FACULTAD DE INGENIERIA U.N.A.M.
DIVISION DE EDUCACION CONTINUA

FACULTAD DE INGENIERIA U.N.A.M.
DIVISION DE EDUCACION CONTINUA

CENTRO DE INFORMACION Y DOCUMENTACION
"ING. BRUNO MASCANZONI"

El Centro de Información y Documentación Ing. Bruno Mascanzoni tiene por objetivo satisfacer las necesidades de actualización y proporcionar una adecuada información que permita a los ingenieros, profesores y alumnos estar al tanto del estado actual del conocimiento sobre temas específicos, enfatizando las investigaciones de vanguardia de los campos de la ingeniería, tanto nacionales como extranjeras.

Es por ello que se pone a disposición de los asistentes a los cursos de la DECFI, así como del público en general los siguientes servicios:

- **Préstamo interno.**
- **Préstamo externo.**
- **Préstamo interbibliotecario.**
- **Servicio de fotocopiado.**
- **Consulta a los bancos de datos: librunam, seriunam en cd-rom.**

Los materiales a disposición son:

- **Libros.**
- **Tesis de posgrado.**
- **Noticias técnicas.**
- **Publicaciones periódicas.**
- **Publicaciones de la Academia Mexicana de Ingeniería.**
- **Notas de los cursos que se han impartido de 1980 a la fecha.**

En las áreas de ingeniería industrial, civil, electrónica, ciencias de la tierra, computación y, mecánica y eléctrica.

El CID se encuentra ubicado en el mezzanine del Palacio de Minería, lado oriente.

El horario de servicio es de 10:00 a 19:30 horas de lunes a viernes.

Palacio de Mineria Calle de Tacuba 5 Primer piso Deleg. Cuauhtémoc 06000 México, D F. APDO. Postal M-2285
Teléfonos: 512-8955 512-5121 521-7335 521-1987 Fax 510-0573 521-4020 AL 26

FACULTAD DE INGENIERIA U.N.A.M.
DIVISION DE EDUCACION CONTINUA

A LOS ASISTENTES A LOS CURSOS

Las autoridades de la Facultad de Ingeniería, por conducto del jefe de la División de Educación Continua, otorgan una constancia de asistencia a quienes cumplan con los requisitos establecidos para cada curso.

El control de asistencia se llevará a cabo a través de la persona que le entregó las notas. Las inasistencias serán computadas por las autoridades de la División, con el fin de entregarle constancia solamente a los alumnos que tengan un mínimo de 80% de asistencias.

Pedimos a los asistentes recoger su constancia el día de la clausura. Estas se retendrán por el periodo de un año, pasado este tiempo la DECFI no se hará responsable de este documento.

Se recomienda a los asistentes participar activamente con sus ideas y experiencias, pues los cursos que ofrece la División están planeados para que los profesores expongan una tesis, pero sobre todo, para que coordinen las opiniones de todos los interesados, constituyendo verdaderos seminarios.

Es muy importante que todos los asistentes llenen y entreguen su hoja de inscripción al inicio del curso, información que servirá para integrar un directorio de asistentes, que se entregará oportunamente.

Con el objeto de mejorar los servicios que la División de Educación Continua ofrece, al final del curso deberán entregar la evaluación a través de un cuestionario diseñado para emitir juicios anónimos.

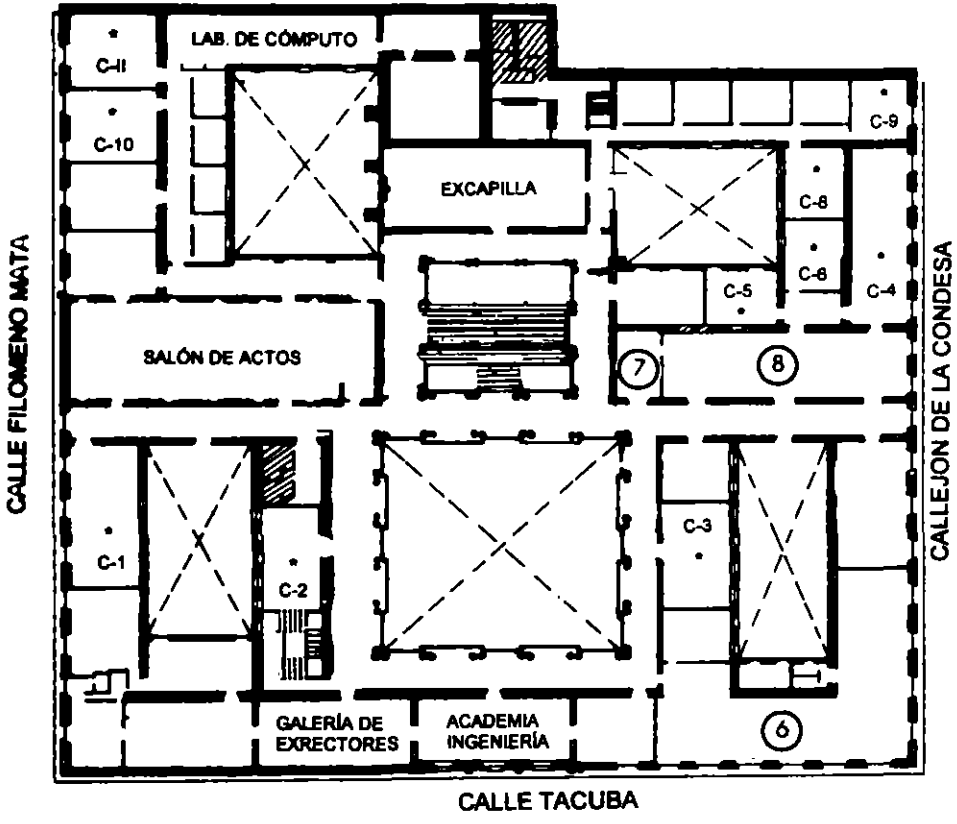
Se recomienda llenar dicha evaluación conforme los profesores impartan sus clases, a efecto de no llenar en la última sesión las evaluaciones y con esto sean más fehacientes sus apreciaciones.

Atentamente

División de Educación Continua.

Palacio de Mineria Calle de Tacuba 5 Primer piso Deleg Cuauhtémoc 06000 México, D.F. APDO. Postal M-2285
Teléfonos: 512-8955 512-5121 521-7335 521-1987 Fax 510-0573 521-4020 AL 26

PALACIO DE MINERIA



1er. PISO

GUÍA DE LOCALIZACIÓN

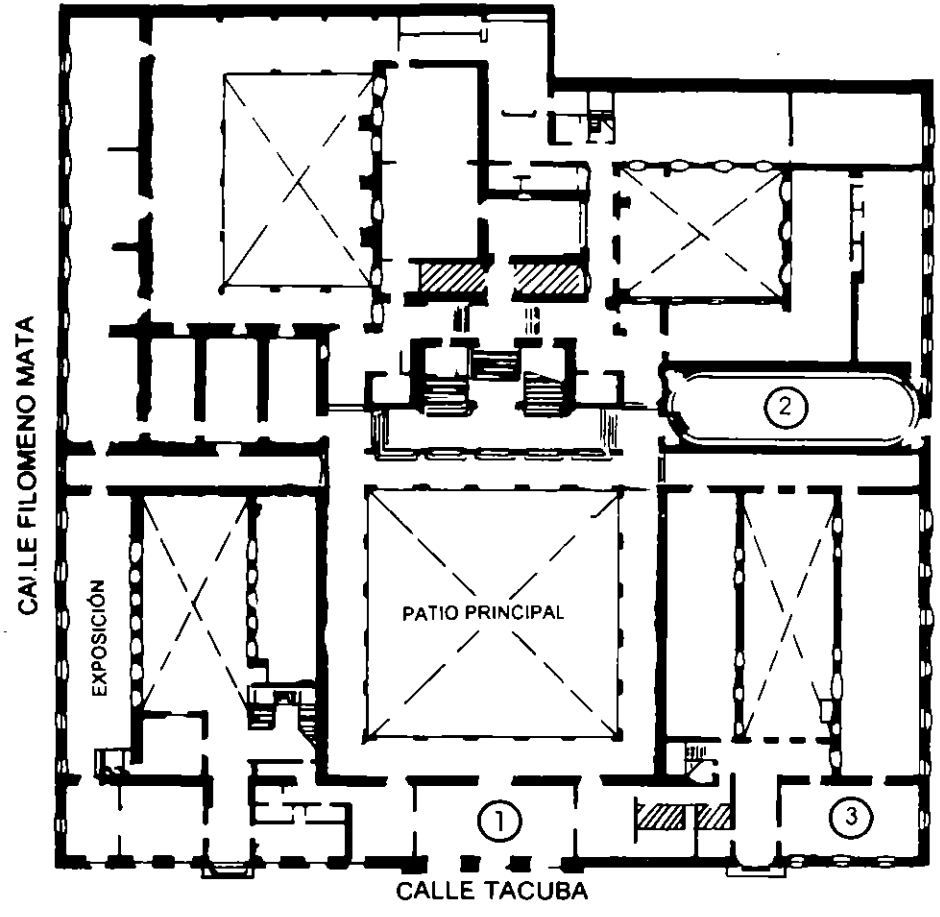
1. ACCESO
2. BIBLIOTECA HISTÓRICA
3. LIBRERÍA UNAM
4. CENTRO DE INFORMACIÓN Y DOCUMENTACIÓN "ING. BRUNO MASCANZONI"
5. PROGRAMA DE APOYO A LA TITULACIÓN
6. OFICINAS GENERALES
7. ENTREGA DE MATERIAL Y CONTROL DE ASISTENCIA
8. SALA DE DESCANSO

