonksiyona pas edeceğiniz boş bir dize "" Autocad komut satırında ENTER ya da SPACE ile aynı işi görecektir. İstisna olarak **(command)** fonksiyonunda SKETCH komutunu ve Autocad komutu olarak tanımlanmış c:nnn lisp fonksiyonlarını kullanamazsınız. Aşağıdaki örnek kod satırları (command) fonksiyonunun bir kaç kullanım şeklini gösterir.

```
(command "circle" "0,0" "3,3")
(command "thickness" 1)
(setq p1 '(1.0 1.0 3.0))
(setq rad 4.5)
(command "circle" p1 rad)
```

Yabancı Dil Desteği

Tasarladığınız programın AutoCAD'in tüm diğer dillerdeki sürümlerinde çalışmasını istiyorsanız, (command) fonksiyonu ile Autocad komutlarını çağırırken alt çizgi (_) kullanmalısınız. Aşağıdaki satır buna örnektir.

```
(command "_line" pt1 pt2 pt3 "_c")
```

Bu bilgiye ek olarak, eğer kullanıcı tarafından yeniden tanımlanmış olma olasılığı olan Autocad komutlarından kaçınmak istiyorsanız nokta (.) karakterini kullanmalısınız. Her ikisini bir arada ya da ayrı ayrı kullanabilirsiniz.

```
(command "._line" pt1 pt2 pt3 "_c")
(command "_.line" pt1 pt2 pt3 "_c")
(command ".line" pt1 pt2 pt3 "c")
```

Kullanıcı girişi için duraklama

Komutun icrası sırasında, nokta girme ya da sündürme gibi kullanıcıdan istenecek girişler için PAUSE değişkeni argüman olarak girilir. Bu menülerdeki ters bölü (\) sembolü ile aynıdır.

Bir örnekle açıklayalım:

```
(command "._circle" PAUSE 50.0)
```

Bu satırı komut satırına girdiğinizde Autocad çember komutuna girecek ve sizden çemberin merkezini girmeniz için duraklayacaktır. Siz herhangi bir nokta girişi yaptığınız anda yarıçapı 50 olan çemberi belirlediğiniz merkeze çizecektir.

Not: Eğer komut icrasını PAUSE ile duraklattıysanız, bu duraklama esnasında şeffaf komutları kullanabilirsiniz. Az önceki örnekte Autocad dairenin merkezini girmeniz için durakladığında 'zoom ya da 'pan şeffaf komutlarını kullanailirsiniz. Farenizin tekerleği dolayısıyal bu bilgi önemini yitirmiş olsa da yazmakta fayda gördüm.

Önemli: Eğer ATTRIBUTE ya da TEXT girişi için PAUSE kullandıysanız, Autocad sadece TEXTEVAL sistem değişkeni 0 değilse veri girişini bekler. TEXTEVAL 0 ise PAUSE duraksaması çalışmayacaktır.

Seçme noktalarının (PICK POINTS) (command) fonksiyonuna pas edilmesi.

TRIM, EXTEND ve FILLET gibi komutlarda hem nesne seçimi hem de seçim noktasının belirlenmesi bir arada yapılır. Bunun gibi hem seçilen nesneyi hem de seçim noktası verisini aynı anda (command) fonksiyonuna pas etmek PAUSE argümanıyla mümkün olamaz. Bunun yerine bu verileri komuttan önce değişkenlere saklamak gerekir. Örnekte bunun yapılışını görüyorsunuz.

```
(command "._circle" "5,5" "2") ;daireyi çizer
(command "._line" "3,5" "7,5" "") ;çizgiyi çizer
(setq el (entlast)) ;son çizilen nesneyi el değişkenine saklar
(setq pt '(5 7)); seçim noktasını pt değişkenine saklar
(command "trim" el "" pt "") ;trim komutu icra edilir
```

Bu tip işlemler için dikkatli olmalısınız.

