

## **APRESENTAÇÃO**

### **1. Apresentação**

O AutoCAD é um programa CAD de propósito geral que pode ser utilizado na criação de uma variedade muito grande de projetos.

Uma das principais características que fazem do AutoCAD um Software de grande penetração e aceitação pelo mercado é sua arquitetura aberta; desta forma, o AutoCAD pode ser adequado às necessidades específicas de seus usuários, através de recursos tais como criação de macro-comandos, configurações especiais de menus e, principalmente através do desenvolvimento de rotinas em AutoLISP.

Não é por acaso que, atualmente, existam milhares de aplicativos desenvolvidos por empresas de software e por usuários ao redor do mundo que, aproveitando os recursos disponíveis no núcleo gráfico do programa, tornam o AutoCAD uma ferramenta de projeto para áreas tão diversas como manufaturas, mapeamento, projetos de plantas industriais, entre outras.

Este curso tem como objetivo apresentar conceitos de programação avançada em AutoLISP.

Através de explicações e exemplos, você aprenderá como tornar o AutoCAD uma ferramenta ainda mais produtiva para o desenvolvimento de projetos, adequando o AutoCAD ao seu ambiente de trabalho.

Esta apostila está organizada da seguinte maneira:

Cap. 2 : apresenta os conceitos básicos da AutoLISP.

Cap. 3 : explica como é feito o gerenciamento de memória do AutoLISP e apresenta as funções do AutoLISP a ele relacionadas.

Cap. 4 : apresenta as funções de entrada e saída do AutoLISP (leitura e gravação de arquivos).

---

Cap. 5: apresenta as funções avançadas de manipulação de listas, tais como: APPLY, FOREACH e MAPCAR.

Cap. 6 : apresenta as funções de acesso às entidades gráficas do AutoCAD.

Cap.7 : apresenta as funções de seleção de entidades gráficas do AutoCAD.

Cap.8 : apresenta as funções de acesso às tabelas internas do AutoCAD.

Cap.9 : faz um resumo dos tópicos apresentados no curso.

**Apêndice I** : apresenta a lista de todas as funções disponíveis no AutoLISP, ordenadas por funcionalidade.

**Apêndice II** : contém as tabelas de códigos DXF das entidades gráficas do AutoCAD.

O AutoLISP é uma linguagem de programação para o desenvolvimento de aplicativos exclusivamente no ambiente do software AutoCAD.

Você pode criar programas em AutoLISP digitando-os na própria área de comando do AutoCAD, ou criando um arquivo ASCII contendo as rotinas.

Arquivos de programas AutoLISP devem possuir extensão LSP e podem ser carregados no AutoCAD via função LOAD do próprio AutoLISP, que pode ser acionada na área de comando da seguinte forma:

Command: (**load "ASHADE"**)

A linha de comando acima carrega as rotinas em AutoLISP contidas no arquivo ASHADE.LSP.

---

## 2.1. Elementos básicos do AutoLISP

O AutoLISP é uma linguagem baseada em LISTAS. A seguir, serão apresentados os elementos básicos que compõem a sintaxe desta linguagem:

**ÁTOMO** : unidade básica do AutoLISP. Pode ser literal ou numérica.

**Exemplos:**

- átomo literal: ABC, PALAVRA TAMANHO GIGANTE, "STRING"
- átomo numérico: 1,0,0.1,-0.32,-93e+3

Num programa AutoLISP, variáveis e valores constantes são átomos.

**LISTA** :agrupamento de átomos ou listas delimitados por "(e)" e identificado como um único elemento.

**Exemplos:**

- () ou NIL ;lista vazia
- (AB "XX" 1'(0.3 Y) ) ; uma lista pode conter átomos ou listas

**EXPRESSÃO** : pode ser um átomo ou uma lista.

Exemplo:

- ( , 89, (Função PARAM1 0)

; : em um arquivo de programa em AutoLISP, o caracter ponto - e- vírgula indica início de comentário.

**Exemplo:**

(SETQ VMIN (MIN 0 1-1)) ; **VMIN recebe o mínimo entre 0, 1 e -1**

\ ou **PAUSE** : indica pausa, espera entrada de dados do usuário.

**T** : indica resultado verdadeiro (TRUE) em funções "booleanas".

**NIL** : indica resultado falso (NULL) em funções "booleanas" ou lista vazia.

## 2.2. Avaliação de expressões

---

Um programa em AutoLISP é uma seqüência de expressões (átomos ou listas) e a execução do programa nada mais é AVALIAÇÃO de cada uma dessas EXPRESSÕES.

O que é **AVALIAR UMA EXPRESSÃO** ?

- se a expressão for um valor constante, avaliar a expressão significa retornar o seu próprio valor.
- se a expressão for uma variável, avaliar a expressão significa retornar o conteúdo da variável.
- se a expressão for uma chamada de função, avaliar a expressão significa executar a função e retornar o valor retornado pela última expressão avaliada dentro da função.

**OBS:** Para diferenciar uma “lista de expressão” de uma “lista de dados” o AutoLISP utiliza o símbolo ou a função QUOTE.

**Exemplo:**

• (+1 2 3) : esta lista é reconhecida como uma função de soma, e a avaliação da mesma retorna o número 6.

• (+ 1 2 3) : esta lista é reconhecida apenas como uma lista que contém 4 átomos. Sua avaliação retorna a própria lista '(+1 2 3).

### 2.3. Funções elementares: CAR e CDR

O AutoCAD possui duas das funções elementares para manipulação de listas:

**CAR** : retorna o primeiro elemento de uma lista.

**Exemplos:**

- (CAR '(( A B ))) retorna **A**
  - (CAR '(( 1 2 3 ) 2 3 )) retorna **(1 2 3)**
-

**CDR** : retorna a lista sem o primeiro elemento.

Exemplos:

- (CDR '(AB)) retorna '(B)
- (CDR '((1 2 3) 2 3)) retorna '(2 3)
- (CDR '(UNICO)) retorna '() ou **NIL**, lista vazia

Essas funções podem ser usadas, combinadas, como nos exemplos abaixo:

(CAAR '((A B) 1 2)) = (CAR (CAR '((A B) 1 2))) = (CAR '(A B)) = A

(CDR '((A B) 1 2)) = (CAR (CDR '((A B) 1 2))) = (CAR '(1 2)) = 1

(CDDR '((A B) 1 2)) = (CDR (CDR '((A B) 1 2))) = (CDR '(1 2)) = '(2)

(CDAR '((A B) 1 2)) = (CDR (CAR '((A B) 1 2))) = (CDR '(A B)) = '(B)

**OBS:** No AutoLISP, pontos são representados por listas com dois o três elementos:

20.3 -32.9)) '(10.5 20.3) ;ponto bi- dimensional

'(10.5 20.3 -32.9) ;ponto tri- dimensional

As funções CAR e CDR são muito utilizadas para decompor coordenadas:

(CAR '(10.5 20.3 -32.9)) = **10.5, coordenada X**

(CADR '(10.5 20.3 -32.9))= **20.3, coordenada Y**

(CADDR '(10.5 = -32.9 , coordenada Z

## 2.4. Sintaxe geral de definição de função

A forma geral para definição de uma função em AutoLISP é uma lista onde:

- o primeiro elemento é o átomo **DEFUN** que indica início de definição de função.
- o segundo elemento é o nome da função (deve ser um átomo literal).

- o terceiro elemento é uma lista dividida em duas partes pelo caracter “/”: os elementos que ficam antes do caracter “/” são os parâmetros de entrada, os que ficam depois, são variáveis locais.

- os demais elementos são expressões (chamadas funções).

**(DEFUN FUNÇÃO (PARAM- 1 ... PARAM-N / VAR- 1 ... VAR-N) (...) ... (...))**

**Exemplo:** função PTMED, retorna uma lista contendo as coordenadas do ponto médio do segmento definido por dois pontos dados.

```
(DEFUN PTMED (p1 p2 / xm ym zm)
  (SETQ xm (/ (+ (CAR p1) (CAR p2)) 2.0)
        ym (/ (+ (CADR p1) (CADR p2)) 2.0)
        zm (/ (+ (CADDR p1) (CADDR p2)) 2.0) )
  (LIST xm ym zm)
```

**Observações:**

**(DEFUN PTMED (p1 p2 / xm ym zm )**

- inicia a definição da função PTMED com dois parâmetros de entrada, p1 e p2, e três variáveis locais : xm, ym e zm.

**SETQ xm (/ (+ CAR p1) (CAR p2)) 2.0)**

- p1 e p2 são os pontos que definem o segmento. Para calcular o ponto médio é necessário somar as coordenadas dos pontos e dividir por dois (em X, Y e Z separadamente).

- (CAR p1) e (CAR p2) são as coordenadas X dos pontos dados.

- SETQ é a função de atribuição : (SETQ VARIÁVEL VALOR), no exemplo, xm recebe a coordenada X do ponto médio.

---

- "+" é a função de soma: (+NUM1 NUM2 ...) retorna a soma dos números.

- "/" é a função de divisão: (/DIV1 DIV2), no exemplo, retorna o resultado da divisão da soma das coordenadas em X por 2.

**ym (/ (+ (CADR p1) (CAR p2)) 2.0)**

**zm (/ (+ (CADDR p1) (CADDR p2)) 2.0)**

- executa a mesma operação para as coordenadas Y e Z.

- a função SETQ pode conter várias atribuições.

**(LIST xm ym zm)**

- LIST é uma função de construção de listas: **(LIST EXPR1 EXPR2 ...)** retorna (EXPR1 EXPR2 ...)

**(LIST xm ym zm)**

## 2.5. Sintaxe geral para chamada de função

A forma geral para chamada de qualquer função em AutoLISP (funções próprias do AutoLISP ou criadas pelo usuário) é uma lista onde:

- o primeiro elemento é o nome da função (deve ser um átomo literal).

- os demais elementos são parâmetros. Parâmetros são expressões (átomos ou listas).

**(FUNÇÃO PARAM1 PARAM2...)**

**Exemplo:**

**PTMED '(1.0 2.3 4.5) '(10.9 -2.0 6.0))**

- a função PTMED retorna a lista '(5.95 0.15 5.25)

## 2.6. AutoLISP e Macros

---

A combinação de macros e rotinas em AutoLISP consiste num poderoso recurso de customização de menus do AutoCAD. Vejamos, através de um exemplo, como isso pode ser feito:

Inclusão da opção **(ZOOM IN)** no menu **DISPLAY** :

```
(ZOOM IN ) ^C ^C ^P (SETQ p1 (GETPOINT "Primeiro canto da janela
"));\+
(SETQ p2 (GETCORNER p1 'Outro canto: "));\ZOOM;w;IP1;IP2;^p
```

**Observações:**

**(ZOOM IN) ^C^C^P**

- “^C^C “ cancela o comando anterior.
- “^P “ faz com que as linhas de programação em AutoLISP não sejam exibidas na área de comando.

**(SETQ p1 (GETPOINT "Primeiro canto da janela: "))**

- esta linha de programa inicial a variável p1 com as coordenadas de um ponto fornecido pelo usuário através da função GETPOINT.
- GETPOINT requisita a entrada de um ponto:

**(GETPOINT (ponto âncora) (mensagem))**

- ponto âncora : ponto usado para gerar “rubber-band” (eco de linha). Este parâmetro pode ser omitido.

- mensagem: mensagem de “prompt”. Este parâmetro pode ser omitido.

•;\

---

---

- ; eqüivale a um < ENTER>/<RETURN>, significa fornecer a expressão para o AutoCAD avaliar (executar as funções GETPOINT e SETQ).

- \espera a entrada do ponto (PAUSE).

+

- indica que a macro continua na próxima linha.

#### **(SETQ p2 (GETCORNER (ponto âncora) (mensagem)**

-(ponto âncora): ponto usado para gerar o “rubber-band” (eco de retângulo). Este parâmetro **NÃO** pode ser omitido.

-(mensagem): mensagem de “prompt “. Este parâmetro pode ser omitido.

