İ.T.Ü. MAKİNA FAKÜLTESİ

AutoCAD'de PROGRAMLAMA

AutoLISP Uygulamaları

Hikmet KOCABAŞ

İ.T.Ü.
Gümüşsuyu-İSTANBUL
2001

İÇİNDEKİLER

AutoCAD'de Programlama İmkanları

Giriş

AutoCAD'de çizimi hızlandıran ve kolaylaştıran makrolar mevcuttur. Bunlarla çeşitli bloklardan oluşan kütüphaneler hazırlayabilir ve bunları çizimlerinizde kullanabilirsiniz.

Çizimi hızlandıran bu tip komut ve blokların yanında, rutin bazı tasarlama hesaplarının da bilgisayara yaptırma imkanları vardır. Bu sayede parametrik tasarlama ve otomatik çizim yapmak mümkün olur.

AutoCAD 'de Parametrik tasarım için iki ana yol mevcuttur.

1. Script ile çalışma: AutoCAD'in icra etmesi istenen komutların yazılı olduğu bir dosyanın (*.scr) hazırlanması ve okutulmasıdır. Bir çözüm için gerçekleştirilir. Farklı parametrelerle çalışma için kesintili bir çalışma imkanı verir
2. İnteraktif çalışan, AutoLISP veya AutoCAD Development System (ADS) ile hazırlanmış programlar: AutoCAD'de çalışırken kullanıcı ile diyaloğa girebilen ve sonuçları programdan çıkmadan verebilen programların hazırlanmasıdır. İnteraktif çalışan programlarda, kullanıcı AutoCAD'de çalışırken, hatta bir çizimin ortasında önceden programda tanımlanmış fonksiyon veya komutları çalıştırabilir. İyi hazırlanmış, interaktif programlarda AutoCAD ile kullanıcı diyalog kutuları ile etkileşime girer ve kullanıcının işi rutin işlemlerde oldukça kolaylaşır.

Script Dosyaların Yapısı ve Programlama Dilleri ile Hazırlanması

Script dosyalar, çeşitli komutların yazılı olduğu dosyalardır. AutoCAD bu dosyaları iki şekilde yükleyebilir. AutoCAD açılırken komuta script dosya ismi eklenirse AutoCAD script dosyanın içinde yazılı komutları sırasıyla icra eder ve durur. İkinci bir yol da, çizimin yarısında script komutu ile dosyanın çağırılmasıdır. Bu takdirde dosyadaki komutlar sona erdiğinde kullanıcı çalışmaya devam edebilir. Script dosyalarla çalışmada, dosya belirli bir hal için hazırlanmıştır ve eğer farklı parametrelerle çalışmak isteniyorsa dosyanın tekrar hazırlanarak çağırılması gereklidir. Bu durum kesintili bir çalışmaya sebep olacaktır.

Bu dosyaların belirli parametrelere bağlı olarakhazırlanmasını farklı programlama dillerinde yazılmış programlara yaptırılması halinde, bu işlem kolaylaşacaktır. Fakat yine de interaktif bir çalışmadan bahsedilemez.

Script dosyaların yapısı oldukça basittir. İlk satırda yeni çizim için 1 rakamı, ikinci satırda ise çizim ismi yazılır. Sonraki satırlarda AutoCAD komutları, opsiyon ve argümanları yazılır.

Misal: P1(1,1) ve P2(2,2) noktaları arasına bir çizgi çizen bir script dosyası hazırlansın.

```
1
deneme
line
1,1
2,2
```

Çizgiyi ortaya çıkaran noktaların koordinatlarını P1(x1,y1) ve P2(x2,y2) hesaplayarak dosyaya yazacak bir program yazılsın.

```
;1
;denemes
line
0,0
@0,250
@250,0
@0,-250
0,0

circle
100,100
dia
50
redraw
```

Veya, dört nokta arasına line (çizgi) çizerek dikdörtgen (rectangle) ve (circle, merkezi x,y 100,100, çapı dia 50) daire ortaya çıkaran program “deneme.scr” notepad dosyaya yazılsın.

```
script
deneme
```

Autocad içinde script komutuyla deneme.scr dosyası çalıştırılabir.

```
...
main()
{
...
FILE *dene;
...
dene=fopen(“deneme.scr”, “w+”);
fprintf(dene,”1\n”);
fprintf(dene,”deneme\n”);
/* Burada x1,x2,y1,y2 hesaplanabilir.*/
fprintf(dene,”line\n”);
fprintf(dene,”%f, %f\n%f, %f \n”,x1,y1,x2,y2);
...
fclose(dene);
}
```

Yukarıdaki, script dosyasını yükleyip çalıştıran ADS C++ programına giriş fonksiyonları eklenerek ve programın defalarca çalıştırlması ile parametrik dizayn gerçekleştirilebilir. Bu yöntem kolay gerçekleştirilebildiği için en çok kullanılan parametrik dizayn yöntemidir.

AutoLISP ve ADS Programlama

AutoLISP ile programlamada, hazırlanan (*.lsp) dosyaları ile çeşitli fonksiyonlar AutoCAD programına yüklenebilir. Parametrik dizayn imkanı veren programlar hazırlanabilir. Ayrıca bir derleyiciye ihtiyaç duymaması önemli üstünlüğüdür. Fakat çok büyük programların hazırlanmasında bazı dezavantajları mevcuttur.

ADS ile programlamada ise bir derleyiciye ihtiyaç duyulur. Derlenerek (*.exe veya *.exp) dosyaları haline getirilen dosyalar AutoCAD tarafından yüklenerek çalıştırılabilir. Çalıştırılabilir dosyaların hazırlanması biraz güç görünebilir. Fakat C++ dilinin imkanlarından faydalanılarak çok karmaşık ve büyük uygulama programlarının hazırlanması mümkün olur.

AUTOLISP PROGRAMLAMA

AutoLISP Programına Giriş

AutoLisp, **Lisp** programlama dilinin AutoCAD içine yerleştirilmiş bir uyarlamasıdır. AutoLisp dilinde programlar yazarak, AutoCAD'e komutlar ve menüler eklenebilir, makrolar (birbirine bağlanan bir dizi komutlar) ve parametrik tasarım oluşturulabilir.

AutoLisp programının bir **yazma** ve bir de AutoCAD'e yükleyip **çalıştırma** işlemi vardır. AutoLisp programını yazmak için bir kelime işlemci (Tekst editör) programı kullanılır. Bunun için AutoCAD'den çıkmadan kullanılabilen ve DOS içinde bulunan **Edit** veya bağımsız olarak çalışan **Write**, **WordPerfect** veya **Word** gibi programlar kullanılabilir. Ancak yazılan programlar AutoCAD tarafından anlaşılması için **ASCII** formatında bir dosyada saklanması gerekir. Edit tamamen ASCII kodunda bir dosya oluşturur. Word ve WordPerfect'te **DOS Text File** olarak saklanması gerekir. Saklanan dosya **.lsp** uzantılı olmalıdır. Edit gibi programlar AutoCAD'deki çizim ortamından çıkmadan **Shell** komutu ile kullanılabilir.

Yazılan bir programın AutoCAD'de **çalışması** için çalışma beleğinde (RAM) bir yer ayrılması gerekir. Belekte yer ayırma işlemi **DOS**'un SET komutu altında bulunan **lispheap**, **lispstack** ve **acadfreeram** komutları ile yapılır. Lispheap ana saklama belleği ve lispstack, AutoLisp'in **if** gibi deyimleri için kulanan bir çalışma beleğidir. Acadfreeram, RAM'da AutoCAd için ayrılan hacimden daha büyük bir hacim kullanılmasını sağlar. Genelde 640KB bilgisayarlarda, acadfreeram'ın **24**'e ayarlanması yeterlidir. Tüm bu ayarlamalar AutoCAD bilgisayara ilk defa yüklendiğinde yapılır. Bu yapılmadığı durumda şu üç yöntem önerilir; üçüncü yöntem bilgisayar tekniğiyle alışık olmayan veya AutoLisp'i çok ender kullanılan kullanıcılar içindir.

a. AUTOEXEC.BAT dosyasına şu satırlar ekleyin.

set lispheap=39000
set lispstack=5000
set acadfreeram=24

b. AutoCAD'de girildiğinde kullanılacak olan bir **batch** (*.bat) dosyasını oluşturun. Eğer AutoCAD programı **acad** adlı bir dizin (directory)'de bulunursa ilk satır olarak **cd \acad** , ondan sonra yukardaki deyimleri ve son satır olarak **acad** yazınız.

c. Ekranda **C:\>** işareti göründükten sonra **a** maddesindeki deyimleri satır satır yazın, ancak her satırdan sonra ENTER'e basın.

Yazım Kuralları

Herhangi bir programlama dilinde olduğu gibi AutoLisp programında da bir takım yazım kurallar vardır. Aşağıda bu kuralların en önemlileri verilmiştir.

- Tüm deyimler, ifadeler veya komutlar parantez ( ) içinde yazılır. Bir komutun başka bir komut üzerine etkisini yazabilmek için parantezler **iç içe** olabilir. Burada dikkat edilmesi gereken husus, sol (parantezler, sağ) parantezlerle eşit sayıda olmalıdır; yani sol parantezlerin toplamından sağ parantezlerin toplamı çıkarılırsa **0** elde edilmesi gerekir. Örneğin:

```
(xxxx (xx (xxx) (xxx) )  )
1s     2s   3s  1r 4s 2r 3r 4r
```

satırında sırasıyla s ve r ile ifade edilen 4 sol ve 4 sağ yani eşit sayıda parantez vardır. Eğer parantez sayısında bir eksik varsa, **n>** mesajı görünür; burada n kapanmayan parantez sayısını gösteren tamsayıdır. Tüm parantezlerin ayni satırda olması gerekmez; örneğin yukarıdaki 4r parantezi alt satırda da olabilir.

- İfadeler birden fazla satıra yazılabilirler. Her deyim, işaret veya sembolden sonra bir aralık bırakılır. Birden fazla aralık, tek bir aralığa eşittir.

- Genelde bir AutoLisp ifadesi

(fonksiyon argüman)

şeklindedir. **Fonksiyon**, AutoLisp'in hangi işlemin yapılacağını belirten deyimdir. Fonksiyon için AutoLisp'e ait ve AutoLisp fonksiyonları denilen deyimler kullanılır; bu fonksiyonlar açılış parantezinden sonra hemen yazılır. **Argüman**, fonksiyona veri temin eden deyimdir. Argümanlar değişken, sabit veya başka bir fonksiyon olabilir. Bir fonksiyonun bir çok argümanı olabilir veya hiç argümanı olmayabilir. Eğer fonksiyonun bir argümanı varsa, argüman için fonksiyona bir değer verilmesi gerekir.

AutoLisp'te program adı **Define function** (**Fonksiyonu tanımla**) anlamına gelen ve ilk satıra yazılan **defun** fonksiyonu ile belirtilir; ondan sonra program adı yazılır. Programın adı hemen parantezle kapatılmaz; esasen programın son parantezi, **(defun**'ı kapatan parantezdir. Örneğin bir program adı

(defun design

şeklinde olabilir. **Not:** eğer AutoCAD programı Türkçe karakterler kabul ederse program adı Türçe de yazılabilir. Program adından sonra üç seçenek vardır. Bunlardan birincisi addan sonra **()** şeklinde parantez eklenmesidir; yani **(defun design ()**. Bu yazılış şekli programda kullanmayı düşündüğünüz tüm değişkenlerin **global** sayıldığı anlamına gelir. Global değişkenler program bittikten sonra da değerini saklarlar ve bu değerler onda sonra gelen programlar için de geçerli olur. () parantezleri aşağıda açıklanacak olan yerel değişkenlerin olmadığını da gösterir. Ikinci bir seçenek, değişkene **yerel (local)** yani sadece o programda geçerli olan bir değer vermektir. Bu amaç, addan sonra **/** slaş işaretinin eklenmesi ile gerçekleştirilir; örneğin

(defun design (/ pt1 pt2)

gibi; burada **pt1** ve **pt2** fonksiyona atanan değişkenlerdir. Üçüncü bir seçenek, değişkene dışardan bir değer vermektir; bu durumda program:

(defun design (a)

şeklinde yazılır. Tabi dördüncü bir seçenek tüm bu seçenekleri birarada yazmak yani:

(defun design (a / r d)

şeklinde yazmaktır. Burada **a**, değerini dişardan alan; **r** ve **d** ise yerel değişkenlerdir. **(defun** fonksiyonunun bir başka kullanma şekli **(defun C:** dir. Bu durumda fonksiyon çağrıldığında parantez kullanılmaz; yani AutoCAD bu ifadeyi **Command:** deyimi olarak kabul eder ve buna göre işlem yapar. Başka bir deyişle, fonksiyondan sonra **C:** gelirse AutoCAD'in **Line**, **Arc** vb. komutları gibi işlem görür. Buradaki **C:**, DOS'un **C:** sürücüsü ile bir ilişkisi yoktur. Örneğin:

(defun c: design ()

ifadesi **design** adlı bir fonksiyonu tanımlar ve bu fonksiyon normal AutoCAD komutları gibi işlem görür. **()** parantezler, programda kullanılan tüm değişkenlerin global olduğunu gösterir.

Programla ilgili açıklamalar noktalı virgül **;** işareti ile başlar; örneğin **; Bir daire çizen program**. Bu durumda ifade program tarafından dikkate alınmaz ve işleme konulmaz; dolayısıyla Türkçe de yazılabilir. **;** işareti BASIC dilindeki REM ile eşdeğerdir.

- İfadelerde şu kodlar kullanılabilir;

****	**** anlamındadır.
\r	**ENTER** anlamındadır.
\"	**"** anlamındadır.
\t	**Tab** anlamındadır.
\e	**Escape** amlamındadır.
\nnn	karakterleri nnn olan **oktal** kod anlamındadır.
\n	Yeni satır anlamındadır.

AutoLisp Sisteminde Değişkenler

AutoLisp'te değişkenler: **tamsayı** (3 45), **gerçek sayı** (2.34), **dizgi** (turbin) veya **nokta** olabilir. Tam sayılar örneğin **67** girilse dahi **67.0**'a dönüştürülür ve öyle işlem görür. Ayrıca gercek sayılar **.7** şeklinde değil de **0.7** şeklinde verilmelidir. Değişkenler, yeni bir değer verilinceye veya çizim oturumu bitinceye kadar değerlerini korurlar. İlk karakter bir harf olması koşuluyla, değişkenlere istenilen ad verilebilir; ancak **pi** deyimi değeri ile eşittir. Dizgi değerleri çift tırnak (" ") içine yazılır. AutoCAD'de değişkenlere **sembol** denilmektedir. Autolisp fonksiyonlarını, AutoCAD komutlarından ayırt etmek için parantez içinde gösterilir.

Değişkenlere bir değer atamak için **(setq)** fonksiyonu kullanılır. Bunun kullanma formatı şöyledir:

(setq değişkenin-adı değer)

Burada değer: sabit, değişken veya dizgi olabilir. Örneğin:

(setq a 4) (setq x a) (setq parça "segman")

AutoCAD'de **Command (komut)**: mesajına **(setq)** ile yanıt verildiğinde, değişken verilen değere ayarlanır ve bu değer ekranda görülür. **Nokta (point)** değişkeni bir noktanın **x**,**y** ve gerektiğinde **z** koordinatlarını ifade eder. Bu koordinatlar **liste (list)** adını taşıyan ve parantez içinde yazılan **2D** için iki sayı ve **3D** için üçsayıdan oluşur; örneğin:

(4.35 6.05) (13.2 56.0 4.0)

Burada birinci sayı **x**'i, ikincisi **y**'i ve varsa üçüncüsü **z**'i temsil etmektedir. Bu koordinatlar **(list)** fonksiyonu ile örneğin

(list 4.35 6.05) (list 13.2 56.0 4.0)

şeklinde de ifade edilebilir. Buna göre bir noktanın değişkenlerine belirli değerler atamak için:

(setq pt (list 4.35 6.05)) (setq pt (list ac 6.0))

ifadeleri kullanılır. Son ifadede **ac** değişkenine **x** koordinatın değeri atanır. Ayrıca koordinatları **(1.0 2.0)** ve **(3.0 4.0)** olan **pt1** ve **pt2** noktaları ele alınırsa, bu noktaların **x** koordinatı **(car)**, **y** koordinatı **(cadr)** ve **z** koordinatı **(caddr)** fonksiyonları ile, başka bir noktanın kordinatları olarak atanabilir. Örneğin bir dikdörtgenin sol-alt ve sağ-üst noktalarını ifade eden ve koordinatları **(1.0 2.0)** ve **(3.0 4.0)** olan **pt1** ve **pt2** noktaları ele alınırsa, dikdörtgenin sol-üst yani **pt3** noktasının koordinatları, **pt1**'in **x** ve **pt2**'in **y** koordinatı olacaktır. Bu husus:

(setq pt3 (list (car pt1) (cadr pt2)))

şeklinde ifade edilir. Bu ifade ile **pt3**'ün koordinatları **(1.0 4.0)** olacaktır. AutoCAD'in bir mesajına bir değişkenin değerini vermek istenildiğinde, ünlem ! işaretinden sonra değişkenin adı yazılır. Örneğin bir çizginin başlangıç noktası olarak yukarıda ifade edilen **pt1**'in koordinatları verilmek istenirse, işlem:

Command (komut): Line (Çizgi)
From point (Başlangıç noktası): !pt1

şeklinde gerçekleştirilir.

Değişkene **0** dahil hiç bir değer verilmediği durumda **nil** olarak ifade edilir; burada **0** bir sayı sayılır. Buna göre hiç bir değeri olmayan **a** değişkeni, **!a** olarak girildiğinde nil mesajı görünür.

Aritmetik İfadeler

AutoLisp ifadeleri ile çeşitli aritmetik, trigonometrik ve geometrik işlemler yapılabilir. Bu ifadelerde tüm değerler tam sayı olarak verilirse sonuç tam sayı; değerlerden biri gerçek sayı ise sonuç gerçek sayı olur. Aritmetik fonksiyonların ifade şekli şöyledir:

(+ a b) **a** ve **b** değerlerini toplar.
(- a b) **b** değerini **a** değerinden çıkarır.
(* a b) **a** ve **b** değerlerini çarpar.
(/ a b) **a** değerini **b** değerine böler.
(max a b) **a** ve **b** değerlerinden en büyük olanını seçer.
(min a b) **a** ve **b** değerinden en küçük olanını seçer.

Esasen yukarıdaki ifadeler ikiden fazla değişken örneğin:

(+ 23 4 15) (- 234 2 16) (* 3 6 2) (/ 142 4 5)

şeklinde içerebilir ve sırasıyla 32; 216; 34 ve 7.1 sonuçları verir. Bunların yanısıra şu fonksiyonlar da vardır:

(abs a) **a**'nın mutlak değerini verir.
(sqrt a) **a**'nın kare kökünü verir.
(expt a p) **a**'nın **p** kuvvetini verir.
(exp p) **e**'nin **p** kuvvetini verir.
(log a) **a**'nın doğal logaritmasını verir.
(float a) tamsayı **a**'nın gerçek sayı halini verir.
(fix a) **a** gerçek sayısının tam sayı kısmını verir.

Trigonometrik fonksiyonlar şöyledir:

(sin açı)	**açı**'nın sinüsünü verir; açı radyan cinsindedir.
(cos açı)	**açı**'nın cosinüsünü verir; açı radyan cinsindedir.
(atan a)	**a**'nın arktanjantını verir (radyan cinsinden).
(1+ x)	**1** ve **x**'in toplamını verir; (+ x 1) gibi.
(1- x)	**1** ve **x**'in farkını verir.
(angle p1 p2)	**p1** ve **p2** noktaları arasında açıyı verir.
(distance p1 p2)	**p1** ve **p2** noktaları arasında uzaklığı verir.
(polar p1 ang d)	**p1** noktasından **d** uzaklıkta **ang** açısını (radyan cinsinden) yapan noktayı verir.
(type a)	**a**'nın tipini (tamsayı, gerçel, dizi, liste) verir.

Yukarıdaki fonksiyonlar böyle yazıldığı durumda, hesapları yapıp sadece sonucu verirler. Ancak bir değişken üzerinde yapılan bir işlemin sonucu yine değişkene atanmak istenirse şu ifade kullanılır:

```
(setq x (- x 2))
```

Burada **x** değişkeninden **2** çıkarılır ve sonuç tekrar **x**'e atanır.

Uygulama.

Bu uygulamada derece cinsiden açıları radyana çeviren bir program yazılacaktır. Programın yazılması için örnek olarak **DOS**'un **Edit** Tekst editörü kullanılacaktır. Bilindiği gibi derece cinsinden bir açı **a** ile ifade edilirse bunun radyan olarak değeri **a.pi/180** bağıntısından bulunur.