- Pick point ve işlem yapılacak nesneler o anda model penceresinin içinde olması gerekir
- Gerekirse "PICKBOX" sistem değişkeninin değerini biraz büyütebilirsiniz.

Parantez içindeki tüm sayıların toplamını döner

```
(+ [sayı sayı] ...)
```

Argümanlar

Sayı
Herhangi bir sayı

Döndüğü değerler

Girdiğiniz sayıların toplamını döner. Eğer tek bir sayı yazdıysanız 0 ile toplamını döner.

Örnekler

`(+ 1 2)`	3 değerini döner
`(+ 3 4 5.54)`	12.54 değerini döner
`(+ 3 4 5.0)`	10.0 Değerini döner

Parantez içindeki birinci sayıdan diğerlerini çıkarır.

```
(- [sayı sayı] ...)
```

Argümanlar

Sayı
Herhangi bir sayı

Döndüğü değerler

Çıkartma işleminin sonucunu döner. Eğer ikiden fazla sayı varsa birinci sayıdan diğer sayıların toplamını çıkartır. Tek sayı girdiyseniz 0 değerinden girdiğiniz sayıyı çıkartır.

Örnekler

(- 5 3)	2 döner
(- 15 4.5 3.0 2)	5.5 döner
(- 10 2.0)	8.0 döner
(- 9)	-9 döner

Girilen değeri 1 artırır .

```
(1+ sayı)
```

Argümanlar

Sayı
Herhangi bir sayı

Döndüğü değerler

Verilen değeri 1 artırır.

Örnekler

`(1+ 5)`	6 değeri döner
`(1+ a)`	a değişkeninde yüklü değeri 1 artırır
`(1+ 0.5)`	1.5 döner

Girilen değeri 1 eksiltir .

```
(1- sayı)
```

Argümanlar

Sayı
Herhangi bir sayı

Döndüğü değerler

Verilern değeri 1 eksiltir..

Örnekler

`(1- 5)`	4 değeri döner
`(1- a)`	a değişkeninde yüklü değeri 1 eksiltir
`(1+- 0.5)`	- 0.5 döner

Girilen sayıların çarpımını verir

```
(* [sayı sayı] ... )
```

Argümanlar

Sayı
Herhangi bir sayı

Döndüğü değerler

Çarpma işleminin sonucunu verir. Eğer tek parametre girilirse değeri 1 le çarpar. Hiç parametre girilmediğinde 0 sonucunu döner.

Örnekler

`(* 3 4)`	12 değeri döner
`(* 2 3.0)`	6.0 döner
`(* 3)`	3 döner

`(* 5 -3.4)`	-17.0 döner

Girilen birinci sayıyı geri kalan sayıların toplamına böler.

`(*/ [sayı sayı] ... )`

Argümanlar

Sayı
Herhangi bir sayı

Döndüğü değerler

Bölme işleminin sonucu. Eğer 2 den fazla sayı girildiyse, ilk sayının geri kalan sayıların toplamına bölündüğünde çıkan değeri döner. Tek sayı girilirse sayıyı 1 değerine böler. Hiç argüman girilmediyse 0 döner.

Örnekler

`(*/ 100 10)`	10 döner
`(/ 12 3.0)`	4.0 döner
`(/ 100 15 25)`	2.5 döner
`(/ 5)`	5 döner

Girilen değerlerin eşitliğini karşılaştırır

`(= [alfasayı alfasayı] ... )`

Argümanlar

AlfaSayı
Herhangi bir alfanümerik değer

Döndüğü değerler

Eğer tüm girilen değerler biribirine eşitse T, değilse nil döner.

Örnekler

`(= 3 3)`	T
`(= 2 2.0)`	T
`(= "ben" "ben")`	T
`(= "ben" "Ben")`	nil
`(= 2 2 3)`	nil

Verilen tüm değerlerin eşit olmadıklarını kontrol eder.