#### **ZOOM;W;lp1;lp2;**

- executa o comando **ZOOM** com opção W (window) fornecendo o conteúdo das variáveis p1 e p2 como pontos da janela.

- o sinal “|” serve para obter o conteúdo de uma variável.

### **3. Gerenciamento de memória do AutoLISP**

#### **3.1. AutoLISP**

trabalhar, o AutoLISP utiliza duas grandes áreas de memória:

- **HEAP** : área onde são armazenados **TODOS** os símbolos (variáveis, nomes de funções, cadeias de caracteres) utilizados pelas rotinas em AutoLISP. Cada um desses símbolos é chamado de nó (NODE). Quanto maior for o número de símbolos utilizados, maior é o espaço de área STACK necessário.

O espaço padrão de memória reservado para essas áreas é:

- 40000 bytes para área HEAP.

- 30000 bytes para área STACK.

Essa configuração pode ser modificada através de duas variáveis de ambiente (“ENVIRONMENT”) do sistema operacional DOS:

---

**C:> SET LISPHEAP = 40000**, configura números de bytes para área **HEAP**  
**C;> SET LISPSTACK = 3000**, configura número de bytes para área **STACK**

**ATENÇÃO:** o total de memória alojada somando-se as duas áreas não pode ultrapassar **45000 bytes**.

Se, durante a execução de uma rotina em AutoLISP, for detectada falta de memória na área STACK, a rotina é interrompida e a seguinte mensagem será exibida na área de comando:

Command : error : LISPSTACK overflow

Se o problema ocorrer na área HEAP, a rotina é interrompida e uma das duas mensagens a seguir aparecerá na área de comando:

Command: Insufficient node space

ou

Command: Insufficient string space

### 3.2. EXTENDED AutoLISP

“Extended AutoLISP “ é um recurso fornecido pelo AutoCAD R10 que permite configurar as áreas HEAP e STACK em memória estendida, acabando com a limitação de 45000 bytes para execução das rotinas em AutoLISP.

O “Extended AutoLISP” exige:

- um microcomputador 286 ou 386 com 640Kb de memória básica (RAM), no mínimo 512Kb de memória estendida (padrão IBM-AT) e disco winchester.
  - versão de sistema operacional PC-DOS/MS-DOS 2.0 ou superior.
-

- versão do AutoCAD R10 com ADE-3 (arquivos **ACADLX.OVL**, **EXTLISP.EXE** e **REMLISP.EXE**)

Para utilizar o “Extended AutoLISP”, é necessário executar o programa **EXTLISP.EXE** antes de entrar no AutoCAD. Este programa fica residente na memória e faz a pré- alocação da memória a ser utilizada pelo AutoLISP. Para retirar o EXTLISP.EXE da memória, após a execução do AutoCAD , execute o programa **REMLISP.EXE**.

A quantidade de memória a ser alojada pelo **EXTLISP.EXE** é configurada pela variável de ambiente LISPXMEN. Esta variável configura o tamanho e a posição inicial da área de memória reservada para o AutoLISP.

**C: SET LISPXMEN=BASE,Nkbytes**

onde BASE estabelece o início da área e Nkbytes o tamanho (em Kbytes).

Se o parâmetro BASE não for fornecido, o AutoCAD faz a alocação partindo do endereço de memória disponível mais alto para o mais baixo. Esta é a configuração mais “segura” tendo em vista possíveis incompatibilidades de alocação de memória envolvendo outros programas residentes.

**Exemplo:**

**C:> SET LISPXMEN=,1K**

ou

**C:> SET LISPXMEN=0,1K**

aloja 1 Kb a partir do endereço mais alto disponível.

Para maiores detalhes, consulte o Installation and Performance Guide - IBM PC and PS/2 ou o manual Autolisp Programmer’s Reference fornecidos junto com a documentação do software AutoCAD.

---

### **3.3 AutoLISP em AutoCAD versão 386**

No AutoCAD versão 386 não é necessário configurar nenhuma das variáveis de ambiente LISPHEAP, LISPSTACK, LISPXMEN, pois o próprio AutoCAD se encarrega de gerenciar a memória utilizada pelo AutoLISP, fazendo paginação automática entre a memória estendida, a memória básica e o disco winchester quando necessário.

#### **3.4.1. ALLOC**

##### **(ALLOC <número-de-nós)**

A função ALLOC configura o tamanho (números de nós) dos segmentos de memória. O valor padrão é 512 nós.

##### **Exemplo:**

##### **(ALLOC 500)**

Considerando a versão PC-DOS/MS-DOS do AutoLISP, na qual um nó ocupa 10 bytes, (ALLOC 500) configura segmentos de memória de 5000 (500 vezes 10) bytes.

#### **3.4.2 EXPAND**

##### **(EXPAND <número-de-segmentos>)**

---

A função EXPAND configura o número de segmentos de memória a serem utilizados pelos programas AutoLISP.

**Exemplo:**

**(ALLOC 500)**

**(EXPAND 5)**

Considerando a versão PC-DOS/MS-DOS do AutoLISP, na qual um nó ocupa 10 bytes, o trecho de programa acima configura 5 segmentos de memória de 5000 bytes a serem utilizados pelos programas AutoLISP, que resulta num total de 25000 bytes necessários na área HEAP.

### **3.4.3 GC**

**(GC)**

A função GC força a execução de uma operação de “garbage collection” (liberação dos nós livres para re-utilização pelos programas AutoLISP).

**Atenção!** O próprio gerenciador do AutoLISP se encarrega de fazer automaticamente um “garbage collection” toda vez que julgar necessário. Chamadas desnecessárias da função GC podem prejudicar a performance dos programas, pois trata de uma função cuja exceção é demorada.

### **3.4.4. MEM**

**(MEM)**

---

Exibe o estado atual do AutoLISP com relação a utilização de memória e número de nós disponíveis.

Command: **(MEM)**

Nodes: 1028 , **número total de nós**

Free nodes: 454 , **total de nós livres**

Segments: 2 , **número de segmentos de memória**

Allocat: 514 , **tamanho (em bytes) de cada segmento**

Collections: 3 , **número de “garbage collections” executados**

### **3.4.6. VMON**

**(VMON)**

A função VMON habilita a utilização de memória virtual (paginação) pelos programas AutoLISP, possibilitando a execução de programas que exigem muito espaço em memória.

## **4. FUNÇÕES DE ENTRADA E SAÍDA**

O AutoLISP possui funções que permitem ler e gravar arquivos em formato ASCII.

Na descrição de comandos apresentadas neste capítulo, os parâmetros entre “( )” são opcionais.

### **4.1.1. CLOSE**

**(CLOSE <“FILE-DESCRIPTOR”> )**

---

Fecha o arquivo associado ao valor retornado pela função OPEN que abriu o arquivo; este valor recebe o nome de "FILE-DESCRIPTOR".

A função sempre retorna **NIL**.

**Exemplo:**

```
(setq fd (open "ARQUIVO.DAT"))  
...  
(close fd)
```

#### 4.1.2. FINDFILE

**(FINDFILE <nome-do-arquivo>)**

Procura um arquivo no disco, considerando o seguinte critério para busca: o diretório corrente, o diretório configurado pela variável de ambiente "ACAD" e os diretórios contidos no "path".

Se o arquivo não for encontrado a função retorna **NIL**, caso contrário, retorna o nome completo do arquivo (incluindo o "pa th").

- nome do arquivo retornado pela função FINDFILE pode ser utilizado diretamente pela função OPEN que abre arquivos (o caracter "\" está substituído por "/").

**Exemplo:** Suponha que o arquivo **TXT.SHX** esteja no diretório **\ACAD**

```
(findfile "TXT.SHX")
```

retorna **"/ACAD/TXT.SHX**

---

### 4.1.3. OPEN

**(OPEN <nome-do-arquivo> <modo> )**

Abre um arquivo. Retorna um "file-descriptor" para ser utilizado como "senha" de acesso a esse arquivo nas funções de entrada/saída.

No parâmetro <modo> configura o tipo de acesso ao arquivo:

- **"r"** : **"read-only"** , abre o arquivo somente para leitura. Se o arquivo não existir, a função retorna **NIL**.

- **"w"** : **"write-only"** , abre arquivo somente para gravação. Se o arquivo já existir, os dados anteriores serão apagados.

- **"a"** : **"append"** , abre o arquivo para anexação de novos dados. Se o arquivo não existir, cria um novo arquivo. Caso contrário, o arquivo é aberto e os dados anteriores são mantidos.

Exemplos: O comando abaixo abre o arquivo **"DADOS.TXT"** para leitura ; se o arquivo existir retorna um "file-descriptor" na forma **<File #nnn>**; caso contrário, retorna **NIL**.

```
(setq fd (open ("DADOS.TXT" "r")))
```

### 4.1.4. PRIN 1

**(PRIN1 < expressão> (<"file-descriptor">))**

Se o parâmetro <"file-descriptor"> (parâmetro opcional) for fornecido, a função grava o resultado da avaliação do parâmetro <expressão> no arquivo associado ao <"file-descriptor">. Caso contrário, exibe esse resultado na área de diálogo. Retorna o valor da expressão.

---

**Exemplo:**

```
(setq VAR "\nBOM DIA") ; a função PRIN1
(PRIN1 VAR FD)          ;grava "\nBOM DIA" no arquivo FD.
```

**4.1.5. PRINC**

**(PRINC <expressão> (<"file-descriptor">))**

Esta função é idêntica à função PRIN1. A única diferença é que a função PRINC grava (ou exibe) a expressão sem expandir os caracteres de controle.

**Caracteres de controle:**

- "\n" : pula linha e vai para a próxima.
- "\t" : tabulação
- "\nnn" : código octal, "\014" é o código octal para o avanço de folha em impressora.

**Exemplo:**

```
(setq VAR "\nBOM DIA ") ; a função PRINC pula uma linha (dentro do
arquivo) e
(princ VAR FD)          ; grava o 'BOM DIA' no arquivo FD.
```

**4.1.6. PRINT**

**(PRINT <expressão > (<"file-descriptor">))**

---

Esta função é idêntica à função PRIN1. A única diferença é que a função PRINT pula uma linha antes de gravar (ou exibir) a expressão.

**Exemplo:**

```
(setq VAR "BOM DIA" ; a função PRINT pula uma linha (dentro do
arquivo) e
(print VAR FD) ; grava o "BOM DIA" no arquivo FD
```

#### 4.1.7. READ-CHAR

**(READ-CHAR (<"file-descriptor">))**

Se o parâmetro <"file-descriptor"> (parâmetro opcional) for fornecido, a função lê um caracter do arquivo associado ao <"file-descriptor">. Caso contrário, lê o caracter do teclado. Retorna o código ASCII do caractere.

**Exemplo:**

```
(setq COD (READ-CHAR FD))
```

**4.1.8. READ-LINE (<"file-descriptor">)**

Se o parâmetro <"file-descriptor"> (parâmetro opcional) for fornecido, a função lê uma cadeia de caracteres do arquivo associado ao <"file-descriptor">. Caso contrário, lê a cadeia de teclado. Retorna a cadeia lida.

```
(setq STR (read-line) FD))
(WRITE-CHAR <código-ASCII> (<"file-descriptor">))
```

Se o parâmetro <"file-descriptor"> (parâmetro opcional) for fornecido, a função grava o caracter correspondente ao <código-ASCII> dado no arquivo

---

associado ao <"file-descriptor">). Caso contrário, exibe o caracter na área de diálogo. Retorna o caracter.

**Exemplo:**

**(write-char 65 FD)** ; grava o carater "A" no arquivo.

#### 4.1.10. WRITE-LINE

**(WRITE-LINE <string> (<"file-descriptor">))**

Se o parâmetro <"file-descriptor"> (parâmetro opcional) for fornecido, a função grava a cadeia na área de diálogo. Retorna a cadeia.