Çizim ortamına girin. **Command (komut):** mesajına **shell** ve sonra **edit** deyip editöre girin. Program alfanumerik ekrana geçer. Burada menüyü aktif duruma geçirmek için **Esc** ve **Alt** tuşuna, **File** menüsüne girmek için **Enter** veya **aşağı-yukarı ok** tuşlarına basın. Burada **New** (Yeni) üzerindeyken **Enter** tuşuna basılarak yeni bir dosya sayfası başlatılır. Burada şu programı yazın:

```
; açıları dereceden radyana çevirir
(defun dtr (a)
        (* pi (/ a 180.0))
)
```

Program bittikten sonra, menüyü yeniden aktif duruma geçirmek için **Alt** tuşuna basın ve **File** menüsüne girmek için **Enter** veya **aşağı-yukarı ok** tuşlarına basın. Burada **Save** (Sakla) üzerindeyken **Enter** tuşuna basılarak dosya saklanır. Yine aynı şekilde menüden **eXit** ile çıkılarak AutoCAD ortamına geri dönülür.

AutoLisp Programların Yüklenmesi ve Çalıştırılması

AutoLisp programları AutoCAD'de: **dolaylı**, **direkt** ve **Acad.lsp** dosyasını kullanarak yüklenir ve çalıştırılabilirler.

a. Dolaylı yöntem bir Tekst Editörü ile yazılmış programlara uygulanır. Burada önce program yüklenir ondan sonra çalıştırılır. Programın yüklenmesi:

(load "dosya adı") örneğin **(load "dtr")**

ifadesi ile gerçekleşir. Buna yanıt olarak mesaj kısmında programın yüklendiğini gösteren ve programın adı olan **dtr** mesajı görünür. Programın çalıştırılması için

Command: mesajına **(dtr 20)** girilir. Dosya adı çift tırnak arasında **(" ")** ve uzantısız yazılır. Program adı parantez içinde yazılır.

b. Direkt yöntemde herhangi bir AutoLisp komutu, **Command (komut):** mesajına parantez içinde yazılarak girilebilir. Örneğin:

Command: (setq a 4) veya **(setq parça "segman")**

girilirse, mesaj kısmında **4** veya **"parça"** görünür. Aynı şekilde **(/ 180 2 4)** girilirse, bu hesabın sonucu olan **22** görünür. Bu şekide değişkene bir değer atadıktan sonra, **Command:** mesajına değişken, önünde **!** işareti ile yazılırsa, değişkenin değeri ekranda görülür. Örneğin **!a** girilirse **4** görünür; **!parça** girilirse **segman** görünür. Bir değişkene bir hesap sonucu da atanabilir. Örneğin **(setq b (/ 180 2 4))** girilirse sonuç **22** çıkar, ancak daha sonra **!b** girilirse yine **22** görülür.

c. Yazılan program **Acad.lsp** dosya sisteminde **Acad.lsp** adlı bir dosyada saklanır ve her AutoCAD'in yüklenişinde bu dosya da otomatikman yüklenir. Burada programın başına, bellekte gerekirse bir genişletme meydana getiren **(vmon)** yazılmasında yarar vardır; Örneğin radyan birimini dereceye çeviren bir program:

```
; Radyan birimini dereceye çeviren program
 (defun rtd (a)
        (/ (* a 180.0) pi)
)
```

şeklinde yazılır ve **Acad.lsp** bir dosyaya saklanırsa, bu program AutoCAD programı ile beraber yüklenir. AutoLisp'te açılar radyan; AutoCAD de ise derece olarak işlem görür. Bu nedenle bir birimi diğerine çevirmek için yukarıdaki uygulamada verilen **(dtr)** ve burada verilen **(rtd)** programlarının **Acad.lsp** dosyasında saklanmasında yarar vardır.

AutoLisp Fonksiyonları

AutoLisp'in komut da denilen bir çok fonksiyonları vardır. Bunların bazıları, örneğin **setvar** gibi hem Autolisp'e hem AutoCAD'e aittir. Autolisp için bu komut **(setvar)** şeklinde yazılır. Bu bölümde sadece çok kullanılan bazı komutlar verilecektir. Diğer komutlar hakkında bilgi için **AutoLisp Programcı Referans Kitabı** bölümüne bakılması önerilir. Yukarıda AutoLisp'e ait **(defun)** ve **(setq)** fonksiyonlar hakkında bilgi verilmiştir; bunlar burada tekrarlanmayacaktır.

a. (prompt); ekranın mesaj kısmına yazı yazmak için kullanılır. İfade formatı şu şekildedir:

(prompt "yazı") örneğin **(prompt "Bir nokta seç")**

(prompt) fonksiyonu ile ayni işlem gören yani ekrana yazı yaıan **(princ)**, **(prin1)** ve **(print)** fonksiyonları vardır. **(princ)** ve **(prin1)**, yazıları tırnak içine alırlar; **(print)** ise yazıdan sonra boşluk bırakmaktadır.

b. (getpoint); bir nokta seçilmesini ister. Nokta seçilinceye kadar ekranda hiç bir şey olmaz. Ancak Ekranda bir nokta gösterildiğinde, noktanın koordinatları mesaj bölgesinde görülür. Komut **(setq)** ile kullandığında, seçilen noktanın koordinatları **(setq)** ile belirlenen değişkene atanır; ayrıca ifadeye bir de mesaj eklenebilir. Örneğin:

(setq a (getpoint "Birinci noktayı seç"))

ifadede **Birinci noktayı seç** mesajı komut bölgesinde görülür ve nokta seçildikten sonra, noktanın koordinatları **a** değişkenine atanır. Bu şekilde noktanın koordinatları **a** değişkeninde saklanmış olur. Şöyleki **Command:** mesajına **!a** ile yanıt verilirse, noktanın koordinatları mesaj bölgesinde görünür.

c. (terpi); ondan sonra gelen mesajların, ondan sonraki satıra yazılmasını sağlar. Fonksiyon **\n** kodu ile eşit anlamındadır.

d. (command); tüm AutoCAD komutlarının kullanma imkanını sağlar; ayrıca takip eden değişkenler komuta aittir. Örneğin yukarıda **b** maddesindeki ifade ile Birinci noktanın koordinatları **a** değişkeninde saklanmış olsun. ikinci noktanın koordinatları aşağıdaki ifadeyle:

(setq b (getpoint "ikinci noktayi seç"))

b değişkenine saklansın. Bu durumda:

(command "line" a b)

ifadesi ile yukardaki mesajlarla koordinatları belli olan **a** ve **b** noktaları arasında bir çizgi çizilir. Çizgi çizildikten sonra **line (çizgi)** komutu geçerli kalır. Bu komutu bitirmek için son noktadan sonra, örneğin **(command "line" a b "")** şeklinde iki çift tırnak yazılır. Çizgi sadece iki nokta arasında değil, koordinaları bilinen bir çok nokta arasında çizilebilir; ayrıca **"c"** ile çizgi kapatılabilir. Örneğin:

(command "line" a b c d "c")

ifadeyle dört kenarlı bir çizim oluşturulur.

Uygulama. Aşağıda iki nokta arasında bir çizgi çizen bir program verilmiştir.

```
; İki nokta arasında çizgi çizen program
 (defun drawline (/ pt1 pt2)
        (setq pt1 (getpoint "\nBirinci noktayi seçin"))
        (setq pt2 (getpoint "\nikinci noktayi seçin"))
        (command "line" pt1 pt2 "")
)
```

Programı herhangi bir Tekst Editörü ile yazın, **uyg1.lsp** adlı bir dosyada saklayın, yukarıda gösterildiği gibi AutoCAD'e yükleyin, çalıştırın, **birinci noktayı seçin** mesajına bir nokta gösterin, daha sonra **ikinci noktayı seçin** mesajına bir başka noktayı gösterin; ekranda çizgi çizilir.

e. (getcorner); (getpoint) fonksiyonuna benzemekle beraber esasen seçilen iki nokta arasında bir dikdörtgen oluşturur. Burada ilk nokta daha önce seçilir veya bir değişkene atanır ve ancak ikinci noktanın seçiminde **(getcorner)** kullanılır; bu durumda kursor ikinci noktaya hareket ettirildiğinde dikdörtgen lastik band şeklinde görünür. İfadeler şöyledir:

```
(setq pt1 (getpoint "Bir nokta seç"))
(setq pt2 (getcorner pt1 "baska bir nokta seç"))
```

f. Çevirme fonksiyonları. (angtos); radyan cinsinden olan bir açıyı, mevcut durumda olan açı birim sistemine çevirir. Burada dikkat edileçek husus **(angtos)**'ın değeri olmayan bir dizgi olmasıdır. ifade şöyledir:

(prompt (angtos b)) veya **(setq a (angtos b))**

ve mevcut birim sistemi **degrees**, **minutes**, **seconds** ise örneğin **23d16'25"** şeklinde bir değer görünür. Başka birimlere göre ifade edilmek istenirse AutoCAD'in:

0. degrees 1. degrees, minutes, seconds
2. grads 3. radians 4. survey units

şeklinde gösterilen ve **flag** adını taşıyan numaralarından faydalanılır. Burada ondalık sayısını gösteren bir ikinci sayı daha eklenir. Örneğin:

(prompt (angtos ang1 0 3))

ifadesi örneğin **46.234** şeklinde bir açı değerini gösterir.

(rtos); (angtos) fonksiyonunun benzeri olup uzunluk birimlerini istenilen birim sistemine çevirir ve sonucu **dizgi** olarak verir. Daha öncede belirtildiği gibi AutoCAD de çizimler **çizim birimleri (unit)** ile çizilir ve bunlar **metrik** veya **inç** sistemine göre ifade edilebilir. Örneğin belirli bir **a** değişkeninin değeri **45** ve mevcut birim **mm** ise **(prompt (rtos a))** ile **45 mm** bir çıkış görünür. Burada da AutoCAD'in.

1. scientific 2. decimal 3. engineering
4. architectural 5. fractional

şeklinde gösterilen birim sisteminin numaralarından yararlanarak:

(prompt (rtos d 1 3))

ifadesi ile uzunluk **bilimsel (scientific)** sisteme göre ifade edilebilir. Burada da ikinci sayı, ondalık sayısını gösterir.

g. Giriş fonksiyonları. (getreal); bir **a** değişkenine klavyeden gerçek sayı bir değer atanmasını sağlar. İfade:

(setq a (getreal "Bir sayi gir"))

olup, **Bir sayı gir** mesajı göründükten sonra sayı girilir.

(getdist); bir uzaklığı sayı olarak girmek veya ekranda iki nokta göstererek, verme imkanını sağlar. İfade şöyledir:

(setq a (getdist "Yükseklik gir"))

(getstring); bir yazı girilmesini sağlar. İfade:

(setq a (getstring T "Yazi gir"))

şeklindedir. Burada **Yazı gir** mesajına bir yazı girilir ve daha sonra **!a** ile yazı tekrar mesaj bölümünde görülür. Buradaki **T** , dizgi içinde boşluk kullanılabileceği anlamına gelir. **T** yazılmadığı zaman girilen yazı içinde boşluk kullanılamaz.

h. Geometrik fonksiyonlar. (angle); iki nokta arasındaki açı değerini ölçer. Burada önce noktalar belirlenir ve sonra açı ölçme ifadesi yazılır. İfadeler şöyledir:

```
(setq pnt1 (getpoint "Bir nokta seç"))
(setq pnt2 (getpoint "ikinci noktayi seç"))
(setq a (angle pnt1 pnt2))
(dtr a)
```

Burada **pnt1** ile **pnt2** noktalar arasındaki açı radyan olarak **a** değişkenine atanır; son ifadede, **(dtr)** programı yüklü ise, açıyı radyandan dereceye çevirir. Açının değerini görmek için **!a** girilir.

(distance); iki nokta arasında uzaklığı ölçer; burada da ilkin noktaların belirlenmesi gerekir. Uzaklık ölçme ifadesi şöyledir:

```
(setq d (distance pnt1 pnt2))
```

(polar); bir noktanın polar olarak, yani uzunluk ve açıya bağlı olarak koordinatlarının verilmesini sağlar. İfade şöyledir:

```
(setq a (polar pnt1 ang1 dst1))
```

(getangle), (getorient); her iki fonksiyon iki nokta göstererek bir açının bulunmasını sağlarlar. İfade:

```
(setq a (getangle "Açiyi seç"))
```

şeklindedir. **(getorient)** her zaman **0** derece olarak **East (Doğu)**; **(getangle)** ise **0** olarak güncel yönünü kabul etmektedir. Örneğin **0**'ın güncel yönü **North (Kuzey)** ise bunu alır. Bu fonksiyonda açılar klavyeden de girilebilir. Ancak açı derece olarak belirlenmişse, bu otomatik şekilde radyan olarak saklanır.

i. (setvar); AutoCAD'in **Setvar** komutu içinde bulunan sistem değişikliklerini ayarlar. İfade şöyledir:

```
(setvar "orthomode" 1)
```

Kenetleme oluşturmak için:

```
(setvar "osmode" n)
```

ifadesi kullanılır. Burada **n** kenetleme şekline bağlı olan ve aşağıda verilen bir sayıdır.

Center 4	**Endpoint 1**	**İnsert 64**	**Intersection 32**
Midpoint 2	**Nearest 512**	**Node 8**	**Perpend 128**
Quadrant 16	**Tangent 256**	**None 0**	

(graphscr) fonksiyonu, tek ekranlı sistemlerde alfanümerik ekrandan grafik ekrana geçilmesini sağlar.

j. Nesneleri seçin fonksiyonları. **(ssget)**, **(entsel)** fonksiyonları nesnelerin seçmesini sağlar. **(entsel)** bir defada bir tek nesne seçer; **(ssget)**, **Window (pencere)** ve **Crossing (kesişme)** dahil olmak üzere çeşitli seçme yöntemlerini kullanarak bir defada bir nesne kümesi seçebilir. Pencere için ifade:

(setq a (ssget "W" pnt1 pnt2))

şeklindedir; kesişme kullanıldığı durumda **W** yerine **C** yazılır. Bu şekilde **a** değişkenine atanan ve adı **<Selection set: 1>** olan bir seçim kümesi meydana getirilmiş olur. Bundan sonra bu kümeye nesne ile ilgili veriler dahil edilebilir. Bilindiği gibi AutoCAD'de her nesnenin **entity type (nesne tipi)**, **layer (tabaka)**, **color (renk)** vb. gibi oldukça geniş bir veri tabanı vardır. Bu veri tabanlarına birer kod verilmiştir şöyleki;

(ssget "x" ' ((0. "text")))

ifadesiyle seçilen nesnelerle ilgili tüm yazı veri tabanı **<Selection set: 1>** seçim kümesine dahil edilmiş olur. Nesnelerle ilgili veri tabanı kodları şu şekilde tertiplenmiştir.

Entity type 0	**Block 2**	**Line type 6**	**Text style 7**	**Color 62**
Elevation 38	**Layer 8**	**Thickness 39**	**Attributes 66**	

(sslength); seçim kümesindeki nesne sayısını verir. İfade: **(setq n (sslength a))** şeklindedir.

(ssname); seçim kümesindeki nesnenin adını verir; ancak bu ad örneğin **70000012** şeklinde, onaltılı (hexadecimal) sayı sistemine göre verilir. İfade: (**setq na (ssname a i))** şeklindedir; burada **a** seçim kümesinin atandığı değişken; **i** nesnenin indis sayısıdır. Bir seçim kümesinde bir çok nesne olabilir; bu nesneler, seçim kümesi içindeki konumunu gösteren sayılarla indekslenebilir. Örneğin **#1** nesne **0**, **#2** nesne **1** vb. şeklinde indekslenebilir.

(enget); nesneleri listeler. ifade **na** nesnenin adı olmak üzere**: (setq b (entget na))** şeklindedir; **!b** girerek nesneler kod numaralarına göre listelenir.

(assoc); bir nesnenin belirli bir özelliğini listelemek için kullanılır. Bu özelliğin değeri daha sonra **(subst)** ile değiştirilebilir.

(const); yeni nesne başa yazılmak üzere yeni bir liste meydana getirir. **(entmod)**, **(const)** ile oluşturulan listeyi kalıcı yapar.

7. Lojik İfadeler

İf (Koşul) olarak da bilinen bu ifadeler **if-then-else** üçlüsüne dayanmaktadır. AutoLisp'te bir **if**'e bir **then** karşılık gelmektedir. ifadenin formatı şu şeklindedir:

(if (xxx1) (xxx2) (xxx3))

Burada **if**'ten sonra **xxx1** kontrol edilmesi gereken deyimdir; örneğin **(= a b)** şeklinde ifade edilen **a=b** eşitliği gibi. Eğer bu deyim doğru ise o zaman **(then) xxx2** deyimi işlem görür; değilse **(else) xxx3** deyimi işlem görür. Örneğin:

```
(setq a 6)
(setq b 9)
(if (= a b) (setq b "esit") (setq b "esit degil"))
```

programı yazılıp işleme konulursa, **!b** girildiğinde **"eşit değil"** sonucu görünür. if ifadesine **(progn)** fonksiyonu kullanılarak bir çok **then (o zaman)** atanabilir. Bu fonksiyon **xxx1**'den

sonra parantezsiz yazılır; eğer if ifadesi sona ererse **(progn)** parantez içine yazılır. **(= a b)** 'in yanısıra, şu lojik fonksiyonlar vardır.

(> a b)	**a b**'den daha büyük
(< a b)	**a b**'den daha küçük
(/= a b)	**a b**'ye eşit değil
(<= a b)	**a b**'den küçük veya eşit
(>= a b)	**a b**'den büyük veya eşit
(and a b)	**a** ve **b**; lojik bağlaç **ve**
(or a b)	**a** veya **b**; lojik bağlaç **veya**
(not ..)	lojik **hayır**
(listp a)	**a** bir liste ise
(null a)	**a nil** ise

Örneğin: **(if (> (getvar "FILLETRAD") 0.25) (setvar "GRİDMODE" 0))**
ifadesi, eğer getvar fonksiyonunun yuvarlatma yarıçapı (Filletrad) 0.25 ten daha büyükse, ızgara modunu (Gridmode) 0'a eşitle anlamına gelir. Burada aksine yani yuvarlatma çapı 0.25'ten daha küçük olduğu durumda ne yapılacağını belirtilmediği için, hiç bir işlem yapılmaz.

Uygulama

Aşağıda küçük çapta iki program verilmiştir; bunları herhangi bir text editörü ile yazıp **Acad.lsp** veya **.lsp** uzantılı bir dosyaya saklayın ve daha sonra çizim sırasında kullanın. Programlar **defun c:** ile meydana getirilmiştir; dolayısıyla program adları AutoCAD'in birer komutu olmuşlardır. Buna göre program yükledikten ve örneğin birinci programda, programın yüklendiğini gösteren c: ekransil göründükten sonra sadece **ekransil** yazılması yeterlidir.

a. Tüm ekranı silme programı.

```
(defun c: ekransil (/ v u)
        (setq v (getvar "limmin"))
        (setq u (getvar "limmax"))
        (command "erase" "w" v u "")
)
```

b. Çizgi tipi ölçeği (Line type scale).

```
(defun c: lscale (/ v u d sc)
        (setq v (getvar "limmin"))
        (setq u (getvar "limmax"))
        (setq d ( - (car u) (car v)
        (setq sc (/ d x))
        (command "ltscale" sc)
)
```

Burada **x** yerine istenilen ölçek yazılabilir.

PROGRAMLANABİLİR DİYALOG KUTULARI

AutoCAD'in kullandıklarına benzer şekilde, kendinize özel diyalog kutularınızı tasarlayabilir ve kullanabilirsiniz. Bu bölümde bunların nasıl yapılacağı anlatılacak. Özel diyalog kutularının oluşturulmasında iki ana işlem vardır :

- **Diyalog kutusunu tasarlamak**

Diyalog kutuları *Diyalog Kontrol Lisanı*'nda (Dialogue Control Language : **DCL**) yazılmış ASCII dosyalar ile tanımlanır. Bir diyalog kutusunun DCL tanımı, kutunun nasıl görüneceği ve düğmeler (**button**), listeler (**list**), yazı (**text**), ve benzeri elemanlardan hangilerini kullanacağını belirler. Diyalog kutularının büyüklüğü ve elemanlarının konumları, çok az bir bilgi ile otomatik olarak belirlenmektedir. Her bir elemanın konumunun kesin bir şekilde belirlenmesi zorunluluğu yoktur.

- **Uygulamanızda diyalog kutusunun desteklenmesi**

Diyalog kutusunun elemanları, onun nasıl davranacağını bir yere kadar belirler. Mesela, düğmeler (button) üzerine basılması, listeler (list) kullanıcının bir seçim yapabilmesi için kullanılır. Diyalog kutularının kullanımı ve davranışı, tamamen onu kullanan uygulamaya bağlıdır. AutoLISP ve AutoCAD Development Sistem (**ADS**), diyalog kutusu fonksiyonlarına (kutuların gösterimi, kullanıcının seçimlerine cevap verme gibi) sahiptir.