`(/= [alfasayı alfasayı] ... )`

Argümanlar

AlfaSayı
Herhangi bir alfanümerik değer

Döndüğü değerler

Eğer tüm değerler birbirine eşit değilse T döner. Girilen değer silsilesinde birbirine eşit en az 2 değer varsa nil döner.

Örnekler

`(/= 3 4)`	T
`(/= "ben" "ben")`	nil
`(/= "ben" "Ben")`	T
`(/= 10 20 30 10)`	nil

Girilen değer silsilesinde öncekinin sonrakinden küçük olup olmadığını hesaplar.

`(< [alfasayı alfasayı] ... )`

Argümanlar

AlfaSayı
Herhangi bir alfanümerik değer

Döndüğü değerler

Eğer girilen değer katarında tüm değerler kendinden sonrakinden küçükse T döner. İlk değerin sonrakinden büyük olması durumunda nil döner.

Örnekler

`(< 3 4)`	T
`(< "a1" "b0")`	T
`(< 3 5 7 6 8)`	nil çünkü 7 altıdan büyük
`(< 1 4 6 7)`	T

Girilen değer silsilesinde öncekinin sonrakinden küçük eşit olup olmadığını hesaplar.

```
(<= [alfasayı alfasayı] ... )
```

Argümanlar

AlfaSayı
Herhangi bir alfanümerik değer

Döndüğü değerler

Eğer girilen değer katarında tüm değerler kendinden sonrakinden küçükse ya da eşitse T döner. İlk değerin sonrakinden büyük olması durumunda nil döner.

Örnekler

`(<= 3 4)`	T
`(<= "a1" "b0")`	T
`(<= 3 5 7 6 8)`	nil çünkü 7 altıdan büyük
`(<= 1 4 6 7 7)`	T

Girilen değer silsilesinde öncekinin sonrakinden büyük olup olmadığını hesaplar.

```
(> [alfasayı alfasayı] ... )
```

Argümanlar

AlfaSayı
Herhangi bir alfanümerik değer

Döndüğü değerler

Eğer girilen değer katarında tüm değerler kendinden sonrakinden büyükse T döner. İlk değerin sonrakinden küçük ya da eşit olması durumunda nil döner.

Örnekler

`(> 3 4)`	nil
`(> "b" "a")`	T
`(> 8 6 5 5)`	nil
`(> 8.5 8.49)`	T

Girilen değer silsilesinde öncekinin sonrakinden büyük ya da eşit olup olmadığını hesaplar.

```
(>= [alfasayı alfasayı] ... )
```

Argümanlar

AlfaSayı
Herhangi bir alfanümerik değer

Döndüğü değerler

Eğer girilen değer katarında tüm değerler kendinden sonrakinden büyük ya da eşitse T döner. İlk değerin sonrakinden küçük olması durumunda nil döner.

Örnekler

`(>= 3 4)`	nil
`(>= "b1" "b0")`	T
`(>= 5 5 4 3)`	T
`(>= 3 8 4 3 2)`	nil

Girilen sayının 1'e tamamlayanının tersini döner

```
(~ tamsayı)
```

Argümanlar

Tamsayı
Herhangi bir tamsayı

Döndüğü değerler

Girilen sayının 1'e tamamlayanının tersini döner

Örnekler

`(~ 1)`	-2
`(~ -5)`	4
`(~ 100)`	-101

DERS.5 Kullanıcı giriş fonksiyonları
Seviye: Orta Kaynaklar: Autodesk, VisualLISP IDE Help file

Merhabalar,

Bu dersimizde kullanıcıdan veri toplamanın yollarını öğreneceğiz. Autocad ile çizim yaparken kullandığımız komutlar bizden çeşitli verileri girmemizi ister. Örneğin bir çember çizerken AutoCAD sizden merkez noktasını ve yarıçapı soracaktır. Siz de AutoLISP ile program geliştirirken kullanıcıdan bir çok kez veri girmesini isteyeceksiniz. Aşağıda kullanıcıdan girmesini isteyebileceğiniz veri tiplerini ve hangi fonksiyonla bu veririn toplanacağını gösteren tabloyu göreceksiniz.