**Exemplo:**

**(write-line "REG.01" FD)** ; grava "Reg.01" no arquivo.

#### 4.2. Exemplo

A rotina AutoLISP a seguir, importa um texto contido num arquivo ASCII para dentro do AutoCAD transformando cada linha deste arquivo num elemento texto.

```

;_____ ;
;LETEXTO ;
(defun LETEXTO (/ arq ; nome do arquivo
                path ; "path" do arquivo
                fd ; "file-descriptor"
                reg ; registro lido (texto)
                pt ; ponto de inserção do texto

; Pede o nome do arquivo de texto
(setq arq (getstring "\nNome do arquivo de texto:"))
;Retorna o "path" completo do arquivo
path (findfile arq)

```

---

```
)  
;Verifica se o arquivo existe  
(if (/=path NIL)  
; Ponto de inserção do texto  
if (setq pt (getpoint "\ninício do texto: "))  
;Abre o arquivo para leitura  
(if (setq fd (open path "r"))  
; Lê o primeiro registro  
(if (setq reg (read-line fd))  
(progn  
;Cria o texto  
  (command "TEXT" pt pause "" REG  
Lê o resto do arquivo  
  (while (setq reg (read-line fd))  
; Cria o texto  
    (command "TEXT" "" reg)  
)  
;Fecha o arquivo  
  (close fd)  
)  
)  
;Erro na abertura do arquivo  
  (princ "\nErro na abertura do arquivo.")  
)  
); Arquivo não encontrado  
(princ "\nArquivo não encontrado.")  
)  
(princ)  
)
```

## 5. FUNÇÕES DE MANIPULAÇÃO DE LISTAS

---

As funções de manipulação de listas apresentadas neste capítulo consistem o ponto forte do AutoLISP . São essas funções que fazem do AutoLISP uma linguagem poderosa e compacta.

### 5.1. Funções de pares associados

O AutoLISP possui um tipo de lista especial composta por apenas dois elementos que são decompostos pelas funções CAR e CDR da seguinte forma:

- a função CAR retorna o primeiro elemento dessa lista.
- a função CDR retorna o segundo elemento dessa lista.

Essa lista recebe o nome de **PAR ASSOCIADO** é representada por uma lista comum com apenas elementos separados por um ponto "." (por essa razão, pares associados também são chamados de "dotted pairs").

#### Exemplos:

```
(setq pa1 '(1 . 2)) ; pa1 contém um par associado  
(car pa1)          ; retorna 1  
(cdr pa1)          ; retorna 2 , ao invés de retornar '(2)
```

```
(setq pa2 ("abc" . ("a" "b" "c")))  
(car pa2)          ; retorna "abc"  
(cdr pa2)          ; retorna '( "a" "b" "c" )
```

Neste item serão apresentadas as funções de manipulação de pares associados . Essas funções são extremamente importantes para o tratamento de elementos gráficos do AutoCAD via AutoLISP.

#### 5.1.1. ASSOC

---

**(ASSOC <item> <lista-de-pares-associados>)**

Dado uma lista de pares associados e um item, retorna o par associado cujo primeiro elemento por igual ao item fornecido. Caso nenhum par seja encontrado, a função retorna **NIL**.

**Exemplo** (veja na próxima página) :

```
(setq LPA (list '(1 . "um") '(2 . "dois") '(3 . "três"))
```

```
(assoc 1 LPA) ; retorna '(1 . "um")
```

```
(assoc "um" LPA) ; retorna NIL
```

**5.1.2. CONS****(CONS <item1> <item2>**

Constrói pares associados. Retorna o par associado formado pelos parâmetros <item1> e <item2>

**Exemplos:**

```
(cons 1 "LINE") ; retorna (1 . "LINE")
```

```
(cons "ab" 2) ; retorna ("ab" . 2)
```

```
(cons 10 '(1.0 2.0 3.0)) ; retorna (10 . '(1.0 2.0 3.0))
```

**5.2. Funções especiais de manipulação de listas**

Algumas funções de manipulação de lista possuem características muito especiais que podem confundir o programador, provocando erros e gerando horas e horas de depuração de rotinas.

Preste atenção às explicações e dicas apresentadas neste item. Isto poderá evitar muitos erros de programação.

**5.2.1. SUBST****(SUBST <novo-item> <velho-item> <lista>**

Substitui toda ocorrência de <velho-item> na lista <lista> por <novo-item>. Retorna a nova lista.

**Exemplo:**

```
(setq L '(1 "AA" (2 3) "AA" (2 3)))  
(subst "BB" (2 3) L) ; retorna '(1 "AA" "BB" "AA" "BB")
```

A função SUBST, juntamente com a função ASSOC apresentada no item anterior, é muito utilizada no acesso a elementos do AutoCAD via AutoLISP.

### 5.2.2. APPEND X LIST

APPEND e LIST são funções de composição de listas, à primeira vista muito parecidas; porém ambas geram resultados totalmente diferentes.

**A função APPEND agrupa várias LISTAS numa única lista.** Por exemplo:

```
(append '(1) '(2) '(3)) ; retorna '(1 2 3)  
(append NIL '(1)) ; retorna '(1)  
(append 1 2 3) ; resulta ERRO (parâmetros inválidos)
```

**A função LIST agrupa vários ELEMENTOS numa nova lista.**

**Por exemplo:**

```
(list '(1) '(2) '(3)) ; retorna '((1) (2) (3))  
(list NIL '(1)) ; retorna '(NIL (1))  
(list 1 2 3) ; retorna '(1 2 3)
```

### 5.2.3. APPLY

---

**(APPLY <função> <lista>)**

Executa a função especificada por <função> utilizando os elementos do parâmetro <lista> como parâmetros.

**Exemplo:**

**; LNUM é uma lista composta por números**

```
(setq LNUM '( 5 10 20 ))
```

**(apply '+ LNUM))**; retorna a **35**, a soma de todos os números da lista.

**5.2.4 FOREACH**

```
(FOREACH <nome> <lista> <expr1> <expr2> ...)
```

Para cada elemento do parâmetro <lista>, a função FOREACH avalia as expressões <expr1>, <expr2>, ... substituindo o identificador <nome> pelo elemento da lista em questão. A função retorna o valor da última expressão avaliada com o último elemento da lista.

**Exemplo:**

**; testa a existência dos arquivos**

```
(foreach ARQ ("ACAD.PAT" "ACAD.LIN" "ACAD.OVL"))
```

```
(princ ARQ)
```

```
(if (null (findfile ARQ)
```

```
(princ ": Arquivo NÃO encontrado)
```

```
)
```

```
)
```

**5.2. LAMBDA**

```
'(LAMBDA <lista-de-parâmetros> <expr1> <expr2> ...)
```

---

Esta função é utilizada para definir funções “anônimas” para serem utilizadas juntamente com as funções APPLY e MAPCAR.

**Exemplo:**

```
; Converte um ângulo de grau para radiano.
;Pi é um valor constante (aproximadamente 3.1415926)
; fornecido pelo AutoLISP
‘(lambda (AG) (/ * AG Pi) 180.0))
```

```
; Incrementa todos os números de uma lista,
; (exemplo do manual de AutoLISP)
(setq L '(10 20 30 ))
(mapcar ‘1 + L) ;retorna ‘(11 21 31 )
```

**;LAMBDA e MAPCAR: retorna o ponto médio de um segmento**  
**;definido por dado dois pontos dados**

```
(setq P1 '( 5 10 20 )
      P2 '( 15 10 30 )
)
(mapcar ‘(lambda ( K1 K2 ) ; .K1 e K2 são coordenadas
          (( + K1 K2) 2.0) ; soma e divide por dois
        ) ; . P1 e P2 são duas listas
      P1P2 ; . correspondentes aos pontos
        ) ; . retorna ‘( 10 10 25 )
      )
```

trecho de programa acima equivale a :

```
(list (/ (+ 5 15) 2.0) ; primeira execução/primeiros elementos)
      (/ (+ 10 10) 2.0) ; segunda execução/segundos elementos)
```

---

(/ ( + 20 30) 2.0) ;terceira execução/terceiros elementos)

; **LAMBDA** e **APPLY** : a função verifica o arquivo  
; cujo nome é “ACAD” e cuja extensão é “CFG” existe

```
(setq NARQ '(lambda (NOME EXTENSÃO / ARQ)
  (apply (lambda ( NOME EXTENSÃI / ARQ)
    (setq ARQ (streat NOME “.” EXTENSÃO))
    (if (findfile ARQ)
      (princ (streat “\nArquivo” “ARQ “ existe.”))
      (princ (streat “\nArquivo” “ARQ” NÃO existe.”))
    )
  )
  NARQ
)
```

Obs1: na função **APPLY**, os elementos da lista **NARQ** são os parâmetros de entrada de função definida por **LAMBDA**.

Obs2: variáveis locais podem ser definidas dentro da função **LAMBDA**. No exemplo, **ARQ** uma variável local.

### 5.2.6. MAPCAR

(**MAPCAR** <função> <lista1> <lista2> ...)

**MAPCAR** executa a função dada pelo parâmetro <função> utilizando os elementos da lista <lista1> <lista2> ... como entrada da seguinte forma:

- o número de listas deve ser igual ao número de parâmetros de entradas da função .
  - essas listas devem ter o mesmo número de elementos.
-

- <função> é executada N-vezes, onde N é o número de elementos de cada lista.
- a primeira execução de<função> utiliza como parâmetro de entrada todos os primeiros elementos das listas, a segunda, utiliza os segundos elementos e assim por diante.
- MAPCAR retorna uma lista de N elementos (tamanho de cada lista) contendo resultado de cada uma das execuções realizadas.

**Exemplos na próxima página:**

## 6. FUNÇÕES DE ACESSO A ENTIDADES

Este capítulo explica como as entidades gráficas do AutoCAD podem ser consultadas e alteradas via AutoLISP.

### 6.1. Conceitos

AutoLISP possui funções que permitem acessar, consultar e editar entidades do banco de dados gráficos do AutoCAD. Isto é feito através de um tipo de dado especial denominado “ ENTITY NAME” ( nome de entidade).

“Entity Name” é um código através do qual se obtém a lista contendo os dados propriamente ditos de uma entidade. Esta lista recebe o nome de “ENTITY LIST” (lista de entidade).

‘ENTITY LIST’ é uma lista composta por pares associados, um para cada dado :

- o primeiro elemento do par associado é um número inteiro que indica o tipo de informação. Este número é considerado como CÓDIGO DXF (é o mesmo código utilizado nos arquivos de DXF)
- o segundo elemento é o próprio dado.

**Exemplo:** “entity list” de uma linha.

---

( (-1 . <Entity name:5000012>) (0 . "LINE") (8 . "0")  
(10 3.0 5.0 0.0) (11 1.0 2.0 1.0) )

- -1 é o código DXF de "entity name":
- 0 é o código DXF de tipo de elemento.

Veja tabela de códigos DXF das entidades do AutoCAD no apêndice II desta apostila.

## 6.2. Acesso a entidades

### 6.2.1. ENTLAST

(ENTLAST)

Retorna o "entity name" da última entidade criada no desenho.

#### Exemplo:

```
(command "LINE" "0,0" "1,1" "") ;cria uma linha  
(setq EN (ENTLAST)) ;"entity-name" da linha.
```

### 6.2.2. ENTNEXT

(ENTNEXT [< "ENTITY-NAME">])

Retorna o "entity-name" da entidade seguinte à <"entity-name"> no banco de dados do AutoCAD. Se <"entity-name"> não for fornecida, retorna a primeira entidade do banco de dados do AutoCAD. Caso não exista entidade seguinte, retorna **NIL**.

#### Exemplo:

---

**(setq EN1 (entnext))** ; primeira entidade no desenho  
**(setq EN2 (entnext EN1))** ; segunda entidade no desenho.

### 6.2.3. ENTSEL

**(ENTSEL [<mensagem>])**

Retorna uma lista contendo o “entity-name” e o ponto de identificação de um elemento selecionado pelo usuário. < mensagem> é o texto de “prompt” a ser exibido na área de diálogo.

#### Exemplo:

**(setq EL (entsel “\nSelecione elemento: “))**

Na linha de comando surgirá a seguinte mensagem:

Command: Selecione elemento:

Após a seleção , a ENTSEL retorna uma lista com dois elementos: ; “entity-name”

do elemento e o ponto de identificação.

