AutoLISP

NGÔN NGỮ LẬP TRÌNH
AutoLISP trong AutoCAD

Mở đầu

- AutoLISP là ngôn ngữ sử dụng để lập trình tự động tạo lập các đối tượng trong bản vẽ AutoCAD.
- Các đối tượng này có thể được tạo ra qua 2 cách:
  - Sử dụng hàm (Command LệnhCAD Thôngsố...)
  - Truy cập trực tiếp CSDL của AutoCAD
- Các nội dung cần nắm trước:
  - Sử dụng AutoCAD cơ bản
  - Có kiến thức cơ bản về lập trình
  - Có hiểu biết ít nhất 1 ngôn ngữ lập trình

1. Khái niệm chung

- Biểu thức AutoLISP

  (setq a 10.0 b "hello!")

  (setq c (+ 100 a))
- Thực hiện biểu thức
  - Gõ trực tiếp biểu thức AutoLISP vào dòng lệnh Command
  - Lưu các biểu thức thành file, ví dụ test1.lsp, sau đó gọi file để thực hiện các biểu thức trong file đó bằng cách gõ (load tes1.lsp) trên dòng lệnh Command
  - Thông qua VisualLISP IDE - gõ VLIDE hoặc VLISP trên dòng lệnh Command hoặc qua thực đơn Tools/AutoLISP/VLisp Editor

Khái niệm chung (2)

- VisualLISP
  - Khởi động: gõ VLIDE hoặc VLISP trên dòng lệnh Command hoặc qua thực đơn Tools/AutoLISP/VLisp Editor
  - Quản lý đề án: trong thực đơn Project
  - Quản lý file: trong thực đơn File
  - Tải và chạy chương trình AutoLISP: Tools/LoadTextInEditor
  - Các tính năng khác:
    - Tìm kiếm và thay thế: Search/Find hoặc Search/Replace
    - Mã màu
    - Định dạng mã nguồn: Tools/EnvironmentOptions/VLFormatOptions
    - Dò lỗi: Tools/CheckTextInEditor

2. Các thành phần cơ bản

- Danh sách và biểu thức
- Các lời chú thích
- Biến, hằng và các kiểu dữ liệu
- Cấu trúc chương trình AutoLISP
- Các hàm cơ sở trong AutoLISP
- ...

2.1. Danh sách và biểu thức

- AutoLISP được xây dựng trên cơ sở các danh sách
  - Danh sách (list) gồm một hoặc nhiều phần tử hoặc rỗng được đặt trong 1 cặp ngoặc đơn. Các phần tử trong danh sách cách nhau bởi ít nhất 1 dấu trắng.
  - Các đối tượng (entity) trong AutoCAD cũng được quản lý như các danh sách
  - Danh sách dot-pair gồm 2 phần tử, liên kết với nhau bằng dấu chấm (dot)
- Biểu thức (câu lệnh) cũng được bao trong cặp dấu ngoặc đơn giống như danh sách, nhưng phần tử đầu của biểu thức phải là 1 hàm.
  - Trong biểu thức có thể có các biểu thức con, ví dụ:
    (setq val1 (- (+ 10 5 2.0))

2.2. Các lời chú thích

- Dấu **;** để ghi chú thích trên 1 dòng
  - (setq a 10) *; gán giá trị 10 cho biến a (chú thích)*
- Chú thích đoạn (inline) được đặt trong cặp "**;|**" và "**|;**"
  - (setq *;| gán giá trị cho biến |;* a 10)
  - *;| Đây là một đoạn chú thích dài, trên một hoặc nhiều dòng, thường dùng để giải thích công dụng của đoạn chương trình.*

    Đoạn chương trình sau thể hiện một hàm tự định nghĩa, dùng để đổi độ ra radian |;

```
(defun do2rad (do / ppi)  ; khai báo hàm
    (setq ppi 3.14159)
    (* do (/ ppi 180.0))  ; giá trị trả về của hàm
) ; kết thúc hàm
```


2.3. Biến và hằng

- Tên biến không phân biệt chữ hoa, thường. Một tên biến có thể được gán dữ liệu các kiểu khác nhau. Kiểu dữ liệu của biến lấy theo dữ liệu được gán, không cần khai báo trước.
- Hàm *(setq tenbien giatri tenbien giatri…)* dùng để gán giá trị cho biến.
  - (setq **a** 10 **b** "Hi" **B1** (+ 5.0 2)) *; gán giá trị 10 cho biến **a**, "Hi" cho biến **b**, 7.0 cho biến **B1***
- Trong hàm AutoLISP sử dụng 2 loại biến: biến tham số cần truyền qua hàm và biến cục bộ.
  - Biến tham số cần khai báo ngay đầu hàm, các biến cục bộ khai báo sau dấu gạch chéo "/". Ví dụ:

```
(defun do2rad (do / ppi)  ; do la biến cần truyền DL,
                          ppi là biến cục bộ
```

Biến và hằng (2)

- Các biến cục bộ nếu không được khai báo có thể làm thay đổi giá trị các biến ngoài hàm (tức là chúng được coi như các biến toàn cục)
- AutoLISP dùng một số ký hiệu dành riêng cho hằng, cần tránh đặt tên biến trùng với các hằng số này.
- Các hằng số:
  - PI – hằng số pi = 3.14159
  - NIL – hằng số logic false
  - T – hằng số logic true (non-NIL)
  - PAUSE – sử dụng với hàm (Command) nhằm tạm dừng chương trình, chẳng hạn để người dùng nhập dữ liệu.

2.4. Các kiểu dữ liệu

- Các kiểu thông dụng: Integer, Real, String như các ngôn ngữ lập trình khác.
- Kiểu Integer cho phép gán số nguyên 32bit, nhưng hàm nhập liệu từ bàn phím GetInt lại chỉ chấp nhận số nguyên 16bit.
- Cũng cần lưu ý kiểu kết quả trả về khi sử dụng các hàm cơ sở.

```
(setq  n (/ 10 4))  ; gán n giá trị phép chia nguyên 10:4 => n = 2
(setq  r (/ 10 4.0))  ; gán r giá trị phép chia 10:4 => n = 2.5
(setq  max 2147483647)  ; gán max giá trị nguyên lớn nhất (32b)
(setq  max 2147483648)  ; max có giá trị  2.14748e+009 (số thực)
(setq  max (+ 2147483647 3))  ; max nhận giá trị  (- 2147483647)
                              sai do vượt quá giới hạn 32 b
```

Các kiểu dữ liệu (2)

- Kiểu danh sách LIST
  - Danh sách list là kiểu dữ liệu đặc thù trong AutoLISP. Các phần tử có thể từ các kiểu dữ liệu khác nhau.
  - Xác định danh sách qua hàm quote (dấu nháy đơn) hoặc hàm list:

```
'(1  2  "abc" (0 . "CIRCLE"))  ; danh sách 4 phần tử
(quote (1  2  "abc"  (0 . "CIRCLE")))  ; tạo từ hàm quote
(list  1  (+ 1 1)  "abc" (cons 0 "CIRCLE")))  ; tạo từ hàm list
```

Lưu ý:

- Hàm list trả về danh sách từ giá trị các biểu thức trong hàm.
- Hàm quote trả về nguyên dạng các phần tử khai báo, không xử lý các biểu thức.
- Hàm (cons x y) tạo danh sách liên kết (dot-pair) từ 2 phần tử x,y

Các kiểu dữ liệu (3)

- Kiểu EName và PickSet (SelectionSet)
  - Hai kiểu dữ liệu đặc thù để quản lý các đối tượng trong bản vẽ AutoCAD.
  - Ename trả về tên đối tượng trong bản vẽ AutoCAD, qua đó truy cập đến CSDL của chúng để xử lý chúng (sửa, xóa, cập nhật…)
  - PickSet quản lý tập hợp các đối tượng được chọn, tương ứng với cách chọn Seclect Objects trong AutoCAD. Có thể thêm vào hoặc loại bớt đối tượng khỏi nhóm đã chọn.

Các kiểu dữ liệu (4)

- File dữ liệu
  - File dữ liệu trong AutoLISP chỉ sử dụng kiểu file text, truy cập tuần tự, không có kiểu file truy cập ngẫu nhiên.
  - Dữ liệu trong file được ghi vào và đọc ra theo dòng, lần lượt từ dòng thứ nhất đến dòng cuối cùng.
  - Để sử dụng dữ liệu dạng bảng (nhiều dòng và cột), dữ liệu được chuyển thành dạng list, chẳng hạn:

Mỗi dòng thành một phần tử của danh sách
Mỗi ô trên dòng thành một phần tử con

```
( (11 12 13 14 15)        ; dòng 1, 5 phần tử
  (21 22 23 24 25))       ; dòng 2, 5 phần tử..
```


3. Cấu trúc chương trình AutoLISP

- Cấu trúc rất tự do:
  - không cần khai báo biến, kiểu dữ liệu; hàm con (hàm người dùng) cũng có thể đặt bất kỳ đâu trong chương trình…
  - Tuy nhiên cần rất chú ý để tránh nhầm lẫn.
- Một số cấu trúc thông dụng
  - Phép gán
  - Hàm người dùng defun
  - Vòng lặp
  - Rẽ nhánh…

3.1. Phép gán

- Hàm Setq dùng gán giá trị cho biến

  Cú pháp ***(setq var1 val1 var2 val2…)***
  - **setq** - từ khóa hàm gán số liệu
  - **var1, var2, …** - tên các biến
  - val1, val2, … - giá trị sẽ gán cho các biến tương ứng
- Ví dụ

```
(setq  a 10)              gán cho biến a giá trị 10
(setq  a 10  b '(5 8))    gán cho biến a giá trị 10, b danh sách (5 8)
(setq  a 10  b "Hello")   gán cho biến a giá trị 10, b chuỗi ký tự "Hello"
```


3.2. Hàm người dùng

- Hàm con được người dùng tự định nghĩa

  Cú pháp ***(defun fun-name([arg] [/ var])***
  - defun - từ khóa để khai báo hàm
  - fun-name - tên hàm do người dùng định nghĩa
  - arg - tên các tham số cần truyền qua hàm
  - var - tên các biến cục bộ
  - / - dấu phân cách (giữa thông số cần truyền và biến cục bộ)
- Gọi hàm đã định nghĩa

  Cú pháp ***(fun-name arg1 arg2 …)***
- Hàm "C:FUN-NAME" tạo lệnh mới cho AutoCAD
- AutoLISP cũng cho phép định nghĩa hàm đệ quy

Hàm người dùng (2)

- Ví dụ: tạo lệnh tính n!

```
(defun giaithua(n) ; khai báo n là biến cần truyền thông số
   (if (= n 0) 1   (* n (giaithua(- n 1))))
   )      ; kết thúc khai báo hàm. Chú ý các dấu ngoặc
; Tạo hàm như lệnh n! trong AutoCAD
(defun C:n! ( / c n) ; khai báo c, n như các biến cục bộ
   (initget 5)      ; hạn chế nhập sai dữ liệu (số âm, số không nguyên,…)
   (setq n (getint "Hãy nhập một số nguyên không âm: "))
   (setq c (giaithua n))
   (princ (strcat (itoa  n) "! = " (itoa c))) ; định dạng và in kết quả
   (princ)          ; rút lui im lặng
   )      ; kết thúc hàm
```


3.3. Cấu trúc rẽ nhánh

- Hai cấu trúc rẽ nhanh thông dụng IF (rẽ đôi) và COND (rẽ nhiều)
- Cấu trúc IF sử dụng hàm if như sau:

(**if** testexpr thenexpr [elseexpr])
  - testexpr - biểu thức được đánh giá, trả về NIL hoặc T
  - thenexpr - biểu thức thực hiện nếu testexpr trả về T
  - elseexpr - biểu thức thực hiện nếu testexpr trả về NIL

Cần lưu ý rằng thenexpr và elseexpr chỉ được phép là các biểu thức đơn. Nếu muốn thực hiện nhiều biểu thức, sử dụng hàm **progn** để gom các biểu thức này thành 1 khối. Khối **progn** có vai trò như khối begin…end trong Pascal hoặc {…} trong C.