İstenen Veri Tipi	Açıklama	Giriş Fonksiyonu
Sayısal değer	Herhangi bir reel ya da tamsayı sayısal değer.	(getint [mesaj]) (getreal [mesaj])
Açı	Autocad açı değeri almak için.	(getangle [nokta] [mesaj])
Diğer köşe	Bir noktadan referans alarak diğer bir köşe almak için kullanılır. Dönüş değeri nokta listesidir	(getcorner nokta [mesaj])
Mesafe	Bir uzaklık değeri girişi için kullanılır	(getdist [nokta] [mesaj])
Seçenek	Kullanıcıdan bir seçenek girmesi için kullanılır.	(getkword [mesaj])
Nokta	Kullanıcıdan nokta girişi istemek	(getpoint [nokta] [mesaj])
Dize	Kullanıcıdan alfanümerik bir giriş istendiğinde kullanılır	(getstring [cr] [mesaj])

Tablo.1 Kullanıcı giriş fonksiyonları

Şimdi bu fonksiyonları birer örnekle kısaca inceleyeceğiz.

(getint) ve (getreal) fonksiyonları

Bu fonksiyonlar kullanıcıdan reel ya da tamsayı girişi istendiği zaman kullanılır. Örnekle

```
(setq (getint "\nKaç adet kat kopyalayacaksınız?:"))

(setq a (getint "\nKaresini almak istediğiniz sayı?:"))
(princ (* a a))

(setq kot (getreal "\nLütfen 1.Kat kotunu girin:"))
```

Gördüğünüz gibi (getint) ve (getreal) fonksiyonları ile birlikte kullanıcıdan veri girişi istemek için bir açıklama mesajı da girebiliyoruz.

Önemli: AutoLISP programlarınızda kullanıcıdan veri girişi esnasında yazacağınız mesaj kullanıcıyı yönlendireceği için çok önemlidir. Eğer kullanıcıdan, nokta, açı, mesafe gibi standart Autocad verileri istiyorsanız mesajınızın da Autocad mesaj standardında olması gerekir. Elbette tüm veri girişi mesajlarınızın da belli bir uslüpta olması tercih edilir. Mesajın başına koyduğunuz "\n" kodu mesajınızın yeni satırda çıkmasını sağlar.

(getangle) ile açı girişi

Kullanıcıdan açı girmesini istiyorsanız bu fonksiyonu kullanırsınız. (getangle) fonksiyonu isteğe bağlı iki adet argüman kullanır:

```
(getangle [nokta] [mesaj])
```

Nokta argümanını girerseniz girdiğiniz nokta referans alınarak açı girişi yaptırırsınız. Bu fazlaca kullanılam bir metot değildir. Biz ikisine de örnek verelim.

```
(setq aci (getangle "\nAçıyı girin:"))
```

Kullanıcının yaptığı açı girişini aci değişkenine radyan olarak girer. Radyanı dereceye çevirmek için

```
(* 180.0 (/ aci pi))

(setq aci (getangle (getvar "LASTPOINT")) "\nYönü girin: "))
```

İkinci örneğimiz girilen son noktadan yön tayini istiyor.

(getcorner) ile diğer köşeyi tayin etmek

Bazı durumlarda kullanıcıdan bir çerçeve tayin etmesini isteyebilirsiniz. Örneğin kendi dikdörtgen rutininizi yazıyorsunuz ya da seçim penceresini önceden tayin ettirmek istediniz. İşte böyle durumlarda (getcorner) çok yararlıdır. Lafı uzatmayalım kendi