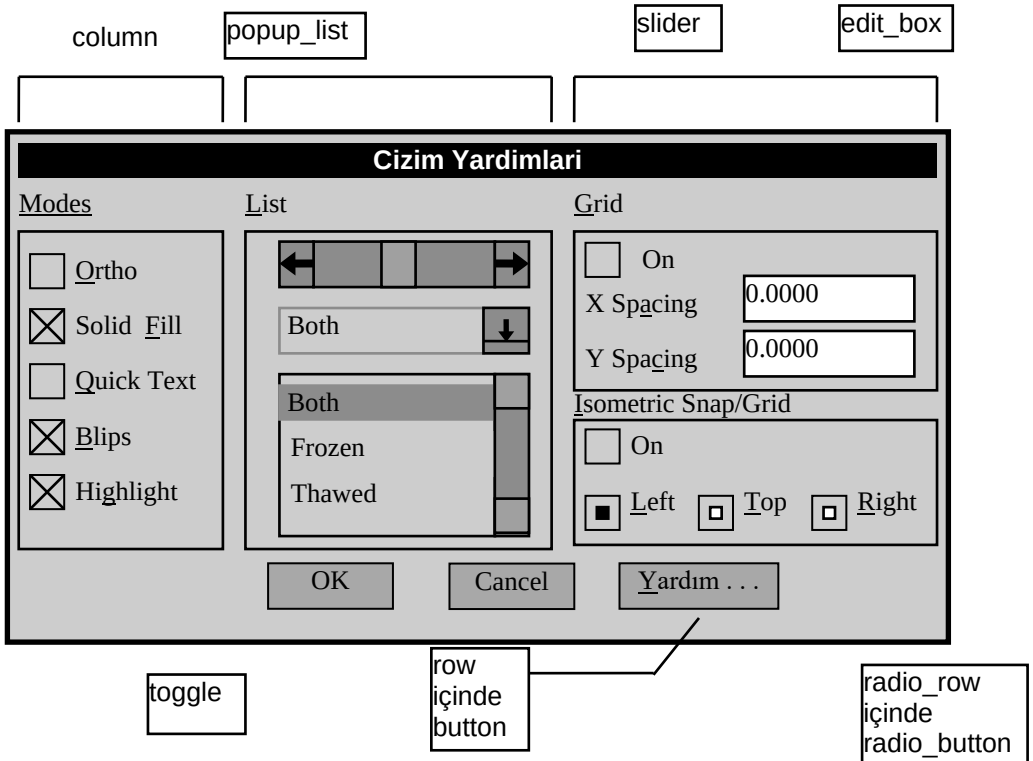
Note: **Programlanabilir Diyalog kutuları (PDB)** tamamen etkileşimli kullanım içindir ve bir kullanıcıyı gerektirir. Bir **script**, bir diyalog kutusunu başlatabilir, fakat açıldıktan sonra onu kontrol edemez, veya herhangi bir değer girişi yapamaz. Bu **AutoLISP (command)** ve **ADS ads_command ( )** ve **ads_cmd ( )** fonksiyonları içinde geçerlidir.

AutoCAD'in Diyalog kutuları desteği, çalışılan platformdan bağımsızdır. Tek bir DCL ifadesi ve uygulaması, bütün platformlarda kutunun şeklini ve işlevini belirler. Bununla birlikte kutunun görünüşü, her platformda farklılıklar gösterebilir.

Diyalog kutularını kullanmanın bir sebebi uygulamanıza içinde çalışılan ortama benzer bir görünüm kazandırmaktır. Diğer bir sebebi de, bir diyalog kutusunun kullanımının, arka arkaya gelen mesajlara cevap vermekten daha kolay, tabii ve hızlı olmasıdır. Eğer diyalog kutunuzu ve uygulamanızı, program kodunu yazmaya başlamadan önce detaylı bir şekilde planlarsanız, daha az zaman kaybedersiniz.

Diyalog Kutularının Kullanımına Genel Bakış

Aşağıdaki resim, bazı bölümleri etiketlenmiş (label) standart AutoCAD diyalog kutularından birini göstermektedir. Diyalog kutu oluşturma ve kullanımında, bu bölümler **tile** (döşeme) olarak bilinir.



Şekil 1. Bir diyalog kutusundaki tipik bölümler

Klavye Kısatuşları

Bazı platformlar, kullanıcının diyalog kutularını fare ile olduğu gibi klavyeden de **Tab**, **Esc**, **Ctrl+C**, **Enter** vb. gibi tuşları kullanarak kontrol edebilmesine izin verir. **Tab** tuşu diyalog kutusundaki bölümler (**tile)** arasında geçiş yapabilmeyi sağlar. Kabul tuşu (**Enter** veya **Return**), diyalog kutusunun kabul edilmesini, iptal tuşu (**Esc** veya **Ctrl+C**), diyalog kutusunun iptalini sağlar. Bunun dışında **mnemonic** kısatuşlarda, bölümler (tile) arasında geçiş yapabilmek için kullanılabilir. Üzerinde bulunulan bir bölümü kabul etmek için, kabul (**enter**) tuşuna basılması veya fare sol tuşu ile iki defa (double click) seçilmelidir. Kendi diyalog kutunuzu oluşturduğunuzda, aktif olan bölümü (tile) klavye ile belirlemeye yarıyan kendi **mnemonic** kısatuşlarınızı belirleyebilirsiniz.

AutoCAD Menüsünden Bir Diyalog Kutusunun Çağırılması

AutoCAD menüsünden programlanabilir bir diyalog kutusunu başlatmak için sadece bu diyalog kutusunu kullanan AutoLISP veya ADS fonksiyonunu yazınız. Bunu yapmanın en kolay yolu, fonksiyonu bir AutoCAD komutu (**C:XXX**) olarak yazmaktır. Örnek olarak, **acad.mnu** dosyasındaki **File** menüsünün başlangıcı aşağıda gösterilmiştir. Buradaki her komut, ilgili diyalog kutusunu çağırmaktadır.

```
***POP1
[File]
[New... ] ^C^C_new
[Open... ] ^C^C_open
[Save... ] ^C^C_qsave
[Save  As ... ] ^C^C_saveas
[Recover... ] ^C^C_recover
```

Diyalog kutularının renklerini değiştirmek için AutoCAD'in **DLGCOLOR** komutu kullanılabilir.

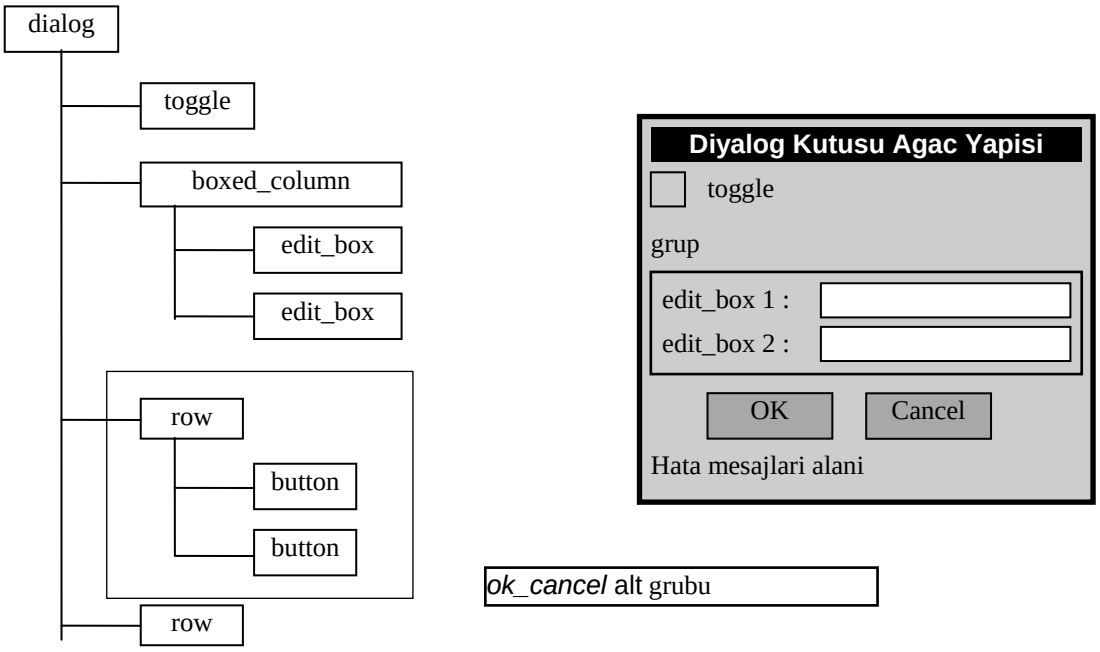
Diyalog Kutusu Elemanları--Tile ve Tile Prototipleri

Bir diyalog kutusu, bir kutu ve içindeki bölümlerden (tile) meydana gelir. Düğmeler (**button**) yazı kutuları (**edit box**), listeler (**list**), resimler (**image**), vb. en temel Tile tipleri, **PDB** 'de önceden tanımlanmıştır. Bunların özellikleri ve kullanımları bu bölümde anlatılmıştır. Bölümlerin (tile), kutulu veya kutusuz (**box**) satırlar (**row**) ve sütunlar (**column**) şeklinde düzenlenmesiyle, daha karmaşık bölümler oluşturulabilir. Bu birleşik-bölümler de tek bir bölüm (tile) gibi kullanılabilirler, ve böylece diyalog kutuları kendi içinde hiyerarşik ve ağaç yapısında düzenlenebilirler. Ağacın en üstünde DCL'de **dialog** olarak bilinen diyalog kutusunun kendisi vardır.

Bir bölümün (tile) veya birleşik-bölümün yerleşimi, görünümü ve davranışı, DCL'de bölümün özelliği (**attribute**) ile tanımlanır. Örneğin **dialog**'un kendisi ve önceden tanımlanmış bölümler (tile), bölümün yanında görünecek olan yazıyı belirleyen bir **label** (etiket) özelliğine sahiptir. Bir **dialog**'un etiketi (label) diyalog kutusunun en üstündeki yazı, bir düğmenin (**button**) etiketi (label) düğmenin içindeki yazıdır.

DCL, herhangi bir diyalog kutusuyla ilişkili olmayan yeni bölümler (prototipler) ve bölüm-grupları (birleşik-bölümler) tanımlamaya da izin verir. Bu prototipleri önceden tanımlanmış bölümlerdeki gibi referans gösterebilir ve özelliklerini gerektiği şekilde değiştirebilirsiniz.

Aşağıdaki şekil örnek bir diyalog kutusunun ağaç yapısını göstermektedir. Ağacın yaprakları, önceden tanımlanmış bölümler (tile) ve ağacın en tepesi ve kökü daima bir **dialog**'dur. DCL'de oluşturacağınız bir diyalog kutusu tanımı bu ağaç yapısını gösterecektir.

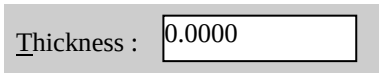


Şekil 2. Bir diyalog kutusunun ağaç yapısı

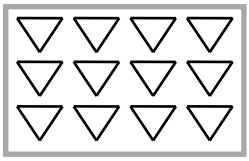
Önceden Belirlenmiş Aktif Bölümler

Önceden Belirlenmiş Aktif Bölümler, AutoCAD PDB tarafından doğrudan desteklenmektedir. Bunların tanımları **base.dcl** dosyası içindeki ifadelerde görülebilir. Kullanıcı bir aktif bölüme (tile) dokunduğunda (örneğin bir düğmeye bastığında), diyalog kutusu ilgili uygulamayı çalıştırarak cevap verir. Bu bir **action** (işlem) veya **callback** (cevap) olarak bilinir. **Action**'lar, işleme neyin sebep olduğunu gösteren bir **reason** (sebep) kodu saklarlar. Bu sebebin manası, onu oluşturan bölüm (tile) tipine göre değişir.

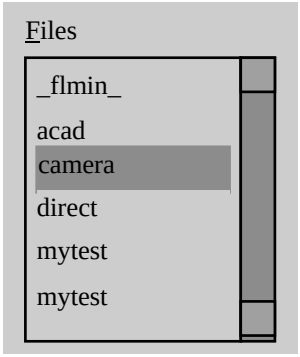
button
Bir düğme belirler. Bir düğme etiketi (*label*), düğmenin içinde görünen yazıyı belirler. Düğmeler, diyalog kutusunun terkedilmesi veya bir alt diyalog kutusuna girmek gibi kullanıcının hemen cevap beklediği işlemlere uygundur. Herbir diyalog kutusunda en azından bir **OK** (tamam) ve **Cancel** (iptal) düğmesi bulunur.



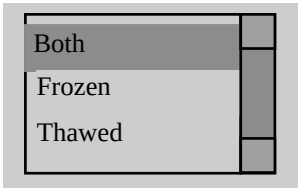
edit_box
Kullanıcının tek satırlık bir yazı girmek ve değiştirmek için kullanabileceği alandır. Eğer bir *label* tanımlanırsa bu kutunun solunda görülür. Yazı bir satırı geçerse aşağı doğru kayabilir.



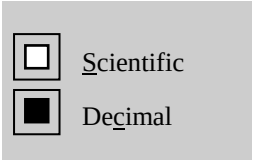
image_button
Bu bölüm bir grafik resim gösteren bir düğmedir.



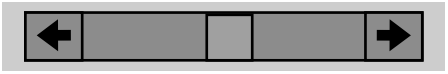
list_box
Liste kutusu, satırlar halinde düzenlenmiş yazı dizgileri listesi içeren bir kutudur. Bu, kullanıcının içinden seçim yapabileceği bir listeyi gösterir. Eğer liste ekranda ayrılan alana sığmazsa, sağ tarafta bir kaydırma sütunu (*scroll bar*) ortaya çıkar. Uygulamaya bağlı olarak listeden birden fazla satırın seçilebilmesi de mümkündür.



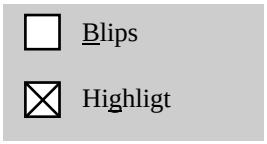
popup_list
Liste kutusuyla aynı işi görür. Tek fark *popup_list* aktif olmadığı zaman sadece seçilmiş satır görülür. Fare ile dokunulduğunda tüm liste ortaya çıkar.



radio_button
radio_column (sütun) veya *radio_row* (satır) olarak düzenlenmiş bir grup düğmedir. Çalışması otomobil radyolarındaki düğmelere benzer. Bir sütun veya satırdaki düğmelerden sadece bir tanesi açık (**on**) olabilir. Bir başka düğmeye basıldığında açık (on) olan düğme kapanır (**off**). Eğer *label* tanımlanırsa, düğmenin hemen sağında görünür.



slider
Sayısal bir değer belirleme aracıdır. *slider*'ın göstergesi sağa sola kaydırılırsa, uygulamaya bağlı olarak maksimum -32768 ile 32767 arasında bir değer dizgi (string) olarak elde edilir. Bu değer gerekirse ölçeklendirerek kullanılabilir.

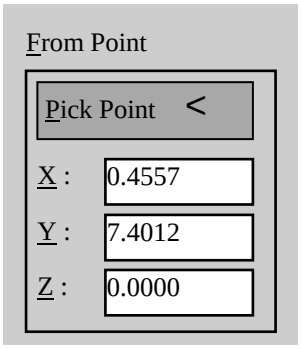


toggle
"0" (off) veya "1" (on) değeri arasında seçim yapmayı sağlar. Sağ tarafında görülen etiketiyle birlikte küçük bir kutudur. Seçim yapıldığında **X** veya **V** seçim işareti görülür (on) veya kaybolur (off).

Bölüm Grupları

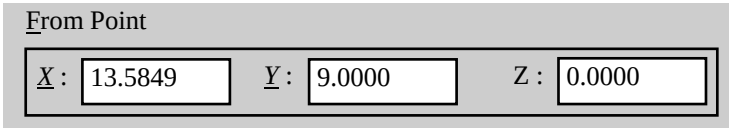
Bölümler karmaşık satırlar ve sütunlar (gruplar) şeklinde düzenlenebilirler. Bu satır (row) ve sütunlar (column) tek bir bölüm gibi davranırlar. Satır ve sütun grupları (boxed_) kutu içine alınabilir ve etiketlenebilirler.

column içindeki bölümler düşey olarak düzenlenir.

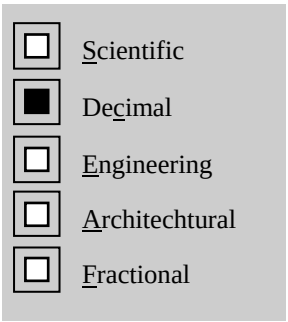


boxed_column bölümleri düşey olarak düzenler ve bir kutu ile sarar.

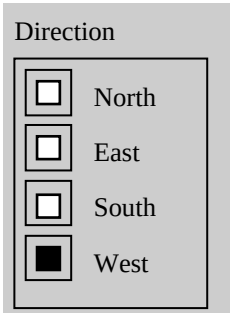
row içinde bölümler yatay olarak düzenlenir.



boxed_row bölümleri yatay olarak düzenler ve bir kutu ile sarar.

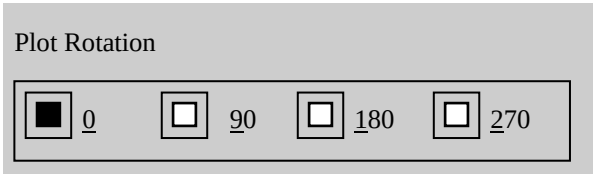


radio_column radio_button bölümleri düşey olarak düzenler.



boxed_radio_column radio_button bölümleri düşey olarak bir kutu ile sarar.

radio_row radio_button bölümleri yatay olarak düzenler.



boxed_radio_row radio_button bölümleri yatay olarak bir kutu ile sarar.

Dekoratif ve Bilgi Bölümleri


image cizgilerden oluşmuş resim gösteren bir alandır.

text başlıkları veya bilgilendirme maksatlı bir yazı alanıdır.

spacer bölüm ve gruplar arasında boşluklar bırakmak için kullanılır.