Cấu trúc rẽ nhánh (2)

- Cấu trúc COND sử dụng cú pháp sau:

```
(cond
    (test1  result1)
    (test2  result2)
    ...
    (testn  resultn)
    (T  elsewise)
    )
```

  - test1, test2.. - biểu thức được đánh giá, trả về NIL hoặc T
  - result1, result2... - biểu thức thực hiện nếu test1,2... trả về T
  - elsewise - biểu thức thực hiện nếu tất cả test1,2... đều trả về NIL

Cấu trúc rẽ nhánh (3)

- Ví dụ:

```
(if  (= 1  3)
     (setq  ans  "Sometime") ; thỉnh thoảng 1 = 3 !!!
     (progn   ; dùng progn để gom 2 biểu thức dưới đây
          (alert  "Đây là chuyện thường: 1 # 3")
          (setq  ans  "Ok")
          ) ; kết thúc progn
     ) ; kết thúc if
```

Do biểu thức được đánh giá luôn là NIL nên kết quả của đoạn chương trình này sẽ hiện hộp thông báo "Đây là chuyện thường..." và biến ans được gán giá trị "Ok"

3.4. Vòng lặp

AutoLISP cung cấp các hàm sau đây để thực hiện vòng lặp:

- (repeat n [expr...]) - thực hiện n lần lặp các biểu thức expr...
- (while test [expr...]) - thực hiện các b.thức expr khi test khác NIL
- (foreach var lst [expr...]) - thực hiện các b.thức expr... với biến var nhận các giá trị lần lượt là các phần tử trong danh sách lst
- (mapcar 'func lst1 ... lstn) – thực hiện hàm func với đối số là các danh sách lst1...lstn. Kết quả trả về là một danh sách mới từ kết quả.

Thực chất thì mapcar không phải là vòng lặp chính thống.

3.4.1. Vòng lặp Repeat

```
; tính tổng 10 số tự nhiên đầu tiên
(setq tong 0)
(setq sohang 1)
(repeat 10       ; lặp 10 lần các biểu thức tiếp theo
   (setq  tong  (+  tong  sohang)
   (setq  sohang (1+  sohang))
) ; hết repeat
```

Kết quả: tong nhận giá trị 55; sohang nhận giá trị 11

3.4.2. Vòng lặp While

```
; tính tổng 10 số tự nhiên đầu tiên
(setq tong 0)
(setq sohang 1)
(while (<=  sohang  10) ; kiểm tra xem sohang có lớn hơn 10 không?
   (setq  tong  (+  tong  sohang)
   (setq  sohang (1+  sohang))
) ; hết while
```

Kết quả: tong nhận giá trị 55; sohang nhận giá trị 11

3.4.3. Vòng lặp Foreach

```
; tính tổng 10 số tự nhiên đầu tiên
; tạo danh sách gồm 10 số dương đầu tiên
(setq lst (list 1 2 3 4 5 6 7 8 9 10))
(setq tong 0)
(foreach sohang lst
   (setq  tong  (+  tong  sohang))
) ; hết  foreach
```

Kết quả: tong nhận giá trị 55; sohang nhận giá trị **10**

3.3.5. Sử dụng Mapcar

; Biến danh sách điểm 2D thành 3D

; Khai báo danh sách gồm 4 điểm 2D (chỉ có tọa độ X, Y)

(setq plst `((1 10) (2 20) (3 30) (4 40)))

(setq xlst (**mapcar** ***'car*** plst)) *; lấy thành phần đầu (tọa độ X) của các điểm, đưa kết quả vào danh sách xlst*

(setq ylst (**mapcar** ***'card*** plst)) *; lấy thành phần thứ 2 (tọa độ Y) của các điểm, đưa kết quả vào danh sách ylst*

(setq zlst `(100 200 300 400)) *; Khai báo tọa độ z*

; Tạo các điểm 3D từ danh sách các tọa độ xlst, ylst và zlst

(setq 3Dplst (**mapcar** ***'list*** xlst ylst zlst))

- **Kết quả trả về danh sách 3Dplst :**

  `((1 10 100) (2 20 200) (3 30 300) (4 40 400))

4. Các hàm cơ sở

- Các hàm số học (+ - * / ...)
- Các hàm lôgic (so sánh, or, and...)
- Các hàm xử lý chuỗi ký tự
- Các hàm xử lý danh sách
- ...

4.1. Các hàm số học

- Thực hiện phép tính trên số nguyên hoặc số thực. Kết quả trả về là dữ liệu dạng số.
- Cộng (**+** n1 n2 ...) *; kq: n1 + n2 + ...*
- Trừ (**-** n1 n2 ...) *; kq: n1 – n2 - ...*
- Đếm tăng (**1+** n) *; trả về giá trị n+1*
- Đếm giảm (**1-** n) *; trả về giá trị n-1*
- Nhân (***** n1 n2 ...) *; trả về n1*n2*...*
- Chia (**/** n1 n2 ...) *; trả về n1/n2/...*
- Trị tuyệt đối (**abs** n) *; trả về /n/*
- Lấy phần nguyên (**fix** n) *; trả về giá trị nguyên sau khi bỏ phần thập phân*

Các hàm số học (2)

- Giá trị nhỏ nhất (**min** n1 n2 ...)
- Giá trị lớn nhất (**max** n1 n2 ...)
- Hàm mũ (**exp** x) *; e^x*
  (**expt** a x) *; a^x*
  (**sqrt** x+) *; căn bậc 2*
- Hàm logarit tự nhiên (**log** x+) *; $\log_e(x)$*
- Lượng giác (**cos** rad), (**sin** rad) *– cos và sin của góc bằng rad*
  (**atan** n1 [n2]) *- hàm arctan (n1 / n2), trả về góc = rad*

Ví dụ:

(atan 1 0) => 1.5708

(atan -1 0) => - 1.5708

4.2. Các hàm logic

- So sánh (**=** ns1 ns2 ...) so sánh giá trị các biểu thức. Nếu tất cả bằng nhau hoặc chỉ có 1 biểu thức thì trả về T, ngược lại sẽ trả về NIL

  (/= ns1 ns2 ...) so sánh giá trị các biểu thức. Nếu chỉ có 1 tham số hoặc các tham số cạnh nhau không giống nhau, trả về T, ngược lại sẽ trả về NIL

  (**equal** expr1 expr2 fuzz) so sánh gần đúng giá trị các biểu thức, với fuzz là sai số. Nếu bỏ qua fuzz hàm thực hiện như so sánh "="

  (**<** ns1 ns2 ...) so sánh giá trị các tham số theo thứ tự tăng dần thì trả về T, ngược lại thì trả về NIL

  (**<=** ns1 ns2 ...) so sánh "bé hơn hoặc bằng"

  (**>** ns1 ns2 ...) so sánh "lớn hơn"

  (**>=** ns1 ns2 ...) so sánh "lớn hơn hoặc bằng"

Các hàm logic (2)

- Liên kết các biểu thức logic

  (**or** expr1 expr2 ...)

  (**and** ns1 ns2 ...)

  (**not** expr) hàm phủ định. Nếu **expr** là T thì tả về NIL và ngược lại.

  (**null** expr) kiểm tra xem một biểu thức có phải là NIL hay không

  (**numberp** item) kiểm tra xem **item** có phải là số (nguyên hoặc thực) hay không. Hàm trả về T nếu **item** là số, ngược lại (chữ hoặc list...), trả về NIL

4.3. Các hàm xử lý ký tự

- Sử dụng với số liệu dạng ký tự: nối, lấy một phần, đổi chữ hoa…

(**read** [str]) lấy thành phần đầu của chuỗi str. Các thành phần được ngăn cách bởi dấu trắng, dòng mới, tab hoặc ngoặc đơn. Giá trị trả về được chuyển đổi về kiểu dữ liệu thích hợp.

(read "abc") – trả về ký hiệu (symbol) ABC

(read "abc df 123) – cũng trả về ký hiệu ABC, nhưng

(read "\"aBc\" df 123") – trả về chuỗi ký tự "aBc"

Lưu ý: **(read "123 df")** – trả về số nguyên 123,
(read "(1 2 3) (d f)") – trả về danh sách (1 2 3)

(**strcase** str [mode]) chuyển chuỗi **str** về chữ IN nếu bỏ qua **mode** hoặc **mode** là NIL. Nếu **mode khác NIL** – chuyển **str** về chữ thường.

(**strcat** [str1] [str2…]) nối các chuỗi **str1**, **str2**… thành chuỗi chung.

(**strlen** [str]) trả về chiều dài (số lượng các kỹ tự) chuỗi ký tự trong chuỗi **str**

(**substr** str start [length]) trả về chuỗi ký tự con, lấy từ chuỗi **str**, bắt đầu từ vị trí **start** với số lượng ký tự bằng **length**. Nếu bỏ qua length chuỗi ký tự được lấy đến ký tự cuối cùng của **str**.

4.4. Các hàm xử lý danh sách

- **Khởi tạo danh sách**

(**quote** expr) trả về biểu thức mà không xử lý giá trị của nó. Hàm được dùng để tạo danh sách từ các phần tử là hằng số. Có thể sử dụng dấu nháy đơn thay cho hàm quote.

(setq pt1 (qoute (1 1 0))) – gán biến pt1 như 1 điểm có 3 tọa độ (1 1 0) hoặc

(setq pt1 '(1 1 0))

Để tạo danh sách rỗng có thể sử dụng cú pháp **(setq nlst '())**

(**list** [expr…]) xử lý các biểu thức **expr** và liên kết chúng thành danh sách

(setq n1 10) ; gán biến n1 giá trị 10

(setq n2 20) ; gán biến n2 giá trị 20

(setq lst (list n1 (+ n1 n2) 'a '(1 2))) ; gán biến lst là danh sách gồm các phần tử **10** (n1) **30** (n1+n2) **a** (hằng số, kiểu symbol) và danh sách con **(1 2)** – hằng số, tức là **lst** sẽ là danh sách **(10 30 a (1 2))**

Các hàm xử lý danh sách (2)

- **Khởi tạo danh sách** (tiếp)

(**cons** ell list-atom) thêm thành phần **ell** vào đầu danh sách hoặc tạo danh sách liên kết dot-pair. Giá trị trả về tùy thuộc kiểu của tham số thứ 2: nếu là danh sách thì hàm này sẽ trả về danh sách mới sau khi đã thêm phần tử **ell** vào đầu nó, còn nếu là giá trị, kết quả là danh sách dot-pair.