**( <Entity name5000e03> (10.0 14.0 0.0))**

### 6.2.4. NENTSEL (R11)

**(NENTSEL [<mensagem>])**

Retorna um elemento identificado pelo usuário:

- se o elemento identificado for uma entidade simples (não for nem polyline nem bloco), a função NENTSEL tem o mesmo comportamento da função ENTSEL e retorna uma lista contendo o “entity-name” da entidade e o ponto de identificação.

- se o elemento for uma polyline, a função NENTSEL retorna uma lista contendo o “entity-name” do vértice mais próximo onde foi identificada a polyline e o ponto de identificação.

- se o elemento for um bloco, a função NENTSEL retorna uma lista com quatro elementos:

- “entity-name” do sub-elemento identificado do bloco.

- o ponto de identificação.

- Matriz de transformação: “Model to World Transformation ”

- lista d “entity-names” de blocos que contém a entidade identificada. Pode ser mais que um dependendo do nível de alinhamento (blocos compostos por outros blocos).

Quando um bloco é definido através de comando **BLOCK** , as entidades que o compõem são armazenadas no desenho numa coordenada especial denominada “MCS-Model Coordinate System “ (Sistema de Coordenadas do Modelo do Sistema).

Quando um bloco é inserido no desenho os seus sub-elementos não são duplicados. O AutoCAD apenas constrói uma matriz de transformação que , aplicada aos sub-elementos , transporta-os do MCS para o WCS e reposiciona-os no espaço na posição, ângulo e escalas desejados. Essa matriz de transformação é retornada pela função NENTSEL na forma de uma lista composta por quatro outras listas de três elementos cada uma:

**(( X1 Y1 Z1 ) ( X2 Y2 Z2 ) ( X3 Y3 Z3 ) ( X4 Y4 Z4 ) )**

---

Para transformar um ponto em MCS para WCS é necessário fazer o seguinte cálculo;

$$X_{wcs} = (X1 * X_{mcs}) + (X2 * Y_{cms}) + (X3 * Z_{mcs}) + X4$$

$$Y_{wcs} = (Y1 * X_{mcs}) + (Y2 * Y_{cms}) + (Y3 * Z_{mcs}) + Y4$$

$$Z_{wcs} = (Z1 * X_{mcs}) + (Z2 * Y_{cms}) + (Z3 * Z_{mcs}) + Z4$$

### 6.2.5. HANDENT

(HANDENT <handle > )

Retorna o "entity-name" da entidade associada ao <handle> fornecido.

**Exemplo:**

```
(setq EL (handent "1E3")) ; retorna a entidade associada ao  
; handle "1E3"
```

## 6.3. Manipulação de entidades

### 6.3.1. ENTDEL

(ENTDEL <" entity-name"> )

Elimina a entidade do desenho. A segunda chamada da função ENTDEL com o mesmo "entity-name" restaura a entidade do desenho.

**Exemplo:**

```
(setq EL (entlast))  
(entdel EL) ; Elimina o último elemento criado  
(entdel EL) ; Restaura o elemento
```

### 6.3.2. ENTGET

**(ENTGET <“entity-name”>**

Retorna a lista de dados da entidade.

**; identifica uma linha**

**(setq EL (entsel “\ nidentifique uma linha; ‘))**

```
(setq NEL (car EL)           ; nome da entidade
          PTI (cadr EL)      ; ponto de identidade
          LEL ( entget NEL ) ; lista de dados
          ;( (-1 . < Entity name:5000012>)
          ;(0 . “LINE” )
          ;(8 . “0”)
          ;(10 3.0 5.0 0.0)
          ;(11 1.0 2.0 1.0)
          ;)
```

### 6.3.3.

### 6.3.4. ENTMAKE

**(ENTMAKE <“entity-list “>**

Cria uma entidade gráfica no AutoCAD a partir de uma lista de pares associados semelhantes às listas retornadas pela função ENTGET. Se a entidade for criada com sucesso a função retorna a própria lista de dados, caso contrário, retorna **NIL**.

**Exemplo:**

```
(setq LD (list
          ( cons 0 “LINE” ) ; tipo de entidade: linha
          ( cons 10 (list 3.0 5.0 0.0)); vértices da linha
          ( cons 11 (list 1.0 2.0 1.0)
          )
          )
```

**(entmake LD)** ;cria a linha

**OBS:** os dados **LAYER, TIPO DE LINHA** e **ÍNDICE DE COR** não foram fornecidos, neste caso, a função assume os valores correntes configurados no momento da criação da entidade.

#### 6.3.4. ENTMOD

**(ENTMOD <"entity-list">)**

Atualiza o banco de dados do AutoCAD com a lista de dados fornecida. A lista de dados deve ser uma lista de pares associados semelhante à lista retornada pela função ENTGET. A função ENTMOD serve para efetivar alterações feitas na "entitu-list" através das funções SUBST, CONS e ASSOC.

**Exemplo:**

```
(setq EL (entget (entlast))) ; EL recebe a "entity-list" da
                             ; última entidade criada:
                             ; (-1 . <Entity name:5000012>)
                             ; (0 . "LINE")
                             ; (8 . "0")
                             ; (10 3.0 5.0 0.0)
                             ; (11 1.0 2 .0 1.0 )
                             ;)
(setq PT (list 10.0 20.0 0.0)) ; Novo vértice

(setq EL subst ( cons 10 PT) ; substitui o vértice
      (assoc 10 EL) ; ( (-1 . <Entity name:5000012>)
      EL           ; (0 . "LINE")
      )           ; ( 8 . "0")
      )           ; (10 10.0 20.0 0.0 )
      )           ; (11 1.0 2.0 1.0 )
```

---

```

; )
(entmod EL) ; atualiza o banco de dados do AutoCAD

```

### 6.3.5. ENTUP

```
(ENTUPD <"entity-name">)
```

A função ENTUP serve para efetivar alterações feitas nas "entity-list's" dos sub-elementos de entidades complexas (polylines e blocos) através das funções SUBST, CONS e ASSOC. O <"entity-name"> fornecido deve ser o nome de uma entidade complexa.

#### Exemplo:

```

; Suponha que o último elemento criado seja uma polyline
(setq PLL (entlast))
(setq VT (entget (entnext PLL) ; VT recebe o "entity-list"
; do primeiro vértice
; ( (-1 . <Entity name:5000012>)
; ( 0 . "VERTEX")
; ( 8 "0" )
; ( 10 3.0 5.0 0.0)
; ( 40.0 ) (41.0)
; ( 42.0 ) ( 50.0 )
; ( 70.0 )
; )

(setp PT (list 10.0 20.0 0.0 )); Novo vértice
(setp VT (subst (cons 10 PT) ; substitui o primeiro vertice
; ( (-1 < Entity name: 50000 12 > )
VT ; ( 0. " VERTEX")
; ( 8 . "0")

```

```

; (10 10.0 20.0 0.0 )
; (40.0) (41.0)
; (42.0) (50.0)
; (70.0)
;)

```

**(entmod TV)** ; atualiza o vértice

**(entupd PLL)** ; atualiza a polylin

## 7. FUNÇÕES DE SELEÇÃO DE ENTIDADES

Este capítulo explica como selecionar conjuntos de entidades gráficas do AutoCAD via AutoLISP.

### 7.1 Conceitos

O AutoLISP possui funções que permitem definir conjuntos de entidades do banco de dados gráfico do AutoCAD. Isto é feito através de um tipo de dado especial denominado "SELECTION-SET".

"Selection-Set" é um código que identifica um conjunto de entidades gráficas. Através desse código é possível obter a seqüência de "Entity-names" das entidades que fazem parte desse conjunto.

"Selection-Sets" são interpretados pelo AutoLISP como seleções simples feitas via AutoCAD, por exemplo, através do comando **SELECT**. No entanto, os recursos de seleção de elementos das funções do AutoLISP são mais poderosos que os fornecidos pelos comandos do AutoCAD.

#### Exemplo:

**(setp SS (ssget))**; exemplo de definição de "selection-set"

**(princ SS)** ;exibe <Selection set:1>

**(command "ERASE" SS")** ; apaga os elementos do "selection-set" SS

### 7.2 Seleção de conjuntos de entidades

#### 7.2.1. SSADO

**(SSADD [< "entity-name"> [<"selection-set">]])**

Adiciona a entidade <entity-name> no conjunto <"selection-set"> e retorna o nome do "selection-set". Se o parâmetro <"selection-set"> não for fornecido, a função cria um novo "selection-set" contendo apenas a entidade fornecida. Se nenhum dos parâmetros for fornecido, a função cria um novo "selection-set" contendo apenas a entidade fornecida. Se nenhum dos parâmetros for fornecido, a função cria um "selection-set" vazio.

No caso em que novos "selection-sets" são criados, a função retorna um novo nome para esses "selection-sets".

Veja exemplo na próxima página.

#### Exemplo:

**(setq LE entlast)** ; última entidade criada no AutoCAD

**(setq SS1 (ssget))** ; conjunto selecionado pelo usuário

**(setq SS1 (ssadd LE SS1))** ; acrescenta a entidade LE ao  
conjunto SS1

**(setq SS2 (ssadd LE))** ; cria SS2 apenas com LE

**(setq SS3 (ssadd))** ; cria SS3, conjunto vazio

#### 7.2.2. SSDEL

**(SSDEL <"ENTITY-NAME"> <"SELECTION-SET">)**

Elimina a entidade <"entity-name"> do conjunto <'selectio-set"> e retorna o nome do "selection-set". Se a entidade não estiver no conjunto, a função não faz nenhuma alteração.

#### Exemplo:

**(setq LE (entlast))** ; ultima entidade criada no AutoCAD

**(setq SS1 (ssdel LE SS1))** ; Se LE estiver no conjunto SS1, a função  
retira LE e retorna o nome do SS1  
; Caso contrário, o nome do SS1 é

---

; retornado sem nenhuma alteração

### 7.2.3. SSGET

**(ssget [<PONTO>])**

Retorna um “selectio-set”> . Se o <ponto> for fornecido, a função retorna um conjunto contendo apenas a entidade identificada por <ponto>. Caso contrário, a função pede para o usuário selecionar um conjunto, considerando válido qualquer modo de seleção disponível no AutoCad.

**Exemplo:**

**(setq PT '(1.0 1.0 0.0))**

**(setq SS1 (ssget PT))** ; Retorna um “selection-set” contendo

    ; a entidade identificada por PT.

    ; Caso nenhuma entidade tenha sido

    ; identificada, retorna NIL

**(setq SS2 (ssget))** ; pede que o usuário selecione um

    ; conjunto. A mensagem “Select objects”

    ; e exibida na área de diálogo.

### 7.2.4. SSGET

**(SSGET “X”<lista>)**

Retorna um “selection-set” contendo entidades selecionadas através de um filtro definido pelos elementos do parâmetro <lista>. Os elementos de<lista> devem ser pares associados cujo primeiro elemento é um código DXF que define o critério de seleção e o segundo, o valor desse código.

Veja tabela com alguns exemplos de códigos DXF

---

<b>DXF</b>	<b>Significado e Exemplo</b>
0	tipo de entidade : ( 0 . "LINE" ) )
2	nome do bloco: ( 2 . "BLK1" )
6	tipo de linha: ( 6 . "DASHED" )
7	estilo de texto: ( 7 . "STANDARD" )
8	"layer": ( 8 . "0" )
38	'elevation': ( 38 . 10.0000 )
39	"thickness": ( 39 . 5.0000 )
62	;indice de cor: ( 62 . 3 )
66	"attribute follow flag": ( 66 . 1 )
210	direção de extrusão: ( 210 ( 0.0 0.0 1.0 ) )

**Exemplo:**

;Seleciona todos os círculos do desenho

```
(setq SSCIRCLE (ssget "X" (list (cons 0 "CIRCLE"))))
```

Seleciona todos os blocos do desenho com nome "BLK1"

```
"(setq SS2 (ssget "X" (list (cons 2 "BLK1"))))
```

Seleciona todos os textos do layer "DESENHO"

```
(setq SS3 (ssget "X" (list (cons 0 "TEXT) (cons 8 "DESENHO"))))
```

**7.2.6. SSLENGTH**

```
(SSLENGTH <"selection-set">)
```

Retorna o número de entidades contidas no <"selection-set">).