Önceden Belirlenmiş Bölüm Özellikleri

Özellik (attribute) İsmi	Özelliği Kullanabilen Bölümler
action	tüm aktif bölümler
alignment	tüm bölümler
allow_accept	edit_box, image_button, list_box
aspect_ratio	image, image_button
big_increment	slider
children_alignment	row, column, radio_row, radio_column, boxed_row, boxed_column, boxed_radio_row, boxed_radio_column
children_fixed_height	row, column, radio_row, radio_column, boxed_row, boxed_column, boxed_radio_row, boxed_radio_column
children_fixed_width	row, column, radio_row, radio_column, boxed_row, boxed_column, boxed_radio_row, boxed_radio_column
color	image, image_button
edit_limit	edit_box
edit_width	edit_box, popup_list
fixed_height	tüm bölümler
fixed_width	tüm bölümler
height	tüm bölümler
initial_focus	dialog
is_bold	text
is_cancel	button
is_default	button
is_enabled	tüm aktif bölümler
is_tab_stop	tüm aktif bölümler
key	tüm aktif bölümler
label	boxed_row, boxed_column, boxed_radio_row, boxed_radio_column, button, dialog, edit_box, list_box, popup_list, radio_button, text, toggle
layout	slider
list	list_box, popup_list
max_value	slider
min_value	slider
mnemonic	tüm aktif bölümler
multiple_select	list_box
small_increment	slider
tabs	list_box, popup_list
value	text, aktif bölümler (button ve image_button 'lar hariç)
width	tüm aktif bölümler

Özellik (attribute) İsmi	Görevi (seçilmiş veya *true* olarak belirlenmişse)
action	Bölüm seçildiğinde çalışması istenen bir AutoLISP ifadesi
alignment	Bir grup içinde bölümün yatay (**left**, **right** veya **centered**) veya dikey (**top**, **bottom** veya **centered**) pozisyonu
allow_accept	Bu bölüm seçildiğinde **is_default** düğmesi aktif duruma gelir (**true**,**false**)
aspect_ratio	Bir *image*'in (resmin) x/y oranı
big_increment	*slider* hareketi için artım mesafesi
children_alignment	Bir grupun alt bölümünün yerleşimi. satır (row) için (**left**, **right** veya **centered**) veya sütun (column) için (**top**, **bottom** veya **centered**) pozisyonu
children_fixed_height	Bir grupun alt bölümünün yüksekliği sabit kalır (**true**,**false**)
children_fixed_width	Bir grupun alt bölümünün genişliği sabit kalır (**true**,**false**)
color	Bir *image*'in arkafon (background) rengi
edit_limit	Kullanıcının girebileceği maksimum karakter sayısı
edit_width	Bölümün giriş (***edit***) alanının genişliği
fixed_height	Yükseklik sabit kalır (**true**,**false**)
fixed_width	Genişlik sabit kalır (**true**,**false**)
height	Bölümün yüksekliği
initial_focus	İlk başlangıçta seçilmesi istenen bölüm ***key***
is_bold	Yazıyı koyu (*bold*) olarak gösterme (**true**,**false**)
is_cancel	İptal (Ctrl+C) tuşu basıldığında düğme çalışır (**true**,**false**)
is_default	Kabul (enter) tuşu basıldığında düğme çalışır (**true**,**false**)
is_enabled	Bölüm ilk değer olarak kullanılabilir belirlenir(**true**)
is_tab_stop	Bölümü bir Tab konumu olarak belirler (**true**,**false**)
key	Uygulama tarafından kullanılan her bir bölümün özel ismi
label	Bölümün görünen etiketi
layout	*slider* yatay (**horizontal**) veya dikey (**vertical**) durumda
list	Listede görülmesi istenen ilk değerler
max_value	*slider*'ın maksimum değeri
min_value	*slider*'ın minimum değeri
mnemonic	Klavyeden seçim için bölüme ait kısatuş karakter
multiple_select	*list_box* 'ın birden fazla satırının seçilebilmesi (**true**,**false**)
small_increment	*slider* hareketi için artım mesafesi
tabs	Listenin gösterimi için Tab durağı
value	Bölümün ilk değeri
width	Bölümün genişliği

Key ve Value Özellikleri

Bütün önceden belirlenmiş aktif bölümler (tile) sahip olduğu özelliklerdir :

key : diyalog içinde her bir bölüm için kullanılan özel isim
value : bölümün kabul edilen dizgi (string) cinsinden ilk değeri

Yerleştirme ve Büyüklük Özellikleri

Bütün bölümler (tile) tarafından kullanılabilen düzenleme özellikleridir :

width : bölüm için karakter genişliği cinsinden genişlik; gerçek veya tamsayı
height : bölüm için karakter genişliği cinsinden yükseklik; gerçek veya tamsayı
alignment : grup içinde bölümün yatay (**left**, **right** veya **centered**) veya dikey (**top**, **bottom** veya **centered**) pozisyonu
children_alignment : grup içindeki tüm alt bölümlerin yatay (**left**, **right** veya **centered**) veya dikey (**top**, **bottom** veya **centered**) pozisyonu
fixed_width : *true* ise (ilk değer: *false*) bölüm genişliği sabit kalır
fixed_height : *true* ise (ilk değer: *false*) bölüm yüksekliği sabit kalır
children_fixed_width : *true* ise (ilk değer: *false*) grup içindeki tüm alt bölümlerin genişliği sabit kalır
children_fixed_height : *true* ise (ilk değer: *false*) grup içindeki tüm alt bölümlerin yüksekliği sabit kalır

Fonksiyonel Özellikler

Dekoratif olmayan tüm aktif bölümler tarafından kullanılabilen özelliklerdir :

is_enabled : *false* (ilk değer: *true*) ise bölüm gri renk alır ve seçilemez olur
is_tab_stop : *false* ise (ilk değer: *true*) bölüme klavyeden *Tab* tuşuyla ulaşılamaz
mnemonic : klavyeden kısatuş ile seçim için bölüme ait tırnak içinde bir karakter
action : bölüm seçildiğinde çalışması istenen bir AutoLISP ifadesi. *action_tile* tanımı bu *action* ifadesini geçersiz kılar.

Önceden Belirlenmiş Bölüm Özellikleri

Bütün bölümlerde geçerli olan yukarıdaki key, value, yerleştirme, büyüklük ve fonksiyonel özellikleri yanında, bölümler şu özelliklere sahip olabilirler :

boxed_column
label : "etiket"
boxed_radio_column
label : "etiket"
value *:* seçilmiş *radio_button*'un (*value* = "1" olan) *key* dizgisi
boxed_radio_row
label : "etiket"
value *:* seçilmiş *radio_button*'un (*value* = "1" olan) *key* dizgisi
boxed_row
label : "etiket"
button
Yerleştirme ve Büyüklük Özellikleri
Fonksiyonel Özellikler
key : "isim"
label : "etiket"
is_cancel : *true* ise (ilk değer: *false*) *Esc* veya *Ctrl+C* tuşuna basıldığında seçilir
is_default : *true* ise (ilk değer: *false*) *Enter* tuşuna basıldığında seçilir
column
Yerleştirme ve Büyüklük Özellikleri
dialog
label : "etiket"
value : "etiket"
initial_focus : ilk başta seçilmiş olması (focus) istenen bölüm *key* dizgisi
edit_box
Yerleştirme ve Büyüklük Özellikleri
Fonksiyonel Özellikler
key : "isim"
label : "etiket"
edit_width : karakter genişliği cinsinden; bir tamsayı veya gerçek sayı
edit_limit : müsaade edilen karakter sayısı; bir tamsayı (ilk değer : 132)
value : kutuda görülmesi istenen ilk yazı dizgi
allow_accept : *true* ise (ilk değer: *false*) *Enter* tuşuna basıldığında seçilir
image
color : arkafon rengi; bir tamsayı (ilk değer : 7) veya belirlenmiş renk ismi
aspect_ratio : resmin genişliğinin yüksekliğine oranı; bir gerçek sayı
image_button
Yerleştirme ve Büyüklük Özellikleri
Fonksiyonel Özellikler
key : "isim"
color : arkafon rengi; bir tamsayı (ilk değer : 7) veya belirlenmiş renk ismi
allow_accept : *true* ise (ilk değer: *false*) *Enter* tuşuna basıldığında seçilir
aspect_ratio : resmin genişliğinin yüksekliğine oranı; bir gerçek sayı

list_box

Yerleştirme ve Büyüklük Özellikleri
Fonksiyonel Özellikler
key : "isim"
label : "etiket"
multiple_select : *true* ise (ilk değer: *false*) *list_box* 'dan birden fazla satır seçilebilir
list : *list* için ilk satırlar listesi; yeni satır sembolü (\n) ile ayrılmış dizgi
tabs : listenin aynı hizada olması için tırnak içinde boşluklarla ayrılmış gerçek sayılar
value : listede seçili satırları gösteren tırnak içinde boşluklarla ayrılmış gerçek sayılar
allow_accept : *true* ise (ilk değer: *false*) *Enter* tuşuna basıldığında seçilir

popup_list

Yerleştirme ve Büyüklük Özellikleri
Fonksiyonel Özellikler
key : "isim"
label : "etiket"
edit_width : karakter genişliği cinsinden; bir tamsayı veya gerçek sayı
value : listede seçili satır numarası; tırnak içinde yazılmış bir tamsayı (ilk değer: "0")
list : *list* için ilk satırlar listesi; yeni satır sembolü (\n) ile ayrılmış dizgi
tabs : listenin aynı hizada olması için tırnak içinde boşluklarla ayrılmış gerçek sayılar

radio_button

Yerleştirme ve Büyüklük Özellikleri
Fonksiyonel Özellikler
key : "isim"
label : "etiket"
value : "1" ise *radio_button* açıktır (*on*), "0" ise *radio_button* kapalıdır (*off*)

radio_column

value *:* seçilmiş *radio_button*'un (*value* = "1" olan) *key* dizgisi

radio_row

value *:* seçilmiş *radio_button*'un (*value* = "1" olan) *key* dizgisi

row

Yerleştirme ve Büyüklük Özellikleri

spacer

Yerleştirme ve Büyüklük Özellikleri

slider

Yerleştirme ve Büyüklük Özellikleri
Fonksiyonel Özellikler
key : "isim"
min_value : *slider* 'ın verebileceği en küçük değer (-32768 <) (ilk değer: 0)
max_value : *slider* 'ın verebileceği en büyük değer (< 32767) (ilk değer: 10000)
small_increment : *slider* 'ın değerini arttırma miktarı (ilk değer: sınırdeğer / 100)
big_increment : *slider* 'ın değerini arttırma miktarı (ilk değer: sınırdeğer / 10)
layout : *slider* 'ın *horizontal* (yatay) veya *vertical* (dikey) konumda durmasını belirler
value : slider için ilk değer; tırnak içinde bir tamsayı (ilk değer: *min_value*)

text

label : "etiket"
value : *text* bölümünde görünmesi istenen yazı
is_bold : *true* ise (ilk değer: *false*) yazı **bold** (koyu) karakterler ile gösterilir

toggle

Yerleştirme ve Büyüklük Özellikleri
Fonksiyonel Özellikler
key : "isim"
label : "etiket"
value : "1" ise *toggle* seçilmiştir (*X veya V seçim işaretli*), "0" ise *toggle* kutusu boştur

Diyalog Kontrol Lisanı (DCL)

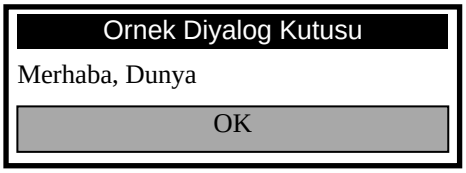
DCL diyalog kutularının tanımları, bir ASCII (text) dosyadır. DCL dosyasının eki **.dcl** 'dir. Bir **.dcl** dosyasında birden fazla diyalog kutusu tanımı veya başka .dcl dosyalarında kullanılabilecek prototip bölümler bulunabilir.

Bir diyalog kutusu, bölümlerin (tile) ağaç yapısında düzenlenmesiyle meydana gelir, ve DCL dosyası ise bu ağaçların hem programcı hem de AutoCAD tarafından okunabilir listesidir. Bir DCL dosyasının hiyerarşisi yazılım ile gösterilir, bu program kodunun kolayca okunabilmesini sağlar.

Örnek Bir DCL Diyalog Kutusu :
Aşağıdaki diyalog kutusu tanımını içeren bir ASCII yazı dosyası oluşturduğumuzu farzedelim :

```
merhaba :  dialog {
        label = "Ornek Diyalog Kutusu";
        : text {
            label = "Merhaba, Dunya";
        }
        : button {
            key = "accept";
            label = "OK";
            is_default = true;
        }
}
```

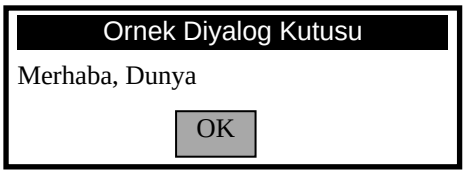
Böyle bir DCL program parçası şu diyalog kutusunu oluşturur :



Şekil 3. Örnek diyalog kutusu

Bu diyalog kutusunun görünümünü düzenlemek için, *OK button* 'ın genişlemesini engelleyen ***fixed_width*** özelliğini ***true*** ve *button* 'ı ortalayan ***alignment*** özelliğini ***center*** (ilk değer: ***left***) yaparsak DCL kodu aşağıdaki şekle gelir :

```
merhaba :  dialog {
        label = "Ornek Diyalog Kutusu";
        : text {
            label = "Merhaba, Dunya";
        }
        : button {
            key = "accept";
            label = "OK";
            fixed_width = true;
            is_default = true;
            alignment = centered;
        }
}
```

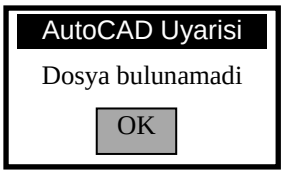


Şekil 4. Yerleşim düzenlemesi sonrası örnek diyalog kutusu

AutoCAD 'de aynı işlemi yapan standart bir çıkış düğmesi (button : **ok_only**) vardır. Bu standart button kullanılırsa DCL kod şöyle yazılabilir :

```
merhaba :  dialog {
        label = "Ornek Diyalog Kutusu";
        : text {
             label = "Merhaba, Dunya";
        }
        ok_only;
}
```

Bu şekilde bölümlerin (tile) ve özelliklerin (attributes) satırlar (row) ve sütunlar (column) şeklinde gruplanmasıyla çok daha karmaşık diyalog kutularını oluşturabilirsiniz. Mesela bir *uyarı mesaj kutusunu*, yukarıdaki programda az bir değişiklik yaparak elde edebiliriz :



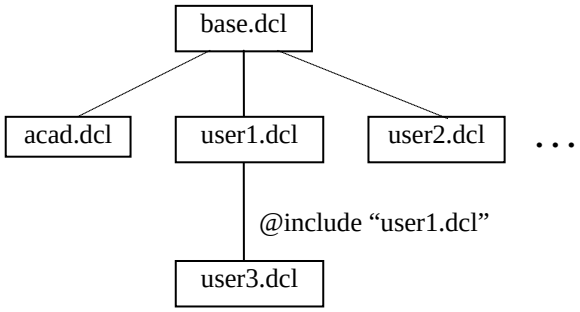
Şekil 5. Uyarı Kutusu

DCL Dosya Yapısı

AutoCAD içinde standart bölümleri tanımlayan ve bunları kullanan *base.dcl* ve *acad.dcl* DCL dosyaları verilmiştir. DCL dosyaları ile, diyalog kutularının tanımlanması yanında yeni bölümlerin (tile) *prototiplerinin* ve *altgruplarının* da tanımlanması mümkündür. Bunlar başka DCL dosyaları tarafından da referans göstererek kullanılabilirler. *base.dcl* dosyası önceden belirlenmiş bölümleri (tile) ve bölüm tiplerini (tile type) tanımlar. AutoCAD PDB uygulaması, kullanıcının *base.dcl* dosyasını değiştirmesine izin vermez. *acad.dcl* dosyası, AutoCAD'in kullandığı tüm standart diyalog kutularını tanımlar. Kullanıcı bu standart diyalog kutularının görünüşlerini özelleştirmek isterse (işlevi değiştirilemez), acad.dcl dosyasını değiştirebilir.

Kullanıcı DCL Dosyası

Yeni bir uygulamaya diyalog kutusu oluşturmak için, bir yazı programı kullanarak yeni bir DCL dosyası oluşturulmalıdır. Kullanıcı tarafından oluşturulan tüm DCL dosyaları otomatik olarak base.dcl dosyasında belirlenmiş bölümlere (tile) referans gösterir. Dolayısıyla bu önceden belirlenmiş bölüm prototipleri ayrıca referans göstermeden yeni DCL dosyada doğrudan kullanılabilmektedir. *base.dcl* dosyası benzeri başka bir bölüm prototipleri ve grupları tanım dosyası oluşturulursa, diğer DCL dosyadan bu prototip ve grupları kullanabilmek için *@include dosya_ismi* satırıyla referans göstermek gerekir.



Şekil 6. DCL dosya hiyerarşisi

Mesela aşağıdaki satır, içinde yazıldığı dosyanın *usercore.dcl* dosyasındaki tanımları kullanabilmesini sağlar.

```
@include “usercore.dcl”
```

Bölüm (tile) Tanımları
Bir bölüm tanımı şu şekildedir :

```
isim : eleman1 [: eleman2 : eleman3 . . . ] {
        özellik = değer;
        . . .
}
```

Burada herbir *eleman* önceden belirlenmiş bir *bölüm*dür. Bu yeni bölüm (*isim*) parantez içindeki ({ }) bütün *elemanların* özelliklerini kazanır.

Bir düğme (*button*) prototipinin tanımı şöyledir :

```
button : tile  {
        fixed_height = true;
        is_tab_stop  = true;
}
```

base.dcl ‘de *default_button* şu şekilde tanımlanmıştır :

```
default_button : button  {
        is_default = true;
}
```

default _button , *button* bölümünün (*tile*) *fixed_height* ve *is_tab_stop* özellik değerlerini alır. Bunlara yeni bir özellik (*is_default*) ekler, ve bu özelliğe *true* değerini verir.

Bölüm (tile) Referansları

Bir bölüm referansı şu yapıya sahiptir :

```
isim;
```
veya
```
: isim {
        özellik = değer;
        . . .
}
```

İlk *isim* tanımı, bölüme ait tüm özellik değerlerini değiştirmeden alır, ikinci isim tanımında ise bölüme yeni bir özellik ekler veya daha önceki özellik değerini değiştirir. Değiştirilen özellik değerleri sadece bu yeni bölüm tanımı için geçerlidir, ilk prototip tanımı içindeki özellik değerlerinde herhangi bir değişiklik olmaz. Bu ikinci referans gösterme yapısı, sadece prototip tanımları için geçerlidir, altgrup tanımlarıiçin kulanılamaz.

Mesela; spacer bölümü herhangi bir özelliğe sahip olmadığından, birinci tanım ile referans gösterilebilir :

```
spacer;
```

ok_cancel bölümü base.dcl 'de tanımlı bir altgrup olduğu için sadece birinci tanım ile referans gösterilebilir :

```
ok_cancel;
```

Bununla birlikte tek bir bölümün özelliği yeni değer verilebilir; aynı özelliklere sahip fakat değişik bir yazı gösteren bir düğme (button) tanımlanabilir :

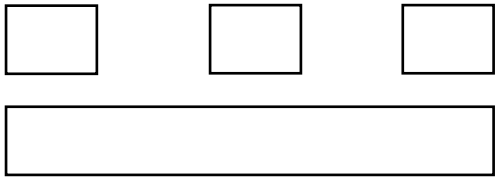
```
: retirement_button {
        label = "Hoscakalin";
}
```

Açıklamalar
Bir DCL dosyasında, *// açıklama //* veya */* açıklama */* şeklinde yazılan ifadeler göz önüne alınmazlar.

DCL Teknikleri

```
: column {
    : row {
        : compact_tile {
        }
        : compact_tile {
        }
        : compact_tile {
        }
    }
    : large_tile {
    }
}
```

Burada compact_tile , fixed_width özelliğine sahip ve large_tile bunların üçünün toplam genişliğinden büyükse, normal olarak DCL bu bölümleri alanı ortalayacak şekilde yayar, ve şöyle bir yerleşim oluşturur :

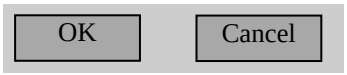


Şekil 7. Normal yatay yerleşim

Diyalog Kutusu Standart Çıkış Düğmeleri
acad.dcl içinde bulunan standart diyalog kutusu çıkış düğmeleri şunlardır :

ok_ button

ok_cancel



ok_cancel_help

ok_cancel_help_info

Diyalog Kutularının AutoLISP ile Kullanılması

Bir Diyalog Kutusunu Kullanmak İçin Örnek Fonksiyon

Şekil 4.'deki diyalog kutusunu oluşturan DCL kodu şöyleydi :

```
merhaba : dialog {
        label = "Ornek Diyalog Kutusu";
        : text {
            label = "Merhaba, Dunya";
        }
        ok_only;
}
```

Bu DCL kodu *merhaba.dcl* isimli bir dosya olarak saklanırsa, bu diyalog kutusunu çağırmak için aşağıdaki AutoLISP fonksiyonu kullanılabilir (burada en az sayıda hata konrolu bulunmaktadır) :

```
( defun uyarigoster ( / dcl_id )
        (setq dcl_id (load_dialog "merhaba.dcl"))     ; DCL dosyasını çağır
        (if (not (new_dialog "merhaba" dcl_id) )            ; dialog'u başlat
            (exit) )                                  ; çalışmazsa fonksiyondan çık

        (action_tile "accept" "(done_dialog)" )
        ; kullanıcının Enter tuşuna dokunması veya fare ile OK  (accept)
        ; düğmesinin seçilmesi sonucu, diyalog kutusundan çıkma
        ; (done_diyalog) fonksiyonu çalışır

        (start_dialog)                            ; diyalog kutusunu göster
        (unload_dialog dcl_id)                    ; DCL dosyasından çık
)
```

(start_dialog) 'dan sonra diyalog kutusu, işlevi **(done_dialog)** fonksiyonu olarak belirlenen bir bölümü (genellikle bir düğme; button tile) kullanıcının seçmesine kadar aktif olarak kalır. **(action_tile)**, bölüm (buradaki örnekte *key* özelliği *"accept"* olan *ok_button* **tile**) ile **action** (eylem) ifadesini ilişkilendirir. Bu sebeple (done_dialog), (start_dialog) ifadesinden önce bir (action_tile) ifadesi içine yerleştirilmiştir.

Daha karmaşık diyalog kutuları daha fazla **(action_tile)** satırları kullanır. Çok çeşitli diyalog kutularının çağırılması için de, **(start_dialog)** ve **(unload_dialog)** çağrıları ard arda daha fazla kullanılır. Fakat genel çağrı sırası bu şekilde kalır.