SANITARIOS

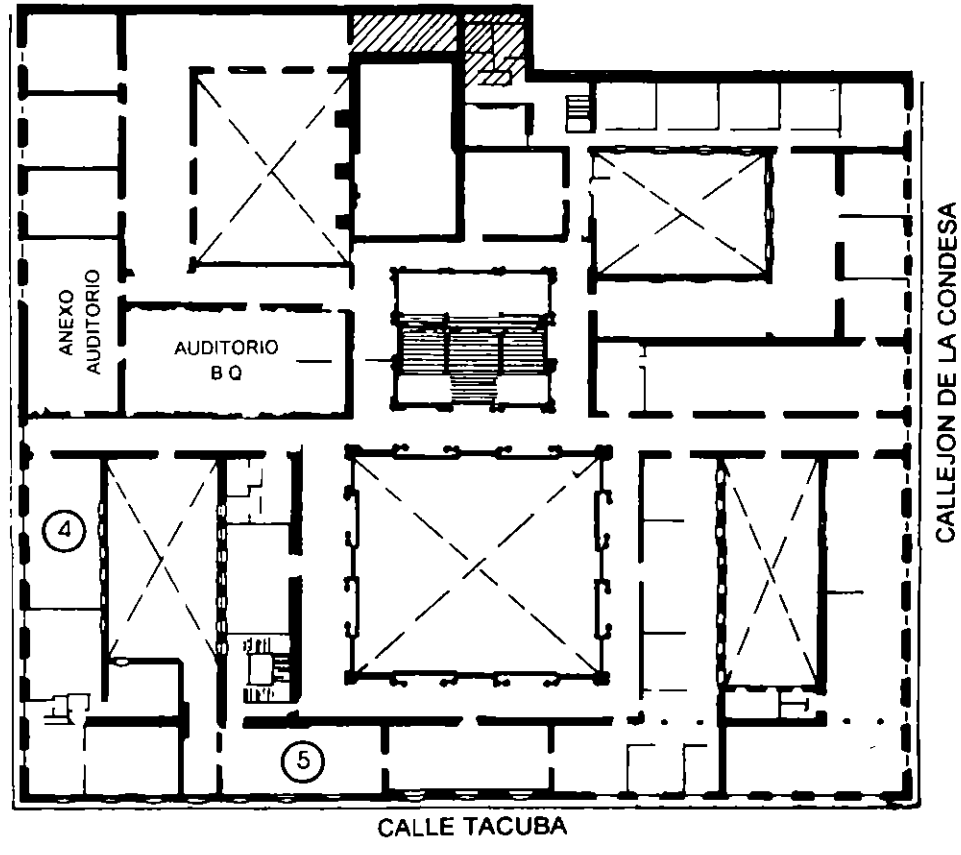
* AULAS

DIVISIÓN DE EDUCACIÓN CONTINUA
FACULTAD DE INGENIERÍA U.N.A.M.
CURSOS ABIERTOS

PALACIO DE MINERIA



PLANTA BAJA

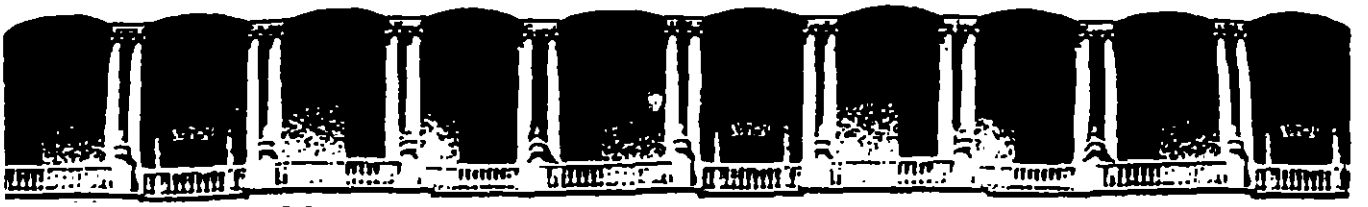


MEZZANINNE

DIRECTORIO DE ASISTENTES AL CURSO DE

AUTOLISP (LENGUAJE DE PROGRAMACION

APELLIDOS	NOMBRE	EMPRESA	CALLE	COLONIA	DELEGACION	TELEFONO
ARUIZU AYALA	DOMINGO	C F E	MISSISSIPPI 71	CUAUHTEMOC	CUAUHTEMOC	6433406
BERNARDINO PEALEZ	DAVID					6223886
BOLAOS SANCHEZ	HECTOR	INFORMATICA Y CONTROL DE OBRA SA DE CV	EJERCITO NACIONAL 254	ANZURES	MIGUEL HIDALGO	2547978
CHACON KURI	LAURA	INAMEX DE CERVEZA Y MALTA	KM 13 3 LOS REYES-TEXCOCO	CUAUTLALPAN.	COCO EDO DE MEXICO	9159210730
DEL VALLE MUNILLA	MARCO	INFORMATICA Y CONTROL DE OBRA SA DE CV	EJERCITO NAL 254 EDF C-02	ANZUREZ	MIGUEL HIDALGO	
GUTIERREZ SAUZA	ENRIQUE	GUSE CONSTRUCCIONES	PERIFERICO SUR 3680	COAPA	TLALPAN	6030657
PEREZ PALACIOS	ALEJANDRO	PROGRAMA UNIVERSITARIO DE ENERGIA	CIRCUITO EXTERIOR	C. U.	COYOACAN	6072875
REYES HERNANDEZ	ANTONIO	PEMEX	MARINA NACIONAL 329	HUASTECA	MIGUEL HIDALGO	2035226
VERA ORTIZ	RENE					5561211
VILLERS RUIZ	ABELARDO	INMOBIMEX SA DE CV				6755173



FACULTAD DE INGENIERIA U.N.A.M.
DIVISION DE EDUCACION CONTINUA

AUTOLISP

(LENGUAJE DE PROGRAMACION PARA AMBIENTE AUTOCAD)

MATERIAL DIDACTICO

SEPTIEMBRE - OCTUBRE

1995

Palacio de Minería Calle de Tacuba 5 Primer piso Deleg. Cuauhtémoc 06000 México, D.F. Tel.: 521-40-20 Apdo. Postal M-2285

INTRODUCCION A AUTOLISP

Autocad nos ofrece otras formas para la manipulación de objetos además de la normal, que consiste en dibujar y editar estos símbolos gráficos; se pueden crear sus propios menúes, así como seleccionar variables del sistema Autocad para obtener resultados más completos y rápidos; pero sin duda alguna el mecanismo más poderoso para manipular el entorno de Autocad es el lenguaje de programación AutoLISP.

AutoLISP es un lenguaje orientado a la manipulación de símbolos; por ello se encontrará que la forma de programar con él es totalmente diferente a la utilizada con los otros lenguajes de programación convencionales, como PASCAL, BASIC ó C.

En LISP se hace uso en gran medida de los paréntesis, que sirven para indicar el inicio y final de una lista. LISP procesa listas de símbolos, en vez de datos numéricos como los lenguajes FORTRAN, BASIC y C.

AutoLISP es un subconjunto del LISP (lenguaje desarrollado para la investigación en Inteligencia artificial) que significa LISt Processing (Procesado de listas). AutoLISP añade algunas funciones especiales que están diseñadas para la manipulación de dibujos de AutoCAD.

Como los sistemas CAD se orientan hacia la manipulación de símbolos gráficos y AutoLISP se basa en el uso de símbolos, es un excelente lenguaje para la programación de sistemas CAD.

LISP es un lenguaje que es evaluado en vez de interpretado ó compilado. Los lenguajes interpretados leen el texto del programa línea por línea y lo convierten a instrucciones de máquina, generando un archivo en código de máquina, el cual es ejecutado rápidamente.