**Exemplo:**

; NSS : número de blocos "BLK1" no desenho

```
(setq SS (ssget "X" (list (cons 2 "BLK1"))))
```

```
(setq NSS (sslength SS))
```

### 7.2.7. SSMEMB

**(SSMEMB <"entity-name"> <"selection-set">**

Retorna T se a entidade estiver no conjunto e NIL caso contrário.

**Exemplo:**

```
(setq SS (ssget))
(if (ssmemb (entlast) SS)
  (princ "\nPertence ao conjunto.")
  (princ "\nNao pertence ao conjunto."))
```

### 7.2.8. SSNAME

**(SSNAME <"selection-set"> <índice> )**

retorna a N-ésima entidade do <"selection-set">, onde N é dado pelo parâmetro <índice>. A contagem começa **a partir do 0**, isto é, a primeira entidade acesa pelo número 0.

**Exemplo:**

```
(setq n 0) ; índice
(setq SS (ssget)) ; Selection-set
(repeat (sslenght)) Acessa todas as entidades do conjunto
  (setq EL (sname SS n )
    n (1+ n)
  )
...
)
```

## 8. FUNÇÕES DE ACESSO A TABELAS

---

AutoLISP possui funções de consulta (apenas consulta) às tabelas do AutoCAD que armazenam informações internas do sistema:

- **“LAYER”** : tabela de descrição de layers
- **“LTYPE”** : tabela de descrição de tipo de linhas
- **“VIEW”** : tabela de descrição de vistas salvas
- **“STYLE”** : tabelas de descrição de estilo de texto
- **“BLOCK”** : tabela de descrição de blocos
- **“UCS”** : tabela de descrição de UCS's salvos
- **“VPORTS”** : tabela de descrição de divisão de janelas
- **“APPID”** : tabela de aplicações ((R11))
- **“DIMSTYLE”** : tabela de estilos de dimensionamento ((R11))

## 8.1. Acesso a Tabelas

### 8.1.1. TBLNEXT

**(TABLMEXT <nome-da-tabela> [<flag>])**

Retorna o próximo registro da tabela. Caso o parâmetro <flag> seja diferente de NIL , volta para o início da tabela e retorna o primeiro registro.

<nome-da-tabela> pode assumir qualquer um dos valores listados anteriormente.

**Veja exemplo na próxima página:**

**Exemplo:**

.desenho definido com dois layers: **“0”** e **“LINHAS\_DE\_CENTRO**

; Primeiro layer definido

**(setq La1 (tblnext “LAYER” T))**

---

```

; TBLNEXT retorna lista:
( ( 0 . "LAYER " ) ; tipo : "LAYER"
  ( 2 . "0" ) ; nome do layer: "0"
  ( 70 . 0 ) ; layer não está "congelado" ("frozen")
  ( 62 . 7 ) ; índice de cor do layer: 7 (branca)

  ( 6 . "CONTINUOUS" ) ; tipo de linha : contínua)

```

;Segundo layer definido

```
(setq LA2 (tblnext "LAYER"))
```

; TBLNEXT retorna a lista:

```

( ( 0 . "LAYER" ) ; tipo: "LAYER"
  ( 2 . "LINHAS_DE_CENTRO" ) : nome do Layer: "LINHAS_DE_CENTRO"
  ( 70 . 1 ) ; layer está "congelado" ("frozen")
  ( 62 . 5 ) ; índice do layer: 5 (azul)
  ( 6 . "CENTER" ) ; tipo de linha: linha de centro
)

```

; Na terceira chamada. A função retorna NIL

```
(setq LA3 (tblnext "LAYER"))
```

### 8.1.2. TBLSEARCH

```
(TBLSEARCH <nome-da-tabela> <símbolo> <símbolo> [<flag>])
```

---

Retorna o registro identificado por <símbolo> . Caso <flag> seja diferente de NIL, chamadas posteriores da função TBLNEXT retornarão subseqüentes ao registro encontrado.

**Exemplo:**

; Procura o estilo "ROMANC" na tabela "STYLE

**“(setq EST1 (tblsearch "STYLE" "ROMANC”))**

; Se o estilo estiver definido, a função retorna uma lista

; semelhante à lista abaixo . Caso contrário, retorna **NIL**

( ( 0 . "STYLE" ) ; tipo "STYLE"

( 2 . "ROMANC" ) ; nome do estilo: "ROMANC"

(70 . 0 ) ; texto definido na horizontal

(40 . 10 .0) ; altura fixa do texto

( 41. 0.75) ; fator de escala de largura ("width factor")

(50 . 0.0) ; ângulo de inclinação

(71 . 0) ; texto gerado da esquerda para direita

(3. "ROMANC") ; arquivo de fonte de texto

(4. "") ; arquivo de "Bigfont

)

**APÊNDICE 1: Lista por funcionalidade das Funções do AutoLISP**

---

**OBS1:** os parâmetros entre [ ] são opcionais

**OBS2:** ((R11)) indica que a função ou a opção só está disponível a partir do Release 11.

## SÍMBOLOS ESPECIAIS

,

- Abreviação da função QUOTE

\

- Indica pausa, espera entrada de dados do usuário

**C:**

- Prefixo utilizado nos nomes de funções para construção de novos comandos do AutoCad em AutoLISP.

**NIL**

- Lista vazia ou símbolo falso.

**PI**

- Valor de PI = 3.1415

**PAUSE**

- Equivale ao símbolo “\”.

**T**

- Símbolo verdadeiro

## FUNÇÕES DE MANIPULAÇÃO DE LISTAS

---

**(APPEND LST1 LST2 ...)**

- Retorna uma lista com posta pela junção de todas as listas dadas.

**(APPLY ITEM ALST)**

- Dada a lista de pares associados ALST , retorna o par associado cujo primeiro elemento é dado por ITEM.

**(ASSOC ITEM ALST)**

- Dada a lista de pares associados ALST, retorna o par associado cujo primeiro elemento é dado por ITEM.

**(CAR LST)**

- Retorna o primeiro elemento da lista LST.

**(CDR LST)**

- Retorna a lista LST sem o primeiro elemento.

**(C??..? R LST)**

- Retorna a lista ou o elemento definido pela combinação das funções CAR e CDR identificadas pelas letras A e D. Exemplo: CADDE , CDADR.

**(CONS EXPR1 EXPR2)**

---

- Retorna o par associado formado por `EXPR1` e `EXPR2`.

### **(LAST LST)**

- Retorna o último elemento da lista `LST`.

### **(MAPCAR FUNC LST1 LST2)**

• Retorna uma lista cujos elementos são o resultado da execução da função especificada por `FUNC` utilizando os elementos das listas `LST1`, `LST2`, etc., como argumentos. Todas as listas devem ter o mesmo número de elementos; além disso, o número de listas fornecidas deve ser igual ao número de parâmetros de entrada da função `FUNC`.

### **(LENGTH LST)**

- Retorna o número de elementos da lista `LST`.

### **(LIST EXPR1 EXPR2 ...)**

- Retorna uma lista formada por todas as expressões dadas.

### **(MEMBER ITEM LST)**

• Retorna o resto da lista `LST` a partir da posição do elemento especificado por `ITE` (inclui o próprio elemento). Caso `ITEM` não pertença a `LST`, a função retorna `NIL`.

### **(NTH N LST)**

• Retorna o `N`-ésimo elemento da lista `LST`. A contagem começa a partir do zero.

---

**(REVERSE LST)**

- Retorna a lista LST invertida.

**(SUBST ITEM1 ITEM2 LST)**

Substitui toda ocorrência de ITEM2 na lista LST por ITEM1 e retorna a nova lista.

**FUNÇÕES ARITMÉTICAS****( + NUM1 NUM2 ...)**

- Retorna o resultado da soma de todos os números.

**( - NUM1 NUM2 ....)**

• Retorna o resultado da diferença do primeiro número menos a soma dos outros .

**( \* NUM1 NUM2 ...)**

- Retorna o resultado do produto de todos os números.

**( / NUM1 NUM2 .. )**

- Retorna o resultado da divisão do primeiro número pelo produto dos outros.

**( 1 + NUM)**

---

- Retorna o valor do número incrementado de 1.

**( 1 - NUM)**

- Retorna o valor do número decrementado de 1.

**(ABS NUM)**

- Retorna o valor absoluto do número.

**(EXP NUM)**

- Retorna E elevado a NUM

**(EXPT NUM1 NUM2)**

- Retorna NUM1 elevado a NUM2.

**( GCD NUM1 NUM2)**

- Retorna o máximo divisor comum dos números.

**(LOG NUM)**

- Retorna o logaritmo neperiano do número.

**(MAX NUM1 NUM2 ...)**

- Retorna o valor máximo.
-

**( MIN NUM1 NUM2 ... )**

- Retorna o valor mínimo.

**(REM NUM1 NUM2 ... )**

- Retorna o resto da divisão do primeiro número pelo produto dos outros.

**( SQRT NUM)**

- Retorna a raiz quadrada do número .

**FUNÇÕES TRIGONOMÉTRICAS****(ATAN NUM1 NUM2]**

- Retorna o arco tangente (em radianos) e NUM1 ou, caso NUM2 seja fornecido , arco tangente de NUM1/NUM2.

**(COS NUM)**

- Retorna o co-seno dos números ( em radianos).

**(SIN NUM)**

- Retorna o seno do número (em radianos).

**FUNÇÕES GEOMÉTRICAS****(ANGLE PT1 PT2)**

- Retorna o ângulo (em radianos) do segmento definido pelos dois pontos.
-

**(DISTANCE PT1 PT2)**

- Retorna a distância entre os dois pontos.

**( INTERS PT1 PT2 PT3 PT4 [FLG])**

- Retorna o ponto de interseção entre o segmento definido pelos dois primeiros pontos e o segmento definido pelos dois últimos. Se FLAG for Nil, o ponto de interseção só é retornado se pertencer aos segmentos.

**(POLAR PT ANG DIST)**

- Retorna o ponto cuja distância polar de PT é definida pelo ângulo ANG e pela distância DIST.

**(> AT1 AT2 ...)**

- Retorna T se os elementos estão em ordem decrescente.

**(< AT1 AT2 ... )**

- Retorna T se os elementos estão em ordem crescente.

**(> = AT1 AT2 ... )**

- Retorna T se os elementos estão em ordem decrescente (faz a comparação com maior ou igual).

**(/ = AT1 AT2)**

---

- Retorna T se os elementos são diferentes.

**( = AT1 AT2 ... )**

- Retorna T se todos os elementos são iguais.

**(AND EXPR1 EXPR2 ..)**

- Retorna T se todas as expressões são verdadeiras.

**( ATOM ELEM)**

- Retorna T se o elemento for um átomo (não for nem uma lista nem NIL)

**(BOUNDUP AT)**

- Retorna T se o átomo contém um valor diferente de NIL.

**(EQ VAR1 VAR2 )**

- Retorna T se os conteúdos das variáveis são iguais.

**(EQUAL EXPR1 EXPR2 [PREC])**

- Retorna T se as expressões são iguais. PREC indica a precisão numérica a ser considerada na comparação.

**(LISTP ELEM)**

- Retorna T se o elemento for uma lista.
-

**(MINUSP NUM)**

- Retorna T se o número for negativo.

**(NOT EXPR)**

- Retorna o contrário da expressão. Se a expressão for NIL, retorna T, caso contrário, retorna NIL.

**(NULL ELEM)**

- Retorna T se o elemento for NIL.

**(NUMBERP ELEM)**

- Retorna T se o elemento for um átomo numérico.

**(OR EXPR1 EXPR2 ... )**

- Retorna T se uma das expressões for verdadeira.