Fonksiyonların Çağrılma Sırası

Yukarıdaki örnek, fonksiyonların tipik bir çağrılma sırasını gösterir :

1. **(load_dialog)** çağrısını kullanarak DCL dosyasını yükle.
2. Bir diyalog kutusunu AutoCAD grafik ekranında görüntülemek için **(new_dialog)** fonksiyonunu çağır. (new_dialog)'un hata mesajını kontrol etmek önemlidir.
3. Gerekli olan diyalog kutusu bölüm değerlerine (*value*), listelere (*list*), ve resimlere (*image*) ilk değerlerini ver. Bu işlem çoğu bölüm (*tile*) değerleri (*value*) ve durumları için **(set_tile)** ve **(mode_tile)**; liste kutuları **(list_box)** için **(start_list)**, **(add_list)** ve **(end_list)**; resimler (*image*) için ise **(start_image)**, **(vector_image)**, **(fill_image)**, **(slide_image), (end_image)** ve boyut fonksiyonlarıdır.

Artık **(action_tile)** ile eylem ifadelerinizi veya fonksiyonlarınızı belirleyebilirsiniz.

1. Şimdi kullanıcının değerlerini ve seçimlerini girebilmesi için kontrolu diyalog kutusuna vermek üzere **(start_dialog)** fonksiyonu çağır.
2. Kullanıcının eylemlerine karşılık gelen fonksiyonları gerçekleştirin. Bu fonksiyonlar içinde **(get_tile)**, **(get_attr)**, **(set_tile)**, **(mode_tile)** ve diğer PDB bölüm işleme fonksiyonlarını kullanabilirsiniz.
3. Kullanıcı, **(done_dialog)** fonksiyonunun çalışmasına sebep olan çıkış düğmesine basar. Bu (start_dialog) fonksiyonunun sona ermesine sebep olur. Burada **(unload_dialog)** fonksiyonunu çağırarak DCL dosyasından çıkın.

Fonksiyon sırası şu şekilde gösterilebilir :

```
(load_dialog)

    (new_dialog)
        (action_tile)   ; ve diğer tanımlar
    (start_dialog)
        ; eylem ifadeleri ve fonksiyon çağrılarından
        (get_tile)      ; ve diğer giriş işlemleri
        (set_tile)
        (done_dialog)

 (unload_dialog)
```

Diyalog Kutusu Aktif Durumdayken Kullanılamayan Fonksiyonlar

Diyalog kutusu aktif durumdayken (*start_dialog* fonksiyonu esnasında) kullanıcının, ekranı etkileyerek diyalog kutusunun görünüşünü bozabilecek, veya diyalog kutusu dışındaki bir alan ile ilgili giriş yapmasını gerektiren (command), (osnap), (getint), getreal), (getpoint), (prompt), (redraw), (ssget), (entdel) vb. gibi AutoCAD ve AutoLISP fonksiyonlarını kullanmasına müsaade edilmemektedir. Bu fonksiyonları kullanabilmek için (done_dialog) fonksiyonu çağırılmalıdır.

AutoLISP : Action (eylem) İfadesi

Bir AutoLISP **(action_tile)** eylem ifadesi aşağıda verilen, hangi bölümün (tile) seçildiğini ve *action*'ın durumunu gösteren değişkenlere ulaşabilir. Bu değişken isimleri AutoCAD tarafından kullanılmaktadır ve sadece okuma izni verilmiştir.

Değişken	Anlamı
$key	seçilen bölümün *key* özelliği
$value	seçilen bölümün o andaki *value* özelliği dizgi olarak; *edit_box*'da bir *"yazı"*, veya *toggle*'da *"1"* veya *"0"*
$data	uygulama tarafından yönetilen veriler
$reason	eyleme (action) sebep olan sebep olan *edit_box*, *list_box*, *image_button*, ve *slider* bölümleri için kullanılır
$x	bir *image_button* seçiminde X koordinatı
$y	bir *image_button* seçiminde Y koordinatı

Mesela, **edit1** bir *text_box* ise, kullanıcının *text_box* 'ı serbest bırakmasıyla aşağıdaki **(action_tile)** eylem ifadesi çalışır :

(action_tile "edit1" "(setq ns $value)")

Burada **$value** kullanıcı tarafından girilen yazıyı tutar ve yukarıdaki ifade bu değeri **ns** değişkeninde saklar.

(action_tile "edit1" "(setq newtile $key)")

Bu ikinci ifade, daha sonra kullanabilmek için seçilen bölümün (tile) ismini (burada: *"edit1"*) **$key** değişkeninden alarak **newtile** değişkeninde saklar.

Fonksiyon Çağrı Sebepleri ($reason)

Kod	ADS sembolü	Anlamı
1	CBR_SELECT	Kullanıcı bölümü (tile) seçti
2	CBR_LOST_FOCUS	Kullanıcı bir *edit_box* bölümünden başka bir bölüme geçti, henüz son seçimi yapmadı
3	CBR_DRAG	Kullanıcı bir *slider* göstergesini sürüklüyerek değer değiştirdi, henüz son seçimi yapmadı
4	CBR_DOUBLE_CLICK	Kullanıcı son seçimini yapmak üzere *list_box* veya *image* button üzerine fare ile iki defa dokundu (double click)

Dialog kutuları birbiri içinden çağrılabilir.

(action_tile "button_1" "(uyarigoster)")

ifadesinde *button_1* düğmesine basıldığında daha önce tanımladığımız *"Merhaba, Dunya"* diyalog kutusu görünür.

Bölümler Üzerinde İşlemler

Başlangıç Modları ve Değerleri

Kursörün diyalog kutusu içinde belirli bir bölüm üzerine odaklanması, seçilebilir (*enable*) veya seçilemez (*disable*) yapılması, bir *edit_box* veya *image* ise odaklanarak parlatılması gibi işlemler **(mode_tile)** fonksiyonuyla gerçekleştirilebilir.

Mesela, önceden belirlenmiş bir soyadı bir edit_box içinde göstermek ve bunu diyalog kutusunun odaklanarak seçilmiş olan bölgesi olarak belirlemek için aşağıdaki kodu kullanabiliriz :

```
(setq  soyisim "Kocabas")     ; ilk değer

(set_tile "soyad" soyisim)    ; key="soyad" alanına başlangıç değeri ver
(mode_tile "soyad" 2)         ; mod=2: bölüm (tile) seçilmiş durumdadır
```

(mode_tile) fonksiyonu için bölüm (tile) mod kodları :

Değer	ADS sembolü	Anlamı
0	MODE_ENABLE	Bölüm (tile) seçilebilir
1	MODE_DISABLE	Bölüm (tile) seçilemez (gri renkte)
2	MODE_SETFOCUS	Kursör bölüm üzerinde odaklanmış
3	MODE_SETSEL	*edit_box* içindeki yazıyı seç (highlight)
4	MODE_FLIP	*image* parlak (highlight: *on*) veya normal (*off*)

Misal :

(action_tile "group_on" "(mode_tile \"group\" (- 1 (atoi $value)))")

list_box ve popup_list Oluşturma

(start_list), **(add_list)** ve **(end_list)** fonksiyonlarını sırayla kullanarak list_box ve popup_list içindeki gösterilen listeler oluşturulabilir.

isimler , *"secimler"* isimli *list_box*'da görünmesi istenen satırları içeren bir liste ise, aşağıdaki program parçası listeyi oluşturur ve gösterir :

```
(start_list "secimler")        ; list_box ismini belirle; operation=3: LIST_NEW
(mapcar 'add_list isimler)     ; AutoLISP listesini belirle
(end_list)                     ; listeyi kapatır
```

isimler , 12 satırlık bir liste ise, ve buna *yenisimler* isimli başka bir liste eklenmek istenirse, yeni bir liste oluşturma programa benzeyen aşağıdaki program parçası kullanılabilir :

```
(start_list "secimler" 2)      ; operation=2: LIST_APPEND
(mapcar 'add_list yenisimler)  ; AutoLISP listesini belirle
(end_list)                     ; listeyi kapatır
```

Tek bir liste satırın değiştirilmesi için (add_list) fonksiyonunun bir defa çağırılması yeterlidir. Bu defa değiştirilmesi gereken satırın numarası (index) belirtilmelidir :

```
(start_list "secimler" 1 5)    ; operation=1: LIST_CHANGE ; index=5
(add_list "SURPRIZ!")          ; listenin index=5. satırına SURPRIZ yazılır
(end_list)                     ; listeyi kapatır
```

list_box Değerlerini İşleme

list_box bölüm (tile) değeri (value) ön boşluklar içerebilir, bu sebeple eğer birden fazla liste satırı okunacaksa karşılaştırmaları dizgi (yazı) olarak yapmayınız, önce (atoi) fonksiyonu ile tamsayı (integer) değere çeviriniz. *"birtane"* isimli listenin sadece bir seçim kabul ettiği kabul edilirse, aşağıdaki program parçası listenin üçüncü satırının seçilip seçilmediğini kontrol eder :

```
(setq satir (get_tile "birtane") )
(cond
    ( (/= satir " ")
        (= 2 (atoi satir) )
        ; üçüncü girişi işle
        . . .
    )
)
```

image Oluşturma

(slide_image) ile oluşturacağınız bir slide (.sld) veya bir slide kütüphanesinin (.slb) parçası olabilir. AutoCAD MSLIDE komutu ile bir slide oluşturabilirsiniz. *ustbakis.sld* isimli slide'ın tek bir dosya içinde kabul edilirse, aşağıdaki kod kullanılabilir :

```
(setq x (dimx_tile "gorunus")
      y (dimy_tile "gorunus") )
(start_image "gorunus")
(slide_image 0 0 x y "ustbakis")
(end_image)
```

image_button ile Girişi

Kullanıcının image button üzerinde fare ile dokunduğu bölgeye göre resim seçen bir program parçası şu şekilde yazılabilir :

```
(action_tile  "resim_sec" "(dokun_golge  $key $value $y)" )
.
.
(defun dokun_golge (key val y)
    (setq bolum (/ (dimy_tile key) 2) )  ; image düşey olarak ikiye bölünür
    (if (> y bolum)                      ; orijin üst-sol köşededir
       (setq sonuc "Light")
       (setq sonuc "Dark")  )
)
```

radio Grup İşlemleri

Kullanıcının diyalog kutusunu serbest bırakmasından sonra üç-boyutlu bir cismin hangi görünüşünün gösterileceğini belirleyen bir radio grubu tasarlıyalım. Burada radio grubu dört adet düğme bulundurur :

```
(action_tile  "gorunus_sec" "(dokun_gorunus $value)" )
.
.
(defun dokun_gorunus (hangi)
    (cond  ( (= hangi "on")        (setq goster_hangi 0)
           ( (= hangi "ust")       (setq goster_hangi 1)
           ( (= hangi "sol")       (setq goster_hangi 2)
           ( (= hangi "sag")       (setq goster_hangi 3)
)
```

slider Kullanımı

Aşağıda bir *slider*'ı kullanabilen basit bir fonksiyon verilmiştir. Burada kullanıcının değeri *slider*'ı kullanarak veya doğrudan girmesi mümkündür. *slider_bilgisi* bölümü (tile), *slider*'ın o andaki değerini (*value*) ondalık sayı olarak gösteren bir fonksiyon (*edit_box*) kullanmaktadır. Eğer değer *slider_bilgisi* 'nde girilirse, *edit_box slid*er'ın değerini güncelleştirir.

```
(action_tile  "benim_slider" "(slider_eylemi  $value $reason)" )
(action_tile  "slider_bilgisi" "(edit_box_eylemi  $value $reason)" )
.
.
```

```
(defun slider_eylemi (val  why)
     (if (or (= why 3) (= why 1) )
          (set_tile “slider_bilgisi” val)      ; sonucu göster
     )
)

(defun edit_box_eylemi (val  why)
     (if (or (= why 2) (= why 1) )
          (set_tile “benim_slider” val)      ; sonucu göster
     )
)
```

edit_box Kullanımı

```
(action_tile  “benim_editbox” “(yazi_eylemi $value $reason)“ )
.
.
(defun yazi_eylemi (val  why)
     (if (or (= why 2) (= why 4) )
          ; burada geçici değer üzerinde sınır kontrolu yapınız
     )
)
```

Fonksiyonların Yapıları

DCL Dosyasının Açılması ve Kapatılması
(load_dialog dcl_dosyası**)** : DCL dosyasını yükler.
(unload_dialog dcl_id**)** : DCL dosyasını kapatır.

Diyalog Kutusunun Açılması ve Kapatılması
(new_dialog dlgname dcl_id [action-ifadesi] [screen-pt]**)**
: diyalog kutusunu tanımlar ve gösterir.
(start_dialog) : diyalog kutusundan kullanıcı girişlerine cevap vermeye başlar.
(done_dialog [status] **)** : diyalog kutusunu kapatır ve göstermeyi durdurur.
(term_dialog) : bütün açık olan diyalog kutularını kapatır.

Action ifadesinin ve cevap fonksiyonlarının tanımlanması
(action_tile key action-ifadesi**)**
: action ifadesi veya cevap fonksiyonu ile belirtilen bölüm (tile) ilişkilendirilir.

Bölüm (tile) İşlemleri ve Özellikleri
(mode_tile key mode**)** : belirtilen bölümün (tile) modunu belirler.
(get_attr key attribute**)** : belirtilen özelliğin DCL dosyasındaki değerini alır.
(get_tile key**)** : belirtilen bölümün çalışma anındaki değerini alır.
(set_tile key value**)** : belirtilen bölümün çalışma anındaki değerini belirler.

list_box ve popup_list Oluşturma
(start_list key [operation [index]] **)** : belirtilen *list_box* veya *popup_list* 'i başlatır.
operation=1: LIST_CHANGE ; 2: LIST_APPEND ; 3: LIST_NEW (default: ilk değer)
(add_list item**)** : açık listeye belirtilen bir satır (dizgi) ekler veya satırı değiştirir.
(end_list) : açık listenin işlenmesini bitirir.

image Oluşturma
(dimx key**)** : belirtilen bölümün (tile) x boyutunu alır.
(dimy key**)** : belirtilen bölümün (tile) y boyutunu alır.
(start_image key**)** : belirtilen resmi (image) oluşturmaya başlar.
(vector_image x1 y1 x2 y2 color**)** : aktif olan resim (image) içinde bir vektör çizer.
(fill_image x1 y1 x2 y2 color**)**: aktif olan resim içinde dolu dikdörtgen çizer.
(slide_image x1 y1 x2 y2 sldname**)** : aktif olan resim içinde AutoCAD slide'ı çizer.
(end_image) : aktif olan resim (image) oluşumunu bitirir.

Uygulamaya Özel Dosya
(client_data_tile key clientdata**)**
: uygulama tarafından yönetilen verilerin belirtilen bölüm (tile) ile ilişkilendirilmesi.

Fonksiyon Kataloğu

(action_tile key action-ifadesi**)**
(add_list item**)**
(client_data_tile key clientdata**)**
dimension fonksiyonları :
(dimx key**)**
(dimy key**)**
(done_dialog [status] **)**
(end_image)
(end_list)
(fill_image x1 y1 x2 y2 color**)**
(get_attr key attribute**)**
(get_tile key**)**
(load_dialog dcl_dosyası**)**
(mode_tile key mode**)**
(new_dialog dlgname dcl_id [action-ifadesi] [screen-pt]**)**
(set_tile key value**)**
(slide_image x1 y1 x2 y2 sldname**)**
(start_dialog)
(start_image key**)**
(start_list key [operation [index]] **)**
(term_dialog)
(unload_dialog dcl_id**)**
(vector_image x1 y1 x2 y2 color**)**

UYGULAMA :

merhaba.dcl

```
uyari : dialog {
                        label = "Ornek Diyalog Kutusu";
                        : toggle {
                                key = "secil";
                                label = "secilebilir";
                        }
                        : popup_list {
                                key= "liste";
                                label= "liste";
                                list="satir 1\nsatir 2\nsatir 3";
                        }
                        : radio_row {
                                : radio_button {
                                        key="acik";
                                        label="acik";
                                        value=1;
                                }
                                : radio_button {
                                        key="kapali";
                                        label="kapali";
                                        value=0;
                                }
                                : radio_button {
                                        key="diger";
                                        label="acik";
                                        value=0;
                                }
                        }
                        : column {
                                : edit_box {
                                        key = "mesaj";
                                        label = "Merhaba, Dunya";
                                        value = "45";
                                }
                                : slider {
                                        key = "bslider";
                                        value = 45;
                                        max_value = 100;
                                        min_value = 0;
                                        big_increment=5;
                                        small_increment=1;
                                        layout =horizontal;
                                }
                        }
                        ok_only;
}

goster : dialog {
                        label = "Goster Kutusu";
                        : button {
                                key = "goster";
                                label = "goster";
                        }
                        ok_only;
}
```

merhaba.lsp

```
(defun secil (val why)
	(if (or (= why 2) (= why 1) ) (progn
		(setq sec (- 1 sec))
		(mode_tile "mesaj" sec )
		(mode_tile "bslider" sec )
	) )
)


(defun bslider (val why)
	(if (or (= why 2) (= why 1) )
	(progn
		(setq sayi (atoi val))
		(set_tile "mesaj" val)
	) )
)

(defun mesaj (val why)
	(if (or (= why 2) (= why 1) )
	(progn
		(setq sayi (atoi val))
		(set_tile "bslider" val)
	) )
)

(defun uyari (/ dcl_id)
	(setq sayi 70 sec 1)

	(setq dcl_id (load_dialog "merhaba.dcl"))
	(if (not (new_dialog "uyari" dcl_id) ) (exit) )

	(set_tile "mesaj" (rtos sayi 2 0) )
	(set_tile "bslider" (rtos sayi 2 0) )
	(set_tile "bslider" (rtos sec 2 0) )

	(action_tile "accept" "(done_dialog)" )
	(action_tile "mesaj" "(mesaj $value $reason)" )
	(action_tile "bslider" "(bslider $value $reason)" )
	(action_tile "secil" "(secil $value $reason)" )

	(start_dialog)
	(unload_dialog dcl_id)
)

(defun c:goster (/ dcl_id)
	(setq dcl_id (load_dialog "merhaba.dcl"))
	(if (not (new_dialog "goster" dcl_id) ) (exit) )

	(action_tile "accept" "(done_dialog)" )
	(action_tile "goster" "(uyari)" )

	(start_dialog)
	(unload_dialog dcl_id)
)
```

Tablo DXF Grup Kodları

Group code	Meaning
-4	Conditional operator (used only with (ssget) and ads_ssget ())
-3	Extended entity data (XDATA) sentinel (fixed)
-2	Entity name reference (fixed)
-1	Entity name (changes each time drawing is opened; never saved);(fixed)
0	Starts an entity. The type of entity is given by the text value that follows this group (fixed)
1	The primary text value for an entity
2	A name: Attribute tag, Block name, and so on
3-4	Other textual or name values
5	Entity handle expressed as a hexadecimal string (fixed)
6	Line type name (fixed)
7	Text style name (fixed)
8	Layer name (fixed)
10	Primary point (start point of a Line or Text entity, centre of a Circle, etc.)
11-18	Other points Note: These are the only coordinate group codes that an application sees. The Y (20-28) and Z (30-38) coordinates that appear in a DXF file are passed to an application as part of an AutoLISP point list or an ADS result buffer
39	This entity's thickness if nonzero (fixed)
40-48	Floating-point values (text height, scale factors, etc.)
49	Repeated value-multiple 49 groups may appear in one entity for variable-length tables (such as the dash lengths in the LNPE table).A 7x group always appears before the first 49 group to specify the table Grc length
50-58	Angles
62	Colour number (fixed)
66	"Entities follow" flag (fixed)
67	Space (that is, model or paper space) (fixed)
70-78	Integer values such as repeat counts, flag bits,0r modes
210	Extrusion direction (fixed) Note: As with point coordinates, an application sees only the 210 group. The Y(220) and Z (230) components of an extrusion vector are passed to an application as part of an AutoLISP point list or an ADS result buffer
999	Comments
1000	An ASCII string (up to 255 bytes long) in XDATA
1001	Registered application name (ASCII string up to 31 bytes long) for XDATA (fixed)
1002	XDATA control string (" { " or " ş "); (fixed)
1003	Layer name in XDATA
1004	Chunk of bytes (up to 127 bytes long) in XDATA
1005	Entity handle in XDATA
1010	A point in XDATA
1011	A 3D World space position in XDATA
1012	A 3D World space displacement in XDATA
1013	A 3D World space direction in XDATA Note: Again, these are the only coordinate group codes that an application sees. The Y (1020,1021, 1022, 0r 1023) and Z (1030, 1031,1032,0r 1033) coordinates

	that appear in a DXF file are passed to an application as part of an AutoLISP point list or an ADS result buffer
1040	Floating-point value in XDATA
1041	Distance value in XDATA
1042	Scale factor in XDATA
1070	16-bit integer in XDATA
1071	32-bit signed long integer in XDATA

Tablo Yapılandırma Fonksiyonları

Fonksiyon	Maksadı
ads_set_tile()	Belirtilen tile için run-time değerini atar.
ads_mode_tile()	Belirtilen tile için mode değerini atar.
ads_start_list()	POP listesini başlatır.
ads_add_list()	Listeye belirtilen stringi ekler
ads_end_list()	Listeyi sonlandırır.
ads_start_image()	Diyalog kutusundaki, belirtilen şekli oluşturur.
ads_vector_image()	Diyalog kutusundaki, aktif şekilde vektör çizer
ads_fill_image()	Diyalog kutusundaki, aktif şekilde dolu bir dikdörtgen çizer
ads_slide_image()	Diyalog kutusundaki, aktif şekilde bir AutoCAD slaytı çizer.
ads_end_image()	Diyalog kutusundaki, aktif şekli kapatır.
ads_action_tile()	Kullanıcının ilgili tile'ı seçmesi durumunda icra edeceği fonksiyon belirtilir.
ads_client_data_tile	Uygulama verilerinin clientdata argümanına

Bir diyalog kutusu aktif iken çalıştırılmasına izin verilmeyen fonksiyonlar şunlardır.