Un lenguaje evaluado se encuentra entre uno interpretado y uno compilado. Cuando se encuentra por primera vez un bloque de código, este se convierte en código compacto; si dicho bloque se encuentra de nuevo mientras se ejecuta el programa, el evaluador detecta que ya ha sido evaluado y lo ejecuta.

Hay tres características de LISP que lo distinguen de la mayoría de los otros lenguajes de programación:

LISP manipula símbolos en vez de números

Es un lenguaje orientado al objeto en vez de ser un lenguaje procedimental.

Es un lenguaje que valúa en vez de interpretar ó compilar.

Resumiendo, las ventajas que se puedan obtener utilizando AutoLISP son, entre otras:

La sistematización de tareas o procedimientos que agilicen el trabajo con Autocad.

Permite automatizar el desarrollo de proyectos, es decir, realizar un agrupamiento de tareas planificadas que siguiendo una estrategia de desarrollo culmine en un objetivo común, teniendo para esta misma estrategia una o más vías de desarrollo que de antemano ya se a hallan previsto.

Se tiene una explotación máxima del paquete ya que se manipula directamente la base de datos sin pesar por el uso convencional de los comandos básicos de Autocad.

Creación de nuevos comandos que se incorporan a los comandos básicos de Autocad.

Interacción mayor con otros paquetes de dibujo e inclusive con otros programas de aplicación.

CONCEPTOS DE PROGRAMACION

La programación es un conjunto de métodos que permiten planificar la solución de un problema. La técnica de la programación estructurada está orientada hacia la elaboración de programas que resulten fáciles de comprender. guardando una estructura sencilla y que observe ciertos estándares, con la finalidad de que otras personas y el mismo programador puedan acceder a este para futuras módificaciones.

La programación estructurada se basa en la aplicación adecuada de las siguientes reglas :

a) Utilización de figuras lógicas que sólo serán :

La secuencia, o descripción de un paso de progra-mación.

La decisión o IF THEN ELSE, que define dentro del flujo de un programa, una condición para que una secuencia pueda ser realizada.

La repetición o WHILE, que define una serie de secuencias que serán realizadas repetidamente en forma ciclica.

La solución de cualquier problema, independientemente de su grado de complejidad, podrá obtenerse eficazmente combinando las tres figuras lógicas.

b) Programar modularmente.

La complejidad de un programa se ve reducida, si se respeta este lineamiento que demanda organizar su estructura en módulos pequeños que desempeñen funciones especificas. Una vez creados todos los módulos, se integran para conformar el programa completo.

El uso de las tres figuras lógicas se recomienda hacerlo en forma escrita (Pseudocódigo) o podrá hacerse también en forma gráfica (Diagramas de flujo). Ejemplos:

a) Obtener la solución del cociente entre dos valores A y B (A / B) evitando dividir entre cero . El Pseudocódigo podrá presentar la siguiente solución :

Se tiene A y B

IF (B es igual a 0) THEN

Error : División entre cero

ELSE

Resultado : A / B

ENDIF

b) Obtener la suma de los primeros N números naturales . El Pseudocódigo podría ser :

Se tiene N

Valor = 1

Suma = 0

WHILE (Valor sea menor o igual a N)

Suma = Suma + Valor

Valor = Valor + 1

ENDDO

Resultado : Suma

Como se puede apreciar en los ejemplos la solución de un problemas utilizando la técnica de la Programación Estructurada através del Pseudocódigo es la forma más amigable de plantear una de las diversas soluciones que llevarían a la obtención de un mismo resultado. Obsérvese la forma tan descriptiva de desglosar el problema, casi en forma "platicada", el uso de las figuras lógicas sólo indica que proceso se deberá efectuar (asignación, decisión o repetición), considere también el uso de una notación matemática muy sencilla (variables y operadores aritméticos).

Tengase presente que la programación de la solución de un problema es sólo una fase y se complementará con la fase de codificación (uso de un lenguaje de programación para computadora) para finalmente determinar una solución por computadora. Una correcta programación redundará en la obtención de una solución eficiente y rápida en la computadora.

Finalmente, se sugieren las siguientes recomendaciones para una mejor codificación de programas para AutoLISP :

Uso de sangrías o tabulados para indicar dependencias entre instrucciones.

Desglosar un problema en partes elementales que subdividan en diferentes procedimientos el plan de solución. Se enfoca toda la atención primero a un procedimiento, después al siguiente y así hasta al último; seguro de que todos los procedimientos funcionaron en forma aislada, se integran y en conjunto se obtendrá una solución más sencilla.

Documentación de todos las codificaciones, uso de comentarios a lo largo de la codificación para explicar las partes más sobresalientes en el programa. Procure ser directo y sencillo en el uso de los comentarios.

ENTORNO DE AUTOLISP

El trabajo con AutoLISP lo podemos ver como un complemento del uso tradicional de Autocad con lo que vamos a lograr mejores y más rápidos resultados. En una forma sencilla puede verse el uso de AutoLISP como la aplicación de nuevos comandos, mas poderosos, desde Autocad. De la correcta combinación de estos nuevos comandos, se podrán obtener resultados cada vez más eficientes.

AutoLISP es un producto adicional que es incorporado al paquete normal de Autocad, para tener disponibilidad de AutoLISP deberá verificarse lo siguiente:

Tener una computadora con memoria principal de 640 kb, por lo menos, y de preferencia con una expansión de 340 kb o mayor.

Con 512 kb es posible incorporar AutoLISP, pero en general se presentan muchos conflictos con la compartición de esta memoria y los programas residentes necesarios (por ejemplo con el programa que maneja el "mouse").

Si se cuenta con una versión de Autocad menor de la 10.0, revise la situación de las siguientes variables de ambiente:

LISPHEAP

LISPSTACK

A partir de la versión 10. dichas variables de ambiente se adecuan automáticamente permitiendo un uso más directo de AutoLISP, aunque sí se tuviera problemas de espacio en memoria principal se revisaría los valores de dichas variables.

Revisar la disponibilidad de Autolisp a partir de Autocad.

Invocar Autocad:

ACAD

Seleccionar opción 5 (configuración de Autocad) del menú principal.

Del submenú de configuración, seleccionar la opción 8 (configuración de los parámetros del sistema).

Seleccionar opción 7 (Disponibilidad de AutoLISP) y verificar que se presente:

Do you want AutoLISP enabled? <Y>: Y

A partir de la versión 10 se presenta la siguiente pregunta:

Do you want to use Extended AutoLISP? <N>: N

Aquí responda con "N" indicando que por el momento no será necesario utilizar la extensión de AutoLISP.

Salvar la actual configuración que contempla el uso de AutoLISP:

Keep configuration changes? <Y>: Y

Regreso al menú principal, en este momento está disponible AutoLISP.

AutoLISP es un programa adicional de Autocad que solamente puede ser invocado desde el editor de dibujo, es decir, es un producto que no puede ser independiente de Autocad. AutoLISP es un programa intérprete (evaluador, propiamente dicho).

Entrando al ambiente de Autocad, ya puede comenzarse a trabajar y AutoLISP es accesado a través del intérprete AutoLISP: cuando se teclea cualquier cosa al prompt "**Command:**" AutoCAD, el intérprete primero lee la información para ver si se trata de una función de AutoLISP. Si es así, AutoLISP la ejecuta; en caso contrario AutoCAD se encarga de procesarla.

Como se ve, uno pude interactuar con AutoLISP intérprete en ambiente AutoCAD cuando se encuentra el prompt "**Command:**", por ejemplo:

Command: (+ 4.5 3)

En el ejemplo anterior se da una expresión de AutoLISP, donde se le indica que evalúe la función "**+**".

Nótese también que la expresión se encuentra entre paréntesis. Esta es la estructura básica para todos los programas en AutoLISP. Todo lo que se quiera evaluar con AutoLISP, desde la expresión más simple hasta el programa más complejo, deberá escribirse con esta estructura.

El resultado de evaluar una expresión es llamado el valor de la expresión.