**(ZEROP NUM)**

- Retorna T se o número for igual a zero.

**FUNÇÕES DE MANIPULAÇÃO DE BITS****( ~ NUM)**

- Retorna o resultado da operação lógica NOT com os bits do número.
-

**(BOOLE FUNC NUM1 NUM2 ... )**

- Retorna uma das 16 possíveis operações booleanas indicada pelo parâmetro FUNC aplicada aos números inteiros fornecidos.

**(LOGAND NUM1 NUM2 ... )**

- Retorna o resultado (um número inteiro) da operação lógica de bits AND entre os números inteiros fornecidos.

**(LOGIOR NUM1 NUM2 ... )**

- Retorna o resultado (um número inteiro) da operação lógica de bits OR entre os números inteiros fornecidos.

**(LSH NUM NUMBITS)**

- Retorna o resultado da operação de “BITWISE SHIFT” sobre o valor do número inteiro NUM, NUMBITS positivo indica o número de bits a serem rodados para a esquerda. NUMBITS negativo indica o número de bits a serem rodados para a direita.

**FUNÇÕES DE CONVERSÃO****(ANGTOS ANG [MODO] [PREC])**

---

- Converte o Ângulo em cadeia de caracteres segundo os parâmetros opcionais MODO e PREC. MODO indica a unidade angular e PREC a precisão numérica a serem utilizadas na conversão. Em caso de omissão desses parâmetros, a função considera os valores configurados na variáveis de sistema AUNITS e AUPREC.

**(ASCII STR)**

- Retorna o código ASCII do primeiro caracter se DTR.

**(ATOF STR)**

- Converte cadeia de caracter em número real.

**(ATOI STR)**

- Converte cadeia de caracter em número inteiro.

**(CHR NUM)**

- Retorna o caracter correspondente ao código ASCII indicado por NUM.

**(FIX NUM)**

- Transforma um número real em número inteiro. A conversão é feita truncando-se a porção fracionária do número real.

**(FLOAT NUM)**

- Transforma um número inteiro em número real.

**(ITOA NUM )**

---

- Converte um número inteiro em cadeia de caracter.

### **(RTOS NUM [MODO] [PREC])**

• Converte número real em cadeia de caracteres segundo os parâmetros opcionais, MODO e PREC. MODO indica a unidade linear e PREC., a precisão numérica a serem utilizadas na conversão. Em caso de omissão desses parâmetros, a função considera os valores configurados nas variáveis de sistema **LUNITS** e **LUPREC**.

### **(TRANS PT COORD1 COORD2 [FLAG])**

• Converte um ponto da coordenada COORD1 para a COORD2. Se FLAG for diferente de NIL, a função considera PT como deslocamento e não como ponto. COORD1 e COORD2 podem assumir os seguintes valores:

<b>VALOR</b>	<b>SISTEMA DE COORDENADAS</b>
0	World Coordinate System (WCS)
1	User Coordinate System (UCS)
2	Display Coordinate System (DCS)
3	Paper Space Coordinate System ((R11))
NENT (nome de entidade)	Entity Coordinate System (ECS)

## **FUNÇÕES DE MANIPULAÇÃO DE CADEIA DE CARACTERES**

### **(READ STR)**

---

- Retorna o primeiro átomo ou lista contida na cadeia de caracteres STR.

**(STRCASE STR [FLAG] )**

- Converte os caracteres de STR para maiúsculo. Se FLAG for T , converte para minúsculo.

**(STRCAT STR1 STR2 ... )**

- Retorna a cadeia resultante da concatenação de todas as cadeias dadas.

**(STRLEN STR)**

- Retorna o número de caracteres que compõem a cadeia.

**(SUBSTR STR INIC [INCAR])**

- Retorna uma sub-cadeia de STR , começando pela posição dada por INIC com comprimento NCAR. Se NCAR não for fornecido , a sub-cadeia vai até o final da cadeia original .

**(WCMATCH STR PADROES) ((R11))**

- Verifica se a cadeia de caracteres STR se encaixa num dos padrões contidos na cadeia de caracteres PADROES.

**FUNÇÕES DE ACESSO A TELA E DISPOSITIVOS DE ENTRADA****(GRAPHSCR)**

- Muda a tela para modo gráfico.

---

**(GRCLEAR)**

- Limpa temporariamente a tela gráfica.

**(GRDRAW PT1 PT2 COR [FLAG])**

. Desenha um traço de PT1 a PT2. O COR indica do traço. Se FLAG for diferente de zero, o traço será desenhado no modo "HIGHLIGHT".

**(GRREAD [FLAG])**

. Lê dados diretamente de um dispositivo de entrada. Retorna uma lista cujo primeiro elemento é o código de identificação do dispositivo de entrada e o segundo, o valor lido. Se FLAG for T, a função retorna uma coordenada de ponto sem que o botão de identificação de ponto tenha que ser pressionado.

<b>CÓDIGO</b>	<b>TIPO DE ENTRADA</b>	<b>VALOR LIDO</b>
2	teclado	código ASCII da tecla
3	seleção de ponto	coordenadas do ponto
4	campo de menu de tela	índice do campo
5	seleção de ponto sem botão de identificação	coordenada do ponto
6	menu BUTTONS	número do botão
7	menu TABLET1	índice do campo
8	menu TABLET2	índice do campo

---

---

9	menu TABLET3	índice do campo
10	menu TABLET4	índice do campo
11	menu AUX1	índice do campo
12	coordenada associada ao menu BUTTONS	coordenada do ponto
13	menu de tela selecionado via teclado	índice do campo

### **(GRTEXT [BOX TEXT [FLAG]])**

.Exibe o texto TEXT na linha de status ou no menu de tela. O parâmetro BOX para assumir os seguintes valores.

<b>BOX</b>	<b>ÁREA</b>
------------	-------------

0 a 20	campos do menu lateral (0 indica o campo no topo da tela)
--------	---

-1	linha de status (região do layer corrente)
----	--

-2	linha de status (região das coordenadas).
----	---

.Se FLAG for T, os textos colocados no menu de tela serão exibidos em modo "highlight". Caso nenhum parâmetro seja fornecido, a função restaura a tela original.

### **(PROMPT STR)**

- Exibe a cadeia de caracteres STR na área de diálogo.

### **(REDRAW NENT [MODO])**

- Redesenha a entidade NENT de acordo com o valor em MODO:
-

---

<b>MODO</b>	<b>EXIBIÇÃO</b>
1	redesenha a entidade na tela
2	apaga a entidade da tela
3	desenha a entidade em “highlight”
4	desfaz o “highligt”

**(TERPRI)**

Provoca um avanço de linha na área de diálogo.

**(TEXTPAGE) ((R11))**

Muda a tela para modo texto, limpando a tela automaticamente.

**(TEXTSCR)**

Muda a tela para modo texto.

**FUNÇÕES DE MANIPULAÇÃO DE ARQUIVOS****(CLOSE FD)**

Fecha arquivo em disco referenciado pelo “file descriptor” FD

**(FINDFILE ARQ)**

Procura o arquivo ARQ. Se encontrar, retorna o “path” completo. Caso contrário retorna **NIL**

---

**(OPEN ARQ MODO)**

Abre arquivo com nome fornecido em ARQ segundo o modo especificado MODO: "r" (READ ONLY), "w" (WRITE ONLY), "a"(APPEND). Se a operação for bem sucedida, retorna o "file descriptor" do arquivo.

**(PRIN1 Expr [FD])**

Se FD for fornecido, grava a expressão Expr no arquivo. Caso contrário, exibe expressão na área de diálogo

**(PRINC Expr [FD])**

Esta função é idêntica à função PRIN1. A única diferença é que a função PRINC grava/exibe a expressão sem expandir os caracteres de controle.

**(PRINT Expr [FD])**

Esta função é idêntica à função PRIN1. A única diferença é que a função PRINT pula uma linha antes de gravar/exibir a expressão.

**(READ-CHAR [FD])**

Se FD for fornecido, lê um caracter do arquivo. Caso contrário, lê o caracter do teclado. A função retorna o código ASCII do caracter lido.

**(READ-LINE [FD])**

Se FD for fornecido, lê uma cadeia de caracteres do arquivo. Caso contrário, lê do teclado.

---

**(WRITE-CHAR NUM [FD])**

Se FD for fornecido, grava o caracter correspondente a o código ASCII dado por NUM. Caso contrário, exhibe o caracter na área de diálogo.

**(WRITE-LINE STR [FD])**

Se FD for fornecido, grava a cadeia de caracteres STR no arquivo. Caso contrário, exhibe na área de diálogo.

**FUNÇÕES DE ESTRUTURA DE CONTROLE DE PROGRAMAÇÃO****(COND (TESTE1 EXPR1) (TESTE2 EXPR2) ...)**

Avalia a expressão EXPR1 pertencente a primeira lista cujo

**(COND (TESTE1 EXPR1) (TESTE2 EXPR2)...) )**

Avalia a expressão EXPR1 pertencente a primeira lista cujo teste TESTE1 resultar T.

**(FOREACH NOME LST EXPR)**

Avalia a expressão EXPR para cada elemento da lista LST. Os elementos da lista LST são identificados dentro da expressão EXPR através do nome fornecido em NOME. A função retorna o resultado da última avaliação executada.

**(IF TESTE EXPR1 [EXPR2])**

Se TESTE resultar verdadeiro, avalia a expressão EXPR1. Se EXPR2 for fornecida, essa expressão é avaliada no caso de TESTE retorna NIL.

---

**(PROGN Expr1 Expr2...)**

Agrupa várias expressões de modo que possam ser executadas como uma única expressão.

**(REPEAT Num Expr1 Expr2)**

Avalia cada uma das expressões dada Num vezes. Num deve ser um número (ou átomo numérico) inteiro.

**(TRACE Func1 Func2)**

Exibe os parâmetros de entrada e o valor retornado pelas funções dadas. A função TRACE é utilizada como ferramenta de depuração de programas.

**(UNTRACE Func1 Func2...)** Retira as funções dadas da lista de funções afetadas pela função TRACE.

**(WHILE Test Expr1 Expr2...)**

Avalia as expressões enquanto a expressão Test retornar T.

**FUNÇÕES DE ENTRADA DE DADOS****(GETANGLE [PT] [MSG])**

Espera a entrada de um valor de ângulo no sistema de unidade angular e retorna o valor em radianos. Se um ponto PT for fornecido, a função pedirá mais um ponto e retornará o ângulo (em relação ao eixo X) do segmento formado por PT e o ponto dado. A mensagem MSG, caso seja fornecida, aparecerá na área de comandos.

---

**(GETCORNER PT [MSG])**

Espera a entrada de um ponto com “rubber-band” (eco de retângulo ancora ponto PT. A mensagem MSG, caso seja fornecida, aparecerá na área de comando.

**(GETDIST [PT] [MSG] )**

Espera a entrada de um valor de distância no sistema de unidade linear corrente retorna esse mesmo valor. Se um ponto PT for fornecido, a função mais ponto e retornará a distância entre o PT e o ponto dado. A mensagem MSG, caso fornecida, aparecerá na área de comando.

**(GETINT [MSG])**

Espera a entrada de um número inteiro. A mensagem MSG, caso seja fornecida, aparecerá na área de comando.

**(GETKEYWORD [MSG])**

Espera a entrada de uma cadeia de caracteres que seja uma “palavra-chave”, “palavra-chave” são configuradas via função INITGET. A mensagem MSG, caso fornecida, aparecerá na área de comando.

**(GETORIENT [PT] [MSG])**

Esta função é similar à GENTANGLE. A única diferença é que quando o ângulo for fornecido por dois pontos, o cálculo será tomando-se como base a direção do ângulo 0 configurado pelo comando **UNITS**.

**(GETPOINT [PT] [MSG])**

Espera a entrada de um ponto com “rubber-band” (eco de linha) ancorado no ponto PT. A mensagem MSG, caso seja fornecida, aparecerá na área de comando.

### **(GETREAL [MSG])**

Espera a entrada de um número real. A mensagem MSG, caso seja fornecida, aparecerá na área de comando.