dikdörtgen rutinimizi yazalım.

```
(setq oldCmdecho (getvar "cmdecho"))
(setvar "cmdecho" 0)

(defun c:dortgen (/ pt1 pt2 pt3 pt4)

  ; Önce kullanıcıdan köşe noktalarını isteyelim
  (setq pt1 (getpoint "\nDörtgenin 1.köşesini tıklayın:")
        pt3 (getcorner pt1 "\nDiğer köşeyi tıklayın:")
        )

  ; Şimdi diğer iki noktayı biz hesaplayalım
  (setq pt2 (list (car pt3) (cadr pt1) 0.0))
  (setq pt4 (list (car pt1) (cadr pt3) 0.0))

  ; artık çizdirebiliriz
  (command "._pline" pt1 pt2 pt3 pt4 "_c")
  )
(setvar "cmdecho" oldCmdecho)
(princ)
```

Resmin üzerine tıklayarak dortgen.lsp dosyasını indirebilirsiniz.

Bu kodu Autocad vlisp komutu ile açtığınız VisualLISP editöründe oluşturun ve **`dortgen.lsp`** diye kaydedin. Sonra CTRL+ALT+E tuşlarına basarak projeyi Autocad'e yükleyin. komut satırından **`dortgen`** yazdığınızda dikdörtgen çizen komutunuz işlemiş olacaktır. Bu projede kullandığım car, cadr ve list fonksiyonlarını ileriki derslerimde açıklayacağım. Burada mühüm olan **`(getpoint)`** ve **`(getcorner)`** fonksiyonlarını nasıl kullandığımıza dikkat etmenizdir.

Önemli:Dikkat ederseniz aynı (setq) eşitleme fonksiyonu içerisinde dikdörtgenimizin her iki köşesini de kullanıcıdan almış oldum. Yani (setq) eşitleme fonksiyonunu aynı anda birden fazla değişkenin değerini eşitlemek için kullanabilirsiniz. Kodlama sırasında hizalamaya dikkat edin yeter.

Ayrıca c:dortgen fonksiyonumu tarif ederken başına koyduğum **`"c:"`** öneki bunun bir Autocad komutu gibi çalışasını sağlayacaktır. Bu ön eki koymadığınızda komut satırından diğer lisp fonksiyonlarını çağırdığınız gibi çağırırsınız.

```
Command:(dortgen)
```

Fonksiyon tanımında **`(/)`** işaretinden sonra tanımladığım değişkenler sadece fonksiyon içinde lokal olarak tanımlandıklarını gösterir.

(getdist) ile mesafe girişi yaptırmak

`(getdist)` fonksiyonunu kullandığınızda, kullanıcı direkt bir değer girerek, iki nokta belirleyerek, ya da bir noktadan referans alarak mesafe girebilir. **`(getdist)`** fonsiyonu ilk girilen noktadan sonra, Autocad'in her türlü nokta belirleme yöntemini kullarak diğer bir nokta girmenizi bekler. Bu bekleme esnasında 1. noktadan kursörünüzün ucuna bir ipcik çizecektir.

```
(setq mesafe (getdist "\nMesafeyi girin:"))
```

(getkword) ile kullanıcıdan seçenek belirlemesini istemek.

Bazı durumlarda kullanıcıdan komut satırı aracılığı ile seçenek isteyebilirsiniz. Buna örnek olarak Autocad CIRCLE komutunu verebiliriz.

```
Command: circle
Specify center point for circle or [3P/2P/Ttr (tan tan radius)]:
```

Yukarıdaki örnekte **`[3P/2P/Ttr (tan tan radius)]:`** birer seçenektir. Kendimiz bir örnek yapalım. Sorduğumuz soruya sadece "Evet" ya da "Hayır" yanıtını giriş olarak kabul eden kod:

```
(initget 1 "Evet Hayır")
(setq cevap (getkword "\nKomut çizimdeki tüm nesneleri silsin mi? [Evet Hayır] : "))
```

Yukarıdaki kod kullanıcıdan sadece "Evet" ve "Hayır" yanıtlarını kabul edecektir. (initget) fonksiyonu veri girişlerini önceden koşullandırmaya yarar. Bu fonksiyonu gelişmiş derslerimizde daha ayrıntılı ele alacağım.