La forma en que se va a trabajar con AutoLISP será la siguiente:

Creación de un programa de instrucciones en AutoLISP utilizando para tal efecto un Editor. Puede utilizarse el editor del Sistema operativo MS-DO (EDLIN) o cualquier otro. El programa de instrucciones en AutoLISP será un archivo en formato ASCII. Para este curso se utilizará el editor NORTON, por lo que la secuencia de trabajo será:

Desde Autocad invocar el editor NORTON:

Command: SHELL

EDIT

--creación del programa *.LSP y al concluirlo se regresa a Autocad--

Command:

Se pondrá crear en el ambiente de Autocad el programa con instrucciones AutoLISP pero como serán realizadas en memoria principal se perderá dicho programa una vez interpretado. Procúrese trabajar utilizando un editor y asignando un nombre adecuado al archivo, revisando que la extensión de este archivo sea ".LSP" . Ejemplo de un nombre de archivo con instrucciones AutoLISP:

FUNCION.LSP

Incorporar el archivo AutoLISP a Autocad para su interpretación:

Command: (LOAD "FUNCION")

(disp)

El mensaje anterior indica que en el archivo FUNCION.LSP venia la función "disp" que fue interpretada y está lista para ser evaluada desde Autocad.

Para evaluar funciones, todo lo que se haga en Autolisp serán funciones, simplemente se hará escribiendo su nombre junto con los posibles parámetros o datos iniciales que requiera:

Command: (disp)

Recuerde que como función, siempre se hará referencia a ella entre paréntesis. Un ejemplo de función con parámetros es:

Command: (disp '(1 1) '(4 4))

En el ejemplo anterior la función "disp" requiere como datos de entrada dos puntos o listas.

TIPOS DE DATOS EN AUTOLISP

Así como en los lenguajes de programación tradicionales se definen los diferentes tipos de valores a partir de los cuales se construirá la mitad de cualquier concepto de programación, en AutoLISP se mantiene el mismo principio. Un dato o valor define la manera en que la información se podrá representar, así por ejemplo, en Autocad gran parte de la información que se necesita manipular son puntos, de tal manera que AutoLISP tiene definido un tipo de valor que representa a esta información (en este caso existe el tipo de dato lista).

En AutoLISP los tipos de datos que se reconocen son:

Enteros

Son los números enteros (sin fracción decimal) que podrán ser positivos o negativos. Cualquier operación que se realice entre enteros resultará invariablemente en otro valor entero. Ejemplo:

```
- 32768

+ 32767

0

( / 25 2 ) -----> 12
```

En el último ejemplo, se define una expresión con la función cociente (/) donde se dividirá 25 entre 2, como bien sabemos, el resultado matemáticamente hablando será 12.5, pero como en la relación operan dos valores enteros el resultado será otro entero, de tal manera que el resultado será 12 .

Reales

Son todos los números reales conocidos (tradicionalmente son todos aquellos números con punto y fracción decimal). Dentro de estos valores podrán definirse magnitudes mucho más grandes que las definidas por un valor entero con la diferencia que el manejo de valores enteros es más ágil y directa que la manipulación de reales. De la misma forma que para los valores enteros, toda operación entre reales resultará en otro real. Ejemplo:

```
3.1416

19.

-0.51012
```

.1843 -----> error, debe ser 0.1843

(/ 25. 2) -----> 12.5

observe la siguiente expresión:

(/ 25. 2) -----> 12.5

el resultado será un real, aunque la operación mezcló un valor real (25.) con un valor entero (2), todos los valores reales tienen dentro de los valores numéricos una categoría mayor; de tal manera que en la mezcla de tipos numéricos, siempre resultará un real, si en la relación existiese un real.

Es importante recalcar que cuando se realicen operaciones aritméticas (suma, resta, multiplicación y división) se conozcan perfectamente bien, que tipos de datos numéricos se manejan (reales o enteros) para poder determinar correctamente el tipo de valor numérico que resultará. Se recomienda que no se utilice la mezcla de tipos numéricos, aunque puede existir.

String

Este tipo de dato es también conocido como tipo de dato cadena o alfanumérico, normalmente se denomina "string" y define a todos aquellos valores que no son numéricos (ni reales o enteros). Ejemplo:

"Proporciona el punto 2"

"Curso AutoLISP"

"123"

(STRCAT "Curso AutoLISP" " DECFI UNAM")

Observe que todos los valores "string" se marcan entre comillas. En el último ejemplo se hace uso de la función de AutoLISP "STRCAT" con la cual se unen cadenas o "strings", el resultado de esta función será otro valor "string" cuyo valor será, para nuestro ejemplo, "Curso AutoLISP DECFI UNAM".

Generalmente, los valores "string" se utilizan como complemento de alguna instrucción que necesite mostrar al usuario algún mensaje en pantalla.

Todos los valores cadena tienen un valor numérico equivalente, dicho valor se utiliza internamente para su manejo, este valor se conoce como código ASCII. A continuación se muestra la representación de los valores cadena más comunes

LISTAS

Son los tipos de datos fundamentales de AutoLISP a través de los cuales se hará referencia a:

Puntos, en 3 y 2 dimensiones

La base de datos de AutoCAD

Son ejemplos de listas :

(4 5 0)

(1. "CIRCLE")

En el primer ejemplo se establece el punto de coordenadas X, Y, Z , donde X vale 4, Y vale 5 y Z vale 0.

En el segundo ejemplo se define una lista muy particular que se conoce como sublista, aquí se tiene una pequeña muestra de como se almacena la información generada por Autocad en la base de datos.

Nótese que todas las listas tendrán siempre que marcarse encerradas entre paréntesis y sus elementos vendrán casi siempre separados por lo menos por un espacio en blanco.

Para el manejo de una lista existe una diversidad de funciones que posteriormente se hará referencia, pero en su manipulación se presentará las siguientes partes de una lista:

(4 5 0) ---- lista normal

4 cabeza de la lista

(5 0) cuerpo de la lista

Descriptores de archivos

Son referencias indirectas que se establecen para indicar el uso de una archivo ya sea para lectura o para escritura. Estos valores son generalmente, nombres tipo variable:

TEXTO

En este ejemplo la referencia "TEXTO" relacionará el uso de un archivo para lectura o para escritura (dependiendo de la función que se utilice). Posteriormente se hará referencia el uso de archivos.

Nombres de entidad

Todos los objetos generados en Autocad (círculos, líneas, textos, bloques, etc) tiene una referencia interna con la cual son identificados dentro de la base de datos. Con el nombre de la entidad u objeto es la llave para extraer toda la información referente a dicho objeto. Ejemplo:

6000142a

Este nombre de entidad es asignado por Autocad y generalmente presentará la anterior apariencia, agrupación de números y letras. Estos nombres son obtenidos a partir del uso de funciones adecuados para su localización y extracción de la base de datos.

Conjuntos

Es el agrupamiento de uno o más objetos de Autocad. Recuérdese el uso del comando SELECT de Autocad, un conjunto tiene su mismo objetivo: seleccionar un grupo de objetos para realizar con este una determinada operación.

Para definir un conjunto se utilizarán funciones de AutoLISP adecuadas que más adelante se mencionarán, dicho conjunto o conjuntos serán identificados por un nombre.

Funciones intrínsecas

Son todas aquellas funciones no generadas por el usuario. Un programa en AutoLISP se compone invariablemente de las siguientes partes:

Datos o valores a manipularse

Funciones intrínsecas o de AutoLISP

Funciones creadas por el usuario.

Esta composición siempre estará presente.

EXPRESIONES EN AUTOLISP

Una expresión de AutoLISP deberá incluir un operador seguido de los elementos que serán operados. Un operador es una instrucción que toma alguna acción específica; por ejemplo, se tienen los operadores matemáticos (+) para sumar o (/) para dividir.

AutoLISP se refiere al operador como a una función y a los elementos a ser operados como a los argumentos de la función.

En la expresión:

(+ 4.5 3)

el signo " + " es la función y los números 4.5 y 3 son los argumentos de la función. Todas las expresiones de AutoLISP, sin importar tamaño, seguirán esta estructura e irán encerradas entre paréntesis:

(función arg1 arg2 ...)

Los paréntesis en una expresión deberán estar balanceados, es decir que por cada paréntesis izquierdo deberá haber un paréntesis derecho. Si se introduce en el intérprete de AutoLISP una expresión desbalanceada, se provocará el siguiente mensaje:

n>

donde n es el número de paréntesis requeridos para completar la expresión. Si manda este mensaje, se deberán dar el número de paréntesis indicado por "n" para que se restablezca el prompt

"**Command:**"

Nótese que se requiere al menos un espacio entre los elementos de una expresión. El orden, el tipo de argumentos y el número de estos que siguen a una función varía dependiendo de la función, pero la estructura de la expresión es siempre la función seguida por los argumentos, todo entre paréntesis.

Cada vez que se encuentra una expresión, AutoLISP evalúa todo, no sólo la expresión; los argumentos también los evalúa. En el ejemplo anterior, AutoLISP evalúa el argumento 4.5 y el argumento 3; en éste caso los números los evalúa a sí mismo.