### **(GETSTRING [FLAG] [MSG])**

Espera a entrada de uma cadeia de até 132 caracteres. Se o parâmetro FLAG não for fornecido, a função não aceitará cadeias de caracteres que contenham espaços. A mensagem MSG, caso seja fornecida, aparecerá na área de comando.

### **(INITGET [BITS] [STR])**

Define restrições para as funções de entrada tipo GETxxx. Os bits do parâmetro BITS possuem os seguintes significados:

- 1 – não permite entrada nula
- 2 – não permite entrada de zero (ex: 0, 0.0000)
- 4 – não permite a entrada de valores negativos
- 8 – não verifica os limites de coordenadas
- 16 – retorna pontos em 3D ao invés de 2D
- 32 – utiliza linha pontilhada para fazer “rubber-band” (eco)

Esses bits podem ser combinados através de operações de adição.

O parâmetro STR contém “palavras-chaves” que serão reconhecidas pela função **GETKEYWORD**

---

## **FUNÇÕES DE ACESSO A ENTIDADES**

### **(ENTDEL NENT)**

Elimina a entidade do desenho. A segunda chamada da função ENTDEL com a mesma entidade faz com que ela seja restaurada ao desenho.

### **(ENTGET NET)**

Retorna a lista de dados correspondente a entidade dada.

### **(ENTLAST)**

Retorna a última entidade criada no desenho.

### **(ENTMAKE [LENT]) (( R11))**

Cria um elemento gráfico no AutoCAD a partir de uma lista de pares associados semelhante às listas retornadas pela função ENTGET. Se o elemento foi criado com sucesso a função retorna à própria LENT, senão retorna NIL.

### **(ENTMOD LENT)**

Modifica a lista de dados de uma entidade.

### **(ENTNEXT [NENT])**

Retorna a primeira entidade do desenho. Caso o parâmetro NENT seja fornecido retorna a entidade seguinte à NENT no banco de dados.

---

**(ENTSEL [MSG])**

Espera que o usuário selecione uma entidade. Retorna uma lista contendo a entidade selecionada e o ponto de identificação da mesma.

**(ENTUPD NENT)**

Modifica entidade complexa (por linhas e blocos).

**(HANDENT HANDLE)**

Retorna a entidade associada ao “handle” fornecido.

**(NENTSEL [MSG]) ((R11))**

Espera que o usuário selecione uma entidade. Se for uma entidade simples, retorna uma lista contendo a entidade selecionada e o ponto de identificação da mesma. Se for uma polyline retorna o “entity-name” do vértice identificado. Se for um bloco, retorna uma lista contendo o “entity-name” da sub-entidade identificada, o ponto de identificação, uma matriz de transformação e a lista dos “entity -names” do bloco que contém a sub-entidade em questão.

**FUNÇÕES DE SELEÇÃO DE ENTIDADES****(SSADD [NENT] [SS])**

Retorna o “selection set” resultante da adição da entidade NENT ao “selection set” SS. Caso os parâmetros NENT e SS não sejam fornecidos, a função retorna um “selection set” vazio.

---

**(SSDEL NENT SS)**

Retorna o “selection set” resultante da eliminação da entidade NENT do “selection set” SS

**(SSGET [PT])**

Retorna um “selection set” com elementos selecionados pelo usuário através dos modos de seleção convencionais do AutoCAD. Se o ponto PT for fornecido, retorna um “selection set” contendo o elemento identificado por PT.

**(SSGET MODO [PT1] [PT2])**

Retorna um “selection set” de acordo com o critério determinado pelo parâmetro MODO: “P” seleciona entidades pré-selecionadas, “L” seleciona a última entidade criada, “W” e “C” selecionam entidades por janela e exigem o fornecimento dos dois pontos que determinam as diagonais da janela de seleção .

**(SSGET “X’ LST)**

Retorna um “selection set” contendo entidades selecionadas através de um filtro definido pelos elementos da lista LST. Os elementos de LST devem ser pares associados ( construídas através da função CONS) cujo primeiro elemento é um código DXF que define o critério de seleção e o segundo elemento é o valor desse código DXF a ser utilizado como “chave” para a busca de entidades, conforme mostrado nos exemplos da tabela:

<b>DXF</b>	<b>SIGNIFICADO</b>	<b>EXEMPLO DE SELEÇÃO</b>
0	Tipo da entidade	(cons 0 “LINE”): linhas
2	Nome de bloco	(cons 2 “FIG1”): blocos”FIG1”

6	Tipo de linha	(cons 6 "DASHED"):linhas pontilhadas
7	Estilo de texto	(cons 7 "FONTE1" :texto com fonte "FONTE1"
8	Layer	(cons 8 "0"): entidades do layer "0"
38	Elevation	(cons 38 10.0): entidades com elevação 10.0
39	Thickness	(cons 39 5.0): entidades com thickness 5.0
62	Índice de cor	(cons 62 2): entidades com cor (vermelha)
66	Atribute follow	(cons 66 1):blocos com atribute follow flag1
210	Vetor de extrusão	(cons 210 (0 0 1): entidades com vetor de extrusão (0 0 1)

**(SSLENGTH SS)**

Retorna o número de entidades do "selection set" SS

**(SSMEMB NENT SS)**

Retorna T se a entidade NENT fizer parte do "selection-set" SS

**(SSNAME SS N)**

Retorna a N-ésima entidade do "selection-set" SS. A contagem começa a partir de 0.

---

## FUNÇÕES DE ACESSO A TABELAS

### (TBLNEXT NTAB [FLAG])

Retorna o próximo registro da tabela NTAB. Caso o parâmetro FLAG seja fornecido, retorna o primeiro registro da tabela NTAB. O parâmetro NTAB pode assumir os seguintes valores:

- **“LAYER”** : tabela de descrição de layers
- **“LTYPE”** : tabela de descrição de tipos de linha
- **“VIEW”** : tabela de descrição de vistas salvas
- **“STYLE”** : tabela de descrição de estilos de fontes de texto
- **“BLOCK”** : tabela de descrição de blocos
- **“UCS”** : tabela de descrição de UCS's
- **“VPORT”** : tabela de descrição de divisão de janelas (viewports)
- **“APPID”** : tabela de aplicações ((R 11))
- **“DIMSTYLE”**: tabela de estilo de dimensionamento ((R11))

### (TBLSEARCH NTAB SIMB [FLAG])

• Retorna o registro da tabela NTAB identificada por SIMB. Caso o parâmetro FLAG seja fornecido, chamadas posteriores da função TBLNEXT retornarão os registros subsequentes ao registro encontrado.

## FUNÇÕES DE GERENCIAMENTO DE MEMÓRIA

### (ALLOC NUM)

• Configura o tamanho de cada segmento de memória reservado para AutoLISP.

---

**(EXPAND N)**

- Aloca N segmentos de memória com tamanho configurado pela função ALLOC.

**(GC)**

- “Garbage Colector” . Libera (torna disponível) todos os nós não utilizados.

**(MEM)**

- Exibe o estado corrente de alocação de memória do AutoLisp.

**(VMON)**

- Ativa o modo de paginação de função em memória virtual.

**FUNÇÕES ESPECIAIS DO AUTOLISP****(ADS) ((R11))**

- Retorna uma lista contendo os nomes dos programas ADS carregados na memória.

**(COMMAND ARG1 ARG2...)**

- Executa comandos do AutoCAD.

**(DEFUN NFUN (PARAM1 PARAM2...) EXPR1 EXPR2...)**

- Define novas funções no AutoLISP.

**(\*ERROR\* MSG)**

- Função de tratamento de erro interno.

**(EVAL EXPR)**

- Retorna o resultado da avaliação da expressão EXPR.

**(GETENV VAR)**

---

- Retorna o valor da variável de ambiente (“environment”) VAR.

**(GETVAR VARSIS)**

- Retorna o valor da variável de sistema VARSIS.

**(LAMBDA (ARG1 ARG2...) EXPR1 EXPR2...)**

- Define funções temporárias sem nomeá-las.

**(LOAD NARQ [EXPR])**

• Carrega um arquivo contendo rotinas AutoLISP. Se o parâmetro EXPR (expressão) for fornecido, em caso de erro no carregamento do arquivo, ao invés de acusar erro, o AutoLISP avaliará a expressão EXPR e a função LOAD retornará o valor retornado por EXPR.

**(MENUCMD STR)**

- Carrega ou mostra um menu na tela gráfica do AutoCAD.

**(OSNAP PT STR)**

• Retorna o ponto notável (“SNAP POINT”) determinado pelo ponto PT e pelo modo de “snap” configurado em STR; por exemplo, “END, CEN” seleciona pontos finais e centros.

**(QUOTE EXPR)**

- Retorna a expressão EXPR sem avaliá-la.

**(SET SIMB EXPR )**

• Atribui o valor da expressão EXPR ao símbolo SIMB, onde SIMB é apenas um símbolo e não uma variável.

**(SET VAR1 EXPR1 [VAR2 EXPR2]...)**

• Atribui o valor da expressão EXPR1 à variável VAR1. A mesma chamada da função SETQ pode conter várias atribuições.

---

**(SETVAR VARSIS EXPR)**

- Atribui o valor da expressão EXPR à variável de sistema VARSIS.

**(TYPE ITEM)**

- Retorna o tipo do item fornecido: **REAL, FILE, STR, INT, SYM, LIST, SUBR, PICKSET, ENAME, PAGETB.**

**(VER)**

- Retorna a versão corrente do AutoLISP.

**(VPORST)**

- Retorna uma lista com a descrição da configuração de janelas (“viewports”) corrente.

**(XLOAD NADS) ((R11))**

Carrega uma aplicação desenvolvida em ADS.

**(XUNLOAD NADS) ((R11))**

Retira a aplicação ADS da memória.

**OBS1:** Se na lista de qualquer entidade a descrição de tipo de linha (código DXF 6) não existir, então a entidade tem tipo de linha definida por layer.

**OBS 2:** Se na lista de qualquer entidade a descrição de cor (código DXF 62) não existir, então a entidade tem cor definida por layer.

**OBS 3:** ((R 11)) indica código disponível a partir do Release 11.

**ENTIDADE: LINE**

<b>CÓDIGO</b>	<b>DESCRIÇÃO</b>
0	tipo do elemento: “LINE”
-1	nome da entidade

---

6	tipo de linha
8	layer
10	primeiro vértice
11	segundo vértice
38	elevação ( "elevation")
39	altura ("thickness")
62	cor da linha
210	direção de extrusão

**Ex:** ((0."LINE") (8. "0") (10 (3.0 4.0 0.0)) (11 (5.0 6.5 0.0)) (-1 . <Entity name: E0000100>) (210 (0.0 0.0 1.0)))

**ENTIDADE: CIRCLE**

<b>CÓDIGO</b>	<b>DESCRIÇÃO</b>
0	tipo de elemento: "CIRCLE"
-1	nome da entidade
6	tipo de linha

<b>CÓDIGO</b>	<b>DESCRIÇÃO</b>
8	Layer
10	Centro do círculo
38	Elevação ("elevation")
39	Altura ("thickness")
40	Raio do círculo
62	Cor do círculo

---

210 Direção de extrusão

**Ex:** ((0. "CIRCLE") (8."0") (10 (3.0 4.0 0.0 )) (40 . 2.0)

(-1 . <Entity name: E0000200>) (210 (0.0 0.0 1.0)))

**ENTIDADE: ARC**

**CÓDIGO DESCRIÇÃO**

0 Tipo de elemento: "ARC"

-1 Nome da entidade

6 Tipo de linha

8 Layer

10 Centro do arco

38 Elevação ("elevation")

39 Altura ("thickness")

40 Raio do arco

50 Ângulo inicial

51 Ângulo final

62 Cor do arco

210 Direção de extrusão

**Ex:** ((0 . "CIRCLE") (8 ."0") (10 (3.0 4.0 0.0)) (40 . 2.0)

(-1 . <Entity name: E00002000>) (210 (0.0 0.0 1.0)))