(cons 10 '(1 2 3)) – trả về danh sách gồm 4 phần tử **(10 1 2 3)**

(cons 10 (list 1 2 3)) – cho kết quả như trên

(cons 10 '()) – trả về danh sách gồm 1 phần tử **(10)**

(cons '(10 20) '(1 2 3)) – trả về danh sách gồm 4 phần tử **((10 20) 1 2 3)**

(cons 10 1) – trả về danh sách dot-pair **(10 . 1)**

(cons '(10 20) 'a) – trả về danh sách dot-pair **((10 20) . A)**

Các hàm xử lý danh sách (3)

- **Xử lý danh sách**

(**append** [lst…]) liên kết các danh sách **lst** và tạo danh sách mới. Nếu không có tham số hàm sẽ trả về NIL.

(append '(10 20) '(1 2 3)) – trả về danh sách gồm 5 phần tử **(10 20 1 2 3)**

(append '("a" "b") '(a b)) – trả về danh sách gồm 4 phần tử **("a" "b" A B)**

(append (list '("a" "b")) (list '(a b))) – trả về danh sách gồm 2 phần tử **(("a" "b") (A B))**

(**reverse** lst) trả về danh sách đảo ngược thứ tự các phần tử trong danh sách **lst**

(setq lst '(1 2 3 4)) ; gán cho biến lst danh sách 4 phần tử **(1 2 3 4)**

(setq newlst (reverse lst)) ;gán cho biến newlst danh sách **(4 3 2 1)**

(**length** lst) cho biết số phần tử trong danh sách **lst.**

(length '(1 2 3 4)) - trả về số nguyên 4

(length '((1 2) (3 4))) - trả về số nguyên 2, còn

(length '()) - trả về số nguyên 0 (danh sách rỗng)

Các hàm xử lý danh sách (4)

- **Xử lý danh sách** (tiếp)

(**subst** newitem olditem lst) tìm kiếm phần tử **olditem** trong **lst** và thay thế nó bằng phần tử **newitem.** Nếu tìm thấy sẽ trả về danh sách mới, nếu không sẽ trả về danh sách ban đầu.

(setq lst '(1 2 3))

(setq newlst (subst "Ab" 2 lst)) – gán cho biến **newlst** danh sách **(1 "Ab" 3)**

Để lấy phần tử trong danh sách có thể dùng các hàm sau:

(**last** lst) lấy phần tử cuối cùng trong danh sách **lst**

(last '("ab" 2 3 4)) - trả về số nguyên 4 (phần tử cuối trong danh sách)

(**car** lst) lấy phần tử đầu trong danh sách **lst**

(last '("ab" 2 3 4)) - trả về chuỗi ký tự "ab" (phần tử đầu trong danh sách)

(**nth** n lst) lấy phần tử thứ **n** trong danh sách **lst.**

Lưu ý thứ tự các phần tử đánh số từ 0, do đó (nth 0 lst) và (car lst) cho kết quả như nhau, còn (nth (1- (length lst)) lst) và (last lst) cũng vậy.

Các hàm xử lý danh sách (5)

- **Xử lý danh sách** (tiếp)

(**cdr** lst) trả về danh sách con từ danh sách **lst**, sau khi đã bỏ đi phần tử đầu.

(setq 3dPoint '(100 10 1))

(setq YZ-lst **(cdr 3dPoint))** sẽ gán cho biến **YZ-lst** danh sách **(10 1)**

Phần tử thứ 2 và 3 trong danh sách trên, có thể được lấy qua các biểu thức:

(setq Ycord (car (cdr 3dPoint)))

(setq Zcord (car (cdr (cdr 3dPoint))))

Để cho tiện, AutoLISP kết hợp các hàm trên như sau:

(caar lst) tương ứng với **(car (car lst))**	
(cadr lst)	**(car (cdr lst))**
(cddr lst)	**(cdr (cdr lst))**
(cadar lst)	**(car (cdr (car lst)))**
(cddar lst)	**(cdr (cdr car lst)))** **v.v…**

Các hàm xử lý danh sách (6)

- **Tìm kiếm trong danh sách và dot-pair**

Danh sách dot-pair được sử dụng rất nhiều trong CSDL của AutoCAD. Để truy cập đến danh sách loại này hoặc danh sách chứa các danh sách con, AutoLISP cung cấp hàm **assoc** sau:

(**assoc** item asslst)

- item khóa cần tìm kiếm, phải là thành phần đầu của list con
- asslst danh sách liên kết cần tìm kiếm

Nếu tìm thấy hàm trả về danh sách con hoặc dot-pair chứa khóa cần tìm, nếu không hàm sẽ trả về giá trị NIL.

Hàm assoc thường được dùng để truy cập CSDL AutoCAD nhằm tìm kiếm một loại đối tượng nào đó, thông qua mã đối tượng (mã nhóm GroupCode). Trong CSDL các code này được thể hiện qua danh sách dot-pair. Chẳng hạn mã 100 thể hiện vòng tròn, mã 62 – màu đối tượng, mã 8 – lớp chứa đối tượng, mã 10 - tọa độ điểm…

Các hàm xử lý danh sách (7)

- **Ví dụ dùng assoc để tìm kiếm danh sách con**

```
(setq  lst '(                ; khai báo danh sách liên kết, thực chất là vòng
       (410 . "Model")       ; tròn vẽ trong không gian mô hình của AutoCAD
       (8 . "L123")
       (62 . 1)
       (100 . "AcDbCircle")
       (10  10.0  20.0  0.0)
       (40  .  5.0)
       ))                    ; kết thúc khai báo lst
```

Các biểu thức sau đây sẽ cho ta các thông tin về đối tượng này:

```
(setq  space (assoc 410 lst)      ; trả về dot-pair (410 . "Model")
       center (assoc 10 lst)      ; trả về danh sách (10  10.0  20.0  0.0)
       layer (assoc 8 lst)        ; trả về dot-pair (8 . "L123")
       its8 (assoc "L123" lst)    ; trả về NIL (không tìm thấy)
       none (assoc "Some" lst)    ; trả về NIL (không tìm thấy)
       )                          ; kết thúc setq
```

5. Lập trình ứng dụng bằng AutoLISP

- Ngôn ngữ AutoLISP có thể thực hiện được hầu hết các chức năng như các ngôn ngữ lập trình khác, nhưng do cách viết rắc rối và chạy trong nền AutoCAD nên chỉ thường được sử dụng cho mục đích thiết kế tự động, nhất là việc xây dựng các bản vẽ thiết kế. Một số vấn đề cần chú ý:
- Tổ chức nhập – xuất dữ liệu trong AutoCAD
- Tạo các đối tượng AutoCAD
- Chỉnh sửa các đối tượng AutoCAD
- Hộp thoại DCL trong môi trường AutoCAD

5.1. Nhập dữ liệu

- Dữ liệu có thể được nhập – xuất trực tiếp qua đối thoại người – máy hoặc từ file đã chuẩn bị sẵn.
- Đối thoại được thực hiện qua dòng lệnh Command hoặc hộp thoại.
- Các thông báo cũng được đưa ra theo 2 cách: qua dòng lệnh Command hoặc qua hộp thoại.
- Dữ liệu cũng có thể được nhập từ file hoặc xuất ra file (dạng file văn bản, truy cập tuần tự)

5.1.1. Thông báo

- Các thông báo được đưa ra theo 2 cách: qua dòng lệnh Command hoặc qua hộp thoại trên màn hình AutoCAD.
- Ngoài các thông báo đi kèm với các nhập dữ liệu (các hàm getX) còn dùng các hàm sau:

  (promt msg) ; đưa thông báo ra dòng lệnh

  (alert msg) ; đưa thông báo ra hộp thoại AutoCAD Message

  Trong các hàm này msg là nội dung thông báo, kiểu chuỗi ký tự. Nếu muốn thể hiện trên nhiều dòng, cần chèn thêm dấu xuống dòng "\n".

Ví dụ (**prompt** "Chọn nhóm đối tượng thứ nhất. \nSau đó chọn nhóm 2…") sẽ đưa ra 2 dòng thông báo trên dòng lệnh Command của AutoCAD:

Chọn nhóm đối tượng thứ nhất.
Sau đó chọn nhóm 2…

5.1.2. Các hàm nhập liệu GetX

- Các hàm nhập dữ liệu trực tiếp trên dòng lệnh Command có cú pháp chung như sau:

  (getX [msg] [...]**)**

  - getX - tên hàm, X thường thể hiện kiểu dữ liệu. Ví dụ GetInt – nhập số nguyên, getString – nhập chuỗi ký tự…
  - msg - chuỗi ký tự thể hiện lời nhắc kèm theo, hiện trên dòng lệnh Command, nhằm nhắc người dùng nhập đúng dữ liệu yêu cầu
  - [...] - các thông số khác của hàm.
- Các hàm này (trừ getString) có thể được dùng kèm theo các hàm khống chế kiểu dữ liệu nhập (**getKword, InitGet**).
- Nếu thực hiện thành công các hàm này sẽ trả về giá trị đã nhập.

Hàm GetInt

- Dùng nhập số nguyên 16b (-32768 đến +32767) từ bàn phím

  (getInt [msg]**)**
  - msg - chuỗi ký tự thể hiện lời nhắc kèm theo, hiện trên dòng lệnh Command, nhằm nhắc người dùng nhập đúng dữ liệu yêu cầu
  - Nếu nhập đúng, kết quả trả về của hàm là số nguyên vừa nhập. Nếu chuỗi ký tự nhập vào không phải là số nguyên, AutoCAD sẽ báo lỗi và nhắc nhập lại.
  - Nếu người dùng gõ ngay Enter, hàm sẽ trả về NIL.
- (setq inum (getInt "\nNhập một số nguyên: "))
- Để hạn chế dữ liệu nhập (ví dụ không cho nhập ngay Enter, không cho nhập số âm…) gần gọi hàm **InitGet** với các thông số thích hợp trước khi gọi hàm GetInt

Hàm GetReal

- Dùng nhập số thực từ bàn phím

  (getReal [msg]**)**
  - Nếu nhập đúng, kết quả trả về của hàm là số nguyên vừa nhập. Nếu chuỗi ký tự nhập vào không phải là số nguyên, AutoCAD sẽ báo lỗi và nhắc nhập lại.
  - Nếu người dùng gõ ngay Enter, hàm sẽ trả về NIL.
- (setq rnum (getReal "\nNhập một số thực: ")) sẽ hiện thông báo "Nhập một số thực: " trên dòng lệnh, chờ người dùng nhập. Giá trị nhập vào được gán cho biến **rnum**.
- Để hạn chế dữ liệu nhập (ví dụ không cho nhập ngay Enter, không cho nhập số âm…) gần gọi hàm **InitGet** với các thông số thích hợp trước khi gọi hàm GetInt

Hàm GetString

- Nhập chuỗi ký tự từ bàn phím. Hàm InitGet không có tác dụng.