Debido a que AutoLISP evalúa los argumentos, se pueden usar expresiones como argumentos de una función; por ejemplo

(+ (- 2 3) 8) -----> 7

Variables

Las variables son localidades de memoria que permiten guardar valores para poder usarlos posteriormente. El valor de la variable puede cambiar en el curso de un programa.

Existen varios tipos de valores que podemos guardar en variables:

Entero (INT)

Real (REAL)

Cadena (STRING)

Lista (LIST)

Archivo (FILE)

Entidad (Entity)

Conjuntos designados (Selection Set)

Símbolo (SYM)

Función (SETQ)

ASIGNACION DE VALORES A VARIABLES

A las variables se les asignan valores con la función "SETQ". Como se ha visto, una función puede ser un simple operador, como el de suma, pero también puede consistir de un conjunto de instrucciones más complejas igual que un programa. Ejemplo:

Command: (SETQ perim 8.5)

Para obtener el valor de una variable, dentro del ambiente de Autocad, se teclea un signo de admiración "!" y el nombre de la variable:

Command: ! Perim -----> el valor 8.5 es regresado

Pruebe la siguiente expresión y analice como trabaja:

Command: (SETQ perim (+ perim 1))

Asignando valores cadena:

Command: (SETQ nombre "Ejemplo")

Command: (SETQ texto "12")

Los valores cadena "Ejemplo" y "12" se guardan en las variables 'nombre' y 'texto' respectivamente. Podría pensarse que la expresión (SETQ perim 8.5) causará error debido a que AutoLISP siempre evalúa los argumentos antes de ejecutar la función. Entonces evaluaría 'perim', el cual tiene valor nulo (pues no se le ha asignado valor alguno) e intentaría asignarle a éste el valor 1, lo cual es imposible: (SETQ valor valor).

¿ Por qué entonces no marca error en la función 'SETQ' ?

La función 'SETQ' evita que se evalúe el primer argumento. Otra forma de evitar evaluar un argumento es anteponiendo un apóstrofe a éste; ejemplo:

(SET 'perim 8.5)

La función 'SET' es semejante a 'SETQ', sólo que evalúa todos los argumentos de la función, a menos que se les anteponga un apóstrofe.

A continuación se lista algunas funciones matemáticas más comunes:

a) Que aceptan múltiples argumentos

(+ número número...) Suma

(- número número...) Resta

(* número número...) Multiplicación

(/ número número...) División

(max número número...) Regresa el mayor número de la lista

(min número número...) Regresa el menor número de la lista

(rem número número...) Residuo de la división

b) Que aceptan uno o dos argumentos

(1+ número) Adiciona 1

(1- número) Resta 1

(abs número) Regresa valor absoluto

(exp número potencia) Numero a la potencia dada

(fix real) Convierte real a entero

(float entero) Convierte entero a real

(gcd entero entero) Encuentra máximo común denominador

(log número) Logaritmo natural de número

(sqrt número) Raíz cuadrada de número

ASIGNACION DE VALORES DE LISTAS

En muchas de las instrucciones de AutoCAD se manejan puntos para definir una posición en el área de dibujo; éstos se definen por coordenadas que no es un simple valor, sino un grupo de valores. Por ser un punto un grupo de valores, en AutoLISP deberá usarse una lista para definirlo. Por ejemplo, para almacenar el punto con coordenadas 7,9 en la variable punto1, se puede utilizar la función 'LIST' :

Command: (SETQ punto1 (LIST 7 9))

Command: (SETQ punto1 '(7 9))

Como se ve, la función 'LIST' regresa una lista de valores. Así también una variable no sólo acepta valores simples; también listas pueden almacenarse en variables:

```
( SETQ  VX  1.0 )
( SETQ  VY  5.2 )
( SETQ  p1  ( LIST  VX  VY ) )
! p1     -----> ( 1.0  5.2 )
```

Funciones CAR y CADR

Para manejar los elementos de listas, existen las funciones CAR y CADR. Suponiendo que se quiere manejar sólo la coordenada X del punto p1; teclee:

Command: (CAR p1) -----> 1.0

Ahora para accesar la coordenada Y, teclee:

Command: (CADR p1) -----> 5.2

éstos valores pueden asignárseles a variables:

```
( SETQ  p1X  ( CAR  p1 ) )
( SETQ  p1Y  ( CADR  p1 ) )
```

Funciones básicas

Funciones de lectura

AutoLISP cuenta con una serie de funciones que permiten que se de información desde el área de dibujo.

Estas funciones llevan el prefijo GET; algunas de estas son:

GETPOINT

Permite proporcionar vía teclado o con "mouse" un punto.

GETCORNER

Permite dar la esquina opuesta de una ventana (window). Requiere que se le dé la primera esquina.

GETANGLE

Permite dar un ángulo. Regresa el valor en radianes.

GETDIST

Permite dar una distancia con las teclas o "mouse".

Ejemplos:

Command: (SETQ punto1 (GETPOINT))

El prompt "**command**" se queda esperando a que se le teclee las coordenadas de un punto o se defina un punto con "mouse".

Los valores de las coordenadas se le asignan a la variable "punto1" en forma de lista.

Se puede usar un prompt (mensaje) en las funciones GET:

Command: (SETQ punto1 (GETPOINT "oprima el primer punto":))

Esta instrucción espera por un punto, pero al mover el "mouse" se ve una caja elástica según lo movamos, teniendo fija la esquina definida con punto1.

Funciones para conversión de datos

(ANGTOS real) convierte de número a string

(ASCII string) convierte un string a su correspondiente código ASCII

(ATOI string) convierte de string a entero

(ITOA integer) convierte de entero a string

(CHR integer) convierte un entero en un string ASCII

(ATOF string) convierte de string a real

Manejo de listas

(CAR) Primer elemento de una lista

(CADR) Segundo elemento de una lista

(CDR) Regresa una lista sin el primer elemento

Ejemplos:

(LIST 1 2 3) -----> lista1

(CAR lista1) -----> 1

(CADR lista1) -----> 2

(CADR (CDR lista1)) -----> 3

(CDR lista1) -----> (2 3)

(NTH) Elemento enésimo de una lista

(LENGTH) Número de elementos

(REVERSE) Lista invertida

(LAST) Ultimo valor de la lista

(APPEND) Junta listas en otra lista

(CONS) Agrega un primer elemento a una lista. Poner lista asociativa

(ASSOC) De una lista obtiene una sublista a partir de una referencia (1er. elemento de la sublista)

(SUBST) Sustituye elementos en una lista

Ejemplos:

(LIST 1 2 3) -----> lista

(LIST (1 2) (3 4)) -----> lista1

(NTH 2 lista) -----> 3

(LENGTH lista) -----> 3

(LENGTH lista1) -----> 2

(REVERSE lista) -----> (3 2 1)

(REVERSE lista1) -----> (3 4) (1 2)

(LAST lista) -----> 3

(LAST lista1) -----> (3 4)

(APPEND lista lista1) -----> (1 2 3 (1 2)(3 4))

(CONS 100 lista) -----> (100 1 2 3)

(CONS '(56) lista1) -----> ((5 6)(1 2)(3 4))

(ASSOC 1 lista) -----> (1 2 3)

(ASSOC 3 lista1) -----> (3 4)

(ASSOC 4 lista1) -----> nil

(SUBST 8 1 lista) -----> (8 2 3)

(SUBST '(5 6) '(1 2) lista1) -----> ((5 6)(3 4))

(LIST (CONS 8 "0")) -----> (8 . "0") lista asociativa

Manejo de caracteres

(STRCASE) Convierte cadenas en mayúsculas o minúsculas

(STRCAT) Junta cadenas

(SUBSTR) Substrae una parte de la cadena

(STRLE N) Tamaño de la cadena

(ITOA) Transforma un entero en cadena

(ATOI) Transforma cadena en entero

(ATOF) Transforma cadena en real

Ejemplos:

(SETQ TEXTO ("A B C D E F"))

(STRCASE TEXTO) -----> "abcdef"

(STRCASE TEXTO T) -----> "ABCDEF"

(STRCAT TEXTO "123") -----> "ABCDEF123"

(SUBSTR TEXTO 4) -----> "DEF"

(SUBSTR TEXTO 2 2) -----> "BC"

(STRLEN TEXTO) -----> 6

(ITOA 45) -----> "45"

(ATOI "114") -----> 114

(ATOF "8.12") -----> 8.12

Función command

La función COMMAND permite invocar a un comando de Autocad dentro de una instrucción de AutoLISP. Está función nos permite hacer cualquier cosa que se permita desde el prompt '**Command**:' de Autocad. La sintaxis de esta función debe ser acorde a lo siguiente :

Dar comandos, opciones y textos entre comillas.

No usar funciones de entrada GET.