AutoLISP
(command)
(osnap)
Kullanıcı veri girişi fonksiyonları:
(getint)
(getreal)
(getstrinq)
(getpoint)
(getcorner)
(getdist)
(getangle)
(getorient)
(getkword)

Görüntü kontrol fonksiyonları:
(prompt) b
(menucmd)
(redraw)
(graphscr)
(textscr)
(textpage)
Low-level grafik fonksiyonları:
(qrclear)
(grdraw)
(grread)
(grtext)
(grvecs)
Seçim seti fonksiyonları:
(ssget)
Entity fonksiyonları:
(entmod)
(entmake)
(entdel)
(entsel)
(nentsel)
(entupd)

Yukarıdaki fonksiyonların çalıştırılması durumunda

AutoCAD rejected function

mesajı gelir.

Diyalog Kutusunda Image Çizdirilmesi:

Şekil çizdirme fonksiyonlarının (Tablo 24), çatının koordinatlarına ihtiyacı vardır. Bu ise ads_dimensions_tile() fonksiyonu ile bulunur.

Tablo 24 Diyalog kutularnda Şekil Fonksiyonları

Fonksiyon	Anlamı
ads_vector_image	image içine vektör çizer
ads_fill_image	image içine içi dolu bir dikdörtgen çizer
ads_slide_image	image içine slide ekler.

Renkler, AutoCAD renk kodları ile (Tablo 25) color argümanında belirtilir.

Tablo 25 Renk Kodları

Kodlar	Anlamı
dialog_line	Geçerli diyalog kutusu çizgi rengi
dialog_foreqround	Geçerli diyalog kutusu ön fon rengi (Text için
dialog_backqround	Geçerli diyalog kutusu arka fon rengi
qraphics_backqround	Geçerli grafik ekranı arka fon rengi
black	AutoCAD colour = 0 (siyah) (siyah arka fon için beyaz)
red	AutoCAD colour =1 (kırmızı)
yellow	AutoCAD colour = 2 (sarı)
qreen	AutoCAD colour = 3 (yeşil)
cyan	AutoCAD colour = 4 (yeşilimsi mavi)
blue	AutoCAD colour = 5 (mavi)
maqenta	AutoCAD colour = 6 (eflatun)
white qraphics_foreqround	AutoCAD colour = 7 (beyaz) (beyaz arka fon için siyah)

Misal:

“cur_color”çatısı kırmızı renkte bir dikdörtgen şeklinde image elde eder.

```
short width, height;


ads_dimensions_tile(hdlg, "cur_color", &width, &height); ads_start_image(hdlg, "cur_color");
ads_fill_image(0, 0, width, height, 1); /* 1 == AutoCAD red */
ads_end_image ( ) ;
```

Çevresinde kırmızı çerçeve bulunan bir tile hazırlanması için,

```
short width, height;


ads_dimensions_tile(hdlg, "border", &width, &height); ads_start_image(hdlg, "border");
ads_vector_image(0, 0, 0, height, 1); /* 1 == AutoCAD red */ ads_vector_image(0, height,
width, height, 1);
ads_vector_image(width, height, width, 0, 1);
ads_vector_image(width, 0, 0, 0, 1);
ads_end_image ( ) ;
```

Yeşil bir dikdörtgen üzerinde kırmızı düşey çizgi çizilsin,

```
short width, height, x;


ads_dimensions_tile(hdlg, "stripe", &width, &height);
ads_start_image(hdlg, "stripe");
ads_fill_image(0, 0, 0, height, 3); /* 3 == AutoCAD green */
x = width/2; /*Centre the vector vertically */
ads_vector_image(x, 0, x, height, 1); /* 1 == AutoCAD red */
ads_end_image();
```

Slide'lar tek başına veya slide kitaplığında bulunabilir. Kitaplıkta bulunan slide'lar "kitaplık dosyası(Slide dosyası)şeklinde argümana yazılır.

Misal:

Kitaplık dosyası: allviews.slb ve

slide dosyası: frntview.sld ise

```
"allviews(frntview)"
```

"topview" İsimli slide dosyasını image olarak görüntülemek istersek,

```
short x, y;


ads_dimensions_tile(hdlg, "view", &x, &y); ads_start_image(hdlg, "view");
ads_slide_image(0, 0, x, y, "topview");
ads_end_image ( ) ;
```

Diyalog Kutusunda Image_Button Kullanılışı:

Diyalog kutusunda görüntülenen image-button'un kullanılması için kullanıcının tıkladığı koordinatların bilinmesi gereklidir. Bu koordinatlar callback paketinde x,y long tipte değişkenlere atanır.

Misal: Image Button'un yarısında "Light (aydınlık)" diğer yarısında "Dark (karanlık)" sembolize edilmiştir. "Light" kısmına basılırsa result değeri "Light", "Dark" kısmına basarsa "Dark" stringi atanıyor.

```
char result[31];          /* Global */


static void CALLB
/*FCN*/pick_shade(ads_callback packet cbpkt)


long threshold, pick_y = cbpkt->y;
ads_hdlg hdlg = cbpkt->dialog;
short x, y;


ads_dimensions_tile(hdlg, "image_sel", &x, &y);
threshold = y/2;
if (pick_y > threshold)
      strcpy(result, "Light");
else
      strcpy(result, "Dark");
}
```

AutoLisp Programcı Referans Kitabı

ÖNSÖZ

AutoLISP, profesyonel AutoCAD kullanıcılarının olduğu kadar amatör kullanıcıların da ilgi duyduğu noktalardan biridir. Ancak bir çok kullanıcı AutoLISP'ten nasıl faydalanabileceğini bilememekten yakınmaktadır.

Elinizdeki bu kitap, AutoLISP konusunda bazı bilgilere sahip olanların yanısıra, yeni ilgilenmeye başlayanların da olabileceği düşünülerek hazırlanmıştır.

Kitabın temel içeriği AutoLISP fonksiyonlarıdır. Fonksiyon tanımları herkesin anlayabileceği bir dille ve örneklerle açıklanmaya çalışılmıştır.

İlk bölümde, AutoLISP için gerekli bellek ayarlamalarının nasıl yapılacağı yeralmaktadır. Bununla birlikte bir AutoLISP dosyasının nasıl hazırlanabileceği, nasıl bir yapıya sahip olması gerektiği ve dikkat edilmesi gereken özel kurallar da bu bölümde yeralmaktadır. İkinci bölüm, AutoLISP'in temel fonksiyonlarını açıklamaktadır.

Üçüncü bölümde alfabetik sırada AutoLISP fonksiyonlarının tanımları ve ilgili örnekler yeralmaktadır. AutoLISP'e yeni başlayan arkadaşların bu fonksiyonları kavramakta güçlük çekmeyeceklerine eminim.

Dördüncü bölüm, obje seçme fonksiyonlarının açıklandığı bölümdür. Bu fonksiyonlar sayesinde kullanıcıların, ekrandaki objeleri yazdıkları programlar içinden seçebilme imkanları vardır.

Son üç bölümde ise, AutoLISP hata mesajları, çalışmaya hazır örnek programlar ve programlarınızı yazarken ihtiyaç duyabileceğinizi düşündüğüm AutoCAD Sistem Değişkenleri yeralmaktadır.

Çalışmalarınızda bu kitabın yardımcı olacağını umut ediyor, başarılar diliyorum.

Ender ÇIKIŞ *Mayıs 1994*

(= <arg1> <arg2> ...)
(/= <arg1> <arg2>)
(< <arg1> <arg2>...)
(<= <arg1> <arg2>...)
(> <arg1> <arg2>...)
(>= <arg1> <arg2>...)
(~ <arg1>)
(1+ <arg1>)
(1- <arg1>)
(abs <arg1
(and <kosul1> <kosul2> <kosul3>...)
(angle <nokta1> <nokta2>)
(angtos <açı> [mode] [hassasiyet])
(append <liste1> <liste2> <liste3>...)
(apply <fonksiyon> <liste>)
(ascii <dizgi>)
(assoc <başlık> <liste>)
(atan <arg1> [<arg2>])
(atof <dizgi>)
(atoi <dizgi>)
(atom <arg1>)
(boole <func> <int1> <int2>...)
(boundp <arg>)
(car <liste>)
(cadr <liste
(cdr <liste
cadr, cddr, caddr, cdddr vb.
(chr <arg>)
(close <dosya>)
(cond (<koşul> <fonksiyon>))
(cons <yeni eleman> <liste>)
(cos <açı>)
(distance <nokta1> <nokta2>)
(eq <arg1> <arg2>)
(equal <arg1> <arg2>)
(eval<arg1>)
(exp <sayı>)
(expt <sayı> <kat>)
(fix <sayı>)
(float <sayı>)
(foreach <isim> <liste> <ifade>)
(gcd <arg1> <arg2>)
(getangle [<nokta>] [<ileti>))
(getcorner <nokta> [<ileti>])
(getdist [<nokta>] [<ileti>))
(getint [<ileti>))
(getkword [<ileti>])
(getorient [<ileti>))
(getpoint [<nokta>] [<ileti>])
(getreal [<ileti>])
(getstring [<T>] [<ileti>])
(getvar <değişken adı>)
(graphscr)
(if <koşul> <doğruysa yap> [<yanlışsa yap>])
(initget [<bit değeri>) [<anahtar sözcük>])
(inters <nok1> <nok2> <nok3> <nok4> [<T>] [<nil>])

(itoa <tamsayı>)
(lambda <arg> <fonk>...)
(last <liste>)
(length <liste>)
(list <deyim>...)
(listp <eleman>)
(load <dosya>)
(log <arg>)
(logand <arg> <arg>...)
(logior <arg> <arg>...)
(lsh <arg> <kaç bit>)
(mapcar <fonk> <liste1>...<listeN>)
(max <arg1> <arg2>...)
(member <arg> <liste>)
(menucmd <dizgi>)
(min <arg1> <arg2>...)
(minusp <arg1>)
(not <arg>)
(nth <n> <liste>)
(null <arg>)
(numberp <arg>)
(open <dosya> <mod>)
(or <arg> <arg>...)
(osnap <nokta> <mod>)
pi
polar.<nokta> <açı> <uzaklık>)
(prin1 <arg> [<dosya>])
(princ <arg> [<dosya>])
(print <arg> [<dosya>])
(progn <arga <arg>...)
(prompt <ileti>)
(quote <arg>)
(read <dizgi>)
(read-char (<dosya>])
(read-line [<dosya>))
(redraw [<şekiladı> [<mod>]])
(rem <arg1> <arg2>...)
(repeat <sayı> <fonk> <fonk>...)
(revers <liste>)
(rtos <arg> [<mod> [<hassasiyet>]])
(set <değişkenadı> <arg>)
(setq <değişkenadı> <değer>...)
(setvar <sistemdeğişkeni> <değer>)
(sin <açı>)
(sqrt <arg>)
(strcase <dizgi> [<T>])
(strcat <dizgi1> <dizgi2>...)
(strlen <dizgi>)
(subst <yeni> <eski> <liste>)
(substr <dizgi> <başlangıç> <uzunluk>)
(terpri)
(textscr)
(trace <fonk>...)
(type <arg>)
(untrace <fonk>...)
(ver)

(while <koşul> <fonk>...)
(write-char <asciikod> [<dosya>])
(write-line <dizgi> [<dosya>])
(zerop <arg>)
(*error* <dizgi>)

SEÇİM SETİ FONKSİYONLARI
(ssget [<mod>] [<nokta1> [<nokta2>]])
(sslength <set>)
(ssname <set> <arg>)
(ssadd [<isim> [<set>]])
(ssdel <isim> <set>)
(ssmemb <isim> <set>)

AutoLISP HATA MESAJLARI

SİSTEM DEĞİŞKENLERİ

GİRİŞ

Asıl anlamda LISP, yapay zeka çalışmalarında kullanılan bir program dilidir. List Processing 'in (Liste İşleme) kısaltılmış ifadesidir. AutoLISP ise LISP'in AutoCAD ile kullanılabilecek şekilde uyarlanmış halidir. AutoLISP sayesinde kullanıcının AutoCAD'e yeni komutlar eklemesi, kişiselleştirmesi ve ondan artan bir verim elde etmesi mümkündür. Tabii ki yeni komutlar'dan kastedilen, kullanıcının AutoLISP fonksiyonlarını kullanarak hazırladığı program dosyalarnı AutoCAD ortamından çağırarak kullanmasıdır.

AutoLISP dosyalarının ASCII dosyalar yaratabilen bir kelime işlemcide (text editor) hazırlanması ve uzantısının *.LSP olması bir zorunluluktur.

AutoLISP dosyaları aslında fonksiyonlardan meydana gelmektedir. Kullanıcı, bir takım standart fonksiyonları kullanarak veya kendisi çeşitli fonksiyonlar tanımlayarak yapmak istediklerini gerçekleştirir. Gerek standart fonksiyonlar gerekse kullanıcı tanımlı fonksiyonlar, değişkenlere değerler atanması, bu değerlerin AutoLISP tarafından değerlendirilerek sonuçlar elde edilmesi mantığına göre çalışır. AutoLISP programlarda büyük küçük harf ayrımı yapmaz.

Bu kitapta < > işaretleri arasında yeralanlar fonksiyona yazılacak değişkenleri .... ise ardışık girişleri temsil eder. <arg>, fonksiyona gerekli olan bir değer veya değeri önceden atanmış bir değişken olabilir.

AutoLISP DOSYALARININ YAPISI

Bir LISP dosyası içinde tüm fonksiyonlar bir sol parantez "(" i1e başlar ve bir sağ parantez ")" ile biter. Parantezler kuralına uyuldukça bir fonksiyon içinde başka alt fonksiyonlar da bulunabilir. AutoLISP dosyalarının i1k satırları daima *(defun* ifadesiyle başlar. Bu AutoLISP'in en temel fonksiyonudur. defun fonksiyonu kullanılmadan AutoLISP programı yazılamaz. AutoLISP, fonksiyonların veya değişkenlerin yazılması veya derlenmesi sırasında büyük küçük harf ayrımı yapmaz.

Örn:

```
(defun fonk.adı ()
        (sub.fonk1 (sub.fonk2))
)                                   ; parantez sayısına dikkat ediniz.
```

Yukandaki örnekte görüldüğü gibi fonksiyon (defun ile başlamıştır. Sub.fonk olarak tanımlanan fonksiyonlar ana fonksiyon içinde kullanılan ve işlemleri yapan alt fonksiyonlardır. Bu alt fonksiyonlar genellikle, ileride anlatılacak olan standart LISP fonksiyonları veya kullamcıların tanımladığı fonksiyonlardır. Daha önce de belirttiğim gibi LISP dosyalarında dikkat edilmesi gereken en önemli hususlardan bir tanesi parantezlerdir. Özetle diyebilirim ki bir AutoLISP dosyasında "AÇILMIŞ OLAN PARANTEZ KADAR PARANTEZİN PROGRAM AKIŞINA UYGUN YERLERDE KAPATILMASI ZORUNLUDUR".

Yazılan AutoLISP dosyaları AutoCAD ortamına çağrıldığında (bu işlem daha sonra detaylı olarak anlatılacak) daha önce de belirttiğim gibi AutoLISP tarafından değerlendirmeye alınır. Bu değerlendirme, dosya AutoCAD ortamında çalıştırılmaya başlanmadan hemen önce yapılır. AutoLISP değerlendiricisine EVULATOR adı verilir. Değerlendirme esnasında ekranda Command: alanında ;

n>

ifadesi görülürse (n bir tamsayıdır) n kadar sağ parantezin eksik olduğu anlaşılır. Yani açılan sol parantezler içinde n tanesi sağ parantez kullanılarak kapatılmamıştır. Bu hatayı düzeltebilmek için LISP dosyanıza dönerek n tane sağ parantezi uygun yerlere koymanız gerekir.

Bazen programın başında açılmış olan bir sol parantez programın sonlarına doğru kapatılabilir (Yukandaki örnekte olduğu gibi). Tabii ki bu parantezlerin yeri fonksiyonun işlevi ile bağlantılıdır.

Fonksiyonlar içindeki ifadeler birden fazla satıra taşabilir ;

```
(prompt "BU YAZI BIR SATIRDAN FAZLA OLDUGU ICIN IKINCI SATIRA DA DEVAM
        EDEBILIR. ONEMLI OLAN ACILMIS SOL PARANTEZLERIN SAG
        PARANTEZLER ILE KAPATILMASIDIR"
)
```

Fonksiyonlar veya ifadeler arasında bir aralık boşluk bırakılır. Birden fazla olan boşluklar tek bir boşluk olarak kabul edilir. Bu, özellikle program satırlarının gerektiğinde içeriden yazılarak program dosyasının anlaşılabilirliğini sağlar (yukandaki ve aşağıdaki örneklere bakınız).

Defun fonksiyonu ile program yazımına başlandıktan hemen sonra bu fonksiyon bir sağ parantez ile kapatılmaz. (defun ile başlayan ve ardından fonksiyon adının yazıldığı bu fonksiyon içinde, yapılacak işlemleri tanımlayan alt fonksiyonlar yeralır.

Örn;

```
(defun toplama ()
        (setq A (getint "BIRINCI SAYIYI GIRINIZ :"))
        (setq B (getint "IKINCI SAYIYI GIRINIZ :"))
        (setq C (+ A B))
        (princ "\nSONUC :")
        (princ C)
        (princ)
)
```

Yukandaki örnekte ilk satır fonksiyonun başlangıç satırıdır. İkinci satır ile kullanıcıdan bir sayı girmesi istenir. Girilecek sayı tamsayıdır. Bu sayı A değişkenine atanır. Üçüncü satırda ikinci tamsayı istenir. Bu sayıda B değişkenine atanır. Dördüncü satırda ise girilmiş olan iki tamsayı toplama işlemine sokulur ve sonuç C değişkenine atanır. Beşinci satırda belirtilmiş olan fonksiyon sayesinde üçüncü satırda elde edilmiş olan sonuç ekranın komut alanına yazdırılır. Son satırdaki parantez ile de ilk satırda açılmış olan parantez kapatılır.

Bu örnekte AutoLISP'in standart fonksiyonlarından olan **setq** ve **princ** fonksiyonlarının nasıl kullanılabileceğini görmüş olduk. Bu fonksiyonlar ileride detaylı olarak anlatılacaklardır.

KULLANILAN VERİ TİPLERİ

AutoLISP programlarında kullanılabilecek veri tipleri aşağıdaki gibi sıralanabilir.

- Tamsayılar,
- Reel sayılar,
- Yazı dizileri (strings),
- AutoCAD Değişkenleri,
- AutoCAD Objeleri (entities),
- AutoCAD ortamından seçilmiş objeler.

Tamsayılar, -32768 ile +32767 arasında olabilir.
1 17 -378 65 24572 -793 350

Reel sayılar,
4.982 -8.015 3.14 225.68 -114.5 0.45

şeklinde gösterilen sayılardır. Ree1 sayıların fonksiyonlar içinde kullanılırken 1'den küçük değerlere sahiplerse ondalık noktanın soluna 0 'ın yazılması program akışı içinde meydana gelebilecek karışıklıkları önleyecektir.

Hatalı	Doğru
.25	0.25
.1	0.1
.285	0.285

Yazı dizileri, tırnak içinde yazılması gereken karakter dizileridir. Diziler herhangi bir uzunlukta olabilir. Bellekte yerleri dinamik olarak ayrılır.

"BIRINCI NOKTA"
"uzaklık"
"Ad Soyad" gibi

Listeler, değişkenlerden oluşan gruplardır. ( ) içinde yazılırlar. Değişik amaçlar için kullanılabilirler. En fazla kullanım alanı nokta koordinatlarının tanımlanmasındadır.