  (getString [cr] [msg]**)**
  - **cr** Nếu có và khác NIL, hàm cho phép nhập cả các dấu trắng (space) trong chuỗi ký tự, cần nhấn Enter để kết thúc nhập.
  - Nếu người dùng gõ ngay Enter, hàm sẽ trả về NIL.
- **(setq sHoten (getString "\nHọ và tên: "))**. Lưu ý rằng tham số **cr** không có, do vậy nếu người dùng nhập Tran (dấu cách) thì sHoten sẽ nhận giá trị "Tran".
- Để nhập đủ cả họ tên, ví dụ "Tran Tien", cần nhập lệnh:
  (setq sHoten (getString T "\nHọ và tên: "))
- Để nhập ký tự đặc biệt như dấu nháy kép " hoặc xổ chéo \ cần thêm vào trước ký tự này một dấu sổ chéo: C:\\AutoCAD\\Alisp\\Test1.lsp sẽ được chuỗi ký tự: C:\AutoCAD\Alisp\Test1.lsp
- Một số ký tự đặc biệt: **** (ký tự \ -xổ chéo); **\n** (dòng mới); **\r** (return); **\"** (nháy kép); **\t** (tab); **\e** (escape); **\nnn** (hiện ký tự mã nnn cơ số 8)

Hàm GetPoint

- Dùng nhập điểm từ bàn phím (gõ tọa độ của nó) hoặc kích chuột chọn điểm trên màn hình AutoCAD.

  (getPoint [pt] [msg]**)**
  - pt điểm tham chiếu, nếu có trên màn hình sẽ xuất hiện đường nối tạm thời từ điểm này đến con trỏ chuột.
  - Hàm trả về điểm dạng danh sách (list) gồm các phần tử là các tọa độ của điểm đã nhập.
  - Các điểm này thường dùng để tạo các đối tượng AutoCAD, chẳng hạn thông qua hàm command như ví dụ sau:
- **(setq pt1 (getPoint "\nNhập điểm đầu: "))** ; nhập điểm đầu
  (setq pt2 (getPoint pt1 "Nhập điểm thứ 2: ")) ; nhập điểm thứ 2
  (command "._Circle" "2P" pt1 pt2) ; vẽ vòng tròn qua 2 điểm đã nhập

Hàm GetDist

- Dùng nhập khoảng cách từ bàn phím (gõ giá trị) hoặc kích chuột chọn 1 hoặc 2 điểm trên màn hình AutoCAD.

  (getDist [pt] [msg]**)**
  - pt điểm tham chiếu, nếu có trên màn hình sẽ xuất hiện đường nối tạm thời từ điểm này đến con trỏ chuột.
  - msg câu nhắc hiện trên dòng lệnh AutoCAD

  Hàm trả về NIL nếu nhấn ngay Enter số thực thể hiện khoảng cách 2 điểm.

  Khoảng cách được nhập theo các cách sau:
  - Nhập giá trị từ bàn phím
  - Nhập 2 điểm trên màn hình CAD
  - Nhập 1 điểm khi điểm tham chiếu **pt** có mặt trong hàm.

Hàm GetAngle

- Dùng nhập góc. Giá trị trả về là góc đã nhập, tính bằng radians. Nếu nhấn ngay Enter mà không nhập gì, hàm trả về NIL.

  (getAngle [pt] [msg]**)**
  - pt điểm tham chiếu.
  - msg câu nhắc hiện trên dòng lệnh AutoCAD

  Góc được nhập theo các cách sau:
  - Nhập giá trị góc từ bàn phím, đơn vị mặc định theo thiết lập của lệnh Unit, thường là độ.
  - Nhập 2 điểm trên màn hình CAD. Giá trị trả về là góc tạo bởi đường nối 2 điểm này và trục X.
  - Nhập 1 điểm khi điểm tham chiếu **pt** có mặt trong hàm. Giá trị trả về là góc tạo bởi đường nối điểm này với điểm tham chiếu và trục X.

  Lưu ý giá trị trả về luôn tính bằng radians **từ 0 đến < 2**π

Hàm GetKWord

- Dùng để nhập từ khóa (keyword). Giá trị trả về là chuỗi ký tự ứng với từ khóa đã được định nghĩa trước bởi hàm InitGet. Từ khóa được nhập đầy đủ hoặc các ký tự viết tắt cho từ khóa tương ứng. Nếu nhập sai AutoCAD sẽ báo lỗi và yêu cầu nhập lại.

  (getKWord [msg]**)**

  - msg câu nhắc hiện trên dòng lệnh AutoCAD.

  Lưu ý hàm không phân biệt nhập chữ hoa hay thường.

```
(InitGet 1 "Tiep" "Khong")      ; định nghĩa từ khóa T (Tiep) và K (Khong).
                                ; Bitcode 1 không cho phép nhấn ngay Enter
                                ; mà không nhập gì.
(setq ans (getkword "Co tiep tuc khong (Co/Khong) ?"))
                ; nhắc người dùng nhập từ khóa. Nếu nhập C hoặc cO biến ans sẽ
                ; được gán giá trị "Co", còn nếu nhập K hoặc kHonG, biến ans được
                ; gán giá trị "Khong". Nếu nhập sai, AutoCAD sẽ nhắc nhập lại.
```


Hàm InitGet

- Dùng trước hàm GetX để hạn chế dữ liệu nhập.

  (InitGet [bit] [str]**)**

  - bit số nguyên 8-bit, được phân tích dưới dạng cơ số 2 thành các bit, có giá trị 0 hoặc 1, tương ứng với việc cho phép hoặc không cho phép nhập một loại dữ liệu nào đó.
  - str chuỗi ký tự thể hiện keyword.
- Ý nghĩa các bit khi bằng 1
  - bit0 không cho phép nhập ngay Enter (NIL)
  - bit1 không cho phép nhập giá trị 0 (Zero)
  - bit2 không cho phép nhập số âm
  - bit3 cho phép nhập điểm nằm ngoài vùng LIMIT
- Nếu muốn cấm nhập nhiều loại dữ liệu thì cần đặt các bit tương ứng với giá trị phù hợp

Hàm InitGet (2)

- Ví dụ cấm nhập nhiều loại dữ liệu:

```
(initGet 3)     ; 3 = 1 + 2 => bit0 và bit1 cùng có giá trị 1 nên
                ; hàm này sẽ cấm nhập NIL và 0
(initGet 6)     ; 6 = 0 + 2 + 4 => bit0 có giá trị 0, bit1 và bit2 cùng có giá trị 1
                ; hàm này sẽ cấm nhập số âm và 0,
                ; nhưng cho phép nhấn ngay Enter
```

- Chuỗi str thể hiện khóa được viết theo quy tắc:
  - Các từ khóa phân cách nhau bởi 1 hoặc nhiều dấu trắng
  - Cụm từ viết tắt trong từ khóa là 1 cụm các chữ viết hoa liền nhau ở bất kỳ đâu trong **str**. Ví dụ: "LType LIne eXit toP"
  - Khi chuỗi ký tự **str** viết hoa, cụm từ viết tắt được viết ngay sau từ khóa, phân cách với từ khóa bằng dấu phẩy. Trường hợp này từ khóa phải chứa chữ cái đầu của chuỗi ký tự **str**. Ví dụ "LTYPE,LT LINE,LI" là hợp lệ, nhưng "EXIT,X" hoặc "TOP,P" là không hợp lệ.

Hàm InitGet (3)

- Lưu ý rằng từ khóa keyword không chỉ dùng cho hàm GetKWord, mà còn dùng cho các hàm GetX khác. Ví dụ:

```
(initGet 4 "Pi Two-pi thRee-pi")  ; không cho phép nhập số âm và
                                  ; khai báo 3 từ khóa
(setq goc (getReal "Nhập góc Pi/Two-pi/thRee-pi <Pi>"))  ; nhập số thực

; xử lý số liệu do người dùng nhập
(if (= goc "Pi") (setq goc 3.14159))          ; nếu nhập từ khóa P hoặc Pi
(if (= goc "Two-pi") (setq goc 6.28318))      ; nếu nhập từ khóa T (Two-pi)
(if (= goc "thRee-pi") (setq goc 9.42477))    ; nếu nhập từ khóa R (thRee-pi)
(if (= goc NIL) (setq goc 3.14159))           ; nếu nhấn ngay Enter (không nhập
                                              ; gì, lấy giá trị mặc định goc = pi)
```

5.2. Chuyển đổi kiểu dữ liệu

- Chuyển đổi kiểu dữ liệu là nhu cầu không thể thiếu trong lập trình. Ví dụ: để hiển thị kết quả thường phải sử dụng kiểu String, ngược lại dữ liệu nhập từ hộp thoại DCL thường ở dạng String, do đó cần chuyển về dạng thích hợp để có thể tính toán, xử lý...
- AutoLISP cung cấp một loạt các hàm dùng cho mục đích này. Cú pháp chung thường có dạng:

  StoD trong đó S là kiểu dữ liệu nguồn, còn D là kiểu cần chuyển đổi tới.
- Dưới đây là một số hàm chuyển kiểu dữ liệu hay dùng.

Hàm atof

- Chuyển đổi chuỗi ký tự dạng số str sang số thực.

  (atof str**)**

  - str chuỗi ký tự
  - Nếu str không phải chuỗi ký tự, autoLISP sẽ báo lỗi. Nếu chuỗi str **không phải là chuỗi số, hàm vẫn thực hiện**. Hàm trả về số thực 0.0 nếu chuỗi str không hợp lệ ngay từ chữ cái đầu tiên, còn không hàm sẽ trả về số tương ứng với các chữ số hợp lệ.
- Ví dụ
  - (atof "97.1") trả về số thực 97.1
  - (atof "97") trả về số thực 97.0
  - (atof "a78") trả về số thực 0.0
  - (atof "123.4a15") trả về số thực 123.4

Hàm atoi

- Chuyển đổi chuỗi ký tự dạng số str sang số nguyên. Lưu ý hàm trả về số nguyên 32bit, nên nếu chuỗi tương ứng với số nguyên vượt khỏi giới hạn 32bit, hàm sẽ trả về giá trị không đúng.