No utilizar variables precedidas por la exclamación '!'.

Llamar sólo comandos básicos de Autocad, no se permite comandos del usuario hechos en AutoLISP hechos con DEFUN.

La función COMMAND tiene la siguiente sintaxis :

```
( COMMAND  arg1  arg2  ... )
```

AutoLISP evalúa primero todos los argumentos (variables o expresiones), manda enseguida cada uno de los argumentos a Autocad. Las expresiones entre comillas son tomadas literalmente por Autocad. Las palabras que no estén entre comillas se tomaran como variables de AutoLISP. Expresiones precedidas por un apóstrofe no serán evaluadas.

Las constantes numéricas pueden o no ir entre comillas. Ejemplos

```
( COMMAND  "line"  "1,1"  "5,1"  "5,4"  "c" )

( COMMAND  "line"  '(1 1)  "5,1"  (LIST  5  4)  "c" )

( COMMAND  "line"  (LIST  1  1)  '(5  1)  (QUOTE (5  4))  "c" )
```

En AutoLISP, para indicar un <RETURN> se utilizará la doble comilla :

```
( COMMAND  "line"  "1,1"  "5,1"  "" )
```

La función COMMAND siempre regresa como valor "nil".

Definición de funciones de usuario

Autolisp como ya se ha visto realiza todo el trabajo a través del uso de un sin número de funciones o procedimientos. AutoLisp reconoce como funciones dos tipos:

. Funciones de Autolisp o intrínsecas

. Funciones del usuario

La posibilidad de crear nuevas funciones hace de Autolisp una excelente herramienta de desarrollo porque permite la conjugación de utilerías de Autolisp con el ingenio y la creatividad del usuario. La forma de definir una función del usuario es, por supuesto, con otra función: DEFUN

```
( DEFUN   cubo ( x )( * x x x ) )
```

En el anterior ejemplo se define la función del usuario 'cubo', que cuenta con un sólo argumento 'x'. La función propiamente dicha está definida por otra función : (* x x x)

Observemos otro ejemplo :

```
( DEFUN   sumcub ( x y / A B )
    ( SETQ   A   ( * x x x) )
    ( SETQ   B   ( * y y y ))
    ( +   A   B )
)
```

Se definió una función que presenta la forma más generalizada, aquí se observa :

Los argumentos, 'x' 'y'

Variables locales, 'A' , 'B'

La función DEFUN tiene como complemento:

Nombre de la función

Argumentos de la función

Definición de las variables locales.

Nombre de la función, es sencillamente un conjunto de caracteres alfanuméricos que identifican a toda la función al momento de invocarla desde el editor de Autocad:

Command: (sumcub 4 5)

Argumentos de la función, es la identificación de cuantos valores se necesitarán para valuar la función. Los argumentos podrán tener la siguiente presentación:

En la definición de la función sólo se marcan los nombres de las variables que asumirán los valores para valuar la función.

En la llamada de la función se marcan los valores (enteros, reales, cadena, listas, etc.) que pasarán representados por las variables referidas en la definición de la función.

Definición: (DEFUN sumcub (X Y))

Llamada: (sumcub 4 5))

En este caso 'X' asumirá temporalmente el valor de 4 y 'Y' el de 5, después de valuar la función 'X' y 'Y' esperarán asumir otros valores al momento de llamar nuevamente la función.

La definición de las variables locales tiene la idea de optimizar el espacio de Autolisp dedicado a guardar variables, esto es, toda variable definida por alguna función interna o del usuario pasa a ocupar ciertas localidades de memoria principal, si algunas de estas variables no es importante mantenerla con su actual valor, podemos anularla de memoria mandando precisamente como local su definición.

Las variables en Autolisp son:

Globales o generales, es decir, que se pueden referenciar con un mismo nombre desde cualquier función. Son todas las variables de Autolisp.

Locales o sea que son sólo utilizadas en una determinada función que así lo defina. Al finalizar la valuación de tal función, dichas variables desaparecerán de la memoria. Ejemplo :

(DEFUN sumcub (X Y / A B))

En esta definición la función 'sumcub' utiliza los siguientes tipos de variables:

Globales : 'X' y 'Y'

Locales : 'A' y 'B'

Expresiones de Control

Las expresiones de control son aquellas funciones que permiten realizar tareas de decisión y repetición condicionada, recuerde las figuras lógicas de la programación estructurada (IF THEN ELSE y DOWHILE). En una decisión se presenta siempre un valor lógico (cierto o falso, True o nil) y es el que marca que camino seguir para hallar la solución.

Funciones Lógicas

En AutoLisp cuando algo no tiene valor, representa 'nil'; si tiene un valor, representa 'T' (true, no-nil). A estos dos valores 'nil' y 'T' se les conoce como valores lógicos de una expresión. Como todo en AutoLisp se valúa, cualquier expresión tiene un valor lógico.

Existen las funciones lógicas, las cuales trabajan con los valores lógicos de las expresiones y regresan también un valor lógico:

(AND arg1 arg2 ...)

Regresa T si el valor lógico de todos los argumentos es true (T); si no, regresa nil.

(OR arg1 arg2 ...)

Regresa T si el valor lógico de uno de los argumentos es true (T); si no, regresa nil.

(NOT arg)

Regresa T si el valor lógico del argumento es nil; si no, regresa nil.

Suponiendo que V1 y V3 tienen valor y V2 no lo tiene:

(AND V1 V3) Regresa T

(AND V1 V2) Regresa nil

(AND V2) Regresa nil

(AND V1 (GETINT "Dame N")) depende de la respuesta al GET

(OR V1 V2 V3) Regresa T

(OR V2) Regresa nil

(NOT V2) Regresa T

(NOT (OR V1 V2)) Regresa nil

Funciones de relación

Las funciones de relación evalúan la relación entre dos expresiones. Si la relación es verdadera regresa 'T', si es falsa regresa 'nil'.

A continuación se listan las funciones de relación, observe el operador para cada caso:

(< arg1 arg2 ...)

Regresa T si cada argumento es menor que el siguiente; si no, regresa nil.

(> arg1 arg2 ...)

Regresa T si cada argumento es mayor que el siguiente; si no, regresa nil.

(< = arg1 arg2 ...)

Regresa T si cada argumento es menor o igual que el siguiente; si no, regresa nil.

(> = arg1 arg2 ...)

Regresa T si el argumento es mayor o igual que el sig; si no, regresa nil.

(/ = arg1 arg2)

Regresa T si los argumentos son diferentes; si son iguales, regresa nil.

Las anteriores funciones sólo pueden llevar argumentos numéricos o strings.

(EQ arg1 arg2)

Regresa T si los argumentos son idénticos.

(EQUAL arg1 arg2)

Regresa T si los argumentos son iguales.

La función 'EQ' se utiliza para comparar variables de listas y detecta si les fue asignado el mismo objeto. 'EQ' es equivalente a las funciones ' = ' y 'EQUAL' para comparaciones numéricas y de strings.

Suponiendo que X vale 10.0, Y vale 15.0, Z vale 7.0, A vale '(1 1 2) y B vale '(1 1 2):

(< X Z) Regresa nil

(< Z X Y) Regresa T

(> 25 Z) Regresa T

(< 15.0 Y) Regresa T

(> 10 X Y) Regresa nil

(= 7 Z) Regresa T

(EQUAL 7 Z) Regresa T

(EQ A B) Regresa nil

(EQUAL A B) Regresa T

(/= X 20.0) Regresa T

Función IF

Con esta función se puede condicionar el flujo de un programa. A la estructura de la función IF también se le reconoce como IF-THEN-ELSE, recuerde la programación estructurada. La función necesita evaluar una condición; si (IF) esta es T, ejecuta una primera expresión; si no (else), ejecuta una segunda expresión (es decir, sí la condición es nil).

La sintaxis de la función IF es la siguiente :

(IF condición expresión1 expresión2)

La condición puede ser cualquier expresión que valúe un valor lógico (T o nil), pero es recomendable utilizar siempre una función lógica o de relación. La expresión2 es opcional. Si la condición es T se valúa la expresión1; si no, valúa la expresión2 (si existe; si no, valúa nil).

La función regresa el valor de la última expresión evaluada.