(X Y) (X Y Z)
(5 13) (10 4 0)
(3.1 0) (2.75 0.0 8.2)

AutoLISP'de çok kullanılan iki fonksiyon vardır. Bunlardan biri pi fonksiyonu diğeri de nil fonksiyonudur. Pi fonksiyonunun değeri önceden belirlenmiştir. Bu değer 3.1415926 'ya eşittir. Yazdığınız programlarda bu değeri kullanmak isteğinizde pi yazmanız yeterli olacaktır.

Nil fonksiyonu "hiç" değerine eşittir. Yani boştur. Program içinde setq fonksiyonu kullanılarak değer atanmamış tüm değişkenlerin değeri nil'dir. Setq fonksiyonu kullanılarak değer atanmış bir değişkene programın daha sonraki satırlarında yine setq fonksiyonu kullanılarak nil değeri atanırsa, değişkenin önceki değeri boşaltılmış, değişken nil'e eşitlenmiş olur. Prompt gibi fonksiyonlar da görevlerini yaptıktan sonra nil değerini verirler.

DEĞİŞKENLER ve SETQ FONKSİYONU

Değişkenler, SETQ fonksiyonu kullanılarak değerlerin. atandığı karakter veya karakterler dizisidir. İlk harfi bir karakter olmak şartıyla değişken isimleri içinde tamsayı veya reel sayı olabilir. Değişkenlere verilecek isimler istenilen uzunlukta olabilir. Büyük küçük harf ayrımı yapılmaz. Ancak değişken ismi içinde boşluk karakteri bulunamaz.

Kullanılabilecek değişken isimlerine aşağıdakileri örnek verebiliriz.
ABX YCAP nokta1 kabin5a

Geçersiz değişken isimlerine örnekler ise ;
12A 1,Z 1nokta TAM SAYI

Değişken isimlerinin seçimi kurallara uymak şartıyla kullanıcının isteğine göre değişebilir. Değişken isimleri genellikle kendisine atanan değeri tanımlayacak şekilde verilir. Bu, daha sonraki fonksiyonlarda hangi değişkenin hangi değere veya fonksiyona ait olduğunun hatırlanması bakımından kolaylık sağlar.

(setq <değişkenadı> <değer>.....)

Örnekler

```
(defun toplama ()
        (setq tamsay1 (getint "BIRINCI SAYIYI GIRINIZ :"))
        (setq tamsay2 (getint "IKINCI SAYIYI GIRINIZ :"))
        (setq toplam (+ tamsay1 tamsay2))
        (princ "\nSONUC : ")
        (princ toplam)
        (princ)
)

(setq A C)
(setq ODA5 2.678)
(setq ISIM "ENDER CIKIS")
```

Yukandaki son örnekte görüldüğü gibi ISIM değişkenine ENDER CIKIS yazı dizisi atanmıştır. Burada dikkat edilmesi gereken değişkenlere atanacak yazı dizilerinin veya ekranda yazdırılacak iletilerin " " içinde yazılmasıdır ("BIRINCI SAYIYI GIRINIZ" gibi).

Bir tek setq fonksiyonu içinde birden fazla değişkene değer ataması da yapılabilir.

```
 (setq A 6
       ODA5 2.678
       ISIM "ENDER CIKIS"
)
```

veya

```
(setq A 6 ODA5 2.678 ISIM "ENDER CIKIS")
```

gibi.

Örneklerde görüldüğü gibi bir tek setq fonksiyonu kullanılarak, aynı fonksiyon içinde A, ODA5 ve ISIM değişkenlerine aynı anda değer ataması yapılmıştır.

Değişkenlere nokta koordinatlarının atanması biraz daha farklıdır. Nokta koordinatları parantez içinde yazılmalıdır. İki boyutlu bir nokta için parantez içine o nokta koordinatının X ve Y değerleri, üç boyutlu bir nokta içinse o nokta koordinatının X, Y ve Z değerleri yazılır.

Örn:

(X Y) → (5 8) (2.25 6.3)

(X Y Z) → (5 8 3) (4 7.45 0) gibi

Bu nokta koordinatları, bir değişkene atanarak işleme sokulmak istendiğinde " LIST " fonksiyonunun kullanılması gerekir. Bu fonksiyon bir liste tanımlaması yapar. *(Bkz. LIST FONKSİYONU)*

Örn 1:

```
(setq POINT1 (list 3.45 8))
```

Bu fonksiyon POINT1 noktası için noktanın X değerinin 3.45 Y değerinin ise 8 olduğunu belirtir.

Örn 2:

```
(setq P1 4)

(setq P2 (list 3.45 P1))
```

Örnek 2 de önce P1 değişkenine 4 değeri atanıyor. İkinci satırda P2 değişkenine bir liste atanıyor. Bu listenin ilk elemanı 3.45 sayısı ikinci elemanı ise P1 değişkeninin değeridir. Yani P2 noktasının koordinatının X değeri 3.45 iken Y değeri P1'e atanmış değerdir.

Aşağıdaki örnekte ise K değişkenine nil (boş) değeri atanmaktadır. Yani K değişkenine daha önce atanmış bir değer varsa bu değer iptal edilmiş olur.

```
(setq K nil)
```

AutoLISP programlarının yazılmaları esnasında açıklamalar yazılması gerektiğinde bu açıklamalar " ; " (noktalı virgül) ile başlar. Noktalı virgül satır başında konmuş ise satırın tamamı açıklama olarak kabul edilir ve işleme konnıaz. Noktah virgül, satır başmda değil de işleme konan bir program satırmda ise noktah virgülden sonraki kısım işleme konmayarak açıklama olarak kabul edilir.

Kısaca noktalı virgül Basic dilindeki açıklama satırlarının başına konan REM komutuna benzer.

Aynca AutoLISP fonksiyonları bölümünde setq fonksiyonuna da bakınız. *(Bölüm 3)*

ÖZEL KURALLAR

- İfadeler ve fonksiyon isimleri içinde ( ). ' ; " gibi karakterler bulunamaz.
- Deyimler ve ifadeler sığmadıkları takdirde birden fazla satıra taşabilir.
- İfadeler arasında birden fazla olan boşluk karakterleri tek bir boşluk olarak kabul edilir.
- Değişkenler ve fonksiyon isimleri bir rakam ile başlayamaz. Değişken isimleri arasında boşluk karakteri kullanılamaz.
- Reel tamsayılar bir veya daha fazla tamsayı, ardından bir nokta ve sonra da bir yada daha fazla ondalık değer içerebilir. Örneğin 0.5 veya 0.8 aynen yazılmalıdır. Eğer .5 veya .8 olarak yazılırsa program tarafindan kabul edilmezler.
- Bir değişkene atanan değer, o değişkene yeni bir değer atanıncaya veya AutoCAD ortamından çıkılıncaya kadar korunur.
- Ekranın komut alanında görüntülenmesi istenen bir yazı, " " içine yazılmış olmadır (prompt örneğine bakınız).
- Bir AutoCAD iletisine bir lisp deyimi ile yanıt verilmek istendiğinde bu yanıtın ilk karakteri "!" olmalı veya ( ) içinde yazılmalıdır.
- Tırnak işaretleri içine konacak \ işareti özel kontrol karakterlerinin kullanılmasına olanak sağlar. Bu kontrol karakterleri ;

\\	\ Karakteri
\e	Escape
\n	Satırbaşı (next line)
\r	Return (Enter)
\t	Tab
\nnn	octal kodu nnn olan karakter
	olarak tanımlanmıştır.

Yukarıda açıklanan kontrol karakterleri içinde " \ " karakterine dikkat ediniz. Bildiğiniz gibi Dos işletim sisteminde sürücü ve dizin adresleri belirtilirken dizinler arasına " \ " karakteri yazılmaktadır. Ancak AutoLISP dosyaları içinde sürücü veya dizin adresleri belirtilmesi gerektiğinde " \ " karakteri " \\ " şeklinde kullanılmalıdır.

İstenirse " \\ " yerine, aynı görevi yapan " / " karakteri de kullanılabilir.

(load "c:/deneme/test.dat")

(load "c:\\deneme\\test.dat")

Satırbaşı (\n) karakteri için prompt örneğimize dönersek ;

```
(prompt "\nBU YAZI BIR SATIRDAN FAZLA OLDUGU ICIN IKINCI")
(prompt "\nSATIRA DA DEVAM EDEBILIR. ONEMLI OLAN ACILMIS")
(prompt "\nSOL PARANTEZLERIN SAG PARANTEZLER ILE")
(prompt "\nKAPATILMASIDIR.")
```

Yukandaki yazıların herbiri ekranda komut alanında yeni bir satıra yazacaktır.

DOSYALARIN YÜKLENMESİ

AutoCAD ortamına, ekranda Command : iletisi varken aşağıdaki satır yazıldığında "dosyaadı" olarak belirtilen AutoLISP dosyası yüklenir.

(load "dosyaadı") şeklinde olur.

Eğer dosya bir diskette veya çalışılmakta olan directory'den başka bir yerde ise dosya adı adresiyle birlikte yazılmalıdır.

Örn:

(load "<drive>:/<directory>/dosyaadı")

Bir sol parantez içinde yazılan Load ifadesinden sonra tırnak içinde yüklenecek *.LSP uzantılı dosyanın adı yazılır. Geliştirilen AutoCAD versiyonlarında LISP dosyalarının yükleme işlemi kolaylaştırılmaktadır. Örneğin AutoCAD R.12 de LISP dosyalarının işlemi Pulldown Menu'deki Files menü başlığında yeralan Applications butonu sayesinde kolaylıkla yapılabilmektedir.

Aşağıda, girilen açı değerini radyan cinsine çeviren bir fonksiyon yeralmaktadır. Fonksiyonu, ASCII forınatta dosya saklayabilen bir kelime işlemcide yazuı. Saklarken ASCII fornıatta olmasına ve uzantısıiiın *.LSP olmasına dikkat ediniz. Bu öıneğimizin saklandığı LSP uzantılı ASCII dosyamı□ın acı.lsp olduğunu varsayalım. Fonksiyonun yazılışına ve AutoCAD ortamına çağrılarak çalıştıntmasına dikkat ediniz ;

Fonksiyonun yazılması ;

```
 (defun cevir (a)  ;fonksiyonu tanımla, fonksiyonun adı cevir
          (* pi (/ a 180.00)) ;a değerini 180'e böl, pi ile çarp
)
```

AutoCAD ortamına çağrılması ;

```
(load "aci")
```

Çalıştırılması ;

```
(cevir 180)
```

Program bu işlemin sonucu olarak 3.1415926 yazacaktır. Bu 180 derecenin radyan cinsinden karşılığıdır. Bu arada şunu da hatırlatmakta yarar var : AutoLISP'de açılar radyan cinsinden hesaba konur. Radyan cinsinde açılar

O^o = 0
90^o = 1.570796
180^o = 3.141592
270^o = 4.7I2389
değerlerine sahiptir.

Dikkat edilirse buraya kadar olan fonksiyon örnekleri, en son anlatılan konu ışığında

(load "dosyaadi")

şeklinde çağrılmakta. Ancak bu fonksiyonları bir AutoCAD komutu gibi çağırmak da mümkün. Aşağıdaki örneği inceleyiniz.

```
(defun c:cevir (/ a)
        (setq a (getint "TAMSAYI ACI DEGERINI GIRINIZ : "))
        (setq a (* pi (/ a 180.00)))
        (princ "\nSONUC : ")
        (princ a)
        (princ)
)
```

Dikkat edilirse fonksiyonda yaptığımız en önemli değişiklik, fonksiyon adı olan cevir'den önce ***c:*** yazmak oldu. Yazdığımız bu program AutoCAD ortamına yüklendiğinde yükleme işlemi tamamlanır tamamlanmaz ekranda c: cevir iletisini görüntüler. Burada kullanılan c: işletim sisteminde kullanılan ve sürücüyü tanımlayan c: anlamında
değildir. Bu iletiden sonra ekranda Command: iletisi varken ***cevir*** yazıldığında program tamsayı olarak açı değerinin girilmesini ister. Değer girildikten sonra da bu değerin radyan cinsinden karşılığını yine ekrana yazar.

Bu şekilde yazıldıktan sonra AutoCAD ortamına yüklenmiş olan bir LISP program dosyası AutoCAD ortamından çıkılana kadar tıpkı bir AutoCAD komutu gibi çalışır. AutoCAD'in her yüklenişinde bu tür fonksiyonların komut gibi çalıştırılması istenirse bu fonksiyon adlarının acad.lsp dosyasına ilave edilmesi gerekir. Acad.lsp dosyası AutoCAD'in her çalıştırılışında otomatik olarak yüklenen (çağrılan) bir dosyadır.

AutoLISP 'in EN TEMEL FONKSİYONLARI

Bu bölümde AutoLISP'in en temel fonksiyonu olan ***defun*** fonksiyonundan ve AutoLISP programlarının sonuçlarının AutoCAD tarafından kullanılabilmesini sağlayan ***command*** fonksiyonundan söz edilecek.

AutoLISP dosyalarının yapısının anlatıldığı bölümde de açıklandığı gibi defun fonksiyonu kullanılmadan bir AutoLISP programı yazılamaz.

.
.
.
.
.
.
.
.

AutoLISP FONKSİYONLARI

Bu bölümde öncelikle aritmetik işlem fonksiyonları üzerinde duracağız. AutoLISP programlarında, toplama, çıkarma, çarpma ve bölme işlemleri bu işlemleri yapan standart fonksiyonların kullanılmasıyla yapılır.

Bu fonksiyonları kullanım formatı ;

(fonk <arg1> <arg2> .....) şeklindedir.

Örnekler :

(- A cap1)	A - cap1	; anlamındadır.
(+ 50 H)	50 + H	; anlamındadır.
(/ YUKSEKLIK 4)	YUKSEKLIK / 4	; anlamındadır.
(* pi 2)	pi * 2	; anlamındadır.

Fonksiyon içinde ikiden fazla argüman işleme sokulduğunda işleme tüm argümanlara alınır. Örneğin bu fonksiyon bir çıkarma işlemi yapan bir fonksiyonsa sıra ile, arg1 'den arg2 çıkarılır, elde edilen sonuçtan arg3 ve yine elde edilen sonuçtan arg4 çıkartılır ve işlem bu şekilde fonksiyondaki tüm argümanlara uygulanır.

☞ (- A b cap1 10.5)
A-b-cap1-10.5 ; anlamındadır.

örneğin :
(- 50 10 25 10.5)
50-10-25-10.5=4.5 ; sonucunu verir.

☞ (+ A b cap1 10.5)
A+b+cap1+10.5 ; anlamındadır.

örneğin :
(+ 50 10 25 10.5)
50+ 10+25+ 10.5=95.5 ; sonucunu verir.

☞ (* pi 2 ycap)
pi * 2 * ycap ; anlamındadır.

örneğin :

(* pi 2 3)
pi*2*5 = 18.8495556 ; sonucunu verir.
(pi = 3.1415926 hatırlayınız)

☞ (/ YUKSEKLIK 4 P5)
YUKSEKLIK / 4 / P5 ; anlamındadır.

ömeğin :

(/ 80 4 2)
80 / 4 / 2 = 10 ; sonucunu verir.

Aynı program satırı içinde birden fazla işlem fonksiyonu ardışık olarak kullanılabilir.

(* pi (/ a 180.00))
pi * a / 180.00 ; anlamındadır.

Bu tür kullanımlarda önce en içteki parantez içinde yeralan işlem yapılır. Yukarıdaki örneğe göre, önce a değeri 180.00'e bölünecek, çıkan sonuç pi ile çarpılacaktır.

(/ 20 (* (+ (- A B) n1) 5))

Yukarıdaki satırda yeralan işlemlerin sırası ise ;

- A 'dan B 'yi çıkar,
- çıkan sonuçla n1 'i topla,
- çıkan sonuçla 5 ‘i çarp,
- 20 'yi çıkan sonuca böl.

.
.
.
.
.
.
.
.

SEÇİM SETİ FONKSİYONLARI

AutoCAD ortamında çalışırken, kullanıcının en çok yaptığı şeylerden biri obje seçmektir. Özellikle hemen hemen bütün Edit komutları işlem yapmaya başlamadan önce işlemin uygulanacağı objenin seçilmesini ister. Bunun sebebi, işlemin uygulanacağı obje hakkındaki mevcut bilgilerin veri tabanından alınabilmesidir. Change komutunu kullanarak ekranda var olan bir objenin değiştirmek istediğinizde öncelikle bu objenin seçilmesi istenir.

Hazırlanan LISP programlarında obje seçimi gerektiren durumlarda Seçim Seti Fonksiyonları kullanılarak seçilmiş objelerden oluşan bir grup meydana getirilebilir. Bu seçim grubundaki obje sayısı öğrenilebilir, sete yeni objeler ilave edilebilir. Bunların dışında, Command fonksiyonu vasıtasıyla AutoCAD komutları bu seçim gıubundaki objeler için kullanılabilir.

AutoCAD objeleri hakkındaki bilgilere objeleri seçmek suretiyle ulaşılabileceğini belirtmiştir.Obje seçimi yapıldığında objeler hakkındaki bilgiler AutoCAD veri tabanından alınır. AutoCAD çizim ortamına katılan her objenin özellikleri

<Entity name: 60000014>

şeklinde kayıt numarasıyla (aşağıdaki fonksiyonlarda <isim> olarak geçecek) veri tabanına kaydedilir. Kullamcı AutoCAD ortamında çalışırken mevcut objelerden birinin herhangi bir özelliğini (örn. rengini) değiştirdiğinde AutoCAD, veri tabanına ulaşarak özelliklerinde değişiklik olan objenin veri tabanındaki kayıtlarını yeniden düzenler.

Aşağıdaki fonksiyonlardan *ssname fonksiyonu* seçilen objenin isminin (kayıt no) öğrenilebilmesini sağlar.

(ssget (<mod> [nokta1> [<nokta2>]])

Bu fonksiyon bir seçim grubu elde edilmesinde kullanılır. [ ] içinde yazılmış ve belirtilmeleri isteğe bağlı olan değişkenlerin hiçbirinin kullanılmaması durumunda fonksiyon obje seçimini kullanıcının yapmasını ister. Bu AutoCAD'in standart "Select object :" iletisi ile yapılacak obje seçimini gerçekleştirir.

<mod>, AutoCAD'in standart obje seçiminde kullandığı window, crossing, previous gibi seçim modlarından biridir.

Mod	*Anlamı*
w	Window yöntemi. Tanımlanan pencere içine tamamı giren objeler seçilir.
"c"	Crossing yöntemi. Tanımlanan pencere içine tamamı giren objeleı1e birlikte pencerenin herhangi bir kenarına en küçük noktası dahi temas eden objeler seçilir.
"L"	Last yöntemi. Çizim ortamınn katılan en son obje seç1lir.
"p"	Previous yöntemi. Bir önceki obje seçme işlemi sırasında seçilmiş olan objeler seçilir.

Belirtilmesi isteğe bağlı olan <nokta1> ve <nokta2> "w" veya "c" yöntemi kullanıldığında açılacak olan pencerenin karşılıklı iki köşe noktasının verilmesinde kullanılır.

Sadece <nokta1> verildiğinde verilen bu noktanın üzerinde olduğu obje seçilir.

Örnekler :

(ssget)	seçimi kullanıcının yapması beklenir.
(ssget "w" '(3 3) '(7 7))	Karşılıklı köşe noktaları 3,3 ve 7,7 koordinatlarında olan pencerenin içine tamamı giren objeler seçilir.
(ssget "c" '(3 3) '(7 7))	Karşılıklı köşe noktaları 3,3 ve 7,7 koordinatlarında olan pencerenin içine tamamı giren objelerle birlikte pencerenin herhangi bir kenarına en küçük noktası dahi temas eden objeler seçilir.
(ssget "p")	Bir önceki obje seçme işlemi sırasında seçilmiş olan objeler seçilir.
(ssget "L")	Çizim ortamına katılan en son obje seçilir.
(ssget '(5 5))	5,5 koordinatından geçen obje seçilir.

Fonksiyon sadece herhangi bir parametre olmadan (ssget) şeklinde kullanıldığında kullanıcının seçtiği objeler aydınlatılır.

Örnek :

```
(defun c:sil (/ p1 p2 s1)
      (setq p1 (getpoint "\nPENCERENIN BIRINCI NOKTASI :"))
      (setq p2 (getcorner "\nDIGER NOKTASI :"))
        (setq s1 (ssget "w" p1 p2))
        (command "erase" s1 "")
        (princ)
)
```

Yukardaki örnek program, açılacak pencere içine tamamı giren objeleri seçtikten sonra AutoCAD'in ERASE komutunu kullanarak siler.