  (atoi str**)**
  - str chuỗi ký tự
  - Nếu str không phải chuỗi ký tự, autoLISP sẽ báo lỗi. Nếu chuỗi str **không hợp lệ, hàm vẫn thực hiện** tương tự như **atof**
- Ví dụ
  - (atoi "97") trả về số nguyên 97
  - (atoi "97.91") cũng trả về số nguyên 97
  - (atoi "097") trả về số nguyên 97
  - (atoi "a97") trả về số nguyên 0
  - **(atoi "123456767889990") trả về số nguyên 2147483647**

Hàm distof

- Chuyển đổi chuỗi ký tự dạng số đo khoảng cách sang số thực

  (distof str [mode]**)**
  - str chuỗi ký tự
  - mode kiểu thể hiện khoảng cách ở chuỗi ký tự (xem hàm **rtos**)
- Nếu chuỗi hợp lệ theo mode đã cho hàm trả về số thực, còn không hàm sẽ trả về NIL.
- Ví dụ
  - (distof "17.5" 2) trả về số thực 17.5
  - (distof "1.75E+01" 1) cũng trả về số thực 17.5
  - (distof "1'-5.5\"" 3) trả về số thực 17.5 (1 ft 5.5 in = 17.5 in)
  - **(distof "1'-5.5\"" 2) trả về NIL do chuỗi không hợp lệ theo mode 2 (kiểu thập phân)**

Hàm angtof

- Chuyển đổi chuỗi ký tự dạng số đo góc sang số thực thể hiện góc này bằng radian (từ 0 đến < 2π)

  (angtof str [mode]**)**
  - str chuỗi ký tự
  - mode kiểu thể hiện góc trong chuỗi ký tự (xem hàm **angtos**)
- Nếu chuỗi hợp lệ theo mode đã cho hàm trả về số thực, còn không hàm sẽ trả về NIL.
- Ví dụ
  - (angtof "45.0" 0) trả về số thực 0.785398 ($\pi/4$)
  - (angtof "-45.0" 0) trả về số thực 5.49779 ($7\pi/4 = 2\pi - \pi/4$)
  - (angtof "45.0" 3) trả về số thực 1.0177 (45rad, sau khi trừ bớt $2k\pi$)
  - (angtof "45r" 3) trả về số thực 1.0177 (45rad, sau khi trừ bớt $2k\pi$)

Hàm ascci và chr

- Hàm **(ascii** str**)** trả về số nguyên ứng với mã ASCII của chữ cái đầu trong chuỗi ký tự **str**

  (ascii "ABC") trả về số nguyên 65 (mã ascii của A)

- Hàm **(chr** int**)**, ngược với **ascii**, trả về chuỗi ký tự gồm chữ cái có mã ascii tương ứng với tham số **int**

  (chr 65) trả về chuỗi ký tự "A", nhưng

  (chr 12) lại trả về chuỗi "\014", thể hiện số 12 trong hệ đếm cơ số 8.

Hàm itoa

- Hàm **(itoa** int**)** trả về chuỗi ký tự thể hiện số nguyên int trong tham số của hàm. Nói cách khác, hàm này chuyển số liệu từ số nguyên sang chuỗi ký tự.
- Lưu ý số nguyên bị giới hạn trong khuôn khổ 32bit.

  (itoa 123) trả về chuỗi "123"

  (itoa 123456767889900) không thực hiện do tham số vượt 32bits

  (itoa 123.456) không thực hiện do tham số không phải là số nguyên

Hàm rtos

- Chuyển đổi chuỗi ký tự dạng số sang số thực

  (rtos num [mode [precision]]**)**
  - num số cần chuyển đổi
  - mode kiểu thể hiện chuỗi ký tự số
  - precision độ chính xác (số chữ số sau dấu thập phân)
- Nếu mode và precision không có trong hàm, chúng được lấy theo mặc định, xác lập từ biến LUNITS và LUPREC của AutoCAD.
- Các mode (setq num 17.5)

1	kiểu khoa học	(rtos num 1 4) trả về chuỗi "1.7500E+01"
2	kiểu thập phân	(rtos num 2 4) trả về chuỗi "17.5000"
3	kỹ thuật (Anh)	(rtos num 3 4) trả về chuỗi "1'-5.5000\"" (1'-5.5000")
4	kiến trúc (Anh)	(rtos num 4) trả về chuỗi "1'-5 1/2\"" (1'-5 1/2")
5	kiểu phân số	(rtos num 5) trả về chuỗi "17 1/2"

Hàm angtos

- Chuyển đổi chuỗi ký tự dạng số đo góc sang số thực thể hiện góc này bằng radian (từ 0 đến < 2π)

  (angtos ang [mode [precision]]**)**
  - ang góc cần chuyển đổi, đo bằng radian
  - mode kiểu thể hiện chuỗi ký tự số
  - precision độ chính xác (số chữ số sau dấu thập phân)
- Nếu mode và precision không có trong hàm, chúng được lấy theo mặc định, xác lập từ biến AUNITS và AUPREC của AutoCAD.
- Các mode (ví dụ với ang = 0.785398 (tức là π/4))

0	kiểu độ thập phân	(angtos ang 0 4) trả về chuỗi "45.0000"
1	XdY'Z"	(angtos ang 1) trả về chuỗi "45d0'0\""
2	Gradian (1v=100g)	(angtos ang 2 4) trả về chuỗi "50.0000g"
3	Radian	(angtos ang 3 4) trả về chuỗi "0.7854r"
4	Hàng hải	(angtos ang 4) trả về chuỗi "N 45d0'0\" E"

5.3. Xuất dữ liệu

- Có thể xuất dữ liệu ra màn hình qua 2 hàm thông báo **prompt** và **alert** ở mục 5.1.1, nhưng dữ liệu cần phải chuyển đổi sang dạng chuỗi ký tự. Giá trị trả về của các hàm này là NIL.
- Các hàm sau đây ghi trực tiếp dữ liệu lên màn hình văn bản của AutoCAD hoặc file dữ liệu. Dữ liệu có thể thuộc các loại khác nhau, cũng có thể là kết quả của 1 biểu thức. Giá trị trả về của hàm cũng khác nhau.
- (**princ** expr [file]) xuất dữ liệu ra màn hình hoặc file (không kèm theo dấu nháy kép đối với chuỗi ký tự)
- (**prin1** expr [file]) xuất dữ liệu ra màn hình hoặc file (kèm cả dấu nháy kép đối với chuỗi ký tự)
- (**print** expr [file]) xuất dữ liệu ra màn hình hoặc file (gồm cả dấu nháy kép đối với chuỗi ký tự), nhưng bắt đầu ở dòng mới và có thêm dấu cách sau DL.

Các hàm prinX

- Các hàm prinX chỉ được phép duy nhất 1 tham số expr, còn tham số file là con trỏ file, nhận được từ hàm open.
- Giá trị trả về của các hàm này là giá trị của tham số expr.
- Ví dụ, chuỗi str có nội dung: "The "allowable" tolerance", các hàm này đều sẽ trả về chuỗi "The \"allowable\" tolerance", nhưng nội dung in ra có khác nhau:

(**princ** str)	in ra:	**The "allowable" tolerance**
(**prin1** str)	in ra:	**"The \"allowable\" tolerance"**
(**print** str)	in ra:	<xuống dòng mới> **"The \"allowable\" tolerance"** <dấucách>
(**princ** (+ 2.5 2.4))	in ra số 4.9 và trả về giá trị số thực 4.9	
(**print** (+ 2.5 2.4))	<xuống dòng mới>, in ra số 4.9, <dấu cách> và trả về giá trị số thực 4.9	

Write-char và Write-line

- Ngoài ra còn sử dụng 2 hàm sau để ghi dữ liệu ra màn hình hoặc vào file.
- (**write-char** int [file]) ghi ký tự có mã ascii bằng **int**. Ví dụ 2 lệnh:

  (write-char 65)
  (write-char 66)

  sẽ in ra màn hình 2 ký tự AB (viết liền nhau trên cùng 1 dòng).
- (**write-line** str [file]) ghi chuỗi ký tự **str** (không kèm theo dấu nháy kép, và sau đó xuống dòng mới. Ví dụ:

  (write-line "Abc")
  (write-line "Def")

  sẽ in ra màn hình 2 dòng Abc và Def.
- Như vậy (**write-line** str [file]) sẽ tương đương với việc thực hiện liên tiếp 2 hàm (**princ** str) và (**print**)

Rút lui im lặng !

- Khi gọi hàm AutoLISP, kết quả biểu thức cuối cùng của hàm được in ra màn hình. Để tránh việc này, thường thêm (princ) ở cuối hàm. Khi đó, hàm sẽ trả về giá trị NIL và cách thức này trong AutoLISP được gọi là "Rút lui im lặng !".
- Ví dụ:

```
(defun vedoanthang( / p1 p2)
  (setq  p1 (getPoint "\nNhập điểm đầu: ")
         p2 (getPoint p1 "\nNhập điểm tiếp: "))
  (command ".line" p1 p2 "")
  (princ)
  )
```


5.4. Làm việc với file

- **File trong AutoLISP thuộc dạng file văn bản, truy cập tuần tự.**
- **Các hàm làm việc với file:**
  - (**open** filename mode) Mở file
  - (**close** file) Đóng file
  - (**findfile** filename) Kiểm tra xem file hoặc folder tồn tại hay chưa
  - (**getfiled** tit def ext flg) Nhập tên file qua hộp thoại file tiêu chuẩn của AutoCAD
- **Ghi dữ liệu vào file** thông qua các hàm **prinX** hoặc **write-char** và **write-line** đã nói ở trên.
- **Đọc dữ liệu từ file** qua các hàm **read-char** và **read-line**

Open

- Hàm (**open** filename mode) dùng để mở file, chuẩn bị cho việc lưu hoặc đọc dữ liệu.
  - Filename chuỗi ký tự, thể hiện tên file cần mở. Nếu không kèm đường dẫn, file được tìm trong folder khởi động của AutoCAD
  - mode chuỗi ký tự thể hiện mục đích mở file, không phân biệt chữ hoa hay chữ thường, gồm các lựa chọn:
    - **"r"** mở file để đọc dữ liệu. File phải tồn tại.
    - **"w"** mở file để ghi dữ liệu. Nếu file chưa tồn tại, nó sẽ được tạo ra, còn nếu đã tồn tại, dữ liệu cũ sẽ bị xóa để ghi mới.
    - **"a"** mở file để ghi thêm dữ liệu. Nếu file chưa tồn tại, nó sẽ được tạo ra, còn nếu đã tồn tại, dữ liệu mới sẽ được ghi tiếp vào sau dữ liệu đã có.
- Nếu thực hiện thành công, hàm trả về con trỏ file để sử dụng với các hàm đọc/ghi dữ liệu. Nếu không thành công, hàm trả về NIL.