(getpoint) ile kullanıcıdan nokta istemek

Kullanıcının tek bir nokta girmesini istediğinizde (getpoint) fonksiyonunu kullanabilirsiniz. Kullanıcının vereceği nokta yanıtı tüm

Autocad nokta tayin etme yöntemleri olabilir.

```
(setq pt1 (getpoint "\nDörtgenin 1.köşesini tıklayın:"))
```

Yukarıdaki dortgen.lsp örneğimizde bu fonksiyonu zaten kullanmıştık.

(getstring) ile kullanıcıdan dize girişi istemek.

Özellikle etiket ya da mahal adları dolduran rutinlerinizde sıklıkla kullanacağınız bir fonksiyondur. Kullanımı:

```
(getstring [cr] [mesaj])
```

cr argümanı ENTER ya da boşluk karakterini kabul edip etmeyeceğinizi belirler. nil ise boşluk karakteri giremezsiniz.

```
(setq mahalAdi (getstring "\nMahal adını girin:"))
(setq mahalAciklamasi (getstring T "\nMahal açıklamasını girin:"))
```

Örnekte görüldüğü gibi, isimde boşluk kabul etmezken, açıklamada boşluklara müsade ediyoruz.

Alıştırmalar

Kendi kendinize yapabileceğiniz alıştırmalar:

- (getpoint) kullanarak Autocad çizgi komutunu taklit edin.
- (getdist) ve (getangle) kullanarak polar koordinat sistemi ile çizgi çizdirin. @mesafe<açı
- (getpoint) ve (getdist) kullanarak merkez/yarıçap çember çizdirin.

DERS.6 AutoLISP'te Listelerin Yönetimi (List Handling) I

Seviye: Orta Kaynaklar: Autodesk, VisualLISP IDE Help file

Nokta Listeleri (Point Lists)

AutoLISP, grafik koordinatları yönetmeniz için içinde 2 ya da 3 değer bulunan listeler sunar. Bu listelere nokta listesi deriz.

2B Nokta listesi:

`(34.5 27.3 )` ; x=34.5 ve Y=27.3 olan koordinatı temsil eder.

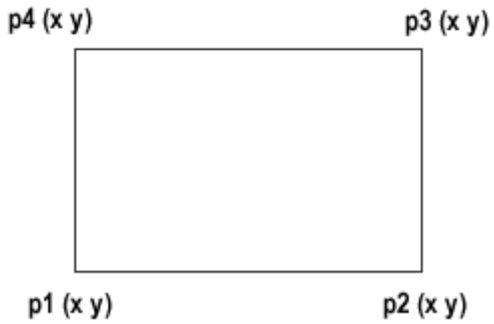
3B Nokta listesi:

`(34.5 27.3 0.0)` ; x=34.5, Y=27.3 ve Z=0.0 olan koordinatı temsil eder.

Herhangi bir Autocad nesnesini oluşturmak istediğinizde nokta listeleri oldukça işinize yarayacaktır.

```
(setq p1 (list 40 40))
(setq p2 (list 100 60))
(command "_.line" p1 p2 "")
```

Bu örnek 40,40 noktasından 100,60 noktasına uzanan bir çizgi oluşturmak için kullanılır. Bu örnekteki gibi basit işler için şu ana kadar öğrendikleriniz yeterlidir. Ancak biraz daha karmaşık örnek yapmaya kalkarsak liste yönetmeyi bilmemiz gerekecektir. Örneğin kendi dörtgen çizme komutumuzu yapalım. Kullanıcıdan dörtgenin iki köşesini isteyebiliriz. Bunu `(getpoint)` ve (getcorner) fonksiyonları ile yaparız. Ancak bu iki fonksiyondan dörtgenimizin sadece çapraz iki köşesini elde edebiliyoruz. Bu değerler bize diğer iki noktayı hesaplamak için yeterlidir. Bu hesabı da liste yönetim fonksiyonları ile yapabiliriz. Şeklimize bir göz atalım:



Şekil.1 Dörtgenin yapısı

```
(setq oldCmdecho (getvar "cmdecho"))
(setvar "cmdecho" 0)

(defun c:dortgen (/ pt1 pt2 pt3 pt4)

  ; Önce kullanıcıdan köşe noktalarını isteyelim
  (setq pt1 (getpoint "\nDörtgenin 1.köşesini tıklayın:")
        pt3 (getcorner pt1 "\nDiğer köşeyi tıklayın:")
        )

  ; Şimdi diğer iki noktayı biz hesaplayalım
  (setq pt2 (list (car pt3) (cadr pt1) 0.0))
  (setq pt4 (list (car pt1) (cadr pt3) 0.0))

  ; artık çizdirebiliriz
  (command "._pline" pt1 pt2 pt3 pt4 "_c")
  )
(setvar "cmdecho" oldCmdecho)
(princ)
```

Noktalı Çiftler (Dotted Pairs)

AutoLISP'te oluşturacağınız listeleri daha iyi organize etmenin bir diğer yolu da noktalı çiftler oluşturmaktır. BU aslında birbirinden nokta (.) ile ayrılmış liste çiftleridir. Soldaki elemanı çiftin adı, sağdakini de değeri olarak düşünebiliriz.

Örneğin;

```
((katmanAdi . "TAL_DUVAR" ) (cizgiKal . 0.5))
```

Listesi bir elemanın katman ve çizgi kalınlığı değerlerini saklar. Bir noktalı çirft oluşturmak için **(cons)** fonksiyonunu kullanmalısınız:

```
$_(cons 'katmanAdi "TAL_DUVAR")
(katmanAdi . "TAL_DUVAR")
```

(cons) fonksiyonunu her kullandığınızda listenin sonuna bir noktalı çift daha ekler:

```
$_(setq a (cons 'adi "Orhan"))
(adi . "Orhan")
$_(setq a (list a (cons 'soyadi "Toker") (cons 'yasi 35)))
((adi . "Orhan") (soyadi . "Toker") (yazi . 35))
```

Noktalı çiftlerin bir önemi de, Autocad nesnelerinin veri tabanına bu listelerle ulaşabilmenizdir. Autocad çizimdeki elemanlarının her birine bir ad **<entity name>** verir ve bu ada bağlı bir noktalı çift listesinde DXF kodlarıyla saklar. Örneğin bir çizgi elemanını

(entget) fonksiyonu ile aldığınızda (nesne yönetimini daha ileride ele alacağız) aşağıdaki liste dönecektir.

```
((-1 . <Entity name: 7ef5efa0>) (0 . "LINE") (330 . <Entity name: 7ef5ecf8>) (5 . "EC") (100 .
"AcDbEntity") (67 . 0) (410 . "Model") (8 . "0") (100 . "AcDbLine") (10 525.388 306.622 0.0) (11
865.626 500.565 0.0) (210 0.0 0.0 1.0))
```

Noktalı Çift listelerinin içinde veri arama

(mapcar) kullanarak liste elemanlarını işleme tabi tutmak

(mapcar) fonksiyonu belirlediğiniz bir eşitliği tüm liste fonksiyonlarına uygular. Örneğin **(1 2 3 4)** diye bir listeniz var.

```
$_(mapcar '1+ '(1 2 3 4))
(2 4 5 6)
```

(mapcar) tüm liste elemanlarına **(1+)** fonksiyonunu uygulayarak tümünün değerini 1'er artırdı. Ya da yine **(mapcar)** ile: **(T nil T T (= 3 3.0))** diye bir listeyi

```
$_(mapcat 'null (T nil T T (= 3 3.0)))
(nil T nil nil nil)
```

ters çevirebilirsiniz.