(sslength <set>)

Bu fonksiyon bir seçim grubu içinde bulunan objelerin sayısını verir. Bu sayı bir tamsayıdır. Seçim grupları hiçbir zaman aynı şeklin ikinci kez seçimini bulundurmaz. <set>, seçim grubunu belirten değişkendir,

Örnek :

```
(defun c:objesay (/ p1 p2 obje s1 sayi)
      (setq p1 (getpoint "\nPENCERENIN BIRINCI NOKTASI :"))
      (setq p2 (getcorner p1 "\nDIGER NOKTA :"))
      (setq obje (ssget "w" p1 p2))          ; objeleri seç, obje'ye ata
      (if (/= obje nil)                      ; obje değişkeni boş değilse
        (progn                               ; bunları yap
            (setq s1 (sslength obje))        ; obje sayısını öğren, s9' ata
            (setq sayi (itoa s1))            ; s1 'in dizgiye çevirsayı'ya ata
            (princ (strcat "\nSECIM KUMENIZDE" sayi "ADET OBJE VAR") )
        )
        (princ "\nSECILEN OBJE YOK")     ; obje değişkeni boşsa bunu yap )
      )
      (princ)
)
```

(ssname <set> <arg>)

Bu fonksiyon, <set> belirtilmiş olan seçim grubu içindeki <arg> 'ıncı objenin ismini sonuç olarak verir. Fonksiyona sağlanan <arg> değeri bir tamsayı olmalıdır. Eğer <arg> Tamsayı negatif bir sayı yada seçim grubunda yeralan obje adedinden fazla ise nil sonucu elde edilir. Seçim grubundaki ilk eleman için <arg> değeri 0 (sıfir)'dır.

Örnek :

```
(defun c:objeisim (/ p1 p2 ob s1 sayi)
      (setq p1 (getpoint "\nPENCERENIN BIRINCI NOKTASI :"))
      (setq p2 (getcorner p1 "\nDIGER NOKTA :"))
      (setq ob (ssget "w" p1 p2))          ; objeleri seç, ob 'a ata
      (if (/= ob nil)                      ; ob değişkeni boş değilse
          (progn                           ; bunları yap
              (setq s1 (sslength ob))      ; obje sayısını öğren, s1' e ata
              (setq sayi (- s1 1))         ; s1'i 1 eksilt sonucu sayı'ya ata
              (ssname ob sayi)             ; ob'daki sayı'ncı objenin adını ver
          )                                ; obje değişkeni boşsa hiçbirşey
      )                                    ; yapma
)
```

(ssadd [<isim> [<set>]])

Bu fonksiyon, verilmesi isteğe bağlı olan parametreler belirtilmeden kullanıldığında ssget fonksiyonu gibi çalışacak fakat oluşturulacak seçim setinin hiç elemanı bulunmayacaktır.

Fonksiyon <isim> olarak bir obje ismi verilerek çalıştırıldığında, sadece bu <isim> 'deki objeden oluşan yeni bir seçim seti meydana getirecektir.

Her iki parametre de kullanılarak, yani hem obje ismi <isim> , hemde önceden oluşmuş bir seçim setinin adı <set> verilerek çalıştırıldığında ise <isim> olarak belirtilen objeyi <set> olarak belirtilen seçim setine ilave edecektir.

SSADD fonksiyonu sonuç olarak herzaman yeni seçim setini verir. Eğer mevcut <set> bir değişkene atanmışsa değişkenlerde bu düzenlemeden etkilenecek ve yeni seti kullanacaklardır.

Bunlarla birlikte, eğer <isim> olarak belirtilmiş olan obje <set> olarak belirtilmiş seçim setinde zaten mevcut ise fonksiyon kendini otomatik olarak iptal edecek ve bununla ilgili olarak herhangi bir hata oluşmayacaktır.

(ssdel <isim> <set>)

SSDEL fonksiyonu, <isim> olarak belirtilmiş ismin ait olduğu şekli <set> olarak belirtilmiş olan seçim setinden siler. Sonuç olarak da seçim setinin adını verir. Bu işlem sonucunda elde edilecek olan seçim seti ssadd fonksiyonunda olduğu gibi yeni bir seçim seti olacaktır.

Eğer belirtilen <set> içinde <isim> yoksa fonksiyon sonucu nil olacaktır.

(ssmemb <isim> <set>)

SSMEMB fonksiyonu, <isim> olarak belirtilmiş ismin ait olduğu şekli <set> olarak belirtilmiş olan seçim setinin bir elemanı olup olmadığını araştırır. Eğer <isim>, belirtilen <set> 'in bir elemanı ise sonuc olarak bu objenin ismi verilir. Aksi durumda ise fonksiyon sonucu nil olur.

AutoLISP HATA MESAJLARI

AutoLISP programları, yazılmaları sırasında yapılan testlerde veya programıu son şeklini aldığı düşünülerek çalıştırıldığında çeşitli hataların meydana gelmiş olması mümkündür. Özellikle programın yazılma aşamasında gözden kaçan bir parantez veya bir (") programın çalışmaması hatta yüklenememesi için yeterli bir sebeptir.

Bu bölümde, AutoLISP hata mesajları ele alınacaktır. Eğer kullanıcı yazdığı programında *error* fonksiyonu ile hata mesajlarının nasıl görüntüleneceğini belirtmemişse hata mesajları ;

error : hata mesajı

şeklinde görüntülenecektir.

Aşağıdaki listede, meydana gelebilecek hataların tanımları yer almaktadır.

AutoCAD rejected function

AutoCAD'in, tanımlanan fonksiyonu reddettiğini birdirir. Örneğin yanlızca okunabilir (Read Only) durumdaki bir sistem değişenine bir değer atanmaya veya Command fonksiyonu içinde herhangi bir GET???? fonksiyonu kullanılmaya çalışılıyor olabilir.

Bad argument type

Fonksiyonda uygun olmayan değişken tipi kullanılmaktadır. Örneğin CHR fonksiyonuyla bir dizgi işleme sokulmuş olabilir. Yada bir tamsayının, dizgiymiş gibi STRLEN fonksiyonu ile karakter sayısı bulunmaya çalışılıyor olabilir.

Bad association list

ASSOC fonksiyonuyla uygun olmayan bir liste kullanılmış.

Bad entmod list

ENTMOD'a verilen liste uygun birim listesi değil. (ENTGET ile sonuçlandırılan)

Bad formal argument list

Değerlendirme sırasında AutoCAD geçersiz bir formatta değişken listesi ile karşılaşmıştır. Değerlendirilen fonksiyon gerçekte bir fonksiyon değil bir veri listesidir.

Bad function

Yanlış tanımlanmış fonksiyon. Fonksiyon ismi olarak geçerli bir isim verilmemiş olabilir. Ya da fonksiyon tam olarak tanımlanmamış olabilir.

Bad list

Tam olarak oluşturulamamış bir liste bir fonksiyonda kullanılıyor olabilir. Bu durum genellikle reel bir sayının ondalık noktadan önce 0 (sıfir) olmadan .23 şeklinde kullanılmasıyla oluşur. Bu durumu düzeltmek için ondalık noktadan önce 0 konulmalıdır. Örneğin 0.23 veya 0.5 vs.

Bad node

TYPE fonksiyonu için geçersiz eleman tipi belirtilmiş.

Bad node type in list

FOREACH fonksiyonu için belirtilen listede geçersiz eleman tipi belirtilmiş.

Bad point argument

Tam olarak tanımlanmayan ve iki reel sayıdan oluşan liste bir nokta bekleyen bir fonksiyona aktarılmıştır. Reel sayı ondalık noktadan önce 0 (sıfir) olmadan yazılmış olabilir.

Bad point value

Yukarıdaki hata ile aynı hata.

Boole arg1 <0 or >15

BOOLE fonksiyonunda kullanılan ilk argüman 0 ile 15 arasında olmalı.

Can't evaluate expression

Bu hata mesajı genellikle ondalık noktanın doğru yere konulmamasından veya eksik yada yanlış tanımlanmış deyimlerden kaynaklanır.

Console break

Fonksiyonun çalışması esnasında kullanıcı tarafından Ctrl+C tuşlarına basılarak fonksiyon kırılmıştır.

Divide by zero

Programın bir yerinde bir değer 0 (sıfir)'a bölünmeye çalışılıyor.

Extra right paren

En çok rastlanan hata mesajlarından biridir. Programın herhangi bir yerinde fazladan bir sağ parantezin olduğunu belirtir.

File not open

Giriş çıkış işlemi için tanımlanan dosya açık değildir.

Function cancelled

Girdi bekleyen bir iletiye kullanıcı Ctrl+C ile yanıt vermiştir. Bu mesaj fonksiyonu çalışmasının kullanıcı tarafından kesildiğini bildirir.

Function undefined for argument

LOG veya SQRT fonksiyonlarma verilen değişkenler sınır dışında.

Function undefined for real

Bir tamsayı bekleyen değişkene bir reel sayı aktarılmış durumdadır.

Incorrect number of argument to a function

Kullanıcının tanımladığı fonksiyonda bulunan değişken sayısı ile DEFUN da tanımlanan değişkenler arasında bir uyuşmazlık var.

Insuffcient node space

İstenilen işlemin yapılabilmesi için yeterli HEAP alanı yoktur. *(Bellek Gereksinimi bölümüne bakınız)*

Insufficient string space

Belirtilmiş olan yazı dizgisini yerleştirebilmek için HEAP alanı yoktur. *(Bellek Gereksinimi bölümüne bakınız)*

Invalid argument

Hatalı argüman tanımlaması veya tanımlanan argüman sınırlar dışında kalmakta.

Invalid character

Bir deyim, uygun olmayan bir karakter içermekte.

Invalid dotted pair

"sayı-nokta-sayı" şeklinde tanımlanan liste tiplerine noktalı çiftler adı verilmektedir. Bu hata mesajı, iki reel sayının yanlış tanımlanmasıyla oluşan hatalı bir noktalı çift olduğunu belirtir. Bu durum genellikle reel bir sayının ondalık noktadan önce 0 (sıfır) olmadan .75 şeklinde kullanılmasıyla oluşur. Bu durumun düzeltmek için ondalık noktadan önce 0 konulmalıdır. Örneğin 0.75 veya 0.123045 gibi

Lispstack overflow

Bellekte STACK bölgesi olarak ayrılan yer tamamen dolmuştur. Bunun nedeni, oldukça fazla LISP fonksiyonunun tekrarı veya çok fazla fonksiyon değişkeni listesinin olmasıdır.

Misplaced dot

Eğer reel bir sayı 1'in altında bir değer sahip ise ve ondalık noktadan önce 0 (sıfır) olmadan yazılmışca bu hata mesajı ile karşılaşılabilir. Bu durumu düzeltmek için ondalık noktadan önce 0 (sıfir) konulmalıdır.

Null function

Değeri Nil olan bir fonksiyon hesaplanmaya çalışılıyor.

Quit / exit abort

Bu mesaj QUIT veya EXIT fonksiyonlarının kullanılmasının sonucunda görüntülenir. Bu fonksiyonlar AutoLISP uygulamalarında pek fazla kullanılmamaktadır.

Too few arguments

Temel iç fonksiyonlardan birine gerektiğinden az sayıda değişken atanmıştır.

Too many arguments

Temel iç fonksiyonlardan birine gerektiğinden fazla sayıda değişken atanmıştır.

SİSTEM DEĞİŞKENLERİ

Aşağıdaki listede birçok AutoCAD Sistem Değişkeni yeralmaktadır. Değişkenler ve aldıkları değerler AutoCAD çizim ortamındayken ekranın komut alanında bulunan Command : iletisine

SETVAR komutu girilerek görülebilir.

Reed Only olarak belirtilmiş değişkenler sadece okunabilen değişkenleri belirtir. Bu değişkenlerden bazılarının değerleri, programın çalışması esnasında AutoCAD tarafından otomatik olarak ayarlanır (CDATE, DATE gibi), bazıları da programın çalışması esnasında

kullanıcının verdiği değerlere göre ayarlanır (AREA, CECOLOR, CELTYPE, DIMBLOCK, DIMSTYLE, DWGNAME gibi).

AutoCAD sistem değişkenlerinin AutoLISP programlarında kullanıldıklarında değişken adlarının "" içinde yazılması gerektiğini unutmayınız.

DEĞİŞKEN İSMİ	*ANLAMI*
ACADVER *(Read Only)*	: AutoCAD Versiyon numarası.
AFLAGS	: Attdef Komutu için niteleyici işaretleri : 1= Görünmez 2= Sabit 3= Doğrulama
ANGBASE	: 0 Açı yönü.
ANGDI R	: 1= Saat yönü açıları. 0= Saatin ters yönü açıları
APERTURE	: Birimkare olarak hedef kutucuğunun boyutu.
AREA (Read *Only)*	: Area, List veya Dblist ile hesaplanmış en son alan.
ATTDIA	: 1 ise, Insert işlemi sırasında niteleyicideki değerler değiştirilebilir. 0 ise değiştirilmez. Default değeri 0 dır.
ATTMODE	: Niteleyici görüntü modu ; 0 = OFF, 1= NORMAL, 2=ON.
ATTREQ	: 1 ise, Insert işlemi sırasında niteleyicideki değerleri gösteren, istenirse yeni değerlerin girilmesine olanak sağlayan Attribute Tag ve Attribute Prompt iletileri görüntülenmez. 0 ise görüntülenir.
AUNITS	: Açısal birim kodu, 0 = Ondalık derece, 1= derece/dakika,saniye, 2 = Grad, 3 = Radyan, 4 = Topografik ölçü
AUPREC	: Açısal birim ondalık hane.
AXISMODE	: 1 ise eksen çizilir 0 ise çizilmez.
AXISUNIT	: X ve Y eksenlerinde aralık ayarı.
BLIPMODE	: 1 ise blip işareti (+) konur. 0 ise konmaz.
DATE (Read Only)	: Takvim tarih/saat (özel formatla yazılmış).
CECOLOR (Read *Only)*	: Mevcut şeklin rengi.
CELNPE *(Read Only)*	: Mevcut şeklin çizgi tipi.
CHAMFERA	: Birinci pah ölçüsü.
CHAMFERB	: İkinci pah ölçüsü.
CLAYER (Read *Only)*	: Çalışılmakta olan layer adı.
CMDECHO	: AutoLISP'in (command) fonksiyonu kullanıldığı zaman bu değişkenin değeri 1 ise ileti ve girdiler yansıtılır.
COORDS	: 0 ise koordinat görüntüsü sadece seçilen noktada güncelleştirilir.1 ise, bütün koordinatlar sürekli görüntülenir. 2 ise, son noktadan ölçülen mesafe ve açı istendiği zaman görüntülenir.
DATE *(Read Only)*	: Julian tarih/saat (Özel format).

DISTANCE (Read Only) : List veya Dblist ile hesaplanmış en son alan.
DRAGMODE : 0 ise sürükleme görünmez,1 ise istendiğinde görülebilir, 2 ise herzaman görülebilir.
DRAGP1 : Yeniden yaratma, sürükleme girdisi örnekleme oranı.
DRAGP2 : Hızlı - sürükleme örnekleme oranı.
DWGNAME (Read *Only)* : Çalışılmakta olan çizim dosyasının adı.
DWGPREFIX (Read *Only)* : Çalışılmakta olan çizim dosyasının bulunduğu sürücü/directory adı
ELEVATION : Mevcut X-Y düzlemine verilmiş olan yükseklik değeri.
EXPERT : 1 ise iletiler vurgulanır, 0 ise normal şekilde verilir.

EXTMAX *(Read Only)* : Çizimin kullandığı sağ üst kısımlar.
EXTMIN (Read *Only)* : Çizimin kullandığı sol alt kısımlar.
FILLETRAD : Birleştirici radus yarıçapı.
FILLMODE : 1 ise Fill modu açık, 0 ise kapalıdır
GRIDMODE : 1 ise Grid modu açık, 0 ise kapalıdır
GRIDUNIT : Izgara aralığının X ve Y değerleri.
HIGHLIGHT : 1 ise seçilen şekil ışıklandırılır, 0 ise ışıklandırılmaz.
INSBASE : Insert etmek için base point noktası, Base komutu ile ayarlanır.
LASTANGLE (Read *Only)* : Girilen en son yayın bitirme açısı.
LASTPOINT : Girilen en son nokta koordinatı.
LIMCHECK : 1 ise limitler dışına çizim yapılamaz.
LIMMAX : Limitlerin sağ üst köşe noktasının koordinatları.
LIMMIN : Limitlerin sol alt köşe noktasının koordinatları.
LTSCALE : Çizgi tipi ölçeği.
LUNITS : Birim modu,
1 ise Fen işlemleri birim sistemi,
2 ise ondalık sistem,
3 ise mühendislik,
4 ise mimari
5 ise Rasyonel (kesirli) birim sistemi.
LUPREC : Lineer birimlerin virgülden sonraki basamak sayısı
MENUECHO : 0= menü girdisi yansıtılır ve iletiler normal şekilde görüntülenir.
1= menü girdisi yansıtılmaz (^P yansıtmayı başlatır).
2= girdi bir menü elemanından ise iletiler vurgulanır.
3= menü yansıtması ve iletiler vurgulanır.
MIRRTEXT : 0 ise, yazı yansıtıldığı zaman durumunu aynen korur.
1 ise, yazı yansıtıldığında tersten yazılır.
ORTHOMODE : 1 ise ortho modu açık,
0 ise kapalıdır.
OSMODE : Şekil yerleştirme modlarınn bit-kodu
1 = Uç nokta,
2= Orta nokta,
4= Merkez,
8= Düğüm,
16= Kadran,
32= Kesişim,
64= Araya ekleme,
128= Dik,
256= Tanjant,
512= Enyakın,
1024= Hızlı.

PDMODE : Nokta çeşidinin numarası.
PDSIZE : Nokta büyüklüğü.
PERIMETER *(Read Only)* : Area, List veya Dblist ile hesaplanmış en son çevre uzunluğu.
PICKBOX : Obje seçimi hedef kutucuğunun birimkare olarak yüksekliği.
QTEXTMODE : 1 ise hızlı yazı modu açık (yazıları çevreleyen dikdörtgen görünür, yazılar görünmez),
0 ise hızlı yazı modu kapalı (çerçeve görünmez yazılar görünür).
REGENMODE : 1 ise açık (regen işlemi otomatik olarak yapılır)
2 ise kapalı (yapılıp yapılmayacağı kullanıcıya sorulur).
SCREENSIZE (Read Only) : Birim kare olarak grafik ekranın büyüklüğü (X,Y)
SKETCHINC : Elle çizimde birim doğru uzunluğu.
SKPOLY : 1 ise, Sketch komutu ile çizilen çizgi Pline olur.
0 ise, normal çizgi çizilir.
SNAPANG : Snap ve Gridlerin ortak döndürülme açısı
SNAPBASE : Snap ve Gridlerin ortak orijin noktası
SNAPISOPAIR : Mevcut izometrik düzlem ;
0 = Left,
1= Top,
2 = Right
SNAPMODE : 1 ise Snap modu açık 0 ise kapalıdır.
SNAPSTYL : Şekil yerleştirime stili
0= Aynı,
1= Eşölçülü
SNAPUNIT : X, Y olarak Snap aralıklarının ölçüsü
SPLFRAME : 0 ise sadece eğri görünür Pline görünmez.
1 ise eğri ile birlikte Pline'da görünür.
SPLINESEGS : Oluşturulacak eğrideki obje sayısı
SPLINETYPE : 5 ise Quadratic Spline
6 ise Cubic Spline çizilir.
SURFTAB1 : Oluşturulacak yüzeyin kaç sütun örüleceği.
SURFTAB2 : Oluşturulacak yüzeyin kaç satır örüleceği.
TDCREATE *(Read Only)* : Çizim dosyasının ilk yaratıldığı zaman (Özel format)
TDINDWG *(Read Only)* : Toplam düzeltme zamanı (Özel format).
TDUPDATE *(Read* Only) : Çizim dosyasının en son girilerek çalışılıp save edildiği zaman (Özel format).
TDUSRTIMER *(Read Only)* : Çizim dosyası için harcanan toplam süre (Özel format)
TEXTSIZE : Varsayım yazı yüksekliği.
TEXTSTYLE *(Read Only)* : Varsayım yazı stili.
THICKNESS : Çizilecek objelerin 3. boyut yüksekliği
TRACEWID : Varsayım çizgi genişliği.
UCSNAME (Read *Only)* : Aktif UCS adı.
VIEWCTR (Read *Only)* : Mevcut görünüşün merkezi.
VIEWSIZE *(Read Only)* : Mevcut görünüşün yüksekliği.

Referanslar:

1. AutoCAD Customization Manual, Autodesk,1992.
2. AutoCAD Development System Programmer's Manual, Autodesk,1992.
3. AutoCAD R12 ; Prof.Dr. Mustafa AKKURT, Birsen Yayınevi, İstanbul, 1993.
4. AutoLISP ; E. ÇIKIŞ , Türkmen Kitabevi, İstanbul, 1994.