Close

- Hàm (**close** file) dùng để đóng file đã mở. Thông số file là con trỏ file, nhận được từ hàm **open.**
- Ví dụ

```
(setq tfile "vidu6-14.txt")      ; Khai báo tên file
(setq ctfile (open tfile "r")    ; Mở file để đọc, nếu thành công thì
                                 ; con trỏ file được gán cho biến ctfile
(setq  ct  0)                    ; Khởi tạo bộ đếm
(while (read-line ctfile)        ; Đọc từng dòng dữ liệu trong file
       (setq ct (1+ ct)          ; mỗi lần đếm tăng thêm 1 cho đến
       )                         ; khi hết (hàm read-line trả về NIL)
(close ctfile)                   ; Đóng file
(princ "File chứa ") (princ ct) (princ " dòng dữ liệu")
(princ)                          ; Thông báo kết quả và rút lui im lặng
```


Findfile

- Hàm (**findfile** filename) dùng để kiểm tra xem file hay folder có tên chỉ định có tồn tại không.
- Thông số filename là tên file/folder cần kiểm tra. Nếu ghi không kèm đường dẫn, hàm sẽ kiểm tra trong các đường dẫn khai báo trong mục Reference của AutoCAD, ngược lại hàm chỉ kiểm tra trong folder đã chỉ định.
- Nếu tìm thấy hàm sẽ trả về tên đầy đủ của file dưới dạng chuỗi ký tự. Nếu không tìm thấy, hàm trả về NIL.
- **Ví dụ** (setq tfile (findfile "d:/VLisp/vidu7.lsp")) sẽ chỉ tìm file vidu7.lsp trong folder D:\VLisp. Nếu có, biến **tfile** sẽ được gán giá trị "d:\\vlisp\\vidu7.lsp", còn nếu không, **tfile** sẽ được gán giá trị NIL.
- Nếu folder hiện thời của AutoCAD có chứa file "abc.lsp", biểu thức (findfile "abc.lsp") sẽ trả về tên đầy đủ của file, chẳng hạn "C:\\Program Files\\AutoCAD 2007\\abc.lsp"

Getfiled

- Hàm (**getfiled** tit def ext flg) dùng nhập tên file qua hộp thoại file tiêu chuẩn của AutoCAD, kết quả trả về tên đầy đủ của file.
  - tit Chuỗi ký tự sẽ hiện trên thanh tiêu đề của hộp thoại File
  - def Chuỗi ký tự thể hiện tên mặc định. Sử dụng chuỗi rỗng "" nếu không muốn tên mặc định nào cả.
  - ext Chuỗi ký tự chỉ phần mở rộng (phần đuôi) của tên file. Nếu gán chuỗi rỗng, nó được coi như chuỗi "*" (phần đuôi bất kỳ)
  - flg Số nguyên, điều khiển việc thể hiện hộp thoại theo cách thức nhất định. Số nguyên được phân tích thành các bit, giá trị các bit này điều khiển hoạt động của hộp thoại (tham khảo InitGet)
    - **bit0** nếu là 1 sẽ hiện hộp thoại dạng **SaveAs**, còn nếu là 0 – hộp thoại **Open**
    - **bit1** không sử dụng
    - **bit2** nếu là 1 tên file có thể không có phần đuôi hoặc phần đuôi khác với mặc định khai báo trong thông số ext, còn nếu là 0 – tên file sẽ được gắn phần đuôi mặc định

Ghi dữ liệu vào file

- Sử dụng các hàm **prinX** hoặc **write-char** và **write-line**. Ví dụ sau đây ghi tọa độ các điểm chọn trên màn hình vào file.

```
(setq  tfile  "d:/vlisp/p2file.txt")     ; khai báo tên file
(setq  fil (open tfile  "w"))            ; mở file để ghi dữ liệu
(if  fil                                 ; nếu mở file thành công
   (progn
      (princ ";;;; Tọa độ các điểm ;;;;", fil)     ;ghi dòng đầu vào file
      (setq  p  (getPoint "\Nhập điểm đầu: "))  ;nhập điểm đầu
      (while  p
         (print  p  fil)                 ;ghi tọa độ điểm đã nhập vào file
         (setq  p  (getPoint "\Nhập điểm tiếp theo: "))  ;nhập điểm tiếp...
         )                               ;kết thúc vòng lặp while (nhấn Enter)
      (close  fil)                       ;đóng file
      )  ;kết thúc khối progn
   (prompt  "Không mở được file...")     ;thông báo nếu không mở được file
)     ;kết thúc khối If
```


Ghi dữ liệu vào file (2)

- Khi chạy đoạn chương trình này, do vòng lặp while nên trên dòng lệnh AutoCAD sẽ xuất hiện các lời nhắc yêu cầu nhập điểm. Kích chuột hoặc gõ tọa độ điểm để nhập, nhấn Enter (không nhập gì) để kết thúc vòng lặp.
- Nếu thực hiện thành công và nhập vào 4 điểm, dữ liệu lưu vào file sẽ có dạng sau:

```
;;;; Tọa độ các điểm ;;;;
(1.0  1.0  0.0)
(2.5  1.3  0.0)
(4.0  2.4  0.0)
(5.0  3.8  0.0)
```

- Lưu ý rằng dấu ngoặc () trong file dữ liệu là do các điểm được quản lý dưới dạng danh sách list gồm 3 phần tử.

Đọc dữ liệu từ file

- Sử dụng các hàm **read-char** và **read-line**
- Hàm (**read-char** ctfile) đọc từng ký tự trong file và **trả về số nguyên là mã ascii** của ký tự vừa đọc. Nếu gọi tiếp hàm một lần nữa, nó sẽ đọc ký tự tiếp theo. Khi hết dòng, hàm trả về số 10 và con trỏ chuyển xuống dòng kế tiếp. Khi hết file, hàm trả về NIL.
- Hàm (**read-line** ctfile) đọc dòng văn bản từ vị trí con trỏ và **trả về chuỗi ký tự** đã đọc được, không kèm theo ký tự xuống dòng. Con trỏ rời xuống dòng kế tiếp. Khi hết file (không còn gì để đọc), hàm trả về NIL.
- Thông số ctfile là con trỏ file, nhận được từ hàm **open.**

Đọc dữ liệu từ file (2)

- Giả sử file Rtest.txt chứa dữ liệu sau:

```
AB
Dong 2  2XY
Dong 3  3TWZ
<xuống dòng>
```

- Đoạn chương trình sau minh họa các lệnh read-char và read-line

```
(setq tfile "Rtest.txt")            ;gán tên file
(if (setq fil (open tfile "r"))     ;mở file để đọc và nếu mở được
  (progn
    (read-char fil)     ;trả về 65, ứng với ký tự A trên dòng 1
    (read-char fil)     ;trả về 66, ứng với ký tự B trên dòng 1
    (read-char fil)     ;trả về 10, ứng với ký tự hết dòng
    (read-char fil)     ;trả về 68, ứng với ký tự D trên dòng 2
    (read-line fil)     ;trả về "ong 2  2XY" (cả dòng 2, nhưng thiếu D)
    (read-line fil)     ;trả về "Dong 3  3TWZ"  (đầy đủ cả dòng)
    (read-line fil)     ;trả về NIL (chỉ có ký tự dòng mới)
    (read-line fil)     ; trả về NIL (chẳng còn gì để đọc)
    (close fil)         ;đóng file
  ) )
```


Đọc dữ liệu từ file (3a)

- Hàm người dùng sau đọc dữ liệu từ file và chuyển thành danh sách.

```
(defun file2list (fname / f  ln  lst  c1 kq)    ;khai báo hàm
   (setq f (open fname "r"))                    ;mở file để đọc
   (if (not f) (progn
      (alert "Không mở được file !")
      (exit) )        ;nếu không mở được -> thoát
   (setq  kq '())     ;còn được thì tiếp tục, bắt đầu bằng tạo danh sách kq rỗng
   (while (setq  ln  (read-line f))   ;đọc dòng dữ liệu trong file
      (setq  c1 (substr ln 1 1))      ;lấy ký tự đầu của dòng vừa đọc
      (if  (/=  ";"  c1) (progn       ;kiểm tra xem có phải là dòng chú thích
         (if (/= "("  c1) (setq ln (strcat "(" ln)))  ;thêm ( ) nếu chưa có
         (if (/= ")" (substr ln (strlen ln 1))  (setq ln (strcat  ln  ")" )))
         (setq lst (read  ln))        ;đọc dòng dữ liệu và trả về danh sách lst
         (setq kq (append kq (list lst)))     ;thêm danh sách lst vào kq
      )))          ; kết thúc progn, if  và while (khi hết file)
   (close f)       ;đóng file
   kq )            ; trả kết quả về và kết thúc hàm
```


Đọc dữ liệu từ file (3b)

Kết quả gọi hàm **file2list** với file dữ liệu "d://vlisp/p2file.txt"

```
;;;; Tọa độ các điểm ;;;;
(1.0  1.0  0.0)
(2.5  1.3  0.0)
(4.0  2.4  0.0)
(5.0  3.8  0.0)
```

thực hiện qua câu lệnh: (file2list "d://vlisp/p2file.txt") sẽ có kết quả là danh sách liên kết:

```
( (1.0  1.0  0.0)  (2.5  1.3  0.0)  (4.0  2.4  0.0)  (5.0  3.8  0.0) )
```

Dòng đầu bắt đầu bằng dấu ; nên được coi là chú thích và bị bỏ qua, không đưa vào danh sách kết quả.

5.5. Tạo các đối tượng AutoCAD

- Các đối tượng AutoCAD bao gồm các đối tượng đồ họa như đoạn thẳng, cung tròn,... và các đối tượng khác như khối (block), lớp (layer)...
- Tạo và sửa đối tượng AutoCAD có thể thực hiện qua 2 cách:
  - Sử dụng hàm **command** để gửi các lệnh và thông số tương ứng để AutoCAD thực hiện
  - Trực tiếp can thiệp vào **CSDL** của AutoCAD
- Sử dụng hàm command là một cách hữu hiệu và đơn giản, nhưng phải nắm vững các lệnh của AutoCAD, còn việc can thiệp vào CSDL thì khó hơn, cần hiểu rõ cách thức quản lý các đối tượng trong AutoCAD (hiểu rõ mã đối tượng và cách xử lý).
- Cũng có thể kết hợp cả 2 cách thức trên

5.5.1. Hàm Command

- Cú pháp chung của hàm command:

  (command [arg...])

  trong đó arg là các tham số của hàm, bao gồm:
  - Tên lệnh AutoCAD
  - Các lựa chọn con của lệnh tương ứng
  - Các dữ liệu cần nhập theo thứ tự như khi vẽ bằng lệnh AutoCAD
- Các tham số của hàm command có thể thuộc loại dữ liệu chuỗi ký tự, số thực, số nguyên, điểm, nhóm chọn. Chuỗi ký tự trắng "" được hiểu như việc nhấn Enter.
- Hàm command luôn trả về giá trị NIL.
- Lưu ý AutoCAD có phiên bản cho nhiều ngôn ngữ khác nhau, cũng như cho phép tự định nghĩa lại lệnh nên để tránh nhầm lẫn nên thêm vào trước tên lệnh cặp dấu _. hoặc ._

Ví dụ

- Lưu ý: cần nắm rõ các lệnh AutoCAD ở phương án dòng lệnh. Ví dụ để vẽ vòng tròn cần nắm rõ:
  - Tên lệnh: Circle
  - Các lời nhắc tiếp theo: Specify center point for circle or [3P/2P/TTR]
  - Các dữ liệu cần nhập: tương ứng với phương án đã chọn, sẽ được nhắc tiếp sau khi chọn phương án. Chẳng hạn nếu chọn phương án mặc định (nhập tâm vòng tròn) AutoCAD sẽ nhắc nhập tiếp bán kính, còn nếu chọn phương án 2P, AutoCAD sẽ nhắc nhập 2 điểm mút của 1 đường kính...
- Một vài ví dụ minh họa
  - Ví dụ 1: Vẽ vòng tròn biết tâm và bán kính

```
(setq  cen  '(1  2  0))          ;khai báo tâm vòng tròn
(setq  rad  10.0)                ;khai báo bán kính
(command "_.Circle" cen rad)     ;vẽ vòng tròn
```