**ENTIDADE: TEXT**

<b>CÓDIGO</b>	<b>DESCRIÇÃO</b>
0	tipo de elemento: "TEXT"
-1	nome da entidade
1	texto
6	tipo de linha
7	estilo de fonte de texto
8	layer
10	ponto de inserção
11	pon to de alinhamento
38	elevação ("elevation")
39	altura ("thickness")
40	altura do texto
41	fator de escala da largura
50	ângulo de rotação
62	cor do texto
71	flag de geração de texto:  0: texto normal  2: texto escrito de trás para fr ente (espelhado em Y)  4: texto escrito de cabeça para baixo (espelhado em X)
72	flag de justificação horizontal do texto:

---

- 0: justificado "left"
- 1: justificação "center"
- 2: justificação "right"
- 3: justificação "align"
- 4: justificação "midle"
- 5: justificação "fit"

Veja continuação na próxima página:

<b>CÓDIGO</b>	<b>DESCRIÇÃO</b>
73	Flag de alinhamento vertical ((R11)) <ul style="list-style-type: none"> <li>0: alinhado pela linha base</li> <li>1: alinhado pela linha inferior</li> <li>2: alinhado pela linha central</li> <li>3: alinhado pela linha de topo</li> </ul>
210	Direção de extrusão

**Exemplo:**

```
((0. "TEXT") ( 8."0") (10 (.30 4.0 0.0 )) (40 . 2.0)
(41 . 1.0) (50 . 0.0) (70 . 0) (71 . 0) (72 . 0) (73 .0)
(-1. <Entity name: E0000400>) (210 (0.0 0.0 1.0)))
```

**ENTIDADE : POLYLINE**

<b>CÓDIGO</b>	<b>DESCRIÇÃO</b>
0	Tipo de elemento: "POLYLINE"
-1	Nome da entidade
6	Tipo de linha
8	Layer
38	Elevação ("elevation")
39	Altura ("thickness")

---

40	Espessura inicial ("start width")
41	Espessura final ("end width")
62	Cor da polyline
66	"Vértices follow flag" : é sempre 1
67	Tipo de polyline: 0: polyline aberta 1: polyline fechada 2: "curve fit" polyline 4: "spline fit" polyline 8: 3D polyline 16: 3D polygon mesh (ver código 75 a seguir) 32: polygon mesh fechado na direção N 64: polyface mesh ((R11))
68	No caso de 3D MESH: número de vértices na direção M
69	No caso de 3D MESH : número de vértices na direção N
70	No caso de 3D MESH: número de vértices na direção M com "smooth"
71	No caso de 3D MESH: número de vértices na direção N com "smooth"
72	No caso de 3D MESH: 0: superfície simples 5: superfície tipo B-spline quadrática 6: superfície tipo B-spline cúbica 8: superfície de Bezier
210	Direção de extrusão

**Exemplo:** ((0. "POLYLINE") (8. "0") (40. 0.0) (41 . 0.0) (66 . 1)

(70 . 0) ( 71 . 0) (72 . 0) (73 . 0) (74 . 0) (75 . 0)

(-1 . <Entity name: E0000500>) (210 (0.0 0.0 1.0) ))

## ENTIDADE: VERTEX

CÓDIGO	DESCRIÇÃO
--------	-----------

---

---

0	Tipo de elemento: "VERTEX"
-1	Nome da entidade
6	Tipo de linha
8	Layer
10	Coordenada do vértice
39	Altura ("thickness")
40	Espessura inicial ("start width")
41	Espessura final ("end width")
42	Fator "BULGE" de definição de arco
	No caso de "curve fit": tangente da curva
62	Cor
70	Tipo de vértice:
	0: vértice simples
	1: vértice criado por "curve fitting"
	2: no caso de "curve fit": vértice de definição de tangente
	8: vértice criado por "spline fitting"
	16: ponto de controle da spline
	32: vértice de polyline 3D
	64: vértice de 3D MESH
	128: vértice de polyface mesh ((R11))

---

**Exemplo:** ((0. "VERTEX" (8 . "0") (40 . 0.0) (41 . 0.0) (42 . 0.0)  
(70 . 0) ( -1. <Entity name: E0000600>))

#### ENTIDADE: SEQEND

CÓDIGO	DESCRIÇÃO
0	Tipo de elemento: "SEQEND"
-1	Nome da entidade
6	Nome da entidade "pai"
8	Layer

#### Exemplo:

((0 . "SEQEND") (8. "0") (-1 . < Entity name: E0000700>)  
(-2 . < Entity name:E0000500>)

#### ENTIDADE: POINT

CÓDIGO	DESCRIÇÃO
0	Tipo de elemento: "POINT"
-1	Nome da entidade
6	Tipo de linha
8	Layer
10	Coordenada do ponto
38	Elevação ("elevation")

---

39	Altura (“thickness”)
50	Ângulo de rotação
62	Cor do ponto
210	Direção de extrusão

**Exemplo:**

```
((0. “POINT”) ( 8 . “0”) (10 (3.0 4.0 0.0)) (50 0.0)
(-1 . <Entity name: E0000800>) (210 (0.0 0.0 1.0)))
```

**ENTIDADE: SOLID**

<b>CÓDIGO</b>	<b>DESCRIÇÃO</b>
0	Tipo de elemento: “SOLID”
-1	Nome da entidade
1	Tipo de linha
8	Layer
10	Coordenada do primeiro vértice
11	Coordenada do segundo vértice
12	Coordenada do terceiro vértice
13	Coordenada do quarto vértice
38	Elevação (“elevation”)
39	Altura (“thickness”)
62	Cor do sólido
210	Direção de extrusão

**Exemplo:**

```
((0. “SOLID”) (8 . ”0”) (10 (3.0 4.0 0.0))
(11 (3.0 5.0 0.0)) (12 (7.0 5.5 0.0)) (13 (2.0 4.0 0.0)
(-1 . <Entity name: E0000900>) (210 (0.0 0.0 1.0)))
```

**ENTIDADE: INSERT**

<b>CÓDIGO</b>	<b>DESCRIÇÃO</b>
0	Tipo de elemento: "INSERT"
-1	Nome da entidade
2	Nome do bloco
6	Tipo de linha
8	Layer
10	Ponto de inserção
38	Elevação ("elevation")
41	Fator de escala em X
42	Fator de escala em Y
	Fator de escala em Z
50	Ângulo de rotação
62	Cor
210	Direção de extrusão

**Exemplo:** ((0 "INSERT") (2 . "A1") (10 (3.0 4.0 0.0)) (41 . 1.0)  
 (42 . 1.0) (43 . 1.0) (50 . 0.0)  
 (-1 . <Entity name: E0000A00>) (210 (0.0 0.0 1.0)))

#### **ENTIDADE: ATTDEF**

<b>CÓDIGO</b>	<b>DESCRIÇÃO</b>
0	Tipo de elemento: "ATTDEF"
-1	Nome da entidade
1	Valor "default" do atributo
2	"tag" do atributo
3	"prompt" do atributo
6	Tipo de linha
7	Estilo de fonte de texto
8	Layer
10	Ponto de inserção
38	Elevação ("elevation")
39	Altura ("thickness")

---

40	Altura do texto
41	Fator de escala da largura
50	Ângulo de rotação
62	Cor do atributo
70	Tipo de atributo 0: normal 1: atributo invisível 2: atributo constante 4: atributo com verificação 8: atributo pré-definido
71	Flag de geração de texto: 0: texto normal 2: texto escrito de trás para a frente (espelhado em Y) 4: texto escrito de cabeça para baixo (espelhado em X)
72	Flag de justificação do texto: 0: justificação "left" 1: justificação "center" 2: justificação "right" 3: justificação "align" 4: justificação "middle" 5: justificação "fit"
210	Direção de extrusão

**Exemplo:**

((0. "ATTDEF") (1 . "0") (2. "COD") (3. "Código")

(10 ( 3.0 4.0 0.0)) (40 . 2.0) (41. 1.0) (70 . 0) (71 . 0)

(72 . 1 ) (-1. < Entity name: E0000B00>) (210 (0.0 0.0 1.0)))

**ENTIDADE: ATTRIB**

---

---

<b>CÓDIGO</b>	<b>DESCRIÇÃO</b>
0	Tipo de elemento: "ATTRIB"
-1	Nome da entidade
1	Valor do atributo
2	"Tag" do atributo
3	"prompt" do atributo
6	Tipo de linha
7	Estilo de fonte de texto
8	Layer
10	Ponto de inserção
38	Elevação ("elevation")
39	Altura ("thickness")
40	Altura do texto
41	Fator de escala da largura
50	Ângulo de rotação
62	Cor do atributo
70	Tipo de atributo 0: normal 1: atributo invisível 2: atributo constante 4: atributo com verificação 8: atributo pré-definido
71	Flag de geração de texto:

---

	2: texto escrito de trás para a frente (espelhado em Y)
	4: texto escrito d cabeça para baixo (espelhado em X)
72	Flag de justificação do texto:
	0: justificação "left"
	1: justificação "center"
	2: justificação "right"
	3: justificação "align"
	4: justificação "middle"
	5: justificação "fit"
210	Direção de extrusão

**Exemplo:**

```
((0 . "ATTRIB") (1 . "90-A") (2 . "COD") (3 . "Código")
(10 (5.0 3.0 0.0)) (40 . 2.0) (41 . 1.0) (70 . 0) ( 71 . 0)
(72 . 1) (-1 . <Entity name: E0000C00>) (210 ( 0.0 0.0 1.0)
```

**ENTIDADE: 3DFACE**

<b>CÓDIGO</b>	<b>DESCRIÇÃO</b>
0	Tipo de elemento: "3DFACE"
-1	Nome da entidade
6	Tipo de linha
8	Layer
10	Coordenada do primeiro vértice
11	Coordenada do segundo vértice
12	Coordenada do terceiro vértice
13	Coordenada do quarto vértice
62	Cor da 3DFACE
70	Tipo de 3DFACE:

---

0:3DFACE normal  
 1: primeira aresta invisível  
 2: segunda aresta invisível  
 4: terceira aresta invisível  
 8: quarta aresta invisível

**Exemplo:**

```
((0."3DFACE") ( 8. "0") ( 10 ( 1.0 2.0 0.0)
(11 (2.0 2.0 0.0)0 (12 (3.0 4.0 0.0 ) (13 (5.0 2.0 0.0)
(70 . 0) (-1 . <Entity name: E0000D00>) (210 (0.0 0.0 1.0)))
```

**ENTIDADE: DIMENSION**

<b>CÓDIGO</b>	<b>DESCRIÇÃO</b>
0	Tipo de elemento: "DIMENSION"
-1	Nome da entidade
2	Nome do bloco (de dimensionamento)
6	Tipo de linha
8	Layer
10	Ponto de definição do cota
11	Ponto médio do texto de cota
12	Ponto auxiliar de definição de cota
13	Ponto auxiliar de definição de cota
14	Ponto auxiliar de definição de cota
15	Ponto auxiliar de definição de cota
16	Ponto auxiliar de definição de cota
38	Elevação ("elevation")

---

39	Altura ("thickness")
40	Comprimento da linha guia
50	Ângulo de rotação da cota
62	Cor da cota

Veja continuação na próxima página:

70	Tipo da cota:
	0: rodada, horizontal ou vertical
	1: alinhada
	2: angular
	3: diametral
	4: radial
	5: angular por três pontos ((R11))
	6: ordenada ((R11))
	64: ordenada em X ((R11))
210	Direção de extrusão

**Exemplo:**

```
((0 . "DIMENSION") (2 . "**D1") (10 (3.0 4.0 0.0)
(12 (5.0 4.0 0.0)) (11 (4.0 4.0 0.0)) (50 . 0.0)
(13 (0.0 0.0 0.0)) (14 (0.0 0.0 0.0) ) (15 ( 0.0 0.0 0.0)
(16 (0.0 0.0 0.0)) (70 . 0) (-1 . <Entity name: E0000E00>)
(210 ( 0.0 0.0 1.0)))
```