Ejemplo:

Suponiendo que la variable 'a', tiene valor de 10.0:

```
( IF   a   "tengo valor"   "no tengo valor" )
```

La función 'IF' valúa la condición, la cual da T debido a que "a" tiene valor, por lo tanto la primera expresión es evaluada y como es una cadena, se valúa a sí misma; la función regresa la cadena "tengo valor". Si la variable b no tiene valor:

```
( IF   b   "tengo valor"   "no tengo valor" )
```

regresa "no tengo valor"

Si: x = 10 y y = 20

```
( IF  ( >  x  y )  ( setq  menor x ) ( setq  menor y ) )
```

en este caso, la condición es una función de relación que valúa T (verdadero), puesto que el contenido de 'x' (10) es menor que el de 'y' (20); por lo tanto se evalúa la primera expresión que asigna a la variable 'menor' el contenido de 'x' (10). La función regresa el valor 10. La siguiente expresión es semejante a la anterior, aunque no funcionan igual. ¿Cuál es la diferencia?:

```
( SETQ   menor ( IF  ( <  x  y )  x  y ) )
```

Función PROGN

La función 'IF' tiene la limitación de evaluar sólo una expresión sí cumpla o no la condición. En ocasiones es necesario evaluar varias expresiones si la condición resulta 'T' o 'nil'.

AutoLISP proporciona la función 'PROGN' para resolver este problema.

La función 'PROGN' permite incluir varias expresiones donde una expresión es requerida. AutoLISP trata al grupo de expresiones como una sola:

```
( PROGN   ( expresión1 )  ( expresión2 )  ( expresión 3 ) ... )
```

Se usa generalmente para evaluar múltiples expresiones en un IF. La función 'PROGN' regresa siempre el último valor evaluado en la expresión. Ejemplo :

```
( IF  ( =  ( GETVAR  "GRIDMODE" )  0 )
   ( PROGN
      ( SETVAR  "GRIDMODE"  1 )
      ( REDRAW )
```

```
            "Malla on"
        );
        "ya está prendida la malla"
    ) ; ENDIF
```

Función COND

Esta función trabaja en forma muy parecida a la función IF. Los argumentos de esta función consisten de una ó mas expresiones, cada una de ellas incluye una expresión condicional seguido de una ó mas expresiones a ser evaluadas si la condición regresa T. La función COND evalúa cada una de las expresiones condicionales hasta que evalúe a alguna con T; entonces evalúa las expresiones asociadas a esa condición e ignora todas las restantes:

```
( cond
    ( condición1   expresión   expresión ... )
    ( condición2   expresión   expresión ... )
    .
    .
    .
```

La función COND puede sustituir a la función IF, como lo muestra el siguiente ejemplo:

```
( COND
    (( < x y) ( SETQ  menor  x ))
    (( > x y) ( SETQ  menor  y ))
    (T  ( SETQ  menor  x )  "valores iguales" )
) ; END
```

Función REPEAT

Esta función permite evaluar un conjunto de expresiones repetidamente un número determinado de veces. A esta evaluación repetitiva de un conjunto de funciones se le llama iteración fija o "FOR". Su sintaxis es:

```
( REPEAT   n   ( expresión1 )  ( expresión2) ...
```

expresión1, expresión2... serán evaluadas una vez en cada iteración.

La función REPEAT regresa el valor de la última expresión en el proceso de iteración. Ejemplo:

```
( SETQ  x1  1  y1  1  x2  1  y2  6 )

( REPEAT  10

   ( COMMAND  "line"  ( LIST  x1  y1 )  ( LIST  x2  y2 )  "" )

   ( SETQ  x1  (+ x1  0.5 ) )

   ( SETQ  x2  (+ x2  0.5 ) )

 ) ; END
```

El ejemplo anterior dibuja 10 líneas paralelas con una separación de 6.5 unidades en sentido x.

Función WHILE

Esta función al igual que REPEAT evalúa varias expresiones repetidamente, pero a diferencia del REPEAT en el que se especifica el número de iteraciones, en la función WHILE el número de iteraciones está en función de una determinada condición. Su sintaxis es:

```
( WHILE  condición  ( expresion1 )  ( expresión2 ) ... )
```

Condición debe ser una expresión condicional,expresión1, expresión2... serán evaluadas repetidamente, evaluándose la condición antes de cada iteración tantas veces mientras la condición regrese el valor T .

Usualmente la expresión condicional contiene variables, cuyo valor puede cambiar en el curso de una iteración. A continuación se tiene un ejemplo similar al utilizado para el caso de la función REPEAT solo que ahora usando la función WHILE:

```
( SETQ  x1  1  y1  1  x2  1  y2  6 )
```

```
(SETQ  cont  1)
(WHILE (<=  cont  10)
  (COMMAND  "line"  (list  x1  y1)  (list  x2  y2)  "")
  (SETQ  x1  (+  x1  0.5)  x2  (+ x2  0.5))
  (SETQ  cont  (1+  cont))
);END
```

Con su editor cree la siguiente función y pruébela en Autocad:

```
(DEFUN  C:OPERA  (/ C1  D1  D2  r)
  (SETQ  d1  (GETREAL  "\n Dame el primer dato :"))
  (SETQ  d2  (GETREAL  "Dame el segundo dato :"))
  (WHILE  (AND (/= c1 1) (/=  c1  2) (/= c1  3) (/= c1  4))
     (SETQ  (1 (GETINT  "Deme la clave de operación: "))
     (COND
        ((=  C1  1)  (SETQ  r  (+  d1  d2)))
        ((=  C1  2)  (SETQ  r  (-  d1  d2)))
        ((=  C1  3)  (SETQ  r  (*  d1  d2)))
        (T  (PRINC  "\nNúmero de clave inválido!"))
     );END
  );END
  (PROGN  (PRINC  "El resultado es:")  (PRINC  r)  (PRINC))
);END OPERA
```

MANIPULACION DE ENTIDADES

Cada entidad de Autocad como una línea, arco ó circulo tiene un nombre que es reconocido por AutoCad, este nombre cambia cada vez que se elabora un dibujo, por lo que no sería adecuado intentar recordar cada uno de ellos.

AutoLISP usa funciones de entidades para preguntar por el nombre de entidades de la base de datos del dibujo.

Realice el sig. ejercicio:

Cree el Layer EJEMPLO1 y ubíquese en él para trabajar

LAYER/MAKE

Dibuje una line del punto (1 ,1) al punto (10, 1)

LINE 1,1 10,1

Invoque las siguientes funciones de AutoLISP:

(SETQ enti1 (ENTLAST))

(SETQ datos1 (ENTGET ent1))

En la primera instrucción, la función ENTLAST regresa el nombre de la última entidad en la base de datos en la forma:

< Entity name: 60000064 >

el nombre de la entidad se le asigna a la variable enti1.

En la segunda instrucción, la función ENTGET regresa una lista con los datos que describen a la entidad que se especifica, en éste caso la entidad enti1. Obsérvese que esta función requiere como argumento un nombre de entidad. La lista que regresa se le asigna a la variable datos1:

((-1. < Entity name: 60000064 >) (0. "LINE") (8."EJEMPLO1")(10 1.000000 1.00000) (11-10.000000 1.000000))

AutoLISP permite manipular entidades haciendo referencia a estas con el nombre de entidad (Entity name).

Los datos asociados a una entidad se regresan en una lista con código DXF que indica el tipo de dato que está contenido en la sub-lista. Cada sub-lista tiene dos partes,

la primera es el código DXF y la segunda es el valor del dato. El número 0 es el código para el tipo de entidad, que en el ejemplo anterior indica que es "LINE". El número 8 es el código para layer, y es indicado en el ejemplo como "EJEMPLO1". Los códigos 10 y 11 indican el punto inicial y punto final respectivamente.

Los códigos tienen diferentes significados con diferentes entidades. Se puede manipular todas las partes de cualquier dato de la entidad de un dibujo, lo que hace que AutoLISP sea una poderosa herramienta que permite acceso directo a la base de datos del dibujo de AutoCAD.

Funcion ASSOC

Esta función permite accesar parte de una lista de datos de entidad por asociación, a través de el código DXF. Por ejemplo, si se quiere obtener la sub-lista que tiene un código DXF de 10, AutoLISP busca en la lista de datos de la entidad una sublista con código 10 y regresa ésta:

Del el ejemplo anterior:

(ASSOC 8 datos1) regresa la sub-lista: (8. "EJEMPLO1")

(CDR (ASSOC 8 datos1)) extrae el dato del grupo layer, regresa:"EJEMPLO1"

Función TBLSEARCH

Autocad guarda elementos como blocks, layers, styles, linetypes, viewports y vistas en tablas. La función TBLSEARCH busca a través de una tabla y regresa información de ésta. Requiere dos argumentos, la tabla a buscar y el nombre de elementos de la tabla:

(TBLSEARCH "LAYER" "EJEMPLO1")

regresa: ((0. "LAYER") (2. "EJEMPLO1") (70.0) (62.7) (6. "Continuous"))

(TBLSEARCH "Style" "standard")

regresa :((0. "STYLE") (2. "STADARD") (70.0) (40.0.000000) (41.1.000000) (50.0.000000) (71.0) (42.0.200000)

(3."TXT") (4." "))

MANEJO DE LA BASE DE DATOS

El uso de las características de cada uno de los objetos generados por Autocad se efectúa utilizando las siguientes funciones de Autolisp:

a) Selección de objetos

(SSGET)

Define una selección de objetos

(SSGET "W" pto1 pto2)

Defina selección estilizando una ventana de extremos opuesto pto1 y pto2

(SSGET "P")

Define una selección previa

(SSGET "L")

Define una selección utilizando el último elemento incorporado a la base de datos.