Ví dụ (2)

- Ví dụ 2: Vẽ vòng tròn qua 2 và qua 3 điểm (nhập trực tiếp). Vẽ các đoạn thẳng nối 3 điểm đã nhập.

```
(setq  p1 (getpoint "Nhập điểm đầu: ")        ;khai báo điểm 1
       p2 (getpoint  p1 "Điểm thứ 2: "))      ;khai báo điểm 2
(command "_.Circle" "_.2P" p1 p2)     ;vẽ vòng tròn qua 2 điểm.
;nhập thêm điểm thứ 3
(setq  p3 (getpoint  "Điểm thứ 3: "))
(command "_.Circle" "3P"  p1  p2  p3) ;vẽ vòng tròn qua 3 điểm
;vẽ các đoạn thẳng nối 3 điểm đã nhập
(command "_.Line"  p1 p2 p3 "")

; lưu ý thêm dấu ký tự trống vào sau điểm cuối cùng của lệnh Line do
; lệnh này vẽ các đoạn liền nhau cho đến khi nhấn Enter để kết thúc.
```

Ví dụ (3)

- Ví dụ 3: Vẽ các đoạn thẳng nối các trung điểm 3 đoạn thẳng ở ví dụ 2

Để vẽ các đoạn thẳng này cần tính trước tọa độ các trung điểm M, N, P của các đoạn trên. Có thể sử dụng các hàm số học và xử lý danh sách để thực hiện việc tính toán.

```
(setq  x (* 0.5 (+ (car p1) (car p2)))
       y (* 0.5 (+ (cadr p1) (cadr p2)))
       m (list x y))          ;điểm giữa M của đoạn p1-p2
(setq  x (* 0.5 (+ (car p2) (car p3)))
       y (* 0.5 (+ (cadr p2) (cadr p3)))
       n (list x y))          ;điểm giữa n của đoạn p2-p3
(setq  x (* 0.5 (+ (car p1) (car p3)))
       y (* 0.5 (+ (cadr p1) (cadr p3)))
       p (list x y))          ;điểm giữa n của đoạn p1-p3
(command "_.Line"  m  n  p "")
```


Các hàm hình học

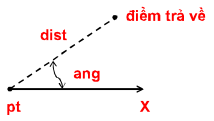
- Việc tính toán tọa độ điểm như ở ví dụ 3 khá phức tạp, chưa kể đến các thao tác khó hơn như quay đối tượng, lấy đối xứng...
- AutoLISP cung cấp các hàm hình học để xác định tọa độ điểm, đo khoảng cách, tính góc...

Polar	- tìm điểm qua tọa độ tương đối với 1 điểm khác
Distance	- xác định khoảng cách giữa 2 điểm
Angle	- xác định góc đường nối hai điểm và trục X
Inters	- tìm giao điểm của 2 đoạn thẳng
Osnap	- tìm điểm qua cách truy bắt điểm đặc trưng (Object Snap) như điểm mút, trung điểm, chân đường vuông góc, tâm vòng tròn...

Hàm Polar

- Hàm polar trả về một điểm, xác định thông qua tọa độ cực tương đối với một điểm khác

  (**polar** pt ang dist)
  - **pt** - điểm tham chiếu (đóng vai trò gốc tọa độ tạm thời)
  - **ang** - góc (radian) giữa đường nối điểm tham chiếu với điểm cần tìm so với trục X của hệ toạ độ chuẩn World.
  - **dist** - khoảng cách giữa 2 điểm

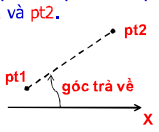


Hàm Distance và Angle

- Hàm **distance** trả về khoảng cách giữa 2 điểm

  (**distance** pt1 pt2)
  - **pt1 và pt2** - các điểm 2D hoặc 3D.
  - Lưu ý rằng khoảng cách giữa 2 điểm có thể tính bằng căn bậc 2 của tổng bình phương các tọa độ, nhưng rõ ràng là rắc rối hơn nhiều

- Hàm (**angle** pt1 pt2) trả về góc tạo bởi trục X của hệ tọa độ hiện hành và đường nối 2 điểm pt1 và pt2.

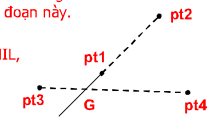


Hàm Inters

- Hàm **inters** trả về giao điểm của 2 đoạn thẳng

  (**inters** pt1 pt2 pt3 pt4 [onseg])
  - **pt1 và pt2** - các điểm nằm trên đoạn thẳng thứ nhất.
  - **pt3 và pt4** - các điểm nằm trên đoạn thẳng thứ hai.
  - **onseg** - tham số tùy chọn.
- Nếu bỏ qua tham số onseg hoặc onseg khác NIL hàm sẽ chỉ tìm giao điểm nằm giữa các điểm đã cho. Nếu chúng không có, hàm sẽ trả về NIL. Còn nếu onseg là NIL, hàm sẽ tìm cả giao điểm trên cả phần kéo dài của các đoạn này.
- Trên hình vẽ
  (inters pt1 pt2 pt3 pt4) sẽ trả về NIL,

  (inters pt1 pt2 pt3 pt4 NIL)
  sẽ trả về điểm G.



Hàm Osnap

- Hàm **Osnap** trả về điểm 3D theo cách truy bắt điểm đặc trưng trên đối tượng như ở chế độ Object Snap của AutoCAD

  (**osnap** pt mode)
  - **pt** - điểm, nơi sẽ truy bắt điểm đặc trưng.
  - **Mode** - chuỗi ký tự thể hiện đặc trưng Osnap, chẳng hạn mid – điểm giữa, cen – tâm, end – điểm cuối...
- Sự thành công của việc truy bắt điểm theo cách này không những phụ thuộc vào tọa độ điểm pt mà còn phụ thuộc vào độ lớn của vùng truy bắt, xác lập bởi biến hệ thống Aperture của AutoCAD. Do đó, hàm này ít được sử dụng trong lập trình thiết kế tự động do chương trình chạy không ổn định.

Các biến hệ thống trong AutoCAD

- Các biến hệ thống trong AutoCAD như Osmode (chế độ truy bắt điểm đặc trưng), Orthomode (chế độ vẽ ngang/dọc)... có thể gây ảnh hưởng đến hàm command do việc các điểm nhập vào có thể bị rời đến vị trí khác. Do đó, trước khi chạy chương trình, thường các biến hệ thống nhạy cảm này được cài đặt về 0, sau đó, khi chương trình chạy xong, chúng phải được khôi phục lại giá trị ban đầu.
- AutoLISP cung cấp các hàm tương ứng để thực hiện việc này:
  - **Getvar** - lấy giá trị hiện thời của biến hệ thống
  - **Setvar** - đặt giá trị cho biến hệ thống

Hàm Getvar và Setvar

- Hàm **getvar** lấy giá trị hiện thời của biến hệ thống với cú pháp:

  (**getvar** varname)

  trong đó varname là chuỗi ký tự hoặc ký hiệu thể hiện tên biến hệ thống cần truy cập. Nếu tham số này không đúng, hàm trả về NIL.
  Xem *Command Reference / System Variable* để rõ hơn về các biến này.
- Hàm **setvar** đặt giá trị cho biến hệ thống, dùng với cú pháp sau:

  (**setvar** varname value)
  - varname - chuỗi ký tự hoặc ký hiệu thể hiện tên biến hệ thống
  - Value - giá trị cần đặt cho biến

  Cần lưu ý là một số biến hệ thống chỉ cho phép đọc (read only), do đó không thể đặt lại giá trị cho các biến hệ thống này.

Lưu ý thêm về hàm command

- Hàm **command** không tham số tương đương với nhấn Escape để thoát lệnh trong AutoCAD.
- Hàm **command** còn gửi **giá trị** đến AutoCAD để xử lý. Có thể sử dụng phương thức này để gửi các điểm tính trước hoặc lưu từ file để tạo các đối tượng (ví dụ vẽ đường gấp khúc mà không cần biết trước số lượng các điểm này)
- Ví dụ các điểm được lưu dưới dạng danh sách **plst**, cần vẽ đường Spline kín nối các điểm này.

```
(setq  plst '((2 1) (5 3) (6 9) (12 17) (19 29) (24 1) (6 78)) )
(command "_.Spline" ) ; gọi lệnh Spline để vẽ đường cong nối các điểm trên
(foreach  pt  plst  (command  pt))  ; gửi các điểm trong danh sách cho Spline
(command "_c")          ; đóng kín Spline, hoàn tất lệnh
(princ)                 ; rút lui im lặng
```


5.5.2. CSDL AutoCAD

- AutoCAD quản lý các đối tượng dưới dạng CSDL gồm các bản ghi dạng danh sách liên kết và dot-pair. Đối tượng mới có thể được tạo ra bằng cách thêm trực tiếp các bản ghi mới vào CSDL này.
- Các đối tượng trong AutoCAD thường được truy cập qua tên của chúng. Tên này được AutoCAD đặt riêng, tự thay đổi mỗi khi mở bản vẽ, do vậy thường được gán cho biến nào đó và thực hiện các thao tác qua biến này.
- Các hàm truy cập đối tượng:
  - **(entlast)** - lấy tên đối tượng vẽ sau cùng
  - **(entnext [ename])** - tên đối tượng liền sau đối tượng ename. Nếu không có ename, hàm trả về tên đối tượng đầu.
  - **(entget ename)** – lấy mã đối tượng có tên ename
  - **(entmake [elist])** – tạo đối tượng mới từ mã elist (danh sách)

Ví dụ

Ví dụ 1: Vẽ đoạn thẳng nối 2 điểm (1 2) và (6 5) trong mặt phẳng XY sau đó chạy đoạn chương trình sau:

```
(setq ename (entlast))  ;lấy tên đối tượng vừa vẽ, gán cho biến ename
(setq ecode (entget ename)) ;lấy mã đối tượng gán cho biến ecode
```

Kết quả:

- Biến ename có giá trị <Entity name: 7ef4fea0>
- Biến ecode là danh sách:

```
((-1 . <Entity name: 7ef4fea0>)  (0 . "LINE")
   (330 . <<Entity name: 1bbd0c8>)  (5 . "6A")  (100 . "AcDbEntity")
   (67 . 0)  (410 . "Model")  (8 . "0")  (100 . "AcDbLine")
   (10  1.0  2.0  0.0)  (11  6.0  5.0  0.0)  (210  0.0  0.0  0.0  1.0))
```

Danh sách này thể hiện mã đối tượng của đoạn thẳng được quản lý trong CSDL của AutoCAD. Để thao tác với dữ liệu này cần nắm được ý nghĩa các thành phần. Tham khảo *DXF Reference trong Help của AutoCAD*.

Ví dụ (2)

Ví dụ 2: Tạo lớp MYLAYER, vẽ đối tượng vòng tròn có tâm (5.0 7.0 0.0) và bán kính 1.0 trên lớp vừa tạo.

```
(setq elist '((0 . "CIRCLE") (8 . "MYLAYER") (10 5.0 7.0 0.0) (40 . 1.0)))
(entmake elist)
```

Vòng tròn đã được vẽ và nếu chạy tiếp các biểu thức ở ví dụ 1, kết quả nhận được như sau:

- Biến ename có giá trị <Entity name: 7ef4feb8>
- Biến ecode là danh sách:

```
((-1 . <Entity name: 7ef4feb8>)  (0 . "CIRCLE")
   (330 . <<Entity name: 7ef4fcf8>)  (5 . "8F")  (100 . "AcDbEntity")
   (67 . 0)  (410 . "Model")  (8 . "MYLAYER")  (100 . "AcDbCircle")
   (10  5.0  7.0  0.0)  (40 . 1.0)  (210  0.0  0.0  0.0  1.0))
```

Có thể thấy rằng AutoCAD đã tự thêm các mã nhóm chung vào mã của đối tượng vừa tạo như tên đối tượng, nét, màu... theo giá trị mặc định.

5.6. Chỉnh sửa đối tượng AutoCAD

- Các lệnh chỉnh sửa trong AutoCAD nói chung liên quan đến việc chọn đối tượng (nhóm đối tượng) trên bản vẽ.
  - Để thực hiện, AutoCAD đưa lời nhắc "Select Object(s)... ở dòng lệnh.
  - Trong lập trình tự động thiết kế, người dùng cần truyền dữ liệu cho lệnh sửa đổi một cách tự động, không qua thao tác chọn trực tiếp. AutoLISP sử dụng một loại dữ liệu kiểu nhóm chọn, gọi là Selection Set hoặc Pick Set.
- Một số hàm thông dụng xử lý dữ liệu loại này:
  - **ssget** – chọn nhóm đối tượng theo các cách khác nhau
  - **ssadd** – thêm đối tượng vào nhóm chọn
  - **ssdel** – loại đối tượng khỏi nhóm chọn
  - **sslength** – số lượng đối tượng trong nhóm chọn
  - **ssname** – tên đối tượng trong nhóm chọn
  - **ssmemb** – kiểm tra xem đối tượng có thuộc nhóm chọn không

ssGet

- Hàm dùng để chọn nhóm đối tượng như khi thao tác trực tiếp trong môi trường AutoCAD. Cú pháp chung của hàm như sau:

  (ssget [sel-method] [pt1 [pt2]] [pt-list] [filter-list]**)**
  - sel-method – chuỗi ký tự thể hiện cách chọn nhóm đối tượng "C"-CrrosingWindow, "CP"-CrossingPolygon, "F"-Fence, "L"-Last, "P"-Previous, "W"-Window, "WP"-Polygon, "X"-chọn tất cả, kể cả trên các lớp Off, Frozen và nằm ngoài vùng nhìn thấy.
  - pt1, pt2 – điểm xác định vùng sẽ chọn các đối tượng (C hoặc W).
  - pt-list – danh sách các điểm, xác định đường gấp khúc (F) hoặc đa giác (CP,WP) liên quan đến vùng chọn.
  - filter-list – danh sách liên kết thể hiện cách chọn loại trừ, ví dụ **(ssget "X" '((0 . "LINE") (62 . 5)))** sẽ chỉ chọn các đoạn thẳng Line được vẽ màu xanh (mã màu 5).
  - Khi hàm **ssget không có tham số**, AutoCAD sẽ hiện lời nhắc "Select Objects..." yêu cầu người dùng chọn trực tiếp.

ssAdd

- Hàm dùng để thêm đối tượng vào nhóm chọn.

  (ssadd [ename [ss]]**)**
  - ename – tên đối tượng mới cần thêm vào nhóm chọn.
  - ss – tên nhóm chọn sẽ được bổ sung đối tượng chọn mới.
  - **Nếu hàm không có tham số,** nhóm chọn rỗng sẽ được tạo ra.
  - **Nếu chỉ có tham số ename,** sẽ tạo nhóm chọn chỉ gồm 1 phần tử.
- Kết quả trả về của hàm là tên nhóm chọn ss.
- Ví dụ

```
(setq ss1 (ssadd))       ; tạo nhóm chọn rỗng, gán cho biến ss1
(setq ss2 (entlast))     ; tạo nhóm chọn gồm phần tử vừa vẽ, gán cho biến ss2
(ssadd (entlast) ss1)    ; thêm đối tượng vừa vẽ vào nhóm ss1. Khi đó ss1 và
                         ; ss2 là như nhau: cùng chứa phần tử vừa vẽ.
(ssadd (entnext) ss1)    ; thêm đối tượng đầu trong CSDL vào nhóm ss1.
```

ssDel

- Hàm dùng để loại đối tượng khỏi nhóm chọn, tương tự lựa chọn "R" (remove) trong yêu cầu chọn đối tượng của AutoCAD.

  (ssdel ename ss**)**
  - ename – tên đối tượng cần loại khỏi nhóm chọn.
  - ss – tên nhóm chọn.
  - **Nếu ename thuộc nhóm ss,** nhóm chọn ss sẽ được loại bớt phần tử có tên ename. Hàm trả về tên nhóm chọn.
  - **Nếu ename không thuộc nhóm ss,** hàm trả về NIL.
- Ví dụ

```
(setq ss (ssadd (entlast))) ; tạo nhóm ss gồm đối tượng vừa vẽ.
(ssadd (entnext) ss)        ; và bổ sung thêm đối tượng đầu trong CSDL.
(ssdel (entlast) ss)        ; loại đối tượng cuối khỏi nhóm chọn ss
```

ssLength, ssName và ssMemb

- **(sslength ss)** trả về số lượng đối tượng trong nhóm chọn ss.
- **(ssname ss index)** trả về tên đối tượng thứ index trong nhóm chọn ss. Lưu ý các đối tượng trong nhóm được đánh số từ 0.
- **(ssmemb ename ss)** trả về tên đối tượng ename nếu đối tượng này nằm trong nhóm ss, còn nếu không, hàm trả về NIL.
- **Ví dụ áp dụng**: nhập 2 nhóm chọn vào 1 nhóm.

```
(setq ss1 (ssget)) ; yêu cầu người dùng chọn đối tượng, đưa vào nhóm ss1.
(setq ss2 (ssget)) ; yêu cầu người dùng chọn đối tượng, đưa vào nhóm ss2.
(setq num (sslength ss2)) ; lấy số phần tử trong nhóm ss2, gán cho biến num
(setq ind 0)              ; khởi tạo bộ đếm
(repeat num               ; lặp num lần
   (setq ent (ssname ss ind))   ; lấy tên phần tử thứ ind
   (ssadd ent ss1)              ; thêm vào nhóm ss1
   (setq ind (1+ ind))          ; tăng ind thêm 1
   )            ; kết thúc vòng lặp
```


Chỉnh sửa các đối tượng

- Trong thiết kế các thao tác chỉnh sửa đối tượng như sao chép, xóa, quay... không thể thiếu.
- Để thực hiện tự động các thao tác này, có thể sử dụng hàm command gọi các lệnh AutoCAD và các nhóm chọn.
- Cũng có thể truy cập trực tiếp vào CSDL của AutoCAD để chỉnh sửa. Thao tác thay đổi trực tiếp mã đối tượng thực hiện bằng hàm **(entmod** elist**)** và cập nhật lại hình ảnh mới của đối tượng bằng hàm **(entupd** ename**)**.
- Một cách hữu hiệu khác là kết hợp cả 2 cách trên để thực hiện chỉnh sửa đối tượng AutoCAD.
- Các ví dụ minh họa:
- Ví dụ 1: Hàm người dùng ss-rcopy tạo nhóm đối tượng mới quay một góc so với nhóm đối tượng gốc.
- Ví dụ 2: Sửa đối tượng bằng entmod và entupd
- Ví dụ 3: Hàm người dùng CenterLine để thêm đường tâm cho các vòng tròn trên bản vẽ.

Ví dụ 1

```
(defun ss-rcopy (ss pt ang / ss1 ent num ind)        ; khai báo hàm ss-rcopy
   (if (null ss) (setq ss1 NIL)                      ; ss – nhóm chọn
      (progn                                         ; pt – tâm quay
         (setq ss1 (ssadd))                          ; ang – góc quay
         (setq num (sslength ss)    ind 0)
         (while (< ind num)
            (setq ent (ssname ss ind))
            (command "_.Copy" ent "" pt pt)          ; copy tại chỗ
            (command "_.Rotate" (entlast) "" pt ang) ; quay
            (ssadd (entlast) ss1)
            (setq ind (1+ ind))
         )))        ; kết thúc while, progn, if
   ss1   ; kết quả trả về: nhóm đối tượng mới tạo hoặc NIL nếu nhóm chọn rỗng.
   )     ; kết thúc hàm
```


Ví dụ 2

Giả sử đã tạo lớp MYLAYER, vẽ vòng tròn có tâm (5.0 7.0 0.0), bán kính 1.0

```
(setq elist '((0 . "CIRCLE") (8 . "MYLAYER") (10 5.0 7.0 0.0) (40 . 1.0)))
(entmake elist)
```

sau đó muốn đổi bán kính thành 10.0 và màu từ "ByLayer" thành đỏ.

```
(setq  newrad (cons 40  10.0) ; khai báo danh sách liên kết thể hiện bán kính mới
       newclr (cons 62  1))   ; danh sách liên kết thể hiện màu mới (1 = đỏ)
(setq  ent  (entlast))        ; lấy tên đối tượng vừa vẽ (vòng tròn)
(setq  ecode  (entget ent))   ; lấy thông tin mã đối tượng và thay đổi chúng
(if (setq  clr  (assoc  62  ecode))          ; nếu đã đặt màu (có mã 62)
   (setq  ecode  (subst  newclr  clr  ecode))  ; thì thay mới
   (setq ecode (append ecode (list newclr)))  ; nếu chưa thì thêm vào
   )
(entmod ecode)  ; cập nhật dữ liệu
(entupd ent)    ; cập nhật đối tượng
```

Ví dụ 3

Ví dụ này vẽ thêm đường tâm vào các vòng tròn do người dùng chọn

```
(setq  ss  (ssget  '((0 . "CIRCLE"))))  ; yêu cầu người dùng chọn đối tượng, nhưng
(setq  ind 0                            ; chỉ đưa các vòng tròn vào nhóm chọn.
       num (length ss))         ; đếm số phần tử đã chọn
(repeat  num
   (setq  ent (ssname ss ind)
          ecode (entget ent)                ; lấy mã đối tượng
          cen (cdr (assoc 10  ecode))       ; tâm vòng tròn
          rad (cdr (assoc  40  ecode)))     ; bán kính
   (setq  c1 (polar cen 0.0 (* 1.1 rad))    ; vẽ 2 đường tâm, độ dài 1,1 đkính
      c2 (polar cen pi (* 1.1 rad))
      c3 (polar cen (* pi 0.5) (* 1.1 rad))
      c4 (polar cen (* pi 0.5) (* 1.1 rad)))
   (command "_.Line" c1 c2 "")
   (command "_.Line" c3 c4 "")
   (setq  ind (1+  ind))    ; đối tượng tiếp theo
)                           ; kết thúc vòng lặp
```