(SSGET '(X Y))

Define una selección, marcando un punto referido de un elemento.

(SSGET "C" pto1 pto2)

Define una selección utilizando una ventana cruzada.

(SSGET "X") Define como selección todos los elementos de la base de datos.

((SSGET "X" (Filtro))

Define como selección todos los elementos de la base que cumplen con una condición (filtro)- listas asociativas. Ejemplo :

(SSGET "X" (LIST (CONS 8 "0")))

(SSGET "X" (LIST (CONS 8 "MUEBLES")))

b) Manipulación de objetos

(SSLENGTH)

Número de objetos en una selección SSGET.

(ENTSEL)

Proporciona nombre de un sólo objeto proporcionando interactivamente.

(ENTLAST)

Proporciona nombre del último objeto incorporado.

(SSNAME)

Proporciona los objetos y de una selección atrevés de un indica que indica el orden dentro del SSGET.

(SSADD)

Selección vacía, asigna valor vacío a una selección.

(ENTNEXT)

Proporciona la referencia al primer objeto incorporado a la base de datos.

(ENTGET)

Proporciona la información de un objeto através de su nombre

TABLA DE CODIGOS PARA LA BASE DATOS AUTOCAD

0

Tipo de entidad

2

nombre del bloque

6

tipo de línea

7

estilo de texto

8

nombre del LAYER

FUNCIONES PREDEFINIDAS DE AUTOLISP

FUNCIONES ARITMETICAS

```
(+  <num>  <num>  ... )
(-  <num>  <num>  ... )
(*  <num>  <num>  ... )
(/  <num>  <num>  ... )
(1+  <num> )
(1-  <num> )
(abs  <num> )
(atan  <num1>  [<num2>] )
(cos  <ángulo> )
(exp  <num> )
(expt  <base>  <potencia> )
(gcd  <num1>  <num2> )
(log  <num> )
(max  <num>  <num>  ... )
(min  <num>  <num>  ... )
(minusp  <num> )
pi
(rem  <num1>  <num2>  ... )
(sin  <ángulo> )
(sqrt  <num> )
(zerop  <elem> )
```

FUNCIONES PARA EL MANEJO DE CADENAS

```
(strcase  <cadena>  [<cómo>] )
(strcat  <cadena1>  <cadena2>  ... )
(strlen  <cadena> )
(substr  <cadena>  <inicio>  [<longitud>] )
```

FUNCIONES PARA EL MANEJO DE LISTAS

```
(append  <lista>  ... )
(assoc  <elem>  <lista> )
(car  <lista> )
(cdr  <lista> )
(caar,cadr,cddr,cadar, etc. )
(cons  <nuevo elemento>  <lista> )
(last  <lista> )
(length  <lista> )
(list  <expr>  ... )
(listp  <elem> )
(member  <expr>  <lista> )
(nth  <n>  <lista> )
(reverse  <lista> )
(subst  <nuevoelem>  <viejoelem>  <lista> )
```

FUNCIONES PARA ENTRADA DE USUARIO

```
(getangle  [<pt>]  [<mensaje>] )
(getcorner  <pt>  [<mensaje>] )
(getdist  [<pt>]  [<mensaje>] )
(getint  [<mensaje>] )
(getkword [<mensaje>] )
(getorient  [<pt>]  [<mensaje>] )
(getpoint  [<pt>]  [<mensaje>] )
(getreal  [<mensaje>] )
(getstring  [<cod-blancos>]  [<mensaje>] )
```

FUNCIONES DE RELACION

```
(=  <átomo>  <átomo>  ... )
(/=  <átomo1>  <átomo2> )
(<  <átomo>  <átomo>  ... )
(<=  <átomo>  <átomo>  ... )
(>  <átomo>  <átomo>  ... )
(>=  <átomo>  <átomo>  ... )
(eq  <expr1>  <expr2>)
(equal  <expr1>  <expr2>  [<to1>])
```

FUNCIONES LOGICAS

```
(and  <expresión> ... )
(not  <expresión> )
(null <expresión> )
(or  <expr>  ... )
```

Para manejo de BITS:

```
(~  <num> )
(logand  <num>  <num>  ... )
(logior  <num>  <num>  ... )
(lsh  <num1>  <numbits> )
```

FUNCIONES PARA CALCULO DE DATOS

```
(angle  <pt1>  <pt2> )
(distance  <pt1>  <pt2> )
(inters  <pt1>  <pt2>  <pt3>  <pt4>  [<prolong>] )
(osnap  <pt>  <cadena-modo> )
(polar  <pt>  <ángulo>  <distancia> )
(trans  <pt>  <sis-origen>  <sis-destino>  [<desplaz>] )
```

FUNCIONES PARA CONVERSION DE TIPOS DE DATOS

```
(angtos  <ángulo>  [<modo>  [<precisión>]] )
(ascii  <cadena> )
(atof  <cadena> )
(atoi  <cadena> )
(chr  <num> )
(fix  <num> )
(float  <num> )
(itoa  <entero> )
(rtos  <num>  [<modo>  [<precisión>]] )
(type  <elem> )
```

FUNCIONES PARA ARCHIVOS

```
(close  <apunt-arch> )
(findfile  <nombre-arch> )
(load  <nombre-arch>  [<onerror>] )
(open  <nombre-arch>  <modo> )
(prin1  <expr>  [<apunt-arch>] )
(princ  <expr>  [<apunt-arch>] )
(print  <expr>  [<apunt-arch>] )
(read-char  [<apunt-arch>] )
(read-line  [<apunt-arch>] )
(write-char  <num>  [<apunt-arch>] )
(write-line  <cadena>  [<apunt-arch>] )
```

FUNCIONES DE CONTROL DE FLUJO

```
(cond  (<expr-cond1>  <expresión1>  ...)  ... )
(if  <expr-cond>  <expr1>  [<expr2>] )
(progn  <expr>  ... )
(repeat  <number>  <expr>  ... )
(while  <expr-cond>  <expresión>  ... )
```

OTRAS FUNCIONES

```
(apply <función> <lista> )
(atom <elem> )
(boole <func> <entero1> <entero2> ... )
(boundp <átomo> )
(command <args> ... )
(defun <sim> <lista argumentos> <expr> ... )
(eval <expr>)
(foreach <nom> <lista> <expr> ... )
(getenv <nom-var-amb> )
(getvar <nom-var-acad> )
(graphscr)
(initget [<bits>] <cadena> )
(lambda <args> <expr> ... )
(mapcar <función> <lista1> ... <listan> )
(menucmd <cadena> )
(numberp <elem> )
(prompt <mensaje> )
(quote <expr> )
(read <cadena> )
(redraw [<ename> [<modo>]] ))
(set <sim> <expr> )
(setq <sim1> <expr1> [<sim2> <expr2>] ... )
(setvar <nom_var-acad> <valor> )
(terpri)
(textscr)
(trace <función> ... )
(untrace <función> ... )
(ver)
(vports)
(*error* <cadena> )
```

FUNCIONES PARA MANIPULACION DE LA BASE DE DATOS DE AUTOCAD

A) Manipulación de Selecciones (sel)

```
(ssget  [<modo>]  [<pt1>  [<pt2>]] )
                  <modos>: "W", "C", "L", "P"
                           "X" <lista-filtro>
                               <lista-filtro>: lista asociativa
(sslength  <sel> )
(ssname  <sel> <índice> )
(ssadd  [<nombre-ent>  [<sel>]] )
(ssdel  <nombre-ent>  <sel> )
(ssmemb  <nombre-ent>  <sel> )
```

B) Manipulación de Nombres de Entidades (nombre-ent)

```
(entnext  [<nombre-ent>] )
(entlast)
(entsel  [<mensaje>] )
(handent  <cadena> )
```

C) Manipulación de los datos de una entidad

```
(entdel  <nombre-ent> )
(entget  <nombre-ent> )
(entmod  <lista-ent> )
(entupd  <nombre-ent> )
```

D) Manipulación de Tablas

```
(tblnext  <nombre-tabla>  [<rewind>] )
(tblsearch  <nombre-tabla>  <sim>  [<sigue>])
```